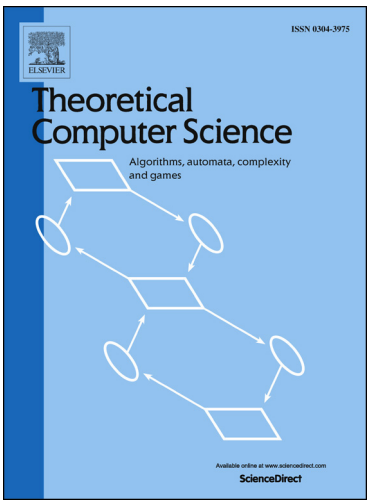
Accepted Manuscript



Symbolic computation of differential equivalences

Luca Cardelli, Mirco Tribastone, Max Tschaikowski, Andrea Vandin

PII: S0304-3975(19)30171-9
DOI: https://doi.org/10.1016/j.tcs.2019.03.018
Reference: TCS 11950

To appear in: *Theoretical Computer Science*

Received date: 15 June 2018
Revised date: 11 December 2018
Accepted date: 9 March 2019

Please cite this article in press as: L. Cardelli et al., Symbolic computation of differential equivalences, *Theoret. Comput. Sci.* (2019), https://doi.org/10.1016/j.tcs.2019.03.018

Symbolic Computation of Differential Equivalences

Luca Cardelli

University of Oxford, UK

Mirco Tribastone

IMT School for Advanced Studies Lucca, Italy

Max Tschaikowski

Vienna University of Technology, Austria

Andrea Vandin

DTU Compute, Technical University of Denmark, Denmark

Abstract

Ordinary differential equations (ODEs) are widespread in many natural sciences including chemistry, ecology, and systems biology, and in disciplines such as control theory and electrical engineering. Building on the celebrated molecules-as-processes paradigm, they have become increasingly popular in computer science, with high-level languages and formal methods such as Petri nets, process algebra, and rule-based systems that are interpreted as ODEs.

We consider the problem of comparing and minimizing ODEs automatically. Influenced by traditional approaches in the theory of programming, we propose *differential equivalence relations*. We study them for a basic intermediate language, for which we have decidability results, that can be targeted by a class of high-level specifications. An ODE implicitly represents an uncountable state space, hence reasoning techniques cannot be borrowed from established domains such as probabilistic programs with finite-state Markov chain semantics. We provide novel *symbolic* procedures to check an equivalence and compute the largest one via partition refinement algorithms that use satisfiability modulo theories.

We illustrate the generality of our framework by showing that differential equivalences include (i) well-known notions for the minimization of continuous-time Markov chains (lumpability), (ii) bisimulations for chemical reaction networks recently proposed by Cardelli *et al.*, and (iii) behavioral relations for process algebra with ODE semantics. Using ERODE, the tool that implements our techniques, we are able to detect equivalences in biochemical models from the literature that cannot be reduced using competing automatic techniques.

Keywords: Quantitative Equivalence Relations, Satisfiability Modulo Theory, Ordinary Differential Equations, Partition Refinement

1. Introduction

Ordinary differential equations (ODEs) are a widespread mathematical model to describe the time-course evolution of systems that can be characterized by continuously varying quantities. Classical examples are concentrations of species in chemical reactions and in biological processes, pressure and temperature in a plant, and voltage and current in an electrical circuit. Much more recently there has been an increasing attention to quantitative models of computation

Email addresses: luca.a.cardelli@gmail.com (Luca Cardelli), mirco.tribastone@imtlucca.it (Mirco Tribastone), max.tschaikowski@tuwien.ac.at (Max Tschaikowski), anvan@dtu.dk (Andrea Vandin)

based on ODEs, for example to use formal languages to describe biochemical models [73, 36, 11, 16, 20, 69, 31, 79] or as a deterministic approximation for languages with stochastic semantics [31, 55, 91].

In this paper we consider the fundamental problem of automatically comparing and minimizing programs with ODE semantics. From a mathematical viewpoint, the models of our interest are systems of coupled equations in n variables, $x = (x_1, \ldots, x_n)$, where

$$\dot{x}_i = f_i(x), \qquad i = 1, \ldots, n,$$

and f_i is the *drift*, a real valued function giving the derivative with respect to time of variable x_i when the system's state is x.

IDOL. We study a basic formalism called IDOL— Intermediate Drift Oriented Language. It essentially gives a syntax for the drifts, covering a class of nonlinear ODEs for which the reasoning is decidable, hence amenable to automatic treatment. Although not every ODE system can be directly written in IDOL (e.g., ODEs with trigonometric functions or exponentials), it covers a wide range of ODE models including:

- **Linear ODE systems.** This is a very important class of models in many disciplines including control theory and electrical engineering. Here we remark that a continuous-time Markov chain (CTMC), a very popular stochastic semantics for higher-level quantitative languages (see [10] and references therein) can also be directly seen as a linear ODE system through its Kolmogorov equations (also called the *master equation*). These equations give the probability of being in each state of the chain at any point in time [75].
- **Chemical reaction networks.** Chemical reaction networks (CRNs) express interactions between chemical species or molecular compounds. IDOL allows the specification of relevant (nonlinear) kinetics such as the well-known *law of mass action*, where the reaction rate is proportional to the product of the concentrations of the reagents; and the Hill kinetics, which involves rational expressions of polynomials in the species variables [97].
- **Quantitative models of computing systems.** Some formal methods with ODE semantics such as Petri nets [39] and process algebra [55, 91] have nonlinear laws of interaction based on threshold-like functions to model resource contention. For instance, these are used to model the firing rate of transition in a Petri net as being proportional to the *minimum* among the number of tokens at its incoming places [39].

Relating IDOL programs. We cast the problem of relating IDOL programs into the traditional context of equivalences for more classical models of computation based on labeled transition systems (LTS). We put forward the analogy between states of an LTS and IDOL variables. Thus, our equivalences are between variables, (exactly) preserving their ODE solutions in some appropriate sense. We propose two variants of *differential equivalence*.

The first variant is *forward differential equivalence* (FDE). This is such that an ODE system can be written for the variables that represent the equivalence classes. To be more concrete, let us consider the following trivial, yet illustrative, ODE example:

$$\dot{x}_1 = -x_1, \qquad \dot{x}_2 = k_1 \cdot x_1 - x_2, \qquad \dot{x}_3 = k_2 \cdot x_1 - x_3, \tag{1}$$

where k_1 and k_2 are constants. Then, it turns out that there is an FDE relating x_2 and x_3. Indeed, we have

$$\dot{x}_1 = -x_1, \qquad (x_2 \dot{+} x_3) = \dot{x}_2 + \dot{x}_3 = (k_1 + k_2) \cdot x_1 - (x_2 + x_3).$$

By the change of variable $y = x_2 + x_3$, this is equivalent to writing

$$\dot{x}_1 = -x_1 \qquad \dot{y} = (k_1 + k_2) \cdot x_1 - y.$$

This *quotient ODE model* recovers the sum of the solutions of the variables in each equivalence class. That is, we have that setting the *initial condition* $y(0) = x_2(0) + x_3(0)$ yields that $y(t) = x_2(t) + x_3(t)$ at all time points t.

Our second variant is *backward differential equivalence* (BDE). It equates variables that have the same solutions if they start from the same initial conditions. In (1), it can be shown that x_2 and x_3 are related also by BDE when $k_1 = k_2$. In this case, we obtain a quotient ODE by removing either equation, say x_3, and rewriting every instance of x_3 into x_2:

$$\dot{x}_1 = -x_1 \qquad \dot{x}_2 = k_1 x_1 - x_2.$$

If one starts with $x_2(0) = x_3(0)$ in (1) then the solution of the quotient ODE gives that $x_2(t) = x_3(t)$ at all time points t.

Since in BDE every variable in the same equivalence class has the same solution, the original model can be fully recovered. On the other hand, from the quotient FDE model one cannot recover the original solutions, but FDE poses no restriction on the initial conditions.

Checking and computing IDOL equivalences. An IDOL variable corresponds to a real function, thus it represents a *continuous state space*: proving two IDOL variables equivalent concerns relating two real-valued functions for all possible assignments—which involves reasoning over an uncountable state space. A major consequence is that established techniques for checking and computing equivalence relations over models based on LTSs with discrete state spaces (e.g., [77, 64, 61, 58, 6]) do not carry over.

We tackle this problem by proposing a *symbolic* approach based on satisfiability modulo theories (SMT) [7]. We encode differential equivalences into satisfiability problems of quantifier-free first-order logic formulae containing IDOL terms. Checking candidate relations amounts to establishing their validity, as usual through the unsatisfiability of their negation. The SMT solver that we use, the well-known Z3 [40], is a *decision procedure* for such formulae, so it can answer whether or not they are valid.

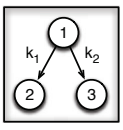
More importantly, we provide an automatic technique to compute the *largest* differential equivalence for an IDOL model, which is very relevant for minimization because it yields the smallest quotient ODE system. We do this by developing a partition refinement algorithm (cf. [77]) to which we introduce two novelties.

For FDE, we are able to establish a key technical result. We start from its classic definition in terms of a linear transformation of the ODE variables that preserves the aggregated dynamics, e.g., [89, 76, 2]. This definition requires to check "higher-order" properties, i.e., it involves (partition) blocks of variables, instead of individual variables. Thus, if a block does not satisfy the FDE condition, no information can be derived about how to split the current candidate partition. Instead, we equivalently verify FDE in terms of checks that involve only two IDOL variables at a time; that is, we characterize FDE in a form that does enable partition refinement.

For BDE, the partition refinement is counter-example guided. We fully exploit the ability of an SMT solver to produce an assignment of the variables that falsifies the assertion that a candidate partition is a BDE. The algorithm splits partition blocks according to such assignment; the iterative procedure terminates with the coarsest BDE partition when a distinguishing assignment is not found.

Applications. Many apparently unrelated formalisms and languages can benefit from the common framework provided by IDOL. In order to support our claim we consider three applications: CTMCs, CRNs, and process algebras.

Continuous-time Markov chains. The properties captured by FDE and BDE are analogous to ordinary and exact lumpability for CTMCs [14], respectively, and many behavioral equivalences for higher-level languages based on these notions (e.g., [70, 59, 57, 15, 37, 85, 10, 41, 48]). Indeed, we use the terms "forward" and "backward" to align with the terminology used in some of this literature (e.g., [85, 48, 23]). Actually, we show that FDE and BDE correspond to their respective variants of lumpability when the IDOL program is a linear ODE system representing a CTMC. For instance, the ODEs (1) are the Kolmogorov equations of the simple CTMC with state-transition diagram in the inset below, where x_i is the probability of finding the process in state i. However our differential equivalences are more general: they do not require k_1 and k_2 to be nonnegative, and can establish relations also for nonlinear ODEs. For instance, using the nonlinear ODE $\dot{x}_1 = -x_1^2$ in (1) we would equate x_2 and x_3 in the forward as well as in backward sense.



Chemical reaction networks and their extensions. CRNs have received increased attention in computer science due to the powerful analogy between computational processes and biological systems [82, 49, 22]. In addition to being a relevant model *per se*, CRNs are also closely related to many other languages. Cardelli establishes a correspondence between his Chemical Ground Form and CRNs [20]; rule-based languages such as κ [36] and BioNetGen [11] provide compact descriptions of biomolecular systems that can be "compiled down" to CRNs; Petri nets with an appropriate mass-action semantics on the transitions correspond to CRNs (e.g., [56]).

The idea of formally relating the dynamics of CRNs has recently emerged. In [21] Cardelli presents the notion of *emulation* between two CRNs as a property that exactly relates the ODE trajectories of a source CRN to those of a target CRN. Syntactic conditions are given to establish an emulation under the assumption of mass-action kinetics. Here we show that BDE is more general than emulation. As an application, we find that the emulations found for a

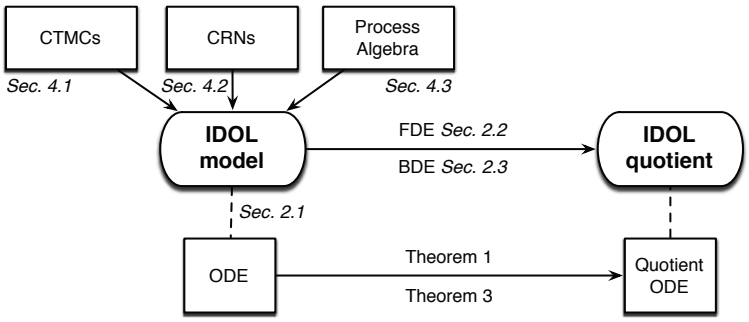


Figure 1: Paper overview.

class of biological processes in [21] are preserved even when an alternative dynamics based on the Hill kinetics is considered. This reinforces the findings in [21] that networks with different biological functionality are indeed related structurally, in a way that is insensitive to the underlying kinetics assumed.

In [23] Cardelli *et al.* present *forward* and *backward* bisimulations. The intent is analogous to ours, but those are equivalences that can be detected syntactically by inspecting the set of reactions. However, they only apply to a class of *elementary* CRNs with mass-action semantics (namely, reactions with at most two reagents). Under these restrictions we show that forward bisimulation is only a sufficient condition for FDE; instead, backward bisimulation corresponds to BDE. Furthermore, in [23, 25] a polynomial time partition refinement algorithm to compute the largest bisimulations is provided.

Forward and backward bisimulation have been recently extended in [27] to forward and backward *equivalence*, allowing for a finitary syntactic characterization of ODE systems whose right-hand sides are given in terms of multivariate polynomials of any degree. Similarly to the largest forward and backward bisimulation [23], the largest forward and backward equivalence is described in terms of syntactic checks and can be computed by an algorithm whose time and space complexity is polynomial in the number of ODE variables and monomials [27]. At the same time, forward and backward equivalence characterize FDE and BDE, respectively, provided that FDE and BDE are restricted to polynomial ODE systems. We use benchmarks, including those from [23], to present two findings of experimental nature: (i) we confirm that FDE and BDE coincide with forward and backward equivalence, respectively, with FDE and BDE being more computationally expensive, and we show that FDE is in turn more computationally expensive than BDE; and (ii) despite certain non-polynomial ODE systems can be transformed into polynomial ones using transformations like [53], this "polynomization" might break equivalences present in the original system.

Process algebra. Our last application is a fragment of Hillston's PEPA [59]. This is a stochastic process algebra which has been more recently equipped also with an ODE semantics with nonlinear minimum-based drifts that approximate the average evolution of underlying CTMCs with massively parallel computations [60, 55, 91]. We take PEPA as the representative of a family of languages for the quantitative evaluation of computing systems — indeed it is expressive enough to cover the semantics of a popular class of queuing networks [90] and Petri nets [13, 50]. We show that recently proposed behavioral equivalences for PEPA [92, 62] are special, language-specific cases of our differential equivalences.

Summary. Figure 1 provides a pictorial representation of the structure and the main results of this paper. The experiments herein reported are replicable via the tool ERODE [28] available at `http://sysma.imtlucca.it/tools/erode`, following the instructions provided in the accompanying web-page `http://sysma.imtlucca.it/tools/erode/comparison-syntactic-symbolic`. This paper extends [24] by publishing all proofs and by relating to forward and backward equivalence which were published after our conference paper [24]. In particular, in Section 4.2 we first compare against [27] in terms of performance. Then, we compare in terms of the reduction power of our approach when [27] is combined with the polynomial transformation technique [53] which allows one to transform a (sufficiently smooth) non-polynomial ODE system into a polynomial one.

Further related work. We are not aware of general automated approaches to ODE equivalences as done in this paper, though there is a large literature of techniques in domain-specific situations. The combinatorial explosion of CRN biochemical models has spurred considerable research in this area, e.g., [34, 35, 47, 38, 18, 45, 17, 23, 46, 19]. In particular, the fragmentation approach for κ identifies a coarse-grained ODE system for models with mass-action semantics through sums of variables; this is weaker than an equivalence relation over species, because one variable may appear in more than one block (a *fragment*) [47, 38]. Instead, [18, 17] discuss equivalence relations for species of reaction networks that are implicitly described by a set of κ rules. The algebraic approach [46] focusses on transformation rules of site graphs and deals in a unifying way with single push-out semantics. In particular, sufficient conditions for forward and backward bisimulation over population semantics are derived. Using the terminology of [76], fragmentation is a form of *improper lumping*, as opposed to our differential equivalences where species belong to a single block. As such it can still be seen as an ODE aggregation obtained through a transformation of the variables by a linear matrix, for which the general theory is well established (see [89, 76, 2]) but no general algorithms for computing the largest equivalences are available.

For polynomial ODE systems, the conditions of backward differential equivalence describe a particular family of differential invariants [52, 12, 81]. While [52, 81] use differential invariants to estimate the reachable set of polynomial ODE systems, [12] considers also their reduction. The corresponding reduction algorithm, however, does not enjoy a polynomial time complexity in general. Instead, for polynomial ODE systems, differential equivalences can be computed by algorithms whose time and space complexity are polynomial in system's size [27].

