Biochemical reaction systems may be viewed as discrete event processes characterized by a number of states and state transitions. These systems may be modeled as state transition systems with transitions representing individual reaction events. Since they often involve a large number of interactions, it can be difficult to construct such a model for a system, and since the resulting state-level model can involve a huge number of states, model analysis can be difficult or impossible. Here, we describe methods for the high-level specification of a system using hypergraphs, for the automated generation of a state-level model from a high-level model, and for the exact reduction of a state-level model using information from the high-level model. Exact reduction is achieved through the automated application to the high-level model of the symmetry reduction technique and reduction by decomposition by independent subsystems, allowing potentially significant reductions without the need to generate a full model. The application of the method to biochemical reaction systems is illustrated by models describing a hypothetical ion-channel at several levels of complexity. The method allows for the reduction of the otherwise intractable example models to a manageable size. © 2012 American Institute of Physics. [http://dx.doi.org/10.1063/1.4758074]
be composed of millions of cells, each expressing a variable number of copies of a gene product associated with a gene regulatory network. Thus, stochastic simulation of a physiological process at a particular scale of interest potentially requires a large number of reaction trajectories per sample run, and therefore is generally not a suitable approach for use in multiscale modeling applications.

Although a large focus on the rule-based modeling of biochemical systems has been geared toward simulation using KMC algorithms, there exists a variety of powerful numerical approaches for the analysis of a HLM that have been developed and applied to many different problems (particularly in the field of operations) over the last few decades. These approaches may be divided into two major classes: (1) those designed to tolerate largeness; and (2) those designed to avoid largeness. Approaches to tolerate largeness are mainly centered upon use of symbolic data structures and associated algorithms, whereas approaches to avoid largeness are based on either using a HLM to generate only a part of the state-level model (e.g., truncation and/or probabilistic evaluation), or on using a HLM to identify structure in the associated state-level model (e.g., lumpability, invariant manifolds, product form solutions, etc.). A major advantage of all these approaches over stochastic simulation is that they allow direct computation of measures of interest (subject to an approximation error) such that rare events are much less of an issue. In addition, since the probability distribution of the states of a system is computed directly, incorporating a biochemical network into a multiscale physiological model is straightforward.

Here, we present an approach for largeness avoidance based on model structure identification from a HLM. Specifically, we present a formal description of a HLSM for models of biochemical systems composed of multiple interacting components, we describe how a HLM specified according to our method may be used to identify symmetries and independent subsystems, and we describe methods for using the identified structure to construct a reduced LTS describing the system. Finally, we illustrate the application of the method with examples.

II. RELATED METHODS

There exist a number of different classes of HLSMs for the specification of an LTS/CTMC, each with different advantages and disadvantages for particular application domains. Many techniques developed for applications in the fields of dependability and performability of computer and communication systems have been adopted in the field of computational biology. One popular approach is to use stochastic Petri nets (SPNs). A SPN consists of a set of places P, a set of transitions T, and a set of arcs A ⊆ (P × T) ∪ (T × P) between places and transitions. Each place contains an integer number of tokens, and the number of tokens in a place pi is its marking. The state of a system specified by a SPN is the vector of markings M = (p1, . . . , pn). In representing a well-mixed biochemical reaction network as an SPN, places represent molecular species, tokens represent individual molecules, and transitions represent molecular reaction events.

One drawback to representing a biochemical reaction network using a classical SPN is that it requires that all molecular species be specified and assigned to places. Since many biochemical systems give rise to a combinatorial blow-up of reachable species, enumerating the reachable species may be difficult or impossible. A second drawback is the well-mixed assumption required in the classical SPN representation. Because of this assumption, it is not possible to distinguish individual molecules on the basis of position or to model systems with compartments.

Extensions of SPN models using composition operations allow specification of hierarchical models helping to overcome the above limitations. Using these operators, molecular species can be specified componentwise such that the state of an individual domain of a species is represented by a much simpler atomic-level SPN. Then the global state of a system is given by the composition of the states of each of the individual atomic-level submodels. Interactions between components are accounted for by either synchronization of transitions or state-variable sharing. Using this approach, specification of a biochemical system does not require specification of the molecular species. In addition, replicates of submodels may represent components of different spatial orientations or in different compartments so that the well-mixed assumption need not hold. The structure of a compositional SPN model containing replicate components may give rise to symmetries that may be used to reduce the state space of a model. Such an approach is implemented in the Mòbius framework and has been used to model Ca2+ signaling complexes.

Another approach that avoids the need to enumerate molecular species in a biochemical system is that of rule-based modeling. In the field of biological systems modeling, the term rule-based modeling refers broadly to the class of specification methods that do not require the enumeration of all molecular species in a biochemical system (thus, compositional SPNs may be classified as a rule-based modeling approach under this definition). The methods leverage on the fact that the kinetics of elementary events operating on a molecular species usually do not depend on the species itself (i.e., the global state of a molecular complex), but rather on the limited context of a partially defined species. In this approach, reaction rules are generally specified as

\[
\text{if conditions met then reaction fires with rate } \lambda, 
\]

where conditions are local to a partially defined species.

There exist a variety of languages and tools to specify and construct a rule-based model. Although the rule-based modeling framework avoids complexity associated with system specification, the state-space of networks associated with a rule-based model is often very large due to the combinatorial nature of such a system. Therefore, methods to reduce the state-space of a rule-based model directly from the system specification are necessary to obtain tractable models. Previously, a method and theory for reducing a rule-based model specified in the Kappa language based on differential and stochastic fragments has been suggested. In this approach, a dependency analysis performed on an annotated contact map is used to analyze correlations between reaction sites in a molecular complex in order to identify a
minimal partitioning (i.e., a partitioning with the finest possible partition classes) of the sites. This partitioning is used to identify independent subsets of reaction sites which are used to define molecular fragments representing aggregations of molecular species.

One drawback with Kappa$^{3,19,20}$ and other similar rule-based modeling languages is the assumption that the order of agents and reaction sites does not matter. Thus, biochemical systems with spatial structure cannot be specified using these languages. Here, we propose a high-level rule-based specification method that allows for the specification of position-dependent reaction rules. Individual components (agents) in a system are specified in a component state-space graph as finite-state automata, and interactions between components occur as synchronization of transitions and/or as functional dependencies of the rate of transitions on the state of nonlocal components. These functional dependencies are specified as a hypergraph called a dependency graph. The dependency graphs are used to define a system characteristic graph which may be used to identify symmetries among system components, allowing for an automated reduction of the state-space associated with the model.

In a previous application of symmetry identification in compositional models with general components,$^{21}$ interactions between components are based on state-variable sharing. In that approach, a model composition graph is described in which shared state-variables represented as vertices of a graph are used to connect components in the system. The symmetries among components are then given as the automorphism group of the resulting simple graph. Our approach differs from that of Douglas Obal II et al.$^{21}$ in that it does not require a user to define shared state-variables, which can be cumbersome and error prone. Instead, connections between components follow directly from the dependency graphs used to define reaction rules. The dependency graphs are used to construct a hypergraph called a system characteristic graph, the automorphism group of which determines the component symmetries. Because this graph is constructed directly from the reaction rules, it can be mapped to from any rule-based specification language and does not require any additional input from the user to define.

III. SYSTEM SPECIFICATION AND GENERATION

A. State-level model

As described in the introduction, a biochemical reaction may be viewed as a discrete event occurring in a molecule or between sets of molecules such that the possible behaviors of a system may be described completely by a set of discrete states and transitions between states. Thus, the qualitative behavior of a biochemical reaction system may be described by a directed graph called an LTS, defined as

$$\mathcal{L} = (F, R, \Delta),$$

where

1. $F$ is a set of system states;
2. $R$ is a set of transition labels; and
3. $\Delta \subseteq F \times R \times F$ is the transition relation.

An example of a simple LTS is shown in Figure 1.

In addition, quantitative measures on a biochemical system may be obtained by mapping $\mathcal{L}$ to a homogeneous CTMC using a weighting function $W_0 : R \rightarrow \mathbb{R}_{\geq 0}$ in the case of constant transition rates, or to an inhomogeneous CTMC using a weighting function $W_r : R \rightarrow \mathbb{R}_{\geq 0}$ in the more general case of time-varying rate constants ($\mathbb{R}_{\geq 0}$ denotes the set of positive real-valued functions of time $t$). The resulting CTMC may be used to derive the master equation describing the time-evolution of the probabilities of the system occupying each of the states in $F$. This is done by imposing some order on the system states using an indexing function $J$ such that $J : I \rightarrow F$, where $I = \{1, 2, \ldots, |F|\}$ is the index set. The master equation may then be written as

$$\frac{dP}{dt} = AP,$$

where $P = [P_i]$ is a column vector defined such that $P_i$, under the restriction that $P_i > 0$ and $\sum_j P_j = 1$, is the probability of the system occupying state $J(i)$ at time $t$, and $A = [A_{ij}]$ is a matrix of transition rates given by

$$A_{ij} = \begin{cases} W(r, t) & \text{for } J(j) \xrightarrow{r} J(i) \in \Delta, \\ -\sum_{i\neq j} A_{ij} & \text{for } i = j, \text{ and} \\ 0 & \text{otherwise}, \end{cases}$$

where $W(r, t)$ is the weight (transition rate) associated with label $r \in R$ at time $t$.

The LTS and associated CTMC are analytical (state-level) representations of a system that may be used directly to analyze its behavior; however, they are far too cumbersome to be of much value in system specification. For that, we rely on a high-level representation of the system as described below.

B. High-level model

A HLSM allows specification of complex models in a compact and intuitive form with the purpose of simplifying the modeling process and avoiding modeling errors. A HLSM should contain all the information required to automatically generate a state-level model of the system. Our approach formulates a system as a set of components that may interact via both synchronizing transitions (transitions occurring simultaneously among a subset of components) and/or functional transitions (transitions in a subset of components with rates that depend on the state of disjoint subsets of components). These types of interactions are common in many biochemical processes, especially in signaling complexes, enzymes, ion channels, etc.
1. Component state space graph

The first step of specifying a biochemical reaction system using our approach is to specify the components. This is done using a structure called the component state space graph $C$, defined as

$$C = (S, Q),$$

where

1. $S = \{s_1, s_2, \ldots, s_n\}$ is a set of $n$ component states;
2. $Q = \{Q_1, Q_2, \ldots, Q_m\}$, with $Q_j = (X_j, Y_j, Z_j)$ for $j = 1, \ldots, m$, is a set of component transitions where each $X_j \subseteq S$ is a set of source states, each $Y_j$ is a mapping $Y_j: X_j \to S$ of source states to destination states, and each $Z_j$ is a mapping $Z_j: X_j \to \mathbb{N}$ such that $Z_j(s_i)$ is the multiplicity, or number of occurrences, of components in state $s_i$ required to enable transition $Q_j$.

