ON THE DETECTION OF SYMMETRIES IN COMPOSITIONAL MARKOV MODELS

Ruth Lamprecht, Peter Kemper

Department of Computer Science
College of William & Mary

Abstract—A model-based evaluation of a system’s design often considers to what degree components need to be available multiple times in order to reach a desired level of availability, reliability or dependability. Multiple components of the same kind then lead to models with regular structures and symmetries. In stochastic models, especially Markovian models, such regularities have been used to establish lumpability results. In this paper, we propose a procedure to detect symmetries in a Markovian model that is built in a compositional manner by sharing state variables. The symmetries give insight into a model and help to achieve a significant state space reduction, which alleviates the effects of the infamous state space explosion problem. The results extend existing work of Obal, McQuinn, and Sanders; in particular, we focus on variables in functional transition rates that commute in order to take additional symmetries into account. The overall approach contributes to Möbius, a multi-paradigm, multi-solution framework for the model-based dependability and performance assessment of systems.

I. INTRODUCTION

An assessment of the dependability of a system is often performed with a stochastic model that constitutes a continuous time Markov chain (CTMC). Dependability measures are then obtained from impulse or rate rewards measured at certain time points (transient analysis) or in the long run (steady state analysis). For applications, a discrete event simulation is often applied due to its few restrictions and broad applicability; however, simulation results in statistical estimates and confidence intervals and finds its limits if events of interest like failures are rare. State-based numerical solution methods compute results from a transient or steady state distribution that are computed exactly (up to numerical precision). However, the infamous state space explosion problem limits that applicability of state-based numerical solution methods to CTMCs with state spaces in the range of $10^6 - 10^8$ states. Much research has gone into pushing the limits for numerical CTMC analysis in the last two decades, with notable successes based on symbolic and other structured representations of a generator matrix $Q$ [1], [2] to the extent that the state space exploration and generation of extremely large CTMCs is not considered a limiting factor any more. For the iterative solution of CTMCs with $10^7$ states, be it with randomization for transient analysis or some iterative scheme for steady state, the limiting factor is the space used to represent intermediate and final results for a transient or steady state distribution vector $\pi$ in the size of the state space. To deal with this problem, we see largeness tolerance techniques like symbolic representations of iteration vectors with variants of decision diagrams [3] and most recently approximations with Kronecker representations [4]. A complementary line of research is on largeness avoidance techniques that focus on a smaller, reduced model to obtain results. Many results in the literature have their fundamentals in an exact aggregation based on lumpability.

Lumpability is defined as the process of masking the states and transitions inside a partition group of a Markov chain while still producing a Markov chain [5]. For a given CTMC, exact and ordinary lumpability can be detected by efficient partition refinement methods that vary according to the employed data structure to represent a CTMC. Most notably, Derisavi, Hermanns, and Sanders show how to perform partition refinement efficiently [6] while Derisavi shows this with a symbolic data structure [7]. Instead of partitioning a state space at the detailed level of individual states, there are several approaches that take advantage of structural information that can be obtained for certain compositional classes of models, where certain symmetries can be proven to induce lumpability for the associated CTMC. Early work dates back to Sanders and Meyer [8] and stochastic activity networks (SANs), which have two composition operations $\text{Rep}$ and $\text{Join}$ that allow for generation of a reduced, lumped CTMC. Obal, McQuinn and Sanders generalized the concept to a graph composition [9], [10]. A separate thread of results has been obtained for Stochastic Well-formed Nets (SWNs) [11], [12], where the concept of colors in stochastic Petri nets and a particular way of using colors helps to establish theoretical results that guarantee lumpability on the associated CTMC based on structural conditions observed in an SWN. SWNs and SANs are technically different, but share the idea that structural information is obtained from a model description to reduce the associated CTMC with the help of lumpability. A related partitioning method is exact performance equivalence [13], which employs bisimulation to preserve quantitative results from stochastic automata.

In this paper, we extend the approach of Obal, McQuinn and Sanders [10]. The original idea is to map the compositional structure of a composed model to an undi-
rected graph, the model composition graph (MCG), such that symmetries in the graph relate to lumpable states in the associated CTMC. Since algorithms are known that identify symmetries in graphs, there is an automated way to obtain a method that yields the representative state of a set of lumpable states from a permutation of nodes in the model composition graph. The approach guarantees to find all symmetries in a given MCG. The only way to miss a symmetry in a composed model is in the way the MCG is defined. From working with Markovian models that take spatial aspects into account, we realized that certain symmetries that a composed model had did not carry over to the MCG of Obal et al. The extension we propose in this paper allows us to express and subsequently exploit more symmetries in a model to obtain a reduced, lumped CTMC.

We begin our discussion by defining the basic aspects of the model and present our findings using the notation of [10].

II. SYMMETRIES AND LUMPABILITY

We begin our discussion by defining the basic aspects of the model and present our findings using the notation of [10].

A. Models and their Composition

The base level model consists of the variables used to define the state of the model, the events that can occur given specific values for the variables, and the new values that are assigned when an event occurs. Formally, we have the following definition.

