Efficient Local Computation of Differential Bisimulations via Coupling and Up-to Methods

Giorgio Bacci*, Giovanni Bacci*, Kim G. Larsen*, Mirco Tribastone†, Max Tschaikowski*, and Andrea Vandin‡
*Department of Computer Science, Aalborg University, Denmark
†SysMA Research Unit, IMT Lucca, Italy
‡Institute of Economics, Scuola Superiore Sant'Anna, Italy

***Abstract*—We introduce polynomial couplings, a generalization of probabilistic couplings, to develop an algorithm for the computation of equivalence relations which can be interpreted as a lifting of probabilistic bisimulation to polynomial differential equations, a ubiquitous model of dynamical systems across science and engineering. The algorithm enjoys polynomial time complexity and complements classical partition-refinement approaches because: (a) it implements a local exploration of the system, possibly yielding equivalences that do not necessarily involve the inspection of the whole system of differential equations; (b) it can be enhanced by up-to techniques; and (c) it allows the specification of pairs which ought not be included in the output. Using a prototype, these advantages are demonstrated on case studies from systems biology for applications to model reduction and comparison. Notably, we report four orders of magnitude smaller runtimes than partition-refinement approaches when disproving equivalences between Markov chains.**

I. Introduction

Ordinary differential equations (ODEs) are a fundamental tool for modelling systems with continuous-time dynamics across science and engineering. In computer science, ODEs are central for the quantitative analysis of systems. For instance, in a continuous-time Markov chain (CTMC) a system of linear ODEs gives the forward equations of motion of the probability distribution [1]; ODEs also serve as the underlying semantics of formal languages based on process algebra, with application to the analysis of distributed computing systems [2] or computational systems biology [3].

Analogously to the classical non-deterministic setting based on labeled transition systems, bisimulations for ODEs have been proposed for the related purposes of model *comparison* and model *reduction* [4], [5], [6], [7], with applications that have transcended computer science (e.g., [8], [9]). "Lumping" refers to a class of methods to reduce a system of ODEs onto lower-dimensional space such that each variable in the reduced ODE system represents an appropriate mapping of the set of original variables [10], [11].

An equivalence relation over the variables of an ODE system can be seen as a specific type of lumping. Indeed, this is well-known for CTMCs, where *backward* and *forward bisimulations* (e.g., [12], [13], [14]) can be computed using lumping algorithms based on partition refinement [15], [16]. More recently, partition refinement algorithms have been provided for a class of nonlinear ODEs, by means of symbolic approaches based on satisfiability modulo theories [5], and polynomial ODEs, generalizing bisimulation relations and related lumping algorithms for CTMCs [17].

Partition refinement is efficient in finding the largest equivalence, i.e., the coarsest aggregation of an ODE system. However, it performs a *global* exploration of the state space, i.e., it requires the availability of the whole system of ODEs. Moreover, there are certain applications for which alternative approaches may be more desirable. For example, when comparing two models, it would suffice to know whether there exists *some* bisimulation relating the pair of initial states. For large-scale models in particular, it would be useful to prove (or disprove) this by computing smaller relations that do not necessarily involve exploring the whole state space.

Another question where partition refinement may not be appropriate regards the quest for a bisimulation that *does not* contain some pairs. This is motivated by the fact that relating two variables may impose certain pre-conditions on the quotient model that domain-specific knowledge must exclude. For instance, in a backward bisimulation for CTMCs (related to the notion of exact lumpability [14]), related states must be initialized with the same initial probability; however, this may not always be meaningful from a modeling viewpoint. For example, variables that represent distinct discrete states of some model components (e.g., genes which can be active/inactive) must start from independent initial conditions. In applications to dynamic models of regulatory networks, interesting research findings regard genes that respond identically to different external stimuli, a.k.a. *gene co-expression networks* (e.g., [18], [19]). To formally prove such a fact in the model, one would wish to find a bisimulation relation that relates the variables representing the genes of interest, but not those encoding the external stimuli. The problem with such kinds of *negative constraints* is intrinsic to the refinement technique. Indeed, it works by splitting candidate equivalence classes into finer partitions; crucially, however, the modeler is left with the choice of the initial guess among the possibly exponentially many satisfying given negative constraints.

In this paper we propose a new approach that aims to tackle the above issues by presenting an algorithm for the construction of bisimulations based on a local exploration of the model, starting with a candidate relation containing the pairs of ODE variables to be proved equivalent as well as a set of constraints containing pairs that are not allowed in the output. We

978-1-6654-4895-6/21/$31.00 ©2021 IEEE

consider ODE systems with polynomial derivatives. This is an important class of ODEs on its own—e.g., it subsumes models with the well-known mass-action kinetics arising in many natural sciences [20]. Additionally, ODEs with other forms of nonlinearity (e.g., trigonometric functions, exponentials, rational expressions) can be algorithmically translated into polynomial ODEs [21].

We study bisimulations for two equivalences, and related partition-refinement algorithms, proposed for polynomial ODEs: *backward differential equivalence* (BDE) and *forward differential equivalence* (FDE) [5], [17]. The former relates ODE variables whose solutions are equal when started from identical initial conditions. The latter, instead, guarantees that the solution of the induced aggregated ODE corresponds to the exact sum of the ODE variables in each equivalence class. Notably, when restricted to a CTMC, BDE and FDE correspond to CTMC backward and forward bisimulation, respectively. Therefore, our results for polynomial ODEs yield a local algorithm for CTMC bisimulations as a special case.

Our approach for the comparison of polynomials fundamentally exploits (a variant of) the *coupling method*, a standard tool in probability theory [22], [23]. In our setting, we adapt the notion of probabilistic couplings to that of *linear* and *monomial couplings*. Intuitively, a coupling is a pairing of equivalent variables that respects the sum of the coefficients in the polynomials. Proving the existence of such a coupling is equivalent to proving the equivalence of the polynomials.

The benefit in using them is that couplings can be computed efficiently by solving a transportation problem [24], [25] in strongly polynomial time (e.g., by employing Orlin's algorithm [26]). This allows us to design a local algorithm that runs in polynomial time in the number of monomials present in the polynomial ODE system.

For further computational improvement, we enhance the local algorithm by employing *coinduction up-to techniques*. Since their introduction [27], coinduction up-to techniques were proved useful in numerous proofs about concurrent systems (see [28] for a list of references), found applications in abstract interpretation [29], and more recently, to improve standard automata algorithms [30], [31].

Using a prototype, we apply our algorithm to a number of case studies from different domains (chemical reaction networks, gene regulatory networks, CTMCs, and epidemiological models) to show: (i) how our algorithm allows the natural encoding of domain-relevant queries that would otherwise require tedious trial-and-error with partition refinement; (ii) how our algorithm allows for a local analysis of the polynomial ODE system possibly avoiding a full exploration of the state space; and (iii) how the enhancement of up-to-techniques leads to faster runtimes. When applied to the problem of checking equivalence between CTMCs, motivated by fundamental research questions in evolutionary biology (e.g., [32]), our local algorithm outperforms the partition-refinement algorithms for BDE and FDE by up to four orders of magnitude.

Further Related Work: The closest line of research to this paper is by Boreale [33], [7], who has introduced the notion $\mathcal{L}$-bisimulation for polynomial ODEs and an algorithm for computing it using up-to context techniques. $\mathcal{L}$-bisimulation generalizes BDE because it can prove more general invariants. However, it is not comparable to FDE [33]. In addition, the procedure for computing $\mathcal{L}$-bisimulations exploits Buchberger's algorithm [34], which has doubly exponential time complexity.

Polynomial invariants of dynamical systems have been also studied in the field of verification where model minimization is not sought per se. For instance, [35] computes these by maximizing the dimension of the kernel of a symbolic matrix. As further works we mention [36], [37] and refer to [33] for a detailed discussion.

Probabilistic couplings have been used to characterize [38] and provide efficient methods [39] to test probabilistic bisimulation for discrete-time Markov chains [12]. More recently, they were instrumental for devising efficient methods for computing bisimilarity distances [40], [41]. Coupling methods, are successfully employed for proving invariants in probabilistic programs [42], [43], with applications in formal program verification [44], [45], security [46], [47], and randomized algorithms [48]. To our knowledge, this is the first time that coupling methods are employed in the analysis of ODEs.

Synopsis: Section II recalls the concepts of BDE and FDE and fixes notation. Section III introduces linear and monomial couplings and shows how these are used to compare polynomials; Section IV provides fixed point characterizations of BDE and FDE using coupling methods. Section V presents the local algorithm for computing differential bisimulations and discusses how to further enhance its performance by integrating up-to techniques. In Section VI we report experimental results on a number of applications highlighting the flexibility of our method as opposed to partition refinement.

II. Preliminaries

Notation: Fix a finite set of variables X which will appear in polynomial ODE systems. For a binary relation $R \subseteq X \times X$, we denote by $r(R)$, $s(R)$, $t(R)$, and $e(R)$ respectively the reflexive, symmetric, transitive, and equivalence closure of R.

We denote by $\mathbf{M}[X]$ the set of monomials over X, ranged over by $m, n, \ldots \in \mathbf{M}[X]$; by $\mathbf{L}[X]$ the set of linear combinations over X, ranged over by $g, h, \cdots \in \mathbf{L}[X]$; and by $\mathbf{P}[X]$ the set of polynomials over X, ranged over by $p, q, \ldots \in \mathbf{P}[X]$. Throughout the paper, we will assume that every polynomial is expressed as a linear combination of monomials, i.e., $p = \sum_{m_i \in I} \alpha_{m_i} m_i$ where $\alpha_{m_i} \in \mathbb{R}$.[1]

A linear combination $g \in \mathbf{L}[X]$ can be decomposed into its *positive* and *negative* parts so that $g = g^+ - g^-$, where $g^+ = \sum_{x_i \in I} \alpha_{x_i} x_i$ and $g^- = \sum_{x_j \in J} \alpha_{x_j} x_j$ for positive coefficients $\alpha_{x_l} > 0$. This decomposition is unique once we require I and J to be disjoint. For a monomial $m = \prod_{x_i \in I} x_i^{a_i}$ we write $m(x_i)$ for the exponent $a_i \in \mathbb{N}$ associated with the variable x_i, whereas for a linear combination $g = \sum_{x_i \in I} \alpha_{x_i} x_i$ we write $g(x_i)$ for the coefficient α_{x_i} associated with the variable x_i. For

[1]This assumption is not only for convenience but will allow for an unambiguous definition of the size of a polynomial vector field.

a polynomial $p = \sum_{m_i \in I} \alpha_{m_i} m_i$, instead, $p(m_i)$ denotes the coefficient α_{m_i} associated with the monomial m_i. When X is clear from the context, we will write $\mathbf{M}$, $\mathbf{L}$, and $\mathbf{P}$ respectively for $\mathbf{M}[X]$, $\mathbf{L}[X]$, and $\mathbf{P}[X]$.

Polynomial vector fields: A *vector field* over X is a map $f\colon \mathbb{R}^X \to \mathbb{R}^X$ that is totally differentiable. It is called *polynomial* when, for all $x \in X$, f_x is a polynomial over X. Given a polynomial vector field f, we write $f(v)$ for the evaluation of f at $v \in \mathbb{R}^X$. For an initial condition $v(0) \in \mathbb{R}^X$, Picard-Lindelöf's theorem ensures that the ODE system $\partial_t v(t) = f(v(t))$ induced by f has a unique solution $v\colon dom(v) \to \mathbb{R}^X$, $t \mapsto v(t)$, where ∂_t denotes derivative with respect to time.

Next we introduce an example of a polynomial vector field arising from a simple application from systems biology which will be used throughout the paper for illustration.

