Evaluation, Reduction, and Approximation of Dynamical Systems and Networks with ERODE

Luca Cardelli[1], Giuseppe Squillace[2(✉)], Mirco Tribastone[2], Max Tschaikowski[3], and Andrea Vandin[4]

[1] University of Oxford, Oxford, UK
[2] IMT Lucca, Lucca, Italy
giuseppe.squillace@imtlucca.it
[3] Aalborg University, Aalborg, Denmark
[4] Sant'Anna School for Advanced Studies Pisa, Pisa, Italy

Abstract. We present ERODE, a tool introduced in 2016 to analyze and reduce differential equations and chemical reaction networks. Over the years, it has been extended with further analysis and reduction techniques, as well as formalisms including differential algebraic equations, Boolean networks, networks, and Markov chains. ERODE can import-export towards several tools, including Matlab, BioNetGen, Modelica, PRISM, STORM, Stochkit, GINsim, SBML, and CoLoMoTo. It also has Python APIs.

1 Introduction

We present ERODE, a software tool originally introduced in [32] for the analysis and reduction of ordinary differential equations and chemical reaction networks (CRNs) [93]. Since its initial release, ERODE has seen substantial advancements, with expanded support for modeling formalisms and the addition of new analysis and reduction techniques.

This paper offers the first comprehensive overview of the current version of ERODE, emphasizing its major enhancements compared to the original release. Over the years, ERODE has been extended to support a broader class of dynamical models, including differential algebraic equations (DAEs), Boolean networks (BNs) [56], and general network structures [65]. In addition to exact reduction methods for ODEs presented in [32], the tool now offers exact reductions for DAEs and Boolean networks, as well as approximate reduction techniques for large-scale ODE systems.

Another key advancement is a newly introduced Python APIs that facilitate the interaction with data science libraries like NetworkX [50] and Pandas [63]. In this direction, the Python APIs provide new techniques for the computation of network embeddings, enabling their use in machine learning tasks and data-driven modeling. Interoperability has also been significantly improved,

M. D'Souza et al. (Eds.): ATVA 2025, LNCS 16145, pp. 345–362, 2026.
https://doi.org/10.1007/978-3-032-08707-2_16

with import/export support for tools and formats such as Matlab [5], Modelica [46], SBML [52], PRISM [57], STORM [42], BioNetGen [15], StochKit [73], and the CoLoMoTo environment [64]. We summarize in Table 1 the main differences between the features provided by the previous ERODE version [32] with respect to the current version.

Table 1. Differences between the features provided by the old and the current version of ERODE.

Features	ERODE [32]	ERODE state-of-the-art
Exact ODE reduction	✓	✓
SMT-based exact reduction	✓	✓
CRNs reduction	✓	✓
Approximate ODE reduction	✗	✓
Exact DAE reduction	✗	✓
Networks reduction	✗	✓
Boolean networks reduction	✗	✓
Python extension	✗	✓
Network embedding	✗	✓

2 Overview

ERODE is a modern IDE offering several functionalities, including syntax highlighting, in-line error annotation, fix suggestions, and auto-completion. As depicted in Fig. 1, the tool presents a flexible and configurable GUI composed by several tabs with ERODE models in some of the supported formalisms. Independently from the used format, an ERODE model contains: *(i)* a model specification section comprising parameters, variables with optional initial values, model dynamics, etc.; *(ii)* a list of commands to analyze, reduce, and export the model.

The output of analyses is information on dynamics given in interactive plots and CSV files. Reductions generate new ERODE models with fewer variables while preserving the dynamics of interest, exactly or approximately, depending on the used technique. Import and export are available through appropriate commands or GUI's wizards. ERODE can also be accessed programmatically from Python.

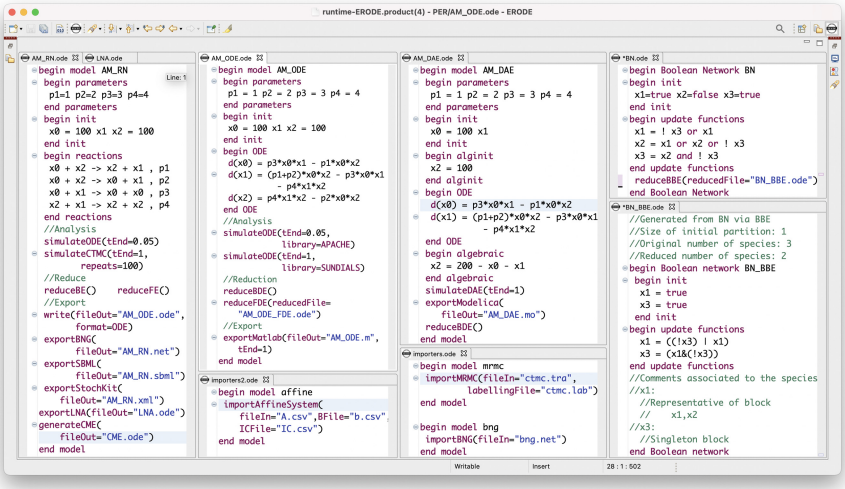


Fig. 1. ERODE: overview of input/output formats, and analysis/reduction commands

3 Input and Output Languages and Formalisms

Ordinary Differential Equations (ODEs). Tab `AM_ODE` of Fig. 1 provides an ODE system with 3 variables, `x0`, `x1`, and `x2`, initialized to `100`, `0`, and `100`, respectively. These variables change value continuously according to their derivatives (block `begin/end ODE`). This is a model of a cell cycle switch, needed to avoid genetic instability during replication [25]. Interestingly, it also describes a distributed algorithm for computing approximately majority among two outcomes (yes/no) [25]. Figure 2 depicts the model dynamics through an ERODE plot view. Here, `x0` approaches 200, while `x1` and `x2` approach 0. Indeed, `x0` and `x2` denote the two different outcomes, while `x1` is an intermediate status.

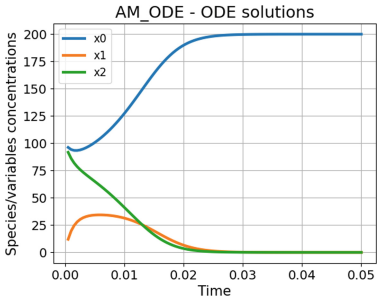


Fig. 2. Majority vote model dynamics

Differential Algebraic Equations (DAEs). DAEs [83] extend ODEs with algebraic constraints. DAEs are often used to describe quantities, following conservation laws like Kirchhoff's laws in electric circuits. The ODEs in tab `AM_ODE` of Fig. 1 model a closed system, meaning that `x0+x1+x2` remains constant to 200. Once the values of two variables are known, that of the third one can be derived. Tab

`AM_DAE` of Fig. 1 shows a DAE system with two differential variables, `x0` and `x1`, and an algebraic one `x2` whose solution is given by the constraint `x2=200-x0-x1`.