**Definition 2.1:** A model is a five-tuple \((S, E, \varepsilon, \lambda, \tau)\) where

- \(S\): set of state variables \(\{s_1, s_2, \ldots, s_n\}\) with mapping \(\mu\) for values of each state variable. Let \(M = \{\mu|\mu : S \rightarrow \mathbb{N}\}\) be the set of all such mappings where \(\mu(s)\) is the value of the state variable. Let \(SV \subseteq S\) denote a set of variables to be shared.
- \(E\): set of events \(\{e_1, e_2, \ldots, e_m\}\) that may occur.
- \(\varepsilon\): event enabling function with \(E \times M \rightarrow \{0, 1\}\), where \(\forall e \in E, \mu \in M, \varepsilon(e, \mu) = 1\) if event \(e\) can occur in state \(\mu\) and 0 otherwise.
- \(\lambda\): transition rate function with \(E \times M \rightarrow (0, \infty)\), where \(\forall e \in E, \mu \in M\) s.t. \(\varepsilon(e, \mu) = 1, e\) occurs in state \(\mu\) with exponential rate \(\lambda(e, \mu)\).
- \(\tau\): state transition function with \(E \times M \rightarrow M\), where \(\forall e \in E, \mu \in M, \tau(e, \mu) = \mu'\), the new state reached when \(e\) occurs in state \(\mu\).

Def. 2.1 describes a state transition system where a transition from a state \(\mu_i\) to \(\mu_j\), due to some event \(e\), subject to an existing condition \(\varepsilon(e, \mu_i) = 1\), occurs exponentially distributed random delay \(\lambda(e, \mu_i)\), with \(\mu_j\) determined by \(\tau(e, \mu_i)\). Def. 2.1 constitutes a CTMC with an \(|M| \times |M|\) generator matrix \(Q\) defined by

\[
Q_{ij} = \sum_{e \in E(i,j), i \neq j} \lambda(e, \mu_i)
\]

where \(E(i,j) = \{e \in E|\varepsilon(e, \mu_i) = 1, \tau(e, \mu_i) = \mu_j\}\) and \(Q_{ii} = -\sum_{i \neq j} Q_{ij}\).

For illustrating purposes, we consider a two-state dependability model \(A\) with a single state variable, \(S = \{a\}\), and two states, \(M = \{\mu_0, \mu_1\}\), that indicate if the system is up \((\mu_1(a) = 1)\) or down \((\mu_0(a) = 0)\). There are two events, failure \(f\) and repair \(r\), hence \(E = \{f, r\}\). \(\varepsilon(e, \mu) = 1\) for \((e = f, \mu = \mu_1)\) or \((e = r, \mu = \mu_0)\) and 0 otherwise.

For those pairs of events and states \((e,\mu)\) where \(\varepsilon(e, \mu) = 1\), we define \(\lambda(f, \mu_1) = \alpha_1\), \(\lambda(r, \mu_0) = \alpha_0\), \(\alpha_0, \alpha_1 > 0\), \(\tau(f, \mu_1) = \mu_0\), and \(\tau(r, \mu_0) = \mu_1\). The generator matrix is

\[
Q = \begin{pmatrix}
\alpha_0 & \alpha_1 \\
-\alpha_0 & -\alpha_1
\end{pmatrix}
\]

We will use this model as a building block to compose a larger model in what follows.

For models where state variables encode input variables, we observed cases where state variables can exchange values with no effect on the dynamic behavior of the model. We denote this formally by saying that two variables commute.
if they enable the same events, they have the same effect on transition rates, and they cause the same state transitions.

**Definition 2.2:** Two state variables $s_1$ and $s_2$ commute if for all $\mu \in M$ and $e \in E$ the following holds:

1. $\varepsilon(e, \mu) = \varepsilon(e, \bar{\mu})$
2. $\lambda(e, \mu) = \lambda(e, \bar{\mu})$
3. $\tau(e, \mu) = \mu'$ and $\tau(e, \mu) = \bar{\mu}'$ (i.e., $\tau(e, \bar{\mu}) = \tau(e, \mu)$)

where $\bar{\mu}(s_i) = \begin{cases} \mu(s_2) & \text{if } i = 1 \\ \mu(s_1) & \text{if } i = 2 \\ \mu(s_i) & \text{otherwise} \end{cases}$

and analogously for $\mu'$. $T$ is the partition of $SV$ that is induced by the relation commute.

It is easy to see that commute is an equivalence relation and its equivalence classes induce a partition of $SV$ into disjoint sets, which we denote by $\tilde{\Sigma}$.

We consider a second model $B$ with variables $S = \{b_0, b_1, b_2\}$ and states $M = \{\mu_0, \ldots, \mu_7\}$. As for $A$, there are events for failure and repair, $E = \{f, r\}$. Variable $b_1$ and $b_2$ are used as input-only variables. The following table lists variable settings for individual states and functions $\varepsilon()$, $\lambda()$ and $\tau()$ that are defined and non-zero.

<table>
<thead>
<tr>
<th>$M$</th>
<th>$b_0$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$\varepsilon(r, \cdot)$</th>
<th>$\lambda(r, \cdot)$</th>
<th>$\tau(r, \cdot)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_0$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$\gamma$</td>
<td>$\mu_4$</td>
</tr>
<tr>
<td>$\mu_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$\gamma$</td>
<td>$\mu_5$</td>
</tr>
<tr>
<td>$\mu_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\gamma$</td>
<td>$\mu_6$</td>
</tr>
<tr>
<td>$\mu_3$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$\gamma$</td>
<td>$\mu_7$</td>
</tr>
<tr>
<td>$\mu_4$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$\gamma$</td>
<td>$\mu_0$</td>
</tr>
<tr>
<td>$\mu_5$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\gamma$</td>
<td>$\mu_1$</td>
</tr>
<tr>
<td>$\mu_6$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$\gamma$</td>
<td>$\mu_2$</td>
</tr>
<tr>
<td>$\mu_7$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$\gamma$</td>
<td>$\mu_3$</td>
</tr>
</tbody>
</table>

In model $B$, variables $b_1$ and $b_2$ commute if $g_1 = g_2$. For states $\mu_0, \mu_3, \mu_4, \mu_7$, this is immediate since exchanging values of $b_1$ and $b_2$ yields the same state again, i.e., $\mu_0 = \mu_6$.