SMT has become a cornerstone in the programming languages and in the verification community, with contributions to program synthesis [54], constraint programming [67], and symbolic optimization [71]. The combination of SMT and equivalence relations has been the subject of recent investigations. In [9] partition-refinement algorithms are proposed to compute equivalences between terms over arbitrary theories inferred from a set of axioms. Applied to our context, these partition-refinement algorithms could be used to check if a candidate partition is a differential equivalence, but not to compute the largest equivalence for an IDOL program. In [42] the authors present an SMT-based approach for the computation of the coarsest ordinary lumpable partition of a Markov chain defined in a fragment of PRISM's [68] input language.

Finally, links between ODEs and SMT are established in the formal verification community, especially for hybrid systems (e.g., [51, 83, 72]); however none of these works considers ODE comparisons and minimizations through equivalence relations. Still at the interface between control theory and computer science, the idea of bisimulation for dynamical systems has been developed in a series of works by Pappas and coauthors [78] and van der Schaft [96]. These works are similar in spirit to ours, but the setting is different because the focus is on *control systems*, i.e., dynamical systems with internal states, external inputs, and output maps. In that context, bisimulation relates internal states mapped to the same output, i.e., they cannot be told apart by an external observer. The largest bisimulation is therefore related to the maximal unobservability subspace of a control system (e.g., [96, Corollary 6.4]) while our largest differential equivalences provide the coarsest partition of ODE variables that preserves the dynamics.

2. Intermediate Drift Oriented Language

We first introduce IDOL as a language to define a class of ODEs. More precisely, IDOL can describe nonlinear, first-order, autonomous, and explicit finite systems of coupled ODEs. Then, we present the notions of FDE and BDE as equivalence relations over IDOL variables, with their characterizations in terms of properties enjoyed by the underlying semantics.

2.1. Syntax and Semantics

Definition 1 (IDOL syntax)**.** *The syntax of programs of the* intermediate drift oriented language *(IDOL) is given by*

$$p ::= \varepsilon \mid \dot{x}_i = f,\ p \qquad\qquad f ::= n \mid x_i \mid f + f \mid f \cdot f \mid f^{\frac{1}{m}},$$

where $x_i \in \mathbb{V}$ *and* $n, m \in \mathbb{Z}$ *and* $m \neq 0$*.*

The set $\mathbb{V}$ represents ODE variables. A program is a list of elements $\dot{x}_i = f$ where each element gives the drift f for ODE of the variable x_i. Given an IDOL program p, we define $\mathcal{V}_p = \{x_1, \ldots, x_n\}$ as the set of variables in p. We say

that p is *well-formed* if for every $x_i \in \mathcal{V}_p$ there exists a unique term $\dot{x}_i = f$ in p. We denote its drift by f_i. From now on we assume to work with well-formed programs only. Finally, we remark that the restriction to integer parameters allows us to encode rationals, which is without loss of generality in practice (e.g., [4]).

Although some results presented below hold for richer classes of ODE systems, drifts expressible in IDOL form a class for which our differential equivalences are decidable. Despite the minimality of IDOL, it is possible to encode frequently used dynamics such as:

- the law of mass action, with drifts such as $x_1 \cdot x_2$;
- the Hill kinetics for CRNs, with drifts such as $x_1^2/(1 + x_1^2)$;
- and the minimum function for threshold based drifts, where

$$\min(x_1, x_2) := \frac{1}{2}(x_1 + x_2 - |x_1 - x_2|), \quad \text{with } |x| := (x \cdot x)^{\frac{1}{2}}.$$

For the semantics of IDOL, we interpret each term f_i in a standard way, as a real function of real variables on an appropriate domain, $D(f_i) \subseteq \mathbb{R}^{\mathcal{V}_p}$, where the function is *well-defined*, i.e., with no division by zero or negative arguments in roots. We denote by σ an assignment of variables in p, thus $\sigma \in \mathbb{R}^{\mathcal{V}_p}$. The semantics of IDOL depends on a *context*: this is a pair $c = (T, \hat{\sigma})$ that contains a time horizon $T > 0$ and an *initial assignment* $\hat{\sigma}$. The semantics of a program p is a function that maps the variables $\mathcal{V}_p$ to a continuous trajectory $[\![x]\!]_c^p : [0; T] \to \mathbb{R}^{\mathcal{V}_p}$ that describes the time course of every variable when starting from a given initial assignment $\hat{\sigma}$. In other words, $[\![x_i]\!]_c^p(t)$ is the value of the variable x_i at time t when $[\![x_i]\!]_c^p(0) = \hat{\sigma}(x_i)$.

Definition 2 (IDOL Semantics)**.** *The semantics of an IDOL program p in a context $c = (T, \hat{\sigma})$ is the unique differentiable function*

$$[\![x]\!]_c^p := \left([\![x_i]\!]_c^p\right)_{x_i \in \mathcal{V}_p}, \quad [\![x]\!]_c^p : [0; T] \to \mathbb{R}^{\mathcal{V}_p}$$

that satisfies

$$[\![x_i]\!]_c^p(t) = \hat{\sigma}(x_i) + \int_0^t [\![f_i]\!]_c^p([\![x]\!]_c^p(s))ds, \quad \textit{for all } 0 \le t \le T,$$

where $[\![f_i]\!]_c^p : D(f_i) \to \mathbb{R}$ is recursively defined as follows:

$$\begin{aligned}
[\![n]\!]_c^p &: \mathbb{R}^{\mathcal{V}_p} \to \mathbb{R}, & [\![n]\!]_c^p(\sigma) &= n \\
[\![x]\!]_c^p &: \mathbb{R}^{\mathcal{V}_p} \to \mathbb{R}, & [\![x]\!]_c^p(\sigma) &= \sigma(x) \\
[\![g + h]\!]_c^p &: D(g) \cap D(h) \to \mathbb{R}, & [\![g + h]\!]_c^p(\sigma) &= [\![g]\!]_c^p(\sigma) + [\![h]\!]_c^p(\sigma) \\
[\![g \cdot h]\!]_c^p &: D(g) \cap D(h) \to \mathbb{R}, & [\![g \cdot h]\!]_c^p(\sigma) &= [\![g]\!]_c^p(\sigma) \cdot [\![h]\!]_c^p(\sigma) \\
[\![g^{\frac{1}{m}}]\!]_c^p &: D(g^{\frac{1}{m}}) \to \mathbb{R}, & [\![g^{\frac{1}{m}}]\!]_c^p(\sigma) &= ([\![g]\!]_c^p(\sigma))^{\frac{1}{m}}
\end{aligned}$$

with $D(g^{\frac{1}{m}}) = \{\sigma \in D(g) : ([\![g]\!]_c^p(\sigma))^{\frac{1}{m}}$ is defined$\}$. If no such unique function exists, we call p ill-posed*.*

As usual we call $[\![f]\!]_c^p = ([\![f_i]\!]_c^p)_{x_i \in \mathcal{V}_p}$ the *vector field* of program p in context c. Also, the semantics of an even root term is given by the nonnegative solution; e.g., in any context c, $([\![4]\!]_c^p)^{1/2}$ is 2 and not -2. We remark that, in general, no closed-form expressions for $[\![x_i]\!]_c^p$ exist. However these functions can be computed using standard numerical integration algorithms, cf. [3].

Assumptions. IDOL is permissive enough to define somewhat degenerate ODEs with no solutions like $\dot{x}_1 = x_1^{-1}$, with $\hat{\sigma}(x_1) = 0$, or multiple solutions as in $\dot{x}_1 = |x_1|^{1/2}$, with $\hat{\sigma}(x_1) = -1$. We exclude these cases making certain assumptions that are usual when dealing with ODEs (e.g., [83]). For this, we define a notion of invariance which considers a subset of the drifts' domain containing the trajectories of the IDOL variables starting from any initial condition within that set.

Definition 3. *Given a program p and a time horizon $T > 0$, a set $E \subseteq \bigcap_{x_i \in \mathcal{V}_p} D(f_i)$ is* invariant with respect to p and T *if, for all $\hat{\sigma} \in E$ and $t \in [0; T]$, it holds that $[\![x]\!]_c^p(t) \in E$, where $c = (T, \hat{\sigma})$.*

Now we make the following assumptions:

A1 For a given time horizon $T > 0$ and IDOL program p, we can fix some $E(p) \in \{\mathbb{R}_{>0}^{\mathcal{V}_p}, \mathbb{R}_{\geq 0}^{\mathcal{V}_p}, \mathbb{R}^{\mathcal{V}_p}\}$ that is invariant and that satisfies $E(p) \subseteq \bigcap_{x_i \in \mathcal{V}_p} D(f_i)$.

A2 For all $x_i \in \mathcal{V}_p$, the function $[\![f_i]\!]_c^p : D(f_i) \to \mathbb{R}$ is locally Lipschitz continuous at any point of $E(p)$.

A1 is a technical assumption that allows us to work with nice enough domains when reasoning about differential equivalences. Our theory can be developed for more general invariant sets, but at the expense of significantly more convoluted mathematical definitions which do not seem to add substantial value to our contribution. Instead, **A2** is a standard textbook condition to ensure the existence of a unique solution, hence to exclude ill-posedness.

In many applications, models are typically such that a) the solution will be positive if the initial condition is positive and b) the drift is well-defined on positive reals. Under such circumstances, local Lipschitz continuity is usually immediate. Indeed, all IDOL programs presented in this paper satisfy these assumptions (and we will avoid stating which invariant set they have).

Notation. Differential equivalences are partitions of IDOL variables. Whenever convenient, for a program p and a given partition $\mathcal{H}$ of $\mathcal{V}_p$, we write $H = \{x_{H,1}, \ldots, x_{H,|H|}\}$ for any $H \in \mathcal{H}$. As usual, we denote by $\psi[t/s]$ the term that arises by replacing each occurrence of t in ψ by s.

2.2. Forward Differential Equivalence

With FDE one can write an IDOL program with one variable for each equivalence class, representing the sum of the trajectory solutions of its members. A partition $\mathcal{H}$ is induced by an FDE if the *aggregated drift* $\sum_{x_i \in H} f_i$ of any block $H \in \mathcal{H}$ can be written in terms of the sums of the variables within the block $\{\sum_{x_i \in H} x_i : H \in \mathcal{H}\}$. For instance, in the IDOL program

$$\dot{x}_1 = -2x_1 - 3x_2 - 4x_3 \qquad \dot{x}_2 = -3x_1 - 4x_2 - 5x_3$$
$$\dot{x}_3 = -6x_1 - 4x_2 - 2x_3 \qquad \dot{x}_4 = x_1 + x_2 + x_3 - 2x_4$$

the aggregated drifts for the partition $\{\{x_1, x_2, x_3\}, \{x_4\}\}$ are

$$f_1 + f_2 + f_3 = -11 \cdot (x_1 + x_2 + x_3) + 0 \cdot x_4, \qquad f_4 = 1 \cdot (x_1 + x_2 + x_3) - 2 \cdot x_4.$$

Clearly they depend only on the values of $x_1 + x_2 + x_3$ and x_4.

Answering the question whether sums of variables can be factored out from the aggregated drifts means finding new appropriate functions with arity equal to the number of equivalence classes. In this example, we would have drifts g_1 and g_2 defined as

$$g_1 = -11 \cdot y_1 \qquad g_2 = 1 \cdot y_1 - 2 \cdot y_2$$

where y_1 and y_2 represent blocks $\{x_1, x_2, x_3\}$ and $\{x_4\}$, respectively. We avoid synthesizing these functions directly by exploiting an alternative characterization that involves reasoning on properties concerning only the original variables: The evaluation of the aggregated drift must be invariant under any change of assignment of the variables that preserves the sum of values across each block.

To do this formally, we rewrite each variable as a scaling of the corresponding sums-of-variables of its block, such that all scaling factors are nonnegative and sum to one; in the example, we rewrite x_1 with $s_1(x_1 + x_2 + x_3)$, x_2 with $s_2(x_1 + x_2 + x_3)$, and $s_3(x_1 + x_2 + x_3)$ with scaling factors s_1, s_2, and s_3. The alternative characterization consists in proving that the aggregated drifts do not depend on the assignments of the scaling factors.

Importantly, following [89] it can be shown that if such rewriting does not change the values of the aggregated drifts for *some* choice of the scaling factors, then *any* choice will enjoy this property. The notion of FDE checks this using the *uniform* scaling that gives equal weight to every variable in the block (for instance $s_1 = s_2 = s_3 = 1/3$ in the example above). By assumption **A1**, the uniform scaling ensures that terms are always rewritten into terms that give rise to well-defined functions.

We encode this property in first order logic with function symbols from IDOL that are interpreted in the standard way, having $\mathcal{V}_p$ as free variables. We denote by $\Theta(p)$ the logical formula that encodes $E(p)$ (e.g., if $E(p) = \mathbb{R}_{>0}^{\mathcal{V}_p}$ then $\Theta(p) := \bigwedge_{x_i \in \mathcal{V}_p} x_i > 0$).

Definition 4 (FDE)**.** *Let p be an IDOL program and $\mathcal{H}$ a partition of $\mathcal{V}_p$. Then, $\mathcal{H}$ is a* forward differential equivalence *if the following formula is valid:*

$$\Theta(p) \rightarrow \bigwedge_{H\in\mathcal{H}} \Big(\sum_{x_i\in H} f_i = \sum_{x_i\in H} f_i\Big[x_j \Big/ \frac{\sum_{x_k\in H'} x_k}{|H'|} : H' \in \mathcal{H},\ x_j \in H'\Big]\Big) \qquad (\mathbf{\Phi}^{\mathcal{H}})$$

The next definition provides the quotient IDOL program with respect to an FDE.

Definition 5 (FDE Quotient)**.** *Let p be an IDOL program and $\mathcal{H}$ an FDE partition. Then, the* forward quotient of p *with respect to $\mathcal{H}$, denoted by $\overrightarrow{p}_{\mathcal{H}}$, is given by:*

$$\dot{y}_H = \sum_{x_i\in H} f_i\Big[x_j \Big/ \frac{y_{H'}}{|H'|} : H' \in \mathcal{H},\ x_j \in H'\Big],\ \textit{for all } H \in \mathcal{H}.$$

The uniform scaling and **A1** ensure that $\overrightarrow{p}_{\mathcal{H}}$ is *not* ill-posed.

We now state a crucial *dynamical characterization* theorem: A partition of IDOL variables is FDE if and only if the ODEs of the quotient program preserve the sums of the original trajectories in each equivalence class. Hence the largest FDE represents the best possible aggregation that can be obtained in this sense.

Theorem 1 (Dynamical FDE Characterization)**.** *Let p be an IDOL program, $T > 0$ a time horizon and $\mathcal{H}$ a partition of $\mathcal{V}_p$. Then, $\mathcal{H}$ is an FDE partition with forward quotient $\overrightarrow{p}_{\mathcal{H}}$ if and only if*

$$[\![y_H]\!]_{\tilde{c}}^{\overrightarrow{p}_{\mathcal{H}}}(t) = \sum_{x_i\in H} [\![x_i]\!]_c^p(t)$$

for all $\hat{\sigma} \in E(p)$, $H \in \mathcal{H}$ and $t \in [0;T]$, where $c := (T, \hat{\sigma})$, $\tilde{c} := (T, \hat{\sigma}_{\mathcal{H}})$ and $\hat{\sigma}_{\mathcal{H}}(y_H) := \sum_{x_i\in H} \hat{\sigma}(x_i)$ for all $H \in \mathcal{H}$.

Proof. The proof follows as a special case of Theorem 1.2 and Theorem 1.3 from [89] which consider general aggregation functions of which sums are a special case. In particular, the aforementioned theorems remain valid if f and $\hat{f}$ in [89] are assumed to be (only) continuous. Thus, since we instantiate aggregation functions h from [89] by matrices (see also the proof of Theorem 2 of the current work), the claim follows from Theorem 1.3 of [89]. □

Let us remark that FDE is stated in terms of a partition and is thus consistent with the notion of lumpability for ODEs [89]. This has the advantage that the above theorem is a direct consequence of the theory presented in [89], hence we omit the proof here. However, this is not in a form that enables an algorithm for computing the largest FDE using partition refinement, because an assignment that falsifies the FDE conditions $\Phi^{\mathcal{H}}$ does not provide information about which variables to tell apart in the refinement step.

We tackle this problem by providing a characterization of FDE in terms of *binary* checks, i.e., involving two variables only at a time. Intuitively, for each block $H \in \mathcal{H}$ and any pair $x_i, x_j \in H$, such characterization allows to check if the fact that x_i and x_j belong to the same block prevents $\mathcal{H}$ from being an FDE.

More precisely, an equivalence relation $\mathcal{R}$ over $\mathcal{V}_p$ is FDE if and only if for all $(x_i, x_j) \in \mathcal{R}$ it holds that the aggregated drifts of all blocks in $\mathcal{H}$ are invariant under a scaling of the sum-of-variables which involves *only two* variables belonging to the same block, rather than *all of them* as per Definition 4. The intuition is that any scaling considered in Definition 4 can be equivalently achieved as a composition of such "binary" scalings. To our knowledge, such a binary characterization is proved here for the first time.