Chemical Reaction Networks (CRNs). CRNs [93] are a popular model for biological processes supporting both a deterministic and a stochastic interpretation based on ODEs and continuous-time Markov chains (CTMCs), respectively [24,48]. Tab `AM_RN` of Fig. 1 shows a CRN encoding of AM. It has 3 variables, or *species*, `x0`, `x1`, and `x2` that interact through 4 reactions (block `begin-end reactions`). A reaction contains multisets of species on the left-hand side, the *reagents*, that interact and produce the multiset of species on the right-hand side, the *products*. A reaction executes (or *fires*) with a rate proportional to the kinetic constant, given in ERODE after the products, and to the current amount of its reagents (following the *law of mass-action* [93]). Alternatively, a reaction can have an arbitrary rate (keyword `arbitrary`). The first reaction in the figure states that whenever an instance of `x0` and `x2` interact, the former changes to `x1`, while the latter is preserved. CRNs where all reactions have singleton reagents and products can be seen as CTMCs or weighted graphs, formalisms for which ERODE provides specific analysis techniques (see, e.g., [13,22,27,28,82]). CTMCs in explicit format supported by PRISM [57] and STORM [42] (.tra and .lab files) can be imported using the command `importMRMC` shown in tab `importers` of Fig. 1. Given the relationship between CRN and ODE systems [48], ERODE provides the `write` command, which enables conversion between an ODE system, its corresponding CRN, and vice versa. In Fig. 1 tab `AM_RN`, the command is invoked as `write(fileOut=''AM_ODE.ode'',format=ODE)`, specifying both the output file name and the desired format. This generates the ODE system associated with the original CRN model.

Networks. Networks are mathematical structures used to model entities and their relationships [65]. The entities are given as a set of nodes $N = \{1, ..., n\}$, while the relationships by a symmetric adjacency matrix $A = (a_{i,j}) \in \mathbb{R}^{n \times n}$, where $a_{i,j} = 1$ if there is an edge between node i and j, and $a_{i,j} = 0$ otherwise. In other words, we consider undirected unweighted networks [65]. Networks can be imported into ERODE, considering the associated linear dynamical system $\dot{x} = Ax$, with one equation and variable per node in the network. As discussed later, we have established a formal relation among the backward reductions in ERODE, and notions of network science like role equivalence [62], equitable partitions [49], and centralities [65].

Networks in the de facto standard of format known as *edgelist* can be imported in ERODE via Python APIs. The edgelist format, widely adopted by major network libraries, e.g., the widely-used Python library NetworkX [50], represents a network as a list of lines, each containing a pair of node identifiers

`''IDnode1 IDnode2''`, corresponding to a single edge. In the case of undirected unweighted networks, like ours, weights are implicitly assumed to be 1, and the edges `''i j''` and `''j i''` are identical. We show in Sect. 5.1 how ERODE has been integrated with NetworkX to import popular network repositories, process and visualize them.

Boolean Networks (BNs). BNs [56,79] are established models to qualitatively describe biological systems (e.g., [6,16]). In its simplest form, a BN can be seen as a discrete-time qualitative abstraction of an ODE system: it consists of a set of Boolean variables, each having a Boolean update function associated. Starting from an initial assignment for each variable, discrete-time dynamics is obtained by setting the new state of each variable as the evaluation of its associated function. Tab `BN` of Fig. 1 shows a BN with variables `x1`, `x2`, and `x3`.

4 Underlying Techniques

4.1 Analysis

Numerical Solution of Differential Equations. ERODE is integrated with state-of-the-art numerical solvers for ODEs and DAEs, namely Apache Commons Math library [8] and the SUNDIALS[1] library [51]. Given initial conditions for all variables, we can invoke these solvers to study the evolution of the variables' values over time for a given time horizon. Figure 1 shows the commands used for the different formats: `simulateODE` for CRNs and ODEs, for which we can specify the library to use, and `simulateDAE` for DAEs, where SUNDIALS is used by default. Alternatively, one can use the exporting commands `exportMatlab`, `exportModelica`, and `exportBNG` to use the state-of-the-art solvers offered by Matlab [5], OpenModelica [46] and BioNetGen [15].

Stochastic Analysis and Simulation. CRNs can also be analyzed stochastically by simulating their underlying CTMC, the *master equation* [48], whose states are integer vectors specifying the current *population* of each species. We interface with the FERN library [44] featuring Gillespie's Direct Method, Gibson and Bruck's Next Reaction, and tau-leaping (cf. [47]). Figure 1 shows the corresponding command, `simulateCTMC`. One can also export to the state-of-the-art simulator StochKit [73], or use `generateCME` to build the whole master equation.

Approximate Analysis for CRNs. ERODE supports the linear noise approximation (LNA) of CRNs [17,20] (`exportLNA`, tab `AM_RN` of Fig. 1). LNA analyzes CRNs using a Gaussian process approximating its first and second-order moments (Fig. 3).

[1] Due to upgrade issues, the SUNDIALS library is not currently supported in Ubuntu.

LNA extends the ODEs (the first order moments) of a CRN of N species with N^2 variables for the covariances of each pair of species (the second order moments). Tab `LNA` of Fig. 4 shows the LNA of AM. Another supported analysis is the novel finite state expansion (FSE) [94], which improves the approximation error when the CRN is analyzed by ODEs instead of the ground-truth master equation. FSE extends the ODEs of a CRN with a user-defined part of the master equation dynamics. It can be understood as an instance of a framework of Markov chain truncation where the truncated state space moves dynamically [71]. FSE can be applied to AM using `fse(fileOut=AM_fse.ode,limits=[x0:1])`, obtaining the model in tab `AM_FSE` of Fig. 4. For each species, one has to specify a `limit` below which the stochastic dynamics should be exactly preserved. Beyond such a limit, the species dynamics is given by the original ODE variable. In the figure, we gave limit 1 to `x1` and 0 to the others, leading to species `Dec0` (population 1 for `x0` and 0 for the others) and `Dec1` (only 0-populations).