For state $\mu_1$, we have $\mu_1 = \mu_2$ and $\tau(r, \mu_1) = \tau(r, \mu_2)$ and vice versa for $\mu_2$. For state $\mu_5$, we have $\mu_5 = \mu_6$ and $\tau(r, \mu_5) = \tau(r, \mu_6)$ and vice versa for $\mu_6$. Conditions for $\varepsilon()$ and $\lambda()$ are straightforward to check. We see that rates only match if $g_1 = g_2$.

Conditions for state variables to commute appear to be very restrictive. However, for certain types of variables this property is natural. Shared state variables can either be used as input-only (the model does not change the values of the variables itself) or not. For input-only shared state variables, the last condition in Def. 2.2 is simple to fulfill. If two state variables commute and are not input-only, then the model must change them in the same manner, as well as use them in the same manner, which makes them redundant. Thus, if two state variables commute, we expect to see that they are mainly used in the model as input-only. For the second condition (and similar for the first), consider a model where the transition rate $\lambda(e, \mu)$ for some event $e$ is dependent on the values of $s_1$ and $s_2$ by a commutative function (e.g. $\mu(s_1) + \mu(s_2)$, $\mu(s_1) \ast \mu(s_2)$, min($\mu(s_1), \mu(s_2)$), or max($\mu(s_1), \mu(s_2)$)). The increase (or decrease) in this rate is the same if either variable is increased (or decreased). So there are cases where state variables are present that commute.

We can consider building a larger model by composing instances of submodels through sharing state variables. A **composed model** will contain the instances of the submodels, as well as the connection sets of state variables shared between the instances. The composed model has a state variable set that is the union of the state variables $S_i$ of all instances with the connection sets merging the state variables that are shared between instances.

**Definition 2.3:** A composed model is a four-tuple $(\Sigma, I, \kappa, C)$ where

- $\Sigma$: set of models
- $I$: set of instances of models in $\Sigma$
- Each instance is a complete and independent copy of a model, except as defined in the connection set.
- $\kappa$: instance type function with $I \rightarrow \Sigma$
- $C$: set of connections

Each connection $c \in C$ is a set of subsets of shared variables such that only one shared variable from a particular instance is in any one subset. $\mu(c)$ is the mapping of the shared variable $s_{ij} \in SV_i$ of instance $I_i$ to exactly one connection set $c$. Shared variables then map to the same connection set.

Enabling, rate and state transition functions of any event $e_i$ of instance $i \in I$ carry over. $\varepsilon(e_i, \mu) = \varepsilon(e_i, \mu(i))$, $\lambda(e_i, \mu) = \lambda_i(e_i, \mu(i))$, and $\tau(e_i, \mu) = (\tau_i(e_i, \mu(i)))$. $\lambda_i(\cdot), \tau_i(\cdot)$ are functions of instance $i$.

The composed model divides the state variable set of each instance into two groups: those variables that are shared with other instances and those that are not. Subsets of the state variable set are called state variable fragments, while the subset of variables that are not shared will be referred to as the private state variable fragment. If all state variables are shared, this subset will be empty. The public state variable fragment for instance $I_i$, denoted as $SV_i$, then contains the state variables that are shared. We can partition the set of all shared variables, $\cup SV_i$, in one of two ways: namely with respect to sharing and with respect to commuting. Variables that are shared across several instances actually represent only a single variable and need to be merged; each connection $c$ describes a set of variables that need to be merged and the set of connections partitions $\cup SV_i$. $T_{\kappa}$ denotes the partition of $SV_i$ into subsets of variables that commute. Each instance partitions its own variables, however all instances of the same submodel do so in the same way.

We create a composed model that consists of instances $I = \{A_1, A_2, B_1\}$ of models in $\Sigma = \{A, B\}$, type function $\kappa(A_1) = \kappa(A_2) = A, \kappa(B_1) = B$, and connections...
\( C = \{c_1, c_2\} \), where \( c_1 \) merges variable \( A_1.a \) (variable \( a \) of instance \( A_1 \)) with variable \( B_1.b_1 \) \((b_1 \text{ of } B_1)\) and \( c_2 \) merges variable \( A_2.a \) \((a \text{ of } A_2)\) with variable \( B_1.b_2 \) \((b_2 \text{ of } B_1)\).

