Using Transitivity and Modularity in Chemical Reaction Network Bisimulation

Robert F. Johnson[1]

Bioengineering, California Institute of Technology

The (stochastic) Chemical Reaction Network (CRN) model is commonly used in molecular programming to represent both real and abstract systems. Abstract CRNs can be designed to perform tasks ranging from simple tasks, such as comparing two counts [1] or oscillating with a given period [4], to complex tasks such as simulating Turing machines [10]. To make CRNs practical, there are multiple ways to construct a DNA strand displacement (DSD) system that implements a given abstract CRN [11, 3]. From this perspective the CRN model can be used as a programming language, with the various DSD implementation schemes taking the role of compilers. With compilers comes a desire for verification that the compiled programs are correct. Reaction enumerators such as Visual DSD [8] and Peppercorn [5] enumerate an "implementation CRN" that models a given DSD system, at which point there are a number of formal verification methods to compare an implementation CRN to the original "formal CRN", including pathway decomposition [9], serializability analysis [7], and CRN bisimulation [6]. This talk focuses on CRN bisimulation, and is based largely on our previously published work on the topic [6], but also covers some practical uses of CRN bisimulation not mentioned in that paper.

CRN bisimulation takes a formal CRN with species $\mathcal{S}$ and reactions $\mathcal{R}$, and an implementation CRN with species $\mathcal{S}'$ and reactions $\mathcal{R}'$, and asks whether there is an "interpretation" of the implementation species as multisets of formal species under which the implementation CRN "looks like" the formal CRN. If so, then the implementation is correct, and the verifying interpretation is called "a CRN bisimulation". Algorithms to find such an interpretation, and to check whether such an interpretation is a bisimulation, are given in our previous work, but they are intractable: PSPACE- or NP-complete depending on the assumptions used [6]. To help make checking bisimulation more tractable, we proved a transitivity property and a modularity condition for CRN bisimulation [6]. Transitivity states that the composition of two interpretations that are CRN bisimulations is a CRN bisimulation; thus a correct implementation of a correct implementation of a formal CRN is a correct implementation of the formal CRN. Modularity states that if two implementation CRNs are correct implementations of their respective formal CRNs and satisfy a certain modularity condition with respect to the species they have in common, then combining those implementation CRNs produces a correct implementation of the combined formal CRN. The condition is that any implementation species can "decompose" into species common to both implementation CRNs, ensuring the permissive condition for the combined CRN. Formal definitions of and theorem statements for CRN bisimulation, transitivity, and the modularity condition are given below.

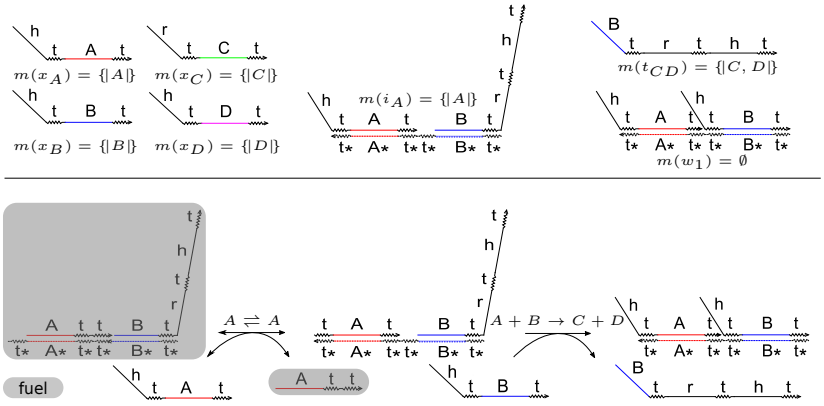

Fig. 1. Interpreted DSD implementation [11] of $A + B \to C + D$. Figure from [6].

Definition 1 (modified from [6]). *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 to states and reactions. If* $m(R') = m(P')$ *we write* $m(R' \to P') = \tau$.

Definition 2 (modified from [6]). *Given a formal CRN* $(\mathcal{S}, \mathcal{R})$ *and implementation CRN* $(\mathcal{S}', \mathcal{R}')$*, an interpretation* m *is a* CRN bisimulation *if:*

(i) ***Atomic Condition:*** *Every* $A \in \mathcal{S}$ *has an* $s_A \in \mathcal{S}'$ *with* $m(s_A) = A$.
(ii) ***Delimiting Condition:*** *For each* $r' \in \mathcal{R}'$*, either* $m(r') = \tau$ *or* $m(r') \in \mathcal{R}$.
(iii) ***Permissive Condition:*** *For* $r = R \to P \in \mathcal{R}$ *and implementation state* S' *with* $m(S') \supseteq R$*, an* r' *with* $m(r') = r$ *can occur after* τ *reactions from* S'.

This talk focuses on the use of transitivity and modularity to make CRN bisimulation tractable. I discuss some use cases of transitivity for multi-step verification. I discuss the use of modularity to break large CRNs into smaller pieces. When the division into modules is known, such as for systematic translation schemes [11, 3], verifying this division becomes much easier. This is used for efficient verification by the Nuskell compiler [2]; for an example 6-reaction formal CRN with an implementation for which the non-modular bisimulation algorithm [6] could not find an interpretation within 2 hours, the modular algorithm proved it correct in 8 seconds.

