Differential Bisimulation for a Markovian Process Algebra [⋆]

Giulio Iacobelli[1], Mirco Tribastone[2], and Andrea Vandin[3]

[1] Computing and Systems Engineering, Federal University of Rio de Janeiro, Brazil
[2] IMT Institute for Advanced Studies Lucca, Italy
[3] Electronics and Computer Science, University of Southampton, UK

Abstract. Formal languages with semantics based on ordinary differential equations (ODEs) have emerged as a useful tool to reason about large-scale distributed systems. We present *differential bisimulation*, a behavioral equivalence developed as the ODE counterpart of bisimulations for languages with probabilistic or stochastic semantics. We study it in the context of a Markovian process algebra. Similarly to Markovian bisimulations yielding an aggregated Markov process in the sense of the theory of lumpability, differential bisimulation yields a partition of the ODEs underlying a process algebra term, whereby the sum of the ODE solutions of the same partition block is equal to the solution of a single (lumped) ODE. Differential bisimulation is defined in terms of two symmetries that can be verified only using syntactic checks. This enables the adaptation to a continuous-state semantics of proof techniques and algorithms for finite, discrete-state, labeled transition systems. For instance, we readily obtain a result of compositionality, and provide an efficient partition-refinement algorithm to compute the coarsest ODE aggregation of a model according to differential bisimulation.

1 Introduction

There has been increasing attention to models of computation based on ordinary differential equations (ODEs). This has been mainly prompted by a line of research which interprets an ODE as the deterministic (called *fluid* or *mean-field*) approximation [16,17] of a continuous time Markov chain (CTMC) underlying languages with Markovian semantics [6,11,24]. The ODE semantics provides the behavior of a (concurrent) program as a continuous trajectory representing the *concentration* of processes over time.

In this paper we consider the following problem: *How to compare programs with ODE semantics?* Our main contribution is to lift the notion of bisimulation to languages with ODE semantics. To put it in context, let us draw a parallel with established results of aggregation of CTMCs obtained from a Markovian semantics of a high-level language such as process algebra (e.g., [2,14,4]). This involved finding behavioural relations that induce a partition of the CTMC states which satisfies the property of *ordinary lumpability* [3]: a smaller CTMC can be constructed where each state (a *macro-state*) is the representative of the states in a block; the probability of being in a macro-state is

⋆ This work was partially supported by the DFG project FEMPA, TR 1120/1-1, and by the EU project QUANTICOL, 600708. The postdoctoral fellowship of G.I. is supported by CAPES.

equal to the sum of those of being in the block's states. Here we proceed analogously. We introduce *differential bisimulation* (DB), an equivalence relation that captures symmetries in the ODE semantics according to the well-known theory of ODE *lumpability* [23]: the solution to each ODE representing an equivalence class is equal at all time points to the sum of the solutions of the ODEs of the states in that equivalence class.

We study DB for Fluid Extended Process Algebra (FEPA) [25], a fragment of PEPA [14] with ODE semantics, extended to also capture the product-based synchronisation mechanism of [4,12]. A FEPA model is a composition of *fluid atoms*, each representing a population of identical copies in parallel of the same sequential process, describing its evolution over its set of *local states*. The interaction between fluid atoms occurs via shared channels. A FEPA model encodes a family of systems, parametric in the population sizes of each fluid atom. Under appropriate scaling conditions each member is represented by the same ODEs, one for each local state of each fluid atom, giving the evolution of the number of sequential processes exhibiting that local state.

Differential bisimulation is an equivalence relation over local states of a process. This is in contrast to Markovian bisimulations, which are defined over states of a CTMC. However, DB can be seen as a natural generalization. Indeed it consists of two conditions, the first of which is essentially a Larsen-Skou style bisimulation (cf. [18]) over local states. When a process consists of a fluid atom with one replica (i.e., a single sequential process), the ODE and the CTMC semantics coincide, and DB collapses onto strong equivalence, PEPA's Markovian bisimulation. In the CTMC case such a condition suffices to imply lumpability, informally because the CTMC transition diagram of a process term with an arbitrary synchronization tree structure is isomorphic to the transition system of a single sequential process (by mapping each CTMC state to a named choice term). In the ODE semantics, instead, the synchronization structure is encoded in the function governing the ODE evolution. This is taken into account with the second condition of DB: we introduce the novel concept of *structural interface*, an equivalence relation for local states with same capability to interact with the environment. Both conditions can be checked statically, i.e., syntactically over the process term. Due to the relation with Markovian bisimulation, it is possible to adapt partition-refinement algorithms available for discrete-state labeled transition systems (e.g. [20,13,1]), offering an efficient way to compute the coarsest ODE aggregation of a model up to DB.

2 Preliminaries: FEPA

The grammar of FEPA has two levels. The first level specifies a *fluid atom*, i.e. a sequential process evolving over a discrete state space. Let $\mathcal{A}$ denote the set of actions and $\mathcal{K}$ the set of constants. Each $P \in \mathcal{K}$ is a *sequential component*, defined as $P \overset{def}{=} \sum_{i \in I_P} (\alpha_i, r_i).P_i$, where I_P is an index set, $\alpha_i \in \mathcal{A}$, $r_i \in \mathbb{R}_{\geq 0}$ is a rate, and $P_i \in \mathcal{K}$. The multi-set of outgoing transitions from P, denoted by $out(P)$, is defined as the one containing a transition $P \xrightarrow{(\alpha_i, r_i)} P_i$ for each occurrence of $(\alpha_i, r_i).P_i$ in the definition of P. We now define the second level of the grammar. The parallel operator is parameterised by a binary *synchronisation function*, denoted by $\mathcal{H}(\cdot, \cdot)$. As discussed, we support two such functions, $\mathcal{H} = \min$ and $\mathcal{H} = \cdot$ (product). According to the chosen interpretation, fluid atoms may correspond to, e.g., jobs and servers in a computing system, or to molecular species in a chemical reaction network.