(Here, the notation $X_j^{(1)}, X_j^{(2)}, \ldots, X_j^{(\leq n)}$, $Z_j^{(X_j^{(1)}}), Z_j^{(X_j^{(2)}}, \ldots, Z_j^{(X_j^{(\leq n))}}$, $Y_j^{(X_j^{(1)}}, Y_j^{(X_j^{(2)}}, \ldots, Y_j^{(X_j^{(\leq n))}}$ is used to represent a component transition $Q_j$.) The definition above implies a graph topology in which the component states $S$ represent the vertices and the component transitions $Q$ represent connections (arcs) of the graph. An important aspect of the above definition is that it allows specification of synchronized transitions. Any component transition $Q_j$ with more than one element in $X_j$ or with $Z_j(s_i) > 1$ for any $s_i \in X_j$ is a synchronizing transition.

Each component state in $S$ may be associated with a component type by finding the unique partition $\{S_1, S_2, \ldots, S_r\} \vdash S$ into $r$ maximal connected subsets (i.e., the coarsest partition such that every pair of vertices $s \in S_1$ are connected via some transition $Q_j \in Q$) and defining a function $T: S \to \{1, 2, \ldots, r\}$ such that $T(s_i) = i$ for all $s_i \in S_i$. Then, $T(s_i)$ is the component type associated with component state $s_i \in S$. The usefulness of this definition is described below. (The practical application of these definitions is illustrated below in the section Simple Example).

2. System domain

The next step in system specification is to define the numbers and positions of system components. The system domain $D = \{d_1, d_2, \ldots, d_p\}$ is an indexed set of $p$ component positions or places. The state of the system is described by a mapping of each element (position) in $D$ to a component state in $S$. That is, the system state is a function $f$ from $D$ to $S$. (For convenience, $f$ may be represented as a $p$-tuple.) The type of component assigned to each position $d_i \in D$ is given by $T(f_{init}(d_i))$, where $f_{init}$ is an initial system state. In general, the set $F$ of system states reachable from some initial system state $f_{init}$ is a proper subset of $S^D$ (where $S^D = \{f: D \to S\}$) such that the subset of possible component states assigned to $d_i$ is restricted to $S_{i,T(f_{init}(d_i))}. F$ may be further restricted by the existence of synchronizing and/or functional interactions.

3. Dependency graph

The final step in system specification is to define the functional dependencies of component transitions; in other words, to define a set of rules for mapping a particular component transition to a transition rate. These rules are given in the form of a directed hypergraph with labeled hyperarcs, which we refer to as the dependency graph $D(\alpha)$ of a transition rate $\alpha$, defined as

$$D(\alpha) = (D, E, L),$$

where

1. $D = \{d_1, d_2, \ldots, d_p\}$ is the system domain;
2. $E = \{E_1, E_2, \ldots, E_q\}$, with $E_y = (A_y, B_y)$ for $y = 1, \ldots, q$, is a set of hyperarcs with tail sets $A_y \in \mathcal{P}(D)$ and head sets $B_y \in \mathcal{P}(D)$, where $\mathcal{P}(D)$ is the power set of $D$;
3. $L = \{L_y \in \mathcal{P}(D): B_y \to \mathcal{P}(S)\}$ for $y = 1, \ldots, q$, is a set of labeling functions mapping elements of the head sets of each $E_y$ to elements (subsets of component states) in the power set of $S$.

The hyperarcs $E_y \in E$ represent dependencies of transitions in components pointed to by tail sets $A_y$ on the state of components pointed to by head sets $B_y$. All $A_y$ are singleton for local (non-synchronizing) transitions. Synchronizing transitions have $E$ with each $A_y$ containing $d_i$ for each component involved in the transition. Each element $d_i$ in each $B_y$ is labeled according to a subset of states required to enable a transition in the components of $A_y$ at a specific rate constant $\alpha$. Two $E_y \in E$ may have identical tail sets if there is more than one condition under which a specific rate constant applies for a particular $A_y$. The set $E$ is empty if there are no functional dependencies of the corresponding rate constant.

C. Generating the state-level model

The HLM of a system is used to generate the state-space graph $S$ of a system, which is a modified LTS describing the set of all system states $f \in S^D$ that are reachable from an initial system state $f_{init}$ (i.e., it is the reachability graph spanning the set of all $f \in S^D$ such that there exists a path starting from $f_{init}$ and ending at $f$). $S$ is defined as follows:

$$S = (D, F, R, \Delta, C, D, f_{init}),$$

where

1. $D = \{d_1, d_2, \ldots, d_p\}$ is the system domain;
2. $F \subseteq S^D$, with $S^D = \{f| f: D \to S\}$, is a set of system states;
3. $R = \{R_1, R_2, \ldots, R_m\}$ is a family of sets of transition labels, each $R_j$ being associated with a component transition $Q_j$;
4. $\Delta \subseteq F \times \bigcup_j R_j \times F$ is the transition relation;
5. $C$ is the component state-space graph;
6. $D = \{D(k, l)| (k, l) \in \bigcup_j R_j\}$ is the set of all dependency graphs; and
7. $f_{init} \in F$ is an initial system state.

An algorithm (modified from Algorithm 2.1 in Ref. 22 to account for the HLM) for generating the state-space graph $S$ is given below:
**ALGORITHM 1.** An algorithm for generating the full state-space graph.

1: Set \(unprocessed = \{f_\text{ini}\}\)
2: Set \(F = \{f_\text{ini}\}\)
3: Set \(\Delta = \emptyset\)
4: while \(unprocessed \neq \emptyset\) do
5: Select any \(f \in unprocessed\) and set \(unprocessed = unprocessed \setminus \{f\}\)
6: for all \(Q_j \in \{Q_j \in Q(X_j, Z_j) \subseteq f[D]\}\) do
7: for all \(D' \in \{D' \in \mathcal{P}(D) | f[D'] = (X_j, Z_j)\}\) do
8: Set \(f' = f\)
9: for all \(D \left( r^{(j)}_k \right) \in \left[ D \left( r^{(j)}_k \right) \in D | r^{(j)}_k \in R_j \right]\) do
10: if \(E = \emptyset\) then goto 13
11: for all \(E_i \in \{E_i \in E | A_E \neq D'\}\) do
12: if \(f(d) \in L_j(d) \forall d \in B_j\) then
13: for all \(d \in D'\) do
14: Set \(f' = f(d)\)
15: Set \(\Delta = \Delta \cup \{(f, r^{(j)}_k, f')\}\)
16: if \(f' \notin F\) then
17: Set \(unprocessed = unprocessed \cup \{f'\}\)
18: Set \(F = F \cup \{f'\}\)
19: return \(S = (D, F, \Delta, C, D, f_\text{ini})\)

---

In the last 6 and 7 of the above algorithm, the notation \(f[D]\) and \(f[D']\) denote the multiset image (rather than the usual set image) of \(D\) and \(D'\), respectively, under \(f\). A multiset is a pair \((A, m)\) where \(A\) is a set, or underlying set of elements, and \(m: A \rightarrow \mathbb{N}\) is a multiplicity function mapping elements in \(A\) to the number of occurrences of those elements in the multiset. The relations in lines 6 and 7 compare these multiset images with the multiset \((X_j, Z_j)\). In the following, the image \(f[A]\) of a set \(A\) under a function \(f\) should be understood as the multiset image when given in the context of a relation involving a multiset, and as the usual set image otherwise.

\(S\) may be mapped to a CTMC by assigning rates to the transition labels using a weighting function as described above for a general LTS.

---

**D. Simple example**

To illustrate the high-level specification of a simple biochemical system using our method, consider a reaction volume containing three molecules, two labeled A and one labeled B, as depicted in Figure 2(a). Molecules A can be in one of two conformational states, folded or unfolded, and when in the folded conformation can bind to molecule B. Molecule B has only one conformation and binds to folded molecules A. The possible molecular states and state transitions (reaction events) of all molecules (A and B) in this reaction volume are given by the component state space graph \(\mathcal{C}\) illustrated in Figure 2(b) with transitions tabulated in Table 1. There, the components are defined as molecule A and B binding sites and molecule A conformation. Because, there are two molecules A and one molecule B in the system, the system domain must contain five elements (two molecule A binding sites, two molecule A conformations, and one molecule B binding site), and each element in the domain is assigned to an appropriate component type by the initial state \(f_\text{ini}\).

This system contains functional transitions and therefore requires rules for the mapping of those transitions to transition rates in the form of dependency graphs. Specifically, the unfolding of a molecule A \((s_2 \rightarrow s_1)\) at rate \(r^{(2)}_1\) requires that it is unbound, and the binding of a molecule A to B \((s_5, s_5 \rightarrow s_4, s_6)\) at rate \(r^{(3)}_1\) requires that A is in the folded conformation. The dependency graphs \(\mathcal{D}(r^{(2)}_1)\) and \(\mathcal{D}(r^{(3)}_1)\) for

![Figure 2](image)

**FIG. 2.** Example biochemical reaction system. (a) An illustration of a small reaction volume containing two molecules of type A and one of type B. Molecules A can bind to molecules B, but only when in a folded conformation. (b) An illustration of the component state space graph \(\mathcal{C}\) associated with the reaction system depicted in panel (a). Circles represent component states \(S\) and arrows represent component transitions \(Q\) which are labeled with a multiplicity. The system domain \(D\) and an initial system state \(f_\text{ini}\) are also shown.

| TABLE 1. Component transitions \(Q\) of the component state-space graph \(\mathcal{C}\) for the example system illustrated in Figure 2. The mappings \(Y_j\) and \(Z_j\) are listed such that the first element in a set \(X_j\) maps to the first elements listed in \(Y_j\) and \(Z_j\), the second element listed in a set \(X_j\) maps to the second elements listed in \(Y_j\) and \(Z_j\), etc. Also given in the table are the sets of labels \(R_j\) associated with each transition \(j\). |
|---|---|---|---|---|
| \(j\) | \(X_j\) | \(Y_j\) | \(Z_j\) | \(R_j\) |
| 1 | \{1\} | \{2\} | \{1\} | \{r^{(1)}_1\} |
| 2 | \{2\} | \{1\} | \{1\} | \{r^{(2)}_1\} |
| 3 | \{5\}, \{3\} | \{4\}, \{6\} | \{1\}, \{1\} | \{r^{(3)}_1\} |
| 4 | \{4\}, \{6\} | \{3\}, \{5\} | \{1\}, \{1\} | \{r^{(4)}_1\} |
the rate constants $r_1^{(2)}$ and $r_1^{(3)}$ are shown in Figure 3 and tabulated in Table II. (In this system, it is possible to define a different component state space graph with course components so that the functional transitions described above become non-functional, or constant transitions.) The component state space graph, system domain, dependency graphs, and initial state completely describe the HLM for this simple system, and may be used to generate a state-space graph $\mathcal{S}$ using Algorithm I. The generated graph is shown in Figure 4(a).