Example 1. *Consider a chemical reaction network (CRN) where an enzyme B can bind forming a complex with a substrate A through two independent binding sites according to the following reversible reactions R1, ..., R4:*

$$\begin{array}{llll} R1\colon & A_{00} + B \underset{3}{\overset{2}{\rightleftharpoons}} A_{10} & R2\colon & A_{00} + B \underset{3}{\overset{2}{\rightleftharpoons}} A_{01} \\ R3\colon & A_{10} + B \underset{3}{\overset{1}{\rightleftharpoons}} A_{11} & R4\colon & A_{01} + B \underset{3}{\overset{1}{\rightleftharpoons}} A_{11} \end{array}$$

The subscripts i, j in chemical species A_{ij} denote the availability of either binding site in the substrate A. Reactions R1 and R2 model reversible binding at either site. Reactions R3 and R4 model the case when only one site is available for binding. The value on each arrow indicates the kinetic rate parameter *for the reaction. By mass-action kinetics [20], the above CRN gives rise to the ODE system*

$$\begin{aligned} &\partial_t v_{A_{00}}(t) = f_{A_{00}}(v(t)), \qquad \partial_t v_{A_{01}}(t) = f_{A_{01}}(v(t)), \\ &\partial_t v_{A_{10}}(t) = f_{A_{10}}(v(t)), \qquad \partial_t v_{A_{11}}(t) = f_{A_{11}}(v(t)), \\ &\partial_t v_B(t) = f_B(v(t)) \end{aligned}$$

induced by the following polynomial vector field

$$\begin{aligned} f_{A_{00}} &= -4A_{00}B + 3A_{10} + 3A_{01} \\ f_{A_{01}} &= 2A_{00}B - 3A_{01} - A_{01}B + 3A_{11} \\ f_{A_{10}} &= 2A_{00}B - 3A_{10} - A_{10}B + 3A_{11} \\ f_{A_{11}} &= A_{10}B + A_{01}B - 6A_{11} \\ f_B &= -4A_{00}B + 3A_{10} + 3A_{01} - A_{10}B - A_{01}B + 6A_{11} \end{aligned} \tag{1}$$

with variables $X = \{A_{00}, A_{01}, A_{10}, A_{11}, B\}$.

Backward differential equivalence: We recall the notion of backward differential equivalence (BDE) from [5].

Definition 1 (Backward differential equivalence)**.** *Let f be a vector field over X. An equivalence relation $R \subseteq X \times X$ is a BDE for f if the implication*

$$\Big(\bigwedge_{(x,y)\in R} v_x = v_y \Big) \Rightarrow \Big(\bigwedge_{(x,y)\in R} f_x(v) = f_y(v) \Big)$$

is true for all $v \in \mathbb{R}^X$.

Example 2. *Consider the vector field f from Example 1. Then, the equivalence relation*

$$R = \mathtt{id} \cup \{(A_{01}, A_{10}), (A_{10}, A_{01})\},$$

where $\mathtt{id} = \{(x, x) \mid x \in X\}$ denotes the identity relation, is a BDE for f given in (1).

A BDE relates variables with identical ODE solutions when initialized equally [5, Theorem 3]. This property allows one to reason about the solutions of the ODE system induced by f by looking at the smaller ODE system induced by the vector field $\hat{f}\colon \mathbb{R}^{X/R} \to \mathbb{R}^{X/R}$ obtained by a change variable $H = x$, for each $H \in X/R$ and $x \in H$, mapping each variable to its equivalence class.

Example 3. *The equivalence classes of R from Example 2 are*

$$H_1 = \{A_{00}\}, H_2 = \{A_{01}, A_{10}\}, H_3 = \{A_{11}\}, H_4 = \{B\}.$$

Then, the BDE-reduced vector field of f given in (1) *is*

$$\begin{aligned} \hat{f}_{H_1} &= -4H_1H_4 + 6H_2 \\ \hat{f}_{H_2} &= 2H_1H_4 - 3H_2 - H_2H_4 + 3H_3 \\ \hat{f}_{H_3} &= H_2H_4 + H_2H_4 - 6H_3 \\ \hat{f}_{H_4} &= -4H_1H_4 + 6H_2 - H_2H_4 - H_2H_4 + 6H_3 \end{aligned}$$

Let v and $\hat{v}$ denote the solutions of the ODEs induced respectively by f and $\hat{f}$. Since R is a BDE for f, for all $t > 0$, we have that

$$\begin{aligned} &\hat{v}_{H_1}(t) = v_{A_{00}}(t) \qquad \hat{v}_{H_2}(t) = v_{A_{10}}(t) = v_{A_{01}}(t) \\ &\hat{v}_{H_3}(t) = v_{A_{11}}(t) \qquad \hat{v}_{H_4}(t) = v_B(t), \end{aligned}$$

provided that the above identities are satisfied at $t = 0$.

Forward equivalence: We recall the definition of forward differential equivalence (FDE) from [5].

Definition 2 (Forward differential equivalence)**.** *Let f be a vector field over X. An equivalence relation $R \subseteq X \times X$ is an FDE for f if $(x, y) \in R$ implies*

$$\sum_{z\in H} f_z(v) = \sum_{z\in H} f_z[x/\lambda(x+y), y/(1-\lambda)(x+y)](v)$$

for all $H \in X/R$ and $v \in \mathbb{R}^{X \uplus \{\lambda\}}$, where $f[x/y]$ denotes the term arising when x is replaced with y in f.

An FDE yields a self-consistent reduced ODE system that gives the dynamics of the sum of the original variables for each equivalence class [5, Theorem 1]. Given an FDE R for the vector field f, one can define the corresponding R-quotient vector field $\hat{f}\colon \mathbb{R}^{X/R} \to \mathbb{R}^{X/R}$ obtained from f by a change of variable $H = \sum_{x\in H} x$, for each $H \in X/R$.

Example 4. *Let R and $X/R = \{H_1, \dots, H_4\}$ be as in Example 3. The relation R can be also shown to be an FDE for f in* (1)*. The FDE-reduced vector field of f w.r.t. R is*

$$\begin{aligned}
\hat{f}_{H_1} &= -4H_1H_4 + 3H_2 \\
\hat{f}_{H_2} &= 4H_1H_4 - 3H_2 - H_2H_4 + 6H_3 \\
\hat{f}_{H_3} &= H_2H_4 - 6H_3 \\
\hat{f}_{H_4} &= -4H_1H_4 + 3H_2 - H_2H_4 + 6H_3
\end{aligned}$$

Let v and $\hat{v}$ denote the solutions of the ODEs induced respectively by f and $\hat{f}$. Since R is an FDE for f, for all $t > 0$, we have that

$$\begin{aligned}
\hat{v}_{H_1}(t) &= v_{A_{00}}(t) \qquad & \hat{v}_{H_2}(t) &= v_{A_{10}}(t) + v_{A_{01}}(t) \\
\hat{v}_{H_3}(t) &= v_{A_{11}}(t) \qquad & \hat{v}_{H_4}(t) &= v_B(t),
\end{aligned}$$

provided that the above identities are satisfied at $t = 0$.

In contrast to BDE, whose relation with the ODEs solution are conditional on the initial value $v(0)$, FDE does not make any such assumptions. At the same time, FDE preserves only the sums of the solutions of the original variables, while BDE preserves the solutions of the original variables in full. Although in the previous example the same relation is both an FDE and an BDE, the two notions are not comparable in general [49]. Indeed, this derives also from the fact that forward and backward bisimulations for Markov chains are not comparable.

For polynomial vector fields, the largest BDE and FDE exist and can be computed by a partition refinement algorithm with polynomial complexity in the size of the vector field [17]. If applied to (1), for instance, the algorithm confirms that R from Example 2 (reps. Example 4) is the largest BDE (resp. FDE).

III. Coupling method for polynomials

Our approach for reasoning about equivalences for polynomials is based on the *coupling method* and on the celebrated proof technique by Strassen [50] for checking stochastic dominance of random variables. Here we extend the concept of coupling to linear combinations and monomials and prove Strassen-like theorems for these expressions. This will provide us with an efficient method for checking equivalence of polynomials based on the existence of suitable couplings.

Definition 3 (Linear coupling)**.** *Let g, h be two linear combinations over X. A map $\omega\colon X \times X \to \mathbb{R}_{\geq 0}$ is a* linear coupling *for (g, h) if the following conditions hold*

(i) $\forall x \in X.\ \sum_{y \in X} \omega(x, y) = (g^+ + h^-)(x)$;
(ii) $\forall y \in X.\ \sum_{x \in X} \omega(x, y) = (h^+ + g^-)(y)$;

where $g = g^+ - g^-$ and $h = h^+ - h^-$.

We denote by $\Gamma_{\mathbf{L}}(g, h)$ the set of linear couplings for (g, h). Note that $\Gamma_{\mathbf{L}}(g, h) \neq \emptyset$ iff $\sum_{x \in X}(g(x) - h(x)) = 0$.

Example 5. *Consider the linear combinations*

$$g = \overbrace{2x + 3y}^{g^+} - \big(\overbrace{3z + z'}^{g^-}\big)\,, \quad h = \overbrace{2x + 3y}^{h^+} - \big(\overbrace{3w + w'}^{h^-}\big)\,.$$

ω	x	y	z	z'	
x	2	0	0	0	$g^+(x)$
y	0	2	1	0	$g^+(y)$
w	0	1	2	0	$h^-(w)$
w'	0	0	0	1	$h^-(w')$
	$h^+(x)$	$h^+(y)$	$g^-(z)$	$g^-(z')$	

Fig. 1: Tabular representation of the linear coupling ω for (g, h) from Example 5. As for Definition 3, the sum of each row equals the corresponding coefficient in $g^+ + h^-$, while the sum of each column equals that of $h^+ + g^-$.

A linear coupling for (g, h) is

$$\begin{aligned}
&\omega(y, y) = 2\,, \qquad \omega(y, z) = 1\,, \qquad \omega(w, y) = 1\,, \\
&\omega(w, z) = 2\,, \qquad \omega(x, x) = 2\,, \qquad \omega(w', z') = 1\,,
\end{aligned}$$

and $\omega(\cdot, \cdot) = 0$ otherwise. Figure 1 provides a tabular visualization of the coupling ω. Another coupling for (g, h) is

$$\hat{\omega}(x, x) = 2\,, \quad \hat{\omega}(y, y) = 3\,, \quad \hat{\omega}(w', z') = 1\,, \quad \hat{\omega}(w, z) = 3\,,$$

and $\hat{\omega}(\cdot, \cdot) = 0$ otherwise.

The following theorem establishes an equivalence between an existential property. [2]

Theorem 1. *Let $R \subseteq X \times X$ be an equivalence relation and g, h two linear combinations over X. Then, the following are equivalent:*

1) There exists $\omega \in \Gamma_{\mathbf{L}}(g, h)$ such that $supp(\omega) \subseteq R$.
2) For all $v \in \mathbb{R}^X$, $\bigwedge_{(x,y) \in R} v_x \leq v_y$ implies $g(v) \leq h(v)$.
3) For all $v \in \mathbb{R}^X$, $\bigwedge_{(x,y) \in R} v_x = v_y$ implies $g(v) = h(v)$.

Moreover, (1) $\Rightarrow$ (2) $\wedge$ (3) *holds for arbitrary relations R.*

Remark 1. *The requirement of R being an equivalence cannot be relaxed as $g = 3x$, $h = x + y + z$ and $R = \{(x, y), (y, z)\}$ is a counterexample.*

Next we provide an analogous theorem for monomials. To this end we introduce the notion of monomial coupling.