Definition 1 (FEPA Model). *A FEPA model $\mathcal{M}$ is generated by*

$$\mathcal{M} ::= P \;:\; \mathcal{M} \parallel_L^{\mathcal{H}} \mathcal{M}\ , \qquad \textit{with } L \subseteq \mathcal{A} \textit{ and } P \in \mathcal{K}$$

Let $\mathcal{G}(\mathcal{M})$ be the set of *fluid atoms* of a FEPA model $\mathcal{M}$, recursively defined as $\mathcal{G}(P) = \{P\}$, and $\mathcal{G}(\mathcal{M}_1 \parallel_L^{\mathcal{H}} \mathcal{M}_2) = \mathcal{G}(\mathcal{M}_1) \cup \mathcal{G}(\mathcal{M}_2)$. For $P \in \mathcal{G}(\mathcal{M})$, the *local states* of P, denoted $\mathcal{B}(P)$, are the smallest set such that $P \in \mathcal{B}(P)$ and if $P' \in \mathcal{B}(P)$ and $P' \xrightarrow{(\alpha,r)} P'' \in out(P')$, then $P'' \in \mathcal{B}(P)$. We use $\mathcal{B}(\mathcal{M})$ for $\bigcup_{P \in \mathcal{G}(\mathcal{M})} \mathcal{B}(P)$. For any two $P, Q \in \mathcal{G}(\mathcal{M})$, we assume $\mathcal{B}(P) \cap \mathcal{B}(Q) = \emptyset$. This is without loss of generality (e.g., by renaming with fresh variables). For $P \in \mathcal{B}(\mathcal{M})$ we use $\mathcal{A}(P)$ for the set of actions labeling transitions from P. The compositional operator $\parallel_L^{\mathcal{H}}$, parametrized by an action set and by the function $\mathcal{H}$, specifies the type of synchronisation and the channels used for interaction. Notably, different instantiations of $\mathcal{H}$ can appear in a FEPA model.

Example 1. Let $\mathcal{M}_{\mathrm{F}} \triangleq P_1 \parallel_{\{\alpha\}}^{\mathcal{H}} Q_1$, with P_1, Q_1 defined as

$$P_1 \overset{def}{=} (\beta, r).P_2 + (\beta, r).P_3, \qquad P_2 \overset{def}{=} (\alpha, s).P_1, \qquad P_3 \overset{def}{=} (\alpha, s).P_1$$
$$Q_1 \overset{def}{=} (\gamma, 2r).Q_2, \qquad Q_2 \overset{def}{=} (\alpha, s).Q_1$$

We now move to the semantics of FEPA, starting from two quantities specifying local states' dynamics, independently from their possible interaction with other local states.

Definition 2 (Apparent and total conditional rate). *Let $\mathcal{M}$ be a FEPA model, $P \in \mathcal{B}(\mathcal{M})$, $B \subseteq \mathcal{B}(\mathcal{M})$ and $\alpha \in \mathcal{A}$. The α-*apparent rate *of P and the* total α-conditional transition rate *from P to B are defined, respectively, as*

$$r_\alpha(P) \triangleq \sum_{P \xrightarrow{(\alpha,r)} P' \in out(P)} r \qquad\qquad q[P, B, \alpha] \triangleq \sum_{P' \in B} \sum_{P \xrightarrow{(\alpha,r)} P' \in out(P)} r$$

The *α-apparent rate* of a local state P can be understood as a normalized capacity, i.e., the capacity at which a unitary concentration of P-processes performs α-transitions. The *total α-conditional transition rate* restricts the former with respect to a set of target local states; e.g., for $\mathcal{M}_{\mathrm{F}}$ of Example 1 we have $r_\beta(P_1) = 2r$, and $q[P_1, \{P_2\}, \beta] = r$.

Since a fluid atom is a representative of a group of sequential components of the same type, the specification is completed by fixing the group size.

Definition 3 (Concentration function). *Let $\mathcal{M}$ be a FEPA model. We define an* initial population function *for $\mathcal{M}$ as $\nu_0 : \mathcal{B}(\mathcal{M}) \to \mathbb{N}_0$, and a* concentration function *for $\mathcal{M}$ as $\nu : \mathcal{B}(\mathcal{M}) \to \mathbb{R}_{\geq 0}$.*

Definition 4 (Population-dependent apparent rate). *Let $\mathcal{M}$ be a FEPA model, ν a concentration function, and $\alpha \in \mathcal{A}$. The apparent rate of α in $\mathcal{M}$ with respect to ν is*

$$r_\alpha(\mathcal{M}_1 \parallel_L^{\mathcal{H}} \mathcal{M}_2, \nu) \triangleq \begin{cases} \mathcal{H}\big(r_\alpha(\mathcal{M}_1, \nu), r_\alpha(\mathcal{M}_2, \nu)\big), & \textit{if } \alpha \in L, \\ r_\alpha(\mathcal{M}_1, \nu) + r_\alpha(\mathcal{M}_2, \nu), & \textit{if } \alpha \notin L, \end{cases}$$

$$r_\alpha(P, \nu) \triangleq \sum_{P' \in \mathcal{B}(P)} \nu_{P'} \cdot r_\alpha(P')\ .$$

The α-apparent rate in $\mathcal{M}$ is the total rate at which α can be performed, for some ν. It is affected by synchronisations, e.g., in $\mathcal{M}_{\mathrm{F}}$ of Example 1 we have $r_\alpha(\mathcal{M}_{\mathrm{F}}, \nu) = \min(s\,\nu_{P_2} + s\,\nu_{P_3}, s\,\nu_{Q_2})$, or $r_\alpha(\mathcal{M}_{\mathrm{F}}, \nu) = (s\,\nu_{P_2} + s\,\nu_{P_3}) s\,\nu_{Q_2}$, depending on the chosen synchronisation function $\mathcal{H}$. The α-apparent rate in $\mathcal{M}$ is intended as the overall speed at which α is performed in the model; e.g., it is zero if ν_{Q_2} is zero, capturing the blocking effect of synchronisation for both choices of $\mathcal{H}$.

Definition 5 (Model influence). *Let $\mathcal{M}$ be a FEPA model, ν a concentration function for $\mathcal{M}$, $\alpha \in \mathcal{A}$, and $P \in \mathcal{B}(\mathcal{M})$. The* model influence *on P due to α in $\mathcal{M}$ is defined as*

$$\mathcal{F}_\alpha(\mathcal{M}_1 \parallel_L^{\mathcal{H}} \mathcal{M}_2, \nu, P) \triangleq \begin{cases} \mathcal{F}_\alpha(\mathcal{M}_i, \nu, P) \frac{r_\alpha(\mathcal{M}_1 \parallel_L^{\mathcal{H}} \mathcal{M}_2, \nu)}{r_\alpha(\mathcal{M}_i, \nu)}, & \textit{if } P \in \mathcal{B}(\mathcal{M}_i),\ \alpha \in L, \\ \mathcal{F}_\alpha(\mathcal{M}_i, \nu, P), & \textit{if } P \in \mathcal{B}(\mathcal{M}_i),\ \alpha \notin L, \end{cases}$$

$$\mathcal{F}_\alpha(P, \nu, P') \triangleq \begin{cases} 1 & \textit{if } P' \in \mathcal{B}(P), \\ 0 & \textit{otherwise}, \end{cases}$$

where $\frac{r_\alpha(\mathcal{M}_1 \parallel_L^{\mathcal{H}} \mathcal{M}_2, \nu)}{r_\alpha(\mathcal{M}_i, \nu)}$ is defined as 0 when $r_\alpha(\mathcal{M}_i, \nu) = 0$.