IV. EXACT REDUCTION METHODS

A. Bisimilarity

A bisimulation of an LTS $\mathcal{L} = (F, R, \Delta)$ is an equivalence relation $\sim$ over $F$ such that for any $f_1, f_2 \in F$, $f_1 \sim f_2$ implies that $f_1$ and $f_2$ behave identically. In this case, $f_1$ and $f_2$ are said to be bisimilar.

There are two principle forms of bisimulation on an LTS, namely forward and backward bisimulation. Forward bisimulation requires equivalence of outgoing transitions. More specifically, letting $\Lambda(f_1, f_2)$ be the multiset of transition labels associated with all transitions from $f_1$ to $f_2$ in $\Delta$, $\sim$ is a forward bisimulation if for all $f_1, f_2 \in F$ such that $f_1 \sim f_2$, and for all equivalence classes $C \in F/\sim_f$, the following holds:

$$\biguplus_{f' \in C} \Lambda(f_1, f') = \biguplus_{f' \in C} \Lambda(f_2, f'),$$

where $\biguplus$ denotes the multiset union. (The multiset union of two multisets $(A, m_A)$ and $(B, m_B)$ is the multiset $(A \cup B, m_A + m_B)$.) This condition implies ordinary, or strong, lumpability of any CTMC to which the LTS is mapped. An example of an LTS that can be reduced via forward bisimulation (but not via backward bisimulation) is shown in Figure 5(a).

The lumping of the CTMC associated with an LTS reduced by forward bisimulation allows for the exact calculation of the probability distribution of the aggregated states and is valid for any initial probability distribution of the states in the original system. However, in general it is not possible to recover the full solution to the original system from the solution to the reduced system (that is, the lumping cannot be inverted).

By contrast, backward bisimulation requires equivalence of incoming transitions. More specifically, $\sim_b$ is a backward bisimulation if for all $f_1, f_2 \in F$ such that $f_1 \sim_b f_2$, and for all equivalence classes $C \in F/\sim_b$, the following holds:

1. $\biguplus_{f' \in C} \Lambda(f', f_1) = \biguplus_{f' \in C} \Lambda(f', f_2)$
2. $\biguplus_{f' \in F} \Lambda(f_1, f') = \biguplus_{f' \in F} \Lambda(f_2, f')$

Condition 2 above requires an equivalent multiset of exit transitions of backward bisimilar states (note that this condition is implicit in the definition of forward bisimulation). Condition 1 and 2 together imply exact lumpability of any CTMC to which the LTS is mapped.

| Table II. Dependency graphs for the rate constants of the example system illustrated in Figure 2. |

| $D(r_1^{(2)})$ | $y$ | $A_y$ | $B_y$ | $L_y$ |
|-----------------|-----|-------|-------|-------|
| $D(r_1^{(2)})$ | 1   | 1     | (3)   | (3)   |
|                 | 2   | 1     | (3)   | (3)   |
| $D(r_1^{(3)})$ | 1   | 3.5   | (1)   | (2)   |
|                 | 2   | 4.5   | (2)   | (2)   |

Legend:

- $\triangleright := \{s_i\}$
- $\blacksquare := \{s_j\}$
- $\circ := A$ conformation
- $\lozenge := A$ binding site
- $\blacklozenge := B$ binding site

FIG. 3. Dependency graphs for the rate constants of the example system illustrated in Figure 2.

FIG. 4. State-space graphs associated with the system illustrated in Figure 2.

(a) Full state-space graph $\mathcal{S}$. Circles represent the system states $F$ and arrows represent the system transitions $\Delta$. (b) Reduced state-space graph $\mathcal{S}$. Circles represent system state equivalence class representatives $\hat{F}$ and arrows represent the transitions $\Delta$ between equivalence classes. The coefficients of the rate labels are the multiplicities of the transitions.
B. Symmetry reduction method

As with forward bisimulation, the lumping of the CTMC associated with an LTS reduced by backward bisimulation allows for the exact calculation of the probability distribution of the aggregated states. However, the solution is only valid when the initial probability distribution of the states in the original system satisfies the condition that for all \( f_1, f_2 \in F \) such that \( f_1 \sim_b f_2 \), \( P(f_1) = P(f_2) \), where \( P(\cdot) \) is the initial occupancy probability associated with a state. If this condition is not satisfied, the solution will not be valid. One advantage of backward bisimulation over forward bisimulation is that all states belonging to a given equivalence class will have the same occupancy probability so that it is possible to compute the solution to the full original system directly from the solution to the reduced system.

An efficient algorithm exists for computing the coarsest forward (or backward) bisimulation of an LTS; however, this algorithm operates on the full state-space and therefore cannot be used to reduce a system a priori (i.e., before state-space generation). It is possible to compute a bisimulation directly from a HLM by considering states that are both forward and backward bisimilar. This is the basis of the symmetry reduction method described below. An example of an LTS containing states that are both forward and backward bisimilar, and can therefore be reduced via the symmetry reduction method, is shown in Figure 5(c).

In other words, a symmetry is a permutation of \( F \) which preserves \( \Delta \). Since symmetries are permutations, it follows that the set of all symmetries of a state-space graph \( S \) forms an automorphism group, denoted \( \text{Aut}(S) \), under the function composition operation. \( \text{Aut}(S) \) induces an equivalence relation \( \sim \) in \( F \) such that for any two states \( f_1, f_2 \in F \), \( f_1 \sim f_2 \) implies that \( \pi(f_1) = f_2 \) for some \( \pi \in \text{Aut}(S) \). Furthermore, \( f_1 \sim f_2 \) implies both forward and backward bisimilarity of \( f_1 \) and \( f_2 \).

Symmetries can arise in biochemical systems containing replicated components. In the case that the well-mixed assumption applies for a particular subset of components (that is, the position of the components in the system has no effect on dynamics), every permutation of that subset induces a symmetry in the state-space graph. If however the transition rate in one component depends on the state and position of neighboring components, not all permutations of the components will induce a symmetry in the state-space graph, just the subgroup of those permutations corresponding to spatial symmetries among the components.

It is possible to find \( \text{Aut}(S) \) without generating the full state-space graph (which may be impractical or impossible for large models). This is done by identifying a system description level group \( G \) of symmetries from the HLM along with a group action \( h : G \rightarrow \text{Sym}(F) \) of \( G \) on \( F \) satisfying

\[
h(g) = \pi \Rightarrow f(\pi(d)) = \pi(f)(d),
\]

for all \( g \in G, \pi \in \text{Aut}(S), f \in F \) and \( d \in D \). \( G \) can be readily identified from the set of dependency graphs \( D \) according to the following procedure:

1. For each dependency graph \( D(r_k^{ij}) \), assign a unique label (e.g., \( r_k^{ij} \)) to its hyperarcs.
2. Define a new graph, called the system characteristic graph, as \( G = (V, U) \), where
   i. \( V = D \) and
   ii. \( U = \bigcup_{D(v) \in D} E_k^{(ij)} \).
3. Label each \( d \in V \) by \( T(f_{m0}(d)) \).
4. Find the automorphism group of \( G \).

The characteristic graph describes all dependencies among all components in a system and is used to determine which permutations of the system domain (in other words, which rearrangements of the components) leave system dynamics invariant. The group of all these rearrangements is
the automorphism group Aut(\(G\)) of the graph. Note that non-
equivalent components (i.e., components of different type \(T\)) 
are labeled differently (see step 3 in the above procedure) so 
that they cannot be mapped to one another; thus the calcu-
lated automorphism group is actually a subgroup of the full 
automorphism group of the component labels. More specific-
ally, \(G\) is the subgroup stabilizing the component labels. The characteristic 
graph \(G\) of the example shown in Figure 2 is illustrated in 
Figure 6.

Algorithm II

An algorithm for generating the reduced state-space graph.

1: Set unprocessed = \{\(f_{\text{init}}\)\}
2: Set \(\tilde{F} = \{f_{\text{init}}\}\)
3: Set \(\bar{\Delta} = \varnothing\)
4: while unprocessed \(\neq \varnothing\) do
5: Select any \(f \in \text{unprocessed}\) and set unprocessed 
   = unprocessed\(\setminus\{f\}\)
6: for all \(Q_i \in \{Q_j \in \mathcal{Q}(X_j, Z_j) \subseteq \mathcal{D}\}\) do
7: for all \(D' \in \mathcal{D}' \in \mathcal{P}(D)|\{D'\} = \{X_j, Z_j\}\} do
8: Set \(f' = f\)
9: for all \(D\left(r_{ij}^{(k)}\right) \in \mathcal{D}\left(r_{ij}^{(k)}\right) \in \mathcal{R}_{ij}\) do
10: if \(\mathcal{E} = \varnothing\) then go to 13
11: for all \(E_r \in \mathcal{E} | E_r = f\{D'\}\} do
12: if \(\mathcal{E} = \varnothing\) then \(f' = f\) then
13: Set \(f' = f\)
14: do
15: \(d\tilde{P} = \mathcal{A}\tilde{P};\)
16: while \(d\tilde{P} \neq 0\) do
17: \(\delta = \mathcal{J}(j) \to \mathcal{J}(i) \in \bar{\Delta},\)
18: \(\sum_{i \neq j} \tilde{A}_{ij} = \mathcal{A}_{ij}\)
19: \(\mathcal{A}_{ij} = \begin{cases} 
M(\delta) \cdot W(r, t) & \text{for } \delta = \mathcal{J}(j) \to \mathcal{J}(i) \in \bar{\Delta}, \\
0 & \text{otherwise,} 
\end{cases}\)
20: \(\mathcal{J}(i) \to \mathcal{J}(i) \in \bar{\Delta},\)
21: \(\bar{\Delta} = \{A_{ij}\}\) is a matrix of transition rates 
given by
22: \(\tilde{P} = [\tilde{P}_i]\) is a column vector defined such that \(\tilde{P}_i\), 
under the restriction that \(\tilde{P}_i > 0\) and \(\sum_i \tilde{P}_i = 1\), is the 
probability of the system occupying a state in equivalence class 
\(\mathcal{J}(i)\) at time \(t\), and \(\bar{\Delta} = \{A_{ij}\}\) is a matrix of transition rates 
given by
23: \(\mathcal{A}_{ij} = \begin{cases} 
M(\delta) \cdot W(r, t) & \text{for } \delta = \mathcal{J}(j) \to \mathcal{J}(i) \in \bar{\Delta}, \\
0 & \text{otherwise,} 
\end{cases}\)
24: \(\mathcal{J}(i) \to \mathcal{J}(i) \in \bar{\Delta},\)
25: \(\mathcal{A}_{ij} = \begin{cases} 
M(\delta) \cdot W(r, t) & \text{for } \delta = \mathcal{J}(j) \to \mathcal{J}(i) \in \bar{\Delta}, \\
0 & \text{otherwise,} 
\end{cases}\)
26: return \(\tilde{S} = (D, \tilde{F}, \tilde{R}, \tilde{\Delta}, M, C, D, f_{\text{init}})\)
the occupancy probability \( P_i \) of any state \( J(i) \) in \( J(j) \) is given by

\[
P_i = \frac{P_j}{|J(j)|},
\]

where \(|J(j)|\) denotes the cardinality of equivalence class \( J(j) \). (This formula only holds if the initial probability of all states of a given equivalence class are equivalent. If this condition is not satisfied, the formula will not be valid for all time \( t > 0 \).)