Definition 4 (Monomial coupling)**.** *Let m, n be two monomials over X. A map $\rho\colon X \times X \to \mathbb{R}_{\geq 0}$ is a* monomial coupling *for (m, n) if the following conditions hold:*

(i) $\forall x \in X.\ \sum_{y \in X} \rho(x, y) = m(x)$;
(ii) $\forall y \in X.\ \sum_{x \in X} \rho(x, y) = n(y)$.

We denote by $\Gamma_{\mathbf{M}}(m, n)$ the set of monomial couplings for (m, n). Note that $\Gamma_{\mathbf{M}}(m, n) \neq \emptyset$ iff $\sum_{x \in X}(m(x) - n(x)) = 0$.

Example 6. *Consider the monomials $A_{01}B$ and $A_{10}B$ in (1). A monomial coupling ρ for pair the $(A_{01}B, A_{10}B)$ is*

$$\rho(A_{01}, A_{10}) = 1, \quad \rho(B, B) = 1 \quad \textit{and} \quad \rho(\cdot, \cdot) = 0 \quad \textit{otherwise}.$$

Again, there may be several couplings for the same pair, e.g.,

$$\hat{\rho}(A_{01}, B) = 1, \quad \hat{\rho}(B, A_{10}) = 1 \quad \textit{and} \quad \hat{\rho}(\cdot, \cdot) = 0 \quad \textit{otherwise}$$

is also a coupling for $(A_{01}B, A_{10}B)$.

[2] The proofs of all the results are available in [51].

The following is the variant of Strassen's theorem for monomials.

Theorem 2. *Let $R \subseteq X \times X$ be an equivalence relation and m, n two monomials over X. Then, the following are equivalent*

1) *There exists $\rho \in \Gamma_{\mathbf{M}}(m,n)$ such that $supp(\rho) \subseteq R$;*
2) *For all $v \in \mathbb{R}^X_{>0}$, $\bigwedge_{(x,y)\in R} v_x \leq v_y$ implies $m(v) \leq n(v)$;*
3) *For all $v \in \mathbb{R}^X$, $\bigwedge_{(x,y)\in R} v_x = v_y$ implies $m(v) = n(v)$.*

Moreover, (1) $\Rightarrow$ (2) $\wedge$ (3) *holds for arbitrary relations R.*

Remark 2. *The requirement of R being an equivalence cannot be relaxed as $m = x^3$, $n = xyz$, and $R = \{(x,y),(y,z)\}$ is a counterexample.*

Theorems 1 and 2 provide us with a convenient method to prove the equivalence of polynomials by using the coupling method. Before stating this result, we introduce the following notation which will be useful in the reminder of the paper.

Definition 5 (Liftings)**.** *For a relation $R \subseteq X \times X$, define*

$$\mathbf{L}[R] = \{(g,h) \in \mathbf{L} \times \mathbf{L} \mid \exists \omega \in \Gamma_{\mathbf{L}}(g,h)\,.supp(\omega) \subseteq R\}\,,$$
$$\mathbf{M}[R] = \{(m,n) \in \mathbf{M} \times \mathbf{M} \mid \exists \rho \in \Gamma_{\mathbf{M}}(m,n)\,.supp(\rho) \subseteq R\}\,.$$

We call $\mathbf{L}[R]$ the lifting of R over linear combinations, and $\mathbf{M}[R]$ the lifting of R over monomials.

Since we assume polynomials to always be expressed as linear combinations of monomials, we define the lifting of R over polynomials as $\mathbf{P}[R] = \mathbf{L}[\mathbf{M}[R]]$. With this in place, we state our desired result.

Corollary 1. *Let $R \subseteq X \times X$ and p, q two polynomials over X. Then, for all $v \in \mathbb{R}^X$,*

$$(p,q) \in \mathbf{P}[R] \quad \textit{implies} \quad \Big(\bigwedge_{(x,y)\in R} v_x = v_y\Big) \Rightarrow p(v) = q(v)\,.$$

If R is an equivalence, also the converse implication holds.

The following example illustrates how one can use Corollary 1 to prove equivalence among polynomials.

Example 7. *Consider the relation R from Example 2 and the polynomials $f_{A_{01}}$ $f_{A_{10}}$ from* (1)*. A linear coupling for $(f_{A_{01}}, f_{A_{10}})$ is*

$$\omega(A_{00}B, A_{00}B) = 2 \qquad \omega(A_{11}, A_{11}) = 3$$
$$\omega(A_{10}B, A_{01}B) = 1 \qquad \omega(A_{10}, A_{01}) = 3$$

and $\omega(\cdot,\cdot) = 0$ otherwise (cf. Example 5). Note that the support of ω identifies a matching of monomials that, if assumed equal, imply $f_{A_{01}} = f_{A_{10}}$ (by Theorem 1) as illustrated below:

$$f_{A_{01}} = 2A_{00}B + 3A_{11} - \big(3A_{01} + A_{01}B\big)\,,$$
$$f_{A_{10}} = 2A_{00}B + 3A_{11} - \big(3A_{10} + A_{10}B\big)\,.$$

Clearly $(A_{11}, A_{11}), (A_{10}, A_{01}) \in \mathbf{M}[R]$, and as illustrated in Example 6, also $(A_{00}B, A_{00}B), (A_{10}B, A_{01}B) \in \mathbf{M}[R]$. Therefore, $(f_{A_{01}}, f_{A_{10}}) \in \mathbf{P}[R]$. Again, the supports of the monomial couplings identify a matching of the variables that, by Corollary 1, when assumed to be equal, imply $f_{A_{01}} = f_{A_{10}}$ as illustrated below:

$$f_{A_{01}} = 2A_{00}B - 3A_{01} - A_{01}B + 3A_{11}\,,$$
$$f_{A_{10}} = 2A_{00}B - 3A_{10} - A_{10}B + 3A_{11}\,.$$

Note that this is in line with the fact that R is a BDE for the polynomial vector field f in (1)*.*

IV. From Couplings to Differential Equivalences

Here we give a coinductive characterization of BDE and FDE over polynomial vector fields as the greatest fixed point of two monotone operators whose definition is based on the notion of couplings. This will give us a coinduction proof principle which exploits coupling-based methods and will constitute the formal basis of our local algorithm (Section V).

Coinduction proof principle: For a monotone map b on the lattice of relations, the Knaster-Tarski fixed-point theorem characterizes the greatest fixed point $\mathtt{gfp}(b)$ as the greatest lower bound of all its post-fixed points (i.e., $\mathtt{gfp}(b) = \bigcup\{R \mid R \subseteq b(R)\}$). This leads to the coinduction proof principle illustrated below:

$$\frac{\exists R'\,, R \subseteq R' \subseteq b(R')}{R \subseteq \mathtt{gfp}(b)}\,.$$

By characterising BDE (resp. FDE) as the greatest fixed point of some monotone operator, we can exploit the above principle to show that a set R is contained in a BDE (resp. FDE) by providing a post-fixed point R' containing it. The operators will be defined in terms of a *backward* (resp., *forward*) *differential bisimulation*, discussed next.

A. Backward Differential Bisimulation

Definition 6. *Let f be a polynomial vector field over X. A relation $R \subseteq X \times X$ is a* backward differential bisimulation *(BDB) for f, if it is a post-fixed point of the following operator:*

$$\mathcal{B}^f(R) = \{(x,y) \in X \times X \mid (f_x, f_y) \in \mathbf{P}[R]\}.$$

It is easy to show that $\mathcal{B}^f$ is monotone in the lattice of relations, therefore $\mathtt{gfp}(\mathcal{B}^f)$ exists and is the greatest BDB. In the remainder, whenever the vector field f is clear from the context, we write $\mathcal{B}$ in place of $\mathcal{B}^f$.

Thanks to Corollary 1, we prove the following result. It ensures that finding a BDB, instead of a BDE, is enough to imply that related variables have the same solutions if initialized equally (cf. [5, Theorem 3]).

Proposition 1. *Let $f\colon \mathbb{R}^X \to \mathbb{R}^X$ be polynomial vector field and R a BDB. Then, for any $v \in \mathbb{R}^X$ we have*

$$\Big(\bigwedge_{(x,y)\in R} v_x = v_y\Big) \Rightarrow \Big(\bigwedge_{(x,y)\in R} f_x(v) = f_y(v)\Big)\,.$$

Example 8. *Consider the vector field $f : \mathbb{R}^X \to \mathbb{R}^X$ from (1) and let $R' = id \cup \{(A_{01}, A_{10})\}$. Then, it can be shown that*

$$\{(3A_{01}, 3A_{10}), (A_{01}B, A_{10}B)\} \subseteq \mathbf{M}[R']$$

From this, it is possible to infer that $(f_{A_{10}}, f_{A_{01}}) \in \mathbf{P}[R']$. It can be noted that R' is a BDB but not a BDE because it is not an equivalence relation.

The above example shows that not every BDB is a BDE. The following theorem clarifies the close connection among the notions of BDB and BDE.

Theorem 3 (Fixed-point characterization of BDE)**.** *Let f be a polynomial vector field over X and $R \subseteq X \times X$. Then, the following hold:*

(i) If R is a BDB then $e(R)$ is a BDE.
(ii) If R is a BDE then R is a BDB.
(iii) $\mathtt{gfp}(\mathcal{B})$ is the greatest BDE.

B. Forward Differential Bisimulation

Forward differential bisimulation (FDB) can be tied to the notion of *total derivative of a vector field* $f\colon \mathbb{R}^X \to \mathbb{R}^X$, commonly denoted by ∂f. It is given by the Jacobian matrix $(\partial_{x_j} f_{x_i})_{x_i,x_j \in X}$, where $\partial_{x_j} f_{x_i}$ denotes the partial derivative of $f_{x_i}\colon \mathbb{R}^X \to \mathbb{R}$ with respect to x_j. If f is a polynomial vector field, it is well-known (e.g., [52]) that the total derivative can be written as

$$(\partial_{x_j} f_{x_i})_{x_i,x_j \in X} = \textstyle\sum_{k=1}^{\kappa} m_k \cdot J_k, \tag{2}$$

where $m_1, \ldots, m_\kappa$ are pairwise different monomials over X and $J_1, \ldots, J_\kappa \in \mathbb{R}^{X \times X}$.

Example 9. *In the case of our running example (1), the Jacobian matrix $(\partial_{x_j} f_{x_i})_{x_i,x_j \in X}$ is given by*

$$\begin{pmatrix} -4B & 3 & 3 & 0 & -4A_{00} \\ 2B & -B-3 & 0 & 3 & 2A_{00}-A_{01} \\ 2B & 0 & -B-3 & 3 & 2A_{00}-A_{10} \\ 0 & B & B & -6 & A_{01}+A_{10} \\ -4B & 3-B & 3-B & 6 & -4A_{00}-A_{01}-A_{10} \end{pmatrix}$$

The decomposition (2) can be expressed by five matrices J_1, J_B, $J_{A_{00}}$, $J_{A_{01}}$ and $J_{A_{10}}$, where J_1 accounts for the constant monomial, while J_{x_i} accounts for the monomial x_i. For instance, $m_1 = 1$ and $m_B = B$, while

$$\underbrace{\begin{pmatrix} 0 & 3 & 3 & 0 & 0 \\ 0 & -3 & 0 & 3 & 0 \\ 0 & 0 & -3 & 3 & 0 \\ 0 & 0 & 0 & -6 & 0 \\ 0 & 3 & 3 & 6 & 0 \end{pmatrix}}_{J_1} \quad \underbrace{\begin{pmatrix} -4 & 0 & 0 & 0 & 0 \\ 2 & -1 & 0 & 0 & 0 \\ 2 & 0 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ -4 & -1 & -1 & 0 & 0 \end{pmatrix}}_{J_B}$$

We are now in the position to introduce the operator that will be used for the fixed-point characterization of FDE.

