Contextual Lumpability*

J. Hillston
University of Edinburgh, UK
Jane.Hillston@ed.ac.uk

A. Marin and S. Rossi
Università Ca' Foscari
Venezia, Italy
{marin,srossi}@dais.unive.it

C. Piazza
Università di Udine, Italy
carla.piazza@uniud.it

ABSTRACT

Quantitative analysis of computer systems is often based on Markovian models. Among the formalisms that are used in practice, Markovian process algebras have found many applications, also thanks to their compositional nature that allows one to specify systems as interacting individual automata that carry out actions. Nevertheless, as with all state-based modelling techniques, Markovian process algebras suffer from the well-known state space explosion problem. State aggregation, specifically *lumping*, is one of the possible methods for tackling this problem. In this paper we revisit the notion of Markovian bisimulation which has previously been shown to induce a lumpable relation in the underlying Markov process. Here we consider the coarser relation of *contextual lumpability*, and taking the specific example of strong equivalence in PEPA, we propose a slightly relaxed definition of Markovian bisimulation, named *lumpable bisimilarity*, and prove that this is a characterisation of the notion of contextual lumpability for PEPA components. Moreover, we show that lumpable bisimilarity induces the largest contextual lumping over the Markov process underlying any PEPA component. We provide an algorithm for lumpable bisimilarity and study both its time and space complexity.

1. INTRODUCTION

Performance evaluation of computer systems often relies on stochastic models whose underlying processes are Continuous Time Markov Chains (CTMCs). However, specifying complex hardware and/or software architectures by explicitly defining all the possible transitions between all the possible states of the models can be very complicated and prone to design errors. Higher level formalisms allow for compact, modular and often hierarchical descriptions of complex systems consisting of numerous components. Examples of higher level formalisms are queueing networks [16], stochastic Petri nets [18, 20] (SPNs), and Markovian process algebras [12, 13]. Specifically, in this paper we will focus on the Performance Evaluation Process Algebra (PEPA) which is one of the most popular Markovian process algebras. When dealing with models that consist of several components interacting with each other the number of states of the underlying CTMC may grow exponentially or more than exponentially with the number of components as happens for instance in SPNs. This problem, known as the *state space explosion*, makes the general algorithms for the performance and reliability analysis time and space consuming and numerically unstable [22].

Related work. At the stochastic process level of abstraction, several approaches, both exact and approximate, have been proposed to cope with the state space explosion problem. Hereafter we focus on lumping methods. In [14, Ch. 6] the authors introduce the notion of lumping of states in a Discrete Time Markov Chain (DTMC) but the concept can be straightforwardly extended to CTMCs. In *strong lumping* the states of the Markov chain are clustered according to some structural properties of the transition rate matrix so that a CTMC with a smaller number of states can be defined. Since the complexity of the analysis of this latter chain is lower than that required by the original one, lumping can be an effective way for studying the properties of large Markov chains. However, although some clustering of states can be suggested by intrinsic symmetries of the considered models, in general the complexity required for finding an optimal lumping can be prohibitive since it is still exponential with the number of model's components when the state space explosion occurs. The problem of defining efficient algorithms for lumping CTMCs is addressed in several papers, e.g., [2, 10, 23]. These papers propose different algorithms for solving the problem of deriving a lumping of the CTMC, however they can be applied once the joint process of the model is built and hence they do not affect the complexity of its generation.

A structural-based approach to lumping for SPNs is studied in [1, 3], where structural symmetries of the net are exploited to derive a lumped underlying CTMC in an efficient way. In the context of Markovian process algebras, structural process properties are studied in [4, 5, 6, 8, 13, 17] for state space reduction purposes by means of equivalence relations inspired by bisimulation. In [4, 8] Markovian bisimulations are deeply studied for Interactive Markov Chains and we will review their results in Section 3. In [5, 6, 13] the lumping is applied to the single components rather than to the joint process. The approach is interesting because the

*Work partially supported by the MIUR Project CINA "Compositionality, Interaction, Negotiation and Autonomicity" and by the EU project QUANTICOL, 600708.

ValueTools'13, December 10 – 12 2013, Turin, Italy

$$\frac{}{(\alpha,r).P \xrightarrow{(\alpha,r)} P} \qquad \frac{P \xrightarrow{(\alpha,r)} P'}{P+Q \xrightarrow{(\alpha,r)} P'} \qquad \frac{Q \xrightarrow{(\alpha,r)} Q'}{P+Q \xrightarrow{(\alpha,r)} Q'} \qquad \frac{P \xrightarrow{(\alpha,r)} P'}{P/L \xrightarrow{(\alpha,r)} P'/L}\ (\alpha \notin L) \qquad \frac{P \xrightarrow{(\alpha,r)} P'}{P/L \xrightarrow{(\tau,r)} P'/L}\ (\alpha \in L)$$

$$\frac{P \xrightarrow{(\alpha,r)} P'}{A \xrightarrow{(\alpha,r)} P'}\ (A \stackrel{def}{=} P) \qquad \frac{P \xrightarrow{(\alpha,r)} P'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha,r)} P' \underset{L}{\bowtie} Q}\ (\alpha \notin L) \qquad \frac{Q \xrightarrow{(\alpha,r)} Q'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha,r)} P \underset{L}{\bowtie} Q'}\ (\alpha \notin L)$$

$$\frac{P \xrightarrow{(\alpha,r_1)} P' \quad Q \xrightarrow{(\alpha,r_2)} Q'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha,R)} P' \underset{L}{\bowtie} Q'} \qquad R = \frac{r_1}{r_\alpha(P)} \frac{r_2}{r_\alpha(Q)} \min(r_\alpha(P), r_\alpha(Q))\ \ (\alpha \in L)$$

Table 1: Operational semantics for PEPA components

complexity of lumping a single component is much lower than that required by lumping the joint process and hence one can obtain a strong reduction of the cardinality of the state space with an acceptable computational complexity. In particular, an equivalence relation in the style of bisimulation (coinductive definition) is introduced. If two components are equivalent it is possible to replace one of them (that with more states) with the other without affecting the behaviour of the remaining parts of the system. Specifically, in [5, 6] the author proposes different weak Markovian bisimulation equivalences in the context of a Markovian process calculus. In all cases he shows that the CTMC-level aggregation induced by the bisimulation is a lumping only for specific classes of processes. Conversely, the notion of *strong equivalence* for PEPA processes introduced in [13] always induces a lumping of the CTMC underlying a PEPA process, although in general the opposite is not true.

Contribution. In this paper we propose a notion of Markovian bisimulation which is a *characterization* of a lumpable relation over the terms of a stochastic process algebra preserving contextuality and inducing a lumping in the underlying Markov processes. Specifically, we introduce the relation of *contextual lumpability* for PEPA components which is a congruence for the particular class of *evaluation* (or *static* [19]) contexts and complies with the ordinary lumping for Markov processes. Moreover, we require that terms in the same equivalence class may next engage in the same set of action types. We define a Markovian bisimulation, named *lumpable bisimilarity*, for PEPA terms which is a slightly relaxed variant of strong equivalence [13] and prove that this is a *characterisation* of the notion of contextual lumpability for PEPA components. Moreover, we show that lumpable bisimilarity induces the largest contextual lumping over the Markov process underlying any PEPA component. To the best of our knowledge, this is the first characterization of the class of processes in a Markovian process algebra whose underlying CTMC is (contextually) lumpable. Finally, starting from [23], we provide an algorithm for lumpable bisimilarity and study both its time and space complexity.

Structure of the paper. Section 2 introduces the syntax and the semantics of PEPA. In Section 3 we give the definition of contextual lumpability and in Section 4 we provide a coinductive characterisation of it. Section 5 presents an algorithm to derive the optimal lumping according to our characterisation. Section 6 concludes the paper.

2. THE CALCULUS

PEPA (Performance Evaluation Process Algebra) [13] is an algebraic calculus based on a classical process algebra and enhanced with stochastic timing information. It provides an expressive formal language which may be used to calculate performance measures as well as deduce functional system properties.

The basic elements of PEPA are *components* and *activities*. Each activity is represented by a pair (α, r) where α is a label, or *action type*, identifying it, and r is its *activity rate*, that is the parameter of a negative exponential distribution determining its duration. We assume that there is a countable set, $\mathcal{A}$, of possible action types, including a distinguished type, τ, which can be regarded as the *unknown* type. Activity rates may be any positive real number, or the distinguished symbol $\top$ which should be read as *unspecified*.

The syntax for PEPA terms is defined by the grammar:

$$\begin{aligned} P &::= P \underset{L}{\bowtie} P \mid P/L \mid S \\ S &::= (\alpha, r).S \mid S + S \mid A \end{aligned}$$

where S denotes a *sequential component*, while P denotes a *model component* which executes in parallel. A stands for constants which denote sequential components. We write $\mathcal{C}$ for the set of all possible components.