C. Decomposition of independent subsystems

It may be possible to achieve an exact reduction of Eq. (1) beyond what can be achieved through the symmetry reduction technique alone if there exists independence among subsystems of a system. Independence of subsystems occurs when there are no interactions, neither functional nor synchronizing, between disjoint subsets of system components. Let \( \delta \subseteq D \) be a subsystem with state-space graph given by

\[ S_\delta = (\delta, F_\delta, L_\delta, A_\delta, C, D, f_{\text{init}}). \]

If \( D \) can be partitioned into a set \( \{\delta_1, \delta_2, \ldots, \delta_n\} \) of subsystems with state spaces \( \{F_{\delta_1}, F_{\delta_2}, \ldots, F_{\delta_n}\} \) such that all components in subsystem \( \delta_i \) are independent from (i.e., have no functional or synchronizing interactions between) all components in subsystem \( \delta_j \) for \( j \neq i \), then the solution to the master equation associated with the CTMC generated on \( D \) can be expressed exactly as

\[
P = \prod_{i=1}^{n} P_i,
\]

where \( P_i \) is a probability distribution associated with subsystem \( i \). (Note that Eq. (6) holds for both the steady-state and transient solutions provided that initial transients have decayed.) It is straightforward to test for independence of subsystems using the HLM (see Appendix C for a procedure and associated algorithms).

The symmetry reduction technique and the reduction by decomposition of independent subsystems are not mutually exclusive. A general approach to combine the methods is to first identify independent subsystems, and then apply symmetry reduction during the state-space generation for each subsystem using Algorithm II. Applying this approach allows for potentially significant reductions in models of biochemical reaction systems \( a \) \( priei \), as illustrated by the examples given below.

V. EXAMPLES

A. Example 1: Reduction scalability in a simple example

As a first example, we illustrate how our techniques may be used to alleviate the combinatorial state-space explosion problem in a class of models taken from Ref. 19, and we compare the reductions achieved on this system in that study using the method of stochastic fragments with the reductions achieved using the present methods.

This example is similar to the simple example illustrated in Figure 2 in that it consists of a biochemical reaction system involving two types of particles labeled A and B which can bind to (form complexes with) one another. Both types of particles can be in an activated or an inactivated state. Two cases are considered: one in which the binding rate depends on the activation state of the particles and one in which it does not. A high-level model for the first case specified using our framework is illustrated in Figure 7 and Table III.

Expressions were derived for the number of states of such a system for three different levels of abstraction: identified (distinguishable) particles; anonymous (indistinguishable) particles; and anonymous fragments.19 The number of states assuming identified particles is given by

\[
2^m \cdot 2^n \cdot \sum_{k=0}^{\min(n, m)} k! \cdot C^m_k \cdot C^n_k,
\]

where \( m \) and \( n \) are the total number of particles A and particles B, respectively, \( k \) is the number AB complexes, and \( C^m_k = m! / (m-k)! k! \). The summation in Equation (7) is over all possible number of complexes.

The first level of abstraction involves de-identifying particles by assuming that any two states that can be mapped to one-another by a reordering of particle indices are equivalent. (Note that this abstraction is only sound if it is assumed that the system is well-mixed.) Assuming anonymous
TABLE III. Component transitions $Q$ associated with the component state-space graph $C$ shown in Figure 7. The mappings $Y$ and $Z$ are listed such that the first element in a set $X_j$ maps to the first elements listed in $Y_j$ and $Z_j$; the second element listed in a set $X_j$ maps to the second elements listed in $Y_j$ and $Z_j$, etc. Also given in the table are the sets of labels $R_j$ associated with each transition $j$.

| $j$ | $X_j$ | $Y_j$ | $Z_j$ | $R_j$ |
|-----|-------|-------|-------|-------|
| 1   | {1}   | (2)   | (1)   | $[r_1^{(1)}]$ |
| 2   | {2}   | (1)   | (1)   | $[r_1^{(2)}]$ |
| 3   | {7}   | (8)   | (1)   | $[r_1^{(3)}]$ |
| 4   | {8}   | (7)   | (1)   | $[r_1^{(4)}]$ |
| 5   | {1.7} | (3.9) | (1.1) | $[r_1^{(5)}]$ |
| 6   | {3.9} | (1.7) | (1.1) | $[r_1^{(6)}]$ |
| 7   | {1.8} | (5.10)| (1.1) | $[r_1^{(7)}]$ |
| 8   | {5.10}| (1.8) | (1.1) | $[r_1^{(8)}]$ |
| 9   | {2.7} | (4.11)| (1.1) | $[r_1^{(9)}]$ |
| 10  | {4.11}| (2.7) | (1.1) | $[r_1^{(10)}]$|
| 11  | {2.8} | (6.12)| (1.1) | $[r_1^{(11)}]$|
| 12  | {6.12}| (2.8) | (1.1) | $[r_1^{(12)}]$|
| 13  | {3.9} | (4.11)| (1.1) | $[r_1^{(13)}]$|
| 14  | {4.11}| (3.9) | (1.1) | $[r_1^{(14)}]$|
| 15  | {5.10}| (6.12)| (1.1) | $[r_1^{(15)}]$|
| 16  | {6.12}| (5.10)| (1.1) | $[r_1^{(16)}]$|
| 17  | {3.9} | (5.10)| (1.1) | $[r_1^{(17)}]$|
| 18  | {5.10}| (3.9) | (1.1) | $[r_1^{(18)}]$|
| 19  | {4.11}| (6.12)| (1.1) | $[r_1^{(19)}]$|
| 20  | {6.12}| (4.11)| (1.1) | $[r_1^{(20)}]$|

particles, the number of states is

$$\sum_{k=0}^{\min(n,m)} (m-k+1) \cdot (n-k+1) \cdot C_3^{k+3}. \tag{8}$$

This number of states is obtained by applying symmetry reduction to the model illustrated in Figure 7. There are no functional dependencies among the components, and thus the symmetries in this case consist of all possible permutations of the individual particles of which there are $m!n!$. In the case that the binding rate of particles depends on activation state, it is not possible to achieve a further abstraction using either our symmetry reduction approach or the stochastic fragments approach.\(^{19}\)

In the case that the binding rate does not depend on the activation state of the particles, we can construct a simpler model of the system as shown in Figure 8 and Table IV. In this case, it is possible using the stochastic fragments approach to abstract the system into anonymous fragments, the number of which is given by\(^{19}\)

$$\sum_{k=0}^{\min(n,m)} (m-k+1) \cdot (n-k+1) \cdot (k+1)^3 | k \in \mathbb{N}. \tag{9}$$

Our algorithm goes beyond this. Again, in this case there are no functional dependencies between system components and the system characteristic graph is disconnected. However, there is more symmetry in this case than in the first case; in this case we have components corresponding to both activation and binding state of the two types of particles, and the symmetries consist of the group of all permutations of components of a given type of which there are $(m!)^2 \cdot (n!)^2$. Applying our symmetry reduction technique to the model yields

$$(m+1)(n+1)(\min(m,n)+1) \tag{10}$$

system states, a substantial reduction compared to the number of fragments given by Eq. (9).

Since there exists independence among subsets of components in this case, we can go one step further by decomposing the independent subsystems. All components corresponding to activation state are independent from one another. However, the components corresponding to binding state are correlated due to synchronizing interactions among the components and therefore cannot be decomposed. We can construct models for three independent subsystems and use these models to obtain an exact solution to the full model using Eq. (6). The three subsystems consist of: one subsystem describing the activation state of a particle $A$; one subsystem

![Diagram](image_url)

FIG. 8. Component state-space graph $C$ for Example 1 case #2 (binding rate is independent from activation state). Circles represent component states $S$ and arrows represent component transitions $Q$ labeled with multiplicities. The system domain $D$ and an initial system state $f_{init}$ are also shown.

TABLE IV. Component transitions $Q$ associated with the component state-space graph $C$ shown in Figure 8. The mappings $Y_j$ and $Z_j$ are listed such that the first element in a set $X_j$ maps to the first elements listed in $Y_j$ and $Z_j$, the second element listed in a set $X_j$ maps to the second elements listed in $Y_j$ and $Z_j$, etc. Also given in the table are the sets of labels $R_j$ associated with each transition $j$.

| $j$ | $X_j$ | $Y_j$ | $Z_j$ | $R_j$ |
|-----|-------|-------|-------|-------|
| 1   | {1}   | (2)   | (1)   | $[r_1^{(1)}]$ |
| 2   | {2}   | (1)   | (1)   | $[r_1^{(2)}]$ |
| 3   | {3}   | (4)   | (1)   | $[r_1^{(3)}]$ |
| 4   | {4}   | (3)   | (1)   | $[r_1^{(4)}]$ |
| 5   | {5.7} | (6.8) | (1.1) | $[r_1^{(5)}]$ |
| 6   | {6.8} | (5.7) | (1.1) | $[r_1^{(6)}]$ |
(a) FIG. 9. Reduction scalability in Example 1. The number of states in case #1 assuming identified particles (solid black) and anonymous particles (dashed black), and the number of states in case #2 obtained by applying stochastic fragments (dashed grey) and symmetry reduction techniques with (dotted black) and without (dotted grey) applying independent subsystem decomposition are plotted against particle number \( n \) for \( m = n \).

(b) FIG. 9. Reduction scalability in Example 1. The number of states in case #1 assuming identified particles (solid black) and anonymous particles (dashed black), and the number of states in case #2 obtained by applying stochastic fragments (dashed grey) and symmetry reduction techniques with (dotted black) and without (dotted grey) applying independent subsystem decomposition are plotted against particle number \( n \) for \( m = n \).

describing the activation state of a particle B; and one subsystem describing the binding state of the \( m + n \) particles in the system. Applying symmetry reduction to each of the three submodels, we can obtain a reduced model with a total state-space of size:

\[
\min(m, n) + 5
\]

for \( m, n \geq 1 \). Thus, by applying both the symmetry reduction and independent subsystem decomposition techniques to the system, we are able to obtain exact solutions for the stochastic dynamics of a system using a model of dimension that scales (sub-)linearly with the number of particles in the system. A summary of the reductions achieved for this system using the method of stochastic fragments and using the methods described here is illustrated in Figure 9.

B. Example 2: Single ion channel

We next consider an example consisting of a cluster of four identical Na\(^+\) channels arranged in a ring configuration. Suppose that we first wish to develop a single channel model in which the individual channels are composed of one \( \alpha \) subunit and two \( \beta \) subunits, each of which may reside in one of the following possible conformations: inactive; permissive; or open. In addition, the \( \beta \) subunits bind to a sodium ion and can also form dimers by binding to one another. We assume that Na\(^+\) binds to the intracellular side of the channel.