Since instances of \( A \) share all their variables with \( B \) and \( B \) does not modify them, the set of states for the composed model coincides with that of \( B \). This results in an 8 \( \times \) 8 generator matrix

\[
Q = \begin{pmatrix}
0 & \alpha_0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_1 & \alpha_0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_1 & 0 & \alpha_0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_1 & \alpha_1 & \alpha_1 & \alpha_1 & \alpha_1 & \alpha_1 & \alpha_1 & \alpha_1 \\
\lambda & 0 & 0 & 0 & \alpha_0 & 0 & 0 & 0 \\
g_2 \lambda & 0 & 0 & 0 & \alpha_0 & 0 & 0 & 0 \\
g_1 \lambda & 0 & 0 & 0 & \alpha_1 & \alpha_1 & \alpha_1 & \alpha_1 \\
g_1 \lambda & \alpha_1 & \alpha_1 & \alpha_1 & \alpha_1 & \alpha_1 & \alpha_1 & \alpha_1
\end{pmatrix}
\]

where \( \ast \) denotes negative row sums of off-diagonal entries. We observe that the following sets of states are lumpable \( \{\mu_1, \mu_2\} \) and \( \{\mu_5, \mu_6\} \) if \( g_1 = g_2 \). So we can reduce the matrix to a 6 \( \times \) 6 matrix. We can even identify the reasons for this from the compositional structure of the model: instances of \( A \) are equal and can switch roles for the instance of \( B \) because functions \( \epsilon_B(), \lambda_B(), \) and \( \tau_B() \) of \( B \) do not distinguish among the value settings of \( b_1 \) and \( b_2 \), i.e., those variables commute for \( B \). For this to take place, it is not essential that \( A \) shares all variables with \( B \) (a situation we purposely selected to retain a small set of states for illustration purposes). What is essential is that \( B \) contains a symmetry with respect to its variables \( b_1 \) and \( b_2 \) that extends into the composed model since those variables are shared with instances of the same model.

In [10], Obal et al. describe how to recognize lumpability for state sharing composed models with the help of symmetries that are detected for an associated MCG. We closely follow their notation and concept. However, note that the aforementioned example is not included in Obal’s approach, since all shared variables of a model are considered distinct and separate; the possibility of commuting variables is not taken into account. In the following, we show how to extend the approach by Obal et al. to consider commuting variables. This requires us to resolve two issues: 1) describe an MCG whose symmetries include those already covered by the approach of Obal et al. as well as additional symmetries that are introduced by variables that commute, and 2) we need to prove that the symmetries of that MCG imply an equivalence relation on states that matches with lumpability of the associated CTMC.

We define a model composition graph for a given composed model, which is an undirected graph \( G = (V, W) \), whose symmetries will help us identify lumpable states in the associated CTMC. The vertex set, \( V \), has as elements the private state variable fragments (represented by each instance \( I_0 \)), the shared state variables, and the connection nodes. A connection node \( c_k \) represents a set of shared variables such that only one shared variable from a particular instance is in the set. The edge set, \( W \), must then satisfy two rules: (1) every shared variable is connected to the private segment for that instance, and (2) every shared variable is connected to exactly one connection node as defined by \( \mu_C \).

Formally, we have the following definition:

**Definition 2.4:** A model composition graph (MCG) is an undirected graph \( G = (V, W) \), where \( V = \bigcup_{i \in I} SV_i \cup C \) and \( W = \{(v, w)|\{v, w\} \in (I \cup C) \times \bigcup_{i \in I} SV_i\}\).

The MCG for our illustrating composed model is shown in Fig. 1. The MCG contains 3 instance nodes labeled with \( A_1, A_2, \) and \( B_1, 4 \) shared variable nodes labeled with \( A_1.a, A_2.a, B_1.b_1, \) and \( B_1.b_2, \) and 2 connection nodes \( c_1 \) and \( c_2 \).

![MCG diagram](image)

**Figure 1.** MCG of our sample model having three instances.

For any undirected graph, there are many methods to find automorphisms (e.g. [16]) that produce a set of permutations that allow us to permute nodes such that edges still match. We are looking for permutations on the MCG that, if applied to a state of the associated CTMC, yield an equivalent state with respect to lumpability. Relevant permutations need to take into account that the MCG does not have a homogeneous vertex set since this set is comprised of instances, shared variables, and connection nodes. So an automorphism of an MCG must not only match vertices by the number of edges, but also by the type. In order to guarantee this, we define \( \Xi = \{\xi_1, \xi_2, \ldots, \xi_n\} \), a partition of \( V \), that satisfies the following requirements. Two vertices are in the same partition element if:

1. they both correspond to private state variable fragments of the same model and contain the same private state variables, \( \text{OR} \)
2. they both correspond to public state variable fragments of the same model and contain the same shared state variable, \( \text{OR} \)
3. they are both connection nodes, \( \text{OR} \)
4. they both correspond to public state variable fragments of the same model and the two shared state variables commute.

The first 3 requirements are the same as originally used in [10]. The fourth condition relaxes the second since it allows for state variables that commute by Def. 2.2 to be in the same partition.

The MCG in Fig. 1 contains a permutation that is compatible with the connectivity in the MCG and the four requirements enumerated above. The permutation exchanges \( A_1 \) with \( A_2, A_1.a \) with \( A_2.a, B_1.b_1 \) with \( B_1.b_2, c_1 \) with \( c_2 \), and \( B_1 \) with itself. Note that this permutation relies on the fourth condition above to work, which extends the original approach of Obal et al. In Obal’s work, that permutation...
would not be considered since variables $b_1$ and $b_2$ are not the same as required by the second condition.

As in [10], we now let $\Gamma$ be the automorphism group of the graph with respect to $\Xi$. Since $\Gamma$ is a general automorphism group, it follows that by [10] this automorphism group can be used to detect symmetry in a model. In the next section we justify why these symmetries correspond to lumpability in the associated CTMC.