Model influence captures the effect exerted by the model $\mathcal{M}$ on the rate at which a local state P performs an action. In other words, the actual α-component rate of P in $\mathcal{M}$ with concentration ν is given by the rate at which P would evolve on its own, i.e., $\nu_P \cdot r_\alpha(P)$, weighted by the influence of the model on it, i.e., $\mathcal{F}_\alpha(\mathcal{M}, \nu, P)$.

We are now ready to define the ODE semantics of a FEPA model.

Definition 6 (ODE semantics). *Let $\mathcal{M}$ be a FEPA model, $\mathcal{E} \subseteq \mathbb{R}^{\mathcal{B}(\mathcal{M})}$ and $f : \mathcal{E} \to \mathbb{R}^{\mathcal{B}(\mathcal{M})}$ the vector field whose components are defined for each $P \in \mathcal{B}(\mathcal{M})$ as:*

$$f_P(\nu) \triangleq \sum_{\alpha \in \mathcal{A}} \sum_{P' \in \mathcal{B}(\mathcal{M})} \nu_{P'} q(P', P, \alpha) \mathcal{F}_\alpha(\mathcal{M}, \nu, P') - \sum_{\alpha \in \mathcal{A}} \nu_P r_\alpha(P) \mathcal{F}_\alpha(\mathcal{M}, \nu, P)$$

The ODE system $\dot{\nu} = f(\nu)$ with initial condition ν_0 governs the evolution of ν over time.

The rate of change in the concentration of a local state P depends on the actual rate at which each local state P' performs transitions towards P, minus the actual rate at which P performs any transition. For instance, the ODEs of $\mathcal{M}_{\mathrm{F}}$ of Example 1 are:

$$\begin{aligned} \dot{\nu}_{P_1} &= s\,\mathcal{H}(\nu_{P_2} + \nu_{P_3}, \nu_{Q_2}) - 2r\,\nu_{P_1} & \dot{\nu}_{Q_1} &= s\,\mathcal{H}(\nu_{P_2} + \nu_{P_3}, \nu_{Q_2}) - 2r\,\nu_{Q_1} \\ \dot{\nu}_{P_2} &= r\,\nu_{P_1} - s\,\nu_{P_2} \frac{\mathcal{H}(\nu_{P_2} + \nu_{P_3}, \nu_{Q_2})}{\nu_{P_2} + \nu_{P_3}} & \dot{\nu}_{Q_2} &= 2r\,\nu_{P_1} - s\,\mathcal{H}(\nu_{P_2} + \nu_{P_3}, \nu_{Q_2}) \\ \dot{\nu}_{P_3} &= r\,\nu_{P_1} - s\,\nu_{P_3} \frac{\mathcal{H}(\nu_{P_2} + \nu_{P_3}, \nu_{Q_2})}{\nu_{P_2} + \nu_{P_3}} \end{aligned} \tag{1}$$

3 Differential Bisimulation and ODE Lumpability

The second level of the FEPA grammar defines a tree-like structure which strongly affects the ODE semantics. To take this into account in our differential bisimulation, we introduce the notion of *interface actions*, which intuitively captures all actions which affect the dynamics of a local state as a result of an interaction.

Definition 7 (Bound and interface actions). *Let $\mathcal{M}$ be a FEPA model, and $P \in \mathcal{B}(\mathcal{M})$. The set of* bound actions *of P in $\mathcal{M}$ is defined as*

$$\mathcal{D}(P, \mathcal{M}) \triangleq \begin{cases} L \cup \mathcal{D}(P, \mathcal{M}_i) \ , & \text{if } \mathcal{M} = \mathcal{M}_1 \parallel_L^{\mathcal{H}} \mathcal{M}_2 \text{ and } P \in \mathcal{B}(\mathcal{M}_i), \\ \emptyset \ , & \text{otherwise} \ . \end{cases}$$

Also, the interface actions *of P in $\mathcal{M}$ are $\mathcal{I}(P, \mathcal{M}) \triangleq \mathcal{D}(P, \mathcal{M}) \cap \mathcal{A}(P)$. Lastly, for any $B \subseteq \mathcal{B}(\mathcal{M})$, we use $\mathcal{D}(B, \mathcal{M})$ for $\bigcup_{P \in B} \mathcal{D}(P, \mathcal{M})$, and $\mathcal{I}(B, \mathcal{M})$ for $\bigcup_{P \in B} \mathcal{I}(P, \mathcal{M})$.*

The following notion of *structural interface* captures symmetries among the states of a FEPA model with respect to the rigid tree-like structure of the model.

Definition 8 (Structural interface). *Let $\mathcal{M}$ be a FEPA model, and $P, Q \in \mathcal{B}(\mathcal{M})$. Then P and Q have the same* structural interface *in $\mathcal{M}$, written $P \overset{s.i.}{=}_{\mathcal{M}} Q$, iff*

(i) $\mathcal{A}(P) = \mathcal{A}(Q)$, and
(ii) if there exists an $\overline{\mathcal{M}} = \mathcal{M}_1 \parallel_L^{\mathcal{H}} \mathcal{M}_2$ within $\mathcal{M}$ with $P \in \mathcal{B}(\mathcal{M}_1)$, and $Q \in \mathcal{B}(\mathcal{M}_2)$ (or vice versa), then $\mathcal{I}(P, \overline{\mathcal{M}}) = \mathcal{I}(Q, \overline{\mathcal{M}}) = \emptyset$.

Proposition 1. *For $\mathcal{M}$ a FEPA model, $\overset{s.i.}{=}_{\mathcal{M}}$ is an equivalence relation.* [4]

Considering Example 1 we have $D(P_1, \mathcal{M}_{\mathrm{F}}) = D(P_2, \mathcal{M}_{\mathrm{F}}) = \{\alpha\}$, $\mathcal{I}(P_1, \mathcal{M}_{\mathrm{F}}) = \emptyset$, and $\mathcal{I}(P_2, \mathcal{M}_{\mathrm{F}}) = \{\alpha\}$. Also, we have $P_2 \overset{s.i.}{=}_{\mathcal{M}_{\mathrm{F}}} P_3$, $P_3 \overset{s.i.}{\neq}_{\mathcal{M}_{\mathrm{F}}} Q_2$, and $P_2 \overset{s.i.}{\neq}_{\mathcal{M}_{\mathrm{F}}} Q_2$ (capturing, for instance, that α is used by P_2 and Q_2 to interact in a specific fashion).

We can now provide the notion of differential bisimulation for FEPA models.

Definition 9 (Differential bisimulation). *Let $\mathcal{M}$ be a FEPA model, $\mathcal{R}$ an equivalence relation over $\mathcal{B}(\mathcal{M})$, and $\mathcal{P} = \mathcal{B}(\mathcal{M})/\mathcal{R}$. We say that $\mathcal{R}$ is a* differential bisimulation *for $\mathcal{M}$ (DB) iff for all $(P, P') \in \mathcal{R}$ and $\alpha \in \mathcal{A}$ we have:*

(i) $q[P, B, \alpha] = q[P', B, \alpha]$*, for all $B \in \mathcal{P}$,*
(ii) $P \overset{s.i.}{=}_{\mathcal{M}} P'$.