We begin by modeling the \( \beta \) subunit, which we divide into three functional domains: a Na\(^+\) binding domain with two possible states (unbound and bound); a \( \beta \) subunit binding domain with two possible states (unbound and bound); and a hinge domain with three possible states (inactive, permissive, open). Thus, we require three system components for each \( \beta \) subunit. Because the \( \alpha \) subunit does not have any binding domains, it may be modeled using a single component with three possible states (inactive, permissive, and open). The component state-space graph \( C \) for the channel may be formulated as shown in Figure 10 and Table V. We assign the initial system state \( f_{\text{init}} \) of the seven system components as \((1,1,4,4,6,6,8)\), i.e., both \( \beta \) subunits and the \( \alpha \) subunit in the inactive state and both binding domains of the two \( \beta \) subunits in the unbound state. According to \( C \), there are 10 possible component transitions for the Na\(^+\) channel, four of which represent synchronizing transitions (i.e., for \( j = 2, 3, 7, \) and 8), and five of which represent functional transitions (i.e., for \( j = 2, 5, 6, 7, \) and 8).

The functional transitions require rules for their mapping to transition labels (rates), given in the form of dependency

![System Domain and Initial State](image)

Legend:

| Component States | Component Types |
|------------------|----------------|
| \( s_j := \) \( \beta \) inactive | \( \alpha \) hinge domain |
| \( s_2 := \) \( \beta \) permissive | \( \beta \) hinge domain |
| \( s_3 := \) \( \beta \) open | \( \beta -Na^+ \) binding domain |
| \( s_4 := \) \( \beta -Na^+ \) unbound | \( \beta -Na^+ \) binding domain |
| \( s_5 := \) \( \beta -Na^+ \) bound | \( \beta -Na^+ \) binding domain |
| \( s_6 := \) \( \beta -\beta \) unbound | \( \beta -\beta \) binding domain |
| \( s_7 := \) \( \beta -\beta \) bound | \( \beta -\beta \) binding domain |
| \( s_8 := \) \( \alpha \) inactive | \( \alpha \) hinge domain |
| \( s_9 := \) \( \alpha \) permissive | \( \alpha \) hinge domain |
| \( s_{10} := \) \( \alpha \) open | \( \alpha \) hinge domain |

System Domain:
\[ D = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7\} \]

Initial State:
\[ f_{\text{init}} = (s_1, s_4, s_4, s_6, s_6, s_8) \]
graphs. The dependency graphs are listed in Table VI. The first dependency graph \( \mathcal{D}(r^{(5)}) \) encodes the rule that the two \( \beta \) subunits must not be associated in order for the channel to open. \( \mathcal{D}(r^{(1)}) \) and \( \mathcal{D}(r^{(5)}) \) encode the rule that \( \text{Na}^{+} \) binds to a \( \beta \) subunit at rate \( r^{(5)} \) if the channel is not open and at rate \( r^{(2)} \) if the channel is open (the biophysical mechanism being that the local \( \text{Na}^{+} \) concentration is elevated when the channel is open). \( \mathcal{D}(r^{(6)}) \) and \( \mathcal{D}(r^{(8)}) \) encode the rule that \( \text{Na}^{+} \) dissociates with the rate \( r^{(6)} \) if the \( \beta \) subunit binding domain is unbound and with the rate \( r^{(2)} \) if the \( \beta \) subunits are in a dimer configuration (i.e., the affinity for \( \text{Na}^{+} \) depends on whether the \( \beta \) subunits are in a dimer configuration or not). \( \mathcal{D}(r^{(7)}) \) encodes the rule that the \( \beta \) subunits can only form dimers when the channel is closed. Finally, \( \mathcal{D}(r^{(8)}) \), \( \mathcal{D}(r^{(8)}) \), and \( \mathcal{D}(r^{(8)}) \) encode the rule that the dissociation rate of the \( \beta \) subunit dimer depends on whether \( \text{Na}^{+} \) is bound to one, both, or none of the \( \beta \) subunits.

VERNAN, our MATLAB implementation of the method, can be used as a convenient platform for specifying and constructing the model described above (codes are available upon request). MATLAB codes for specifying and constructing the model using VERNAN are given in the supplementary material. In addition, a mat-file (Example1.mat) containing the model specifications is included in the VERNAN package. VERNAN gives the option of generating the full state-space graph (using Algorithm I) or the reduced state-space graph (using Algorithm II). Generating the full state-space graph yields a model with 68 system states and 400 system transitions. (Recall that the full system state-space includes only those states that are reachable from the initial state \( f_{\text{init}} \) which is why the full state-space size is much less than the product \( 3^{2} \times 2^{2} \times 2^{3} \times 3 = 432 \) of the number of states of each component.) On the other hand, generating the reduced state-space graph generates a model with 39 states and 178 transitions, a substantial reduction compared to the full state-space. There are eight symmetries in this example consisting of the group of permutations generated by independently swapping components 1 with 2, 3 with 4, and 5 with 6. Because the characteristic graph of this system is connected (i.e., there are dependencies between each component, and no two subsets of components are independent), it is not possible to achieve a further reduction of this system using the techniques described here.

The solution to the reduced model can be used to compute an exact solution to the full model using a differential equation of order 39, rather than 68 (or 432), which would be required without applying the reduction technique. To illustrate this, we assign random values in the interval (0,10) to 12 of the 14 rates in the model. We assign time-varying sinusoidal functions to the remaining two rates \( r^{(5)} \) and \( r^{(5)} \) such that \( r^{(5)} \) \( r^{(5)} \) \( f \), where \( r^{(6)} \) is set to 5 second\(^{-1} \), \( \omega \) is set to 2\( \pi \) radians/second, and values for \( k_{s} \) and \( k_{b} \) are chosen randomly in the interval (0,10) with the constraint that \( k_{b} \) be greater than \( k_{s} \).

Next we numerically solve the master equations associated with the full and reduced systems (i.e., equations (1) and (3), respectively) using MATLAB’s built-in solver ode15s using an initial probability of \( P_{i} = 0 \) for all \( i \) such that \( \mathcal{J}(i) \in \mathcal{F}_{\text{init}} \) and \( P_{i} = 1 \) for the \( i \) satisfying \( \mathcal{J}(i) = f_{\text{init}} \) in the full system, and of \( \tilde{P}_{i} = 0 \) for all \( i \) such that \( \tilde{f}(i) \in \tilde{\mathcal{F}} \setminus \mathcal{F}_{\text{init}} \) and \( \tilde{P}_{i} = 1 \) for the \( i \) satisfying \( \tilde{f}(i) = f_{\text{init}} \) in the reduced system. (MATLAB codes are given in the supplementary material.) If we are interested in the open probability of the channel, we identify all states \( f \in \mathcal{F} \) and all states \( \tilde{f} \in \tilde{\mathcal{F}} \) such that the \( \alpha \) subunit and two \( \beta \) subunits are in the open conformation. There are four such \( f \) in the full model, given by \( \{3,3,4,4,6,6,10\}, \{3,3,4,5,6,6,10\}, \{3,3,5,4,6,6,10\}, \{3,3,5,5,6,6,10\} \), and three such \( \tilde{f} \) in the reduced model, given by \( \{3,3,4,4,6,6,10\}, \{3,3,4,5,6,6,10\}, \{3,3,5,5,6,6,10\} \).

The macroscopic open probability is then simply the sum of the occupancy probabilities of these three (four) microscopic open states of the reduced (full) model. Plots of the time-evolution of the macroscopic open probability computed using the full and reduced models are shown in Figure 11(a), demonstrating the equivalence of the two solutions.

Suppose now that we are interested in the occupancy probability of the microscopic state \( \{3,3,4,4,6,6,10\} \) and wish

| \( j \) | \( X_{j} \) | \( Y_{j} \) | \( Z_{j} \) | \( R_{j} \) |
|---|---|---|---|---|
| 1 | \{1\} | \{2\} | \{1\} | \{r^{(1)}\} |
| 2 | \{2,9\} | \{3,10\} | \{2,1\} | \{r^{(5)}\} |
| 3 | \{3,10\} | \{2,9\} | \{2,1\} | \{r^{(5)}\} |
| 4 | \{2\} | \{1\} | \{1\} | \{r^{(5)}\} |
| 5 | \{4\} | \{5\} | \{1\} | \{r^{(5)},r^{(5)}\} |
| 6 | \{5\} | \{4\} | \{1\} | \{r^{(6)},r^{(6)}\} |
| 7 | \{6\} | \{7\} | \{2\} | \{r^{(7)}\} |
| 8 | \{7\} | \{6\} | \{2\} | \{r^{(8)}\} |
| 9 | \{8\} | \{9\} | \{1\} | \{r^{(9)}\} |
| 10 | \{9\} | \{8\} | \{1\} | \{r^{(10)}\} |
C. Example 3: Ion channel complex with fine components

In the next hypothetical example, the Na⁺ channels modeled above are not physiologically observed alone, but rather as a complex of four such channels arranged in a ring configuration with each neighbor spaced equally apart.

Whereas there were seven components in the individual channel model, there are now 28 (7 × 4) components in the complex model. We assign the initial state \( f_{\text{init}} \) of the complex to be \((1,1,1,1,1,1,4,4,4,4,4,4,6,6,6,6,6,6,6,6,6,6,8,8,8,8)\); that is, all four channels are in the inactive, unbound, dissociated state.

We assume that the four channels are close enough together that they are coupled through local Na⁺ concentration elevations caused by the opening and closing of the individual Na⁺ channels. Assuming that local Na⁺ concentrations reach a steady-state on a time-scale much faster than Na⁺ channels openings and closings, we can assume that this coupling is instantaneous. These are the same assumptions that are made in models of intracellular calcium release channel (ryanodine receptor) clusters. This coupling depends on the spatial orientation of the channels, and because of our assumed orientation of the channels, there exist symmetries that can be exploited when constructing the Na⁺ channel complex model.

The component state-space graph \( C \) for the complex model is the same as the individual channel model. However, there are now more rate constants associated with transition \( Q_5 \), i.e., the binding of a sodium ion to a β subunit. Recall that in the single channel model, there were two possible binding rates depending on whether the channel was open or closed. In the channel complex model, there are 12 possible binding rates which depend on the open/closed configuration of all four channels in the complex. Consider the case where a sodium ion binds to a β subunit of the channel labeled #1 in Figure 12. The 12 binding rates then depend on the open/closed configurations of all four channels as depicted in the diagram of Figure 12.