B. Lumpability

The concept of lumpability is that given two (or more) states, the states have the same transition function and move to states that are themselves lumpable. Formally, as presented by Kemeny and Snell [17] for discrete time Markov chains,

Definition 2.5: A necessary and sufficient condition for a Markov chain to be lumpable with respect to a partition $\Xi = \{\xi_1, \xi_2, \ldots, \xi_n\}$ is that for every pair of sets $\xi_i$ and $\xi_j$, the probability to move into $\xi_j$ is the same for all states in set $\xi_i$.

The corresponding formulation for CTMC requires the consideration of the aggregated rate instead of the probability to move into $\xi_j$ from any state in set $\xi_i$. The states can then be lumped together, with one chosen as the representative state. Since we know we can find the automorphisms of the graph, we can now use this information to find the symmetries of the graph. With the symmetries defined by the automorphisms, we will have sets of states that can be lumped to reduce the model state space.

We define $\Gamma$ as the automorphism group of the graph with respect to $\Xi$. More specifically, $\Gamma$ is a permutation group on the vertex set of the composition graph, which means that a vertex is in a given partition if and only if that vertex can be mapped to that partition by one of the permutations. Formally, this means that for all $\gamma \in \Gamma$, $v \in \xi_i$ iff $\gamma(v) \in \xi_i$.

Permutations in $\Gamma$ map $V$ onto itself. $\gamma(s)$ denotes the state variable in the fragment $\gamma(v)$ that is the image of $s$ under $\gamma$. Now consider the effect of an automorphism $\gamma$ on the composed model state $\mu$. The action of $\gamma$ on $\mu$ is defined as

$$\mu^\gamma = \mu \circ \gamma$$

where $\circ$ denotes composition of functions. Furthermore, $\mu^\gamma(s) = \mu(\gamma(s))$ for every state variable $s$. If two state variables, $s_1$ and $s_2$, can commute by Def. 2.2, then $\mu^\gamma(s_1) = \mu^\gamma(s_2)$. In this way the permutation or commutation takes a given composed model and composed model state, and rearranges the vertex names according to the automorphism or commutation while the composed model state remains the same.

In terms of the composed model, the automorphism of the MCG permutes the states of the instances among themselves. That is, if $B^\gamma = A$ for two instances $A$ and $B$ with some automorphism $\gamma$, $\mu^\gamma(B.s) = \mu(A.s')$ for all state variables $s$ where $s$ and $s'$ commute (note that $s$ commutes with itself, so the case $s = s'$ is included).

We can define an equivalence relation $L$ as follows:

Definition 2.6: $L$ is a relation such that for two composed model states, $\mu_1$ and $\mu_2$, $\mu_1L\mu_2$ if there exists a $\gamma \in \Gamma$ such that $\mu_2 = \mu_1^\gamma$.

Proposition 2.7: $L$ is an equivalence relation. (See [9] for this proof.)

By Prop. 2.7, we know that $\Gamma$ partitions the state space of the composed model into equivalence classes defined by $L$. Now consider what that means for the transition of states in the same equivalence class. That is, when $\mu_1L\mu_2$, we want to show that $\mu_1' \mu_2'$, where $\tau(e, \mu_1) = \mu_1'$ and $\tau(e, \mu_2) = \mu_2'$, for all $e \in E$. This means that if two composed model states are in the same equivalence class, they have the same set of next possible states. To show this, first it is necessary to discuss the relation between permutations of the state space and permutations of an instance. We now define what it means for the permutation $\gamma$ to act on an instance $A$, which is a subset of $S$, the set of state variables of the composed model.

Definition 2.8: The domain of $\mu_A$ is denoted as $D(\mu_A)$.

The action of $\gamma$ on $\mu_A$ is defined as

$$[\mu_A]^\gamma = \{(s, \mu_A(\gamma(s))) | \gamma(s) \in D(\mu_A)\}.$$ 

Note that if two state variables, $s_1$ and $s_2$, of instance $A$ commute by Def. 2.2, then $\gamma(s_1), \gamma(s_2) \in D(\mu_A)$.

Using Def. 2.8, if two instances $A$ and $B$ permute, that is $B^\gamma = A$, then the permutation of the local state of $A$, $[\mu_A]^\gamma$, leads to the same state as the permutation of the state space, $\mu^\gamma$, and then examining the block of states for the local state of $B$, $[\mu_A]^\gamma|_B$. Formally,

Proposition 2.9: For all instance pairs $A$ and $B$ and for all $\gamma \in \Gamma$ such that $B^\gamma = A$, $[\mu_A]^\gamma = [\mu_B]^\gamma$. This equality is conditioned on the shared variables that commute by Def. 2.2 in that two commutable variables may need to be commuted for true equality.

Proof: Since automorphisms are one-to-one and onto, for any instance $A$, there exists an instance $B$ such that $B^\gamma = A$. Then the set of state variables of instance $A$ such that $\gamma(s) \in D(\mu_A)$ is the same as the subset of state variables projected from the set of the composed model state variables onto instance $B$. That is, $D([\mu^\gamma]_B) = D([\mu_A]^\gamma)$ and $[\mu_A]^\gamma$ assigns the state variables of $A$ onto the corresponding, or a commutably corresponding, state variables of $B$ since by definition of $\Gamma$, $\gamma$ can only permute variables of the same type, or variables that can commute.