We define differential bisimilarity *for $\mathcal{M}$, denoted by $\dot{\sim}$, as the union of all DBs for $\mathcal{M}$, and we say that $P, P' \in \mathcal{B}(\mathcal{M})$ are differential bisimilar iff $s \dot{\sim} s'$.*

As usual, we are interested in the largest differential bisimulation. We now show that differential bisimilarity is a DB, and thus it is the largest one. To do this, we prove that the transitive closure of the union of DBs is a differential bisimulation.

Proposition 2. *Let $\mathcal{M}$ be a FEPA model, I be a set of indices, and $\mathcal{R}_i$ a DB for $\mathcal{M}$, for all $i \in I$. The transitive closure of their union $\mathcal{R} = (\bigcup_{i \in I} \mathcal{R}_i)^*$ is a DB for $\mathcal{M}$.*

The next theorem states that DB is preserved under composition of FEPA models.

Theorem 1 (Differential bisimulation is a congruence). *Let $\mathcal{M}_1$, $\mathcal{M}_2$ be two FEPA models, and $\mathcal{R}_1$, $\mathcal{R}_2$ be two differential bisimulations for $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. Then $\mathcal{R}_1 \cup \mathcal{R}_2$ is a differential bisimulation for $\mathcal{M}_1 \parallel_L^{\mathcal{H}} \mathcal{M}_2$, for any $L \subseteq \mathcal{A}$.*

[4] All proofs are provided in the extended technical report [15].

Remark 1. An interesting connection between differential bisimulation and its Markovian analogues, like *Markovian bisimulation* [13] and PEPA's *strong equivalence* [14] arises: condition (i) of DB corresponds to the condition required by Markovian bisimulation and by strong equivalence. However, in the Markovian cases states of the underlying labelled transition system (semantic elements) are related, while DB relates the states of the fluid atoms (syntactic elements). This requires to explicitly treat the influence exerted by the model on each local state (condition (ii)). Such information is instead implicitly present in the transition systems considered in the Markovian cases.

We now show that DB induces an ODE aggregation in the sense of the theory of ODE lumpability (e.g., [23]). We first exemplify it considering Example 1, for which it can be shown that $P_2 \dot{\sim} P_3$. Using the variable renaming $\nu_{P_{23}} = \nu_{P_2} + \nu_{P_3}$, by the linearity of the differential operator we can aggregate Equation (1) as

$$\begin{aligned} \dot{\nu}_{P_1} &= s\,\mathcal{H}(\nu_{P_{23}}, \nu_{Q_2}) - 2r\,\nu_{P_1} & \qquad \dot{\nu}_{Q_1} &= s\,\mathcal{H}(\nu_{P_{23}}, \nu_{Q_2}) - 2r\,\nu_{Q_1} \\ \dot{\nu}_{P_{23}} &= 2r\,\nu_{P_1} - s\,\mathcal{H}(\nu_{P_{23}}, \nu_{Q_2}) & \qquad \dot{\nu}_{Q_2} &= 2r\,\nu_{P_1} - s\,\mathcal{H}(\nu_{P_{23}}, \nu_{Q_2}) \end{aligned}$$

If the initial conditions are such that $\nu_{0P_{23}} = \nu_{0P_2} + \nu_{0P_3}$, the solutions satisfy $\nu_{P_{23}}(t) = \nu_{P_2}(t) + \nu_{P_3}(t)$ for all t. As discussed, this is analogous to ordinary lumpability in CTMCs, where the probability of being in a state of the aggregated chain is equal to the sum of the probabilities of being in the states of the related equivalence class [3].

Noteworthy, condition (i) of DB does not capture ODE aggregation if ignoring structural interface. Assuming $\beta = \gamma$, $\{\{P_1, Q_1\}, \{P_2, P_3, Q_2\}\}$ satisfies condition (i). Yet, $P_2 \overset{s.j.}{\not\doteq}_{\mathcal{M}_\mathrm{F}} Q_2$ and $P_3 \overset{s.j.}{\not\doteq}_{\mathcal{M}_\mathrm{F}} Q_2$. This results in ODEs with nonlinear terms in ν_{P_2} and ν_{Q_2}, such as $\mathcal{H}(\nu_{P_2} + \nu_{P_3}, \nu_{Q_2})$, which cannot be written in terms of $\nu_{P_2} + \nu_{Q_2}$.

We formalize such ODE aggregation in terms of ODE lumpability by an aggregation matrix. Given a FEPA model $\mathcal{M}$ and a partition $\mathcal{P}$ of $\mathcal{B}(\mathcal{M})$, the *aggregation matrix* of $\mathcal{P}$ has $|\mathcal{P}| \times |\mathcal{B}(\mathcal{M})|$ components given as $(M_\mathcal{P})_{i,j} = 1$ if $P_j \in B_i$, and $(M_\mathcal{P})_{i,j} = 0$ otherwise, where $B_i \in \mathcal{P}$ and $P_j \in \mathcal{B}(\mathcal{M})$, with $i \in \{1, \ldots, |\mathcal{P}|\}$ and $j \in \{1, \ldots, |\mathcal{B}(\mathcal{M})|\}$.

Definition 10 (ODE lumpability). *Let $\mathcal{M}$ be a FEPA model, f its vector field, and $\mathcal{P}$ a partition of $\mathcal{B}(\mathcal{M})$. The ODE system $\dot{\nu} = f(\nu)$ is* lumpable *by $M_\mathcal{P}$ if and only if*

$$M_\mathcal{P} f(\nu) = M_\mathcal{P} f(\overline{M}_\mathcal{P} M_\mathcal{P} \nu)\ , \qquad \textit{for all } \nu\ , \tag{2}$$

where $\overline{M}_\mathcal{P}$ is any generalized right inverse of $M_\mathcal{P}$, i.e., a matrix satisfying $M_\mathcal{P} \overline{M}_\mathcal{P} = \mathbb{I}$.

The vector ν has $|\mathcal{B}(\mathcal{M})|$ components, each being the concentration of a local state of $\mathcal{M}$ at a certain time. For $\mathcal{P}$ a partition of $\mathcal{B}(\mathcal{M})$, $M_\mathcal{P}\nu$ has $|\mathcal{P}|$ components, each equal to the sum of the components of ν in the corresponding block. The vector $\overline{M}_\mathcal{P} M_\mathcal{P} \nu$ has again $|\mathcal{B}(\mathcal{M})|$ components, obtained by first summing the components of ν in each block ($M_\mathcal{P}\nu$) and subsequently redistributing it to the local states of the block. Equation (2) demands that the sum of the dynamics of local states of a block, i.e., $M_\mathcal{P} f(\nu)$, can be expressed as a function of the aggregated vector, i.e., $M_\mathcal{P}\nu$, only.