Another difference is that transition \( Q_3 \) is now a functional transition (recall that it was a constant transition in the single channel model). \( Q_3 \) is a synchronizing transition involving two β subunits and one α subunit. The requirement that \( Q_3 \) be a functional transition in the complex model is a consequence of the fact that multiple channels may be in the open state simultaneously, allowing for \( Q_3 \) to occur between
multiple combinations of components (whereas in the single channel model there was only one possible combination of components that could participate in \( Q_1 \)). However, in our model, we assume that \( Q_1 \) can only occur between subunits of the same channel, and this requires a functional dependency to restrict the set of possible combinations of components that may be involved in the transition. Note that all synchronizing transitions in the complex model require a functional dependency for this reason, but all other synchronizing transitions were already functional transitions in the single channel model. Thus, it might have been easier to overlook this requirement for \( Q_1 \) compared to the others.

Since the system domain \( D \) for the complex model is different from the single channel model, new rules, along with a new set of dependency graphs \( D \), must be specified. Because the dependency graphs are rather complicated, we do not discuss them in detail here. However, a table listing the graphs can be found in the supplementary material. In addition, a .mat-file containing all specifications of the ion channel complex model, including the dependency graphs, as well as VERNAN outputs, is included in the VERNAN toolbox package (Example2.mat). Generating the reduced-state space graph using the VERNAN tool generates a model with 304 590 system states and 5 414 760 system transitions, representing an over 70-fold reduction in state-space when compared to the expected state-space size \( (68^4 = 21 381 376) \) of the full model. There are 32 768 symmetries in this example consisting of the group of permutations generated by rotating the ring, flipping the ring, and independently swapping the components corresponding to the \( \beta \) subunits of each channel. It is not possible to achieve a larger reduction, as the characteristic graph of the model is connected. (For this example, it is not possible to generate the full model on a 32-bit system because the sparse transition matrix is too large to store in the available address space.)

D. Example 4: Ion channel complex with coarse components

The large number of components in the ion channel complex model can make model specification a rather tedious task. However, there is flexibility in defining system components, which may be made as large or small as desired. In the preceding examples, the components were essentially specified to be as fine as possible (for the most part, constituting elementary reactions). In the case of the ion channel complex, it would have been possible to construct the model using only four components, where each component represents an individual ion channel. By doing this, the dependency graphs would be much smaller and easier to specify. However, the component state-space graph would be more complex. In addition, certain symmetries or independence between subsets of components that may exist in models specified using finer components may not exist in models specified using coarser components. Keeping this in mind, we next demonstrate the use of coarser components, namely the entire single channel model, in specifying the four channel complex model.

First, we must redefine the component state-space graph \( C \) as follows: we let the set of component states \( S \) now be the set of state equivalence class representatives \( F \) of the reduced single-channel model and the set of component transitions \( Q \) be the set of system transitions \( \Lambda \) in the reduced single-channel model. All transitions in the new component state-space graph are local transitions. However, all transitions representing a \( \text{Na}^+ \) binding event in the single channel model must be made to be functional transitions in the complex model. Next, the rules and associated dependency graphs need to be defined. One difficulty that arises when specifying the rules for the coarser components is that, although there are many more component transitions, each of these transitions involves only one of a much smaller number of elementary reactions so that the number of labels (rates) is much smaller than the number of component transitions. Also, since we are dealing with the reduced single-channel model, a multiplicity is associated with each transition which must be accounted for. To avoid all of these difficulties, this model composition process can be automated using the VERNAN tool. In VERNAN, a function (SystemToComponent.m) is provided which takes the system state-space graph as input, and produces a new component state-space graph \( C \), a set of rules, and a mapping of transitions to a set of labels as output.

Finally, we define the new dependency graphs in \( D \) for the rates associated with the transition \( Q_5 \). (A .mat-file, Example3.mat, containing the specifications of this model and VERNAN outputs is included in the VERNAN toolbox package.) In this case, there are only six graphs that need to be defined, which are shown in Table VII. Although \( \text{Na}^+ \) can bind with twelve different rates depending on the open/closed configuration of the four channels in the complex (as illustrated in Figure 12), any given channel only depends on the other channels, and there are only six possible configurations that these other channels can be in. With the finer components, we also had to account for the dependencies between subunits of the same channel, but these dependencies are implicitly accounted for by using the coarser components.

With the new set of model specifications tailored for the larger components, we generate the reduced-state space graph of the system using VERNAN, generating a model with 304 590 system states and 5 414 760 system transitions as before. In this example however, there are only eight symmetries rather than 32,768 as in the previous example. This is because the symmetries corresponding to swapping the \( \beta \) components have already been accounted for in constructing the single channel model, and we are left with only those symmetries that can be generated by rotating or flipping the four-channel ring.

If we index the system states of the two models in the same way, we can verify that the transition matrices associated with the two models are identical (see supplementary material for MATLAB codes), validating the use of coarser components for specifying the model. There was a slight improvement in efficiency in generating the model using coarser components. Where it took \( \sim 140 \) minutes (on a desktop with 8-core 3.4 GHz Intel Xeon processors and 16 GB RAM) to generate the model using the 28 component specification, it took \( \sim 100 \) minutes using the 4 component specification. This speed up is attributed to the smaller characteristic graph size when using the larger components.
which greatly improved the efficiency of the canonical graph labeling routine.

**TABLE VII.** Dependency graphs for the rate constants associated with the binding of Na\(^+\) to a β subunit in the four component Na\(^+\) sodium channel complex model. Each graph is associated with more than one rate label (due to the fact that multiple component transitions in the four component complex model correspond to a Na\(^+\) binding event), which is why rate label superscripts are given as asterisks.

| y | A\(_y\) | B\(_y\) | L\(_y\) |
|---|---|---|---|
| \(D(r^*_y)\) | 1 | \{1\} | (2,3,4) | (C,C,C) |
| 1 | \{2\} | (1,3,4) | (C,C,C) |
| 3 | \{3\} | (1,2,4) | (C,C,C) |
| 4 | \{4\} | (1,2,3) | (C,C,C) |
| \(D(r^*_y)\) | 1 | \{1\} | (2,3,4) | (O,C,C) |
| 2 | \{1\} | (2,3,4) | (C,C,O) |
| 3 | \{2\} | (1,3,4) | (O,C,C) |
| 4 | \{2\} | (1,3,4) | (C,O,C) |
| \(D(r^*_y)\) | 1 | \{1\} | (2,3,4) | (C,O,C) |
| 2 | \{2\} | (1,3,4) | (O,C,O) |
| 3 | \{3\} | (1,2,4) | (O,C,O) |
| 4 | \{3\} | (1,2,4) | (C,O,O) |
| \(D(r^*_y)\) | 1 | \{1\} | (2,3,4) | (O,C,O) |
| 2 | \{1\} | (2,3,4) | (C,O,O) |
| 3 | \{2\} | (1,3,4) | (O,O,O) |
| 4 | \{2\} | (1,3,4) | (O,O,O) |
| \(D(r^*_y)\) | 1 | \{1\} | (2,3,4) | (O,O,O) |
| 2 | \{1\} | (2,3,4) | (O,O,O) |
| 3 | \{2\} | (1,3,4) | (O,O,O) |
| 4 | \{2\} | (1,3,4) | (O,O,O) |

E. **Example 5: Mean-field coupled ion channels**

Although it is not possible to achieve a further exact reduction of the ion-channel complex model described above using the techniques described here, it is certainly possible to make simplifying assumptions that would allow a further approximate reduction. One such simplifying assumption that is commonly applied to calcium-release site models is the mean-field coupling approximation,\(^{29,30}\) in which the local calcium concentration in a cluster of channels is assumed to be uniform across all channels so that the concentration at any one channel depends only on the numbers of open/closed channels in the cluster and not any specific spatial configuration of open/closed channels. We can apply this same approximation to our Na\(^+\) channel complex model (here, we apply it to our 28 component model; model specifications and VERNAN outputs are included in the file Example4.mat). By doing this, the number of different rates at which Na\(^+\) binds to the β subunit changes from 12 in the spatially coupled model, to five in the mean-field coupled model. The rules by which transition \(Q_s\) maps to these five rates must be defined, requiring us to re-specify our dependency graphs. The dependency graphs for all other rates remain the same.

As we have seen when specifying the spatially-coupled 28 component model, defining the dependency graphs can be a tedious process when a large number of components are involved. Specifying dependency graphs of a mean-field coupled model can be even more daunting, requiring a hyperarc for every possible configuration of open/closed channels. Although this process may be automated by writing a simple script, VERNAN provides the option of defining dependency graphs in a mean-field coupling format (versus the spatially-coupled format which has been used up to now). Using the mean-field coupling format, we only need to specify one or more classes of component states along with the number of components required to occupy each of the classes of states. Dependency graphs in mean-field coupling format for the five rate constants mapping to \(Q_s\) are given in Table VIII for the 28 component model.

Generating the reduced state-space graph of the newly specified system yields a model with 111 930 system states and 1 897 480 system transitions, an approximately three-fold reduction compared to the spatially-coupled model. In this example, there are 98,304 symmetries consisting of the group of permutations generated by independently swapping the β components of each channel and by rearranging the four-channels (there are 4! possible rearrangements of the channels). This number of states is simply the number of combinations (with repetition) of the four channels on the set of 39 possible channel states in the reduced single channel model, given by the binomial coefficient

\[ |F| = \binom{39}{4} + 4 - 1 \]
Although we refer to the mean-field coupled model as an approximation, we emphasize that we are referring to an approximation in the mechanism of coupling, not an approximation in the solution of the master equation. Using the system specification of the mean-field coupled model, the 119930 state model generated using Algorithm II is an exact reduction of the 21 381 376 state model that would be generated using Algorithm I given sufficient time and memory capacity.

F. Example 6: Independent ion channels

Next, we assume that each individual channel in the complex is spaced far enough apart that the channels are effectively uncoupled from one another. In this case, the characteristic graph of the system becomes disconnected, and since no synchronizing transitions between channels exist, each channel is independent. (VERNAN gives the option of testing for these conditions.) Then the channel complex model can be decomposed into four equivalent subsystems, each subsystem being the single channel model. In this case, an exact solution to the 21 382 376 state ion channel complex model may be obtained directly from the solution to the 39 state reduced single channel model using Eq. (6).

G. Example 7: Gene regulatory circuit

As a final example, we illustrate the regulation of a transcription factor by three different promotors labeled promoter A (P_A), promoter B (P_B), and promoter C (P_C). Each promoter binds to the same transcription factor (labeled TF), but with different binding kinetics. Thus, in this case the promotors are distinguishable whereas the transcription factors are not. The component state-space graph \( \mathcal{C} \) for the channel may be formulated as shown in Figure 13 and Table IX. We assume that the network contains two of each type of promoter and two transcription factors so that the system domain contains eight elements. We assign the initial system state \( f_{\text{init}} \) of the eight components as \( (1,1,3,3,5,5,7,7) \), i.e., all promoters and transcription factors in the dissociated state. According to \( \mathcal{C} \), there are 6 possible component transitions, all of which represent synchronizing transitions. In this example, there are no functional transitions, so there is no need to specify any dependency graphs for the transitions.

