A bisimulation approach to verification of molecular implementations of formal chemical reaction networks

A Thesis Presented

by

Qing Dong

to

The Graduate School in Partial Fulfillment of the Requirements for the

Degree of

Master of Science

in

Computer Science

Stony Brook University

December 2012

Stony Brook University

The Graduate School

Qing Dong

We, the thesis committee for the above candidate for the
Master of Science degree, hereby recommend
acceptance of this thesis.

Steve Skiena - Thesis Advisor
Distinguished Teaching Professor, Computer Science

I.V. Ramakrishnan - Second Reader
Professor, Computer Science

Himanshu Gupta
Associate Professor, Computer Science

Erik Winfree
Professor, Computer Science,
California Institute of Technology

This thesis is accepted by the Graduate School.

Charles Taber
Interim Dean of the Graduate School

Abstract of the Thesis

A bisimulation approach to verification of molecular implementations of formal chemical reaction networks

by

Qing Dong

Master of Science

in

Computer Science

Stony Brook University

2012

In molecular computing, chemical reaction networks have been proposed as a high-level programming language. For a formal specification represented by a CRN, the implemented system will be more complex and, thus, will be represented by another CRN with additional intermediate states and details. Here we study the problem of how to define and verify the correctness of the implementation. First, the equivalence between a formal CRN and an implementation CRN is defined using a bisimulation approach. Then, an algorithm is constructed for verifying CRN equivalence by looking for a valid interpretation from the implementation CRN to the formal CRN. Finally, the algorithm is implemented, and tested on a variety of practical cases.

Table of Contents

List of Figures

Chapter 1

Introduction

Since the foundation of chemical reaction network (CRN) theory in the 1970s [13][14], it has been one of the most fundamental mathematical models for characterizing the behavior of chemical systems. Furthermore, in the area of molecular computing, CRNs have become a programming language for molecular systems [1][12]. Soleveichik et. al. [15] have shown that stochastic CRNs can emulate Turing machines with an arbitrarily small probability of error. As a programming language, CRNs are powerful, but there is another important question for a programming language, which is "correctness": How do we know that a set of designed molecules and their detailed interactions constitute a correct implementation of the formal chemical reaction network specification that we want? In general, the implemented system will consist of many more species and many more reactions than the formal specification, because a variety of molecular details must be accounted for in the model. In this study, we explored the equivalence between an implementation CRN and a formal CRN, which can be used to define the correctness of an implementation. Computational models that are closely related to CRNs, such as vector addition systems (VASs) and Petri nets, have been extensively explored in the literature. A variety of equivalence notions have been explored for these two models [4][6]. For example, equivalence based on reachability sets for VASs was proven to be undecidable by Hack [4]. The reachability set was defined by Hack

as the set of all the states that can be reached by the system with a specific starting state [4], however, it contains no information about the pathway through which one state gets to another. Studies of strong bisimulation [16] are based on establishing a match for each step in the pathway when comparing two systems. Equivalence based on strong bisimulation was explored by Larroussinie and Schnoebelen [6], who showed that any relation between the simulation preorder and bisimilarity is EXPTIME-hard.

In the current study, we explored the CRN equivalence based on weak bisimulation [16], which tries to establish the correspondence for some steps along the pathway while ignoring those that are considered to be silent. This approach was based on the idea that the implementation CRN would contain more species, states, and reactions than the formal CRN, which are added into the system to help the implementation practically. Thus, those extra implementation details should be ignored when trying to compare the essential behavior of the two CRNs. Firstly, we defined this CRN equivalence formally using three different forms of representation, to show some insights of the definition. Then based on this definition, we proceeded to establish an algorithm to verify the equivalence between a formal CRN and an implementation CRN.

Chapter 2

CRN Equivalence

This section presents the basic concepts and terminology, notions, and relationships between notions that define CRN equivalence.

2.1 Basic Concepts

A state is a multiset of species.

A reaction is an ordered pair of states (R, P), where R is called **reactants** and P is called **products**. We say the reaction is **trivial** if $R = P$.

A CRN is a set of nontrivial reactions. We refer to a pair of CRNs as formal CRN and implementation CRN, where the second CRN containing extra species, and the first CRN acts as a reference.

The notation $r \mid S$ denotes that the reaction $r = (R, P)$ can occur in state S, which means $R \subseteq S$. The notation $S \oplus r$ denotes $(S \setminus R) \cup P$, which is the resulting state after the reaction $r = (R, P)$ occurs in state S, if r can occur in S. Otherwise, $S \oplus r$ is invalid.

An interpretation is a mapping from the species in an implementation CRN to multisets of the species in a formal CRN, i.e. every implementation species is labeled with a multiset of formal species. For an interpretation m, if an implementation species p' is interpreted to a formal species p, then $m(p') = p$, which can be similarly applied to states and reactions. Thus, it is clear that if in an implementation CRN, $S' \oplus r' = T'$, then $m(S') \oplus m(r') = m(T')$ if $r = m(r')$ is a reaction in the formal CRN.

In our study, $S' \overset{r'}{\Rightarrow}_m T'$ denotes the following: with interpretation m, in the implementation CRN, state T' can be produced by $S' \oplus r'_1 \oplus r'_2 \cdots \oplus r'_k \oplus r' \oplus r'_{k+1} \oplus \cdots \oplus r'_n$, where $r'_i (i = 1 \ldots n)$ is interpreted to trivial reactions by m. It then follows that $m(S') \oplus m(r') = m(T')$.

The notaion $r' \parallel_m S'$ denotes that with interpretation m, the reaction r' can occur in some state produced by $S' \oplus r'_1 \oplus r'_2 \oplus \cdots \oplus r'_n$, where $r'_i (i = 1 \ldots n)$ is interpreted to trivial reactions.

A trajectory is an initial state followed by a finite or infinite sequence of reactions $S, r_1, r_2, \cdots$. A trajectory is **valid** if every reaction can occur in the state produced by all prior reactions. In an implementation CRN with interpretation m, it is evident that if $S' \overset{r'_1}{\Rightarrow}_m S'_1 \overset{r'_2}{\Rightarrow}_m S'_2 \cdots$ in CRN $\mathbb{B}$, and $m(r'_1) = r_1, m(r'_2) = r_2, \cdots$, then there exists a valid implementation trajectory $S', r'_{11}, \cdots, r'_{1n_1}, r'_1, r'_{21}, \cdots, r'_{2n_2}, r'_2 \cdots$, where $r'_{11}, \cdots, r'_{1n_1}, r'_{21}, \cdots, r'_{2n_2}, \cdots$ are all interpreted to trivial reactions. So the implementation trajectory becomes $S, r_1, r_2, \cdots$ after it has been interpreted.

2.2 Notions of CRN Equivalence

We define three different notions of equivalence between a formal CRN $\mathbb{A}$ and an implementation CRN $\mathbb{B}$ with interpretation m showing the right mapping between species.

I Equivalence in terms of trajectory

(i) For any valid trajectory $S, r_1, r_2, \cdots$ in CRN $\mathbb{A}$, there exists at least one state S' of CRN $\mathbb{B}$, such that $m(S') = S$. For every such state S', there exists a valid trajectory $S' \overset{r'_1}{\Rightarrow}_m S'_1 \overset{r'_2}{\Rightarrow}_m S'_2 \overset{r'_3}{\Rightarrow}_m \cdots$ in CRN $\mathbb{B}$, such that $m(r'_i) = r_i$.