Operational semantics. PEPA is given a structural operational semantics, as shown in Table 1. The component $(\alpha, r).P$ carries out the activity (α, r) of type α at rate r and subsequently behaves as P. When $a = (\alpha, r)$, the component $(\alpha, r).P$ may be written as $a.P$. The component $P+Q$ represents a system which may behave either as P or as Q. $P+Q$ enables all the current activities of both P and Q. The first activity to complete distinguishes one of the components, P or Q. The other component of the choice is discarded. The continuous nature of the probability distributions ensures that the probability of P and Q both completing an activity at the same time is zero. The cooperation combinator $\underset{L}{\bowtie}$ is in fact an indexed family of combinators, one for each possible set of action types, $L \subseteq \mathcal{A} \setminus \{\tau\}$. The *cooperation set* L defines the action types on which the components must synchronise or *cooperate* (the unknown action type, τ, may not appear in any cooperation set). It is assumed that each component proceeds independently with any activities whose types do not occur in the cooperation set L (*individual activities*). However, activities with action types in the set L require the simultaneous involvement of both compo-

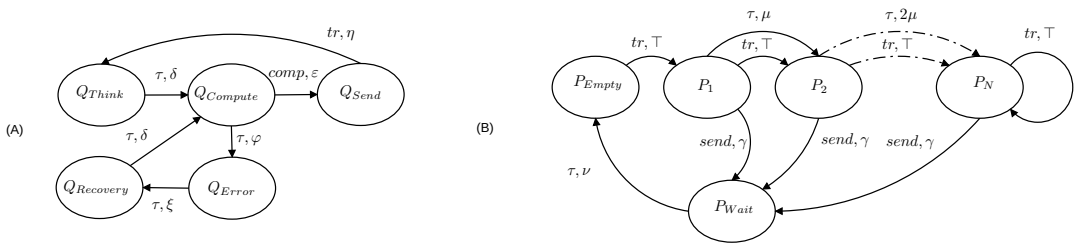

Figure 1: Producer (A) and consumer (B) models considered in Example 1

nents (*shared activities*). These shared activities will only be enabled in $P \underset{L}{\bowtie} Q$ when they are enabled in both P and Q. The shared activity will have the same action type as the two contributing activities and a rate reflecting the rate of the slower participant. If an activity has an unspecified rate in a component, the component is passive with respect to that action type. In this case the rate of the shared activity will be completely determined by the other component. In general, the rate of a shared activity will reflect the capacity of each component to carry out activities of that type. For a given P and action type α, this is the *apparent rate* of α in P, denoted $r_\alpha(P)$, that is the sum of the rates of the α activities enabled in P. The component P/L behaves as P except that any activity of type within the set L are *hidden*, i.e., they are relabelled with the unknown type τ. Finally, we assume that there is also a countable set of *constants*. Constants are components whose meaning is given by a defining equation such as $A \stackrel{def}{=} P$ which gives the constant A the behaviour of the component P.

The semantics of each term in PEPA is given via a labelled *multi-transition system* where the multiplicities of arcs are significant. In the transition system, a state or *derivative* corresponds to each syntactic term of the language and an arc represents the activity which causes one derivative to evolve into another. The set of reachable states of a model P is termed the *derivative set* of P ($ds(P)$) and constitutes the set of nodes of the *derivation graph* of P ($\mathcal{D}(P)$) obtained by applying the semantic rules exhaustively. We denote by $\mathcal{A}(P)$ the set of all the *current action types* of P, i.e., the set of action types which the component P may next engage in. We denote by $\mathcal{A}ct(P)$ the multiset of all the *current activities* of P. For any component P, the *exit rate* from P will be the sum of the activity rates of all the activities enabled in P, i.e., $q(P) = \sum_{a \in \mathcal{A}ct(P)} r_a$, where hereafter r_a denotes the rate of activity a. If P enables more than one activity, $|\mathcal{A}ct(P)| > 1$, then the dynamic behaviour of the model is determined by a race condition. This has the effect of replacing the nondeterministic branching of the pure process algebra with probabilistic branching. The probability that a particular activity completes is given by the ratio of the activity rate to the exit rate from P.

The underlying CTMC. The derivation graph describing the possible behaviour of any PEPA component is the basis of the construction of the underlying Continuous Time Markov Chain (CTMC). To form the underlying CTMC a state is associated with each component of the derivative set of P ($ds(P)$) and the transitions between states are derived from the arcs of the derivation graph. The total transition rate between two states will be the sum of the activity rates labelling arcs connecting the corresponding nodes in the derivation graph. This use of the derivation graph is analogous to the use of the reachability graph in stochastic extensions of Petri nets such as SPNs [20]. We assume that the model is finite, i.e., the number of nodes in the derivation graph is finite. The following theorem holds [13].

THEOREM 1. *For any finite PEPA model $P \stackrel{\text{def}}{=} P_0$ with $ds(P) = \{P_0, \ldots, P_n\}$, if we define the stochastic process $X(t)$, such that $X(t) = P_i$ indicates that the system behaves as component P_i at time t, then $X(t)$ is a continuous time Markov chain.*

The *transition rate* between two components P_i and P_j, denoted $q(P_i, P_j)$, is the rate at which the system changes from behaving as component P_i to behaving as P_j. It is the sum of the activity rates labelling arcs which connect the node corresponding to P_i to the node corresponding to P_j in the derivation graph, i.e.,

$$q(P_i, P_j) = \textstyle\sum_{a \in \mathcal{A}ct(P_i|P_j)} r_a$$

where $P_i \neq P_j$ and $\mathcal{A}ct(P_i|P_j) = \{\!|\, a \in \mathcal{A}ct(P_i) |\ P_i \xrightarrow{a} P_j \,|\!\}$. Clearly if P_j is not a one-step derivative of P_i, $q(P_i, P_j) = 0$. The $q(P_i, P_j)$ (also denoted q_{ij}), are the off-diagonal elements of the infinitesimal generator matrix of the Markov process, $\mathbf{Q}$. Diagonal elements are formed as the negative sum of the non-diagonal elements of each row, i.e., $q_{ii} = -q(P_i)$. For any finite and irreducible PEPA model P, the steady-state distribution $\Pi(\cdot)$ exists and it may be found by solving the normalization equation and the global balance equations: $\sum_{P_i \in ds(P)} \Pi(P_i) = 1$ and $\Pi\mathbf{Q} = \mathbf{0}$. The *conditional transition rate* from P_i to P_j via an action type α is denoted $q(P_i, P_j, \alpha)$. This is the sum of the activity rates labelling arcs connecting the corresponding nodes in the derivation graph which are also labelled by the action type α. It is the rate at which a system behaving as component P_i evolves to behaving as component P_j as the result of completing a type α activity.

DEFINITION 1. (Total conditional transition rate) *For a PEPA component P and a set of possible derivatives $S \subseteq ds(P)$, the* total conditional transition rate *from P to S, denoted $q[P, S, \alpha]$, is defined as*

$$q[P, S, \alpha] = \sum_{P' \in S} q(P, P', \alpha)$$

where $q(P, P', \alpha) = \sum_{P \xrightarrow{(\alpha, r_\alpha)} P'} r_\alpha$.

EXAMPLE 1. *We consider a model of a system consisting of a producer-like process and a consumer-like process. The former alternates a thinking-phase, a computing phase and*

Producer	Consumer
$Q_{Think} \stackrel{def}{=} (\tau, \delta).Q_{Compute}$	$P_{Empty} \stackrel{def}{=} (tr, \top).P_1$
$Q_{Compute} \stackrel{def}{=} (comp, \varepsilon).Q_{Send} + (\tau, \varphi).Q_{Error}$	$P_i \stackrel{def}{=} (\tau, i\mu).P_{i+1} + (tr, \top).P_{i+1} + (send, \gamma).P_{Wait} \quad 1 \leq i < N$
$Q_{Send} \stackrel{def}{=} (tr, \eta).Q_{Think}$	$P_N \stackrel{def}{=} (tr, \top).P_N + (send, \gamma).P_{Wait}$
$Q_{Error} \stackrel{def}{=} (\tau, \xi).Q_{Recovery}$	$P_{Wait} \stackrel{def}{=} (\tau, \nu).P_{Empty}$
$Q_{Recovery} \stackrel{def}{=} (\tau, \delta).Q_{Compute}$	

Table 2: PEPA equations for the models of Example 1

then sends the output to the consumer. During the computation phase some errors may arise and hence, after a recovering phase, the computation must be newly done. The consumer enqueues the jobs worked by the producer and transmits them in batches. A maximum buffer size of N jobs is set and, in case of saturation, further arrivals are lost. Each of the jobs waiting to be sent can generate other jobs (e.g., because the time spent in the queue causes an update of the information stored in the jobs). The transition diagrams of the producer and the consumer are shown in Figure 1-(A) and (B), respectively, while their encoding in PEPA is shown in Table 2. The whole system is modelled by:

$$S \stackrel{\text{def}}{=} P_{Empty} \underset{\{tr\}}{\bowtie} Q_{Think}\,.$$

The independence and exponential distribution of the transition times are assumed.

3. CONTEXTUAL LUMPABILITY

In order to tackle the state space explosion problem, a variety of model simplification techniques have been proposed at the level of the Markov process. Among them, *aggregation*, can be formalized in terms of equivalence relations over the state space of the model. Any such equivalence induces a *partition* on the state space of the model and aggregation is achieved by aggregating equivalent states into macro-states, thus reducing the overall state space. If the original state space is $\{X_0, X_1, \ldots, X_n\}$ then the aggregated state space is some $\{X_{[0]}, X_{[1]}, \ldots, X_{[N]}\}$, where $N \leq n$, ideally $N \ll n$. In general, when a CTMC is aggregated the resulting stochastic process will not have the Markov property.
However if the partition can be shown to satisfy the so-called *lumpability* condition [14, 2], the property is preserved and the aggregation is said to be *exact*.