Definition 7. *Let f be a polynomial vector field over X and $F = \{J_1, \ldots, J_\kappa\}$ be as in (2). A relation $R \subseteq X \times X$ is a* forward differential bisimulation *(FDB) for f if it is a post-fixed point of the following operator:*

$$\mathcal{F}^f(R) = \bigcap_{J \in F} \mathcal{B}^{J^T}(R)\,.$$

Essentially, R is an FDB for f whenever it is a BDB of all $J_1^T, \ldots, J_\kappa^T$ in (2), where A^T is the transpose of matrix A.

The monotonicity of $\mathcal{F}^f$ follows by that of $\mathcal{B}$. Therefore, by Knaster-Tarski fixed-point theorem, $\mathtt{gfp}(\mathcal{F}^f)$ exists. In the following, whenever the vector field f is clear from the context, we write $\mathcal{F}$ in place of $\mathcal{F}^f$.

The following pivotal observation relies on [53, Lemma I.1] and reduces FDE to BDE.

Theorem 4. *Fix some index set X, some equivalence relation $R \subseteq X \times X$ and a polynomial vector field $f : \mathbb{R}^X \to \mathbb{R}^X$. Then, R is an FDE of f if and only if R is a BDE of each linear vector field $J_k^T : \mathbb{R}^X \to \mathbb{R}^X, v \mapsto J_k^T v$ from (2).*

We can now establish the connection between FDB and FDE, similarly to Theorem 3.

Theorem 5. *(Fixed-point characterization of FDE) Let f be a polynomial vector field over X and $R \subseteq X \times X$. Then, the following hold:*

(i) If R is an FDB, then $e(R)$ is an FDE.
(ii) If R is an FDE, then R is an FDB.
(iii) The $\mathtt{gfp}(\mathcal{F})$ is the greatest FDB.

Example 10. *It can be noted that R from Example 2 is an FDB because R is a BDB of J_1^T, J_B^T, $J_{A_{00}}^T$, $J_{A_{01}}^T$ and $J_{A_{11}}^T$.*

C. Constrained bisimulation

In line with Proposition 1, a BDB imposes pre-conditions on the initialization of related variables. As anticipated in Section I, from a modeling viewpoint one may want to express the need of *not* relating variables. A constrained BDB excludes certain pairs from being used in the relation.

Definition 8. *Let f be a polynomial vector field over X and $C \subseteq X \times X$. A relation $R \subseteq X \times X$ is a C-*constrained BDB *for f, if R is a BDB and $R \cap C = \emptyset$.*

Next, we show that also the the concepts of C-constrained BDB has a coinduction proof principle. To this end, consider the following operator:

$$\mathcal{B}_C^f(R) = \mathcal{B}^f(R \setminus C) \setminus C \tag{3}$$

As usual, when the vector field f is clear from the context, we write $\mathcal{B}_C$ in place of $\mathcal{B}_C^f$; and when $C = \emptyset$, we write $\mathcal{B}$ in place of $\mathcal{B}_C$.

Theorem 6. *Let f be a polynomial vector field over X and $R, C \subseteq X \times X$. Then, R is a C-constrained BDB for f iff $R \subseteq \mathcal{B}_C(R)$.*

For arbitrary $C \subseteq X \times X$, $\mathcal{B}_C^f$ is monotone because $\mathcal{B}^f$ is. Hence, by Knaster-Tarski fixed-point theorem, $\mathtt{gfp}(\mathcal{B}_C^f)$ exists and, by Theorem 6, is the greatest C-constrained BDB.

Remark 3. *By Theorem 3 and 6 we have that if R is a C-constrained BDB, then $e(R)$ is a BDE. Note however that, depending on the choice of $C \subseteq X \times X$, it may be the case that $e(R) \cap C \neq \emptyset$. In some cases, one needs to impose additional constraints to be able to find a C-constrained BDB which*

extends to a C-constrained BDE. Nevertheless, proving that no C-constrained BDB exists, directly implies that there is no BDE R with $R \cap C = \emptyset$.

V. Local Algorithm

As described in Section IV, one can exploit the coinduction proof principle to show that two variables x and y are related by a BDE, by providing a BDB R that contains (x, y). The classical approach based on partition refinement consists in computing $\mathtt{gfp}(\mathcal{B})$ as the limit of the decreasing chain $\{\mathcal{B}^i(X \times X)\}_{i \in \mathbb{N}}$. Here we propose a local approach that, starting from a relation $R \subseteq X \times X$ containing some query pairs, iteratively updates it until R is proven to be a BDB. This is done by performing a local exploration of the system, led by the dependencies that are discovered at each iteration. The following example illustrates the intuition behind our algorithm.

Example 11. *Consider the vector field in Eq.* (1) *and suppose we want to prove A_{01} and A_{10} to be backward equivalent. We do so by finding a relation R such that $(A_{01}, A_{10}) \in R$ and $R \subseteq \mathcal{B}(R)$. We start with $R = \{(A_{01}, A_{10})\}$, but we notice that $(f_{A_{01}}, f_{A_{10}}) \notin \mathbf{P}[R]$. In line with Corollary 1, to equate $f_{A_{01}}$ and $f_{A_{10}}$ we may add (A_{00}, A_{00}), (B, B), and (A_{11}, A_{11}) to R as illustrated below:*

$$f_{A_{01}} = 2A_{00}\,B - 3A_{01} - A_{01}\,B + 3A_{11}$$
$$f_{A_{10}} = 2A_{00}\,B - 3A_{10} - A_{10}\,B + 3A_{11}$$

While this yields $(f_{A_{01}}, f_{A_{10}}) \in \mathbf{P}[R]$, for all new pairs in R we need to check that the corresponding pairs of polynomials are in $\mathbf{P}[R]$. This is not the case yet, since $(A_{00}, A_{00}) \notin \mathbf{P}[R]$. To remedy this, we add (A_{10}, A_{01}) to R:

$$f_{A_{00}} = -4A_{00}\,B + 3A_{10} + 3A_{01}$$
$$f_{A_{00}} = -4A_{00}\,B + 3A_{10} + 3A_{01}$$

This yields the following relation

$$R = \{(A_{01}, A_{10}), (A_{10}, A_{01}), (A_{00}, A_{00}), (A_{11}, A_{11}), (B, B)\},$$

which one can verify to be a BDB. To equate $f_{A_{00}}$ with itself, in the last step, we could have added the pairs (A_{10}, A_{10}) and (A_{01}, A_{01}) instead. However, the above choice resulted in a smaller BDB.

Algorithm 1 implements a procedure that, given a query set $Query \subseteq X \times X$, discovers which query pairs can be related by some C-constrained BDB. This is carried out by constructing a C-constrained BDB R following a local exploration of the system as illustrated in Example 11.

Starting from $R = Query \setminus C$, and $\hat{R} = C$ the algorithm iteratively updates R and $\hat{R}$ maintaining the invariant $R \cap \hat{R} = \emptyset$ until R is a C-constrained BDB. At each iteration of the while-loop (lines 4–15) we check if, whenever $(x, y) \in R$, $(f_x, f_y) \in \mathbf{P}[R]$ by searching for some couplings having support disjoint from $\hat{R}$. If this is possible, then we add the support of such couplings to R; otherwise x and y cannot be related by any C-constrained BDB and we move (x, y) from R to $\hat{R}$. In line with $\mathbf{P}[R] = \mathbf{L}[\mathbf{M}[R]]$ we simplify the search of linear couplings and monomial couplings by using Q and $\hat{Q}$. The former mimics $\mathbf{M}[R]$, while the latter stores monomial pairs which have been found not possible to relate as the lifting of some C-constrained BDB. The while-loop terminates when R and Q *have not* changed with respect to the previous iteration.

Algorithm 1 Local computation of BDB

FindBDB($f\colon \mathbb{R}^X \to \mathbb{R}^X$, $Query \subseteq X \times X$, $C \subseteq X \times X$)

1 initialize $R = Query \setminus C$ and $\hat{R} = C$
2 initialize $Q = \emptyset$ and $\hat{Q} = \emptyset$
3 $Prev = (\emptyset, \emptyset, \emptyset, \emptyset)$
4 **while** $Prev \neq (R, \hat{R}, Q, \hat{Q})$
5 $\quad Prev = (R, \hat{R}, Q, \hat{Q})$
6 $\quad$ **for each** $(x, y) \in R$
7 $\quad\quad$ **if** $\exists \omega \in \Gamma_{\mathbf{L}}(f_x, f_y)$ s.t. $supp(\omega) \cap \hat{Q} = \emptyset$
8 $\quad\quad\quad Q = Q \cup supp(\omega)$
9 $\quad\quad$ **else**
10 $\quad\quad\quad$ move (x, y) from R to $\hat{R}$
11 $\quad$ **for each** $(m, n) \in Q$
12 $\quad\quad$ **if** $\exists \rho \in \Gamma_{\mathbf{M}}(m, n)$ s.t. $supp(\rho) \cap \hat{R} = \emptyset$
13 $\quad\quad\quad R = R \cup supp(\rho)$
14 $\quad\quad$ **else**
15 $\quad\quad\quad$ move (m, n) from Q to $\hat{Q}$
16 **return** R

The following result states the correctness of Algorithm 1.

Theorem 7. *Let f be a polynomial vector field over X and $Query, C \subseteq X \times X$. Then* FindBDB$(f, Query, C)$ *is terminating and returns a relation $R \subseteq X \times X$ such that:*

(i) R is a C-constrained BDB;
(ii) if $(x, y) \in Query$, then $(x, y) \in R$ iff $(x, y) \in \mathtt{gfp}(\mathcal{B}_C)$

Lines 7 and 12 in Algorithm 1 can be respectively implemented by solving a linear program. Specifically, computing a monomial coupling ω satisfying the condition in Line 7 can be done by solving:

$$\begin{aligned} v = \min_{\omega} \ & \textstyle\sum_{(m,n) \in \hat{Q}} \omega(m, n) \\ & \textstyle\sum_n \omega(m, n) = (f_x^+ + f_y^-)(m) && \forall m \in \mathbf{M} \\ & \textstyle\sum_m \omega(m, n) = (f_y^+ + f_x^-)(n) && \forall n \in \mathbf{M} \\ & \omega(m, n) \geq 0 && \forall m, n \in \mathbf{M} \end{aligned}$$

Each feasible solution corresponds to a linear coupling $\omega \in \Gamma_{\mathbf{L}}(f_x, f_y)$. In particular, the optimal value v is 0 if and only if $\omega(m, n) > 0$ implies $(m, n) \notin \hat{Q}$, i.e., $supp(\omega) \cap \hat{Q} = \emptyset$.

Analogously, one can find a monomial coupling satisfying the condition in Line 12 by solving the following linear program

$$v = \min_{\rho \in \Gamma_{\mathbf{M}}(m,n)} \textstyle\sum_{(x,y) \in \hat{R}} \rho(x, y)$$

As before, monomial couplings $\rho \in \Gamma_{\mathbf{M}}(m, n)$ are modeled by means of linear constraints, and the optimal value v equals zero if and only if $supp(\rho) \cap \hat{R} = \emptyset$.