(ii) For any valid trajectory in CRN $\mathbb{B}$, its interpreted trajectory is a valid trajectory in CRN $\mathbb{A}$.

II Three conditions of interpretation

(i) Atomic condition: For every formal species, there is at least one implementation species which is interpreted to exactly that formal species.

(ii) Delimiting condition: The set of formal reactions and the set of interpreted implementation reactions are the same, ignoring trivial reactions.

(iii) Permissive condition: If $r \mid S$, and $m(S') = S$, then there exists an implementation reaction r' for which $m(r') = r$ and $r' \parallel_m S'$.

III Weak Bisimulation

For every formal state S, there is at least one implementation state S', where $m(S') = S$. Further more, $m(S') = S$ implies,

(i) if $S \oplus r = T$, then there exists T' and r' such that $m(r') = r$, $m(T') = T$ and $S' \overset{r'}{\Rightarrow}_m T'$

(ii) if $S' \oplus r' = T'$, $m(r') = r$, then there exists a formal state T such that $m(T') = T$, and $S \oplus r = T$.

We can illustrate these definitions with some examples. Consider the following case,

formal CRN	implementation CRN	interpretation
$a + b \to c$	$a \leftrightarrow x$	$m(a) = a, m(b) = b, m(c) = c$
	$b + x \to c + y$	$m(x) = a, m(y) = \phi$

In the formal CRN, $\{a, a, b\} \overset{a+b\to c}{\longrightarrow} \{a, c\}$ is a valid trajectory, while in the implementation CRN, $\{b, x, x\} \overset{x\to a}{\longrightarrow} \{a, b, x\} \overset{b+x\to c+y}{\longrightarrow} \{a, c, y\}$ is a valid trajectory. After being interpreted, it becomes $\{a, a, b\} \overset{a\to a}{\longrightarrow} \{a, a, b\} \overset{a+b\to c}{\longrightarrow} \{a, c\}$, then after removing the first trivial step, it becomes the same as the formal trajectory.

For the interpretation, the atomic condition is satisfied by $m(a) = a, m(b) = b, m(c) = c$. The implementation CRN becomes $\{a \leftrightarrow a, a + b \to c\}$ after being interpreted, it would be exactly the same as the formal CRN after removing the first trivial reaction. Thus, the delimiting condition is satisfied. Further more, it is clear that for any implementation state that contains $\{a, b\}$ after being interpreted, $b + x \to c + y$, which is interpreted to $a + b \to c$, would be able to occur either immediately, or after $a \to x$ occurs. Thus the permissive condition is satisfied, since $a \to x$ is interpreted to a trivial reaction.

From a weak bisimulation point of view, we can observe the implementation state $\{b, x, x\}$ as an example: $m(b, x, x) = \{a, a, b\}$. In the formal CRN, $\{a, a, b\} \oplus (a + b \to c) = \{a, c\}$, while in the implementation CRN, $\{b, x, x\} \oplus (b + x \to c + y) = \{c, x, y\}$. With $m(b + x \to c + y) = (a + b \to c)$ and $m(c, x, y) = \{a, c\}$, the requirements for weak bisimulation are fulfilled, for this case.

The second example, below, illustrates why implementation species can be interpreted

as multisets. As an exercise, the reader can verify that the given interpretation is valid and prove that there is no valid interpretation that maps every implementation species to either a single formal species or to null.

formal CRN	implementation CRN	interpretation
$a + b \to c$	$a + b \leftrightarrow x$	$m(a) = a, m(b) = b, m(c) = c$
	$x \to c + y$	$m(x) = a + b, m(y) = \phi$

2.3 Equivalence between Notions

The three defined notions of CRN equivalence have their own intuition, respectively, and their own different forms of representation. Even so, we are able to prove that they are essentially equivalent to each other.

I $\to$ II

Proof For II(i), the proof is straight forward: every formal state is required to have at least one implementation state, if we consider formal states consisting of exactly one species. For II(ii), the prove is by contradiction: we suppose the two sets of reactions are different. This means either (a) there is a reaction $r = (R, P)$ in CRN $\mathbb{A}$, which does not have an implementation reaction, or (b) there is a reaction $r' = (R', P')$ in CRN $\mathbb{B}$, whose interpretation is not in CRN $\mathbb{A}$. For (a), a valid trajectory start with $R \oplus r$ would have no valid implementation trajectory, which is in contradiction with I(i). For (b), a valid trajectory start with $R' \oplus r'$ would have no valid interpreted trajectory, which is in contradiction with I(ii). Thus, we establish the proof for II(ii).

For II(iii), again the proof is by contradiction: we suppose $r \mid S$, and $m(S') = S$, however, there does not exists an implementation reaction r' in which $m(r') = r$ and $r' \mid S'$. A valid trajectory starting with $S \oplus r$ would have no valid implementation trajectory with initial

state S', which is in contradiction with I(i). Thus, the proof for II(iii) is established.

II $\rightarrow$ III

Proof Atomic condition directly implies that every formal state has at least one implementation state.

For III(i), $m(S') = S$, and $S \oplus r = T$, which implies $r \mid S$. Using II(iii), there exists r' such that $m(r') = r$ and $r' \parallel_m S'$. Also, there exists a state T', such that $S' \overset{r'}{\Rightarrow}_m T'$, and $m(S') = S$, $m(r') = r$. Then $S \oplus r = m(S') \oplus m(r') = m(T')$, which implies $T = m(T')$.

For III(ii), $S' \oplus r' = T'$, $m(r') = r$, so r is a reaction in CRN $\mathbb{A}$ using II(ii). Then $m(S') \oplus m(r') = m(T')$, which means $S \oplus r = m(T')$. So $m(T')$ is the formal state we want.

III $\rightarrow$ I

Proof For I(i), there is a valid formal trajectory $S, r_1, r_2, \cdots$ in CRN $\mathbb{A}$. As every formal state has at least one implementation state, there exists at least one implementation state S', such that $m(S') = S$. If $S_1 = S \oplus r_1$, then using III(i), there exists S'_1 and r'_1 such that $m(r'_1) = r_1$, $m(S'_1) = S_1$, and $S' \overset{r'_1}{\Rightarrow}_m S'_1$. With this step by step calculation for each reaction in the trajectory in CRN $\mathbb{A}$, we get a valid trajectory in CRN $\mathbb{B}$.

For I(ii), there is a valid implementation trajectory $S', r'_1, r'_2, \cdots$ in CRN $\mathbb{B}$. Let $r_1 = m(r'_1)$ and $S'_1 = S' \oplus r'_1$, then using III(ii), there exists a formal state S_1 such that $m(S'_1) = S_1$, and $S \oplus r_1 = S_1$. Again, with this step by step calculation for each reaction in

the trajectory in CRN $\mathbb{B}$, we get a valid trajectory in CRN $\mathbb{A}$.

Chapter 3

An Algorithm for Finding Valid Interpretations

Given a formal CRN $\mathbb{A}$ and an implementation CRN $\mathbb{B}$, our goal is to either find an interpretation that satisfies the three conditions we have described, which we call a **valid** interpretation, or assert that no such interpretation exists.

3.1 Complexity Analysis

We refer to CRNs that only contain single molecular to single molecular reactions smCRNs. Each species in a smCRN can be assigned a vertex. Each reaction can then be represented as a directed edge from one vertex to another (a bidirectional reaction would be represented as two edges between the same pair of vertices but with opposite directions). The smCRN can then be converted to a directed graph. Similarly, any directed graph can be converted to a smCRN in an analogous manner.