Theorem 2 (Differential bisimulation and lumpability). *Let $\mathcal{M}$ be a FEPA model, $\mathcal{R}$ a differential bisimulation, and $\mathcal{P} = \mathcal{B}(\mathcal{M})/\mathcal{R}$. The ODEs of $\mathcal{M}$ are lumpable by $M_\mathcal{P}$.*

Proof (sketch). We have to show that Equation (2) holds. The proof uses Proposition 4 and Lemma 4 given in [15], and here discussed. For ν a concentration function for $\mathcal{M}$ and $\mathcal{P}$ a partition of $\mathcal{B}(\mathcal{M})$, we define $[\nu]^{\mathcal{P}}$, the *$\mathcal{P}$-redistribution of* ν, as

$$[\nu]^{\mathcal{P}} = \overline{M}_{\mathcal{P}} M_{\mathcal{P}} \nu \ . \tag{3}$$

Thus, we have to show that for any ν it holds $M_{\mathcal{P}} f(\nu) = M_{\mathcal{P}} f([\nu]^{\mathcal{P}})$. Recalling the definition of the aggregation matrix $M_{\mathcal{P}}$, it is enough to show that for any $B \in \mathcal{P}$ and ν

$$\sum_{P \in B} f_P(\nu) = \sum_{P \in B} f_P(\overline{M}_{\mathcal{P}} M_{\mathcal{P}} \nu) = \sum_{P \in B} f_P([\nu]^{\mathcal{P}}) \ ,$$

i.e., we verify Equation (2) componentwise. Summing over $P \in B$ both sides of f of Definition 6, and using that $\sum_{P \in B} q(P', P, \alpha) = q[P', B, \alpha]$, as well as a decomposition of the sum over states, i.e., $\sum_{P' \in \mathcal{B}(\mathcal{M})}(\cdot) = \sum_{B \in \mathcal{P}} \sum_{P' \in B}(\cdot)$, we obtain

$$\begin{aligned}
\sum_{P \in B} f_P(\nu) = &\sum_{\alpha \in \mathcal{A}} \sum_{P' \in \mathcal{B}(\mathcal{M})} \nu_{P'} \ q[P', B, \alpha] \mathcal{F}_\alpha(\mathcal{M}, \nu, P') \\
&- \sum_{P \in B} \nu_P \sum_{\tilde{B} \in \mathcal{P}} \sum_{\alpha \in \mathcal{A}} q[P, \tilde{B}, \alpha] \mathcal{F}_\alpha(\mathcal{M}, \nu, P) \\
= &\sum_{\tilde{B} \in \mathcal{P}} \sum_{P' \in \tilde{B}} \sum_{\alpha \in \mathcal{A}} q[P', B, \alpha] \mathcal{F}_\alpha(\mathcal{M}, \nu, P') \nu_{P'} \\
&- \sum_{P \in B} \sum_{\tilde{B} \in \mathcal{P}} \sum_{\alpha \in \mathcal{A}} q[P, \tilde{B}, \alpha] \mathcal{F}_\alpha(\mathcal{M}, \nu, P) \nu_P
\end{aligned} \tag{4}$$

We are left with showing that for any ν Equation (4) does not change if we replace ν with $[\nu]^{\mathcal{P}}$. This corresponds to saying that it can be expressed as a function of the sums of the concentrations in each block of $\mathcal{P}$ only. For any P and any $B \in \mathcal{P}$ we can write $\sum_{\alpha \in \mathcal{A}} q[P, B, \alpha] \mathcal{F}_\alpha(\mathcal{M}, \nu, P) = \sum_{\alpha \in \mathcal{A}(P)} q[P, B, \alpha] \mathcal{F}_\alpha(\mathcal{M}, \nu, P)$, which follows from observing that any $\alpha \notin \mathcal{A}(P)$ brings 0-contribution to the equation (because $\alpha \notin \mathcal{A}(P) \implies r_\alpha(P) = 0 \implies q[P, B, \alpha] = 0, \forall B$). We now exploit the fact that $\mathcal{P}$ is induced by a DB on $\mathcal{B}(\mathcal{M})$, as sketched below in the following three points.

(i) We have that for all $B \in \mathcal{P}$, for all $Q, Q' \in B$, $q[Q, \tilde{B}, \alpha] = q[Q', \tilde{B}, \alpha]$ for all $\tilde{B} \in \mathcal{P}$ and all $\alpha \in \mathcal{A}$, which, in turn, implies $\mathcal{A}(Q) = \mathcal{A}(Q')$.

(ii) We show, in Proposition 4, that for all $B \in \mathcal{P}$, and all $Q, Q' \in B$, $\mathcal{F}_\alpha(\mathcal{M}, \nu, Q) = \mathcal{F}_\alpha(\mathcal{M}, \nu, Q')$ for all ν and all $\alpha \in \mathcal{A}(Q) = \mathcal{A}(Q')$. Thus, it holds that for all $B, \tilde{B} \in \mathcal{P}$, all $P, P' \in B$, and all ν, $\sum_{\alpha \in \mathcal{A}} q[P, \tilde{B}, \alpha] \mathcal{F}_\alpha(\mathcal{M}, \nu, P) = \sum_{\alpha \in \mathcal{A}} q[P', \tilde{B}, \alpha] \mathcal{F}_\alpha(\mathcal{M}, \nu, P')$. That is, the summation is equal for all local states of block B. Proposition 4 establishes a relation between structural interface (Definition 8) and model influence (Definition 5), essentially saying that if two local states have the same structural interface within a model, then they receive the same influence from the model.

(iii) We show, in Lemma 4, that for any $P \in \mathcal{B}(\mathcal{M})$, α and ν it holds $\mathcal{F}_\alpha(\mathcal{M}, \nu, P) = \mathcal{F}_\alpha(\mathcal{M}, [\nu]^{\mathcal{P}}, P)$. This is used to infer that for any $B, \tilde{B} \in \mathcal{P}$, any $P \in B$ and any ν:

$$\sum_{\alpha \in \mathcal{A}} q[P, \tilde{B}, \alpha] \mathcal{F}_\alpha(\mathcal{M}, \nu, P) = \sum_{\alpha \in \mathcal{A}} q[P, \tilde{B}, \alpha] \mathcal{F}_\alpha(\mathcal{M}, [\nu]^{\mathcal{P}}, P).$$

```
DifferentialBisimilarity($\mathcal{M}$,$\mathcal{P}$) :=
 RefineSI($\mathcal{M}$,$\mathcal{P}$)                //Refine $\mathcal{P}$ wrt condition (ii)
 RefineQ($\mathcal{M}$,$\mathcal{P}$)                 //Iteratively refines $\mathcal{P}$ wrt condition (i)
RefineSI($\mathcal{M}$,$\mathcal{P}$) :=
  forall($\alpha \in \mathcal{A}(\mathcal{M})$)
   refineAccordingToComp($\alpha$,$\mathcal{P}$)     //Refine $\mathcal{P}$ wrt $comp[\alpha]$, for all $\alpha$

RefineQ($\mathcal{M}$,$\mathcal{P}$) :=
 Spls = $\mathcal{A}(\mathcal{M}) \times \mathcal{P}$            //All $(\alpha, B)$ are considered as candidate splitters
 while(Spls $\neq \emptyset$)
  $(\alpha, B_{spl})$ = pop(Spls)          //choose and remove a candidate splitter
  Split($\alpha$, $B_{spl}$, $\mathcal{P}$,Spls)          //split all blocks of $\mathcal{P}$ wrt $(\alpha, B_{spl})$
```