Generating the full state-space graph yields a model with 28 system states and 144 system transitions. Generating the reduced state-space graph yields a model with 10 system states and 24 transitions. In this case, there are 16 symmetries consisting of those generated by independently swapping components corresponding to a given type of promoter or the transcription factor. Although the characteristic graph of the system is disconnected, none of the components are independent from one another because each component participates in a synchronizing transition with disjoint components (see Appendix C for discussion about independence). Therefore, it is not possible to achieve a further reduction on this system using the techniques described here.

The reductions achieved for the above examples and computation times for the examples are summarized in Tables X and XI, respectively.

VI. DISCUSSION

A. Summary

The system description formalism presented here offers a convenient and intuitive approach for specifying complex state transition systems with many different types of interactions. The technique allows the transition rules of a system to be systematically defined, helping to minimize the potential for modeling errors that could occur using a more direct manual approach. In addition, rule definitions do not require learning complicated language syntax and semantics, as they are easily understood in terms of simple graph theory. The way in which rules are specified allows a natural and automated application to the high-level model of the symmetry reduction technique and reduction by decomposition of independent subsystems to achieve potentially significant exact reductions of master equations associated with the models.
To illustrate how the method may be applied to the modeling of biochemical systems, we have presented a hypothetical Na\(^+\) channel complex model involving several different types of components and many different interactions. (All models were constructed using VERNAN, our MATLAB implementation of the method (codes are available upon request).) Using the reduction techniques described here, we generated a reduced model with an over 70-fold reduction in state-space when compared to the full (unreduced) version of the model. In general, highly symmetric dependencies among system components (as with the mean-field coupling approximation), or independence between components, results in large state-space reductions.

### B. Improvements and future directions

Although the reduction techniques described here focused on those that can be done a priori using a HLM, further exact reduction may be possible using techniques that operate directly at the state-level model. This further reduction may be possible for two reasons: (1) symmetries or independence between components exist at the state-space level that are non-identifiable at the system-description level; and (2) states exist that are either forward or backward bisimilar, but not both. The first situation can be avoided by careful modeling at the system-description level (see discussion in Appendix C for discussion). The second situation on the other hand cannot be managed at the system-description level and requires a posteriori processing to account for. In this case, the algorithm of Derisavi et al.\(^{26}\) may be applied to compute the coarsest possible aggregation of the generated state-space.

In addition, it is also possible to compliment the exact reduction strategies described here with one or more approximate reduction strategies to achieve additional reduction of a model. Potential avenues for further reduction using approximation strategies include truncation\(^7\) and probabilistic evaluation\(^8,9\) techniques and perturbation methods such as time-scale separation,\(^31,32\) aggregation of nearly lumpable states,\(^10\) and decomposition of nearly independent subsystems. Probabilistic evaluation techniques such as the finite state projection algorithm of Munska\(^9\) and Khammash\(^9\) are particularly attractive as they offer the potential to automatically truncate the state-space of a very large, possibly infinite, state-space to a much smaller projection space of size just large enough to satisfy a user-specified tolerance in the total probability density error. (Although our system description method is restricted to systems with finite numbers of states and state transitions, it is possible to extend the specification method to allow specification of components with infinite state-spaces.) Algorithm II could be implemented directly in the finite state projection algorithm to expand the projection space on each iteration.

Finally, there is room for improvement in the efficiency of the reachability analyses described by Algorithms I and II. In particular, Algorithm II would benefit from implementing one of the more efficient algorithms described by Junttila\(^{33,34}\) for finding the canonical representative of a system state. (Using NAUTY\(^{27}\) to find a canonical labeling of a characteristic graph is a bottleneck in computing the reachable state-space for systems with large characteristic graphs.) Also, computing the multiplicity for transitions with many possible combinations of reactants is costly using Algorithm II, as it requires finding a canonical representative for the destination state of each combination. A better algorithm for computing the multiplicity function would therefore result in a significant speed up. Additionally, there is much room for improvement in the speed of our VERNAN implementation of these algorithms by incorporating more efficient data structures and search methods.

### ACKNOWLEDGMENTS

This work is supported by the National Institute of Health under Grant No. P50-GM094503.

### APPENDIX A: CONVERTING A DIRECTED ARC-LABELED HYPERGRAPH TO AN EQUIVALENT DIRECTED VERTEX-LABELED SIMPLE GRAPH

Reaction dependencies, or rules, are specified in the form of directed arc-labeled hypergraphs. These graphs are used to construct a characteristic graph \(\mathcal{G}\) for a system, as described in the text. In VERNAN, automorphisms of \(\mathcal{G}\) are identified using the NAUTY algorithm\(^{27}\) which requires that graphs be given in simple vertex-labeled form. Converting \(\mathcal{G}\) (a hypergraph) to an equivalent simple graph \(\tilde{\mathcal{G}}\) is done according to the procedure described as follows:

1. Generate a partition \(\{U_1, U_2, \ldots, U_n\} \vdash U\) such that all hyperarcs belonging to a given \(U_i\) have the same tail and head sets, and all hyperarcs belonging to \(U_j\) for \(j \neq i\) have tail and head sets different from those in \(U_i\)

2. For each \(U_i\), define a labeling function \(\mu_i\) assigning a multiset of labels to each head \(d \in B\), such that:

\[
\mu_i(d) = \psi_{ij}\left(L_i(d), r_k^{(ij)}\right).
\]

### TABLE X. State-space size and symmetry summary.

| Example | Full | Symm | Red | Rel. size (%) |
|---------|------|------|-----|--------------|
| 2       | 68   | 8    | 39  | 57           |
| 3       | 21381376 | 32768 | 304590 | 1.4 |
| 4       | 21381376 | 8    | 304590 | 1.4 |
| 5       | 21381376 | 98304 | 111930 | 0.5 |
| 6       | 21381376 | 8    | 39   | <0.01        |
| 7       | 28   | 16   | 10  | 36           |

### TABLE XI. Computation times (in seconds) for algorithms.

| Example | Algorithm I | Algorithm II | Algorithm III and IV |
|---------|-------------|--------------|----------------------|
| 2 and 6 | 0.155       | 0.171        | 0.031                |
| 3       | NA          | 8621         | 5367                 |
| 4       | NA          | 6185         | 680                  |
| 5       | NA          | 2637         | 1562                 |
| 7       | 0.067       | 0.08         | 0.01                 |

\(\mu_i(d) = \psi_{ij}\left(L_i(d), r_k^{(ij)}\right)\),
APPENDIX B: STATE EQUIVALENCE TESTING AND CANONIZATION METHOD

There are several different ways of testing the equivalence of two states $f_1$ and $f_2$. A simple but naïve approach is to generate the full equivalence class $[f_1]$ of state $f_1$ by applying each element $g \in G$ to $f_1$, and to then test whether any function in $[f_1]$ is equal to $f_2$. This method works fine if $G$ is of low order but quickly becomes problematic for larger order $G$. For example, consider the case where all $p$ components of a system are identical symmetrically (that is, all permutations of the $p$ components are symmetries). Then, the group $G$ is the symmetric group of order $p!$, which can be very large for even relatively small $p$.

A better approach is to compute the canonical representatives of the states $f_1$ and $f_2$, and to then test whether the canonical representatives are the same. Several alternative methods for computing canonical representatives are presented in Refs. 33 and 34. Each of these methods exploits a structure known as the Schreier-Sims representation of a group, which is a compact representation of a group in the form of a base and strong generating set that allows linear time computations with groups. In our implementation, we apply the method described in section 3 of Ref. 33, which is based on the use of a graph canonizer to find the canonical version of a system state’s characteristic graph.

The characteristic graph of a system state $f$ is defined as a graph $G_f$ whose vertex set contains $D$ such that, for all system states $f_1$ and $f_2$, the following holds:

1. if $g \in G$ maps $f_1$ to $f_2$, then there is an isomorphism $\gamma$ from $G_{f_1}$ to $G_{f_2}$ such that $\gamma$ restricted to $D$ equals $g$, and
2. if $\gamma$ is an isomorphism from $G_{f_1}$ to $G_{f_2}$, then $\gamma$ restricted to $D$ belongs to $G$ and maps $f_1$ to $f_2$.

Thus, the characteristic graphs $G_{f_1}$ and $G_{f_2}$ of $f_1$ and $f_2$ are isomorphic if and only if $f_1 \sim f_2$.

The bulk of the work for finding characteristic graphs of system states is done by constructing the system characteristic graph $G$. The characteristic graph $G_f$ of a system state $f$ is simply $G$ with vertices corresponding to $D$ labeled according to $f(d)$ for all $d \in D$. Next, a black-box graph canonizer, such as the one provided in the NAUTY tool,37 is applied to find the canonical version of $G_f$. A function $\mathcal{K}$ is a graph canonizer if, for all graphs $G_1$ and $G_2$, the following holds:

1. $\mathcal{K}(G_1)$ is isomorphic to $G_1$ and
2. $\mathcal{K}(G_1) = \mathcal{K}(G_2)$ if and only if $G_1$ and $G_2$ are isomorphic to each other.

By applying the canonizer $\mathcal{K}$ to a system state characteristic graph $G_f$, we obtain the canonical version $\mathcal{K}(G_f)$ of the graph along with an isomorphism $\gamma$ carrying $G_f$ to $\mathcal{K}(G_f)$.

Given a permutation group represented in Schreier-Sims form, Junnila describes the concept of compatible permutations and gives an algorithm (Algorithm 2.2 of Ref. 33) for enumerating permutations in the group which are compatible with a place valuation $pval$ (a function of the form $pval : D \rightarrow \mathbb{N}$) under a multiset selector (a function from a multiset to a subset of its underlying set of elements such that each element in the subset is associated with a non-zero
multiplicity). Junttila proves that, if $pval$ is an injective function, and the multiset selector used is a function multiset selector (a multiset selector with a singleton image set), then one and only one unique permutation is compatible with $pval$. Junttila further proves that, by letting $pval$ be equal to the isomorphism $\gamma$ (restricted to $D$), and by finding the unique permutation $g \in G$ compatible with $pval$ under a function multiset selector, then the canonical form of the system state $f$ may be found by applying the inverse of $g$ to $f$ (Theorem 3.1 of Ref. 33). That is, $\mathcal{K}(f) = g^{-1}(f)$ is the canonical representative of system state $f$.

Using the canonical representative approach, Algorithm II may be improved. Specifically, in line 15, instead of testing whether each destination state $f'$ is isomorphic to any state in $\bar{F}$, the canonical version $\mathcal{K}(f')$ of $f'$ may be computed and tested for membership in $\bar{F}$.