In this way, we can prove that if two directed graphs G and H have the same number of vertices, and their corresponding smCRNs are $\mathbb{A}$ and $\mathbb{B}$, then there exists an isomorphism between G and H if and only if there exists a valid interpretation from $\mathbb{B}$ to $\mathbb{A}$. A brief

explanation of this proof follows: A mapping $f : V(G) \to V(H)$ is an isomorphism, if and only if f is a bijection and it keeps the adjacent relation between vertices, which means any two vertices u and v are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H. This is equivalent to the delimiting condition. With the premise that G and H have the same number of vertices (thus $\mathbb{A}$ and $\mathbb{B}$ have the same number of species), bijection is equivalent to the atomic condition. The atomic condition and the delimiting condition imply the permissive condition with that premise.

Therefore, the problem of whether there exists a valid interpretation from one smCRN to another is as difficult as the problem of whether two directed graphs with the same number of vertices are isomorphic or not. But the second problem has no known polynomial time solution [11].

For most cases of interest, such as when a formal CRN is compiled to DNA, we are informed, for each formal species, of one implementation species that is interpreted to it exactly. This implies that the atomic condition is already satisfied. This extra information, however, does not make the problem easier.

3.2 Overview

Since a polynomial time solution is not likely to be found, our aim was to establish an efficient depth-first-search algorithm to solve this problem.

Input:

A formal CRN $\mathbb{A}$ consisting of N reactions; an implementation CRN $\mathbb{B}$ consisting of M reactions. (The extra information for satisfying the atomic condition is provided by using

formal species names for the appropriate species within the implementation CRN.)

Output:

Either a valid interpretation from $\mathbb{B}$ to $\mathbb{A}$, or the assertion that no such interpretation exists.

1. Establish a table T of size $M \times (N+1)$ with each element (i, j) being a Boolean variable which means the i_{th} reaction in CRN $\mathbb{B}$ can/cannot be interpreted to the j_{th} reaction in CRN $\mathbb{A}$. (Can/cannot refers to whether, if this pair of formal reaction and implementation reaction is considered individually, there exists any interpretation (valid or not when the rest of the reactions are considered) that interprets this implementation reaction to this formal reaction.) Initiate the table using $\mathbb{A}$ and $\mathbb{B}$.(Column $N+1$ in this table corresponds to a trivial reaction.)

2. If T has any row or any non-trivial column with all $False$ entries, then backtrack. Otherwise, find the column of T (without the trivial column) with the least possibilities (but at least one possibility) out of the columns that have not been implemented under the current interpretation. When all columns are implemented, go to step 3.

   (a) For each of the possibilities, enumerate all possible interpretations of unknown species in the current reaction. The number of possibilities will be finite here.

   (b) For each of these possibilities, update T, and do step 2 recursively.

3. Find the row of T with the least nontrivial possibilities out of the rows that have not been fully implemented.

   (a) For each of the possibilities, enumerate all possible interpretations of unknown species in the current reaction that satisfy the given formal reaction. For the possibility of being trivial, do not enumerate unknown species, just go to step 3 recursively.

(b) For each of these possibilities, update T. If T has any row or any non-trivial column with all $False$ entries, then backtrack.

(c) If no unknown species remain, then run the **Permissive Test**. If the permissive condition is satisfied, then a valid interpretation is found, so output the result and terminate. Otherwise, backtrack.

(d) If all remaining unknown reactions can be trivial, then run the **Trivial Reaction Solver**.

(e) Do step 3 recursively.

4. Now we have tried all the possibilities, and no valid interpretation is found, so we can assert that there is no valid interpretation from $\mathbb{B}$ to $\mathbb{A}$.

Step 1 is initialization. Step 2 finds an implementation reaction that interpreted to each formal reaction. Step 3 tries to fit the remaining implementation reaction into the formal CRN, leaving some unknown species in those reactions that will be interpreted to trivial reactions which will be decided in the **Trivial Reaction Solver**.

3.3 Trivial Reaction Solver

When the algorithm reaches this section, the problem has been reduced as follows: The delimiting condition is satisfied, and all the unknown species are in reactions that can be individually interpreted to trivial reactions. We want to find an appropriate interpretation that is a completion of the current interpretation and

(1) interprets the reactions containing unknown species to trivial reactions,

(2) does not violate the permissive condition, if possible.

In order to satisfy (1) we establish a system of equations for each formal species, where each equation corresponds to a trivial reaction. There should be the same number of molecules of a formal species in the reactants and the products of a trivial reaction.

For example, if the reactions to be interpreted as trivial are as follows,

$$\begin{aligned} A + B + x_1 + x_2 &\rightarrow 2B + 2x_2 \\ 2B + x_2 &\rightarrow A + x_1 \end{aligned}$$

Then for formal species A, we have the following equations,

$$\begin{aligned} x_1 - x_2 &= -A \\ -x_1 + x_2 &= A \end{aligned}$$

and we rewrite this in matrix form,

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Since we only want non-negative integer solutions, this is actually a system of linear Diophantine equations. Any feasible solution of the equations would satisfy (1). However, the number of feasible solutions could be infinite, and we can not afford to try them one by one. So we need to find solutions that satisfy (2). In order to do this, we introduce the following concept:

A solution is **minimal** if there is no other solution for which every component is less than or equal to its corresponding component. Because there must be at least one different component for two different solutions, there must be a component for the minimal solution that is strictly less than the other solution.

The number of minimal solutions for a system of linear Diophantine equations is limited.

Furthermore with the following theorem, we only need to consider one minimal solution.

Theorem Given a formal CRN $\mathbb{A}$, and an implementation CRN $\mathbb{B}$, if all non-formal species in $\mathbb{B}$ are in reactions that interpret to trivial reactions and a valid interpretation exists, then for any formal species D that ever occurs as a reactant in $\mathbb{A}$, the system of linear Diophantine equations established according to the trivial reactions for D has only one minimal solution.

Proof Suppose $x_1, x_2, \cdots, x_n$ are the non-formal species. If the system of equations has two different minimal solutions,

$$\begin{aligned} u &= (a_1, a_2, \cdots, a_n) \\ v &= (b_1, b_2, \cdots, b_n) \end{aligned}$$

then the reactions can all be trivial for

$$\begin{aligned} x_1 &= a_1 D + \cdots (\text{other formal species}) \\ x_2 &= a_2 D + \cdots (\text{other formal species}) \\ &\vdots \\ x_n &= a_n D + \cdots (\text{other formal species}) \end{aligned}$$

and also for

$$\begin{aligned} x_1 &= b_1 D + \cdots (\text{other formal species}) \\ x_2 &= b_2 D + \cdots (\text{other formal species}) \\ &\vdots \\ x_n &= b_n D + \cdots (\text{other formal species}) \end{aligned}$$

Since they are different minimal solutions, we can suppose u corresponds to a valid interpretation and $a_1 > b_1, b_2 > a_2$ with out loss of generality. Now let us consider the formal reaction containing D as a reactant

$$\boxed{S} + D \rightarrow \boxed{\text{products}}$$

where S stands for the reactants except this D. According to the permissive condition, since $a_1 > b_1 \geq 0$, there must be a series of trivial reactions which makes

$$\boxed{S} + x_1 \Rightarrow \boxed{\text{non-formal products}} + \boxed{\text{formal products}}$$

where the formal products $\supseteq S + D$ in order to let the formal reaction occur at this state (because the non-formal species only occur in trivial reactions, so the state must contain all the reactants explicitly).