Definition 2. (Lumpability) *A CTMC, $\{X_i\}$, is* lumpable *with respect to some partition $\chi = \{X_{[k]}\}$ if for any $X_{[k]}, X_{[l]} \in \chi$ with $k \neq l$ and $X_i, X_j \in X_{[k]}$,*

$$q(X_i, X_{[l]}) = q(X_j, X_{[l]})$$

where $q(X_i, X_{[l]})$ is the aggregated transistion rate from X_i to all states in $X_{[l]}$, i.e., $q(X_i, X_{[l]}) = \sum_{X_m \in X_{[l]}} q(X_i, X_m)$.

Since an equivalence relation $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$ over PEPA components partitions the set of components $\mathcal{C}$, if it is restricted to the derivative set of any component P, the relation partitions this set. Let $ds(P)/\mathcal{R}$ denote the set of equivalence classes generated in this way. Clearly, this induces also a partition on the state space of the CTMC underlying P. Hence, at the level of PEPA models, the concept of lumpability can be expressed as a relation between PEPA components.

Definition 3. (Lumpable relation) *A relation $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$ over PEPA components is* lumpable *if for any component P, $ds(P)/\mathcal{R}$ induces a lumpable partition on the state space of the CTMC corresponding to P.*

By definition of lumpability, a lumpable relation is an equivalence relation. Moreover, the union of lumpable relations is itself a lumpable relation.

Proposition 1. *Let I be a set of indices and $\mathcal{R}_i$ be a lumpable relation for all $i \in I$. Then the union, $\mathcal{R} = \cup_{i \in I} \mathcal{R}_i$, is also a lumpable relation.*

Proof. Given a component P, any equivalence relation over $\mathcal{C}$ will partition the set $ds(P)$ into equivalences classes. Let $ds(P)/\mathcal{R}$ and $ds(P)/\mathcal{R}_i$ denote these sets of equivalence classes for $\mathcal{R}$ and each $\mathcal{R}_i$, respectively. For $i \in I$, any equivalence class $T^i \in ds(P)/\mathcal{R}_i$ is wholly contained within some equivalence class $T \in ds(P)/\mathcal{R}$; moreover there is some set J^i such that $T = \cup_{j \in J^i} T^i_j$. Let S and T be two distinct equivalence classes in $ds(P)/\mathcal{R}$ and $X, Y \in S$. Since $(X, Y) \in \mathcal{R}$, we have $(X, Y) \in \mathcal{R}_i$ for some $i \in I$. Therefore:

$$q(X, T) = \sum_{j \in J^i} q(X, T^i_j) = \sum_{j \in J^i} q(Y, T^i_j) = q(Y, T)$$

□

One of the main advantages of considering models derived from a process algebra such as PEPA is the possibility of establishing useful algebraic compositional properties of the behaviours of systems. In this paper we focus on *contextuality*. In particular, we restrict our attention to standard *evaluation* (or *static* [19]) contexts which are PEPA components with a hole that does not occur under a prefix or a choice operator. Formally, an evaluation context is a term with a hole $[\cdot]$ defined by the grammar:

$$C[\cdot] ::= [\cdot] \;\mid\; [\cdot] \underset{L}{\bowtie} P \;\mid\; P \underset{L}{\bowtie} [\cdot] \;\mid\; [\cdot]/L$$

An equivalence relation is contextual if substituting an equivalent component within a context gives rise to an equivalent model, e.g., if P is equivalent to P', then $P \underset{L}{\bowtie} Q$ is equivalent to $P' \underset{L}{\bowtie} Q$.

Definition 4. (Contextuality) *A relation $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$ over PEPA components is* contextual *if for all PEPA components P, Q such that $(P, Q) \in \mathcal{R}$ and for all contexts $C[\cdot]$,*

$$(C[P], C[Q]) \in \mathcal{R}$$

Finally, we are interested in equivalence relations preserving the current action types of equivalent terms.

DEFINITION 5. (Current action type preservation) *A relation $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$ over PEPA components is* current action type preserving *if for all PEPA components P, Q such that $(P, Q) \in \mathcal{R}$, $\mathcal{A}(P) = \mathcal{A}(Q)$.*

The notion of *contextual lumpability* for PEPA models is defined as follows.

DEFINITION 6. (Contextual lumpability) Contextual lumpability*, denoted $\cong_l$, is the largest contextual, current action type preserving, lumpable relation over PEPA terms.*

Contextual lumpability is thus a congruence with respect to the cooperation and hiding operators.

EXAMPLE 2. *In order to support the intuition of Definition 6, we illustrate the following toy-example. Consider two PEPA terms: $P \stackrel{\text{def}}{=} (a, \lambda).P$ and $Q \stackrel{\text{def}}{=} (a, \mu).Q$. We show that a necessary and sufficient condition for a relation $\mathcal{R}$ to be a contextual lumpability containing the pair (P, Q) is that $\lambda = \mu$. The sufficiency is trivial, we prove that it is necessary. Let us suppose that $\lambda \neq \mu$ and that $P \cong_l Q$. Among the possible contexts we choose the following: $C[\cdot] \stackrel{\text{def}}{=} R \underset{\{a\}}{\bowtie} [\cdot]$ where $R = (a, \top).R'$ and $R' = (b, \gamma).R$. Let $C'[\cdot] \stackrel{\text{def}}{=} R' \underset{\{a\}}{\bowtie} [\cdot]$. By contextuality, we have that $P \cong_l Q$ implies $C[P] \cong_l C[Q]$. The set of derivatives of $C[P]$ and $C[Q]$ is $\{C[P], C[Q], C'[P], C'[Q]\}$. Since $\mathcal{A}(C[P]) = \mathcal{A}(C[Q]) = \{a\}$ and $\mathcal{A}(C'[P]) = \mathcal{A}(C'[Q]) = \{b\}$, there are two equivalence classes, i.e., $\{C[P], C[Q]\}$ and $\{C'[P], C'[Q]\}$. It is easy to see that this is a lumping in the underlying Markov chain only if $\lambda = \mu$. Hence, there cannot be any contextual lumpability such that $P \cong_l Q$ if $\lambda \neq \mu$.*

4. COINDUCTIVE CHARACTERIZATION

In a process algebra, actions, rather than states, are taken to capture the behaviour of a system or model. This leads to a formally defined notion of equivalence, bisimulation, in which components are regarded as equal if, under observation, they appear to perform exactly the same actions. In this section we introduce a bisimulation-like relation, called *lumpable bisimulation*, for PEPA models which provides a coinductive characterisation of contextual lumpability.

Lumpable bisimulation is developed in the style of Larsen and Skou's bisimulation [15]: in PEPA transition rates are used analogously to probabilities in their probabilistic process algebra.

Two PEPA components are *lumpably bisimilar* if there is an equivalence relation between them such that, for any action type α different from τ, the total conditional transition rates from those components to any equivalence class, via activities of this type, are the same.

DEFINITION 7. (Lumpable bisimulation) *An equivalence relation over PEPA components, $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$, is a* lumpable bisimulation *if whenever $(P, Q) \in \mathcal{R}$ then for all $\alpha \in \mathcal{A}$ and for all $S \in \mathcal{C}/\mathcal{R}$ such that*

- *either $\alpha \neq \tau$,*
- *or $\alpha = \tau$ and $P, Q \notin S$,*

it holds

$$q[P, S, \alpha] = q[Q, S, \alpha]$$

where $q[\cdot]$ is as introduced in Definition 1.

REMARK 1. (Comparison with strong equivalence) *According to [13], an equivalence relation $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$ is a* strong equivalence *if whenever $(P, Q) \in \mathcal{R}$ then for all $\alpha \in \mathcal{A}$ and for all $S \in \mathcal{C}/\mathcal{R}$ we have $q[P, S, \alpha] = q[Q, S, \alpha]$. Notice that this definition is stricter than that of lumpable bisimulation because the latter allows arbitrary activities with type τ among components belonging to the same equivalence class. Later on, we will study the properties of lumpable bisimulation showing that it is a congruence with respect to the cooperation and hiding operators, but not for the choice and the prefix. Roughly speaking, we can say that in general the lumpable bisimulation induces a coarser lumping than the strong equivalence but has stricter congruence properties. Nevertheless, although the lumpable bisimulation is contextual only for the particular class of evaluation contexts, this is what one expects in most practical cases in which different components cooperate using the $\bowtie$ operator.*

REMARK 2. (Comparison with prior weak bisimulations) *In [4] a notion of weak bisimulation for CTMCs is introduced. This is based on the idea that the time-abstract behaviour of equivalent states is weakly bisimilar and that the "relative speed" of these states to move to a different equivalence class is equal. The authors show that this intuition is captured by a definition of weak-bisimulation which resembles our notion of lumpable bisimulation if we ignore action types and labels. Differently from our setting, in [4] the authors deal with CTMC without any notion of compositionality and hence of contextuality. Compositionality is considered in [7, 8, 9], where definitions of contextual weak bisimilarities for stochastic process algebra based on the classical concept of weak action are proposed. Our approach shares with these bisimilarities the idea of ignoring the rates for non-synchronizing (labelled τ) transitions between a state and the others belonging to the same equivalence class. With respect to these results, we explicitly study the relationships between our lumpable bisimulation at the process algebra level and the induced lumping of the underlying Markov chains. This leads to a coinductive characterisation of contextual lumpability that is novel, to the best of our knowledge. Moreover, it is important to observe that, due to the different nature of synchronisation, the results proposed for PEPA can not be applied to the process calculi considered in [7, 8, 9], and vice versa.*

It is clear that the identity relation is a lumpable bisimulation. We are interested in the relation which is the largest lumpable bisimulation, formed by the union of all lumpable bisimulations. However, it is not straightforward to see that this will indeed be a lumpable bisimulation.

The following proposition states that any union of lumpable bisimulations generates a lumpable bisimulation.