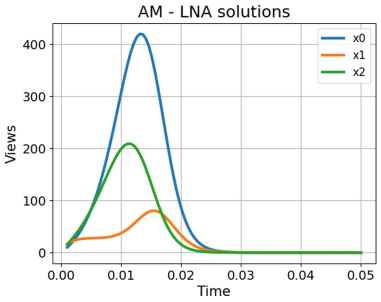


Fig. 3. Majority vote model LNA

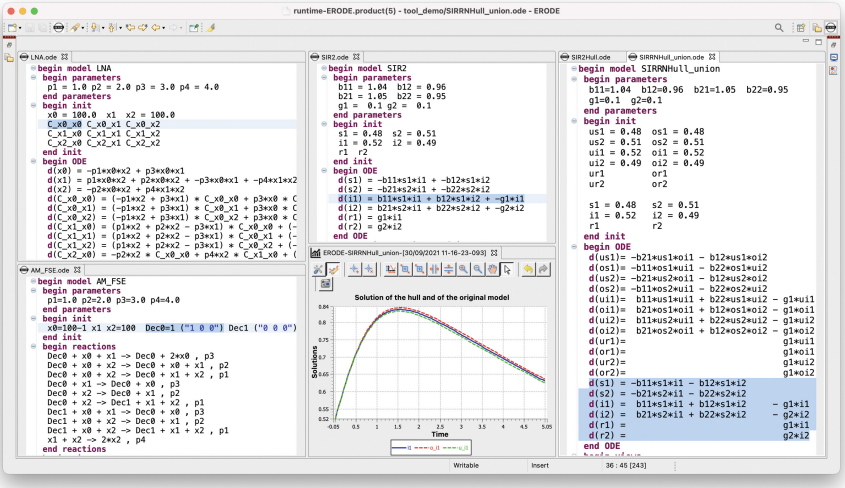


Fig. 4. Linear noise approximation (top-left) and finite state expansion (bottom-left) of AM. Differential hull approximation of a 2-class SIR [87] (centre and right).

4.2 Reduction

Exact Reduction of ODEs with Polynomial Derivatives. These models can be reduced by forward or backward equivalence (FE and BE, respectively) [22,31,33]. The former (`reduceFE`, tab `AM_RN` of Fig. 1) reduces models such that each macro-variable represents the sum of variables in a block. The latter (`reduceBE`) ensures that all variables in a block have the same value at all time points.

SMT-Based Exact Reduction of ODEs with Richer Non-linearities. For ODEs with non-linearities beyond polynomials (e.g., min/max, absolute values, and in general Tarski's decidable theory of reals [78]), we can use forward and backward differential equivalence [23,37], conservative generalizations of FE and BE, respectively (commands `reduceFDE` and `reduceBDE`, resp. see tab `AM_ODE` of Fig. 1).

Exact Reduction of Stochastic CRNs via CRN-to-CRN Transformation. If interested in interpreting a CRN stochastically, one can reduce it by means of species equivalence (SE) [29,34]. SE, command `reduceSE`, identifies a partition of the CRN species and produces a reduced CRN where the marginal probability distribution of each macro-species corresponds to the probability distribution of the sum of the original species in the corresponding block from the original CRN.

Approximate Reduction of ODEs with Polynomial Derivatives. If interested in more aggressive reductions that do not preserve exactly the model dynamics, one can use ε-differential equivalences [30,35] (commands `approxFDE` and `approxBDE`) or differential hull [75,87] (command `computeDifferentialHull`). These are relaxations of differential equivalences where the macro-variables approximately represent the sum of the original variables within some computable bound. Intuitively, the main idea is to perturb the model parameters as little as possible to obtain a new model amenable to exact reduction. The error arising through the parameter change is formally estimated via linearization [35] or differential inequalities [87]. The centre and right of Fig. 4 exemplifies the differential hull for an epidemiologic model from [87] describing a 2-class SIR model. The model has six variables: `si`, `ii`, `ri`, for $i \in \{1, 2\}$ describing the `s`usceptible, `i`nfected, and `r`ecovered individuals in the two classes. The original ODEs are given in tab `SIR2` of Fig. 4. Tab `SIRRNHull_union` of Fig. 4 contains the hull to which, to ease presentation, we added the original model (highlighted). The error estimation for `i1` is depicted with dashed lines in the plot in the center of Fig. 4, while the original solution is provided in solid line.

Similarly, we consider a recurrent model in biochemistry from [30], where we analyze the dynamics of complexes such as receptors and scaffold proteins, which have multiple binding domains [18,40]. In our prototypical model, a molecule A possesses two independent binding sites, each capable of reversibly binding to a molecule B. We denote by A_{10}, A_{01}, and A_{11} the molecular species resulting from the binding of B to the first site, the second site, or both sites of A, respectively. In this model, species with the same number of occupied binding sites are assumed to exhibit identical dynamics, provided they share the same

kinetic parameters. While this assumption simplifies the analysis, it is rather restrictive and often unrealistic in real-world scenarios.

We model the system with ERODE using CRNs, and demonstrate that it is possible to recover underlying symmetries even when the parameters are affected by noise through approximate BDE. In Fig. 5 we show a plot computed by ERODE, where A_{01} and A_{10}, exhibit nearly identical behaviors. This demonstrates the effectiveness of approximate differential equivalence to uncover symmetries and reduce the dimensionality of complex systems under uncertainty.

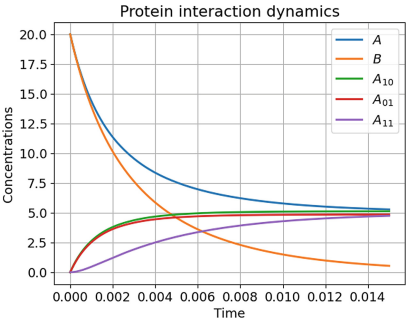


Fig. 5. Protein interaction plot.

Exact Reduction of DAEs with Linear Derivatives. Backward Invariance [81,83] is a generalization of backward equivalence that can be used for linear DAEs, i.e., linear differential equations complemented by algebraic constraints. As shown in tab `AM_DAE` of Fig. 1, this is performed in ERODE using command `reduceBDE`. We note that the considered model is polynomial. Indeed, part of the theory from [83] can be trivially extended to this case.

Network Reduction and Embedding. Networks with adjacency matrix A can be reduced exactly and approximately by ERODE through differential equivalences considering the associated dynamical system $\dot{x} = Ax$. This is because in previous works, we established formal relations among our reduction techniques and notions from network science. In particular, in [82] we showed how our exact reductions preserve centrality measures such as Page Rank, eigenvector, and Katz centrality [65]. Furthermore, we demonstrated how our backward equivalence corresponds to the notion of exact role assignment [62]. This notion, widely applied in the social sciences (see, e.g., [97]), assigns *roles* to each node and states that two nodes with the same role have the same number of neighbors of each role. Furthermore, these partitions are in line with the definition of equitable partitions [49] employed to compute the embedding of a network [76]. A network embedding consists of a matrix $E \in \mathbb{R}^{n \times d}$, with $d < n$, which encodes nodes in a lower-dimensional space while preserving key properties of the original network. The ERODE Python APIs provide exact and approximate versions of BE tailored for networks (methods `reduceNetworkEpsBE`) and the algorithm to compute the embedding (method `computeEmbedding`).