Algorithm 2 Local computation of FDB

FINDFDB($f\colon \mathbb{R}^X \to \mathbb{R}^X$, $Query \subseteq X \times X$)

initialize $R = Query$ and $\hat{R} = \emptyset$
compute $\{J_1, .., J_\kappa\}$ as in (2) and let $F = \{J_1^T, .., J_\kappa^T\}$
$Prev = (\emptyset, \emptyset)$
while $Prev \neq (R, \hat{R})$
    $Prev = (R, \hat{R})$
    for each $(x, y) \in R$
        if $\forall f \in F. \exists \omega_f \in \Gamma_{\mathbf{L}}(f_x, f_y).\ supp(\omega_f) \cap \hat{R} = \emptyset$
            $R = R \cup \bigcup_{f \in F} supp(\omega_f)$
        else
            move (x, y) from R to $\hat{R}$
return R

Remark 4. *The above linear programs are both instances of the transportation problem [24], [25], which in turn is a well-known instance of the (uncapacitated) min-cost network flow problem. Notably, it can be solved efficiently, e.g., by employing Orlin's algorithm [26].*

This ensures that Algorithm 1 runs in polynomial time.

Theorem 8. *Algorithm 1 runs in time $\mathcal{O}(|\mathbf{M}_f|^4(k^2 + h^2))$ where $\mathbf{M}_f$ is the set of monomials occurring in f, k is the maximum number of monomials occurring in a single polynomial expression f_x in f, and h is the maximum number of variables occurring in a monomial expression in $\mathbf{M}_f$.*

Thanks to Theorems 4 and 5, we can provide a local procedure to compute FDB in a similar fashion as Algorithm 1. For this, it is worth noting that the decomposition of the Jacobian matrix as in Equation (2) gives rise to a set $\{J_1, .., J_\kappa\}$ of *linear* vector fields over X. Then, we can approach the construction of an FDB in a similar fashion as for Algorithm 1. This time, however, in line with the definition of the operator $\mathcal{F}$ one has to check, at each step, for the existence of a number linear couplings arising from the set of linear vector fields $\{J_1^T, .., J_\kappa^T\}$. This is illustrated in Algorithm 2. The following result states its correctness.

Theorem 9. *Let f be a polynomial vector field over X and $Query \subseteq X \times X$. Then* FINDFDB$(f, Query)$ *is terminating and returns a relation $R \subseteq X \times X$ such that*

(i) R is a FDB;
(ii) if $(x, y) \in Query$, then $(x, y) \in R$ iff $(x, y) \in \mathtt{gfp}(\mathcal{F})$.

Remark 5. *One may wonder whether it is possible to devise a direct local algorithm for FDE. Unfortunately, while allowing for a global approach in form of partition refinement, FDE appears to escape a direct local construction. More specifically, recall that in the case of BDE the local approach was informed by the fact a BDE pair (x, y) implied the existence of a coupling for (f_x, f_y) which, in turn, ensured the existence of a family of monomial couplings, thus giving rise to a family of further BDE pairs and so on. In contrast, an FDE pair (x, y) implies the existence of partitioning of the variables $H_1, \ldots, H_\nu \subseteq X$ satisfying the condition of Definition 2. Since the partitioning corresponds exactly to the sought FDE, it is not clear how a local approach for FDE may proceed beyond this point.*

We conclude the subsection by discussing the computational complexity of Algorithm 2. The following lemma states that the decomposition (2) can be computed in polynomial time.

Lemma 1. *Let f be a polynomial vector field over X. Then, the $J_1, \ldots, J_\kappa$ from (2) can be computed in $\mathcal{O}(|X||\mathbf{M}_f|)$ steps, where $\mathbf{M}_f$ is the set of all monomials occurring f. Moreover, it holds that $\kappa \leq |X||\mathbf{M}_f|$ and $\sum_{k=1}^{\kappa} |J_k| \leq |X||\mathbf{M}_f|$, where $|A|$ is the number of non-zero entries in a matrix A.*

Now we can provide the complexity of Algorithm 2.

Theorem 10. *Let f be a polynomial vector field over X. Algorithm 2 runs in time $\mathcal{O}(|X|^7|\mathbf{M}_f|)$ where $\mathbf{M}_f$ is the set of monomials occurring in f.*

A. Employing up-to techniques

In this section we briefly recall up-to techniques [28], and describe how to enhance the coinduction proof principle which is at the heart of Algorithms 1 and 2.

For a simpler treatment of up-to techniques that is generic on the monotone operator b defining the different notions of bisimulation encountered so far, we will call b-simulation a post-fixed point of b. In many situations the computation of a b-simulation can be significantly optimized, if instead of computing a post-fixed point of b one exhibits a relaxed invariant, that is, a relation R such that $R \subseteq b(g(R))$ for a suitable monotone function g. The function g is called a *sound up-to technique* when the following proof principle is valid

$$\frac{(x, y) \in R \quad R \subseteq b(g(R))}{(x, y) \in \mathtt{gfp}(b)}\,.$$

As clear from the above discussion, up-to techniques fit to use when one wants to check if x and y can be related by some b-simulation leaving implicit the construction of an actual b-simulation, which is replaced instead by a $(b \circ g)$-simulation.

The above notions are formalized in the following definition.

Definition 9 (Simulation up-to, soundness)**.** *Let b, g be two monotone functions. A b-simulation up-to g is a $(b \circ g)$-simulation. The function g is b-sound if $\mathtt{gfp}(b \circ g) \subseteq \mathtt{gfp}(b)$.*

Establishing the soundness of up-to techniques on a case-by-case basis can be tedious. For this reason [28] describes a framework giving sufficient conditions for proving soundness in a modular fashion, based on the notion of *compatible* function.

Definition 10. *Let b, g be two monotone functions. The function g is b-compatible if $g \circ b \subseteq b \circ g$.*

Compatible functions are also sound up-to techniques [28, Theorem 6.3.9] and, most importantly, they can be composed in several ways yielding another compatible function [28, Proposition 6.3.11]. Therefore, whenever possible, we will prove compatibility in place of soundness.

Our interest in up-to techniques has to do with the construction of a smaller witness proving which query pairs are contained in some BDB (resp. FDB). This will be implemented by computing a $\mathcal{B}_{\mathcal{C}}$-simulation up-to g for some $\mathcal{B}_{\mathcal{C}}$-sound function g given as input to the algorithm. For this, we additionally require g to be an *extensive* function, that is $R \subseteq g(R)$ for any relation R. Extensive functions will be conveniently used to reduce the size of the b-simulations up-to to be constructed in each step of the algorithm.

Given an extensive and $\mathcal{B}_{\mathcal{C}}$-sound up-to technique g, we modify FINDBDB replacing Lines 8 and 13 with

$$R = R \cup (supp(\rho) \setminus g(R)) \ , \tag{4}$$

$$Q = Q \cup (supp(\omega) \setminus \mathbf{M}[g(R) \setminus C]) \ , \tag{5}$$

respectively. Note that, thanks to the fact that g is extensive, instead of adding to R (resp. Q) the entire support of the coupling ρ (resp. ω), we add only those pairs which cannot already be safely inferred from the current value of R by exploiting the up-to proof principle. This makes the algorithm more conservative in expanding the size of R (resp. Q), thus leading to a significant speed-up in performance as demonstrated in the next section.

The following result states that the above explained modification of Algorithm 1 returns a $\mathcal{B}_{\mathcal{C}}$-simulations up-to g witnessing which query pairs are contained in some C-constrained BDB.

Theorem 11. *Let f be a polynomial vector field over X, g an* extensive *and* monotone *$\mathcal{B}_{\mathcal{C}}$-sound up-to technique, and $Query, C \subseteq X \times X$. Then,* FINDBDB$(f, Query, C, g)$ *is terminating and returns a relation $R \subseteq X \times X$ such that:*

(i) R is a $\mathcal{B}_{\mathcal{C}}$-simulation up-to g;
(ii) if $(x, y) \in Query$, then $(x, y) \in R$ iff $(x, y) \in \mathtt{gfp}(\mathcal{B}_C)$.

The following lemma states that reflexive, symmetric, transitive, and equivalence closures are $\mathcal{B}$-compatible up-to techniques, which can be safely used as input in Algorithm 1.

Lemma 2. *r, s, t, and e are $\mathcal{B}$-compatible up-to techniques.*

Remark 6. *Notably, the proof of Lemma 2 relies on the fact that $g \circ \mathcal{B} \circ g = \mathcal{B} \circ g$ for all $g \in \{r, s, t, e\}$. This also implies that for any $g \in \{r, s, t, e\}$ and $R \subseteq X \times X$ $\mathcal{B}$-simulation up-to g, one can retrieve the corresponding $\mathcal{B}$-simulation as $g(R)$. Indeed, $g(R) \subseteq g(\mathcal{B}(g(R))) = \mathcal{B}(g(R))$ where the first inclusion holds true because g is extensive.*

Example 12. *Consider the polynomial vector field f of Eq.* (1). *The call* FINDBDB$(f, \{(A_{01}, A_{10})\}, \emptyset, e)$ *terminates after the first iteration returning the $(\mathcal{B} \circ e)$-simulation $\{(A_{01}, A_{10})\}$. Notably, this means that the algorithm did not need to process any polynomial other than $f_{A_{01}}$ and $f_{A_{10}}$. Note also that $e(\{(A_{01}, A_{10})\})$ is the BDE from Example 2.*

An immediate consequence of Lemma 2 and Definition 7 is the $\mathcal{F}$-compatibility of reflexive, symmetric, transitive, and equivalence closures which can be employed in Algorithm 2.

Corollary 2. *r, s, t, and e are $\mathcal{F}$-compatible up-to techniques.*

Without constraints we can employ a number of up-to techniques. Unfortunately, the up-to techniques considered above turn out not to be $\mathcal{B}_{\mathcal{C}}$-sound for generic choices of the constraints $C \subseteq X \times X$, as illustrated in the following example.

Example 13. *Consider the following linear vector field f over $X = \{x, x', x'', y, y', y'', z, z', z''\}$ defined by*

$$\begin{array}{lll} f_x = 0\,, & f_y = x\,, & f_z = y\,, \\ f_{x'} = 0\,, & f_{y'} = x'\,, & f_{z'} = y'\,, \\ f_{x''} = 0\,, & f_{y''} = x''\,, & f_{z''} = y''\,. \end{array}$$

Fix $C = \{(x, x'')\}$. We have that neither s nor t are $\mathcal{B}_C$-sound. Indeed $(z, z'') \notin \mathtt{gfp}(\mathcal{B}_C)$ but both $\mathtt{gfp}(\mathcal{B}_C \circ s)$ and $\mathtt{gfp}(\mathcal{B}_C \circ t)$ contain (z, z''). This is also a counterexample for the $\mathcal{B}_C$-soundness of e.

Despite this fact, under sensible conditions on C, reflexive and symmetric closures are still sound up-to techniques.

Lemma 3. *If $C \cap \mathtt{id} = \emptyset$, then r is $\mathcal{B}_C$-compatible.*

The following result states that when the relation $C \subseteq X \times X$ symmetric, then the symmetric closure is $\mathcal{B}_{\mathcal{C}}$-compatible.

Lemma 4. *Let $C \subseteq X \times X$. Then, s is $\mathcal{B}_{s(C)}$-compatible.*

Remark 7. *Up-to techniques are particularly useful in discovering if the backward equivalence of two variables x and y is dependent on the equivalence of some other (pairwise-distinct) pairs of variables $C \subseteq (X \times X)$. Indeed, this can be done by checking if* FINDBDB$(f, \{(x, y)\}, s(C), r \circ s) = \emptyset$.