PROPOSITION 2. *Let I be a set of indices and $\mathcal{R}_i$ be a lumpable bisimulation for all $i \in I$. Then the transitive closure of their union, $\mathcal{R} = (\cup_{i \in I} \mathcal{R}_i)^*$, is also a lumpable bisimulation.*

Based on the above result we define the maximal lumpable bisimulation as the union of all lumpable bisimulations.

DEFINITION 8. (Lumpable bisimilarity) *Two PEPA components P and Q are* lumpably bisimilar*, written $P \approx_l Q$, if $(P, Q) \in \mathcal{R}$ for some lumpable bisimulation $\mathcal{R}$, i.e.,*

$$\approx_l = \bigcup \{\mathcal{R} \mid \mathcal{R} \text{ is a lumpable bisimulation}\}.$$

$\approx_l$ *is called* lumpable bisimilarity *and it is the largest symmetric lumpable bisimulation over PEPA components.*

It is easy to show that lumpable bisimilarity is a congruence for evaluation contexts.

PROPOSITION 3. *If* $P_1 \approx_l P_2$ *then*

- *for all* $L \subseteq \mathcal{A}$, $P_1 \underset{L}{\bowtie} Q \approx_l P_2 \underset{L}{\bowtie} Q$;
- $P_1/L \approx_l P_2/L$.

PROOF. The proof is analogous to that of Proposition 8.3.1, items 3 and 4, in [13]. □

Moreover, lumpable bisimilarity is a lumpable relation.

PROPOSITION 4. *For all PEPA components* P, Q *such that* $P \approx_l Q$ *and for all* $S \in \mathcal{C}/\approx_l$ *such that* $P, Q \notin S$, $q(P, S) = q(Q, S)$.

PROOF. Let P, Q such that $P \approx_l Q$ and $S \in \mathcal{C}/\approx_l$ such that $P, Q \notin S$. By Definition 7, $q[P, S, \alpha] = q[Q, S, \alpha]$ for all $\alpha \in \mathcal{A}$. Hence, $q(P, S) = \sum_{\alpha \in \mathcal{A}} q[P, S, \alpha] = \sum_{\alpha \in \mathcal{A}} q[Q, S, \alpha] = q(Q, S)$. □

The next Theorem shows that lumpable bisimilarity is a characterization of contextual lumpability.

THEOREM 2. *Let* P *and* Q *be two PEPA components. It holds that:* $P \approx_l Q$ *if and only if* $P \cong_l Q$.

PROOF. [Soundness]: $\approx_l \subseteq \cong_l$. Indeed, $\approx_l$ is symmetric (since, by Definitions 7 and 8, $\approx_l$ is an equivalence relation), contextual (by Proposition 3), current action type preserving and a lumpable relation (by Proposition 4).

[Completeness]: $\cong_l \subseteq \approx_l$. It is sufficient to show that $\mathcal{R} = \{(P, Q) |\ P \cong_l Q\}$ is a lumpable bisimulation. Let $(P, Q) \in \mathcal{R}$ with $P \cong_l Q$. Then $\mathcal{A}(P) = \mathcal{A}(Q)$.

We first prove the following claim.

CLAIM 1. *Let* P *and* Q *be two PEPA components such that their derivation graphs,* $\mathcal{D}(P)$ *and* $\mathcal{D}(Q)$*, are graph isomorphic. Then* $P \cong_l Q$.

Proof of Claim. From the fact that $\mathcal{D}(P)$ and $\mathcal{D}(Q)$ are graph isomorphic, there exists a bijection f from $ds(P)$ to $ds(Q)$ such that $Q = f(P)$ and $P \xrightarrow{a} P'$ if and only if $f(P) \xrightarrow{a} f(P')$. Let $S \in \mathcal{C}/\cong_l$. It trivially follows that $\{(P', f(P')) |\ P' \in ds(P)\}$ is a lumpable bisimulation. Hence, by the fact that $Q = f(P)$ and that lumpable bisimulation implies contextual lumpability, we have $P \cong_l Q$. □

We are now in position to prove that for any $\alpha \neq \tau$ and $S \in \mathcal{C}/\mathcal{R}$, $q[P, S, \alpha] = q[Q, S, \alpha]$.

For a component P, let $\vec{\mathcal{A}}(P) = \cup_{P_i \in ds(P)} \mathcal{A}(P_i)$. Let $\alpha \in \mathcal{A}(P) = \mathcal{A}(Q)$. Let $\bar{\alpha} \notin \vec{\mathcal{A}}(P) \cup \vec{\mathcal{A}}(Q) = \{\alpha, \beta_1, \ldots, \beta_n\}$ with $\bar{\alpha} \neq \tau$. Consider the context

$$\begin{array}{lcl} C[\cdot] & \stackrel{\text{def}}{=} & R \underset{L}{\bowtie} [\cdot] \\ R & \stackrel{\text{def}}{=} & (\alpha, \top).R' + (\bar{\alpha}, r).R' \\ R' & \stackrel{\text{def}}{=} & (\alpha, \top).R' + (\beta_1, \top).R' + \cdots + (\beta_n, \top).R' \end{array}$$

for some rate r and $L = \{\alpha, \beta_1, \ldots, \beta_n\}$. Let $S \in \mathcal{C}/\mathcal{R}$. By definition of $\mathcal{R}$ and by contextuality, $S_C = \{R' \underset{L}{\bowtie} X |\ X \in S\} \subseteq T \in \mathcal{C}/\cong_l$ for some T. Any one-step derivative of $C[P]$ (resp. $C[Q]$) has one of the following forms:

- $R \underset{L}{\bowtie} P'$ with $P \xrightarrow{(\tau, r')} P'$ for some rate r' (resp. $R \underset{L}{\bowtie} Q'$ with $Q \xrightarrow{(\tau, r')} Q'$ for some rate r');
- $R' \underset{L}{\bowtie} P'$ with $P \xrightarrow{(\alpha, r')} P'$ for some rate r' (resp. $R' \underset{L}{\bowtie} Q'$ with $Q \xrightarrow{(\alpha, r')} Q'$ for some rate r');
- $R' \underset{L}{\bowtie} P$ (resp. $R' \underset{L}{\bowtie} Q$).

Since $\mathcal{A}(R' \underset{L}{\bowtie} P) \neq \mathcal{A}(R \underset{L}{\bowtie} P')$ for all $P' \in ds(P) \cup ds(Q)$, we have that $R \underset{L}{\bowtie} P' \notin T$, and in particular, it holds $R \underset{L}{\bowtie} P, R \underset{L}{\bowtie} Q \notin T$. Since for any $P' \in \mathcal{C}$, the derivation graphs of $R' \underset{L}{\bowtie} P'$ and P' are isomorphic, by the above claim we have $R' \underset{L}{\bowtie} P' \cong_l P'$ for all $P' \in \mathcal{C}$. Hence, $S = T$. From $P \cong_l Q$, we have $C[P] \cong_l C[Q]$ and hence $q(C[P], T) = q(C[Q], T)$. Moreover,

$$\begin{array}{rcl} q(C[P], T) & = & q(C[P], T, \alpha) + q(C[P], T, \bar{\alpha}) \\ & = & q(C[P], T, \alpha) + r \\ & = & q[P, S, \alpha] + r. \end{array}$$

Similarly, we can prove that $q(C[Q], T) = q[Q, S, \alpha] + r$. Hence, $q[P, S, \alpha] = q[Q, S, \alpha]$.

We now prove that for any $S \in \mathcal{C}/\mathcal{R}$ such that $P, Q \notin S$, $q[P, S, \tau] = q[Q, S, \tau]$. Let $S \in \mathcal{C}/\mathcal{R}$ such that $P, Q \notin S$. By definition of $\mathcal{R}$, $S \in \mathcal{C}/\cong_l$. From the fact that $P \cong_l Q$, we have $q(P, S) = q(Q, S)$. Precisely,

$$\begin{array}{rcl} q(P, S) & = & \sum_{\alpha \in \mathcal{A}(P)} q[P, S, \alpha] + q[P, S, \tau] \\ q(Q, S) & = & \sum_{\alpha \in \mathcal{A}(Q)} q[Q, S, \alpha] + q[Q, S, \tau] \end{array}$$

Since $\mathcal{A}(P) = \mathcal{A}(Q)$ and, as proved above, for any $\alpha \neq \tau$ and $S \in \mathcal{C}/\mathcal{R}$, $q[P, S, \alpha] = q[Q, S, \alpha]$, we have $q[P, S, \tau] = q[Q, S, \tau]$. □

EXAMPLE 3. *Let us consider the model depicted in Example 1. If we consider the producer component (Figure 1-(A)) we notice that* $Q_{Think} \approx_l Q_{Recovery}$ *and hence they belong to the same equivalence class, say* $[Q']$. *The lumped producer component is shown in Figure 2-(A). If we consider the consumer, we observe that* $P_1 \approx_l P_2 \approx_l \cdots \approx_l P_N$ *and let* $[P']$ *be the associated equivalence class. Figure 2-(B) shows the lumped consumer model. Notice that, while we can say that* Q_{Think} *is strongly equivalent to* $Q_{Recovery}$ *(see the analysis of a similar model in [11]) we can show that* P_i *is not strongly equivalent to* P_j, $i \neq j$. *Indeed, the total rate of the action with type* τ *outgoing from term* P_i *is* $i\mu$ *for* $1 \leq i < N$, *and is* 0 *if* $i = N$, *i.e., different for each term* $P_1, \ldots, P_N$. *Therefore, the joint model obtained by applying lumpable bisimilarity has less states than that obtained by strong equivalence.*

5. AN ALGORITHM FOR CONTEXTUAL LUMPABILITY

In [23], Valmari and Franceschinis proposed an algorithm for computing lumpability over Markov Chains, i.e., directed weighted graphs. In particular, the algorithm exploits a partition refinement strategy, similar to that of Paige-Tarjan's algorithm for bisimulation [21], enriched with sorting of classes. In this section we exploit Valmari and Franceschinis' algorithm to design a procedure for computing the contextual lumpability. We first reformulate the notion of lumpable bisimulation.