Algorithm 1. An algorithm for computing differential bisimilarity

That is, the summation can be expressed as a function of the sums of the concentration in each block of $\mathcal{P}$. In other words, the model influence received by a local state depends on the concentration of the other local states only through the sum of the concentrations within blocks of $\mathcal{P}$, thus a change in the concentrations which preserves the total concentrations of each block does not affect the model influence.

Now that all the proof ingredients have been provided, we can rewrite Equation (4) as follows, where we use that for any $B \in \mathcal{P}$, $\sum_{P \in B} [\nu]_P^{\mathcal{P}} = \sum_{P \in B} \nu_P$, which arises from Equation (3) and the fact that the matrix $\overline{M}_{\mathcal{P}}$ must satisfy $M_{\mathcal{P}} \overline{M}_{\mathcal{P}} = \mathbb{I}$:

$$\begin{aligned}
\sum_{P \in B} f_P(\nu) &= \sum_{\tilde{B} \in \mathcal{P}} \sum_{\alpha \in \mathcal{A}} q[P', B, \alpha] \mathcal{F}_\alpha(\mathcal{M}, \nu, P') \sum_{P' \in \tilde{B}} \nu_{P'} \\
&\quad - \sum_{\tilde{B} \in \mathcal{P}} \sum_{\alpha \in \mathcal{A}} q[P, \tilde{B}, \alpha] \mathcal{F}_\alpha(\mathcal{M}, \nu, P) \sum_{P \in B} \nu_P \\
&= \sum_{\tilde{B} \in \mathcal{P}} \sum_{\alpha \in \mathcal{A}} q[P', B, \alpha] \mathcal{F}_\alpha(\mathcal{M}, [\nu]^{\mathcal{P}}, P') \sum_{P' \in \tilde{B}} [\nu]_{P'}^{\mathcal{P}} \\
&\quad - \sum_{\tilde{B} \in \mathcal{P}} \sum_{\alpha \in \mathcal{A}} q[P, \tilde{B}, \alpha] \mathcal{F}_\alpha(\mathcal{M}, [\nu]^{\mathcal{P}}, P) \sum_{P \in B} [\nu]_P^{\mathcal{P}} = \sum_{P \in B} f_P([\nu]^{\mathcal{P}}) \quad \square
\end{aligned}$$

4 Computing Differential Bisimilarity

We now provide an efficient algorithm for computing differential bisimilarity obtained by extending and reusing well-known partition refinement algorithms, e.g. [20,13,1].

In order to apply partition refinement to differential bisimilarity, let us first note that condition (ii) of DB can be dealt with as an initialization step that pre-partitions the local states according to their structural interface. Instead, condition (i) requires the usual partition-refinement treatment: starting from the partition obtained after initialization, the blocks are iteratively *split* until there exists a block and an action (i.e., a candidate splitter) for which condition (i) does not hold. The algorithm takes in input any initial partition $\mathcal{P}$, useful e.g. to specify local states that should not be equated, and terminates giving the largest differential bisimilarity which refines $\mathcal{P}$ for the considered model.

Overview. `DifferentialBisimilarity`, our algorithm, is given in Algorithm 1, where $\mathcal{M}$ is the input FEPA model and $\mathcal{P}$ the initial partition. We use $\mathcal{A}(\mathcal{M})$ for the set of actions in $\mathcal{M}$, and $\mathcal{T}(\mathcal{M}) \triangleq \{\![P' \xrightarrow{(\alpha,r)} P'' \in out(P') \mid P' \in \mathcal{B}(\mathcal{M})]\!\}$ for its multi-set of transitions. Note that $|\mathcal{A}(\mathcal{M})| \leq |\mathcal{T}(\mathcal{M})|$. Also, we use $t_{\mathcal{M}}$ for $|\mathcal{T}(\mathcal{M})|$, and $s_{\mathcal{M}}$ for $|\mathcal{B}(\mathcal{M})|$, and we do not distinguish an equivalence relation from its induced partition. `RefineSI` implements the initialization step, yielding the coarsest refinement of $\mathcal{P}$ with respect to condition (ii). `RefineQ` iteratively computes the coarsest refinement satisfying condition (i). Overall, the algorithm is correct, as the iterative refinements preserve condition (ii). It is assumed that $\mathcal{M}$ is stored as the list $\mathcal{T}(\mathcal{M})$, requiring $O(t_{\mathcal{M}})$ space. In order to represent partitions $\mathcal{P}$, $\mathcal{B}(\mathcal{M})$ is stored as a list, while a block of $\mathcal{P}$ is a list of pointers to its states, requiring in total $O(t_{\mathcal{M}} + s_{\mathcal{M}})$ to store $\mathcal{M}$.