Theorem 2 (Binary FDE characterization)**.** *Let p be an IDOL program, $\mathcal{R}$ be an equivalence relation on $\mathcal{V}_p$, and $\mathcal{H} = \mathcal{V}_p/\mathcal{R}$. Then $\mathcal{H}$ is an FDE if and only if for all distinct $x_i, x_j \in \mathcal{V}_p$ we have that $(x_i, x_j) \in \mathcal{R}$ implies that the following formula is valid:*

$$\Theta(p) \rightarrow \bigwedge_{H\in\mathcal{H}} \Big(\sum_{x_k\in H} f_k = \sum_{x_k\in H} f_k[x_i/s\cdot(x_i+x_j), x_j/(1-s)\cdot(x_i+x_j)]\Big) \qquad (\mathbf{\Phi}^{\mathcal{H}}_{\mathbf{x_i},\mathbf{x_j}})$$

Proof. Note that s is not an ODE variable from $\mathbb{V}$ but a variable of the first order logic. However, we still denote the interpretation of any variable $\tilde{s} \notin \mathbb{V}$ by $\sigma(\tilde{s})$. To increase readability in the proof we shall also write $[\![\cdot]\!]$ instead of $[\![\cdot]\!]_c^p$.

Let us assume that $\Phi^{\mathcal{H}}_{x_i,x_j}$ is valid for all $H \in \mathcal{H}$ and $x_i, x_j \in H$. We have to show that $\Phi^{\mathcal{H}}$ is valid. For this, we fix arbitrary $\sigma \in E(p)$ and $H \in \mathcal{H}$ and assume without loss of generality that $H = \{x_1, \ldots, x_m\}$ and that $\sigma(x_i) \geq \sigma(x_{i+1})$ for all $1 \leq i \leq m-1$. Together with $\mu_H := \frac{1}{|H|} \sum_{1 \leq i \leq m} \sigma(x_i)$ we then define $\sigma^i \in \mathbb{R}^{\mathcal{V}_p}$, where $1 \leq i \leq m-1$, as

$$\sigma(x_k)^{i+1} := \begin{cases} \mu_H & , \ k = i \\ (1 - \sigma(s_i)) \cdot (\sigma^i(x_i) + \sigma^i(x_{i+1})) & , \ k = i+1 \\ \sigma^i(x_k) & , \ \text{otherwise} \end{cases}$$

with $\sigma^1 := \sigma$ and $\sigma(s_i) := \mu_H / (\sigma^i(x_i) + \sigma^i(x_{i+1}))$ for all $1 \leq i \leq m-1$. Since $\sigma(x_i) \geq \sigma(x_{i+1})$ for all $1 \leq i \leq m-1$, it holds that $0 < \sigma(s_i) \leq 1$ for all $1 \leq i \leq m-1$. Thus, since $\Phi^{\mathcal{H}}_{x_i,x_j}$ is valid for all $x_i, x_j \in H$, we infer that $[\![f]\!](\sigma^{i+1}) = [\![f]\!](\sigma^i)$ because both terms are equal to

$$[\![f\,[x_i/s_i(x_i + x_{i+1}), x_{i+1}/(1 - s_i)(x_i + x_{i+1})]]\!]\,(\tilde{\sigma}^i),$$

where $\tilde{\sigma}^i := \sigma^i \cup \{(s_i, \sigma(s_i))\} \in \mathbb{R}^{\mathcal{V}_p \cup \{s_i\}}$, for all $1 \leq i \leq m-1$. Hence, $[\![f]\!](\sigma^{|H|}) = [\![f]\!](\sigma^{|H|-1}) = \ldots = [\![f]\!](\sigma^1) = [\![f]\!](\sigma)$. Note also that $\sigma^{|H|}$ satisfies $\sigma^{|H|}(x_k) = \sigma^{|H|}(x_l)$ for all $x_k, x_l \in H$. By applying the above argument to the remaining blocks $\mathcal{H} \setminus \{H\}$ (i.e., in second step we would consider a block $H' \neq H$ and the vector $\sigma^{|H|}$), we infer that $\Phi^{\mathcal{H}}$ is true under the assignment σ. Since σ was chosen arbitrarily, $\mathcal{H}$ is an FDE.

For the proof of the converse, we first consider the case $E(p) = \mathbb{R}_{>0}^{\mathcal{V}_p}$ and define, for any partition $\mathcal{H}$ of $\mathcal{V}_p$, the matrix $M_{\mathcal{H}} \in \{0,1\}^{\mathcal{H} \times \mathcal{V}_p}$ by setting $(M_{\mathcal{H}})_{H,x_k}$ to 1 if $x_k \in H$, and 0 otherwise. The matrix $M_{\mathcal{H}}$ can be thought of as an "aggregation" matrix. In particular, the rows of the matrix $M_{\mathcal{H}}$ encode the blocks of $\mathcal{H}$. Then, for any positive generalized right inverse of $M_{\mathcal{H}}$, i.e. a matrix $\overline{M}_{\mathcal{H}} \in (0;1]^{\mathcal{V}_p \times \mathcal{H}}$ that satisfies $M_{\mathcal{H}} \overline{M}_{\mathcal{H}} = \mathbb{I}$, the function $\sigma \mapsto M_{\mathcal{H}} \overline{M}_{\mathcal{H}} \sigma$ defines a scaling on $\mathcal{H}$. Since the entries of $\overline{M}_{\mathcal{H}}$ are positive, we infer $M_{\mathcal{H}} \overline{M}_{\mathcal{H}} \sigma \in \mathbb{R}_{>0}^{\mathcal{V}_p}$ for all $\sigma \in \mathbb{R}_{>0}^{\mathcal{V}_p}$. As pointed out at the beginning of Section 2.2, any scaling that yields well-defined terms is equivalent to the notion of FDE [89]. Consequently, $\mathcal{H}$ is an FDE if and only if there exists a generalized right inverse $\overline{M}_{\mathcal{H}} \in (0;1]^{\mathcal{V}_p \times \mathcal{H}}$ of $M_{\mathcal{H}}$ such that $M_{\mathcal{H}}([\![f]\!](\sigma)) = M_{\mathcal{H}}([\![f]\!](\overline{M}_{\mathcal{H}} M_{\mathcal{H}} \sigma))$ for all $\sigma \in \mathbb{R}_{>0}^{\mathcal{V}_p}$. With this in mind, let us now assume that $\mathcal{H}$ is an FDE partition and fix arbitrary $\sigma \in E(p)$, $H_0 \in \mathcal{H}$, $x_i, x_j \in H_0$ and $\sigma(s) \in (0;1]$. We next show that $\Phi^{\mathcal{H}}_{x_i,x_j}$ is true for the assignment σ. Fix the generalized right inverse

$$(\overline{M}_{\mathcal{H}})_{x_k,H} = \begin{cases} \frac{\sigma(s)(\sigma(x_i)+\sigma(x_j))}{\sum_{x_l \in H_0} \sigma(x_l)} & , \ x_k = x_i \\ \frac{(1-\sigma(s))(\sigma(x_i)+\sigma(x_j))}{\sum_{x_l \in H_0} \sigma(x_l)} & , \ x_k = x_j \\ \sigma(x_k) / (\sum_{x_l \in H} \sigma(x_l)) & , \ x_k \notin \{x_i, x_j\} \end{cases}$$

It is straightforward to show that

$$(M_{\mathcal{H}} \overline{M}_{\mathcal{H}} \sigma)(x_k) = \begin{cases} \sigma(s)(\sigma(x_i) + \sigma(x_j)) & , \ x_k = x_i \\ (1 - \sigma(s))(\sigma(x_i) + \sigma(x_j)) & , \ x_k = x_i \\ \sigma(x_k) & , \ \text{otherwise} \end{cases}$$

Since $\mathcal{H}$ was assumed to be an FDE, the above discussion implies that $M_{\mathcal{H}}([\![f]\!](\sigma)) = M_{\mathcal{H}}([\![f]\!](\overline{M}_{\mathcal{H}} M_{\mathcal{H}} \sigma))$. This, however, implies that $\Phi^{\mathcal{H}}_{x_i,x_j}$ holds true for the assignment σ. Since $\sigma \in E(p)$, $H_0 \in \mathcal{H}$ and $x_i, x_j \in H_0$ were chosen arbitrarily, we infer the claim in the case where $E(p) = \mathbb{R}_{>0}^{\mathcal{V}_p}$.

The case $E(p) \in \mathbb{R}^{\mathcal{V}_p}$ (resp. $E(p) \in \mathbb{R}_{\geq 0}^{\mathcal{V}_p}$) follows by generalizing our argumentation. In particular, the generalized right inverse has to be chosen from $\mathbb{R}^{\mathcal{V}_p \times \mathcal{H}}$ (resp. $[0;1]^{\mathcal{V}_p \times \mathcal{H}}$). Moreover, note that $(\overline{M}_{\mathcal{H}})_{x_k,H}$ is well-defined only if all aggregated variables underlying the fixed $\sigma \in \mathbb{R}^{\mathcal{V}_p}$ are nonzero. However, such assignments build a (Lebesgue) zero set of $E(p)$ and the vector field $[\![f]\!]$ is continuous on $E(p)$, which yields the claim. □

2.3. Backward Differential Equivalence

BDE relates IDOL variables having the same semantics whenever they are given the same initial assignment. This property is characterized by the following implication: if the variables in each block of the partition have the same equal assignments, then the drifts of any two variables of a block have equal values. Similarly to FDE, we formalize this in first order logic.

Definition 6 (BDE)**.** *Let p be an IDOL program and $\mathcal{H}$ a partition of $\mathcal{V}_p$. Then $\mathcal{H}$ is a* backward differential equivalence *if the following formula is valid:*

$$\Theta(p) \to \Big(\bigwedge_{H\in\mathcal{H}} (x_{H,1} = \ldots = x_{H,|H|}) \to \bigwedge_{H\in\mathcal{H}} (f_{H,1} = \ldots = f_{H,|H|}) \Big) \tag{$\Psi^{\mathcal{H}}$}$$

For instance, let us consider the IDOL program

$$\dot{x}_1 = -\min(x_1, 1) + x_2 \qquad\qquad \dot{x}_2 = -\min(x_2, 1) + x_1 \tag{2}$$

We seek to verify that $\{\{x_1, x_2\}\}$ is a BDE partition. Indeed this holds since $\Psi^{\mathcal{H}}$ becomes

$$x_1 = x_2 \to -\min(x_1, 1) + x_2 = -\min(x_2, 1) + x_1$$

Definition 7 (BDE Quotient)**.** *Let p be an IDOL program and $\mathcal{H}$ a BDE partition of $\mathcal{V}_p$. The* backward quotient of p *with respect to $\mathcal{H}$, denoted by $\overleftarrow{p_{\mathcal{H}}}$, is given by*

$$\dot{y}_H = f_{H,1}[x_{H',1}/y_{H'}, \ldots, x_{H',|H'|}/y_{H'} : H' \in \mathcal{H}], \text{ for } H \in \mathcal{H}.$$

Similarly to FDE, the BDE quotient is not ill-posed. For instance, the BDE quotient of (2) with respect to $\{\{x_1, x_2\}\}$ is given by

$$\dot{y} = (-\min(x_1, 1) + x_2)[x_1/y, x_2/y], \text{ i.e., } \dot{y} = -\min(y, 1) + y$$

The next characterization result is analogous to Theorem 1.

Theorem 3 (Dynamical BDE Characterization)**.** *Let p be an IDOL program, $T > 0$ a time horizon and $\mathcal{H}$ a partition of $\mathcal{V}_p$. Then, $\mathcal{H}$ is a BDE partition with backward quotient $\overleftarrow{p_{\mathcal{H}}}$ if and only if $\hat{\sigma}_{\mathcal{H}}(y_H) = \hat{\sigma}(x_{H,1}) = \ldots = \hat{\sigma}(x_{H,|H|})$ for all $H \in \mathcal{H}$ implies*

$$[\![y_H]\!]^{\overleftarrow{p_{\mathcal{H}}}}_{\tilde{c}}(t) = [\![x_{H,1}]\!]^{p}_{c}(t) = \ldots = [\![x_{H,|H|}]\!]^{p}_{c}(t)$$

for all $H \in \mathcal{H}$ and $t \in [0; T]$, with $c := (T, \hat{\sigma})$ and $\tilde{c} := (T, \hat{\sigma}_{\mathcal{H}})$.

Proof. The proof of Theorem 6 from [23] carries over to vector fields induced by IDOL. □

The statement is shown by using the same strategy as in the proof of Theorem 6 in [23].

Let us point out that the notions of FDE and BDE are not comparable. For instance, the partition $\{\{x_1, x_2\}\}$ is not an FDE of (2) because the formula

$$-\min(x_1, 1) + x_2 - \min(x_2, 1) + x_1 = -\min\Big(\frac{x_1 + x_2}{2}, 1\Big) + \frac{x_1 + x_2}{2} - \min\Big(\frac{x_1 + x_2}{2}, 1\Big) + \frac{x_1 + x_2}{2}$$

is not true for the assignment $\sigma(x_1) = 2$ and $\sigma(x_2) = 0$. Conversely, $\{\{x_1\}, \{x_2, x_3\}\}$ is an FDE partition of (1) for any choice of k_1 and k_2, but a BDE only if $k_1 = k_2$.

3. Computing Differential Equivalences

We now discuss how to compute differential equivalences and how to implement this using SMT. We first consider the problem of checking if a given partition is a differential equivalence. Then we focus on computing the largest differential equivalence for an IDOL program using partition refinement.

3.1. Checking Differential Equivalences

Tarski's famous result ensures that one can decide whether $\Phi^{\mathcal{H}}$, $\Phi^{\mathcal{H}}_{x_i,x_j}$ and $\Psi^{\mathcal{H}}$ are valid because the functions supported by IDOL can be expressed in the theory of reals $(\mathbb{R}, +, -, \cdot, 0, 1, <, =)$. For instance, terms with roots like $y = x^{\frac{1}{2}}$ can be encoded as $\exists y(y^2 = x)$, while the encoding of fractions is straightforward. However, no efficient computation is possible in general.

Proposition 1. *Deciding a differential equivalence is coNP-hard.*

Proof. Let $\mathcal{L}$ denote the language of valid sentences of the form $\forall x_1 \dots \forall x_n (f_i = f_j)$ where f_i and f_j denote the drifts of x_i and x_j, respectively. Further, let $\mathcal{T}$ denote the set of tautologies of propositional logic. We next show that there exists a reduction from $\mathcal{T}$ to $\mathcal{L}$ in polynomial time. Since there exists a polynomial time reduction from the complement of SAT to $\mathcal{T}$, this shows that deciding $\mathcal{L}$ is coNP-hard.

We next discuss the reduction from $\mathcal{T}$ to $\mathcal{L}$. Given a propositional formula ϕ with boolean variables $\underline{x}_1, \dots, \underline{x}_n$, we define f_ϕ by means of structural induction as follows:

$$\phi = \underline{x}_i : \text{set } f_\phi := x_i \qquad \phi = \neg\phi_0 : \text{set } f_\phi := 2 - f_{\phi_0}$$
$$\phi = \phi_0 \wedge \phi_1 : \text{set } f_\phi := \min(f_{\phi_0}, f_{\phi_1}) \qquad \phi = \phi_0 \vee \phi_1 : \text{set } f_\phi := \max(f_{\phi_0}, f_{\phi_1})$$

Note that the maximum function can be expressed in IDOL because the maximum can be expressed using the absolute value function. Intuitively, we model $\underline{x}_i = \bot$ and $\underline{x}_i = \top$ by $x_i \in (0, 1)$ and $x_i \in (1, 2)$, respectively. With this idea in mind, we define $\psi := \bigwedge_{i=1}^{n} (\underline{x}_i = \top \leftrightarrow 1 < x_i < 2 \wedge \underline{x}_i = \bot \leftrightarrow 0 < x_i < 1)$ for $\underline{x}_1, \dots, \underline{x}_n \in \{\top, \bot\}$ and $x_1, \dots, x_n \in \mathbb{R}^{\mathcal{V}}$, meaning that ψ is true if and only if the reals $x_1, \dots, x_n$ are consistent with the vector of booleans $\underline{x}_1, \dots, \underline{x}_n$. By performing a proof on the structural definition of ϕ, it is possible to show that

$$\forall \underline{x}_1 \forall x_1 \dots \forall \underline{x}_n \forall x_n (\psi \to (0 < f_\phi < 1 \vee 1 < f_\phi < 2) \ \wedge \ \psi \to (\phi \leftrightarrow f_\phi > 1)) \tag{3}$$

Note that the above formula is not in the fragment of first order logic in which FDE and BDE are encoded because it contains boolean variables. This, however, is not important because the above statement is only needed to ensure the correctness of our encoding which is defined as follows. If ϕ is not a well-defined propositional formula, we set $\theta(\phi) := \forall x_1 \dots \forall x_n (x_1 = 0)$ which is obviously not in $\mathcal{L}$. Instead, if ϕ is a well-defined propositional formula with boolean variables $\underline{x}_1, \dots, \underline{x}_n$, we define

$$\theta(\phi) := \forall x_1 \dots \forall x_n (\chi \cdot \max(f_{\neg\phi}, 1) = \chi)$$
$$\chi := \min_{1 \le i \le n} (|x_i - 1| \cdot \max(x_i, 0) \cdot \max(2 - x_i, 0))$$

Note that our reduction is polynomial in space and time. Moreover, observe that the nonnegative function χ is positive if and only if for all $1 \le i \le n$ it holds that $0 < x_i < 1 \vee 1 < x_i < 2$. With this, we infer using (3) that

$$\forall \underline{x}_1 \dots \forall \underline{x}_n (\phi = \top) \ \Leftrightarrow \ \forall x_1 \dots \forall x_1 (\chi > 0 \to f_{\neg\phi} < 1) \Leftrightarrow \ \forall x_1 \dots \forall x_n (\chi \cdot \max(f_{\neg\phi}, 1) = \chi)$$

□

Despite Proposition 1, in many cases the computation is feasible in practice. We provide examples in Section 4.2.2. Here we briefly discuss how an SMT solver can be used for this purpose. The validity of the quantifier-free formulae $\Phi^{\mathcal{H}}$, $\Phi^{\mathcal{H}}_{x_i,x_j}$ and $\Psi^{\mathcal{H}}$ can be encoded, as usual, into the unsatisfiability problem of their negation, i.e., by invoking $\mathsf{sat}(\neg\Phi^{\mathcal{H}})$, $\mathsf{sat}(\neg\Phi^{\mathcal{H}}_{x_i,x_j})$, and $\mathsf{sat}(\neg\Psi^{\mathcal{H}})$. These can be decided using the decision procedure nlsat [63], which is implemented in Z3 v4.0 [40]. Thus checking differential equivalences is sound and complete using state-of-the-art SMT technology: A partition $\mathcal{H}$ is FDE (resp., BDE) if and only if $\mathsf{sat}(\neg\Phi^{\mathcal{H}})$ (resp., $\mathsf{sat}(\neg\Psi^{\mathcal{H}})$) returns "unsatisfiable". As a concrete example, consider the IDOL program (1) and the partition $\bar{\mathcal{H}} = \{\{x_1\}, \{x_2, x_3\}\}$, which, as discussed, is a BDE if and only if the parameters k_1 and k_2 are equal. The executable Z3 encoding of $\neg\Psi^{\mathcal{H}}$ for both the cases $k_1 \neq k_2$ and $k_1 = k_2$ is available at `https://rise4fun.com/Z3/e9vV`.