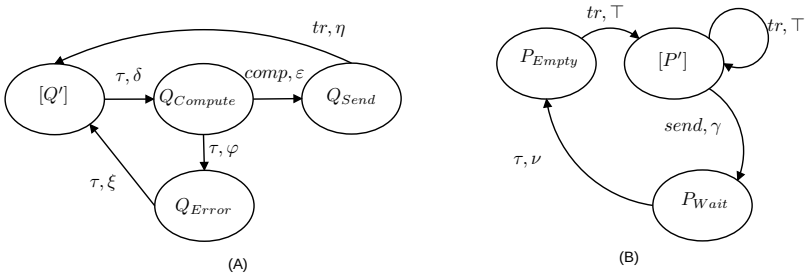

Figure 2: (A)- Lumping of the producer model of Figure 1-(A) (B)- Lumping of the consumer model of Figure 1-(B)

LEMMA 1. *An equivalence relation over PEPA components, $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$, is a* lumpable bisimulation *if the following conditions hold:*

- *for each $\alpha \in \mathcal{A}$ with $\alpha \neq \tau$, for each $S, S' \in \mathcal{C}/\mathcal{R}$ if $P, Q \in S'$, then $q[P, S, \alpha] = q[Q, S, \alpha]$;*
- *for each $S, S' \in \mathcal{C}/\mathcal{R}$ with $S' \neq S$, if $P, Q \in S'$, then $q[P, S, \tau] = q[Q, S, \tau]$.*

PROOF. This is an immediate consequence of the fact that $(P, Q) \in \mathcal{R}$ implies that there exists $S' \in \mathcal{C}/\mathcal{R}$ such that $P, Q \in S'$. Hence, $(P, Q) \in \mathcal{R}$ and $P, Q \notin S$ implies that there exists $S' \in \mathcal{C}/\mathcal{R}$ such that $P, Q \in S'$ and $S' \neq S$. □

Notice that $\mathcal{C}$ is infinite, since it contains all PEPA components. However, when we are interested in testing $P \approx_l Q$ it is sufficient to consider the set of components $ds(P) \cup ds(Q)$, which can be finite.

DEFINITION 9. (Lumpable bisimulation over a set of components) *Let $\mathcal{D}$ be a set of PEPA components, $\mathcal{R} \subseteq \mathcal{D} \times \mathcal{D}$ is a* lumpable bisimulation over $\mathcal{D}$ *if:*

- *for each $\alpha \in \mathcal{A}$ with $\alpha \neq \tau$, for each $S, S' \in \mathcal{D}/\mathcal{R}$ if $P, Q \in S'$, then $q[P, S, \alpha] = q[Q, S, \alpha]$;*
- *for each $S, S' \in \mathcal{D}/\mathcal{R}$ with $S' \neq S$, if $P, Q \in S'$, then $q[P, S, \tau] = q[Q, S, \tau]$.*

Of course we are interested in lumpable bisimulations only in the case of sets of components $\mathcal{D}$ which are *closed under derivative behaviours*, i.e., such that $\mathcal{D} = ds(\mathcal{D})$, where $ds(\mathcal{D}) = \cup_{P \in \mathcal{D}} ds(P)$.

LEMMA 2. *Let $\mathcal{D}$ be closed under derivative behaviours, i.e., $\mathcal{D} = ds(\mathcal{D})$.*

(a) If $\mathcal{R}$ is a lumpable bisimulation, then $\mathcal{R} \cap (\mathcal{D} \times \mathcal{D})$ is a lumpable bisimulation over $\mathcal{D}$;

(b) If $\mathcal{R}$ is a lumpable bisimulation over $\mathcal{D}$, then $\mathcal{R} \cup Id$, where Id is the identity relation over $\mathcal{C}$, is a lumpable bisimulation.

PROOF. First we prove item (a). Let $\mathcal{R}^{\mathcal{D}} = \mathcal{R} \cap (\mathcal{D} \times \mathcal{D})$. Let $\alpha \in \mathcal{A} \setminus \{\tau\}$, $S, S' \in \mathcal{D}/\mathcal{R}^{\mathcal{D}}$, and $P, Q \in S'$. Since $P, Q \in S'$, there exists $\overline{S'} \in \mathcal{C}/\mathcal{R}$ such that $P, Q \in \overline{S'}$ (i.e., $S' = \overline{S'} \cap \mathcal{D}$). Moreover, there exists $\overline{S}$ such that $S = \overline{S} \cap \mathcal{D}$. Since $\mathcal{D} = ds(\mathcal{D})$ we have that for each $O \in \mathcal{D}$ it holds $q[O, S, \alpha] = q[O, \overline{S}, \alpha]$. So, since $P, Q \in \mathcal{D}$, $P, Q \in \overline{S'} \in \mathcal{C}/\mathcal{R}$, and $\mathcal{R}$ is a lumpable bisimulation, we get $q[P, S, \alpha] = q[P, \overline{S}, \alpha] = q[Q, \overline{S}, \alpha] = q[Q, S, \alpha]$.

As far as τ is concerned, since $P, Q \in S' \subseteq \overline{S'}$ and $S' = \overline{S'} \cap \mathcal{D} \neq S = \overline{S} \cap \mathcal{D}$ we have $\overline{S'} \neq \overline{S}$. So, again, since $P, Q \in \mathcal{D}$, $P, Q \in \overline{S'} \in \mathcal{C}/\mathcal{R}$, and $\mathcal{R}$ is a lumpable bisimulation, we get $q[P, S, \tau] = q[P, \overline{S}, \tau] = q[Q, \overline{S}, \tau] = q[Q, S, \tau]$.

We now prove item (b). Let $\mathcal{R}' = \mathcal{R} \cup Id$. We have that $\mathcal{C}/\mathcal{R} = \mathcal{D}/\mathcal{R} \cup \{\{O\} \mid O \in \mathcal{C} \setminus \mathcal{D}\}$. So if $|S'| = 1$, we immediately get that for each α (including τ), for each $S \in \mathcal{C}/\mathcal{R}$, it holds that if $P, Q \in S'$, then $q[P, S, \alpha] = q[Q, S, \alpha]$, since $P = Q$.

On the other hand, if $|S'| > 1$, then $S' \in \mathcal{D}/\mathcal{R}$. Since $\mathcal{D} = ds(\mathcal{D})$ we get that for each $O \in \mathcal{D}$ and for each $S \in (\mathcal{C}/\mathcal{R}) \setminus (\mathcal{D}/\mathcal{R})$ it holds $q[O, S, \alpha] = 0$. Hence, for each $P, Q \in S'$ we get that:

- for each $\alpha \in \mathcal{A} \setminus \{\tau\}$, for each $S \in \mathcal{C}/\mathcal{R}$ it holds $q[P, S, \alpha] = q[Q, S, \alpha]$;
- for each $S \in \mathcal{C}/\mathcal{R}$ with $S' \neq S$ it holds $q[P, S, \tau] = q[Q, S, \tau]$.

This proves that $\mathcal{R}'$ is a lumpable bisimulation. □

The largest bisimulation over a set of components closed under derivative behaviours can be characterised as follows.

PROPOSITION 5. *Let $\mathcal{D}$ be a set of PEPA components such that $\mathcal{D} = ds(\mathcal{D})$. The largest lumpable bisimulation over $\mathcal{D}$ is the union of all lumpable bisimulations over $\mathcal{D}$ and it coincides with $\approx_l \cap (\mathcal{D} \times \mathcal{D})$.*

PROOF. This is an immediate consequence of Lemma 2. □

In particular, when we are interested in $P \approx_l Q$, the following lemma characterises lumpable bisimulations exploiting the smallest possible sets of components.

LEMMA 3. *Let P and Q be two PEPA components. $P \approx_l Q$ if and only if $(P, Q) \in \mathcal{R}$ for some lumpable bisimulation $\mathcal{R}$ over $ds(P) \cup ds(Q)$.*

PROOF. Let $\mathcal{D} = ds(P) \cup ds(Q)$. We have $ds(\mathcal{D}) = \mathcal{D}$.

Hence if $P \approx_l Q$ then there exists a lumpable bisimulation $\mathcal{R}$ such that $(P, Q) \in \mathcal{R}$. By Lemma 2 (a) we have that $(P, Q) \in \mathcal{R} \cap (\mathcal{D} \times \mathcal{D})$ and $\mathcal{R} \cap (\mathcal{D} \times \mathcal{D})$ is a lumpable bisimulation over $\mathcal{D}$.

On the other hand, if there exists $\mathcal{R}$ which is a lumpable bisimulation over $\mathcal{D}$ such that $(P, Q) \in \mathcal{R}$, then, by Lemma 2 (b), $\mathcal{R} \cup Id$ is a lumpable bisimulation and $(P, Q) \in \mathcal{R}$, i.e., $P \approx_l Q$. □

Hence, from now on we focus on sets of components $\mathcal{D}$ such that $ds(\mathcal{D}) = \mathcal{D}$.