Now that we know the effect of the automorphism on the composed model state, we consider the effect of the automorphism on the state transition function. Specifically, we want to show that if two composed model states are in the same equivalence class, the transition rates out of those states are equal for any event.

By definition of $L$, $A$ and $B$ must be in the same class if $B^\gamma = A$. By Prop. 2.9 and Def. 2.2, the projection of the
local state of $A$ is the same as the subset of the projection of
the composed model state corresponding to the local state of $B$, with allowance for differences due to commutable
state variables. Therefore, elements of the same equivalence
class are symmetric. This means that all elements in the
same class of $L$ have future behavior that is statistically
indistinguishable. So, given two composed model states, the
sets of next possible states are equivalent under $L$. Formally,

Proposition 2.10: $\forall \mu \in M, \epsilon \in E, \text{ and } \gamma \in \Gamma,$

$$\tau(\epsilon, \mu)^{\gamma} = \tau(\epsilon^{\gamma}, \mu^{\gamma})$$

Proof: The definition of the composed model state transition function
is $\tau(\epsilon, \mu) = (\mu - \mu_A) \cup \tau_A(\epsilon, \mu_A)$, for all
$\epsilon \in E_A$ and composed model states $\mu$, where $\tau_A$ is
the state transition function for instance $A$. Applying the
automorphism $\gamma$ to this, we get:

$$\tau(\epsilon, \mu)^{\gamma} = [(\mu - \mu_A) \cup \tau_A(\epsilon, \mu_A)]^{\gamma}$$
$$= [\mu_{S} - \mu_A]^{\gamma} \cup [\tau_A(\epsilon, \mu_A)]^{\gamma}$$

since they are disjoint sets
$$= [\mu_{S} - \mu_A]^{\gamma} \cup [\tau_B(\epsilon^{\gamma}, [\mu^{\gamma}]_{B})]$$

by Prop. 2.9
$$= (\mu^{\gamma} - [\mu^{\gamma}]_{B}) \cup \tau_B(\epsilon^{\gamma}, [\mu^{\gamma}]_{B})$$

The last step is a simple rewriting of the previous step. We
can then apply the definition of the composed model state
transition function and use the fact that $A$ and $e$ are arbitrary
to get the result.

We define the set of states reached from a composed
model state $\mu$ as

$$\Delta_{\mu} = \bigcup_{\{\epsilon \in E|\tau(\epsilon, \mu)\neq\emptyset\}} \tau(\epsilon, \mu)$$

Since each state in $\Delta_{\mu}$ is also in some equivalence class of
$L$, we can define a destination class of $\mu$ as any class in $L$
that contains a state from $\Delta_{\mu}$. We now state that

Proposition 2.11: Every pair of composed model states
$\mu_1$ and $\mu_2$ such that $\mu_1 \mu_2 \mu_2$ will have the same destination
classes and the same transition rates to those classes.

Proof: We know that if $\mu_1 \mu_2$, then they have the same set
of possible next states. That is, they have the same reachable
states and $\Delta_{\mu_1} = \Delta_{\mu_2}$. By Prop. 2.10, $\tau(\cdot, \mu_1)$ has the same
destination classes as $\tau(\cdot, \mu_2)$. By definition of $\Gamma$, for every
event that can occur in $\mu_1$, there is a corresponding event
in $\mu_2$. Then for each destination class $H$, both $\mu_1$ and $\mu_2$
have the same possible events. Since the transition rates are
defined on the events, this means that $\mu_1$ and $\mu_2$ have the
same transition rates to any destination.

Prop. 2.11 establishes equivalent behavior for one step in
the future. That is, given two composed model states, if they
are in the same equivalence class, they have the same set
of transition functions and transition rates. We now have
an equivalence class that partitions the state space of the model
in a way that allows us to use a single representative state
in place of each class. Formally,

Theorem 2.12: Replacing the equivalence classes of $L$
with a representative state produces a model that satisfies
the Markov property.

Proof: Follows by using Prop. 2.11 and Def. 2.5.

Therefore, by Theorem 2.12 we can use the automorphism
groups of the MCG to find the symmetries in the model and
then use these symmetries to reduce the state space through
lumpability.

III. Results

The motivation for the work presented here is to model
problems that belong to a large class of problems that involve
some kind of spatial awareness as well as state variables that
can commute. The spatial awareness of a model is related
to the distance between units in the model, either in a two-
dimensional space or a three-dimensional space, where the
different distances lead to different effects in the model. For
example, in the sample model used earlier in this paper,
the value of $g_1$ could be defined in terms of the distance
between $B$ and $A_1$ and similarly for $g_2$. Symmetries in
the model arise from regularity in the layout of the units,
which could be either in an engineered manner for a man-
made system or as an approximation of the natural layout.
In the sample model, the symmetries are present because
$A_1$ and $A_2$ are of an identical type and because they are
both the same distance away from $B$. The commutable state
variables are related to input variables whose values can
be permuted in such a way as to have the same effect
on the overall system, as discussed earlier for $b_1$ and $b_2$.
This class of problems includes models of failure or error
propagation due to workload redistribution or environmental
issues such as heat dispersion or radiation diffusion. In this
type of model, for example, when one unit fails because of
overheating, other units nearby will also feel an increased
temperature, thereby increasing their probability of failing
because of overheating. Another example might be where
a unit fails because of overloading, causing the work to be
redistributed to nearby units, which increases their workload,
putting them closer to failure because of overloading. Other
possible models could consist of communication networks
(where the time for communication is based on the distance
between units) or transportation networks (congestion and
capacity are influenced by the distance between units). Both
of these types of models can have symmetries because units
are placed at regular intervals along a line or in a grid. There
are also biological models where the activity of one unit
causes an increased likelihood of activity in nearby units [18]
or epidemic models studying the spread of disease, where,
again, symmetries are present through the regular layout of
units.