Suppose the number of Ds that S contains is r, the number of Ds that the formal products contains is t. Then the number of Ds that x_1 contains is no less than $r - t + 1$ according to the series of trivial reactions. So we have $a_1 \geq r - t + 1$ and $b_1 \geq r - t + 1$.

If the non-formal products contain no Ds after interpretation, then $a_1 = b_1 = r - t + 1$, which is contradictory to $a_1 > b_1$. If the non-formal products contain some non-formal species x_i, which contains some Ds after interpretation, we can apply the above argument on x_i, and the same contradiction will occur. This recursion will end in a limited number of steps, since the number of Ds contained in the non-formal species we are considering is always decreasing, and the initial number a_1 is a limited amount.□

For any formal species that occur as a reactant, with the above theorem. we conclude that any minimal solution of the equation will satisfy both (1) and (2). For formal species that never occur as a reactant, it is clear that any feasible solution will not influence the permissive condition. Thus, we have shown that to find one minimal solution of the equation

and to use that to complete the current interpretation is sufficient for correctness of the **Trivial Reaction Solver**. The next question is how to find a minimal solution for a system of linear Diophantine equations. We use the algorithm introduced by Contejean and Devie [8], which established an efficient incremental algorithm for homogeneous equations; for non-homogeneous equations $A\vec{x} = \vec{b}$, a new matrix A' is constructed by first appending $-\vec{b}$ to A as the last column, and then looking for a solution of $A'\vec{x} = \vec{0}$ with the last component being 1.

3.4 Permissive Test

For the permissive test, we start with a formal CRN $\mathbb{A}$, an implementation CRN $\mathbb{B}$, and an interpretation from $\mathbb{B}$ to $\mathbb{A}$. The task is to assert whether or not this interpretation satisfies the permissive condition. We consider the complexity of this problem.
Consider a reachability problem: suppose CRN $\mathbb{H}$ consists of n reactions,

$$\{R_1 \to P_1, R_2 \to P_2, \cdots, R_n \to P_n\}$$

where all the species involved are $x_1, x_2, \cdots, x_m$. We ask: from a starting state S, can the target state T be reached or not, by carrying out a series of reactions in CRN $\mathbb{H}$?
For such a reachability problem, we can construct the following permissive problem:

- Let the formal CRN $\mathbb{A}$ be $\{a \to b\}$
- Let the implementation CRN $\mathbb{B}$ be

$$\{R_1 \to P_1, R_2 \to P_2, \cdots, R_n \to P_n, a \to a + S, a + T \to b\}$$

- Let the interpretation be $x_1 = \phi, x_2 = \phi, \cdots, x_m = \phi$

Observe that the interpretation from $\mathbb{B}$ to $\mathbb{A}$ would satisfy the permissive condition if and only if state S can reach state T in CRN $\mathbb{H}$. The reachability problem in a CRN is almost the same as the reachability problem in a VAS, where the exact complexity is still unknown [2]. And the best known algorithm for reachability problem in a CRN is bounded by a primitive recursive function [1]. So it is not likely that a polynomial algorithm can be found for the reachability problem, or for the permissive test we are considering here.

Thus, our algorithm tests the permissive condition in an exhaustive way. This means for a formal reaction $R \to P$, and an implementation state S that is a superset of R after being interpreted, we enumerate all the possible trivial reaction chains starting from S, up to a manually set maximum length, and determine if any implementation reaction that interprets to $R \to P$ can occur after the trivial reactions.

We wanted the set of implementation states that must be tested to be as small as possible. The smallest set is the collection of all the **minimal** states, where "minimal" means that the state has no strict subset that is a superset of R after being interpreted. To construct the set of minimal states for a formal reaction is also a challenging problem, so we elected to use a slower but simpler method. Our algorithm enumerates all such implementation states T where the i_{th} element of T contains the i_{th} element of R after being interpreted. For each T, the algorithm considers all its subsets, and if the subset contains R after being interpreted, that subset is tested. In this way, all the minimal states are be tested (as well as some superfluous states).

Chapter 4

Results

In order to run our algorithm on practical test cases, we used the BioCRN compiler introduced by shin [7]. The BioCRN compiler takes a formal CRN and a translation scheme as input, and produces a DNA-based implementation CRN as output. As an example translation scheme, the following two figures (taken from Soloveichik et al. [9]) briefly explains how the translation scheme invented by David Soloveichik works to construct an implementation CRN.

Figure 4.3 and figure 4.4 present example results from our program.

Figure 4.5 shows the results of all test cases; and compares them with the results of Seung Woo Shin's verifier (introduced in chapter 4 of his thesis [7]). A different notion of CRN equivalence was defined by shin [7]. Instead of looking for a valid interpretation from the implementation CRN to the formal CRN, Shin tried to decompose the implementation CRN into prime formal pathways and then compare the set of pathways with the formal CRN. The data in this figure indicate that, in some cases, the two different definitions of CRN equivalence yield different conclusions.

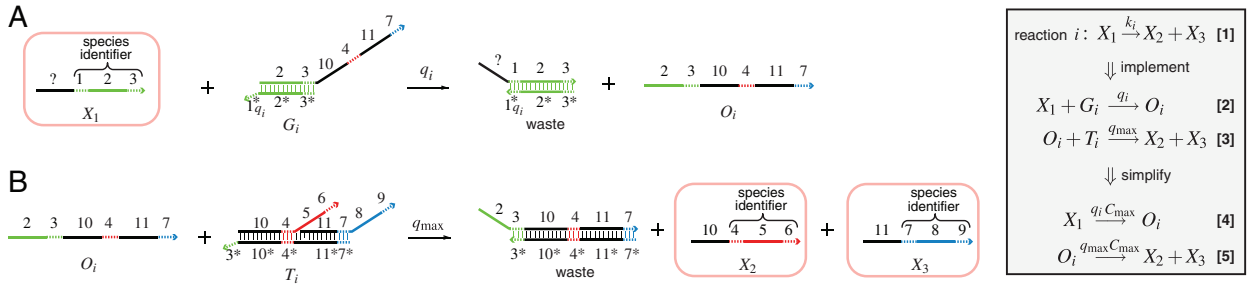

Figure 4.1: Unimolecular module of Soloveichik's translation scheme, shows the DNA implementation of the formal unimolecular reaction $X_1 \rightarrow X_2 + X_3$ with reaction index i. Orange boxes highlight the DNA species that correspond to the formal species X_1 (species identifier 1-2-3), X_2 (4-5-6), and X_3 (7-8-9). Domains identical or complementary to species identifiers for X_1, X_2, and X_3 are colored red, green, and blue, respectively. Black domains (10 and 11) are unique to this formal reaction.
A. Complex G_i undergoes a strand displacement reaction with strand X_1, with X_1 displacing strand O_i.
B. O_i displaces X_2 and X_3 from complex T_i.