`RefineSI`. This procedure is based on a simple rephrasing of Definition 8: given a FEPA model $\mathcal{M}$ and $P_1, P_2 \in \mathcal{B}(\mathcal{M})$ with $\mathcal{A}(P_1) = \mathcal{A}(P_2)$, we have $P_1 \overset{s.i.}{=} P_2$ if and only if for all $\alpha \in \mathcal{A}(P_1)$ and for all occurrences $\overline{\mathcal{M}} = \mathcal{M}_1 \parallel_L \mathcal{M}_2$ within $\mathcal{M}$ with $\alpha \in L \cap \mathcal{A}(P_1)$ we have that P_1 and P_2 either belong to the same $\mathcal{M}_i$, or do not belong to any of the two (i.e., $P_1, P_2 \notin \mathcal{B}(\overline{\mathcal{M}})$). Also, if two states have the same innermost compositional operator binding α, then they share all outers too. No further information is required about compositional operators, and thus we assume that each $P \in \mathcal{B}(\mathcal{M})$ has a list *comp* containing an entry per action in $\mathcal{A}(P)$, each being a triple storing the action, the (identifier of the) innermost compositional operator affecting P and binding the key action, and the side of the operator to which P belongs. Also, each *comp* is assumed to be sorted with respect to a total oderdering on $\mathcal{A}$. We use $comp[\alpha]$ for the values associated with α in *comp*. For instance, for $\mathcal{M} = \mathcal{M}_1 \parallel_L \mathcal{M}_2$, id^* the identifier of $\parallel_L$, $P_1 \in \mathcal{B}(\mathcal{M}_1)$ and $\alpha \in \mathcal{A}(P_1) \cap L$, we have $P_1.comp[\alpha] = (id^*, left)$ if no further compositional operators binding α appear in the syntax-tree path leading to P_1. All *comp* require $O(t_{\mathcal{M}})$ space in total: each $P.comp$ has at most one entry per transition with source P, and thus at most $t_{\mathcal{M}}$ entries appear in all *comp*. By defining a total ordering on *comp*'s values, `RefineSI` reduces to iteratively sorting all $P \in \mathcal{B}(\mathcal{M})$ according to $P.comp[\alpha]$ for all $\alpha \in \mathcal{A}(\mathcal{M})$ (Line 6). [5] The sorting for each α can be performed in $O(s_{\mathcal{M}} \cdot \log s_{\mathcal{M}})$, and if we scan $\mathcal{A}(\mathcal{M})$ according to the ordering of $\mathcal{A}$ we can access the elements of the lists in constant time, requiring $O(t_{\mathcal{M}} \cdot s_{\mathcal{M}} \cdot \log s_{\mathcal{M}})$ time to perform the sorting. Overall, this yields $O(t_{\mathcal{M}} \cdot s_{\mathcal{M}} \cdot \log s_{\mathcal{M}})$ time complexity.

Theorem 3. *Let $\mathcal{M}$ be a FEPA model and $\mathcal{P}$ a partition of $\mathcal{B}(\mathcal{M})$.* `RefineSI` *computes the coarsest refinement of $\mathcal{P}$ satisfying condition* (ii)*. It can be implemented with time and space complexities $O(t_{\mathcal{M}} \cdot s_{\mathcal{M}} \cdot \log s_{\mathcal{M}})$ and $O(t_{\mathcal{M}} + s_{\mathcal{M}})$, respectively.*

`RefineQ`. Condition (i) ignores compositional operators. Thus, `RefineQ` treats $\mathcal{M}$ as a *stochastic labeled transition system* (STLS), i.e. a transition system (with a root per fluid atom) where transitions are labeled by an action and a real. This allows us to use the algorithm for *Markovian bisimilarity* of SLTSs presented in [13,9]. In fact, as discussed, condition (i) corresponds to Markovian bisimulation. Indeed, `RefineQ` is

[5] $P.comp[\alpha]$ is nil if $\alpha \notin \mathcal{A}(P)$, and $free$ if $\alpha \in \mathcal{A}(P)$ and $\alpha \notin \mathcal{D}(P, \mathcal{M})$, so to tell apart states performing different actions.

a straightforward rephrasing of the algorithm of [13,9] to FEPA notation. An in-depth discussion of the algorithm can be found in [13,9], while we hereby give a high-level description. We start recalling the algorithm's complexities.

Theorem 4 (Adapted from [13]). *For $\mathcal{M}$ a FEPA model and $\mathcal{P}$ a partition of $\mathcal{B}(\mathcal{M})$,* `RefineQ` *gives the coarsest refinement of $\mathcal{P}$ satisfying condition* (i) *of DB. It can be realized with time and space complexities $O(t_{\mathcal{M}} \cdot \log s_{\mathcal{M}})$ and $O(t_{\mathcal{M}} + s_{\mathcal{M}})$, respectively.*

Refinements are based on *splitters* (α, B_{spl}), with $\alpha \in \mathcal{A}(\mathcal{M})$ and $B_{spl} \in \mathcal{P}$: a block $B \in \mathcal{P}$ is split with respect to (α, B_{spl}) in disjoint sub-blocks, each containing states with same total α-conditional transition rate towards B_{spl}. `RefineQ` starts (Line 9) generating a set `Spls` of initial potential splitters (α, B) for each $\alpha \in \mathcal{A}(\mathcal{M})$ and $B \in \mathcal{P}$. Then, Lines 10-12 iterate until there are potential splitters to be considered: a splitter is selected and removed from `Spls`, and the procedure `Split` is invoked to refine each block of $\mathcal{P}$ according to the selected splitter, and to generate new candidate splitters. Due to space constraints we do not detail the `Split` procedure.

Summary. Theorems 3, 4 allow us to conclude that `DifferentialBisimilarity` has time and space complexities $O\big(t_{\mathcal{M}} \cdot s_{\mathcal{M}} \cdot \log s_{\mathcal{M}}\big)$ and $O(t_{\mathcal{M}} + s_{\mathcal{M}})$, respectively.

5 Related Work

The *label equivalence* presented in [26] captures *exact fluid lumpability*, a different notion of ODE lumpability than the one captured by DB, where processes are equivalent whenever their ODE solutions are equal at all time points, provided they have same initial conditions. Label equivalence works at a coarser level of granularity than DB, as it relates whole fluid atoms, and not their individual local states, essentially requiring an isomorphism between them. Further, the conditions for equivalence in [26] include universal quantifiers over the uncountable set of concentration functions which are difficult to check automatically. Indeed, no algorithm for computing the coarsest partition was developed for label equivalence. In contrast, DB is given in terms of syntactic elements only, allowing us to provide an efficient algorithm to compute the largest one of a model. In [25] the same authors extended the framework of [26] to the notion of ODE lumpability considered in this paper, for which, however, the same limitations as those of label equivalence apply.

The relationship between formal languages and ODEs induced by their semantics has been studied also in other contexts, with complementary approaches. In [7] it is presented a model-order reduction technique for κ [8], a rule-based language for chemical systems representing bindings between molecules in an explicit graph-based way. The aggregation method, called *fragmentation*, identifies a linear transformation of the state space yielding a subspace with a closed dynamics, i.e., whose ODEs depend only on the variables of that subspace. This may give an *improper lumping* (see [19]), as the same state may appear in more than one aggregate, and thus it is not necessarily induced by a partition of the state space. More practically, it can be shown that $\mathcal{M}_{\mathrm{F}}$ of Example 1 can be encoded in κ in case $\mathcal{H} = \cdot$, but it is not reduced by fragmentation. (Dually, there exist κ's models which can be encoded in FEPA that are reduced by fragmentation but not by DB). However, clearly, the two target languages are different; κ is based on the *law of mass action*, where the rate of interaction is proportional to the product of the participants' concentrations, similarly to FEPA's $\mathcal{H} = \cdot$. Instead, FEPA is process-based, with the rule of interaction implicit in the rigid compositional structure, while a chemical system is an unstructured set of interacting species. Also, FEPA allows for a synchronisation semantics based on capacity-sharing arguments (in the case $\mathcal{H} = \min$).