Exact Reduction of BNs. Boolean BE [10,12] is a recasting of BE to BNs. It finds species that maintain the same activation status if initialized equally (`reduceBBE` in tab `BN` of Fig. 1). BNs can be reduced by forward notions thanks to the generalized forward bisimulation (GFB) [9]. It is a reduction for general dynamical systems over commutative monoids, extending and unifying forward equivalences.

GFB can handle various monoids, e.g., on the reals $\mathbb{R}$ or Booleans $\mathbb{B}$, and is applicable to both discrete- and continuous-time dynamical systems. E.g., using monoid $(\mathbb{R}, +)$ we recover forward equivalence for ODEs; using $(\mathbb{B}, \vee)$ we obtain forward reductions of BNs with macro-variables representing the disjunction of the variables in a block. For monoids $(\mathbb{R}, \cdot)$ and $(\mathbb{B}, \wedge)$ we obtain non-linear reductions of non-linear systems. The command supporting it is `reduceFME`.

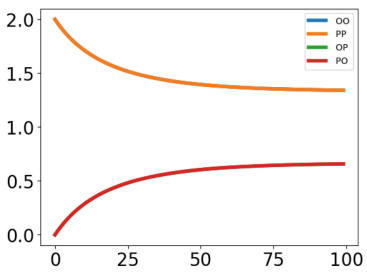
5 Integration with Python

Recently, ERODE has also been offered as an API to be invoked from Python, as exemplified in Fig. 6a. All features are made available through the `ErodeHandler` class. First, it is necessary to start and connect to a JVM (lines 1–2). ERODE models can be loaded as shown in line 4, while further formalisms can be imported with corresponding `import` methods. Line 4 loads KAIC [41], a biological system with 4 species: `OO`, `PP`, `OP`, and `PO`. The model is reduced using BE in line 5, obtaining the partition {`OO`,`PP`} and {`OP`,`PO`}.

Line 7 simulates KAIC specifying the time horizon (1), the number of steps (100), and the output file which can be processed using famous Python libraries such as Pandas and Matplotlib (not shown in the figure). Line 9 generates the plot in Fig. 6b. We can see pairwise equal lines in accordance with the BE reduction. Finally, in line 11, we stop the JVM. Additional features (approximations, initial partitions, etc.) are specified in the GitHub page [2], where we provide a Colab Notebook to use the tool online without any installation [1].

```
1 erodeHandler = ErodeHandler()
2 erode=erodeHandler.start_JVM()
3
4 erode.loadModel("kaic.ode")
5 obtained = erode.computeBE();
6
7 erode.simulateODE(1.0,100,"kaic")
8 df_csv=pd.read_csv("kaic.cdat")
9 df_csv.plot()
10
11 erodeHandler._stop_server()
```

(a) ERODE-Python snippet



(b) KAIC plot.

Fig. 6. Example of loading, reduction, and plotting of a model with ERODE via Python and the resulting plot.

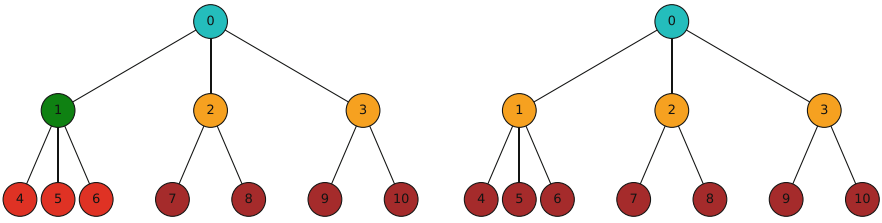
5.1 Extension for Network Embedding

We extended the functionalities of ERODE to compute network embeddings via Python, integrating with the popular NetworkX library. In this section, we

present a guided toy example to illustrate how network embeddings are computed, demonstrate the interoperability between Python and ERODE, and clarify the connection between ODE model reduction and the field of network embedding. The example reported in this section and further experiments are available on the Colab Notebook in [1].

ERODE analyzed networks by modeling them as linear dynamical systems of the form $\dot{x} = Ax$. Each node corresponds to a variable and an associated differential equation. The backward reductions can be used on this dynamical system to find nodes with similar equations. Notably, these reductions are in line with several well-established concepts in network science, such as equitable partitions and exact role assignment, which are based on the idea that nodes sharing similar roles have the same number of neighbors of each role. In [76] we exploit this idea, proposing ε-BE, an approximate version of backward equivalence tailored for networks where the tolerance ε relaxes the original exact definition.

In Fig. 7, we report two partitions computed with ε-BE setting ε equal to 0 and 1, computed on the running example reported in [76]. Here, nodes belonging to the same block present the same color (i.e., role) and the same number of neighbors for each color, fostering the connections with exact role assignment and equitable partitions. Notably, the exact reduction fails to group structurally similar nodes into the same block, separating nodes 2 and 3 from node 1. This occurs because node 1 has a different degree compared to nodes 2 and 3, despite occupying a similar structural position within the network. In contrast, in the approximate setting, a tolerance of $\varepsilon = 1$ is sufficient to overcome this limitation. The reduction successfully groups these nodes together, effectively identifying their shared role and capturing their intuitive structural similarity.



$\{\{0\},\{1\},\{2,3\},\{4,5,6\},\{7,8,9,10\}\}$ $\{\{0\},\{1,2,3\},\{4,5,6,7,8,9,10\}\}$

Fig. 7. Resulting partitions with ε-BE equal to 0 (left) and 1 (right). Nodes in the same block have the same color.

In Fig. 8, we report a snippet of code illustrating how to compute the 1-BE partition and its corresponding embedding with the ERODE Python APIs. In line 1, we compute the partition (method `reduceNetworkEpsBE`) specifying the path to the edgelist file (`''example.edgelist''`), the number of nodes (11), and the ε value (1.0). In line 2, the resulting partition is converted into a Python list. Lines 4–8 show how to integrate with NetworkX to visualize the partition on

```
1 P = erode.reduceNetworkEpsBE("example.edgelist",11,1.0)
2 P_py=erodeHandler.j_to_py_list(P,int)
3
4 plt.figure(figsize=(6,2.8))
5 colors = ["brown","c","orange"]
6 nx.draw(G,pos, node_color=[colors[i-1] for i in P_py], font_color = 'k',
7         edgecolors = "k", node_size=node, with_labels=True )
8 plt.show()
9
10 matrix= np.array(nx.adjacency_matrix(G).todense())
11 E = erodeHandler.computeEmbedding(matrix,P_py)
```

Fig. 8. Snippet of code showing how to use the ERODE Python API and its integration with NetworkX.

networks. Finally, in line 10, the adjacency matrix is extracted using NetworkX, and in line 11 we compute the embedding (method `computeEmbedding`) using the adjacency matrix and the partition.