To illustrate the benefits of the implementation of comm-
utable state variables, we provide a failure propagation
A model of a sensor network with a hexagonal layout for the sensors, shown in Fig. 2a. Each sensor was considered to be a simple two-state machine, working or failed. The rate to move from a working state to a failed state was dependent on the state of the other sensors in the system, and how far away each sensor was. There was a base failure rate, \( f_i \), for each sensor \( i \) that had failed, defined on the distance \( d \) between the two sensors, with larger distances having less of an effect than smaller ones. For instance, consider sensor \( A \) from Fig. 2a. The failure rate of \( A \) is \( f + f(d_{AB})s_B + f(d_{AC})s_C + f(d_{AD})s_D + f(d_{AE})s_E + f(d_{AF})s_F \) where \( s_B \) is 0 when \( B \) is working and 1 otherwise and \( d_{AB} \) is the distance between \( A \) and \( B \). The other distance and status variables are defined similarly. Defining the failure rate in this manner means that the failure rate of a sensor increases as more sensors fail. Since we are using the hexagon, \( d_{AB} = d_{AF} \) and \( d_{AC} = d_{AE} \). Therefore, the increase of the failure rate of \( A \) is the same if either \( B \) or \( F \) fails; likewise, the increase of the failure rate is the same if either \( C \) or \( E \) fails.

Using the regularity of a hexagon, we know that the order of the automorphism group is 12; six symmetries from rotations and six symmetries from reflections. The full state space has \( 2^6 \) states (six models with two possible states each), which can be reduced to 13 lumped states. The 13 resulting lumped states for the hexagonal layout can be seen in Fig. 3. However, since we must specify the failure rate of \( A \) in terms of the distance from \( A \) to each of the other sensors, the reflection symmetries cannot be found if there is no way to tell that two state variables (\( s_B \) and \( s_F \) or \( s_C \) and \( s_E \)) can commute. With the approach defined by Obal et al. [10], one can only find the six symmetries from rotation because of the required distinction of the distances of a sensor to all other sensors. Because of this the number of states is increased by one for the now distinct states of 110100 and 110010. In the first case, sensor \( C \) has the two closest sensors, its “neighbors”, failed. A reflection to the second case where \( F \) now its two closest sensors (neighbors) failed would match \( C \)'s failed left neighbor (\( D \)) with \( F \)'s failed right neighbor (\( E \)). In the method from [10], this symmetry is not found because left and right are not the same. With the new method presented here, the modeler can state that the left and right neighbors can commute, so the reflection symmetry can be found.

With the simple model shown here, we found simple improvement. As we increase the size of the system with more sensors, layers, and/or states per sensor, these savings increase to a more substantial value. By increasing the number of states per sensor from 2 up to 4 (available, processing, failed, sleeping) or 8 (addition partial states), we can gain savings of nearly 50% from lumping using reflection symmetries. We can also increase the number of sensors in the network. The hexagonal layout can still be used by adding a sensor between every pair, thus doubling the system size to twelve (see Fig. 2b). This system also gains a reduction by nearly a factor of 2. If we have a two-tier system with an inner ring that has six sensors and an outer ring with twelve sensors, giving a system with eighteen sensors, we can calculate the reductions for the 2- or 3-state sensors. The values were calculated by iteration of the state space and itemizing rotations and reflections. Table I lists our results, which show that our algorithm produces significant reduction. The first column in the table states the number of sensors in the system while the second column lists the number of states in each sensor. The third column states the number of states found through the reduction method of Obal et al.[10], the fourth column states the number of states from the method presented here, and the last column shows the percentage difference between the two methods. It should be noted that different system configurations can produce the same number of states in the full state space, but there will be a different number of rotation and reflection symmetries due to the different configurations.

Fig. 4 shows another example application where six routers are composed in a ring by sharing the state variables for the left and right links for each pair of routers. The

![Figure 2. Six (a) and twelve (b) sensors in a hexagonal layout with full connectivity between sensors.](image-url)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

![Figure 3. The 13 states for a hexagonal layout sensor system with six sensors.](image-url)
example is modeled with Möbius and with the Möbius Graph Composer that implements the symmetry detection and lumping approach by Obal et al. It is a simple example problem of a dependability model where each submodel has three basic state variables $priv$, $link1$, and $link2$. Each variable is used to indicate if that part of the system is up (1) or down (0). Obviously, if $priv$ is 0, then both $link1$ and $link2$ must also be 0. Thus, the status of each router is affected by the status of its left and right neighbors, and its own status affects its neighbors, but the effect is the same in both directions. That is, the status of router $A$ changes in the same way if either $B$ or $F$ fails and the failure of $A$ affects $B$ and $F$ in the same way. Therefore, the router does not really need the information of which neighbor is or is not up, it just needs to know that zero, one or both have failed. A router also does not need to share its own status to its neighbors separately. But if we have only a single variable, $link$, all submodels would share this one variable, and the system is no longer a ring. Thus the models specify left and right (or $link1$ and $link2$). The $Join$ nodes are where the state variables are shared. For instance, $JoinAB$ shares $A→link1$ with $B→link2$. We have kept the routers simple for illustration purposes, but they can easily be made to have more complicated actions, including being connected to processors or other units.