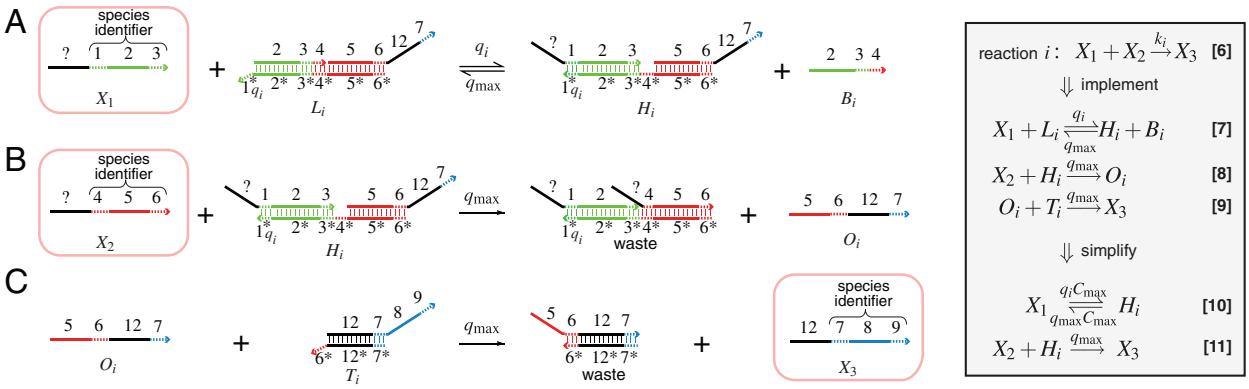

Figure 4.2: Bimolecular module of Soloveichik's translation scheme, shows DNA implementation of the formal bimolecular reaction $X_1 + X_2 \rightarrow X_3$ with reaction index i. The black domain (12) is unique to this formal reaction.
A. X_1 reversibly displaces B_i from complex L_i producing complex H_i.
B. X_2 displaces O_i from complex H_i . Occurrence of reaction B precludes the backward reaction of A.
C. O_i displaces X_3 from complex T_i.

```
verify ts/soloveichik.ts crn/crn6.crn --bisimulation
Original CRN:
    A + B -> A + C
    B + A -> X + B + A
    X -> X + Y + X
    Y + B -> Y + X + A
    X ->
    X + A -> C

Compiled CRN:
    i119 -> i194 + A + B + X
    i2232 -> A
    i2232 + B -> i2300 + i2340
    i2300 -> i2392 + A + C
    i394 -> i1457 + X + X + Y
    i575 -> X
    i575 + A -> i599 + i631
    i599 -> i3032 + C
    i73 -> B
    i73 + A -> i119 + i120
    i778 -> Y
    i778 + B -> i842 + i886
    i842 -> i969 + A + X + Y
    A -> i2232
    B -> i73
    X -> i14 + i15
    X -> i15 + i394
    X -> i575
    Y -> i778

Valid interpretation:
    i842 -> Y + X + A
    i394 -> X + Y + X
    i119 -> X + B + A
    i2300 -> A + C
    i778 -> Y
    i575 -> X
    i599 -> C
    i2232 -> A
    i73 -> B
    i886 ->
    i14 ->
    i15 ->
    i631 ->
    i2340 ->
    i120 ->
    i194 ->
    i1457 ->
    i2392 ->
    i969 ->
    i3032 ->
verify: compilation was correct.
```

Figure 4.3: First example result, used the translation scheme from the file "soloveichik.ts", and the formal CRN from the file "crn6.crn", which is shown in the "Original CRN" section. The BioCRN Compiler constructed the implementation CRN as shown in "Compiled CRN" section. The program successfully identified a valid interpretation.

```
verify ts/qian.ts crn/crn4.crn --bisimulation
Original CRN:
    A + B -> B + B
    B + C -> C + C
    C + A -> A + A

Compiled CRN:
    i124 -> i157 + A
    i124 -> i59 + A
    i157 -> i218 + A
    i157 + A -> i124
    i218 -> i1003 + i1004
    i218 + A -> i157
    i358 -> A
    i358 + B -> i407
    i407 -> i358 + B
    i407 -> i448 + B
    i448 -> i509 + B
    i448 + B -> i407
    i509 -> i1078 + i1079
    i509 + B -> i448
    i59 -> C
    i59 + A -> i124
    i660 -> B
    i660 + C -> i730
    i730 -> i660 + C
    i730 -> i780 + C
    i780 -> i853 + C
    i780 + C -> i730
    i853 -> i1153 + i1154
    i853 + C -> i780
    A -> i358
    B -> i660
    C -> i59
Delimiting condition cannot be satisfied.
There is implementation reaction not in formal CRN.
Max search depth reached: 3
with interpretation:
    i218 -> A + B
    i1003 -> B + B
    i509 -> C + A
    i1078 -> A + A
    i853 -> B + C
    i1153 -> C + C
    i1004 ->
    i1079 ->
    i1154 ->
verify: compilation was incorrect.
```

Figure 4.4: Second example result, used the translation scheme from the file "qian.ts" (introduced by Qian et al. [10]), and the formal CRN from the file "crn4.crn". The program failed to find a valid interpretation, implying that no valid interpretation exists; as an aid to the user, it output the best (invalid) interpretation that it found.

[pathway/bisimulation]	crn1	crn2	crn3	crn4	crn5	crn6	crn7	crn8
cardelli_2domain	T/T	I/T	I/T	I/T	R/R	I/T	I/I	I/T
cardelli_2domain_nogc	T/T	I/T	T/T	I/T	R/R	I/T	I/I	I/T
cardelli_SA	T/T	T/T	T/T	T/T	R/R	T/T	T/T	T/T
cardelli_SA_noGC	C/C	C/C	C/C	C/C	R/R	C/C	C/C	C/C
cardelli_fj	I/I	I/I	I/I	C/C	R/R	T/T	T/C	T/T
qian	I/I	I/I	I/T	I/I	I/I	I/T	C/I*	I/I
qian_bug	C/C	C/C	I/I	I/T	I/I	I/T	I/I	I/I
qian_fixed	C/C	C/C	T/T	C/C	I/T	T/T	C/C	T/C
qian_fixedrev	C/C	C/C	T/T	C/C	I/T	T/T	C/C	T/T
qian_rev	I/I	I/I	I/T	I/I	I/I	I/T	C/I*	I/I
soloveichik	C/C	C/C	C/C	C/C	I/I	C/C	I/I	I/I

[pathway/bisimulation]	crn9	crn10	crn11	crn12	crn13	crn14	crn15	crn16
cardelli_2domain	R/R	I/T	I/T	I/T	I/T	T/C	I/T	R/R
cardelli_2domain_nogc	R/R	I/T	I/I	I/T	I/T	T/C	I/T	R/R
cardelli_SA	R/R	T/T	T/C	T/T	T/T	C/C	T/T	R/R
cardelli_SA_noGC	R/R	T/C	C/C	C/C	C/C	C/C	C/C	R/R
cardelli_fj	R/R	T/T	T/T	T/T	I/I	I/C*	T/T	R/R
qian	I/I	I/I	C/I*	I/C*	I/T	I/C*	I/C*	I/I
qian_bug	I/I	I/I	C/I*	I/I	I/I	C/C	I/C*	I/I
qian_fixed	I/C*	T/T	C/C	T/C	T/T	C/C	T/C	I/C*
qian_fixedrev	I/C*	T/T	C/C	I/C*	I/C*	C/C	I/C*	I/C*
qian_rev	I/I	I/I	C/I*	I/C*	I/I	I/C*	I/C*	I/I
soloveichik	I/I	I/I	I/I	I/I	C/C	C/C	I/I	I/C*

C=correct, I=incorrect, T=timeout, R=resultless due to compiler crash, e.g. 0 arguments.
* indicates significant difference.