The resulting embedding $E \in \mathbb{R}^{n \times d}$ encodes each node in a smaller dimensional space $\mathbb{R}^d$ where nodes with the same role present similar vectors. The number of columns d in the embedding corresponds to the number of blocks in the partition, resulting in a significant reduction in matrix size.

Increasing the tolerance parameter ε leads to a more compact network embedding, but it may compromise the accuracy of node aggregation. To address this issue, we introduced in [76] an iterative ε-BE technique (method `reduceIterativeNetworkEpsBE`). This method accepts an initial tolerance ε_0, a maximum tolerance Δ, and a step size δ as inputs. It begins by computing the partition with ε_0 and incrementally increases the tolerance by δ. At each iteration, the partition is refined by collecting the blocks identified at different tolerance levels until reaching the maximum tolerance Δ. This approach avoids greedy node aggregation and allows the obtaining of useful aggregations of large-scale benchmark networks [76]. The ERODE integration with Python allows to carry further analysis on such embeddings using well-known machine-learning libraries like scikit-learn [68] and SciPy [92].

6 Tool Availability, Targets, Interoperability, Case Studies

The tool is freely available at [4] with a manual, and models. Its source code is available at [3]. ERODE is multi-platform and has a rich modern GUI. Tool papers [11,32], and tutorial-like presentations [85,90] have been provided. So far, the tool targeted both teaching and academics. As regards teaching, it is used in courses in the authors' institutions and has been presented in PhD schools [90]. As regards academics, it has been used in many publications, also by external researchers (e.g., [19,39,74]), or to compare with further techniques (e.g., [60,67]).

The import/export to several formalisms and the interoperability with languages like Python [2,80] and Matlab [21,36] made ERODE applicable in a wide variety of domains, including reachability analysis [35,75,77], model reduction with formal bounds [30], network reduction and role assignment [61,82,84], machine learning [76], control theory [14,28,43,55,58], biological systems [54,59,69,95] and others [26,53,86,88,91,96]. In some of these domains, ERODE was evaluated against state-of-the-art tools, providing competitive results in terms of effectiveness and efficiency. For example, in [35,75] we tested ERODE against state-of-the-art tools in reachability analysis such as CORA [7], FLOW* [38], and C2E2 [45]. In the machine learning field [76], we compare the quality of our embedding against struc2vec [72], SEGK [66], DRNE [89], and others.

Overall, ERODE has been applied for the reduction and analysis of many models, including the about 600 CRNs and ODEs from the BioModelsDB [69,70], several BioNetGen models (CRNs) [22], the rouhgly 100 BNs from GinSim [10], and several electrical networks [35]. A selection of these models and all the ones presented here is available at www.erode.eu/examples.html.

This paper summarizes the significant extensions made to ERODE through years of ongoing research and development. The tool is actively maintained and evolving, with new features targeting emerging application areas. Recent advancements, such as network embedding techniques and Python integration, demonstrate its growing versatility. In future work, we plan to continue the theoretical and technical development of ERODE, while also extending its applicability to new domains where formal analysis and reduction techniques can be effectively applied.

Acknowledgments. Several co-authors contributed to ERODE. We thank Georgios Argyres, Giorgio and Giovanni Bacci, Josu Doncel, Nicolas Gast, Alberto Lluch Lafuente, Isabel Perez-Verona, Stefano Tognazzi, Tabea Waizmann. This work was partially supported by the Poul Due Jensen Foundation (883901), the project SERICS (PE00000014), the project Tuscany Health Ecosystem (THE), CUP: B83C22003920001, SMaRT COnSTRUCT (CUP J53C24001460006), within FAIR (CUP I53C22001380006); the last two under the MUR National Recovery and Resilience Plan funded by the EU - NextGenerationEU.

References

1. Erode-python online notebook. https://colab.research.google.com/github/andrea-vandin/erode-python/blob/main/erodePython.ipynb
2. Erode-python website. https://github.com/andrea-vandin/erode-python/wiki
3. Erode source code. https://github.com/IMTAltiStudiLucca/ERODE4.18
4. Erode website. www.erode.eu
5. Matlab. https://www.mathworks.com/products/matlab.html
6. Abou-Jaoudé, W., et al.: Logical modeling and dynamical analysis of cellular networks. Front. Gen. **7**, 94–94 (05 2016). https://doi.org/10.3389/fgene.2016.00094