VI. APPLICATIONS

We present applications which show how our local algorithm equipped with up-to-techniques and constraints is complementary to one based on partition refinement for the computation of differential equivalences. Section VI-A discusses how the local algorithm can be used to find relations between ODE variables that are independent of the choice of model parameters. Section VI-B shows that the up-to techniques can speed up BDB computations by several orders of magnitude. Section VI-C discusses how the use of constraints allows one to compute BDBs satisfying domain-specific properties not supported by partition refinement. Section VI-D shows how the local algorithm disproves equivalence by exploring only a small part of the model, thus outperforming the partition refinement approach. Section VI-E computes FDBs of epidemiological models on graphs.

Results are based on a prototype implementation of Algorithm 1; the computations of BDE and FDE were performed with the tool ERODE [54], run on a common laptop with 3.1 GHz Dual-Core Intel Core i5 and 8GB RAM.

A. Parameter-independent bisimulations

Building mechanistic models in biology is hindered by the difficulty in observing all biochemical interactions [55], which may lead to uncertainties in the choice of the parameters. When analyzing a model built under these conditions, it is helpful to discover properties that depend only on the *structure* and

Model			*Base algorithm*			*Up-to reflexivity*			*Up-to symmetry*			*Up-to transitivity*			*Up-to equivalence*		
n	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{R}\rvert$	*Time (s)*	*TP*	$\lvert R\rvert$	*Time (s)*	*TP*	$\lvert R\rvert$	*Time (s)*	*TP*	$\lvert R\rvert$	*Time (s)*	*TP*	$\lvert R\rvert$	*Time (s)*	*TP*	$\lvert R\rvert$
2	1.80E1	4.80E1	**8.00E-3**	9.46E2	3.00E1	4.00E-3	4.32E2	1.40E1	7.50E-3	8.39E2	2.50E1	8.00E-3	9.26E2	2.90E1	**3.00E-3**	3.23E2	9.00E0
3	6.60E1	2.88E2	**1.00E-1**	2.68E4	2.34E2	6.50E-2	1.75E4	1.71E2	7.40E-2	1.96E4	1.53E2	3.70E-2	9.10E3	1.57E2	**1.50E-2**	2.94E3	5.10E1
4	2.58E2	1.54E3	**6.70E-1**	3.09E5	1.77E3	5.20E-1	2.45E5	1.52E3	4.90E-1	2.01E5	1.02E3	1.70E-1	6.69E4	7.63E2	**3.90E-2**	1.99E4	2.34E2
5	1.03E3	7.68E3	**6.90E0**	3.23E6	1.34E4	5.40E0	2.85E6	1.24E4	4.23E0	1.95E6	7.22E3	1.91E0	4.19E5	3.45E3	**4.10E-1**	1.15E5	9.89E2
6	4.10E3	3.69E4	**9.30E1**	3.16E7	9.95E4	6.80E1	2.95E7	9.54E4	5.91E1	1.82E7	5.18E4	2.71E1	2.39E6	1.50E4	**4.00E0**	6.08E5	4.04E3

TABLE I: Speed-up of BDB computation with up-to techniques. We highlight in bold the best and worst runtimes.

not on the specific choice of parameter values. Even if the parameters were precise, discovering structural properties can be beneficial, for instance, to conduct sensitivity analysis for predictive purposes across different parameterizations.

With Example 1, we discuss how to compute bisimulations that hold independently of the values of some kinetic rate parameters. For this, we consider a variant of the model where the parameters are explicitly treated as ODEs. This is done by building an *extended* model where the kinetic rate parameter of each reaction is treated as a further species in the CRN (we call them *parameter-species* to ease the presentation), and every modified reaction occurs with fixed rate 1. For each reaction *Ri*, we name the parameter-species of the forward and reverse part as k_i and $\overline{k}_i$, respectively. We obtain the extended CRN:

$$\textit{R1}\colon\ A_{00} + B + k_1 \xrightarrow{1} A_{10} + k_1 \qquad A_{10} + \overline{k}_1 \xrightarrow{1} A_{00} + B + \overline{k}_1$$
$$\textit{R2}\colon\ A_{00} + B + k_2 \xrightarrow{1} A_{01} + k_2 \qquad A_{01} + \overline{k}_2 \xrightarrow{1} A_{00} + B + \overline{k}_2$$
$$\textit{R3}\colon\ A_{10} + B + k_3 \xrightarrow{1} A_{11} + k_3 \qquad A_{11} + \overline{k}_3 \xrightarrow{1} A_{10} + B + \overline{k}_3$$
$$\textit{R4}\colon\ A_{01} + B + k_4 \xrightarrow{1} A_{11} + k_4 \qquad A_{11} + \overline{k}_4 \xrightarrow{1} A_{01} + B + \overline{k}_4$$

By the law of mass action, the derivatives associated with the parameter-species in the extended CRN are 0: thus the ODEs of the original CRN and those of the extended one coincide when the values of the original parameters are used as the initial conditions of the corresponding parameter-species.

On this extended model one could use the partition refinement algorithm from [17], starting from the initial partition

$$\{\{A_{01}, A_{10}, A_{11}, B\}, \{k_1, k_2\}, \{k_3, k_4\}, \{\overline{k}_1, \overline{k}_2, \overline{k}_3, \overline{k}_4\}\}.$$

This equates parameters that have the same values in the original model. The coarsest refinement by BDE equates species A_{01} and A_{10}, as in the original model, without splitting the initial blocks of parameter-species. Thus, A_{01} and A_{10} are backward equivalent *as long as all parameters of the reverse reactions* $\overline{k}_i$ *are equal*.

Checking if the equivalence carries over under fewer constraints for the kinetic rate parameters amounts to asking whether there exists a BDE that refines the blocks of parameter-species. With a partition refinement algorithm, this may involve checking exponentially many initial partitions (i.e., the ones singling out each reverse rate, the ones with 2 blocks of size 2 for the reverse rates, and so on). Instead, by initializing our local algorithm with the query $\{(A_{01}, A_{10})\}$, we obtain a BDB that only relates $\overline{k}_1$ with $\overline{k}_2$ and $\overline{k}_3$ with $\overline{k}_4$, respectively, as well as k_1 with k_2, and k_3 with k_4. This gives such a desired refinement: the BDB requires one equality for the reverse rates of reactions *R1-R2*, and one for the reverse rates of reactions *R3-R4*.

The above BDB query required no longer than 70ms and the solution of 103 transportation problems.

B. Computational speed-up using up-to techniques

To show how up-to techniques can speed up the computation of bisimulations, we consider a benchmark model featuring complex formation by multisite phosphorylation from [56, Supplementary Note 7]; it has been used in [57], [54] to assess the scalability of the partition refinement algorithms for BDE and FDE. Example 1 can be seen as a simpler version.

Here we consider a family of models of varying size obtained by changing the number of binding sites, n, from 2 to 7. For each n, the query encoded the question whether there exists a BDB relating all species representing complexes where all n binding sites are unphosphorylated; this leads to a query set Q of size $|Q| = n^2$ for each n.

Table I shows the runtimes for BDB computations without up-to techniques and with four different up-to techniques, namely, reflexive, symmetric, transitivive, and equivalence closures. For each case we measured the runtime of the local algorithm (column *Time*), the number of solved transportation problems (column *TP*), and the number of pairs in the relations (column $|R|$). For each n, the table also shows the number of species and reactions of the resulting model (columns $|\mathcal{S}|$ and $|\mathcal{R}|$, respectively). The use of up-to techniques leads to smaller relations, fewer transportation problems to be solved, and therefore lower runtimes. Improvements can be found already for $n = 2$, where, e.g., the runtime for the up-to equivalence case is about 37% of the base algorithm case. For larger models, runtimes are up to one order of magnitude smaller.

C. Input-preserving BDB for Boolean networks

Boolean networks (BN), proposed in 1969 [58], [59], are an established model of biological systems (e.g., [60], [61]). A BN is given by a set of Boolean variables and a Boolean update function associated with each variable. Given an initial state, i.e., an assignment for each variable, a discrete-time dynamics is obtained by setting the new state of each variable as the evaluation of its associated function. Figure 2 is an excerpt of the graphical representation of the largest BN from the GinSim repository [62], taken from [63]. Nodes denote variables related to biochemical species (four are highlighted in cyan for the forecoming discussion), while directed edges are drawn if the source node appears in the Boolean update function of the target.

In some cases, a Boolean representation of the state may be too crude an approximation, e.g., one would like to model the level of activity of a gene continuously in the interval $[0, 1]$. To cope with this, ODE interpretations of BNs have been proposed in the literature (e.g. [64]), essentially by interpolating the Boolean function with a real one that agrees with the Boolean output when evaluated with 0/1 inputs. *Odefy* provides such an approximation with polynomial ODEs [65]. The ODE encoding of the BN in Figure 2 has 129 ODEs with derivatives containing 554 monomials of degrees ranging from 1 to 6.

Motivated by the need to reduce complexity of BNs for their analysis [66], [67], [68], [69], [70], BDE has been used for the reduction of these ODEs in the past [17]. Here we show how the use of constraints with our local algorithm can provide informative relations that are not directly obtainable using partition refinement.

In applications to gene regulatory networks, certain variables are usually designated as *input* variables, typically representing species at the top of a signalling pathway (and modeled with the identity update function). The modeler then conducts several analyses by varying their initial value and observing the values of *output* variables of interest (e.g., gene expression levels at the bottom of the pathway). In Figure 2, $CD6$ and $TLR5$ are two inputs (out of 16 not shown in the excerpt).

The maximal BDE, obtained by partition refinement using the initial trivial partition with one block only, contains an equivalence class with 20 variables that also includes the four highlighted in Figure 2, and a number of input variables. This suggests a biologically relevant pattern of co-expression whereby a number of variables of the network simultaneously respond with the same output to the same input (e.g., [18]). However, from the maximal BDE one cannot conclude that this pattern does not depend on other network conditions, because the larger equivalence class in which it is found imposes the pre-condition that *all* equivalent species start from the same initial condition, including the other inputs in the class.

Specifically, one would wish to find that this pattern holds independently of the values of the input variables. With partition refinement, this can be done by keeping the ODE variables corresponding to the inputs as distinct singleton blocks of the initial partition, as done, e.g., in [17]. However, since inputs are separated from all other variables, the coarsest BDE refinement of the as-constructed partition cannot yield the relation among the highlighted variables in Figure 2 (which, in fact, are separated into 3 distinct blocks).

We perform three experiments showing how BDB can be successfully used in this context. In order to check whether we can relate the four highlighted species without relating any inputs, or any other species in general, we run our algorithm with query the Cartesian product of the 4 highlighted species, and with constraints the Cartesian product of the 16 input species minus the identity (240 pairs). Our prototype computed a BDB with 16 pairs relating all highlighted species. The BDE obtained as the equivalence closure of such BDB contains a block consisting of the highlighted species, and 1 singleton block for each other species, proving that no further

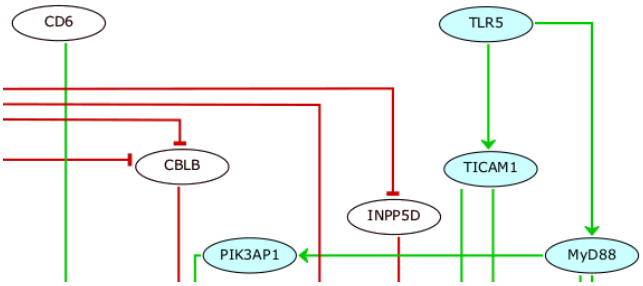

Fig. 2: Excerpt of the BN TCR-TLR5 [63] using GinSim [62].

assumptions are required to relate the highlighted species.

As a second example, we use as query the Cartesian product of the 3 non-input highlighted species, obtaining a BDB with 12 pairs. The BDB again relates all highlighted species, and its equivalence closure gives the same BDE as in the previous experiment. This suggests that in order to prove the query we need to relate all highlighted species.