Figure 4.5: Results compared with Shin's verifier

Chapter 5

Discussion

In this study, we defined the equivalence between a formal CRN and an implementation CRN using a bisimulation approach. The definition was presented using three different forms of representation, each of which provides a distinct intuition for how to characterize the behavior of a CRN. Based on the definition, we developed an algorithm to verify CRN implementations by trying to construct a valid interpretation from the implementation CRN to the formal CRN.

There are many opportunities for further exploration of the bisimulation approach to CRN equivalence:

- In the well-established theory of bisimulation, various properties and theorems could also be applied in this context of CRN equivalence, and these may leads to deeper insights into the behavior of CRNs.

- Further exploration of the CRN equivalence properties we have defined is needed. For example, with a given implementation CRN and set of formal species, is the formal CRN uniquely equivalent to the implementation CRN? Or, if implementation CRN $\mathbb{B}$ is equivalent to formal CRN $\mathbb{A}$ and implementation CRN $\mathbb{D}$ is equivalent to formal CRN $\mathbb{C}$, and if we then compose $\mathbb{B}$ with $\mathbb{D}$, and $\mathbb{A}$ with $\mathbb{C}$, are they still equivalent?

- For translation schemes, such as those mentioned in the results section, which translate a formal CRN to a DNA-based implementation CRN, the relationship between the two CRNs should be further explored. It maybe true that the implementation CRN is always equivalent to the formal CRN for some translation schemes. This would be an important property to prove mathematically if so.

- Augmented translation schemes exist that yield infinite CRNs by implementing DNA polymer reactions, thus yielding efficient Turing-universal behavior [10]. Extending the bisimulation approach to validate such implementations is an important goal.

Also for the algorithm of finding a valid interpretation, we envision two possible improvements:

- For the complexity analysis, we have proven that the problem is at least as difficult as graph isomorphism. This should be further studied to determine if this is actually NP-complete or not.

- Although there is no polynomial time algorithm to solve the permissive test, there could be implementations with better efficiency. For example, a well-constructed hash table and using BFS instead of DFS may lead to better performance.

Reference

[1] Cook, M., Soloveichik, D., Winfree, E., Bruck, J. (2009). Programmability of chemical reaction networks. *Algorithmic Bioprocesses*, 543-584.

[2] Leroux, J. (2011). Vector addition system reachability problem: a short self-contained proof. *ACM SIGPLAN Notices*, 46(1), 307-316.

[3] Leroux, J. (2012). Vector Addition Systems Reachability Problem (A Simpler Solution). *EPiC*, 10, 214-228.

[4] Hack, M. (1976). The equality problem for vector addition systems is undecidable. *Theoretical Computer Science*, 2(1), 77-95.

[5] Esparza, J. (1998). Decidability and complexity of Petri net problems - an introduction. *Lectures on Petri Nets I: Basic Models*, 374-428.

[6] Laroussinie, F., Schnoebelen, P. (2000). The state explosion problem from trace to bisimulation equivalence. *Foundations of Software Science and Computation Structures* 192-207.

[7] Shin, S. W. (2011). *Compiling and Verifying DNA-based Chemical Reaction Network Implementations*. Master Thesis, California Institute of Technology.

[8] Contejean, E., Devie, H. (1994). An efficient incremental algorithm for solving systems of linear Diophantine equations. *Information and Computation*, 113, 143-172.

[9] Soloveichik, D., Seelig, G., Winfree, E. (2010). DNA as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences*, 107(12), 5393-5398.

[10] Qian, L., Soloveichik, D., Winfree, E. (2011). Efficient Turing-universal computation with DNA polymers. *DNA Computing and Molecular Programming*, 123-140.

[11] Jenner, B., Köbler, J., McKenzie, P., Torán, J. (2003). Completeness results for graph isomorphism. *Journal of Computer and System Sciences*, 66(3), 549-566.

[12] Liekens, A., Fernando, C. (2007). Turing complete catalytic particle computers. *Lecture Notes in Computer Science*, 1202-1211.

[13] Horn, F., Jackson, R. (1972). General mass action kinetics. *Archive for Rational Mechanics and Analysis*, 47(2), 81-116.

[14] Feinberg, M. (1972). Complex balancing in general kinetic systems. *Archive for Rational Mechanics and Analysis*, 49(3), 187-194.

[15] Soloveichik, D., Cook, M., Winfree, E., Bruck, J. (2008). Computation with finite stochastic chemical reaction networks. *Natural Computing*, 7(4), 615-633.

[16] Milner, R. (1989). *Communication and Concurrency*. Prentice-Hall, Inc..

Appendix: Detailed Pseudo-code

The detailed pseudo-code for the algorithm is listed here, consisting of seven functions: *Substitute*, *EquationsSolve*, *UpdateTable*, *PermissiveTest*, *SearchRow*, *SearchColumn*, and *Main*.

function *Substitute*(*CRN*, *interpretation*)
▷ Substitute implementation species for formal species in *CRN* according to *interpretation*
  $newCRN \leftarrow emptyList$
  for $reaction \in CRN$ **do**
    $[reactant, product] \leftarrow reaction$
    $newReactant \leftarrow emptySet$
    for $i \in reactant$ **do**
      if i is formal species **then**
        $newReactant \leftarrow newReactant \cup i$
      else
        for $j \in interpretation$ **do**
          $[implementationSpecies, setOfFormalSpecies] \leftarrow j$
          if $i = implementationSpecies$ **then**
            $newReactant \leftarrow newReactant \cup setOfFormalSpecies$
          end if
        end for
      end if
    end for
    $newProduct \leftarrow emptySet$
    for $i \in product$ **do**
      if i is formal species **then**
        $newProduct \leftarrow newProduct \cup i$
      else
        for $j \in interpretation$ **do**
          $[implementationSpecies, setOfFormalSpecies] \leftarrow j$
          if $i = implementationSpecies$ **then**
            $newProduct \leftarrow newProduct \cup setOfFormalSpecies$
          end if
        end for

end if
end for
$append(newCRN, [newReactant, newProduct])$
end for
return $newCRN$
end function

Example input:
$CRN = [[\{a\}, \{b, x\}], [\{b\}, \{b, y\}]]$
$interpretation = [[x, \{a\}], [y, \{a, b\}]]$
Example output:
$[[\{a\}, \{b, a\}], [\{b\}, \{b, a, b\}]]$
This example means CRN:

$$\begin{aligned} a &\rightarrow b + x \\ b &\rightarrow b + y \end{aligned}$$

with interpretation:

$$\begin{aligned} x &= a \\ y &= a + b \end{aligned}$$

becomes:

$$\begin{aligned} a &\rightarrow b + a \\ b &\rightarrow b + a + b \end{aligned}$$