A *weighted directed graph* is a triple $G = (D, \Delta, W)$ such that:

- D is a set of nodes;
- $\Delta \subseteq D \times D$ is a set of edges;
- $W : \Delta \to \mathbb{R}$ is a weight function which assigns a real value to each edge.

If $G = (D, \Delta, W)$ is a weighted directed graph and $P, Q \in D$ are nodes, with a slight abuse of notation, $W(P, Q)$ is used to denote $W(\langle P, Q\rangle)$ when $\langle P, Q\rangle \in \Delta$ and 0 otherwise. Moreover, if $S \subseteq D$ is a set of nodes, $W(P, S)$ is $\sum_{Q \in S} W(P, Q)$.

Given a set $\mathcal{D}$ of PEPA components we introduce a weighted directed graph $G^{\mathcal{D}}_\alpha = (\mathcal{D}, \Delta^{\mathcal{D}}_\alpha, W^{\mathcal{D}}_\alpha)$, for each action type $\alpha \in \mathcal{A}$. We call such a graph the *conditional CTMC of $\mathcal{D}$ under action type α.*

Definition 10. (Conditional CTMC of $\mathcal{D}$ under action type α) *Let $\mathcal{D}$ be a set of components such that $\mathcal{D} = ds(\mathcal{D})$ and let $\alpha \in \mathcal{A}$ be an action type. We define the* conditional CTMC of $\mathcal{D}$ under action type α*, denoted by $G^{\mathcal{D}}_\alpha = (\mathcal{D}, \Delta^{\mathcal{D}}_\alpha, W^{\mathcal{D}}_\alpha)$, as the weighted directed graph such that:*

- *$\mathcal{D}$ is the set of nodes;*
- *$\Delta^{\mathcal{D}}_\alpha = \{\langle C_i, C_j\rangle | C_i, C_j \in \mathcal{D},\ C_i \xrightarrow{(\alpha, r)} C_j$ for some $r\}$ is the set of edges;*
- *$W^{\mathcal{D}}_\alpha : \Delta^{\mathcal{D}}_\alpha \to \mathbb{R}$ is the weight function defined as $W^{\mathcal{D}}_\alpha(\langle C_i, C_j\rangle) = q[C_i, \{C_j\}, \alpha]$.*

In the above definition $q[C_i, \{C_j\}, \alpha]$ is the sum of the activity rates labelling arcs connecting C_i to C_j and having action type α, also in the case $C_j = C_i$.

Notice that $W^{\mathcal{D}}_\alpha(C_i, S)$ is equal to $q[C_i, S, \alpha]$.

Moreover, we introduce the weighted directed graph called the *variant of* $G^{\mathcal{D}}_\alpha$ and denoted by $\widehat{G}^{\mathcal{D}}_\alpha$. In the algorithm $G^{\mathcal{D}}_\alpha$ will be used for all $\alpha \neq \tau$, while $\widehat{G}^{\mathcal{D}}_\tau$ will replace $G^{\mathcal{D}}_\tau$.

Definition 11. (Variant of $G^{\mathcal{D}}_\alpha$) *Let $\mathcal{D}$ be a set of components such that $\mathcal{D} = ds(\mathcal{D})$ and let $\alpha \in \mathcal{A}$ be an action type. We define the* variant of the conditional CTMC of $\mathcal{D}$ under action type α*, denoted by $\widehat{G}^{\mathcal{D}}_\alpha = (\mathcal{D}, \widehat{\Delta}^{\mathcal{D}}_\alpha, \widehat{W}^{\mathcal{D}}_\alpha)$, as the weighted directed graph such that:*

- *$\mathcal{D}$ is the set of nodes;*
- *$\widehat{\Delta}^{\mathcal{D}}_\alpha = \{\langle C_i, C_i\rangle \mid C_i \in \mathcal{D}\} \cup \Delta^{\mathcal{D}}_\alpha$ is the set of edges;*
- *$\widehat{W}^{\mathcal{D}}_\alpha : \widehat{\Delta}^{\mathcal{D}}_\alpha \to \mathbb{R}$ is the weight function defined as*

$$\widehat{W}^{\mathcal{D}}_\alpha(\langle C_i, C_j\rangle) = \begin{cases} W^{\mathcal{D}}_\alpha(C_i, C_j) & \text{if } C_i \neq C_j \\ -W^{\mathcal{D}}_\alpha(C_i, \mathcal{D} \setminus \{C_i\}) & \text{if } C_i = C_j \end{cases}$$

We consider the following definition taken from [23].

Definition 12. (Relation compatible with G) *Let $G = (D, \Delta, W)$ be a weighted directed graph. An equivalence relation R over D is* compatible with G *if for each $S, S' \in D/R$ if $P, Q \in S'$, then $W(P, S) = W(Q, S)$.*

Given $G = (D, \Delta, W)$ with D finite, an initial equivalence relation I over D, and an action type α, we consider the problem of finding the largest equivalence relation $\mathcal{R} \subseteq \mathcal{D} \times \mathcal{D}$ compatible with $G^{\mathcal{D}}_\alpha$.

Lemma 4. *Let $\mathcal{D}$ be a set of components such that $\mathcal{D} = ds(\mathcal{D})$. Let $\mathcal{R}$ be an equivalence relation over $\mathcal{D}$ such that $\approx_l \cap(\mathcal{D} \times \mathcal{D}) \subseteq \mathcal{R}$. Let $\alpha \in \mathcal{A} \setminus \{\tau\}$ and $\mathcal{R}' \subseteq \mathcal{R}$ be the largest equivalence relation compatible with $G^{\mathcal{D}}_\alpha$. It holds that:*

$$\approx_l \cap(\mathcal{D} \times \mathcal{D}) \subseteq \mathcal{R}'$$

Proof. We start by proving the following general result.

Claim 2. *Let $G = (D, \Delta, W)$ be a weighted directed graph. Let R be an equivalence relation over D. Let $V_1, V_2 \subseteq R$ be two equivalence relations included in R and compatible with G. Let $V_1 \sqcup V_2$ be the smallest equivalence relation such that $V_1, V_2 \subseteq V_1 \sqcup V_2$. It holds that $V_1 \sqcup V_2 \subseteq R$ and $V_1 \sqcup V_2$ is compatible with G.*

Proof of Claim. We recall from the theory of lattices of equivalence relations that $V_1 \sqcup V_2$ is the transitive closure of $V_1 \cup V_2$. In other terms $(P, Q) \in V_1 \sqcup V_2$ if and only if there exists a finite sequence $P_1, \ldots, P_n$ such that $P_1 = P$, $P_n = Q$, and for each $1 \leq i \leq n-1$ it holds $(P_i, P_{i+1}) \in V_{k(i)}$ with $k(i) \in \{1, 2\}$. Similarly, as far as $D/(V_1 \sqcup V_2)$ is concerned, $S \in D/(V_1 \sqcup V_2)$ can be seen both as union of blocks of D/V_1 and as union of blocks of D/V_2, i.e., there exist $S_1, \ldots, S_m \in D/V_1$ ($T_1, \ldots, T_p \in D/V_2$) such that $S = \cup_{i=1}^m S_i$ ($S = \cup_{i=1}^p T_i$, respectively).

It trivially holds that $V_1 \sqcup V_2 \subseteq R$, since $V_1, V_2 \subseteq R$ and $V_1 \sqcup V_2$ is the smallest equivalence relation containing both V_1 and V_2.

We have to prove that $V_1 \sqcup V_2$ is compatible with G. Let $S, S' \in D/(V_1 \sqcup V_2)$ and $P, Q \in S'$. This means that $(P, Q) \in V_1 \sqcup V_2$, i.e., there exists a finite sequence $P_1, \ldots, P_n$ such that $P_1 = P$, $P_n = Q$, and for each $1 \leq i \leq n-1$ it holds $(P_i, P_{i+1}) \in V_{k(i)}$ with $k(i) \in \{1, 2\}$. By induction on n we prove that $W(P, S) = W(Q, S)$.

Basis $n = 2$. This means that either $(P, Q) \in V_1$ or $(P, Q) \in V_2$. It is not restrictive to assume $(P, Q) \in V_1$. We know that there exist $S_1, \ldots, S_m \in D/V_1$ such that $S = \cup_{i=1}^m S_i$. Hence, since V_1 is compatible with G, we get $W(P, S) = \sum_{i=1}^m W(P, S_i) = \sum_{i=1}^m W(Q, S_i) = W(Q, S)$.

Inductive Step. By inductive hypothesis we have that $W(P, S) = W(P_{n-1}, S)$ and $W(P_{n-1}, S) = W(Q, S)$. By transitivity of $=$, we get $W(P, S) = W(Q, S)$. □

From the above claim we get that $\mathcal{R}' = \sqcup_{i \in I} \mathcal{R}_i$ where the $\mathcal{R}_i$ are all the equivalence relations included in $\mathcal{R}$ and compatible with $G^{\mathcal{D}}_\alpha$. Hence, if we prove that $\approx_l \cap(\mathcal{D} \times \mathcal{D})$ is compatible with $G^{\mathcal{D}}_\alpha$ we get the thesis. The fact that $\approx_l \cap(\mathcal{D} \times \mathcal{D})$ is compatible with $G^{\mathcal{D}}_\alpha$ immediately follows from Definitions 9, 10, 12, and Proposition 5. □

As in [23] we consider also a variant notion of compatibility (for S is different from S').