In order to confirm this, we perform a third experiment where we explicitly add constraints for preventing $TLR5$ to be related with the other highlighted species. We get a BDB with 6 pairs only, relating only $MyD88$ with $TICAM1$. This confirms that it is possible to relate $PIK3AP1$ with $TICAM1$ and $MyD88$ only if $TLR5$ is also related to them.

All experiments required less than 0.5s and the solution of less than 85 transportation problems

We note that the first experiment could be replicated with partition refinement using as initial partition the computed BDE, while the second and third one could not because partition refinement can only decrease the size of relations while our local approach adds the necessary pairs.

D. Local computation of CTMC lumpability

In this section we study the effectiveness of our local algorithm with respect to partition refinement in proving that two CTMCs are not equivalent. We show this on a case study of comparison between a model of a mutual-inhibition (MI) cell switch, a mechanism present in many biological networks (e.g. [71], [72]), and a simpler switch, which has been shown to correspond to the approximate majority (AM) algorithm from population protocols [73]. These two systems can be modeled as CRNs as follows:

$$\begin{array}{ll|l}
y_0 + z_0 \xrightarrow{1} z_0 + y_1 & y_1 + z_0 \xrightarrow{1} z_0 + y_2 & x_0 + x_2 \xrightarrow{1} x_2 + x_1 \\
y_2 + y_0 \xrightarrow{1} y_0 + y_1 & y_1 + y_0 \xrightarrow{1} y_0 + y_0 & x_1 + x_2 \xrightarrow{1} x_2 + x_2 \\
z_2 + z_0 \xrightarrow{1} z_0 + z_1 & z_1 + z_0 \xrightarrow{1} z_0 + z_0 & x_2 + x_0 \xrightarrow{1} x_0 + x_1 \\
z_0 + y_0 \xrightarrow{1} y_0 + z_1 & z_1 + y_0 \xrightarrow{1} y_0 + z_2 & x_1 + x_0 \xrightarrow{1} x_0 + x_0
\end{array}$$

where species y_0, y_1, y_2, z_0, z_1 and z_2 refer to MI and x_0, x_1, x_2 refer to AM. Being able to compare such kinds of networks is relevant in evolutionary biology to assess, for example, if complex cellular mechanisms can be related to less robust, more primordial variants (e.g., [32], [74]).

It has been established that the species of MI and AM can be related by a BDE when the CRNs are interpreted with ODE

Initial populations		*Union CTMC*		*BDE*	*BDB*
x_0, y_0, z_2	x_2, y_2, z_0	*States*	*Trans.*	*Time (s)*	*Time (s)*
2	1	93	276	1.60E-3	1.00E-3
4	2	762	3 504	7.00E-3	1.00E-3
6	3	2 979	16 164	2.60E-2	1.00E-3
8	4	8 202	48 624	8.90E-2	2.00E-3
10	5	18 375	115 140	2.89E-1	2.00E-3
12	6	35 928	233 856	9.10E-1	2.00E-3
14	8	75 922	511 984	1.18E0	2.00E-3
16	10	142 532	985 504	3.10E0	3.00E-3
18	12	245 550	1 729 680	5.90E0	3.00E-3
20	14	396 304	2 832 064	1.12E1	4.50E-3
22	16	607 658	4 392 496	1.70E1	6.00E-3

TABLE II: Local vs global BDE checks to relate the CTMCs of MI and AM (initial populations not shown are set to 0).

mass-action semantics [75], [76]. Such interpretation can be seen as a deterministic limit description of a CTMC describing discrete molecular interactions, when the number of molecules goes to infinity [77]. In fact, under certain physical conditions, from first principles it is well-known that the ground-truth behavior is given by a CTMC [77]. Starting from an initial population represented by a vector where each component models the amount of elements for each species, the CTMC is generated by exhaustively applying every reaction, generating a new CTMC state where the reaction's reagents are replaced by the reaction's products, respectively referring to the species appearing in the left- and right-hand-side of the reaction.

To see whether MI and AM can be related also under this CTMC interpretation, i.e., under the assumption of finite populations for the species, it is possible to initialize the local algorithm with the pair consisting of the respective initial states of the two CTMCs. Similarly, the partition refinement algorithm can be initialized with a partition with a block containing the two initial states.

The answer to the above comparison question is negative. Table II shows the runtimes of the local and the (global) partition-refinement algorithms to disprove the relation between the initial states of the networks' CTMCs with varying initial populations as specified by the first two columns (species not mentioned are set to 0). Increasing initial population counts leads to a combinatorial explosion of the underlying CTMC: the third and fourth columns provide the number of states and transitions, respectively, of the CTMC formed by the disjoint union of the CTMCs of both networks. While the runtime for partition refinement (column *BDE*) grows with the CTMC size (as expected by the computational complexity of CTMC lumping algorithms [15]), the local algorithm always yields an empty relation by analyzing only one transportation problem within a few milliseconds (column *BDB*).

E. Local FDB computation on epidemiological models

We study how FDB can be applied on epidemiological SIR models evolving on graphs (e.g., [78]). In this model, each vertex describes an individual that: can be susceptible (S) to an infection with a rate proportional to the number of infected (I) neighbors (and parameter β); that can recover (R) from the infection to gain immunity (with parameter γ). Fig. 3(A) shows the model for a star topology.

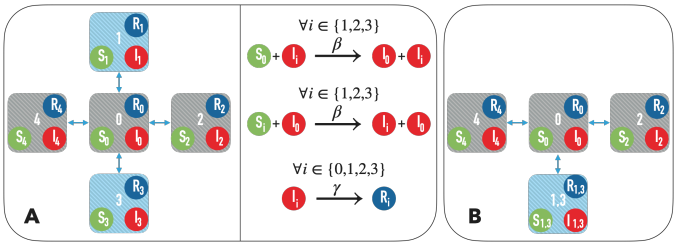

Fig. 3: (A) SIR on a star network where species in the central node interact with those in the peripheral ones. (B) FDB reduction of (A) relating pairwise the variables in nodes 1 and 3 de facto obtaining a coarse-grained network.

Since nodes in the graph may represent locations, a coarse-grained representation may be helpful to reason about the system in terms of larger areas. For example, the cumulative dynamics of nodes 1 and 3 could be obtained by running the FDB local algorithm with the query $\{S_1, S_3\} \times \{S_1, S_3\}$. By solving 295 transportation problems in about 10 ms the output FDB has 21 pairs: a reflexive pair for each of the 15 species, and the 6 pairs $(S_1, S_3), (S_3, S_1), (I_1, I_3), (I_3, I_1), (R_1, R_3), (R_3, R_1)$. Intuitively, as depicted in Fig. 3(B), it does not distinguish among nodes 1 and 3, while leaving unaltered the rest of the graph.

This problem can be semi-automatically addressed using partition refinement, but it requires to perform trial-and-error for the appropriate initial partition to use. E.g., for an initial partition where the only non-singleton block is $\{S_1, S_3\}$, the coarsest FDE refinement is $\{\{S_0\}, \{I_0\}, \{S_1, S_3\}, \{S_2, S_4\}, \{I_0, I_1, I_2, I_3\}, \{R_0, R_1, R_2, R_3\}\}$, but it aggregates too much. The desired reduction is obtained for initial partition $\{S_1, S_3\}$, $\{I_1, I_3\}$, $\{R_1, R_3\}$, $\{S_0, S_2, S_4, I_0, I_2, I_4, R_0\}$, $\{R_2\}$, $\{R_4\}$.

VII. Conclusions and Future Work

We have presented an algorithm for computing bisimulations over variables of a system of polynomial differential equations using a local approach, which complements available global methods based on partition refinements. Crucial to our approach is the introduction of a novel coupling method for reasoning about equivalences over polynomials. Our algorithm computes bisimulations which relate ODE solutions *exactly*. Given that probabilistic couplings have proved instrumental for the development of *bisimilarity distances* [79], [80], a natural question is whether our approach can be lifted to metric spaces for reasoning about approximate bisimulations of polynomial differential equations. We intend to tackle this in future work.

Acknowledgments: This work was supported by the Poul Due Jensen Foundation (grant 883901), ERC Advanced Grant LASSO, and Villum Investigator Grant S4OS.

References

[1] L. Bortolussi and J. Hillston, "Fluid Model Checking," in *CONCUR*, 2012, pp. 333–347.

[2] J. Hillston, "Fluid flow approximation of PEPA models," in *QEST*, Sep. 2005, pp. 33–43.

[3] F. Ciocchetta and J. Hillston, "Bio-PEPA: A framework for the modelling and analysis of biological systems," *Theoretical Computer Science*, vol. 410, no. 33-34, pp. 3065–3084, 2009.

[4] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin, "Comparing chemical reaction networks: A categorical and algorithmic perspective," in *Proceedings of the Thirty-First Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2016.

[5] ——, "Symbolic computation of differential equivalences," in *POPL*, 2016, pp. 137–150.

[6] M. Boreale, "Algorithms for exact and approximate linear abstractions of polynomial continuous systems," in *HSCC*, M. Prandini and J. V. Deshmukh, Eds. ACM, 2018, pp. 207–216.

[7] ——, "Complete algorithms for algebraic strongest postconditions and weakest preconditions in polynomial odes," *Sci. Comput. Program.*, vol. 193, p. 102441, 2020.

[8] G. J. Pappas, G. Lafferriere, and S. Sastry, "Hierarchically consistent control systems," *IEEE Trans. Automat. Contr.*, vol. 45, no. 6, pp. 1144–1160, 2000.

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

[10] 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*, vol. 57, no. 6, pp. 1531–1556, 1997.

[11] M. S. Okino and M. L. Mavrovouniotis, "Simplification of mathematical models of chemical reaction systems," *Chemical Reviews*, vol. 2, no. 98, pp. 391–408, 1998.

[12] K. G. Larsen and A. Skou, "Bisimulation through probabilistic testing," *Inf. Comput.*, vol. 94, no. 1, pp. 1–28, 1991.

[13] J. Sproston and S. Donatelli, "Backward bisimulation in Markov chain model checking," *IEEE Trans. Software Eng.*, vol. 32, no. 8, pp. 531–546, 2006.

[14] P. Buchholz, "Exact and ordinary lumpability in finite Markov chains," *Journal of Applied Probability*, vol. 31, no. 1, pp. 59–75, 1994.

[15] S. Derisavi, H. Hermanns, and W. H. Sanders, "Optimal state-space lumping in markov chains," *Inf. Process. Lett.*, vol. 87, no. 6, pp. 309–315, 2003.

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

[17] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin, "Maximal aggregation of polynomial dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 114, no. 38, pp. 10 029–10 034, 2017.

[18] R. Zhang, M. V. Shah, J. Yang, S. B. Nyland, X. Liu, J. K. Yun, R. Albert, and J. Loughran, Thomas P, "Network model of survival signaling in large granular lymphocyte leukemia," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105, no. 42, pp. 16 308–16 313, 10 2008.

[19] B. Zhang and S. Horvath, "A General Framework for Weighted Gene Co-Expression Network Analysis," *Statistical Applications in Genetics and Molecular Biology*, vol. 4, no. 1, 2005.

[20] E. O. Voit, "Biochemical systems theory: A review," *ISRN Biomathematics*, vol. 2013, p. 53, 2013.

[21] J. Liu, N. Zhan, H. Zhao, and L. Zou, "Abstraction of elementary hybrid systems by variable transformation," in *FM*, vol. 9109, 2015, pp. 360–377.

[22] T. Lindvall, *Lectures on the Coupling Method*, ser. Wiley Series in Probability and Mathematical Statistics. New York: John Wiley, 1992.