function *EquationsSolve*(a,s)
▷ Find a non-negative integer solution $\vec{x}$ of the equation $a\vec{x} = \vec{b}$ or return $\vec{0}$ when there is no such solution
append $-\vec{b}$ to a as its last column
▷ Now the problem is converted to finding a solution $\vec{x}$ of the equation $a\vec{x} = \vec{0}$ with the last component of $\vec{x}$ being 1 for the new a
$q \leftarrow$ number of columns in a
$p \leftarrow Push(\vec{0}, emptyStack)$
$s \leftarrow emptySet$
for $i = 1$ **to** q **do**
$Frozen[1, i] \leftarrow False$
end for
while $p \neq emptyStack$ **do**
$top \leftarrow top(p)$
$p \leftarrow pop(p)$
$n \leftarrow height(p) - 1$
if $a \times top = \vec{0}$ and $top \neq \vec{0}$ **then**
if the last component of top is 1 **then**
▷ Here we find the solution needed. If we remove this if-statement, s will contain all the non-zero minimal solutions of $a\vec{x} = \vec{0}$ after the big **while** loop
return top
end if
$s \leftarrow s \cup \{top\}$
else
for $i = 1$ **to** q **do**
$F[i] \leftarrow Frozen[n + 1, i]$
end for
for $i = 1$ **to** q **do**
if $F[i] = False$ and $(a \times top) \cdot (a \times \vec{e_i}) < 0$ **then**
if $t + \vec{e_i}$ is minimal in s **then**
$p \leftarrow push(t + \vec{e_i}, p)$
$n \leftarrow n + 1$
for $j = 1$ **to** q **do**
$Frozen[n, j] \leftarrow F[j]$
end for
$F[i] \leftarrow True$
end if
end if
end for
end if
end while
return $\vec{0}$

end function

Example input:
$a = [[1, 2, -4], [2, 1, -5]]$
Example output:
$[2, 1, 1]$
This example means equations:

$$\begin{aligned} x_0 + 2x_1 - 4x_2 &= 0 \\ 2x_0 + x_1 - 5x_2 &= 0 \end{aligned}$$

has solution $x_0 = 2, x_1 = 1, x_2 = 1$

function $UpdateTable(formalCRN,\ implementationCRN)$

▷ Calculate the table of every implementation reaction can/cannot be individually interpreted to each formal reaction and trivial reaction. Note that implementation CRN have multiple repeated and/or trivial reactions – this is as it should be.

for $j = 1$ **to** $length(implementationCRN)$ **do**
$[implementationReactant, implementationProduct] \leftarrow implementationCRN[j]$
for $i = 1$ **to** $length(formalCRN)$ **do**
$[formalReactant, formalProduct] \leftarrow formalCRN[i]$
$cR \leftarrow formalReactant \cap implementationReactant$
▷ cR consists of all common species of $formalReactant$ and $implementationReactant$
$fR \leftarrow formalReactant \setminus cR$
$iR \leftarrow implementationReactant \setminus cR$
$cP \leftarrow formalProduct \cap implementationProduct$
▷ cP consists of all common species of $formalProduct$ and $implementationProduct$
$fP \leftarrow formalProduct \setminus cP$
$iP \leftarrow implementationProduct \setminus cP$
$ci \leftarrow$ common non-formal species of iR and iP
$cf \leftarrow fR \cap fP$
if $ci \neq emptySet$ **then**
$iR \leftarrow iR \setminus ci$
$iP \leftarrow iP \setminus ci$
$fR \leftarrow fR \setminus cf$
$fP \leftarrow fP \setminus cf$
end if
▷ Set $table[i, j]$ to $True$ if $implementationCRN[j]$ can be interpreted to $formalCRN[i]$, otherwise set $table[i, j]$ to $False$
if no formal species in iR and iP **then**
if $fR = emptySet$ **then**
if $fP = emptySet$ **then**
$table[i, j] \leftarrow True$
else
if $iP = emptySet$ **then**
$table[i, j] \leftarrow False$
else
$table[i, j] \leftarrow True$
end if
end if
else
if $iR = emptySet$ **then**
$table[i, j] \leftarrow False$
else
if $fP = emptySet$ **then**

```
                    table[i, j] ← True
                else
                    if iP = emptySet then
                        table[i, j] ← False
                    else
                        table[i, j] ← True
                    end if
                end if
            end if
        else
            table[i, j] ← False
        end if
    end for                    ▷ Now check if this implementation reaction can be trivial
    ci ← implementationReactant ∩ implementationProduct
    iR ← implementationReactant \ ci
    iP ← implementationProduct \ ci
    if (iR has no formal species) or (iP has non-formal species) then
        if (iP has no formal species) or (iR has non-formal species) then
            table[i, length(implementationCRN) + 1] ← True
        else
            table[i, length(implementationCRN) + 1] ← False
        end if
    else
        table[i, length(implementationCRN) + 1] ← False
    end if
  end for
  return table
end function
```

Example input:
$formalCRN = [[\{a\}, \{b\}], [\{b\}, \{a\}]]$
$implementationCRNx = [[\{a\}, \{x\}], [\{b\}, \{x\}]]$
Example output:
$[[True, False, True], [False, True, True]]$
This example means with formal CRN: $a \to b$
and implementation CRN: $a \to x, b \to x$
the table is

$$\begin{pmatrix} True & False & True \\ False & True & True \end{pmatrix}$$

which implies implementation reaction $a \to x$ can be interpreted to $a \to b$ and trivial reaction but not $b \to a$, while implementation reaction $b \to x$ can be interpreted to $b \to a$ and trivial reaction but not $a \to b$.

function $PermissiveTest(formalCRN,\ implementationCRN,\ interpretation)$
function $Search(state,\ depth)$
▷ In the implementation CRN, if after a series of trivial reactions starting from *state* within length of *depth*, some reaction in *targetReaction* can occur, then return $True$. Otherwise return $False$
if $depth < 0$ **then**
return $False$
end if
for $i \in targetReaction$ **do**
▷ If any reaction in *targetReaction* can occur in current state, then permissive condition is not violated
$[reactant, product] \leftarrow i$
if $reactant \setminus state = emptySet$ **then**
return $True$
end if
end for
for $i \in trivialReaction$ **do**
▷ If any reaction in *targetReaction* can occur in current state, then search deeper
$[reactant, product] \leftarrow i$
if $reactant \setminus state = emptySet$ **then**
$newState = (state \setminus reactant) \cup product$
if $Search(newState, depth - 1) = True$ **then**
return $True$
end if
end if
end for
end function
$newCRN \leftarrow Substitute(implementationCRN, interpretation)$
$table = UpdateTable(formalCRN, newCRN)$
$trivialReaction \leftarrow emptySet$
▷ Here we collect the set of implementation reactions which can individually be interpreted as trivial
for $i = 1$ **to** $length(newCRN)$ **do**
if $table[i, length(formalCRN + 1)] = True$ **then**
$trivialReaction \leftarrow trivialReaction \cup implementationCRN[i]$
end if
end for
for $i = 1$ **to** $length(formalCRN)$ **do**
▷ Here we collect the set of implementation reactions which can individually be interpreted to the i_{th} formal reaction
$targetReaction \leftarrow emptySet$
for $j = 1$ **to** $length(newCRN)$ **do**

```
            if table[j, i] = True then
                targetReaction ← targetReaction ∪ implementationCRN[i]
            end if
        end for
        [reactant, product] ← formalCRN[i]
    ▷ Construct setsOfSpecies where setsOfSpecies[n] contains all the implementation species whose interpretation contains the n_th element of reactant
        n ← 1
        for j ∈ reactant do
            setsOfSpecies[n] ← j
            for k ∈ interpretation do
                [implementationSpecies, setOfFormalSpecies] ← k
                if j ∈ setOfFormalSpecies then
                    setsOfSpecies[n] ← setsOfSpecies[n] ∪ {implementationSpecies}
                end if
            end for
            n ← n + 1
        end for
        ▷ Generate all the implementation states whose i_th element is in setsOfSpecies[i]
        implementationStates ← [emptySet]
        for j = 1 to length(reactant) do
            newStates ← emptySet
            for k ∈ implementationStates do
                for s ∈ setsOfSpecies[j] do
                    newStates ← newStates ∪ {k ∪ {s}}
                end for
            end for
            implementationStates ← newStates
        end for
        for j ∈ implementationStates do
            for k ⊆ j do
                if interpretation of k is superset of reactant then
                    if Search(k, maxPermissiveDepth) = False then
                        return False
                    end if
                end if
            end for
        end for
    end for
    output interpretation
    return True
end function
```