**APPENDIX C: TESTING FOR INDEPENDENCE OF SYSTEM COMPONENTS**

Independence of subsets of system components implies the absence of both functional and synchronizing interactions between the subsets of components. Determining whether there are any functional interactions between subsets of components can be done at the system description level by examining the characteristic graph $G$ of the system. Similarly, determining whether there are any synchronizing interactions between component subsets can be done by examining the component graph $C$ in conjunction with $f_{\text{ini}}$. A sufficient condition for the existence of independent subsets of components is then given by the following:

1. the system characteristic graph $G$ is disconnected and
2. there exists a maximally connected subgraph of $G$ such that none of its components are involved in a synchronizing transition with the components of any other disjoint maximally connected subgraph of $G$.

Criterion number 1 is straightforward to test using a standard graph search algorithm such as the one given below:

**ALGORITHM III.** An algorithm for computing the connectivity of a characteristic graph.

```
1: Set unprocessed = V
2: while unprocessed ≠ ∅ do
3: Select any source ∈ unprocessed
4: Set $\pi$ (source) = source
5: Set queue = source
6: while queue ≠ ∅ do
7: Select any $v$ ∈ queue and set queue = queue\{v\}
8: Set unprocessed = unprocessed\{v\}
9: for all $E_j, e_i \in E_j \cap (A_i \cup B_j)$ do
10: for all $d \in A_i \cup B_j \cap unprocessed\{d\}$
11: Set $\pi(d) = source$
12: Set queue = queue ∪ $\{d\}$
13: return $\pi$
```

This algorithm generates a partition $\{V_1, V_2, \ldots, V_n\} \vdash V$ of the vertices $V$ of $G$ into $n$ maximal connected subsets such that the value of $\pi(d)$ is the same for all $d \in V_j$ and different for all $d \notin V_j$. If the partition is non-trivial (i.e., $\pi(V)$ is not singleton), then criterion number 1 is satisfied.

Criterion number 2 is more difficult to test, requiring that a reachability analysis be performed on $f_{\text{ini}}$ similar to Algorithms I and II. However, a stricter test can be done at the system description level, therefore not requiring a traversal of the reachability graph. This involves testing whether a synchronizing transition involving components of disjoint maximal connected subsets even exists (whereas criterion number 2 allows such a transition to exist but requires that components not participate in it). An algorithm for the test is given below:

**ALGORITHM IV.** An algorithm to merge disconnected subgraphs of $G$ based on the existence of synchronizing transitions involving components of the subgraphs.

```
1: Set unprocessed = V
2: while unprocessed ≠ ∅ do
3: Select any $v$ ∈ unprocessed and set unprocessed = unprocessed\{v\}
4: for all $Q_j \in \{Q_j \in Q \mid \sum_{e \in X_j} Z_j(s) > 1 \land T(f_{\text{ini}}(v)) = T(s)\}$
5: Select any $s \in \{s \in X_j \mid T(f_{\text{ini}}(v)) = T(s)\}$
6: for all $d \in unprocessed$ do
7: if $\pi(d) \neq \pi(v)$ then goto 11
8: for all $d' \in V \setminus \{\pi(d') = \pi(d)\}$ do
9: Set $\pi(d') = \pi(v)$
10: if $\pi(V) = \{v\}$ then goto 11
11: return $\pi$
```

In line 7, $(X_j, Z_j) \setminus \{s\}$ is the multiset difference rather than the usual set difference.

The above is a union-find algorithm that merges subsets of the partition $\{V_1, V_2, \ldots, V_n\} \vdash V$ (given by $\pi$) together when synchronizing transitions exist between subsets. If the updated partition is non-trivial ($\pi(V)$ is non-singleton), then both criteria of the independence condition are satisfied.

The test given by the above algorithm should be sufficient in most cases. Careful modeling, especially of $f_{\text{ini}}$ and of $Q$, can ensure that the stronger test need never be performed. (Note that criterion number 2 can also be verified directly a posteriori by examining the reachability graph $S$ after it is generated. In general however, it is preferred that the full $S$ is never generated when a reduced version exists.)

When specifying a model at the system description level, we account for all possible interactions between system components. However, as already mentioned in the case of synchronizing transitions, there will be no interaction in the model unless the transition actually occurs in the reachability graph. This is also the case for functional transitions. That is, a functional interaction does not exist in a model unless its associated functional transition actually occurs in the reachability graph.

This subtle but important point has implications regarding both criterion number 1 of the independence condition above as well as in the model symmetry. If we are accounting for a functional interaction in our system characteristic graph, and that interaction does not actually exist in the reachability graph, then it is possible that there are symmetries in the model that we are not accounting for. While it is possible to verify the existence of interactions a posteriori by examining the generated reachability graph $S$, careful modeling at the system description level can render this step
unnecessary. For example, if a particular transition in \( Q \) requires that four components of type \( y \) be in state \( x \) to occur, and there are only three such components specified in \( \text{fin} \), then it is obvious that the transition never occurs and should therefore be removed from \( Q \).

When the independence condition is satisfied, the full system can be decomposed into a number of unique orthogonal subsystems by first partitioning \( D \) according to \( \pi \). That is, \( D \) is partitioned into \( \{D_1, D_2, \ldots, D_n\} \) such that the value of \( \pi(d) \) is the same for all \( d \in D_j \) and different for all \( d \in D_i \). Next, each subsystem can be characterized as unique or non-unique by testing the following:

1. whether any other subsystem contains the same number and type of components and
2. whether its associated subsystem characteristic graph, defined as the induced subgraph of \( G \) with vertex set \( D_i \), is isomorphic to any other subsystem’s characteristic graph.

State-level models generated for each unique subsystem may be used to obtain an exact solution to the full model as described in the main text.

References:

1. N. Le Novere and T. S. Shimizu, *Bioinformatics* 17(6), 575 (2001).
2. M. L. Blinov, J. R. Faeder, B. Goldstein, and W. S. Hlavacek, *Bioinformatics* 20(17), 3289 (2004).
3. V. Danos and C. Laneve, *Theor. Comput. Sci.* 325(1), 69 (2004).
4. T. Junttila, *Ph.D. dissertation*, Helsinki University of Technology, 2003.
5. J. Feret, V. Danos, J. Krivine, R. Harmer, and W. Fontana, *Proc. Natl. Acad. Sci.* 106(16), 6453–6458 (2009).
6. J. Sproston and S. Donatelli, *IEEE Trans. Software Eng.* 28(8), 531 (2006).
7. J. Feret, H. Koeppl, and T. Petrov, “Stochastic fragments: A framework for the exact reduction of the stochastic semantics of rule-based models,” *Int. J. Software Inf.* (in press), available online at http://infoscience.epfl.ch/record/142570.
8. J. Feret, T. Henzinger, H. Koeppl, and T. Petrov, *Theor. Comput. Sci.* 431(4), 137 (2012).
9. V. Nguyen, R. Mathias, and G. D. Smith, *Bull. Math. Biol.* 67(3), 393 (2005).
10. S. Derisavi, H. Herrmanns, and W. H. Sanders, *Inf. Process. Lett.* 87(6), 309 (2003).
11. W. Douglas Obal II, M. G. McQuinn, and W. H. Sanders, *IEEE Trans. Reliab.* 56(4), 643 (2007).
12. J. G. Kemeny and J. L. Snell, *Finite Markov Chains* (Van Nostrand, Princeton, NJ, 1960).
13. S. Derisavi, H. Herrmanns, and W. H. Sanders, *Inf. Process. Lett.* 87(6), 309 (2003).
14. D. D. Devours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. G. Webster, *IEEE Trans. Software Eng.* 28(10), 256 (2002).
15. T. Junttila, Ph.D. dissertation, Helsinki University of Technology, 2003.
16. J. G. Kemeny and J. L. Snell, *Finite Markov Chains* (Van Nostrand, Princeton, NJ, 1960).
17. T. Junttila, *Ph.D. dissertation*, Helsinki University of Technology, 2003.
18. J. Feret, V. Danos, J. Krivine, R. Harmer, and W. Fontana, *Proc. Natl. Acad. Sci.* 106(16), 6453–6458 (2009).
19. J. G. Kemeny and J. L. Snell, *Finite Markov Chains* (Van Nostrand, Princeton, NJ, 1960).
20. P. J. Schweitzer, *In Proceedings of the International Workshop on Computer Performance and Reliability* (North-Holland, Amsterdam, The Netherlands, 1984), pp. 275.
21. W. Douglas Obal II, M. G. McQuinn, and W. H. Sanders, *IEEE Trans. Reliab.* 56(4), 643 (2007).
22. J. G. Kemeny and J. L. Snell, *Finite Markov Chains* (Van Nostrand, Princeton, NJ, 1960).
23. J. Feret, V. Danos, J. Krivine, R. Harmer, and W. Fontana, *Proc. Natl. Acad. Sci.* 106(16), 6453–6458 (2009).
24. J. Feret, H. Koeppl, and T. Petrov, “Stochastic fragments: A framework for the exact reduction of the stochastic semantics of rule-based models,” *Int. J. Software Inf.* (in press), available online at http://infoscience.epfl.ch/record/142570.
25. J. Feret, T. Henzinger, H. Koeppl, and T. Petrov, *Theor. Comput. Sci.* 431(4), 137 (2012).
26. W. Douglas Obal II, M. G. McQuinn, and W. H. Sanders, *IEEE Trans. Reliab.* 56(4), 643 (2007).
27. T. Junttila, *Ph.D. dissertation*, Helsinki University of Technology, 2003.
28. J. Sproston and S. Donatelli, *IEEE Trans. Software Eng.* 28(8), 531 (2006).
29. J. G. Kemeny and J. L. Snell, *Finite Markov Chains* (Van Nostrand, Princeton, NJ, 1960).
30. P. J. Schweitzer, *In Proceedings of the International Workshop on Computer Performance and Reliability* (North-Holland, Amsterdam, The Netherlands, 1984), pp. 275.
31. S. Derisavi, H. Herrmanns, and W. H. Sanders, *Inf. Process. Lett.* 87(6), 309 (2003).
32. B. D. McKay, *NAUTY*, version 2.4, June 2010. Downloaded from the internet at http://cs.anu.edu.au/~bdm/nauty/.
33. See supplementary material at http://dx.doi.org/10.1063/1.4758074 for MATLAB codes and tables for examples.
34. V. Nguyen, R. Mathias, and G. D. Smith, *Bull. Math. Biol.* 67(3), 393 (2005).
35. J. L. Greenstein, R. Hinch, and R. L. Winslow, *Biophys. J.* 90(1), 77 (2006).
36. C. A. Gomez-Uribe, G. C. Verghese, and A. R. Tzafriri, *J. Chem. Phys.* 129(24), 244112 (2008).
37. S. Derisavi, B. Munskey, and M. Khammash, *J. Chem. Phys.* 125(20), 13 (2006).
38. T. Junttila, *Research Report A75* (Helsinki University of Technology, Laboratory for Theoretical Computer Science, Espoo, Finland, 2002).
39. T. Junttila, *Applications and Theory of Petri Nets, Lecture Notes in Computer Science* (Springer-Verlag, Berlin, 2004), Vol. 3099, pp. 258.