Lemma 1 (Transitivity [6]). *If* m_2 *is a CRN bisimulation from* $(\mathcal{S}'', \mathcal{R}'')$ *to* $(\mathcal{S}', \mathcal{R}')$ *and* m_1 *is a CRN bisimulation from* $(\mathcal{S}', \mathcal{R}')$ *to* $(\mathcal{S}, \mathcal{R})$*, then* $m = m_1 \circ m_2$ *is a CRN bisimulation from* $(\mathcal{S}'', \mathcal{R}'')$ *to* $(\mathcal{S}, \mathcal{R})$.

Definition 3 (Modularity Condition [6]). *Let* m *be a CRN bisimulation from* $(\mathcal{S}', \mathcal{R}')$ *to* $(\mathcal{S}, \mathcal{R})$*. Let* $\mathcal{S}'_0 \subset \mathcal{S}'$ *and* $\mathcal{S}_0 \subset \mathcal{S}$ *where* $y \in \mathcal{S}'_0 \Rightarrow m(y) \subset \mathcal{S}_0$*. We say* m *is a* modular interpretation *with respect to the* common (implementation and formal) species $\mathcal{S}'_0$ *and* $\mathcal{S}_0$ *if for all* $x \in \mathcal{S}'$ *there is a sequence of* τ *reactions* $x \stackrel{\tau}{\Rightarrow} Y + Z$ *where* $Y \subset \mathcal{S}'_0$ *and* $m(Z) \cap \mathcal{S}_0 = \emptyset$*, that is, all common formal species in* $m(x)$ *are extracted into (interpretations of) common implementation species.*

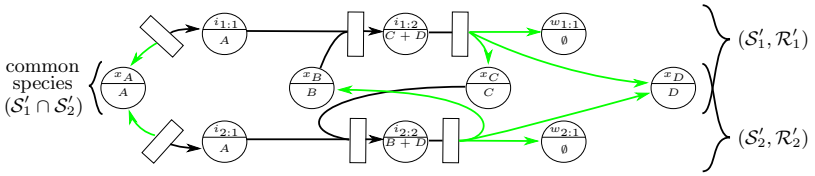

Fig. 2. An implementation CRN satisfying the modularity condition. Green arrows are reactions necessary to satisfy the modularity condition in Definition 3. Figure from [6].

Theorem 1 (Modularity [6]). *Let m_1 be a CRN bisimulation from $(\mathcal{S}_1', \mathcal{R}_1')$ to $(\mathcal{S}_1, \mathcal{R}_1)$ and m_2 be a CRN 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 with respect to some common implementation species $\mathcal{S}_0' \subset \mathcal{S}_1' \cap \mathcal{S}_2'$ and common formal species $\mathcal{S}_0 \subset \mathcal{S}_1 \cap \mathcal{S}_2$, then m is a CRN bisimulation from $(\mathcal{S}', \mathcal{R}')$ to $(\mathcal{S}, \mathcal{R})$ that is modular with respect to $\mathcal{S}_0'$ and $\mathcal{S}_0$.*

References

1. Angluin, D., Aspnes, J., Eisenstat, D.: A simple population protocol for fast robust approximate majority. Distributed Computing 21, 87–102 (2008)
2. Badelt, S., Shin, S.W., Johnson, R.F., Dong, Q., Thachuk, C., Winfree, E.: A general-purpose CRN-to-DSD compiler with formal verification, optimization, and simulation capabilities. In: Woods, D., Rondelez, Y. (eds.) DNA Computing and Molecular Programming. vol. 9818 of LNCS, pp. 232–248. Springer (2017)
3. Cardelli, L.: Two-domain DNA strand displacement. Mathematical Structures in Computer Science 23, 247–271 (2013)
4. Dobrinevski, A., Frey, E.: Extinction in neutrally stable stochastic Lotka-Volterra models. Physical Review E 85, 051903 (2012)
5. Grun, C., Sarma, K., Wolfe, B., Shin, S.W., Winfree, E.: A domain-level DNA strand displacement reaction enumerator allowing arbitrary non-pseudoknotted secondary structures. CoRR p. http://arxiv.org/abs/1505.03738 (2015)
6. Johnson, R.F., Dong, Q., Winfree, E.: Verifying chemical reaction network implementations: A bisimulation approach. Theoretical Computer Science (2018)
7. Lakin, M.R., Stefanovic, D., Phillips, A.: Modular verification of chemical reaction network encodings via serializability analysis. Theoretical Computer Science 632, 21–42 (2016)
8. Lakin, M.R., Youssef, S., Polo, F., Emmott, S., Phillips, A.: Visual DSD: a design and analysis tool for DNA strand displacement systems. Bioinformatics 27, 3211–3213 (2011)
9. Shin, S.W., Thachuk, C., Winfree, E.: Verifying chemical reaction network implementations: A pathway decomposition approach. Theoretical Computer Science (2017)
10. Soloveichik, D., Cook, M., Winfree, E., Bruck, J.: Computation with finite stochastic chemical reaction networks. Natural Computing 7, 615–633 (2008)
11. Soloveichik, D., Seelig, G., Winfree, E.: DNA as a universal substrate for chemical kinetics. Proceedings of the National Academy of Sciences 107, 5393–5398 (2010)