[23] H. Thorisson, "Coupling methods in probability theory," *Scand. J. Stat.*, vol. 22, pp. 159–182, 1995.

[24] G. B. Dantzig, "Application of the Simplex method to a transportation problem," in *Activity analysis of production and allocation*, T. Koopmans, Ed. J. Wiley, New York, 1951, pp. 359–373.

[25] L. R. Ford and D. R. Fulkerson, "Solving the Transportation Problem," *Management Science*, vol. 3, no. 1, pp. 24–32, 1956.

[26] J. Orlin, "A Faster Strongly Polynomial Minimum Cost Flow Algorithm," in *STOC*, 1988, pp. 377–387.

[27] R. Milner, *Communication and Concurrency*. USA: Prentice-Hall, Inc., 1989.

[28] D. Pous and D. Sangiorgi, "Enhancements of the bisimulation proof method," in *Advanced Topics in Bisimulation and Coinduction*, ser. Cambridge tracts in theoretical computer science. Cambridge University Press, 2012, vol. 52, pp. 233–289.

[29] F. Bonchi, P. Ganty, R. Giacobazzi, and D. Pavlovic, "Sound up-to techniques and complete abstract domains," in *LICS*, 2018, pp. 175–184.

[30] F. Bonchi and D. Pous, "Checking NFA equivalence with bisimulations up to congruence," in *POPL*. ACM, 2013, pp. 457–468.

[31] F. Bonchi, B. König, and S. Küpper, "Up-to techniques for weighted systems," in *TACAS*, 2017, pp. 535–552.

[32] L. Cardelli, "Morphisms of reaction networks that couple structure to function," *BMC Systems Biology*, vol. 8, no. 1, p. 84, 2014.

[33] M. Boreale, "Algebra, coalgebra, and minimization in polynomial differential equations," *Log. Methods Comput. Sci.*, vol. 15, no. 1, 2019.

[34] B. Buchberger, "A theoretical basis for the reduction of polynomials to canonical forms," *SIGSAM Bull.*, vol. 10, no. 3, pp. 19–29, 1976.

[35] K. Ghorbal and A. Platzer, "Characterizing algebraic invariants by differential radical invariants," in *TACAS*, E. Ábrahám and K. Havelund, Eds., vol. 8413. Springer, 2014, pp. 279–294.

[36] S. Sankaranarayanan, H. Sipma, and Z. Manna, "Non-linear loop invariant generation using Gröbner bases," in *POPL*, 2004, pp. 318–329.

[37] A. Platzer, "Logics of dynamical systems," in *LICS*. IEEE Computer Society, 2012, pp. 13–24.

[38] B. Jonsson and K. G. Larsen, "Specification and refinement of probabilistic processes," in *LICS*, 1991, pp. 266–277.

[39] C. Baier, "Polynomial time algorithms for testing probabilistic bisimulation and simulation," in *CAV*, ser. Lecture Notes in Computer Science, R. Alur and T. A. Henzinger, Eds., vol. 1102. Springer, 1996, pp. 50–61.

[40] D. Chen, F. van Breugel, and J. Worrell, "On the complexity of computing probabilistic bisimilarity," in *FOSSACS*, 2012, pp. 437–451.

[41] G. Bacci, G. Bacci, K. G. Larsen, and R. Mardare, "On-the-fly exact computation of bisimilarity distances," in *TACAS*, 2013.

[42] G. Barthe, T. Espitau, B. Grégoire, J. Hsu, L. Stefanesco, and P. Strub, "Relational reasoning via probabilistic coupling," in *LPAR*, ser. Lecture Notes in Computer Science, vol. 9450. Springer, 2015, pp. 387–401.

[43] G. Barthe, B. Grégoire, J. Hsu, and P. Strub, "Coupling proofs are probabilistic product programs," in *POPL*, 2017, pp. 161–174.

[44] A. Aguirre, G. Barthe, L. Birkedal, A. Bizjak, M. Gaboardi, and D. Garg, "Relational reasoning for Markov chains in a probabilistic guarded lambda calculus," in *ESOP*, vol. 10801, 2018, pp. 214–241.

[45] G. Barthe, T. Espitau, B. Grégoire, J. Hsu, and P. Strub, "Proving expected sensitivity of probabilistic programs," *Proc. ACM Program. Lang.*, vol. 2, no. POPL, pp. 57:1–57:29, 2018.

[46] G. Barthe, N. Fong, M. Gaboardi, B. Grégoire, J. Hsu, and P. Strub, "Advanced probabilistic couplings for differential privacy," in *CCS*. ACM, 2016, pp. 55–67.

[47] B. Balle, G. Barthe, and M. Gaboardi, "Privacy amplification by subsampling: Tight analyses via couplings and divergences," in *NeurIPS*, 2018, pp. 6280–6290.

[48] G. Barthe, T. Espitau, B. Grégoire, J. Hsu, and P. Strub, "Proving uniformity and independence by self-composition and coupling," in *LPAR*, vol. 46, 2017, pp. 385–403.

[49] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin, "Forward and backward bisimulations for chemical reaction networks," in *CONCUR*, 2015, pp. 226–239.

[50] V. Strassen, "The existence of probability measures with given marginals," *Annals of Mathematical Statistics*, vol. 36, no. 2, pp. 423–439, 04 1965.

[51] G. Bacci, G. Bacci, K. G. Larsen, M. Tribastone, and A. V. Max Tschaikowski and, "Efficient local computation of differential bisimulations via coupling and up-to methods," *CoRR*, vol. abs/2104.13160, 2021. [Online]. Available: https://arxiv.org/abs/2104.13160

[52] G. Li and H. Rabitz, "A general analysis of exact lumping in chemical kinetics," *Chemical Engineering Science*, vol. 44, no. 6, pp. 1413–1430, 1989.

[53] A. Ovchinnikov, I. C. Pérez-Verona, G. Pogudin, and M. Tribastone, "CLUE: exact maximal reduction of kinetic models by constrained lumping of differential equations," *CoRR*, vol. abs/2004.11961, 2020. [Online]. Available: https://arxiv.org/abs/2004.11961

[54] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin, "ERODE: A tool for the evaluation and reduction of ordinary differential equations," in *TACAS*, 2017.

[55] A. Babtie and M. Stumpf, "How to deal with parameters for whole-cell modelling," *Journal of The Royal Society Interface*, vol. 14, no. 133, p. 20170237, 2017.

[56] M. W. Sneddon, J. R. Faeder, and T. Emonet, "Efficient modeling, simulation and coarse-graining of biological complexity with NFsim," *Nature Methods*, vol. 8, no. 2, pp. 177–183, 2011.

[57] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin, "Efficient Syntax-Driven Lumping of Differential Equations," in *TACAS*, 2016, pp. 93–111.
[58] S. Kauffman, "Homeostasis and differentiation in random genetic control networks," *Nature*, vol. 224, no. 5215, pp. 177–178, 1969.
[59] R. Thomas, "Boolean formalization of genetic control circuits," *Journal of Theoretical Biology*, vol. 42, no. 3, pp. 563–585, 1973.
[60] W. Abou-Jaoudé, P. Traynard, P. T. Monteiro, J. Saez-Rodriguez, T. Helikar, D. Thieffry, and C. Chaouiya, "Logical modeling and dynamical analysis of cellular networks," *Frontiers in Genetics*, vol. 7, pp. 94–94, 05 2016.
[61] P. Bloomingdale, V. A. Nguyen, J. Niu, and D. E. Mager, "Boolean network modeling in systems pharmacology," *Journal of pharmacokinetics and pharmacodynamics*, vol. 45, no. 1, pp. 159–180, 02 2018.
[62] A. Naldi, D. Berenguier, A. Fauré, F. Lopez, D. Thieffry, and C. Chaouiya, "Logical modelling of regulatory networks with GINsim 2.3," *Biosystems*, vol. 97, no. 2, pp. 134 – 139, 2009.
[63] O. Rodríguez-Jorge, L. A. Kempis-Calanis, W. Abou-Jaoudé, D. Y. Gutiérrez-Reyna, C. Hernandez, O. Ramirez-Pliego, M. Thomas-Chollier, S. Spicuglia, M. A. Santana, and D. Thieffry, "Cooperation between T cell receptor and Toll-like receptor 5 signaling for CD4+ T cell activation," *Science signaling*, vol. 12, no. 577, 2019.
[64] D. M. Wittmann, J. Krumsiek, J. Saez-Rodriguez, D. A. Lauffenburger, S. Klamt, and F. J. Theis, "Transforming boolean models to continuous models: methodology and application to T-cell receptor signaling," *BMC Systems Biology*, vol. 3, no. 1, p. 98, 2009.
[65] J. Krumsiek, S. Pölsterl, D. M. Wittmann, and F. J. Theis, "Odefy - from discrete to continuous models," *BMC Bioinformatics*, vol. 11, no. 1, p. 233, 2010.
[66] S. Klamt, J. Saez-Rodriguez, J. A. Lindquist, L. Simeoni, and E. D. Gilles, "A methodology for the structural and functional analysis of signaling and regulatory networks," *BMC Bioinformatics*, vol. 7, no. 1, p. 56, 2006.
[67] A. Veliz-Cuba, B. Aguilar, F. Hinkelmann, and R. Laubenbacher, "Steady state analysis of boolean molecular network models via model reduction and computational algebra," *BMC Bioinformatics*, vol. 15, no. 1, p. 221, 2014.
[68] A. Naldi, E. Remy, D. Thieffry, and C. Chaouiya, "Dynamically consistent reduction of logical regulatory graphs," *Theoretical Computer Science*, vol. 412, no. 21, pp. 2207–2218, 2011.
[69] A. Veliz-Cuba, "Reduction of boolean network models," *Journal of theoretical biology*, vol. 289, pp. 167–172, 2011.
[70] A. Saadatpour, R. Albert, and T. C. Reluga, "A reduction method for boolean network models proven to conserve attractors," *SIAM Journal on Applied Dynamical Systems*, vol. 12, no. 4, pp. 1997–2011, 2013.
[71] A. Verdugo, P. K. Vinod, J. J. Tyson, and B. Novak, "Molecular mechanisms creating bistable switches at cell cycle transitions," *Open Biology*, vol. 3, no. 3, p. 120179, 03 2013.
[72] T. S. Gardner, C. R. Cantor, and J. J. Collins, "Construction of a genetic toggle switch in Escherichia coli," *Nature*, vol. 403, no. 6767, pp. 339–342, 01 2000.
[73] L. Cardelli and A. Csikász-Nagy, "The cell cycle switch computes approximate majority," *Scientific Reports*, vol. 2, pp. 656 EP –, 09 2012.
[74] S. Gay, S. Soliman, and F. Fages, "A graphical method for reducing and relating models in systems biology," *Bioinformatics*, vol. 26, no. 18, pp. i575–i581, 2010.
[75] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin, "Comparing chemical reaction networks: A categorical and algorithmic perspective," *Theor. Comput. Sci.*, vol. 765, pp. 47–66, 2019.
[76] ——, "Symbolic computation of differential equivalences," *Theor. Comput. Sci.*, vol. 777, pp. 132–154, 2019.
[77] D. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, December 1977.
[78] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani, "Epidemic processes in complex networks," *Reviews of modern physics*, vol. 87, no. 3, p. 925, 2015.
[79] F. van Breugel and J. Worrell, "Towards Quantitative Verification of Probabilistic Transition Systems," in *ICALP*, 2001, pp. 421–432.
[80] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden, "Metrics for labelled Markov processes," *Theor. Comput. Sci.*, vol. 318, no. 3, pp. 323–354, 2004.