Example input:
$formalCRN = [[\{a\}, \{b\}], [\{b\}, \{c\}]]$
$implementationCRN = [[\{a\}, \{x\}], [\{y\}, \{c\}]]$
$interpretation = [[x, \{b\}], [y, \{b\}]]$
Example output:
$False$
This example means with formal CRN: $a \to b, b \to c$
implementation CRN: $a \to x, y \to b$
and interpretation: $x = b, y = b$
the permissive condition is not satisfied.
This is because in formal state $\{b\}$, reaction $b \to c$ can occur, while in implementation state $\{x\}$, which is interpreted to formal state $\{b\}$, no further reaction can occur.

function $SearchRow(formalCRN, implementationCRN, interpretation, unknownRow)$
  if $unknownRow = emptySet$ **then**
    return $PermissiveTest(formalCRN, implementationCRN, interpretation)$
  end if
  $newCRN \leftarrow Substitute(implementationCRN, Interpretation)$
  $table \leftarrow UpdateTable(formalCRN, newCRN)$
  if there is a row in *table* all being $False$ **then**
    return $False$
  end if
  if all implementation reactions whose index in $unknownRow$ can be trivial **then**
    construct equations of trivial reactions and call $EquationSolve$
    complete the current interpretation as $newInterpretation$
      using the result of $EquationSolve$
    if $PermissiveTest(formalCRN, implementationCRN,$
      $newInterpretation) = True$ **then**
      return $True$
    end if
  end if
  $minIndex \leftarrow indexoftherowwithleast$Trues
  $newUnknown \leftarrow remove(unknownRow, minIndex)$
  if $table[minIndex, lenght(formalCRN) + 1] = True$ **then**
    ▷ Leave the implementation reaction with $minIndex$ as trivial by removing it from $unknowRow$, but not enumerating unknown species in it.
    if $SearchRow(formalCRN, implementationCRN,$
      $interpretation, newUnknown) = True$ **then**
      return $True$
    end if
  end if
    ▷ Get here if the row with $minIndex$ cannot be trivial or no valid interpretation if it is. So we now explore whether there is a valid interpretation in which this reaction is non-trivial
  $[implementationReactant, implementationProduct] \leftarrow newCRN[minIndex]$
  for $k = 1$ **to** $length(formalCRN)$ **do**
    if $table[minIndex][k] = True$ **then**
  ▷ Now we enumerate all possible interpretations which interpreted the implementation reaction with $minIndex$ to the k_{th} formal reaction
      $[formalReactant, formalProduct] \leftarrow formalCRN[k]$
      $unknownReactant \leftarrow implementationReactant \setminus formalReactant$
      $numR \leftarrow length(unknownReactant)$
      $knownReactant \leftarrow formalReactant \setminus implementationReactant$
      $enumR \leftarrow$ all possibility of divide $knownReactant$ into $numR$ sets
      for $i \in enumR$ **do**

```
            newInterpretation ← interpretation
            for p = 1 to numR do
                append(newInterpretation, [unknownReactant[p], i[p]])
            end for
            apply newInterpretation to unknownProduct to get newProduct
            unknownProduct ← newProduct \ formalProduct
            numP ← length(unknownProduct)
            knownProduct ← formalProduct \ newProduct
            enumP ← all possibility of divide knownProduct into numP sets
            for j ∈ enumP do
                for p = 1 to numP do
                    append(newInterpretation, [unknownProduct[p], j[p]])
                end for
                if SearchRow(formalCRN, implementationCRN,
                     newInterpretation, newUnknown) = True then
                    return True
                end if
            end for
        end for
    end if
end for
end function
```

This *SearchRow* function continues the search of *SearchColumn*. It will call *EquationSolve* if all unknown implementation reactions can be trivial. Then the function finds an unknown implementation reaction with smallest number of possible corresponding formal reactions. For each possibility, enumerate all possible interpretations of unknown species in the current reaction. The function keeps doing this recursively until no unknown reactions left, then go for permissive test.

function $SearchColumn(formalCRN, implementationCRN, interpretation, unknownColumn)$
  $newCRN \leftarrow Substitute(implementationCRN, Interpretation)$
  if **then**$unknownColumn = emptySet$
    $unknownRow \leftarrow$ list of indices of reactions in $newCRN$
      containing non-formal species
    return $SearchRow(formalCRN, implementationCRN, interpretation, unknownRow)$
  end if
  $table \leftarrow UpdateTable(formalCRN, newCRN)$
  if there is a row or a column in $table$ all being $False$ **then**
    return $False$
  end if
  $minIndex \leftarrow indexofthecolumnwithleast$Trues
  $newUnknown \leftarrow remove(unknownColumn, minIndex)$
  $[formalReactant, formalProduct] \leftarrow formalCRN[minIndex]$
  for $k = 1$ **to** $length(implementationCRN)$ **do**
    if $table[k][minIndex] = True$ **then**
      ▷ Now we enumerate all possible interpretations which interpreted the k_{th} implementation reaction to the formal reaction with $minIndex$
      $[implementationReactant, implementationProduct] \leftarrow implementationCRN[k]$
      $unknownReactant \leftarrow implementationReactant \setminus formalReactant$
      $numR \leftarrow length(unknownReactant)$
      $knownReactant \leftarrow formalReactant \setminus implementationReactant$
      $enumR \leftarrow$ all possibility of divide $knownReactant$ into $numR$ sets
      for $i \in enumR$ **do**
        $newInterpretation \leftarrow interpretation$
        for $p = 1$ **to** $numR$ **do**
          $append(newInterpretation, [unknownReactant[p], i[p]])$
        end for
        apply $newInterpretation$ to $unknownProduct$ to get $newProduct$
        $unknownProduct \leftarrow newProduct \setminus formalProduct$
        $numP \leftarrow length(unknownProduct)$
        $knownProduct \leftarrow formalProduct \setminus newProduct$
        $enumP \leftarrow$ all possibility of divide $knownProduct$ into $numP$ sets
        for $j \in enumP$ **do**
          for $p = 1$ **to** $numP$ **do**
            $append(newInterpretation, [unknownProduct[p], j[p]])$
          end for
          if $SearchColumn(formalCRN, implementationCRN,$
            $newInterpretation, newUnknown) = True$ **then**
            return $True$
          end if
        end for

end for
end if
end for
end function

This *SearchColumn* function finds a formal reaction with smallest number of possible corresponding implementation reactions. For each possibility, enumerate all possible interpretations of unknown species in the implementation reaction. The function keeps doing this recursively until every formal reaction has a corresponding implementation reaction, then it will call *SearchRow*.

function $Main(formalCRN,\ implementationCRN)$
$interpretation \leftarrow emptySet$
$unknownColumn = [1, 2, \cdots, length(formalCRN)]$
return $SearchColumn(formalCRN, implementationCRN, interpretation, unknownColumn)$
end function

This is the main function of the algorithm, take $formalCRN$ and $implementationCRN$ as input, return $True$ and output a valid interpretation if such an interpretation exists, otherwise return $False$.