3.2. Partition Refinement

We compute the largest differential equivalence for an IDOL program using a partition refinement algorithm. First, however, we show that this is a well-posed problem.

Definition 8 (Refinement)**.** *Let S be a set, and $\mathcal{H}_1$, $\mathcal{H}_2$ two partitions of S. Then, $\mathcal{H}_1$ is a* refinement *of $\mathcal{H}_2$ if for any block $H_1 \in \mathcal{H}_1$ there exists a block $H_2 \in \mathcal{H}_2$ such that $H_1 \subseteq H_2$.*

Theorem 4. *Let p be an IDOL program and $\mathcal{G}$ be a partition of $\mathcal{V}_p$. Then, there exists a unique coarsest FDE/BDE partition refining $\mathcal{G}$.*

Proof. For a given partition $\mathcal{H}$ of $\mathcal{V}_p$, write $x_i \sim_{\mathcal{H}} x_j$ whenever there exists some $H \in \mathcal{H}$ such that $x_i, x_j \in H$.

FDE case. Set $\sim_\Phi^{\mathcal{H}} := \{(x_i, x_j) : x_i = x_j \text{ or } \Phi^{\mathcal{H}}_{x_i,x_j} \text{ is valid}\}$ and note that $\sim_\Phi^{\mathcal{H}}$ is an equivalence relation on $\mathcal{V}_p$. We fix FDE partitions $\mathcal{H}_1, \dots, \mathcal{H}_n$ of $\mathcal{V}_p$ and set for the sake of brevity $\sim_i := \sim_{\mathcal{H}_i}$ and $\sim_* := \sim_{\mathcal{H}^*}$ where $\mathcal{H}^* := \mathcal{V}_p/(\bigcup_{i=1}^m \sim_i)^*$ and the asterisk denotes transitive closure. Thanks to Theorem 2, it suffices to prove that $y_1 \sim_\Phi^{\mathcal{H}^*} y_2$ for all $H^* \in \mathcal{H}^*$ and $y_1, y_2 \in H^*$. Thus, let us fix some $H^* \in \mathcal{H}^*$ and $y_1, y_2 \in H^*$. Since $\sim_\Phi^{\mathcal{H}^*}$ is transitive and $y_1 = x_0 \sim_{i_0} x_1 \sim_{i_1} \dots \sim_{i_{k-1}} x_k = y_2$ for some $x_0, \dots, x_k \in \mathcal{V}_p$ and $i_0, \dots, i_{k-1} \in \{1, \dots, n\}$, it suffices to show that $x_j \sim_\Phi^{\mathcal{H}^*} x_{j+1}$ for all $0 \le j \le k-1$. For this, let us fix some arbitrary $G^* \in \mathcal{H}^*$. Then, it can be easily seen that there exist (unique) subsets $\{G^i_1, \dots, G^i_{m_i}\} \subseteq \mathcal{H}_i$ such that $\biguplus_{l=1}^{m_i} G^i_l = G^*$ for all $1 \le i \le n$. Since $x_j \sim_{i_j} x_{j+1}$ implies that

$$\begin{aligned}\sum_{x_\iota \in G^*} f_\iota &= \sum_{l=1}^{m_{i_j}} \sum_{x_\iota \in G^{i_j}_l} f_\iota \\ &= \sum_{l=1}^{m_{i_j}} \sum_{x_\iota \in G^{i_j}_l} f_\iota[x_j/s(x_j + x_{j+1}), x_{j+1}/(1-s)(x_j + x_{j+1})] \\ &= \sum_{x_\iota \in G^*} f_\iota[x_j/s(x_j + x_{j+1}), x_{j+1}/(1-s)(x_j + x_{j+1})],\end{aligned}$$

we infer that $x_j \sim_\Phi^{\mathcal{H}^*} x_{j+1}$.

BDE case Define $\sim_\Psi^{\mathcal{H}} := \{(x_i, x_j) : \Psi^{\mathcal{H}}_{x_i,x_j} \text{ is valid}\}$, where $\Psi^{\mathcal{H}}_{x_i,x_j} := \Theta(p) \to (\bigwedge_{H \in \mathcal{H}} (x_{H,1} = \dots\dots = x_{H,|H|}) \to f_i = f_j)$ and note that $\sim_\Psi^{\mathcal{H}}$ is an equivalence relation on $\mathcal{V}_p$. Then, we fix BDE partitions $\mathcal{H}_1, \dots, \mathcal{H}_n$ of $\mathcal{V}_p$ and, by applying the very same reasoning as in the case of FDE, we have to show that $x_j \sim_\Psi^{\mathcal{H}^*} x_{j+1}$ for all $0 \le j \le k-1$. Since $x_j \sim_{i_j} x_{j+1}$ implies that $x_j \sim_\Psi^{\mathcal{H}_{i_j}} x_{j+1}$ and any block of $\mathcal{H}^*$ is a union of blocks of $\mathcal{H}_{i_j}$, it can be easily seen that $x_j \sim_\Psi^{\mathcal{H}_{i_j}} x_{j+1}$ implies $x_j \sim_\Psi^{\mathcal{H}^*} x_{j+1}$.

So far, we have shown that the coarsening $\mathcal{V}_p/(\bigcup_{i=1}^m \sim_i)^*$ of FDE (BDE) partitions $\mathcal{H}_1, \dots, \mathcal{H}_n$ is again an FDE (BDE) partition. The claim follows then by noting that Lemma 26 in [23] ensures that $\mathcal{V}_p/(\bigcup_{i=1}^m \sim_i)^*$ is a refinement of $\mathcal{G}$ if each $\mathcal{V}_p/\sim_i$ is a refinement of $\mathcal{G}$. □

The main difference with respect to the classical partition-refinement algorithms developed for discrete-state transition systems (e.g., [64, 43, 6]) is that each IDOL variable represents a continuous (uncountable) state space. To tackle this problem we build a variant which performs a symbolic evaluation at each iteration that checks the validity of the FDE/BDE conditions. As usual, the algorithm returns the coarsest FDE/BDE partition that refines a given input partition: This is the trivial partition $\{\mathcal{V}_p\}$ when computing the largest differential equivalence. We remark that the freedom in choosing the initial partition can be useful. For FDE, it allows to single out variables to be preserved in the aggregated program. These are the variables for which the modeler is interested in obtaining distinct ODE solutions. BDE requires equivalent variables to be initialized with same initial conditions. In this case, an appropriate $\mathcal{G}$ can be used to tell apart variables having different initial conditions. This is similar to the pre-partitioning for the largest bisimulation of a labeled Markov chain (e.g., [5]), where states with different sets of atomic propositions are told apart.

The outer loop of the algorithm, in Algorithm 1, is a classic fixed-point iteration. The specific refinement depends on an inner procedure, parameterized by the notion of differential equivalence that is considered ($\chi = F$ and $\chi = B$).

Algorithm 1 Construction of the largest FDE and BDE.

Require: Program p, context c, partition $\mathcal{G}$ of $\mathcal{V}_p$ and $\chi \in \{F, B\}$.

$\mathcal{H} \leftarrow \mathcal{G}$
while true do
  $\mathcal{H}' \leftarrow refine_\chi(\mathcal{H})$
  if $\mathcal{H}' = \mathcal{H}$ **then**
    return $\mathcal{H}$
  else
    $\mathcal{H} \leftarrow \mathcal{H}'$
  end if
end while

Algorithm 2: Routine $refine_F$

ROUTINE $refine_F(\mathcal{H})$
$\mathcal{H}' \leftarrow \emptyset$
for all $H \in \mathcal{H}$ **do**
  $\mathcal{R} \leftarrow \{(x_i, x_j) : x_i, x_j \in H$ and $(x_i = x_j$ or $\Phi^{\mathcal{H}}_{x_i,x_j}$ is valid$)\}$
  $\mathcal{H}' \leftarrow \mathcal{H}' \cup (H/\mathcal{R})$
end for
return $\mathcal{H}'$

Algorithm 3: Routine $refine_F$

ROUTINE $refine_B(\mathcal{H})$
if $\Psi^{\mathcal{H}}$ is valid **then**
  $\mathcal{H}' \leftarrow \mathcal{H}$
else
  $\sigma_w \leftarrow getWitness(\mathsf{sat}(\neg\Psi^{\mathcal{H}}))$
  $\mathcal{H}' \leftarrow \emptyset$
  for all $H \in \mathcal{H}$ **do**
    $\mathcal{R} \leftarrow \{(x_i, x_j) : x_i, x_j \in H$ and $[\![f_i]\!]^p_c(\sigma_w) = [\![f_j]\!]^p_c(\sigma_w)\}$
    $\mathcal{H}' \leftarrow \mathcal{H}' \cup (H/\mathcal{R})$
  end for
end if
return $\mathcal{H}'$

FDE Partition Refinement. Routine $refine_F$, shown in Algorithm 2, refines each block of the *current* partition of IDOL variables according to FDE. As discussed, we use the binary characterization of FDE in Theorem 2. Specifically, for each block $H \in \mathcal{H}$, routine $refine_F$ computes an equivalence relation $\mathcal{R}$ on H relating variables x_i, x_j of H respecting $\Phi^{\mathcal{H}}_{x_i,x_j}$, and adds to $\mathcal{H}'$ blocks of $\mathcal{R}$-equivalent variables of H. (As discussed, the SMT solver is used when computing $\mathcal{R}$ to check the validity of $\Phi^{\mathcal{H}}_{x_i,x_j}$ for each pair of variables $x_i, x_j \in H$.) The algorithm is correct, since $\mathcal{H}'$ is a refinement of $\mathcal{H}$, and two variables x_i, x_j for which $\Phi^{\mathcal{H}}_{x_i,x_j}$ is not valid cannot belong to the same block of an FDE partition.

Theorem 5. *If p is an IDOL program, $\mathcal{G}$ a partition of $\mathcal{V}_p$ and $\chi = F$, Algorithm 1 returns the coarsest FDE partition refining $\mathcal{G}$.*

Proof. See Section 3.3. □

BDE Partition Refinement. Routine $refine_B$, shown in Algorithm 3, refines the given current partition for computing BDE. Differently from the FDE case, Definition 6 can be directly used for this. Furthermore we fully exploit the SMT technology: If $\Psi^{\mathcal{H}}$ is not valid then $\neg\Psi^{\mathcal{H}}$ is satisfiable, hence by invoking $\mathsf{sat}(\neg\Psi^{\mathcal{H}})$ the SMT solver provides us with a witnessing assignment σ_w for which $\neg\Psi^{\mathcal{H}}$ holds. We use such witness as a "counter-example" to refine $\mathcal{H}$: Each block $H \in \mathcal{H}$ is split in sub-blocks of variables $x_{H,i}$ whose drifts $f_{H,i}$ have same value if evaluated according to σ_w. The algorithm is correct because the obtained partition $\mathcal{H}'$ is a refinement of $\mathcal{H}$, and two variables whose drifts have different values for σ_w cannot belong to the same block of a BDE partition.

Theorem 6. *If p is an IDOL program, $\mathcal{G}$ a partition of $\mathcal{V}_p$ and $\chi = B$, Algorithm 1 returns the coarsest BDE partition refining $\mathcal{G}$.*

Proof. See Section 3.3. □

As with other SMT-based partition refinement algorithms [42, 9], Algorithm 1 as a whole can be implemented in a standard way as a routine in a general purpose programming language which calls the SMT solver when required.

3.3. Proofs of Theorem 5 and Theorem 6

The following auxiliary notions, which will be needed later, have been introduced in the proof of Theorem 4 and are repeated for convenience.

Definition 9. *Let $\mathcal{H}$ denote a partition of $\mathcal{V}_p$.*

- *Write $x_i \sim_{\mathcal{H}} x_j$ whenever there exists some $H \in \mathcal{H}$ such that $x_i, x_j \in H$.*
- *Set $\sim_{\Phi}^{\mathcal{H}} := \{(x_i, x_j) : x_i = x_j \text{ or } \Phi_{x_i,x_j}^{\mathcal{H}} \text{ is valid}\}$.*
- *Using $\Psi_{x_i,x_j}^{\mathcal{H}} := \Theta(p) \rightarrow (\bigwedge_{H\in\mathcal{H}}(x_{H,1} = \ldots = x_{H,|H|}) \rightarrow f_i = f_j)$, define $\sim_{\Psi}^{\mathcal{H}} := \{(x_i, x_j) : \Psi_{x_i,x_j}^{\mathcal{H}} \text{ is valid}\}$.*

It can be easily seen that $\sim_{\Phi}^{\mathcal{H}}$ and $\sim_{\Psi}^{\mathcal{H}}$ are equivalence relations on $\mathcal{V}_p$.

We will need the following auxiliary statements in order to show the correctness of our partition refinement algorithms.

Lemma 1. *Let $\mathcal{H}$ be a partition of $\mathcal{V}_p$. Then, $\mathcal{H}$ is an FDE if and only if $\mathcal{H} = \mathcal{V}_p/(\sim_{\Phi}^{\mathcal{H}} \cap \sim_{\mathcal{H}})$.*

Proof. If $\mathcal{H}$ is an FDE, then $\mathcal{H}$ is a refinement of $\mathcal{V}_p/\sim_{\Phi}^{\mathcal{H}}$. Consequently, it holds that $\mathcal{H} = \mathcal{V}_p/(\sim_{\Phi}^{\mathcal{H}} \cap \sim_{\mathcal{H}})$. Let us now assume that $\mathcal{H} = \mathcal{V}_p/(\sim_{\Phi}^{\mathcal{H}} \cap \sim_{\mathcal{H}})$. Then, it holds that $\mathcal{H}$ is a refinement of $\mathcal{V}_p/\sim_{\Phi}^{\mathcal{H}}$. This, however, implies that $\mathcal{H}$ is an FDE. □

Lemma 2. *Let $\mathcal{H}_1, \mathcal{H}_2$ be two partitions of $\mathcal{V}_p$. Then, the following can be shown.*

- *$x_i \sim_{\Phi}^{\mathcal{H}_1} x_j$ implies $x_i \sim_{\Phi}^{\mathcal{H}_2} x_j$ if $\mathcal{H}_1$ is a refinement of $\mathcal{H}_2$.*
- *$x_i \sim_{\Psi}^{\mathcal{H}_1} x_j$ implies $x_i \sim_{\Psi}^{\mathcal{H}_2} x_j$ if $\mathcal{H}_1$ is a refinement of $\mathcal{H}_2$.*

Proof. Since any block from $\mathcal{H}_2$ can be written as a unique union of blocks from $\mathcal{H}_1$ and the aggregated drifts of each block from $\mathcal{H}_1$ are invariant to the the term rewriting $[x_i/s(x_i + x_j), x_j/(1 - s)(x_i + x_j)]$, the claim holds true. The second claim is trivial. □

Proof of Theorem 5. Assume that $\mathcal{G}'$ denotes the coarsest FDE partition that refines $\mathcal{H}_0 := \mathcal{G}$ and set $\mathcal{H}_{k+1} := \mathcal{V}_p/(\sim_{\Phi}^{\mathcal{H}_k} \cap \sim_{\mathcal{H}_k})$ for all $k \geq 0$. Then, the sequence $(\mathcal{H}_k)_{k\geq 0}$ is such that

- $\mathcal{G}'$ is a refinement of $\mathcal{H}_k$;
- $\mathcal{H}_k$ is a refinement of $\mathcal{H}_{k-1}$;

for all $k \geq 1$. We prove this by induction on k.