7. Althoff, M.: An introduction to Cora 2015. In: Proceeding of the Workshop on Applied Verification for Continuous and Hybrid Systems, pp. 120–151 (2015)
8. Apache commons mathematics library. http://commons.apache.org/proper/commons-math/
9. Argyris, G., Lluch-Lafuente, A., Leguizamon-Robayo, A., Tribastone, M., Tschaikowski, M., Vandin, A.: Minimization of dynamical systems over monoids. In: LICS, pp. 1–14 (2023). https://doi.org/10.1109/LICS56636.2023.10175697
10. Argyris, G., Lluch-Lafuente, A., Tribastone, M., Tschaikowski, M., Vandin, A.: Reducing Boolean networks with backward Boolean equivalence. In: 19th International Conference on Computational Methods in Systems Biology CMSB 2021, pp. 1–18 (2021). https://doi.org/10.1007/978-3-030-85633-5_1
11. Argyris, G., Lluch-Lafuente, A., Tribastone, M., Tschaikowski, M., Vandin, A.: An extension of ERODE to reduce Boolean networks by backward Boolean equivalence. In: Petre, I., Paun, A. (eds.) Computational Methods in Systems Biology - 20th International Conference, CMSB 2022, Bucharest, Romania, September 14-16, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13447, pp. 294–301. Springer (2022). https://doi.org/10.1007/978-3-031-15034-0_16
12. Argyris, G., Lluch-Lafuente, A., Tribastone, M., Tschaikowski, M., Vandin, A.: Reducing Boolean networks with backward equivalence. BMC Bioinform. **24**(1), 212 (2023). https://doi.org/10.1186/S12859-023-05326-9
13. Bacci, G., Bacci, G., Larsen, K.G., Tribastone, M., Tschaikowski, M., Vandin, A.: Efficient local computation of differential bisimulations via coupling and up-to methods. In: 36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, pp. 1–14 (2021). https://doi.org/10.1109/LICS52264.2021.9470555
14. Bacci, G., Bacci, G., Larsen, K.G., Squillace, G., Tribastone, M., Tschaikowski, M., Vandin, A.: Dissimilarity for linear dynamical systems. In: Hillston, J., Soudjani, S., Waga, M. (eds.) Quantitative Evaluation of Systems and Formal Modeling and Analysis of Timed Systems - First International Joint Conference, QEST+FORMATS 2024, Calgary, AB, Canada, September 9-13, 2024, Proceedings. Lecture Notes in Computer Science, vol. 14996, pp. 125–142. Springer (2024). https://doi.org/10.1007/978-3-031-68416-6_8
15. Blinov, M.L., Faeder, J.R., Goldstein, B., Hlavacek, W.S.: BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. Bioinformatics **20**(17), 3289–3291 (2004)
16. Bloomingdale, P., Nguyen, V.A., Niu, J., Mager, D.E.: Boolean network modeling in systems pharmacology. J. Pharmacokinet Pharmacodyn. **45**(1), 159–180 (2018). https://doi.org/10.1007/s10928-017-9567-4
17. Bortolussi, L., Cardelli, L., Kwiatkowska, M., Laurenti, L.: Approximation of probabilistic reachability for chemical reaction networks using the linear noise approximation. In: Agha, G., Houdt, B.V. (eds.) QEST. vol. 9826, pp. 72–88. Springer (2016). https://doi.org/10.1007/978-3-319-43425-4_5
18. Camporesi, F., Feret, J., Koeppl, H., Petrov, T.: Combining model reductions. Electr. Notes Theor. Comput. Sci. **265**, 73–96 (2010)
19. Camporesi, F., Feret, J., Lý, K.Q.: Kade: a tool to compile kappa rules into (reduced) ODE models. In: Feret, J., Koeppl, H. (eds.) 15th International Conference on Computational Methods in Systems Biology CMSB 2017. Lecture Notes in Computer Science, vol. 10545, pp. 291–299. Springer (2017). https://doi.org/10.1007/978-3-319-67471-1_18
20. Cardelli, L., Csikász-Nagy, A., Dalchau, N., Tribastone, M., Tschaikowski, M.: Noise reduction in complex biological switches. Sci. Rep. **6**, 20214–20226 (2016)

21. Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: Comparing chemical reaction networks: a categorical and algorithmic perspective. In: LICS, pp. 485–494 (2016)
22. Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: Efficient syntax-driven lumping of differential equations. In: TACAS, pp. 93–111 (2016)
23. Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: Symbolic computation of differential equivalences. In: POPL, pp. 137–150 (2016). https://doi.org/10.1145/2837614.2837649
24. Cardelli, L.: On process rate semantics. Theoret. Comput. Sci. **391**(3), 190–215 (2008)
25. Cardelli, L., Csikász-Nagy, A.: The cell cycle switch computes approximate majority. Sci. Rep. **2**, 656 EP – (2012)
26. Cardelli, L., Csikász-Nagy, A., Dalchau, N., Tribastone, M., Tschaikowski, M.: Noise reduction in complex biological switches. Sci. Rep. **6**(1), 20214 (2016)
27. Cardelli, L., Grosu, R., Larsen, K.G., Tribastone, M., Tschaikowski, M., Vandin, A.: Lumpability for uncertain continuous-time markov chains. In: Quantitative Evaluation of Systems - 18th International Conference, QEST 2021, pp. 391–409 (2021). https://doi.org/10.1007/978-3-030-85172-9_21
28. Cardelli, L., Grosu, R., Larsen, K.G., Tribastone, M., Tschaikowski, M., Vandin, A.: Algorithmic minimization of uncertain continuous-time Markov chains. IEEE Trans. Autom. Control **68**(11), 6557–6572 (2023). https://doi.org/10.1109/TAC.2023.3244093
29. Cardelli, L., Pérez-Verona, I.C., Tribastone, M., Tschaikowski, M., Vandin, A., Waizmann, T.: Exact maximal reduction of stochastic reaction networks by species lumping. Bioinform. **37**(15), 2175–2182 (2021). https://doi.org/10.1093/bioinformatics/btab081
30. Cardelli, L., Squillace, G., Tribastone, M., Tschaikowski, M., Vandin, A.: Formal lumping of polynomial differential equations through approximate equivalences. J. Log. Algebraic Methods Program. **134**, 100876 (2023). https://doi.org/10.1016/J.JLAMP.2023.100876
31. Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: Forward and backward bisimulations for chemical reaction networks. In: 26th International Conference on Concurrency Theory, CONCUR, pp. 226–239 (2015). https://doi.org/10.4230/LIPIcs.CONCUR.2015.226
32. Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: ERODE: a tool for the evaluation and reduction of ordinary differential equations. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), pp. 310–328 (2017)
33. Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: Maximal aggregation of polynomial dynamical systems. Proc. Natl. Acad. Sci. **114**(38), 10029–10034 (2017)
34. Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: Syntactic Markovian bisimulation for chemical reaction networks. In: Models, Algorithms, Logics and Tools, pp. 466–483 (2017). https://doi.org/10.1007/978-3-319-63121-9_23
35. Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: Guaranteed error bounds on approximate model abstractions through reachability analysis. In: Quantitative Evaluation of Systems - 15th International Conference, QEST 2018, pp. 104–121 (2018). https://doi.org/10.1007/978-3-319-99154-2_7
36. Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: Comparing chemical reaction networks: a categorical and algorithmic perspective. Theor. Comput. Sci. **765**, 47–66 (2019). https://doi.org/10.1016/j.tcs.2017.12.018

37. Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: Symbolic computation of differential equivalences. Theoret. Comput. Sci. (2019). https://doi.org/10.1016/j.tcs.2019.03.018
38. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: an analyzer for non-linear hybrid systems. In: Proceedings of the 25th International Conference on Computer Aided Verification (CAV), pp. 258–263. Springer (2013)
39. Chodak, J., Heiner, M.: Spike - reproducible simulation experiments with configuration file branching. In: 17th International Conference on Computational Methods in Systems Biology CMSB 2019. Lecture Notes in Computer Science, vol. 11773, pp. 315–321. Springer (2019). https://doi.org/10.1007/978-3-030-31304-3_19
40. Conzelmann, H., Fey, D., Gilles, E.: Exact model reduction of combinatorial reaction networks. BMC Syst. Biol. **2**(1), 78 (2008)
41. Dalchau, N., et al.: Computing with biological switches and clocks. Nat. Comput. **17**(4), 761–779 (2018). https://doi.org/10.1007/s11047-018-9686-x
42. Dehnert, C., Junges, S., Katoen, J., Volk, M.: A storm is coming: a modern probabilistic model checker. In: Majumdar, R., Kuncak, V. (eds.) Computer Aided Verification - 29th International Conference, CAV 2017. Lecture Notes in Computer Science, vol. 10427, pp. 592–600. Springer (2017). https://doi.org/10.1007/978-3-319-63390-9_31
43. Doncel, J., Gast, N., Tribastone, M., Tschaikowski, M., Vandin, A.: Utopic: underapproximation through optimal control. In: International Conference on Quantitative Evaluation of Systems, pp. 277–291. Springer (2019)
44. Erhard, F., Friedel, C.C., Zimmer, R.: FERN - a Java framework for stochastic simulation and evaluation of reaction networks. BMC Bioinform. **9**(1), 356 (2008). https://doi.org/10.1186/1471-2105-9-356
45. Fan, C., Qi, B., Mitra, S., Viswanathan, M., Duggirala, P.S.: Automatic reachability analysis for nonlinear hybrid models with C2E2. In: CAV, pp. 531–538 (2016)
46. Fritzson, P.: Principles of Object-Oriented Modeling and Simulation with Modelica 3.3. Wiley-IEEE Press, 2 edn. (2014)
47. Gillespie, D.T.: Stochastic simulation of chemical kinetics. Ann. Rev. Phys. Chem. **58**(1), pp. 35–55 (2007)
48. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. **81**(25), 2340–2361 (1977)
49. Godsil, C.D.: Compact graphs and equitable partitions. Linear Algebra Appl. **255**(1–3), 259–266 (1997)
50. Hagberg, A., Swart, P.J., Schult, D.A.: Exploring network structure, dynamics, and function using networkx. Tech. rep, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States) (2008)
51. Hindmarsh, A.C., et al.: SUNDIALS: suite of nonlinear and differential/algebraic equation solvers. ACM Trans. Math. Softw. (TOMS) **31**(3), 363–396 (2005)
52. Hucka, M., et al.: Systems biology markup language (SBML) level 2 version 5: structures and facilities for model definitions. J. Integrative Bioinformat. **12**(2), 271–271 (2015). https://doi.org/10.2390/biecoll-jib-2015-271
53. Iacobelli, G., Tribastone, M., Vandin, A.: Differential bisimulation for a Markovian process algebra. In: Italiano, G.F., Pighizzini, G., Sannella, D.T. (eds.) Mathematical Foundations of Computer Science 2015, pp. 293–306. Springer, Berlin Heidelberg, Berlin, Heidelberg (2015)
54. Ilieva, M., Tschaikowski, M., Vandin, A., Uchida, S.: The current status of gene expression profilings in Covid-19 patients. Clin. Transl. Discovery **2**(3), e104 (2022)

55. Jiménez-Pastor, A., Leguizamon-Robayo, A., Tschaikowski, M., Vandin, A.: Approximate reductions of rational dynamical systems in clue. In: International Conference on Computational Methods in Systems Biology, pp. 108–116. Springer (2024)
56. Kauffman, S.: Homeostasis and differentiation in random genetic control networks. Nature **224**(5215), 177–178 (1969). https://doi.org/10.1038/224177a0
57. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) 23rd International Conference on Computer Aided Verification (CAV'11). Lecture Notes in Computer Science, vol. 6806, pp. 585–591. Springer (2011). https://doi.org/10.1007/978-3-642-22110-1_47
58. Larsen, K.G., Toller, D., Tribastone, M., Tschaikowski, M., Vandin, A.: Optimality-preserving reduction of chemical reaction networks. In: International Symposium on Leveraging Applications of Formal Methods, pp. 13–32. Springer (2024)
59. Leguizamon-Robayo, A., Jiménez-Pastor, A., Tribastone, M., Tschaikowski, M., Vandin, A.: Approximate constrained lumping of chemical reaction networks. Proc. Royal Soc. A **481**(2317), 20240754 (2025)
60. Leguizamon-Robayo, A., Jiménez-Pastor, A., Tribastone, M., Tschaikowski, M., Vandin, A.: approximate constrained lumping of polynomial differential equations. In: Pang, J., Niehren, J. (eds.) Computational Methods in Systems Biology - 21st International Conference, CMSB 2023, Luxembourg City, Luxembourg, September 13-15, 2023, Proceedings. Lecture Notes in Computer Science, vol. 14137, pp. 106–123. Springer (2023). https://doi.org/10.1007/978-3-031-42697-1_8
61. Leguizamon-Robayo, A., Tschaikowski, M.: Efficient estimation of agent networks. In: International Symposium on Leveraging Applications of Formal Methods, pp. 199–214. Springer (2022)
62. Lerner, J.: Role assignments. In: Network analysis: methodological foundations, pp. 216–252. Springer (2005)
63. McKinney, W., et al.: pandas: a foundational python library for data analysis and statistics. Python High Perform. Sci. Comput. **14**(9), 1–9 (2011)
64. Naldi, A., et al.: The colomoto interactive notebook: accessible and reproducible computational analyses for qualitative biological networks. Front. Physiol. **9**, 680 (2018). https://doi.org/10.3389/fphys.2018.00680
65. Newman, M.: Networks. Oxford University Press (2018)
66. Nikolentzos, G., Vazirgiannis, M.: Learning structural node representations using graph kernels. IEEE TKDE **33**(5), 2045–2056 (2019)
67. Ovchinnikov, A., Pérez-Verona, I.C., Pogudin, G., Tribastone, M.: CLUE: exact maximal reduction of kinetic models by constrained lumping of differential equations. Bioinform. **37**(12), 1732–1738 (2021). https://doi.org/10.1093/BIOINFORMATICS/BTAB010
68. Pedregosa, F., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
69. Pérez-Verona, I.C., Tribastone, M., Vandin, A.: A large-scale assessment of exact model reduction in the biomodels repository. In: 17th International Conference on Computational Methods in Systems Biology CMSB 2019, pp. 248–265 (2019). https://doi.org/10.1007/978-3-030-31304-3_13
70. Pérez-Verona, I.C., Tribastone, M., Vandin, A.: A large-scale assessment of exact model reduction in the biomodels repository. Theoret. Comput. Sci. (2021). https://doi.org/10.1016/j.tcs.2021.06.026