Definition 13. (Relation variant compatible with G) *Let $G = (D, \Delta, W)$ be a weighted directed graph. An equivalence relation R over D is* variant compatible with W *if for each $S, S' \in D/R$ with $S' \neq S$, if $P, Q \in S'$, then $W(P, S) = W(Q, S)$.*

This definition is useful to deal with actions of type τ.

Lemma 5. *Let $\mathcal{D}$ be a set of components such that $\mathcal{D} = ds(\mathcal{D})$. Let $\mathcal{R}$ be an equivalence relation over $\mathcal{D}$ such that $\approx_l \cap(\mathcal{D} \times \mathcal{D}) \subseteq \mathcal{R}$. Let $\mathcal{R}' \subseteq \mathcal{R}$ be the largest equivalence relation variant compatible with $G^{\mathcal{D}}_\tau$. It holds that:*

$$\approx_l \cap(\mathcal{D} \times \mathcal{D}) \subseteq \mathcal{R}'$$

Proof. The proof is analogous to that of Lemma 4. □

Lemma 6. *Let $\mathcal{D}$ be a set of components such that $\mathcal{D} = ds(\mathcal{D})$. Let $\mathcal{R}$ be an equivalence relation over $\mathcal{D}$. If*

- *for each* $\alpha \in \mathcal{A} \setminus \{\tau\}$*, the relation* $\mathcal{R}$ *is compatible with* $G^{\mathcal{D}}_{\alpha}$ *and*
- *the relation* $\mathcal{R}$ *is variant compatible with* $G^{\mathcal{D}}_{\tau}$*,*

then $\mathcal{R}$ *is a lumpable bisimulation over* $\mathcal{D}$*.*

PROOF. We prove that both conditions of Definition 9 hold for $\mathcal{R}$.

Let $\alpha \in \mathcal{A} \setminus \{\tau\}$, $S, S' \in \mathcal{D}/\mathcal{R}$, and $P, Q \in S'$. Since $\mathcal{R}$ is compatible with $G^{\mathcal{D}}_{\alpha}$, we have that $W^{\mathcal{D}}_{\alpha}(P,S) = W^{\mathcal{D}}_{\alpha}(Q,S)$. From the definition of $W^{\mathcal{D}}_{\alpha}$ we get that $q[P,S,\alpha] = q[Q,S,\alpha]$.

Let $S, S' \in \mathcal{D}/\mathcal{R}$, with $S' \neq S$, and $P, Q \in S'$. Since $\mathcal{R}$ is variant compatible with $G^{\mathcal{D}}_{\tau}$, we have that $W^{\mathcal{D}}_{\tau}(P,S) = W^{\mathcal{D}}_{\tau}(Q,S)$. From the definition of $W^{\mathcal{D}}_{\tau}$, we get $q[P,S,\tau] = q[Q,S,\tau]$. □

The authors of [2] proved that the largest equivalence relation $R \subseteq I$ over D variant compatible with $G = (D, \Delta, W)$ is the largest equivalence relation $R \subseteq I$ over D compatible with $\widehat{G} = (D, \Delta, \widehat{W})$, where $\widehat{W}(P,Q) = W(P,Q)$, if $P \neq Q$, while $\widehat{W}(P,P) = -W(P, D\setminus\{P\})$ (see [23] Section 2 Proposition 1). As a consequence we get the following result.

PROPOSITION 6. *The largest equivalence relation* $\mathcal{R}' \subseteq \mathcal{R}$ *over* $\mathcal{D}$ *variant compatible with* $G^{\mathcal{D}}_{\tau}$ *is the largest equivalence relation* $\mathcal{R}' \subseteq \mathcal{R}$ *over* $\mathcal{D}$ *compatible with* $\widehat{G}^{\mathcal{D}}_{\tau}$*.*

The algorithm proposed in [23] that, in this paper, we call LUMPABILITY($G = (D, \Delta, W), I$), computes the largest equivalence relation $R \subseteq I$ over D compatible with G. We exploit it inside our algorithm to compute contextual lumpability. Given a finite set of components $\mathcal{D}$ closed under derivative behaviours, the function reported in Algorithm 1 named CONTEXTUAL_LUMPABILITY($\{G^{\mathcal{D}}_{\alpha}\}_{\alpha\in\mathcal{A}\setminus\{\tau\}}, \widehat{G}^{\mathcal{D}}_{\tau}$) returns the relation $\approx_l \cap(\mathcal{D}\times\mathcal{D})$, i.e., the largest lumpable bisimulation over $\mathcal{D}$.

Algorithm 1 CONTEXTUAL_LUMPABILITY($\{G^{\mathcal{D}}_{\alpha}\}_{\alpha\in\mathcal{A}\setminus\{\tau\}}, \widehat{G}^{\mathcal{D}}_{\tau}$)

Require: $\mathcal{D}$ finite and $\mathcal{D} = ds(\mathcal{D})$;
  $R = \emptyset$;
  $\mathcal{R} = \mathcal{D} \times \mathcal{D}$;
  while $R \neq \mathcal{R}$ **do**
    $R = \mathcal{R}$;
    for all $\alpha \in \mathcal{A} \setminus \{\tau\}$ **do**
      $\mathcal{R}$ =LUMPABILITY($G^{\mathcal{D}}_{\alpha}, \mathcal{R}$);
    end for
    $\mathcal{R}$ =LUMPABILITY($\widehat{G}^{\mathcal{D}}_{\tau}, \mathcal{R}$);
  end while
  return $\mathcal{R}$;

THEOREM 3. (Correctness) *Let* $\mathcal{D}$ *be a finite set of PEPA components such that* $\mathcal{D} = ds(\mathcal{D})$*. A call to the function* CONTEXTUAL_LUMPABILITY($\{G^{\mathcal{D}}_{\alpha}\}_{\alpha\in\mathcal{A}\setminus\{\tau\}}, \widehat{G}^{\mathcal{D}}_{\tau}$) *always terminates and it returns* $\approx_l \cap(\mathcal{D}\times\mathcal{D})$*.*

PROOF. Since LUMPABILITY($\_, \mathcal{R}$) always returns an equivalence relation, it can only remove pairs from $\mathcal{R}$, and $\mathcal{D}$ is finite, we have that the while-loop terminates after at most $|\mathcal{D}|$ iterations.

By Lemma 4 and the correctness of LUMPABILITY($\_,\_$) proved in [23], we have that if $\approx_l \cap(\mathcal{D}\times\mathcal{D}) \subseteq \mathcal{R}$, then for any $\alpha \in \mathcal{A}\setminus\{\tau\}$ algorithm LUMPABILITY($G^{\mathcal{D}}_{\alpha}, \mathcal{R}$) returns a relation $\mathcal{R}'$ such that $\approx_l \cap(\mathcal{D}\times\mathcal{D}) \subseteq \mathcal{R}'$.

Hence, by induction on the number of iterations of the forall-loop, we get that if $\approx_l \cap(\mathcal{D}\times\mathcal{D}) \subseteq \mathcal{R}$ holds at the beginning of the forall-loop, then $\approx_l \cap(\mathcal{D}\times\mathcal{D}) \subseteq \mathcal{R}$ holds at the end of the execution of the forall-loop.

Moreover, by Lemma 5, Proposition 6 and correctness of LUMPABILITY($\_,\_$), we get that if $\approx_l \cap(\mathcal{D}\times\mathcal{D}) \subseteq \mathcal{R}$ holds, then LUMPABILITY($\widehat{G}^{\mathcal{D}}_{\tau}, \mathcal{R}$) returns a relation $\mathcal{R}'$ such that $\approx_l \cap(\mathcal{D}\times\mathcal{D}) \subseteq \mathcal{R}'$ holds.

Hence, by induction on the number of iterations of the while-loop, we get that if $\approx_l \cap(\mathcal{D}\times\mathcal{D}) \subseteq \mathcal{R}$ holds at the beginning of the while-loop, then $\approx_l \cap(\mathcal{D}\times\mathcal{D}) \subseteq \mathcal{R}$ holds at the end of the execution of the while-loop.

So, since at the beginning of the while-loop $\mathcal{R} = \mathcal{D}\times\mathcal{D}$, we get that CONTEXTUAL_LUMPABILITY($\{G^{\mathcal{D}}_{\alpha}\}_{\alpha\in\mathcal{A}\setminus\{\tau\}}, \widehat{G}^{\mathcal{D}}_{\tau}$) returns a relation $\mathcal{R}$ such that $\approx_l \cap(\mathcal{D}\times\mathcal{D}) \subseteq \mathcal{R}$.

To conclude we notice that when the while-loop terminates its last iteration has not modified the relation $\mathcal{R}$. Hence, by Lemma 6, Proposition 6, and correctness of function LUMPABILITY($\_,\_$), we get that $\mathcal{R}$ is a lumpable bisimulation over $\mathcal{D}$. Hence, since by Proposition 5 $\approx_l \cap(\mathcal{D}\times\mathcal{D})$ is the union of all lumpable bisimulations over $\mathcal{D}$, we can conclude that $\mathcal{R} \subseteq \approx_l \cap(\mathcal{D}\times\mathcal{D})$. □

Notice that LUMPABILITY($\widehat{G}^{\mathcal{D}}_{\tau}, \mathcal{D}\times\mathcal{D}$) (correctly) returns $\mathcal{D}\times\mathcal{D}$. However, in the general case with $\mathcal{R} \neq \mathcal{D}\times\mathcal{D}$ LUMPABILITY($\widehat{G}^{\mathcal{D}}_{\tau}, \mathcal{R}$) returns an equivalence relation $\mathcal{R}'$ different from $\mathcal{R}$. For the above reason in our algorithm we start refining $\mathcal{D}\times\mathcal{D}$ on the action types different from τ. A call to LUMPABILITY($\widehat{G}^{\mathcal{D}}_{\tau}, \mathcal{R}$) before the first execution of the forall-loop would be correct, but useless.