- $k = 1$: Since $\mathcal{G}'$ is a refinement of $\mathcal{H}_0$, Lemma 2 ensures the first claim. The second claim is trivial.
- $k \rightarrow k + 1$: Thanks to the fact that $\mathcal{G}'$ is a refinement of $\mathcal{H}_k$ by induction, Lemma 2 ensures the first claim. The second claim is trivial.

Since $\mathcal{G}'$ is a refinement of any $\mathcal{H}_k$, it holds that $\mathcal{G}' = \mathcal{H}_k$ whenever $\mathcal{H}_k$ is an FDE partition. Thanks to the fact that $\mathcal{H}_k$ is a refinement of $\mathcal{H}_{k-1}$ for all $k \geq 1$ and $\mathcal{V}_p$ is finite, we can fix the smallest $k \geq 1$ such that $\mathcal{H}_k = \mathcal{H}_{k-1}$. Since this implies $\mathcal{H}_{k-1} = \mathcal{H}_k = \mathcal{V}_p/(\sim_{\Phi}^{\mathcal{H}_{k-1}} \cap \sim_{\mathcal{H}_{k-1}})$, Lemma 1 ensures that $\mathcal{H}_{k-1}$ is an FDE.

To see that this implies the correctness of the FDE partition refinement algorithm, note that, for a given $\mathcal{H}$, Algorithm 2 calculates $\mathcal{H}' \leftarrow \mathcal{V}_p/(\sim_{\Phi}^{\mathcal{H}} \cap \sim_{\mathcal{H}})$. □

The next auxiliary result will be needed in the proof of Theorem 6.

Lemma 3. *Let us assume that $\mathcal{G}'$ denotes the coarsest BDE partition that refines $\mathcal{G}$ and let $\mathcal{H}$ be such that $\mathcal{G}'$ is a refinement of $\mathcal{H}$. Moreover, let us assume that $\sigma \in E(p)$ is such that*

- *$\sigma(x_i) = \sigma(x_j)$ for all $H \in \mathcal{H}$ and $x_i, x_j \in H$;*
- *$[\![f_i]\!]^p_c(\sigma) \neq [\![f_j]\!]^p_c(\sigma)$ for some $H \in \mathcal{H}$ and $x_i, x_j \in H$.*

Then, $\sim_\sigma := \{(x_i, x_j) : [\![f_i]\!]^p_c(\sigma) = [\![f_j]\!]^p_c(\sigma)\}$ is an equivalence relation on $\mathcal{V}_p$ and $\mathcal{G}'$ is a refinement of $\mathcal{V}_p/(\sim_\sigma \cap \sim_{\mathcal{H}})$. Crucially, it holds that $\mathcal{V}_p/(\sim_\sigma \cap \sim_{\mathcal{H}})$ is a proper refinement of $\mathcal{H}$, meaning that $\mathcal{H} \neq \mathcal{V}_p/(\sim_\sigma \cap \sim_{\mathcal{H}})$.

Proof. It can be easily seen that $\sim_\sigma$ is an equivalence relation on $\mathcal{V}_p$. Thanks to Lemma 2, we know that $x_i \sim^{\mathcal{G}'}_\Psi x_j$ implies $x_i \sim^{\mathcal{H}}_\Psi x_j$ for any $x_i, x_j \in \mathcal{V}_p$. Thus, $x_i \sim^{\mathcal{G}'}_\Psi x_j$ implies $x_i \sim_\sigma x_j$ for any $x_i, x_j \in \mathcal{V}_p$. This shows that $\mathcal{G}'$ is a refinement of $\mathcal{V}_p/(\sim_\sigma \cap \sim_{\mathcal{H}})$. At last, we note that the properties of σ induce that $\mathcal{V}_p/(\sim_\sigma \cap \sim_{\mathcal{H}})$ is a proper refinement of $\mathcal{H}$. □

Proof of Theorem 6. Let us assume that $\mathcal{G}'$ denotes the coarsest BDE partition that refines $\mathcal{G}$. Since Algorithm 3 provides us with a refinement $\mathcal{H}'$ of a given $\mathcal{H}$, the BDE partition refinement calculates a sequence $(\mathcal{H}_0, \ldots, \mathcal{H}_n)$ such that $\mathcal{H}_0 = \mathcal{G}$, $\mathcal{H}_{n-1} = \mathcal{H}_n$ and $\mathcal{H}_{i+1}$ is a refinement of $\mathcal{H}_i$ for all $0 \leq i \leq n-1$. Thanks to Lemma 3, we know that $\mathcal{G}'$ is a refinement of $\mathcal{H}_i$ for all $0 \leq i \leq n$. Since the BDE partition refinement algorithm stops if it finds a BDE partition, this implies that $\mathcal{H}_{n-1} = \mathcal{G}'$. □

4. Applications

In this section we relate IDOL to CTMCs, CRNs, and the Fluid process algebra (FPA) process algebra [94, 93]. We show that differential equivalences include the already available notions of equivalence developed in those domains. To do this in a self-contained manner we present the definitions of the semantics as well as of the original equivalences, while we refer to the literature for the intuitions and motivations behind the languages themselves. In all cases, the encoding of the original semantics into IDOL is straightforward, hence we omit this formal step and directly give the underlying IDOL program.

4.1. Continuous-time Markov Chains

Let us consider a CTMC with states $\{1, \ldots, n\}$ that is given in terms of its generator matrix $Q = (q_{i,j})_{1 \leq i,j \leq n}$ where $q_{i,j} \in \mathbb{Q}$. That is, for $i \neq j$, the entry $q_{i,j} \geq 0$ defines the rate at which the CTMC moves from state i into state j, whereas we set $q_{i,i} = -\sum_{i \neq j} q_{i,j}$ for all $1 \leq i \leq n$. Then, the corresponding IDOL program is given by the Kolmogorov forward equations.

Definition 10. *The IDOL program p_Q of a CTMC $(q_{i,j})_{1 \leq i,j \leq n}$ is*

$$\dot{x}_i = -\sum_{j \neq i} q_{i,j} \cdot x_i + \sum_{j \neq i} q_{j,i} \cdot x_j, \quad \text{for all } 1 \leq i \leq n.$$

Meaningful contexts for p_Q are such that the initial condition $\hat{\sigma}$ is a probability distribution, i.e., $\sum_{1 \leq i \leq n} \hat{\sigma}(x_i) = 1$, with $\hat{\sigma}(x_i) \geq 0$ for all $1 \leq i \leq n$. For such a context c, $[\![x_i]\!]^p_c(t)$ gives the probability of being in state i at time t.

We next provide the notions of lumpability for CTMCs [14].

Definition 11 (Ordinary and Exact Lumpability)**.** *Let $\mathcal{Z}$ be a partition of $\{1, \ldots, n\}$ and set*

$$q[i, Z] := \sum_{j \in Z} q_{i,j} \text{ and } q[Z, i] := \sum_{j \in Z} q_{j,i},$$

where $1 \leq i \leq n$ and $Z \subseteq \{1, \ldots, n\}$.

- *$\mathcal{Z}$ is called* ordinarily lumpable *if $q[i, Z'] = q[i', Z']$ for all $Z, Z' \in \mathcal{Z}$ and $i, i' \in Z$.*
- *$\mathcal{Z}$ is called* exactly lumpable *if $q[Z', i] = q[Z', i']$ for all $Z, Z' \in \mathcal{Z}$ and $i, i' \in Z$.*

This definition motivates our terminology. Ordinary lumpability is a "forward" criterion because it relates states according to their outgoing transitions (toward equivalence classes); exact lumpability is a "backward" criterion since it relates states according to incoming transitions (from predecessor equivalence classes). On the domain of CTMCs, FDE and BDE turn out to be equivalent to ordinary lumpability and exact lumpability, respectively.

Theorem 7. *Fix a CTMC $Q = (q_{i,j})_{1\leq i,j\leq n}$ and let $\mathcal{Z}$ be a partition of $\{1,\ldots,n\}$. Then, $\mathcal{Z}$ is ordinarily lumpable (resp., exactly lumpable) if and only if the partition $\mathcal{H}_{\mathcal{Z}} = \{\{x_i : i \in Z\} : Z \in \mathcal{Z}\}$ of $\mathcal{V}_{p_Q}$ is an FDE (resp., BDE) of p_Q.*

Proof. **FDE case.** As observed already by Proposition 1 in [95], for all $Z \in \mathcal{Z}$ and $i \in Z$ it holds that

$$q[i,Z] = q(i,i) + \sum_{\substack{j\in Z\\ j\neq i}} q(i,j) = -\sum_{\substack{Z'\in\mathcal{Z}\\ Z'\neq Z}} q[i,Z'].$$

That is, $\mathcal{Z}$ is ordinarily lumpable if and only if $q[i,Z'] = q[i',Z']$ for all $Z, Z' \in \mathcal{Z}$ and $i, i' \in Z$ where $Z' \neq Z$. In the following, we will use this alternative formulation of ordinary lumpability to establish the equivalence with FDE. Since [14] and the proof of Theorem 2 ensure that any ordinarily lumpable partition $\mathcal{Z}$ induces an FDE $\mathcal{H}_{\mathcal{Z}}$, let us assume that $\mathcal{H}_{\mathcal{Z}}$ is an FDE and pick some arbitrary $Z, Z' \in \mathcal{Z}$ and $i, i' \in Z$ with $Z' \neq Z$. We have to show that $q[i,Z'] = q[i',Z']$. Thanks to Theorem 2, we know that $\Phi^{\mathcal{H}_{\mathcal{Z}}}_{x_i,x_{i'}}$ holds true. This, however, implies that the value of

$$\sum_{j\in Z'} q_{i,j}\cdot s\cdot(x_i+x_{i'}) + \sum_{j\in Z'} q_{i',j}\cdot(1-s)\cdot(x_i+x_{i'})$$

does not depend on the assignment $0 < \sigma(s) \leq 1$, meaning that $q[i,Z'] = q[i',Z']$.

BDE case. Assume without loss of generality that $\mathcal{Z} = \{\overline{1},\ldots,\overline{M}\}$ with $M = |\mathcal{Z}|$ and $\overline{I} = \{(\overline{I},1),\ldots,(\overline{I},|\overline{I}|)\}$ for any $\overline{I} \in \mathcal{Z}$. We note that

$$\dot{x}_k = -\sum_{j\neq k} q_{k,j}\cdot x_k + \sum_{j\neq k} q_{j,k}\cdot x_j = \sum_j q_{j,k}\cdot x_j,$$

which yields, for any $\overline{I} \in \mathcal{Z}$ and $1 \leq l \leq |\overline{I}|$,

$$\dot{x}_{\overline{I},l} = \sum_{\overline{J}\in\mathcal{Z}}\sum_{k=1}^{|\overline{J}|} q_{(\overline{J},k),(\overline{I},l)}\cdot x_{\overline{J},k} = \sum_{\overline{J}\in\mathcal{Z}}\Big(\sum_{k=1}^{|\overline{J}|} q_{(\overline{J},k),(\overline{I},l)}\Big)\cdot x_{\overline{J},1} =: \wp_{\overline{I},l}$$

in the case of $\bigwedge_{\overline{J}\in\mathcal{Z}}(x_{\overline{J},1} = \ldots = x_{\overline{J},|\overline{J}|})$. Since $\mathcal{H}_{\mathcal{Z}}$ is a BDE and the real polynomials $\wp_{\overline{I},l}$ and $\wp_{\overline{I},l'}$, where $\overline{I} \in \mathcal{Z}$ and $1 \leq l, l' \leq |\overline{I}|$, coincide if and only if they have the same coefficients, we infer that

$$\sum_{k=1}^{|\overline{J}|} q_{(\overline{J},k),(\overline{I},l)} = \sum_{k=1}^{|\overline{J}|} q_{(\overline{J},k),(\overline{I},l')}$$

for all $\overline{J} \in \mathcal{Z}$, thus closing the proof. □

This result explains why differential equivalences can be seen as a somewhat natural generalization of more traditional notions of equivalence for discrete-state stochastic systems. In principle, the coarsest lumpable partition of a CTMC could be computed using the partition refinement algorithm in Section 3.2. However, in practice, one would use the efficient algorithms specialized for CTMCs, which run in polynomial time and space [43]. Still, an SMT-based approach to computing CTMC lumpability can be useful to handle uncertainty in rate values, by treating them symbolically (e.g., as suggested in [42] for PRISM [68]).

As a side product, we remark that Theorem 7 provides a characterization of ordinary and exact lumpability by means of real calculus, based on the Kolmogorov ODEs, instead of the classical argument [14] that combines the well-known concept of *uniformization* (e.g., [86]) with the characterization of lumpability for discrete time Markov chains [65].

4.2. Chemical Reaction Networks

A CRN is a set of rules (reactions) describing interactions between *species*. For instance, the reaction $A + B \xrightarrow{\alpha} 2C$ states that one element (e.g., molecule) of species A interacts with one element of species B to form two elements of species C. The label α decorates the reaction with information about the speed at which the reaction occurs; its signature depends on the chosen kinetics.

Formally, let S be a finite set of species. Either side of a reaction is a multiset of S, i.e., a function in $\mathbb{N}_0^S$ associating each species with its multiplicity (the *stoichiometry*) as a reactant or product. The stoichiometry of a species A in multiset ρ is denoted by ρ_A. A *reaction r over S* is a triple $(\rho, \pi, \alpha) \in R_S \subseteq \mathbb{N}_0^S \times \mathbb{N}_0^S \times \mathcal{L}$, where $\mathcal{L}$ is the label set, represented usually with $\rho \xrightarrow{\alpha} \pi$.

Mass-action CRNs. We now give the semantics of CRNs according to standard mass action kinetics. In this case the labels are *rates*, i.e., positive real numbers; the speed of the reaction is proportional by such rates to the product of the amounts of the reactant species.

Definition 12. *A mass-action CRN is a pair (S, R_S) where R_S is a finite set of reactions over S, with $R_S \subseteq \mathbb{N}_0^S \times \mathbb{N}_0^S \times \mathbb{Q}_{>0}$.*

For mass-action reactions, set $\phi(A, \rho \xrightarrow{\alpha} \pi) := \alpha(\pi_A - \rho_A)$.

Definition 13. *The IDOL program p_S of a mass-action CRN (S, R) is*

$$\dot{x}_A = f_A := \sum_{\rho \xrightarrow{\alpha} \pi \in R_S} \phi(A, \rho \xrightarrow{\alpha} \pi) \prod_{B \in S} x_B^{\rho_B}, \quad \text{for all } A \in S.$$

Hill CRNs. We discuss the semantics of CRNs according to the Hill kinetics (e.g., [97]) in the case of *catalytic* reactions, i.e., reactions which are in the form $B + C \xrightarrow{l} D + C$ with $B \neq D$. Here, C plays the role of a catalyst, a species promoting the reaction but which is not affected by it. Species B is the *substrate* that is modified, becoming D, when the reaction occurs. Each reaction is labeled with a triple $(\beta_1, \beta_2, \nu) \in \mathbb{Q}_{>0}^3$.

Definition 14. *A Hill CRN is a pair (S, R_S) where R_S is a finite set of catalytic reactions with $R_S \subseteq \mathbb{N}_0^S \times \mathbb{N}_0^S \times \mathbb{Q}_{>0}^3$.*

Definition 15. *The IDOL program p_S of a Hill CRN is*

$$\dot{x}_A = h_A := \sum_{\substack{(\rho \xrightarrow{(\beta_1, \beta_2, \nu)} \pi) \in R_S \\ \rho = B + C, \pi = D + C}} (\pi_A - \rho_A) \frac{\beta_1 x_B^\nu}{\beta_2 + x_B^\nu}, \quad \text{for all } A \in S.$$

In both semantics, reasonable contexts for CRNs are such that $\hat{\sigma}(x_A) \geq 0$ for all $A \in S$, since the IDOL variables represent *concentrations* of species, i.e., molecular counts divided by the volume of the environment where the reactions take place.

4.2.1. CRN Emulation

Emulation is a recently developed notion of comparison between mass-action CRNs [21]. The definition is presented below, slightly simplified from [21] and directly stated in IDOL terms.