More closely related is the bisimulation in [5], which induces both ODE lumpabilities of Definition 10 and [26]. The difference is again in the language-specific definitions of equivalence. While DB is a relation over process algebra terms, in [5] symmetries are exploited between binding sites of κ agents. Also, [5] requires stronger symmetries than DB, as the latter considers those specific to the notion of lumpabilty of Definition 10 only. For example, it can be shown that the DB $\{\{P_1\}, \{P_2, P_3\}, \{Q_1\}, \{Q_2\}\}$ of $\mathcal{M}_{\mathrm{F}}$ of Example 1 does not satisfy the notion of lumpability of [26].

The combination of the notion of bisimulation and ODEs has been explored also by the control theory community, most notably in the work of Pappas and co-authors (e.g., [21,10]) and van der Schaft [22]. However, the setting is different. When studied for model reduction, they essentially deal with a state space representation with an explicit output map, e.g., the matrix C in the linear dynamical system $\dot{x} = Ax + Bu$, $y = Cx$. A bisimulation is thus related to unobservability subspaces (cf. [21, Section 8.1] and [22, Corollary 6.4]). By contrast, in this paper we work with a nonlinear system in the form $\dot{x} = A(x)$ (with A a nonlinear vector field) where bisimulation is related to aggregation; in the aggregated model only a linear combination of the original state space variables can be recovered. More in general, the bisimulations in [21,10,22] are defined directly at the level of the dynamical system (either in discrete or continuous time) whereas DB is defined at the language level, as a relation between process terms.

6 Conclusion

We presented differential bisimulation, a behavioral relation for process calculi with ordinary differential equation (ODE) semantics. This study follows the line of research on equivalence relations for quantitative models of computation. In particular, differential bisimulation is defined as a relation over a discrete set of process terms inducing an aggregation of the ODEs, analogously to Markovian bisimulations for process calculi which lead to the lumping of the underlying Markov process. Differential bisimulation allows relating local states of somewhat *heterogenous* processes instead of essentially isomorphic ones, as required in previous work. In addition, it is given in terms of syntactic conditions and it does not involve universal quantifiers over the expressions determining the ODE system. This, together with a conceptual similarity with Markovian bisimulations, allowed for the development of a partition-refinement algorithm for computing differential bisimilarity, largely reusing available results in the Markovian setting. As with its Markovian counterparts, differential bisimulation provides only sufficient conditions for ODE lumping. In this respect, an interesting line of investigation will be how to relax the current assumptions to obtain coarser aggregations. Another interesting problem is whether differential bisimulation implies lumpability also of the underlying Markov chain obtained when considering a Markovian semantics.

References

1. Baier, C., Engelen, B., Majster-Cederbaum, M.E.: Deciding bisimilarity and similarity for probabilistic processes. J. Comput. Syst. Sci. 60(1), 187–231 (2000)
2. Bernardo, M.: A Survey of Markovian Behavioral Equivalences. In: Bernardo, M., Hillston, J. (eds.) Formal Methods for Performance Evaluation, LNCS, vol. 4486, pp. 180–219. Springer (2007)
3. Buchholz, P.: Exact and Ordinary Lumpability in Finite Markov Chains. Journal of Applied Probability 31(1), 59–75 (1994)
4. Buchholz, P.: Markovian Process Algebra: Composition and Equivalence. In: Proc. 2nd PAPM Workshop. Erlangen, Germany (1994)
5. Camporesi, F., Feret, J.: Formal reduction for rule-based models. ENTCS 276, 29–59 (2011)
6. Ciocchetta, F., Hillston, J.: Bio-PEPA: A framework for the modelling and analysis of biological systems. TCS 410(33-34), 3065–3084 (2009)
7. Danos, V., Feret, J., Fontana, W., Harmer, R., Krivine, J.: Abstracting the differential semantics of rule-based models: Exact and automated model reduction. In: LICS. pp. 362–381 (2010)
8. Danos, V., Laneve, C.: Formal molecular biology. TCS 325(1), 69–110 (2004)
9. Derisavi, S., Hermanns, H., Sanders, W.H.: Optimal state-space lumping in Markov chains. Inf. Process. Lett. 87(6), 309–315 (2003)
10. Haghverdi, E., Tabuada, P., Pappas, G.J.: Bisimulation relations for dynamical, control, and hybrid systems. TCS 342(2—3), 229–261 (2005)
11. Hayden, R.A., Bradley, J.T.: A fluid analysis framework for a Markovian process algebra. TCS 411(22-24), 2260–2297 (2010)
12. Hermanns, H., Rettelbach, M.: Syntax, semantics, equivalences, and axioms for MTIPP. In: Proceedings of Process Algebra and Probabilistic Methods. pp. 71–87. Erlangen (1994)
13. Hermanns, H., Siegle, M.: Bisimulation algorithms for stochastic process algebras and their BDD-based implementation. In: ARTS. pp. 244–264 (1999)
14. Hillston, J.: A Compositional Approach to Performance Modelling. CUP (1996)
15. Iacobelli, G., Tribastone, M., Vandin, A.: Differential Bisimulation for a Markovian Process Algebra. Extended Version. QUANTICOL TR-QC-04-2015 (2015), `http://milner.inf.ed.ac.uk/wiki/files/W232G9A7/mfcs2015ExtendedTRpdf.html`
16. Kurtz, T.G.: Solutions of ordinary differential equations as limits of pure jump markov processes. Journal of Applied Probability 7(1), 49–58 (1970)
17. Kurtz, T.G.: Approximation of population processes, vol. 36. SIAM (1981)
18. Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. Information and Computation 94(1), 1–28 (1991)
19. Okino, M.S., Mavrovouniotis, M.L.: Simplification of mathematical models of chemical reaction systems. Chemical Reviews 2(98), 391–408 (1998)
20. Paige, R., Tarjan, R.: Three partition refinement algorithms. SIAM 16(6), 973–989 (1987)
21. Pappas, G.J.: Bisimilar linear systems. Automatica 39(12), 2035–2047 (2003)
22. van der Schaft, A.J.: Equivalence of dynamical systems by bisimulation. IEEE TAC 49 (2004)
23. Toth, J., Li, G., Rabitz, H., Tomlin, A.S.: The effect of lumping and expanding on kinetic differential equations. SIAM Journal on Applied Mathematics 57(6), 1531–1556 (1997)
24. Tribastone, M., Gilmore, S., Hillston, J.: Scalable differential analysis of process algebra models. IEEE TSE 38(1), 205–219 (2012)
25. Tschaikowski, M., Tribastone, M.: A unified framework for differential aggregations in Markovian process algebra. JLAMP 84(2), 238–258 (2015)
26. Tschaikowski, M., Tribastone, M.: Exact fluid lumpability for Markovian process algebra. In: CONCUR. pp. 380–394 (2012)