THEOREM 4. (Complexity) *Let* $\mathcal{D}$ *be a finite set of PEPA components such that* $\mathcal{D} = ds(\mathcal{D})$*. It holds that function* CONTEXTUAL_LUMPABILITY($\{G^{\mathcal{D}}_{\alpha}\}_{\alpha\in\mathcal{A}\setminus\{\tau\}}, \widehat{G}^{\mathcal{D}}_{\tau}$) *runs in* $O(|\mathcal{D}|^3 * \log|\mathcal{D}|)$ *time and* $O(|\mathcal{D}|^2)$ *memory.*

PROOF. From [23], each execution of LUMPABILITY($G^{\mathcal{D}}_{\alpha}, \mathcal{R}$) requires time $O(|\mathcal{D}| + |\Delta^{\mathcal{D}}_{\alpha}| * \log|\mathcal{D}|)$, while each execution of LUMPABILITY($\widehat{G}^{\mathcal{D}}_{\tau}, \mathcal{R}$) requires time $O(|\mathcal{D}| + |\widehat{\Delta}^{\mathcal{D}}_{\tau}| * \log|\mathcal{D}|)$. Moreover, we already observed that the while-loop is executed at most $|\mathcal{D}|$ times, since it always refines equivalence relations. Hence, we get a total time complexity of $O(|\mathcal{D}| * (|\mathcal{D}| + max(\{|\Delta^{\mathcal{D}}_{\alpha}|\}_{\alpha\in\mathcal{A}\setminus\{\tau\}} \cup \{|\widehat{\Delta}^{\mathcal{D}}_{\tau}|\}) * \log|\mathcal{D}|))$. Since both $|\Delta^{\mathcal{D}}_{\alpha}|$ and $|\widehat{\Delta}^{\mathcal{D}}_{\tau}|$ are at most $|\mathcal{D}|^2$, we get that function CONTEXTUAL_LUMPABILITY($\{G^{\mathcal{D}}_{\alpha}\}_{\alpha\in\mathcal{A}\setminus\{\tau\}}, \widehat{G}^{\mathcal{D}}_{\tau}$) runs in $O(|\mathcal{D}|^3 * \log|\mathcal{D}|)$ time.

As far as memory is concerned, we recall that we have to consider only the memory allocated during execution and not the memory required for input and output. We notice that after each call to LUMPABILITY($\_,\_$) we can reuse memory. Hence, the memory used during computation is exactly the memory used by one execution of LUMPABILITY($\_,\_$) plus the memory used to store R and $\mathcal{R}$. Storing R and $\mathcal{R}$ as partitions, instead of as sets of pairs, they require $O(|\mathcal{D}|)$ memory. The calls to LUMPABILITY($\_,\_$) require $\Theta(|\mathcal{D}| + max(\{|\Delta^{\mathcal{D}}_{\alpha}|\}_{\alpha\in\mathcal{A}\setminus\{\tau\}} \cup \{|\widehat{\Delta}^{\mathcal{D}}_{\tau}|\}))$ memory. Since, $|\Delta^{\mathcal{D}}_{\alpha}|$ and $|\widehat{\Delta}^{\mathcal{D}}_{\tau}|$ are at most $|\mathcal{D}|^2$, we obtain that our algorithm requires $O(|\mathcal{D}|^2)$ memory. □

6. CONCLUSION

In this paper we have presented a notion of contextual lumpability for PEPA components and provided a coinductive characterisation. Lumping a Markov chain, both in case

of discrete and continuous time, can be an efficient way to reduce the model's state space, so that the analyses (e.g., deriving the steady-state performance indices) can be carried out efficiently [14, 1, 10, 23, 2]. Markovian process algebras, such as PEPA, are widely appreciated thanks to their compositionality properties. Informally, we can say that in this paper, differently from [10, 23, 2], we aim at reducing the state space of the CTMC underlying the whole model by simplifying its components considered in isolation. To this end, we have introduced a notion of contextual lumping and its coinductive characterisation. We observe that, with respect to the *strong equivalence* proposed in [13], the lumpable bisimilarity is less restrictive since it allows arbitrary transition rates for activities with type τ among states belonging to the same equivalence class, as illustrated in Example 3. On the other hand, while the *strong equivalence* is a congruence with respect to all the PEPA operators [13], the lumpable bisimilarity is a congruence only for a particular class of contexts, named evaluation contexts, that are PEPA terms with a hole that does not occur under a prefix or a choice operator. Evaluation contexts are widely accepted in the analysis of contextual properties (these contexts are called static contexts in [19]) since they describe environments which can communicate with the process being observed without preventing its internal behaviour. Hence,

$$\text{strong equivalence} \implies \text{lumpable bisimilarity} \implies \text{lumping on the CTMC.}$$

Observe that it is clearly untrue that a lumping on the CTMC implies a lumpable bisimilarity over process algebra's terms. Indeed, according to the definition of lumpability given in [14], the Markov chain with one state is always a valid lumping, but clearly not interesting for application purposes. However, lumpable bisimilarity is a full characterisation of contextual lumpability. Finally, we have given an algorithm to compute the optimal contextual lumping. Observe that, although its asymptotic time complexity is $O(|\mathcal{D}|^3 \log |\mathcal{D}|)$, where $|\mathcal{D}|$ is the number of terms, in practice this turns to be *not* a real drawback under the reasonable assumption that the components are much simpler (i.e., they have much less derivatives) than the joint model that they form with their cooperation.

7. REFERENCES

[1] S. Baarir, M. Beccuti, C. Dutheillet, and G. Franceschinis. From partially to fully lumped Markov chains in stochastic well formed Petri nets. In *Proc. of Valuetools 2009 Conf.*, page 44, Pisa, Italy, 2009. ACM.

[2] S. Baarir, M. Beccuti, C. Dutheillet, G. Franceschinis, and S. Haddad. Lumping partially symmetrical stochastic models. *Perf. Eval.*, 68(1):21 – 44, 2011.

[3] S. Baarir, C. Dutheillet, S. Haddad, and J.-M. Iliè. On the use of exact lumping in partially symmetrical Well-formed Petri Nets. In *Proc. of Int. Conf. on the Quantitative Eval. of Systems (QEST)*, pages 23–32, Torino, Italy, 2005. IEEE Comp. Soc.

[4] C. Baier, J.-P. Katoen, H. Hermanns, and V. Wolf. Comparative branching-time semantics for Markov chains. *Inf. Comput.*, 200(2):149–214, 2005.

[5] M. Bernardo. Weak Markovian bisimulation congruences and exact CTMC-level aggregations for concurrent processes. In *Proc. of the 10th Workshop on Quantitative Aspects of Programming Languages and Systems (QALP'12)*, pages 122–136. EPTCS, 2012.

[6] M. Bernardo. Weak Markovian bisimulation congruences and exact CTMC-level aggregations for sequential processes. In *Proc. of the 6th Int. Conf. on Trustworthy Global Computing (TGC'11)*, volume 7173 of *LNCS*, pages 89–103. Springer-Verlag, 2012.

[7] H. Boudali, P. Crouzen, and M. Stoelinga. A compositional semantics for dynamic fault trees in terms of interactive Markov chains. In *Proc. of the 5th Int. Conf. on Automated Technology for Verification and Analysis (ATVA'07)*, pages 441–456. Springer-Verlag, 2007.

[8] M. Bravetti. Revisiting interactive Markov chains. *Electr. Notes Theor. Comput. Sci.*, 68(5):65–84, 2003.

[9] Y. Deng and M. Hennessy. On the semantics of Markov automata. *Inf. Comput.*, 222:139–168, 2013.

[10] S. Derisavi, H. Hermanns, and W. H. Sanders. Optimal state-space lumping in Markov chains. *Elsevier Information Processing Letters*, 87(6):309–315, 2003.

[11] S. Gilmore, J. Hillston, and M. Ribaudo. An Efficient Algorithm for Aggregating PEPA Models. *IEEE Trans. on Software Eng.*, 27(5):449–464, 2001.

[12] H. Hermanns, U. Herzog, and V. Mertsiotakis. Stochastic process algebras: between LOTOS and Markov chains. *Comp. Netw. and ISDN Syst*, 30:901–924, 1998.

[13] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge Press, 1996.

[14] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. D. Van Nostrand Company, Inc., 1960.

[15] K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Inf. Comput.*, 94(1):1–28, 1991.

[16] S. S. Lavenberg. *Computer Performance Modeling Handbook*. Academic Press, New York, 1983.

[17] A. Marin and S. Rossi. Autoreversibility: exploiting symmetries in Markov chains. In *Proc. of IEEE MASCOTS*, pages 151–160, san Francisco, CA, USA, 2013.

[18] M. Ajmone Marsan, G. Conte, and G. Balbo. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Trans. Comput. Syst.*, 2(2):93–122, 1984.

[19] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[20] M. K. Molloy. Performance analysis using stochastic petri nets. *IEEE Trans. on Comput.*, 31(9):913–917, 1982.

[21] R. Paige and R. E. Tarjan. Three Partition Refinement Algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[22] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, UK, 1994.

[23] A. Valmari and G. Franceschinis. Simple O(m logn) Time Markov Chain Lumping. In *Proc. of Int. Conf. TACAS*, volume 6015, pages 38–52, Paphos, Cyprus, 2010. Springer Verlag.