Definition 16. *Let (S, R_S) and $(\tilde{S}, \tilde{R}_{\tilde{S}})$ denote two mass-action CRNs, with vector fields denoted by $[\![f]\!]_c^{p_S}$ and $[\![\tilde{f}]\!]_{\tilde{c}}^{p_{\tilde{S}}}$ and contexts denoted by c and $\tilde{c}$, respectively. A* species morphism *from (S, R_S) to $(\tilde{S}, \tilde{R}_{\tilde{S}})$ is a function $\mu_S : S \to \tilde{S}$. It is an* emulation *when $[\![\mathbf{f}]\!]_c^{p_S}(\tilde{\sigma} \circ \mu_S) = ([\![\tilde{\mathbf{f}}]\!]_{\tilde{c}}^{p_{\tilde{S}}}(\tilde{\sigma})) \circ \mu_S$ for all $\tilde{\sigma} \in \mathbb{R}^{\tilde{S}}$.*

The emulation condition, stated in terms of function composition, can be checked syntactically on the CRN structure by using the notions of reactant morphism and stoichiomorphism presented in [21]. Here we formally relate emulation to BDE.

Proposition 2. *If μ_S is an emulation from (S, R_S) to $(\tilde{S}, \tilde{R}_{\tilde{S}})$ then:*

i) $\{\mu_S^{-1}(\tilde{A}) : \tilde{A} \in \tilde{S}\}$ is a BDE partition of (S, R_S).

ii) Assume $S \cap \tilde{S} = \emptyset$*. Then,* $\{\mu_S^{-1}(\tilde{A}) \cup \{\tilde{A}\} : \tilde{A} \in \tilde{S}\}$ *is a BDE partition of the CRN* $(S \cup \tilde{S}, R_S \cup \tilde{R}_{\tilde{S}})$.

We observe that with i) BDE allows to relate species within the same CRN. By ii), we note that emulation relates species essentially like BDE: whenever all species are initialized with the same conditions as the target species to which they are mapped, then such species have the same ODE traces. We note that the assumption on disjoint sets of species in ii) is without loss of generality since it is always possible to rename species of one CRN with fresh names.

Example 1. The following two mass-action CRNs describe the behavior of AM, a basic biological switch (left) and MI, a mutual inhibition mechanism (right) [21]:

$$\begin{array}{l|l|l} X_0 + X_2 \xrightarrow{\alpha_1} X_2 + X_1 & Y_0 + Z_0 \xrightarrow{\alpha_1} Z_0 + Y_1 & Z_2 + Z_0 \xrightarrow{\alpha_1} Z_0 + Z_1 \\ X_1 + X_2 \xrightarrow{\alpha_2} X_2 + X_2 & Y_1 + Z_0 \xrightarrow{\alpha_2} Z_0 + Y_2 & Z_1 + Z_0 \xrightarrow{\alpha_2} Z_0 + Z_0 \\ X_2 + X_0 \xrightarrow{\alpha_3} X_0 + X_1 & Y_2 + Y_0 \xrightarrow{\alpha_3} Y_0 + Y_1 & Z_0 + Y_0 \xrightarrow{\alpha_3} Y_0 + Z_1 \\ X_1 + X_0 \xrightarrow{\alpha_4} X_0 + X_0 & Y_1 + Y_0 \xrightarrow{\alpha_4} Y_0 + Y_0 & Z_1 + Y_0 \xrightarrow{\alpha_4} Y_0 + Z_2 \end{array}$$

The following species morphism can be shown to be an emulation:

$$\mu_S(Y_0) = X_0,\ \mu_S(Y_1) = X_1,\ \mu_S(Y_2) = X_2, \mu_S(Z_0) = X_2,\ \mu_S(Z_1) = x_1,\ \mu_S(Z_2) = X_0$$

Since emulation is a particular BDE partition, with BDE it is possible to automatically check whether such correspondence of traces carries over to non-mass-action kinetics. In [26, 29, 88] it is shown how BDE can be used within an algorithm for computing all [26, 29] or a subset [88] of emulations between two mass-action CRNs.

The possibility of reasoning using different hypotheses for the reaction kinetics is of biological relevance because in different situations one may find mass-action mechanisms (e.g., phosphotransfers) or Hill-type mechanisms (e.g., enzymes). For instance, much of the utility of Hill kinetics is owed to supporting non-integer exponents. Famously, this ranges in 2.3-3.0 for haemoglobin. Furthermore, biologists often consider exponents less than 1 in order to describe "anticooperative" behavior [74]. Any rational exponent can be expressed in IDOL.

In Example 1 it is possible to show that replacing equal mass-action rates with equal (and arbitrary) Hill triples, a BDE partition that is related to an emulation in the sense of item ii) of Proposition 2 is still BDE for the resulting Hill CRN. This suggests a structural relationship between CRNs with different biological functionality, which is insensitive (to some extent) to underlying kinetics that is considered. Indeed, through BDE it is possible to show that *all* of the thirteen species morphisms found in [21] do enjoy this property. This is particularly interesting because, at the same time, Hill kinetics can never be exactly matched by mass-action kinetics, and vice versa. (This holds because the drift for Hill kinetics has partial derivatives of arbitrary high order that are not identical to zero, whereas the drift for mass-action kinetics does not.)

4.2.2. Bisimulations and Equivalences for CRNs

Forward and backward bisimulations for CRNs have been originally introduced in [23], together with polynomial time and space algorithms for computing the the largest bisimulations. These are equivalence relations over species for *elementary* mass-action reactions, where at most two reactants (possibly of the same species) can appear in the left hand side. A more efficient, but still polynomial in time and space, partition-refinement algorithm has been provided in [25] to compute the largest bisimulations.

These notions have been recently generalized in [27] to forward and backward equivalence for reaction networks, given next.

Definition 17. *A (mass-action) reaction network (RN) is a pair* (S, R_S) *where* R_S *is a finite set of reactions over* S*, with* $R_S \subseteq \mathbb{N}_0^S \times \mathbb{N}_0^S \times \mathbb{Q}$*. The* IDOL *program* p_S *of an RN* (S, R) *is given by*

$$\dot{x}_A = f_A := \sum_{\rho \xrightarrow{\alpha} \pi \in R_S} \phi(A, \rho \xrightarrow{\alpha} \pi) \prod_{B \in S} x_B^{\rho_B}, \quad \textit{for all } A \in S.$$

Table 1: FDE coincides with forward equivalence (FE) for polynomial ODEs, with higher computational cost

Original model			*Reduction*		
Model	$\|R\|$	$\|S\|$	*FE (s)*	*FDE (s)*	*Size*
M1 [44, 84]	8620	745	2.86E–1	7.85E+3	105
M2 [44, 84]	3680	354	7.70E–2	3.22E+3	105
M3 [1]	4944	411	6.61E–2	6.46E+2	47
M4 [8]	3447	348	4.00E–2	5.22E+3	215

RNs extend elementary CRNs by allowing for negative rates and an arbitrary multiset of reactants, as opposed to strictly positive rates and elementary reactions. Notably, this allows for any polynomial ODE system to be encoded into an RN [27]. The largest forward and backward equivalences of an RN can be computed by an algorithm which extends the one from [25], whose time and space complexity is polynomial in the number of ODE variables and monomials [27]. At the same time, forward and backward equivalence characterize FDE and BDE, respectively, provided that FDE and BDE are restricted to polynomial ODE systems.

In this section we compare the performance and generality of forward and backward equivalence with FDE and BDE, showing that:

- Forward and backward equivalence should be used when dealing with polynomial ODE systems. This is due to performance reasons;
- In general, FDE and BDE should be preferred when dealing with non-polynomial ODE systems. Indeed, despite certain non-polynomial ODE systems can be transformed into polynomial ones using transformation techniques like [53], this *polynomization* might break equivalences present in the original system.

Forward equivalence (FE). Forward equivalence depends on the computation of the following quantities from the RN syntax.

Definition 18. *Let (S, R_S) be an RN, $A \in S$, $Z \subseteq S$, and ρ be a multiset of species in S. The* net instantaneous rate of *A, the* cumulative net instantaneous rate of *Z, and the* ρ-forward rate *of A, are defined respectively as*

$$\phi(\rho, A) := \sum_{(\rho \xrightarrow{\alpha} \pi) \in R_S} (\pi_A - \rho_A) \cdot \alpha, \qquad \phi(\rho, Z) := \sum_{A' \in Z} \phi(\rho, A'), \qquad \mathbf{fr}(A, \rho, Z) := \frac{\phi(A + \rho, Z)}{[A + \rho]!}$$

Where the operator $[\cdot]!$ denotes the multinomial coefficient induced by a multiset of species, which for $S = \{A_1, A_2, ..., A_n\}$ is defined as:

$$[\rho]! := \binom{\sum_{A \in S} \rho_A}{\rho_{A_1}, \rho_{A_2}, \ldots, \rho_{A_n}}.$$

Definition 19. *Let (S, R_S) be an RN, $\mathcal{R}$ an equivalence relation over S and $\mathcal{Z} = S/\mathcal{R}$. Then, $\mathcal{R}$ is a* forward equivalence *if for all $(A, A') \in \mathcal{R}$, all multisets of species ρ, and all blocks $Z \in \mathcal{Z}$ it holds that*

$$\mathbf{fr}[A, \rho, Z] = \mathbf{fr}[A', \rho, Z]. \tag{4}$$

Proposition 3 (Forward equivalence corresponds to FDE for polynomial ODEs)**.** *Let (S, R_S) be an RN and $\mathcal{R}$ be an equivalence relation over S. Then, the* IDOL *program p underlying (S, R_S) is such that $\mathcal{V}_p/\{(x_A, x_{A'}) : (A, A') \in \mathcal{R}\}$ is an FDE partition if and only if $\mathcal{R}$ is a forward equivalence.*

We now show that FDE and FE coincide for biochemical models from the literature modelled as mass-action CRNs, but with computation times separated by several orders of magnitude. For this experimental study, we used our tool ERODE [28], which provides full tool support for: (i) IDOL minimization up to FDE and BDE, and (ii) RNs minimization up to forward and backward equivalence. Our tool is available at `http://sysma.imtlucca.it/tools/erode`,

Table 2: BDE coincides with backward equivalence (BE) for polynomial ODEs, with higher computational cost

Original model			*Reduction*		
Model	$\|R\|$	$\|S\|$	*BE (s)*	*BDE (s)*	$\|S\|$
M5 [84]	786432	65538	5.59E+0	1.01E+3	167
M6 [84]	172032	16386	6.18E–1	3.01E+2	122
M7 [84]	48	18	5.00E–4	6.00E–2	12
M8 [87]	194054	14531	9.42E–1	3.45E+3	6634
M9 [44, 84]	187468	10734	5.39E–1	1.57E+3	5575
M10 [32, 33]	5832	730	1.60E–2	3.22E+0	217
M11 [66]	487	85	2.00E–3	2.71E–1	56
M12 [21]	24	18	5.00E–4	5.20E–2	3

while `http://sysma.imtlucca.it/tools/erode/comparison-syntactic-symbolic` provides information on how to replicate our tests.

The results are presented in Table 1. Alongside the model identifier we show the reference from which the CRN was taken; headers $|R|$ and $|S|$ give the number of reactions and species, respectively, of the original CRN. Headers *FE (s)* and *FDE (s)* give the time in seconds to compute the largest equivalences using the corresponding algorithms. Measurements were taken on a 2.6 GHz Intel Core i5 with 4 GB of RAM.

The runtime comparisons show that FDE is computationally more demanding than forward equivalence. The reason is that the computation of $\mathcal{R}$ at each iteration of Algorithm 2 requires in the worst case to establish the validity of $\Phi^{\mathcal{H}}_{x_i,x_j}$ for each pair of IDOL variables x_i, x_j belonging to the same block. Furthermore, each check is performed symbolically using the SMT solver, while the partition refinement algorithm of [23] splits candidate partitions using Definition 18. However we stress that these tests are "unfair" to our differential equivalences because the comparison is with a specialized partition refinement algorithm which iterates using concrete values (that can be computed syntactically).

Backward equivalence (BE). On the domain of RNs, i.e. of polynomial ODEs, BDE corresponds to backward equivalence. Given that backward equivalence is defined similarly to forward equivalence we avoid recalling it.

Proposition 4 (Backward equivalence corresponds to BDE for polynomial ODEs)**.** *Let (S, R_S) be an RN and $\mathcal{R}$ be an equivalence relation over S. Then, the IDOL program p underlying (S, R_S) is such that $\mathcal{V}_p/\{(x_A, x_{A'}) : (A, A') \in \mathcal{R}\}$ is a BDE partition if and only if $\mathcal{R}$ is a backward equivalence.*

In Table 2, we compare BDE and backward equivalence using models from [23]. As expected, the more efficient algorithm for computing the largest backward equivalence from [27] outperforms our SMT-based implementation. Nevertheless, it is worth noting that BDE was able to reduce rather large models. In the largest benchmark, M5, at each iteration of the partition refinement algorithm the SMT solver evaluated equivalences involving ca. 786,000 nonlinear monomials and 1,500,000 linear monomials, from binary and unary reactions, respectively. It is also interesting to note that the formula $\Psi^{\mathcal{H}}$ used at each iteration to refine the current partition $\mathcal{H}$ is much simpler than the ones considered for FDE, as just single drifts are compared rather than cumulative ones over blocks. For instance, it can be shown that in model M11 the coarsest FDE and BDE partitions coincide, but it took 9.90E+1 s to compute the largest FDE, as opposed to the reported 2.71E−1 s for the largest BDE.

On the reduction power of FDE/BDE and FE/BE for non-polynomial ODEs. We next argue that FDE and BDE are complementary to forward and backward equivalence from [27]. To this end, we show that in general the coarsest FDE or BDE underlying an IDOL program cannot be obtained by [27] even if one combines it with the technique from [53] which allows one to transform a (sufficiently smooth) non-polynomial ODE system into a polynomial one.

For the case of FDE, let us consider the program

$$\dot{x}_1 = -\frac{1}{x_1 + x_2} \qquad \dot{x}_2 = x_1 + x_2 \tag{5}$$

whose domain is the set of positive real numbers and which enjoys the FDE partition $\{\{x_1, x_2\}\}$ because $y := x_1 + x_2$ yields $\dot{y} = -1/y + y$. According to [53], the polynomial transformation of (5) is given by

$$\dot{x}_1 = -x_3 \qquad \dot{x}_2 = x_1 + x_2 \qquad \dot{x}_3 = x_3^3 - x_3^2 x_1 - x_3^2 x_2 \tag{6}$$

because the ODE of the auxiliary variable $x_3 := 1/(x_1 + x_2)$ is

$$\dot{x}_3 = -\frac{1}{(x_1 + x_2)^2}(\dot{x}_1 + \dot{x}_2) = -x_3^2(-x_3 + x_1 + x_2)$$

It can be shown that the solutions of (5) and (6) coincide as long as $\hat{\sigma}(x_3) = 1/(\hat{\sigma}(x_1) + \hat{\sigma}(x_2))$. However, while the coarsest FDE partition of (6) can be efficiently computed by the polynomial time algorithm from [27], it will be $\{\{x_1\}, \{x_2\}, \{x_3\}\}$ (and not $\{\{x_1, x_2\}, \{x_3\}\}$) because the original ODE of x_1 has been replaced by $-x_3$.

A similar statement can be made for BDE. Indeed, let us consider the program

$$\dot{x}_1 = \frac{x_1}{x_1 + x_2} + \frac{x_2}{x_1 + x_2} \qquad \dot{x}_2 = 1 \tag{7}$$

whose domain is again the set of positive real numbers. Since $x_1/(x_1 + x_2) + x_2/(x_1 + x_2)$ is equivalent to 1, it obviously holds true that $\{\{x_1, x_2\}\}$ is a BDE partition of (7). At the same time, the polynomial transformation of (7) is given by

$$\dot{x}_1 = x_1 x_3 + x_2 x_3 \qquad \dot{x}_2 = 1 \qquad \dot{x}_3 = -x_3^3 x_1 - x_3^3 x_2 - x_3^2 \tag{8}$$

because the auxiliary variable $x_3 := 1/(x_1 + x_2)$ yields

$$\dot{x}_3 = -\frac{1}{(x_1 + x_2)^2}(\dot{x}_1 + \dot{x}_2) = -x_3^2(x_1 x_3 + x_2 x_3 + 1)$$

Similarly to the foregoing polynomial transformation, the solution of (8) and (7) coincide when $\hat{\sigma}(x_3) = 1/(\hat{\sigma}(x_1) + \hat{\sigma}(x_2))$, while the coarsest BDE partition of (8) is $\{\{x_1\}, \{x_2\}, \{x_3\}\}$.

Notwithstanding the fact that the coarsest FDE/BDE partition of the polynomial transformation must be an FDE/BDE of the original model, the foregoing examples show that it does not have to be necessarily the coarsest FDE/BDE partition of the original model. Intuitively, this may happen whenever the model under study enjoys semantical but not syntactical symmetries because the introduction of auxiliary variables may break the symmetries present in the original model (e.g., in both examples, f_1 and f_2 enjoy semantical symmetry but not syntactical symmetry).

4.3. Process Algebra

Lastly we consider a fragment of Fluid Process Algebra (FPA) presented in [62], which corresponds to the process algebra studied in [92]. The grammar of FPA considers parallel composition of sequential processes with synchronization over shared actions. Let $\mathcal{A}$ denote the set of actions and $\mathcal{K}$ the set of constants. Each process $P \in \mathcal{K}$ is defined as $P \stackrel{def}{=} \sum_{i \in I_P} (\alpha_i, r_i).P_i$, where I_P is an index set, $\alpha_i \in \mathcal{A}$, $r_i \in \mathbb{Q}_{>0}$ is a rate, and $P_i \in \mathcal{K}$.