71. Randone, F., Bortolussi, L., Tribastone, M.: Refining mean-field approximations by dynamic state truncation. Proc. ACM Meas. Anal. Comput. Syst. **5**(2), 25:1–25:30 (2021). https://doi.org/10.1145/3460092, SIGMETRICS 2021
72. Ribeiro, L.F., Saverese, P.H., Figueiredo, D.R.: struc2vec: learning node representations from structural identity. In: Proceedings of 23rd ACM SIGKDD KDD, pp. 385–394 (2017)
73. Sanft, K.R., Wu, S., Roh, M.K., Fu, J., Lim, R.K., Petzold, L.R.: Stochkit2: software for discrete stochastic simulation of biochemical systems with events. Bioinform. **27**(17), 2457–2458 (2011). https://doi.org/10.1093/bioinformatics/btr401
74. Spaccasassi, C., Yordanov, B., Phillips, A., Dalchau, N.: Fast enumeration of non-isomorphic chemical reaction networks. In: 17th International Conference on Computational Methods in Systems Biology CMSB 2019. Lecture Notes in Computer Science, vol. 11773, pp. 224–247. Springer (2019). https://doi.org/10.1007/978-3-030-31304-3_12
75. Squillace, G., Tribastone, M., Tschaikowski, M., Vandin, A.: An algorithm for the formal reduction of differential equations as over-approximations. In: Ábrahám, E., Paolieri, M. (eds.) Quantitative Evaluation of Systems - 19th International Conference, QEST 2022, Warsaw, Poland, September 12-16, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13479, pp. 173–191. Springer (2022). https://doi.org/10.1007/978-3-031-16336-4_9
76. Squillace, G., Tribastone, M., Tschaikowski, M., Vandin, A.: Efficient network embedding by approximate equitable partitions. In: 2024 IEEE International Conference on Data Mining (ICDM), pp. 440–449 (2024). https://doi.org/10.1109/ICDM59182.2024.00051
77. Squillace, G., Tribastone, M., Tschaikowski, M., Vandin, A.: Approximate regular equivalence by partition refinement. Appl. Netw. Sci. 1**10**(1), 39 (2025). issn = 2364–8228. https://doi.org/10.1007/s41109-025-00726-7.
78. Tarski, A.: A decision method for elementary algebra and geometry. In: Caviness, B.F., Johnson, J.R. (eds.) Quantifier Elimination and Cylindrical Algebraic Decomposition, pp. 24–84. Springer Vienna, Vienna (1998)
79. Thomas, R.: Boolean formalization of genetic control circuits. J. Theor. Biol. **42**(3), 563–585 (1973)
80. Tognazzi, S., Tribastone, M., Tschaikowski, M., Vandin, A.: EGAC: a genetic algorithm to compare chemical reaction networks. In: The Genetic and Evolutionary Computation Conference (GECCO) (2017)
81. Tognazzi, S., Tribastone, M., Tschaikowski, M., Vandin, A.: backward invariance for linear differential algebraic equations. In: 57th IEEE Conference on Decision and Control, CDC 2018, pp. 3771–3776 (2018). https://doi.org/10.1109/CDC.2018.8619710
82. Tognazzi, S., Tribastone, M., Tschaikowski, M., Vandin, A.: Differential equivalence yields network centrality. In: 8th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Distributed Systems ISoLA 2018, pp. 186–201 (2018). https://doi.org/10.1007/978-3-030-03424-5_13
83. Tognazzi, S., Tribastone, M., Tschaikowski, M., Vandin, A.: Differential equivalence for linear differential algebraic equations. IEEE Trans. Autom. Control **67**(7), 3484–3493 (2022). https://doi.org/10.1109/TAC.2021.3108530
84. Toller, D., Tribastone, M., Tschaikowski, M., Vandin, A.: Coarse-graining complex networks for control equivalence. IEEE Trans. Autom. Control (2024)
85. Tribastone, M., Vandin, A.: Speeding up stochastic and deterministic simulation by aggregation: an advanced tutorial. In: 2018 Winter Simulation Conference, WSC 2018, pp. 336–350 (2018). https://doi.org/10.1109/WSC.2018.8632364

86. Tschaikowski, M., Tribastone, M.: Exact fluid Lumpability in Markovian process algebra. Theor. Comput. Sci. **538**, 140–166 (2014). Quantitative Aspects of Programming Languages and Systems (2011-12)
87. Tschaikowski, M., Tribastone, M.: Approximate reduction of heterogenous nonlinear models with differential hulls. IEEE Trans. Autom. Control **61**(4), 1099–1104 (2016). https://doi.org/10.1109/TAC.2015.2457172
88. Tschaikowski, M., Tribastone, M.: Spatial fluid limits for stochastic mobile networks. Perform. Evaluation **109**, 52–76 (2017)
89. Tu, K., Cui, P., Wang, X., Yu, P.S., Zhu, W.: Deep recursive network embedding with regular equivalence. In: Proceedings of 24th ACM SIGKDD KDD, pp. 2357–2366 (2018)
90. Vandin, A., Tribastone, M.: Quantitative abstractions for collective adaptive systems. In: SFM 2016, Bertinoro Summer School, pp. 202–232 (2016). https://doi.org/10.1007/978-3-319-34096-8_7
91. Vandin, A., Tribastone, M.: Quantitative abstractions for collective adaptive systems, pp. 202–232. Springer International Publishing, Cham (2016)
92. Virtanen, P., et al.: Scipy 1.0: fundamental algorithms for scientific computing in python. Nat. Methods **17**(3), 261–272 (2020)
93. Voit, E.O., Martens, H.A., Omholt, S.W.: 150 years of the mass action law. PLOS Comput. Biol. **11**(1), 1–7 (2015). https://doi.org/10.1371/journal.pcbi.1004012
94. Waizmann, T., Bortolussi, L., Vandin, A., Tribastone, M.: Improved estimations of stochastic chemical kinetics by finite-state expansion. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 477(2251), 20200964 (2021). https://doi.org/10.1098/rspa.2020.0964
95. Waizmann, T., Bortolussi, L., Vandin, A., Tribastone, M.: Improved estimations of stochastic chemical kinetics by finite-state expansion. Proc. Royal Soc. A **477**(2251), 20200964 (2021)
96. Waizmann, T., Tribastone, M.: Difflqn: differential equation analysis of layered queuing networks. In: Companion Publication for ACM/SPEC on International Conference on Performance Engineering, pp. 63–68. ICPE '16 Companion, Association for Computing Machinery, New York, NY, USA (2016)
97. Wasserman, S., Faust, K.: Social network analysis: Methods and applications (1994)