With the model in Fig. 4, each router has five possible states \{priv, link1, link2\} = \{(1,1,1), (1,1,0), (1,0,1), (1,0,0), (0,0,0)\}. Thus with six routers, there are a total of $5^6$ possible states for the system. However, there are many symmetries present due to the chosen layout and the fact that all routers are of the same submodel type. With a ring, we have rotation and reflection symmetries. Rotations of the ring, moving router $A$ to the position for $B$, $B$ to the position for $C$, and so on, will all have the same Markov chain since the left and right connectivity remains the same. Reflections of the ring, moving router $A$ to the position for $D$, $B$ to the position for $E$, and so on, again will have the same Markov chain.

The approach of Obal et al. [10] is able to detect all symmetries based on rotations, which permute instances of submodels, $link1$ variables among themselves and $link2$ variables among themselves. However, if we take into account that for a single router submodel, variables $link1$ and $link2$ commute, we can find another symmetry which is by reflection: having 3 routers working such that two are neighbors and the third is not. In terms of Fig. 4, the case where routers $A$, $B$, and $D$ are working is equivalent to having routers $A$, $B$, and $E$ working. With the sharing of the state variables, this model has a full state space of 322 states, which can be reduced to 111 by the method of Obal et al. This can be further reduced to 49 (a 56% reduction) by including the method introduced here.

Our algorithm can be applied to other models besides the sensor network or router ring. We have also done work with biochemical systems that involve signaling complexes where units are inter-dependent and the distance between units influences the strength of the effect the state of one unit will have on other units in the system [19]. One such model is a simple two-state $Ca^{2+}$-regulated $Ca^{2+}$ channel, that is, the channel is either open or closed and the amount of calcium surrounding the channel influences the rate at which the channel opens. Specifically, the opening rate is defined in terms of the background $[Ca^{2+}]$ that is always present, plus some amount for every open channel, with the effect of more distant channels being diluted over the distance. In the natural biology, a release site (a collection of several channels) will not necessarily have a regular layout, but a standard modeling practice is to place the channels in a regular layout, such as a hexagonal (beehive) or rectangular (grid) lattice. Fig. 5 shows such a release site with nine channels in a $3 \times 3$ grid. With a grid layout, there are less rotational symmetries, but there are still reflection symmetries. For instance, through the regularity of the grid, the opening rate of channel $A$ is changed in the same manner if either channel $B$ or $D$ opens, since the distance between $A$ and $B$ is the same as between $A$ and $D$. Likewise, $F$ and $H$ effect the opening rate of $I$ in the same manner. Furthermore, the pairs (C, G), (B, D), and (F, H) equally change the opening rate of $E$. This gives a reflection symmetry on the diagonal. For this system, the full state space has $2^9(512)$ states, which can be reduced to 140 states by lumping under rotational symmetries. This can be further reduced to 104 states (a 26% reduction) by also lumping with reflectional symmetries.

**Table I**

<table>
<thead>
<tr>
<th>State Space</th>
<th>Orig. Compact</th>
<th>New Compact</th>
<th>Rel. Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>4096</td>
<td>430</td>
<td>39%</td>
</tr>
<tr>
<td>6</td>
<td>46,656</td>
<td>4291</td>
<td>45%</td>
</tr>
<tr>
<td>8</td>
<td>262,144</td>
<td>23,052</td>
<td>47%</td>
</tr>
<tr>
<td>10</td>
<td>1,000,000</td>
<td>86,185</td>
<td>52%</td>
</tr>
<tr>
<td>12</td>
<td>531,441</td>
<td>44,727</td>
<td>50%</td>
</tr>
<tr>
<td>18</td>
<td>262,144</td>
<td>22,282</td>
<td>52%</td>
</tr>
</tbody>
</table>

*State-space configurations for the composed sensor network.*

**Figure 4.** Models are combined into a composed model via shared state variables. Here we show the Graph composition for a six router system.
IV. Conclusion

In order to make numerical solution methods scale with real world applications, much work has gone into reduction methods for large CTMCs, mainly focused on lumpability. We presented an automated procedure to obtain a lumped CTMC for a compositional Markov model where submodels are composed by sharing state variables. Our approach extends previous work by Obal, McQuinn and Sanders to take into account if shared variables commute. With this additional piece of information, we obtain a variant of a model composition graph such that an automated symmetry detection mechanism can identify and exploit more symmetries and yield a smaller lumped CTMC. From a modeling point of view, variables that commute can be understood as input variables to a submodel where values can be permuted without changing the total effect to the overall model. This new approach is particularly beneficial to a class of models, where a network of components is spatially distributed in a regular manner, e.g., a grid or a hexagonal layout, and failures rates are state-dependent functions defined by the state of other components as well as their distance. Ongoing work is dedicated to a full integration into the multi-paradigm multi-solution framework Möbius. The current implementation relies on a user-given specification of variables that commute. Moving towards a semi- or fully-automated detection of those variables remains for future work.

References