Using the obvious standard operational semantics for the choice and prefix operator, we let $\mathcal{B}(P)$ be the states of the underlying LTS, with transitions denoted by $P \xrightarrow{(\alpha_i, r_i)} P_i$. Furthermore, we let $\mathcal{A}(P)$ denote the set of actions labeling transitions from P.

Definition 20. *An FPA model $\mathcal{M}$ is generated by*

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

We let $\mathcal{G}(\mathcal{M})$ be the set of sequential components appearing in $\mathcal{M}$ and $\mathcal{B}(\mathcal{M})$ for $\bigcup_{P \in \mathcal{G}(\mathcal{M})} \mathcal{B}(P)$. For any two $P, Q \in \mathcal{G}(\mathcal{M})$, we assume $\mathcal{B}(P) \cap \mathcal{B}(Q) = \emptyset$.

We introduce the following elementary concepts that will be needed to define the semantics.

Definition 21. *Let $Z \subseteq \mathcal{K}$ and $\alpha \in \mathcal{A}$. Then*

$$r_\alpha(P) := \sum_{P \xrightarrow{(\alpha,r)} P'} r \quad \text{and} \quad q[P, Z, \alpha] := \sum_{P' \in Z} \sum_{P \xrightarrow{(\alpha,r)} P'} r$$

Also, we say that an action α is enabled *in an FPA model $\mathcal{M}$ if for any submodel $\mathcal{M}_1 \parallel_L \mathcal{M}_2$ of $\mathcal{M}$ with $\alpha \in L$ there exist $P_1 \in \mathcal{B}(\mathcal{M}_1)$ and $P_2 \in \mathcal{B}(\mathcal{M}_2)$ with $r_\alpha(P_1) > 0$ and $r_\alpha(P_2) > 0$.*

An IDOL variable (hence, an ODE) is associated with each LTS state of every sequential process appearing in an FPA model $\mathcal{M}$.

Definition 22. *The IDOL program of an FPA model $\mathcal{M}$ is given by*

$$\dot{x}_P = \sum_{\alpha \in \mathcal{A}} \sum_{P' \in \mathcal{B}(\mathcal{M})} x_{P'} \cdot q[P', \{P\}, \alpha] \cdot \mathcal{R}^*_\alpha(\mathcal{M}, P') - \sum_{\alpha \in \mathcal{A}} x_P \cdot r_\alpha(P) \cdot \mathcal{R}^*_\alpha(\mathcal{M}, P), \qquad \text{for all } P \in \mathcal{B}(\mathcal{M}),$$

where $r^*_\alpha(\mathcal{M})$ is recursively defined as

$$r^*_\alpha(\mathcal{M}_1 \parallel_L \mathcal{M}_2) := \begin{cases} r^*_\alpha(\mathcal{M}_1) + r^*_\alpha(\mathcal{M}_2) & , \alpha \notin L \\ \min\left(r^*_\alpha(\mathcal{M}_1), r^*_\alpha(\mathcal{M}_2)\right) & , \alpha \in L \end{cases}$$

$$r^*_\alpha(P) := \sum_{P' \in \mathcal{B}(P)} x_{P'} \cdot r_\alpha(P')$$

$$\mathcal{R}^*_\alpha(\mathcal{M}_1 \parallel_L \mathcal{M}_2, P) := \begin{cases} \mathcal{R}^*_\alpha(\mathcal{M}_i, P) & , P \in \mathcal{B}(\mathcal{M}_i) \wedge \alpha \notin L \\ \mathcal{R}^*_\alpha(\mathcal{M}_i, P) \frac{r^*_\alpha(\mathcal{M}_1 \parallel_L \mathcal{M}_2)}{r^*_\alpha(\mathcal{M}_i)} & , P \in \mathcal{B}(\mathcal{M}_i) \wedge \\ & \quad \alpha \in L \text{ is enabled in } \mathcal{M}_i \\ 0 & , \text{ otherwise} \end{cases}$$

$$\mathcal{R}^*_\alpha(P, P') := \begin{cases} 1 & , P' \in \mathcal{B}(P) \\ 0 & , \text{ otherwise} \end{cases}$$

We refer the reader to the literature (e.g., [92, 62]) for a more detailed discussion on the semantics of FPA. Here we stress that the crucial definition is $r^*_\alpha(\mathcal{M}_1 \parallel_L \mathcal{M}_2)$ for a synchronization action $\alpha \in L$. Intuitively, it provides a contribution to the drift that establishes a threshold-based contention between the capacities (i.e., rates) of the operands, dictated by the minimum function.

4.3.1. Differential Bisimulation

The first equivalence for FPA that we study is differential bisimulation [62]. It is a relation over the set of constants of an FPA model, defined in terms of conditions on the sequential behavior and on the compositional structure of processes. The latter is captured by collecting actions which affect the sequential behavior.

Definition 23. *Let $\mathcal{M}$ be an FPA model, and $P \in \mathcal{B}(\mathcal{M})$. Then*

$$\mathcal{D}(P, \mathcal{M}) := \begin{cases} L \cup \mathcal{D}(P, \mathcal{M}_i) & , \mathcal{M} = \mathcal{M}_1 \parallel_L \mathcal{M}_2, P \in \mathcal{B}(\mathcal{M}_i) \\ \emptyset & , \text{ otherwise} \end{cases} \qquad \mathcal{I}(P, \mathcal{M}) := \mathcal{D}(P, \mathcal{M}) \cap \mathcal{A}(P)$$

For any $Z \subseteq \mathcal{B}(\mathcal{M})$, we set $\mathcal{D}(Z, \mathcal{M}) = \bigcup_{P \in Z} \mathcal{D}(P, \mathcal{M})$ and $\mathcal{I}(Z, \mathcal{M}) = \bigcup_{P \in Z} \mathcal{I}(P, \mathcal{M})$.

Definition 24. *Let $\mathcal{M}$ be an FPA model, and $P, Q \in \mathcal{B}(\mathcal{M})$. Then we write $P \overset{s.i.}{=}_{\mathcal{M}} Q$ if*

(i) $\mathcal{A}(P) = \mathcal{A}(Q)$

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

Definition 25 (Differential bisimulation)**.** *Let $\mathcal{M}$ be an FPA model, $\mathcal{R}$ an equivalence relation over $\mathcal{B}(\mathcal{M})$, and $\mathcal{Z} = \mathcal{B}(\mathcal{M})/\mathcal{R}$. We say that $\mathcal{R}$ is a* differential bisimulation *for $\mathcal{M}$ if for all $(P, P') \in \mathcal{R}$ we have:*

(*i*) $q[P, Z, \alpha] = q[P', Z, \alpha]$*, for all $Z \in \mathcal{Z}$ and $\alpha \in \mathcal{A}$*

(*ii*) $P \overset{s.i.}{=}_{\mathcal{M}} P'$.

We observe that differential bisimulation can be checked in a non-symbolic fashion. This is because the *aggregate rates* $q[P, B, \alpha]$ only depend on the rates that label the prefix operator. The equivalence relation $\overset{s.i.}{=}_{\mathcal{M}}$ involves parsing the FPA model and appropriately collecting the action types that decorate the compositional operator of FPA. It turns out that differential bisimulation is a sufficient condition for FDE on the corresponding IDOL program.

Proposition 5. *Let $\mathcal{M}$ be an FPA model and $\mathcal{R}$ a differential bisimulation. Then, the IDOL program p underlying $\mathcal{M}$ is such that $\mathcal{V}_p/\{(x_P, x_Q) : (P, Q) \in \mathcal{R}\}$ is an FDE partition.*

We discuss why it is not a necessary condition using the example taken from [62].

Example 2. Let $\mathcal{M}_{\mathrm{F}} := P_1 \parallel_{\{\alpha\}} Q_1$, with P_1, Q_1 defined as

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

Applying Definition 22, its IDOL program is

$$\begin{aligned}
\dot{x}_{P_1} &= s\ \min(x_{P_2} + x_{P_3}, x_{Q_2}) - 2r\, x_{P_1} & \dot{x}_{Q_1} &= s\ \min(x_{P_2} + x_{P_3}, x_{Q_2}) - 2r\, x_{Q_1} \\
\dot{x}_{P_2} &= r\, x_{P_1} - s\, x_{P_2} \frac{\min(x_{P_2} + x_{P_3}, x_{Q_2})}{x_{P_2} + x_{P_3}} & \dot{x}_{Q_2} &= 2r\, x_{P_1} - s\ \min(x_{P_2} + x_{P_3}, x_{Q_2}) \\
\dot{x}_{P_3} &= r\, x_{P_1} - s\, x_{P_3} \frac{\min(x_{P_2} + x_{P_3}, x_{Q_2})}{x_{P_2} + x_{P_3}} &&
\end{aligned} \tag{9}$$

It can be shown that $\mathcal{Z}_{\mathrm{F}} = \{\{P_1\}, \{P_2, P_3\}, \{Q_1\}, \{Q_2\}\}$ is a differential bisimulation, hence the corresponding partition on the IDOL variables $\mathcal{H}_{\mathrm{F}} = \{\{x_{P_1}\}, \{x_{P_2}, x_{P_3}\}, \{x_{Q_1}\}, \{x_{Q_2}\}\}$ is an FDE. Indeed, summing the variables within each equivalence class, we obtain:

$$\begin{aligned}
\dot{x}_{P_1} &= s\ \min(x_{P_2} + x_{P_3}, x_{Q_2}) - 2r\, x_{P_1} & \dot{x}_{Q_1} &= s\ \min(x_{P_2} + x_{P_3}, x_{Q_2}) - 2r\, x_{Q_1} \\
\dot{x}_{P_2} + \dot{x}_{P_3} &= 2r\, x_{P_1} - s\ \min(x_{P_2} + x_{P_3}, x_{Q_2}) & \dot{x}_{Q_2} &= 2r\, x_{P_1} - s\ \min(x_{P_2} + x_{P_3}, x_{Q_2})
\end{aligned}$$

The converse, i.e., that an FDE over the IDOL variables implies a differential bisimulation for the corresponding processes, does not hold in general. For instance, by changing the definition of P_2 of Example 2 with $P_2 \overset{def}{=} (\delta, s).P_1$ we have that $\mathcal{R}_{\mathrm{F}}$ is not a differential bisimulation because we obtain $q[P_2, \{P_1\}, \alpha] = 0$ and $q[P_3, \{P_1\}, \alpha] = s$. Instead, $\mathcal{H}_{\mathrm{F}}$ remains an FDE, since the "domain-specific" information about action types is lost in the IDOL program.

As a further example, let us replace P_2 with the following definition: $P_2 \overset{def}{=} (\alpha, s).P_1 + (\delta, s).P_3$. Again, we have that $\mathcal{R}_{\mathrm{F}}$ is not a differential bisimulation, while $\mathcal{H}_{\mathrm{F}}$ is an FDE. This is because the added δ-transition from P_2 to P_3 distinguishes P_2 and P_3 (i.e., $q[P_2, \{P_2, P_3\}, \delta] = s$ and $q[P_3, \{P_2, P_3\}, \delta] = 0$), but its influence disappears in the lumped ODEs: the negative drift term $-\delta \cdot x_2$ in $\dot{x}_{P_2}$ cancels out the positive drift term $\delta \cdot x_2$ in $\dot{x}_{P_3}$. The above examples is an instance of the more general observation that transitions internal to an equivalence class do not interfere at the FDE level but may tell apart processes according to differential bisimulation. Indeed, it is not difficult to see that for any FPA model $\mathcal{M}$, the trivial partition corresponding to $\{\mathcal{B}(\mathcal{M})\}$ is always an FDE. This is, intuitively, a conservation-of-mass property due to the fact that processes are not created nor destroyed in FPA, hence all transitions are internal to the trivial partition. This remark also stresses the usefulness in having an algorithm that can refine any given initial partition, since computing the largest FDE for an FPA model always collapses to an uninteresting reduction.

4.3.2. Label Equivalence

Similarly to differential bisimulation, we next provide the notion of label equivalence that has been introduced in [92] and that describes a sufficient, but not necessary, condition for a partition of FPA constants to be a BDE.

Definition 26 (Label Equivalence)**.** *Let $\mathcal{M}$ be a FPA model and let $\mathcal{P} = (\mathbf{P}^1, \ldots, \mathbf{P}^N)$, $\mathbf{P}^i = (P^i_1, \ldots, P^i_{K_i})$, be a* tuple partition *on $\mathcal{G}(\mathcal{M}) = \{Q_1, \ldots, Q_n\}$, that is, for each $P \in \mathcal{G}(\mathcal{M})$ there exist unique $1 \leq i \leq N$ and $1 \leq k \leq K_i$ with $P = P^i_k$. $\mathbf{P}^i$ and $\mathbf{P}^j$ are said to be* label equivalent*, written $\mathbf{P}^i \sim_{\mathcal{P}} \mathbf{P}^j$, if $K_i = K_j$ and there exist bijections $\rho^{i,j}_k : \mathcal{B}(P^i_k) \to \mathcal{B}(P^j_k)$, where $1 \leq k \leq K_i$, such that for all $\alpha \in \mathcal{A}$ it holds that $r_\alpha(P^i_k) = r_\alpha(P^j_k)$ and*

- $\forall x_{Q_1} \ldots \forall x_{Q_n} \Big(\mathcal{R}^*_\alpha(\mathcal{M}, P) = \mathcal{R}^*_\alpha(\mathcal{M}, \rho_k(P))[\ldots] \Big)$,
- $\forall x_{Q_1} \ldots \forall x_{Q_n} \Big(\sum_{P' \in \mathcal{G}(\mathcal{M})} q[P', \{P\}, \alpha] \mathcal{R}^*_\alpha(\mathcal{M}, P') = \sum_{P' \in \mathcal{G}(\mathcal{M})} q[P', \{\rho_k(P)\}, \alpha] \mathcal{R}^*_\alpha(\mathcal{M}, P')[\ldots] \Big)$,
- $\forall x_{Q_1} \ldots \forall x_{Q_n} \Big(\mathcal{R}^*_\alpha(\mathcal{M}, P) = \mathcal{R}^*_\alpha(\mathcal{M}, P)[\ldots] \Big)$ *for all P in $ds(P^l_k)$ with $P^l_k \notin \mathbf{P}^i, \mathbf{P}^j$ and*
- $\forall x_{Q_1} \ldots \forall x_{Q_n} \Big(r^*_\alpha(\mathcal{M}) = r^*_\alpha(\mathcal{M})[\ldots] \Big)$,

where $[\ldots]$ abbreviates

$$[x_Q / x_{\rho_k(Q)}, x_R / x_{\rho_k^{-1}(R)} : Q \in \mathcal{B}(P^i_k), R \in \mathcal{B}(P^j_k), 1 \leq k \leq K].$$

It can be proven that label equivalence is an equivalence relation on the tuple partition $\mathcal{P}$. More importantly, the following result from [92] connects label equivalence to the notion of BDE.

Proposition 6. *Fix an FPA model $\mathcal{M}$, a tuple partition $\mathcal{P}$ of $\mathcal{G}(\mathcal{M})$ and let $\sim_{\mathcal{P}}$ be a label equivalence on $\mathcal{P}$. In particular, let $\rho^{i,j}_k : \mathcal{B}(P^i_k) \to \mathcal{B}(P^j_k)$ denote a set of bijections that relates any two label equivalent tuples $\mathbf{P}^i, \mathbf{P}^j \in \mathcal{P}$ and that satisfies $(\rho^{i,j}_k)^{-1} = \rho^{j,i}_k$. Then, $\{\{x_P : Q \in Z\} : Z \in \mathcal{B}(\mathcal{M}) / \approx\}$ is a BDE partition, where $Q \approx Q'$ whenever $Q' = \rho^{i,j}_k(Q)$ for some i, j and k.*

There exist BDE partitions that are not induced by label equivalence. To see this, consider the FPA model $\mathcal{M} = P_1$ with $P_1 = (\alpha, 1).P_2$, $P_2 = (\alpha, 2).P_3$, $P_3 = (\alpha, 1).P_4$ and $P_4 = (\alpha, 2).P_1$. Then, it can be easily verified that the partition $\{\{x_{P_1}, x_{P_3}\}, \{x_{P_2}, x_{P_4}\}\}$ of the underlying IDOL model is BDE. However, it cannot be constructed using label equivalence since this notion relates FPA constants of *distinct* elements of $\mathcal{G}(\mathcal{M})$.

5. Conclusion

We have provided a generic framework for reasoning about languages that have ordinary differential equations (ODEs) as their quantitative semantics. Three main principles can be borrowed from more traditional domains based on labeled transition systems or discrete-state stochastic processes such as Markov chains: program comparison and minimization are understood in terms of equivalence relations over the states of a program; partition-refinement algorithms can be used to compute the largest equivalences; and SMT can be used for program verification. Yet the technical details involved in this transplantation are somewhat intricate: in ODE semantics, the state space is implicitly given as a continuous function. Therefore, proving programs equivalent involves a universal quantification over an uncountable domain. We developed algorithms for our differential equivalences by exploiting the possibility of reasoning over the reals symbolically using SMT.

We have worked on a basic intermediate language for ODEs. Conceptually, it can be seen as the analogous of a "bytecode" format for higher-level languages, where differential equivalences are compiler-optimization techniques that transform the original program while exactly preserving its behavior. Reasoning at such an intermediate level leads to equivalences that are more general than analogous notions developed for higher-level languages, because no domain-specific elements and issues are involved (such as action types and compositionality in process algebra). This can lead to potentially coarser minimizations. However we argue that our contribution can still be useful when the modeler must work with higher-level equivalences to account for domain specificity. In this case, establishing a relationship with a differential equivalence may provide a way to automatizing checks. It appears already to be the case

for differential bisimulation [62] and label equivalence [92] for FPA, which can be encoded into SMT, being a forward and backward differential equivalence, respectively

There are interesting avenues of future research from this work. The most direct one is that of improving the performance and scalability of the presented algorithms. For example, validity checks of different formulae can be performed independently, thus allowing to refine more blocks at a time, or to parallelize the refinement of a single block. Also, it is natural to relax the assumption of exactness in favor of approximate equivalence relations, similarly to what has been done for models with stochastic semantics [80]. First steps in this direction have been recently conducted in [30]. In a similar vein, forward differential equivalence could be extended to capture weighted sums. However, since this would require a non-trivial extension of the partition refinement algorithm, we leave this for future work. At last, the computation of differential equivalences via SMT solver opens a number of possibilities for symbolic computation. Our intermediate language could be extended with parameter variables in order to find, for instance, equivalences that hold under any possible assignment of such variables; or synthesize assignments for which a candidate partition is a differential equivalence. The ability to reason symbolically can be particularly useful in domains such as computational biology, where uncertainty on rate parameters is a well-known hindrance.

[1] MAPK cascade in yeast - dimerization of Ste5. Available at `http://vcell.org/bionetgen/samples.html`.

[2] A. Antoulas. *Approximation of Large-Scale Dynamical Systems*. Advances in Design and Control. SIAM, 2005.

[3] U. M. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, 1988.

[4] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model-checking continuous-time Markov chains. *ACM Trans. Comput. Logic*, 1(1):162–170, 2000.

[5] C. Baier and J.-P. Katoen. *Principles of model checking*. MIT Press, 2008. ISBN 978-0-262-02649-9.

[6] C. Baier, B. Engelen, and M. E. Majster-Cederbaum. Deciding bisimilarity and similarity for probabilistic processes. *J. Comput. Syst. Sci.*, 60 (1):187–231, 2000.

[7] C. Barrett, R. Sebastiani, S. A. Seshia, S. A. S. Cesare Tinelli Clark Barrett, Roberto Sebastiani, and C. Tinelli. *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*, chapter Satisfiability Modulo Theories. IOS Press, 2009.

[8] D. Barua and B. Goldstein. A mechanistic model of early FcεRI signaling: lipid rafts and the question of protection from dephosphorylation. *PLoS One*, 7(12), 2012.

[9] J. Berdine and N. Bjørner. Computing all implied equalities via SMT-based partition refinement. In *Automated Reasoning*, volume 8562 of *LNCS*, pages 168–183. Springer, 2014.

[10] M. Bernardo. A survey of Markovian behavioral equivalences. In *Formal Methods for Performance Evaluation*, volume 4486 of *LNCS*, pages 180–219. Springer, 2007.

[11] M. L. Blinov, J. R. Faeder, B. Goldstein, and W. S. Hlavacek. BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20(17):3289–3291, 2004.

[12] M. Boreale. Algebra, coalgebra, and minimization in polynomial differential equations. In *FOSSACS*, pages 71–87, 2017. doi: 10.1007/978-3-662-54458-7_5.

[13] J. T. Bradley and W. J. Knottenbelt. The ipc/hydra tool chain for the analysis of PEPA models. In *QEST*, pages 334–335, 2004. doi: 10.1109/QEST.2004.1348054.

[14] P. Buchholz. Exact and ordinary lumpability in finite markov chains. *Journal of Applied Probability*, 31(1):59–75, 1994.

[15] P. Buchholz. Exact performance equivalence: An equivalence relation for stochastic automata. *Theoretical Computer Science*, 215(1–2): 263–287, 1999.

[16] L. Calzone, F. Fages, and S. Soliman. BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics*, 22(14):1805–1807, 2006.

[17] F. Camporesi and J. Feret. Formal reduction for rule-based models. *Electronic Notes in Theoretical Computer Science*, 276:29–59, 2011.

[18] F. Camporesi, J. Feret, H. Koeppl, and T. Petrov. Combining model reductions. *Electronic Notes in Theoretical Computer Science*, 265:73–96, 2010.

[19] F. Camporesi, J. Feret, and K. Q. Lý. Kade: A tool to compile kappa rules into (reduced) ODE models. In *CMSB*, pages 291–299, 2017. doi: 10.1007/978-3-319-67471-1_18.

[20] L. Cardelli. On process rate semantics. *Theoretical Computer Science*, 391(3):190–215, 2008.

[21] L. Cardelli. Morphisms of reaction networks that couple structure to function. *BMC Systems Biology*, 8(1):84, 2014.

[22] L. Cardelli and A. Csikász-Nagy. The cell cycle switch computes approximate majority. *Sci. Rep.*, 2, 2012.

[23] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. Forward and backward bisimulations for chemical reaction networks. In *CONCUR*, pages 226–239, 2015.

[24] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. Symbolic computation of differential equivalences. In *POPL*, pages 137–150, 2016.

[25] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. Efficient syntax-driven lumping of differential equations. In *TACAS 2016*, pages 93–111, 2016.

[26] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. Comparing chemical reaction networks: A categorical and algorithmic perspective. In *LICS*, pages 485–494, 2016.

[27] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. Maximal aggregation of polynomial dynamical systems. *Proceedings of the National Academy of Sciences*, 2017. doi: 10.1073/pnas.1702697114.

[28] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. ERODE: A tool for the evaluation and reduction of ordinary differential equations. In *TACAS 2017*, pages 310–328, 2017.

[29] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. Comparing chemical reaction networks: A categorical and algorithmic perspective. In *Theoretical Computer Science*, 2017. doi: https://doi.org/10.1016/j.tcs.2017.12.018.

[30] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. Guaranteed error bounds on approximate model abstractions through reachability analysis. In *QEST*, pages 104–121, 2018. doi: 10.1007/978-3-319-99154-2_7.

[31] F. Ciocchetta and J. Hillston. Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theoretical Computer Science*, 410(33-34):3065–3084, 2009.

[32] J. Colvin, M. I. Monine, J. R. Faeder, W. S. Hlavacek, D. D. V. Hoff, and R. G. Posner. Simulation of large-scale rule-based models. *Bioinformatics*, 25(7):910–917, 2009.

[33] J. Colvin, M. I. Monine, R. N. Gutenkunst, W. S. Hlavacek, D. D. V. Hoff, and R. G. Posner. Rulemonkey: software for stochastic simulation of rule-based models. *BMC Bioinformatics*, 11:404, 2010.

[34] H. Conzelmann, J. Saez-Rodriguez, T. Sauter, B. Kholodenko, and E. Gilles. A domain-oriented approach to the reduction of combinatorial complexity in signal transduction networks. *BMC Bioinformatics*, 7(1):34, 2006.

[35] H. Conzelmann, D. Fey, and E. Gilles. Exact model reduction of combinatorial reaction networks. *BMC Systems Biology*, 2(1):78, 2008.

[36] V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.

[37] V. Danos, J. Desharnais, F. Laviolette, and P. Panangaden. Bisimulation and cocongruence for probabilistic systems. *Information and Computation*, 204(4):503–523, 2006.

[38] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Abstracting the differential semantics of rule-based models: Exact and automated model reduction. In *LICS*, pages 362–381, 2010.

[39] R. David and H. Alla. *Discrete, Continuous, and Hybrid Petri Nets*. Springer, 2005.

[40] L. De Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS*, pages 337–340, 2008.

[41] R. De Nicola, D. Latella, M. Loreti, and M. Massink. A uniform definition of stochastic process calculi. *ACM Computing Surveys*, 46(1): 5:1–5:35, 2013.

[42] C. Dehnert, J.-P. Katoen, and D. Parker. SMT-based bisimulation minimisation of Markov models. In *VMCAI*, volume 7737 of *LNCS*, pages 28–47, 2013.

[43] S. Derisavi, H. Hermanns, and W. H. Sanders. Optimal state-space lumping in Markov chains. *Inf. Process. Lett.*, 87(6):309–315, 2003.

[44] J. R. Faeder, W. S. Hlavacek, I. Reischl, M. L. Blinov, H. Metzger, A. Redondo, C. Wofsy, and B. Goldstein. Investigation of early events in FcεRI-mediated signaling using a detailed mathematical model. *The Journal of Immunology*, 170(7):3769–3781, 2003.

[45] J. Feret. Fragments-based model reduction: Some case studies. *Electronic Notes in Theoretical Computer Science*, 268:77–96, 2010.

[46] J. Feret. An algebraic approach for inferring and using symmetries in rule-based models. *Electr. Notes Theor. Comput. Sci.*, 316:45–65, 2015. doi: 10.1016/j.entcs.2015.06.010.

[47] J. Feret, V. Danos, J. Krivine, R. Harmer, and W. Fontana. Internal coarse-graining of molecular systems. *Proceedings of the National Academy of Sciences*, 106(16):6453–6458, 2009.

[48] J. Feret, T. Henzinger, H. Koeppl, and T. Petrov. Lumpability abstractions of rule-based systems. *Theoretical Computer Science*, 431:137–164, 2012.

[49] J. Fisher and T. Henzinger. Executable cell biology. *Nature Biotechnology*, 25(11):1239–1249, 2007.

[50] V. Galpin. Continuous approximation of PEPA models and Petri nets. *International Journal of Computer Aided Engineering and Technology*, 2:324–339, 2010.

[51] S. Gao, S. Kong, and E. Clarke. Satisfiability modulo ODEs. In *FMCAD*, pages 105–112, 2013.

[52] K. Ghorbal and A. Platzer. Characterizing Algebraic Invariants by Differential Radical Invariants. In *TACAS*, pages 279–294, 2014. doi: 10.1007/978-3-642-54862-8_19.

[53] C. Gu. QLMOR: A Projection-Based Nonlinear Model Order Reduction Approach Using Quadratic-Linear Representation of Nonlinear Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(9):1307–1320, 2011. doi: 10.1109/TCAD.2011.2142184.

[54] S. Gulwani, S. Jha, A. Tiwari, and R. Venkatesan. Synthesis of loop-free programs. In *PLDI*, pages 62–73, 2011.

[55] R. A. Hayden and J. T. Bradley. A fluid analysis framework for a Markovian process algebra. *Theoretical Computer Science*, 411(22-24): 2260–2297, 2010.

[56] M. Heiner, D. Gilbert, and R. Donaldson. Petri nets for systems and synthetic biology. In *Formal Methods for Computational Systems Biology*, volume 5016 of *LNCS*, pages 215–264. Springer, 2008.

[57] H. Hermanns and M. Rettelbach. Syntax, semantics, equivalences, and axioms for MTIPP. In *Proceedings of Process Algebra and Probabilistic Methods*, pages 71–87, Erlangen, 1994.

[58] H. Hermanns and M. Siegle. Bisimulation algorithms for stochastic process algebras and their BDD-based implementation. In *ARTS*, pages 244–264, 1999.

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

[60] J. Hillston. Fluid flow approximation of PEPA models. In *QEST*, pages 33–43, Sept. 2005.

[61] D. T. Huynh and L. Tian. On some equivalence relations for probabilistic processes. *Fundam. Inform.*, 17(3):211–234, 1992.

[62] G. Iacobelli, M. Tribastone, and A. Vandin. Differential bisimulation for a Markovian process algebra. In *MFCS*, pages 293–306, 2015.

[63] D. Jovanovic and L. M. de Moura. Solving non-linear arithmetic. In *IJCAR*, pages 339–354, 2012.

[64] P. C. Kanellakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Inf. Comput.*, 86(1):43–68, 1990.

[65] J. Kemeny and J. Snell. *Finite Markov Chains*. Springer New York, Heidelberg, Berlin, 1976.

[66] P. Kocieniewski, J. R. Faeder, and T. Lipniacki. The interplay of double phosphorylation and scaffolding in MAPK pathways. *Journal of Theoretical Biology*, 295:116–124, 2012.

[67] A. S. Köksal, V. Kuncak, and P. Suter. Constraints as control. In *POPL*, pages 151–164, 2012.

[68] M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In *CAV*, pages 585–591, 2011.

[69] M. Kwiatkowski and I. Stark. The continuous pi-calculus: A process algebra for biochemical modelling. In *CMSB*, pages 103–122, 2008.

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

[71] Y. Li, A. Albarghouthi, Z. Kincaid, A. Gurfinkel, and M. Chechik. Symbolic optimization with SMT solvers. In *POPL*, pages 607–618, 2014.

[72] S. Mover, A. Cimatti, A. Tiwari, and S. Tonetta. Time-aware relational abstractions for hybrid systems. In *EMSOFT*, pages 1–10, 2013.

[73] M. Nagasaki, S. Onami, S. Miyano, and H. Kitano. Bio-calculus: Its concept and molecular interaction. *Genome Informatics*, 10:133–143, 1999.

[74] D. L. Nelson and M. M. Cox. *Lehninger Principles of Biochemistry*. Palgrave Macmillan, 6th edition, 2013.

[75] J. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

[76] M. S. Okino and M. L. Mavrovouniotis. Simplification of mathematical models of chemical reaction systems. *Chemical Reviews*, 2(98): 391–408, 1998.

[77] R. Paige and R. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[78] G. J. Pappas. Bisimilar linear systems. *Automatica*, 39(12):2035–2047, 2003.

[79] M. Pedersen and G. Plotkin. A language for biochemical systems: Design and formal specification. In *Transactions on Computational Systems Biology XII*, volume 5945 of *LNCS*, pages 77–145. Springer, 2010.

[80] A. D. Pierro, C. Hankin, and H. Wiklicky. Quantitative relations and approximate process equivalences. In *CONCUR*, pages 498–512, 2003.

[81] A. Platzer and Y. K. Tan. Differential Equation Axiomatization: The Impressive Power of Differential Ghosts. In *LICS*, pages 819–828, 2018. doi: 10.1145/3209108.3209147.

[82] A. Regev and E. Shapiro. Cellular abstractions: Cells as computation. *Nature*, 419(6905):343–343, 2002.

[83] S. Sankaranarayanan and A. Tiwari. Relational abstractions for continuous and hybrid systems. In *CAV*, pages 686–702, 2011.

[84] M. W. Sneddon, J. R. Faeder, and T. Emonet. Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. *Nature Methods*, 8(2):177–183, 2011.

[85] J. Sproston and S. Donatelli. Backward bisimulation in Markov chain model checking. *IEEE Trans. Software Eng.*, 32(8):531–546, 2006.

[86] W. J. Stewart. *Probability, Markov Chains, Queues, and Simulation*. Princeton University Press, 2009.

[87] R. Suderman and E. J. Deeds. Machines vs. ensembles: Effective MAPK signaling through heterogeneous sets of protein complexes. *PLoS Comput. Biol.*, 9(10):e1003278, 10 2013.

[88] S. Tognazzi, M. Tribastone, M. Tschaikowski, and A. Vandin. EGAC: A genetic algorithm to compare chemical reaction networks. In *The Genetic and Evolutionary Computation Conference (GECCO)*, 2017. To appear.

[89] J. Toth, G. Li, H. Rabitz, and A. S. Tomlin. The effect of lumping and expanding on kinetic differential equations. *SIAM Journal on Applied Mathematics*, 57(6):1531–1556, 1997.

[90] M. Tribastone. A fluid model for layered queueing networks. *IEEE Trans. Software Eng.*, 39(6):744–756, 2013.

[91] M. Tribastone, S. Gilmore, and J. Hillston. Scalable differential analysis of process algebra models. *IEEE Trans. Software Eng.*, 38(1):205–219, 2012.

[92] M. Tschaikowski and M. Tribastone. Exact fluid lumpability for Markovian process algebra. In *CONCUR*, pages 380–394, 2012.

[93] M. Tschaikowski and M. Tribastone. Tackling continuous state-space explosion in a Markovian process algebra. *TCS*, 517:1–33, 2014. doi: 10.1016/j.tcs.2013.08.016.

[94] M. Tschaikowski and M. Tribastone. A unified framework for differential aggregations in Markovian process algebra. *Journal of Logical and Algebraic Methods in Programming*, 84(2):238–258, 2015.

[95] A. Valmari and G. Franceschinis. Simple $O(m \log n)$ time Markov chain lumping. In *TACAS*, pages 38–52, 2010.

[96] A. J. van der Schaft. Equivalence of dynamical systems by bisimulation. *IEEE Transactions on Automatic Control*, 49, 2004.

[97] E. O. Voit. Biochemical systems theory: A review. *ISRN Biomathematics*, 2013:53, 2013.