ERODE: A Tool for the Evaluation and Reduction of Ordinary Differential Equations

Luca Cardelli[1], Mirco Tribastone[2], Max Tschaikowski[2], and Andrea Vandin[2]

[1] Microsoft Research & University of Oxford, UK
[2] IMT School for Advanced Studies Lucca, Italy

Abstract. We present *ERODE*, a multi-platform tool for the solution and exact reduction of systems of ordinary differential equations (ODEs). *ERODE* supports two recently introduced, complementary, equivalence relations over ODE variables: *forward differential equivalence* yields a self-consistent aggregate system where each ODE gives the cumulative dynamics of the sum of the original variables in the respective equivalence class. *Backward differential equivalence* identifies variables that have identical solutions whenever starting from the same initial conditions. As back-end *ERODE* uses the well-known Z3 SMT solver to compute the largest equivalence that refines a given initial partition of ODE variables. In the special case of ODEs with polynomial derivatives of degree at most two (covering affine systems and elementary chemical reaction networks), it implements a more efficient partition-refinement algorithm in the style of Paige and Tarjan. *ERODE* comes with a rich development environment based on the Eclipse plug-in framework offering: (i) seamless project management; (ii) a fully-featured text editor; and (iii) importing-exporting capabilities.

1 Introduction

Ordinary differential equations (ODEs) have gained momentum in computer science due to the interest in formal methods for computational biology [35,14,20] and for their capability of accurately approximating large-scale Markovian models [24,37,5,40,30]. This has led to a number of results concerning the important, cross-disciplinary, and longstanding problem of reducing the size of ODE systems (e.g., [32,2,27]) using techniques such as abstract interpretation [18,13] and bisimulation [39,19,26,9,12].

Our contribution borrows ideas from programming languages and concurrency theory to recast the ODE reduction problem into finding an appropriate equivalence relation over ODE variables [9,11,12]. Two equivalence relations are presented in [12] for a class of nonlinear systems that covers multivariate rational derivatives and minimum/maximum operators. *Forward differential equivalence* (FDE) identifies a partition of the ODE variables for which a self-consistent aggregate ODE system can be provided which preserves the sums of variables within each block. Variables related by a *backward differential equivalence* (BDE) have the same solution whenever initialized equally. The largest differential equivalence that refines a given input partition is computed via an SMT encoding, using Z3 [15] as a back-end.

ODEs with derivatives that are multivariate polynomials of degree at most two are an important sub-class, covering notable models such as affine systems and *elementary* chemical reaction networks (CRNs) with mass-action semantics (where each reaction has at most two reagents). For this class, in [9] we presented the notions of *forward bisimulation* (FB) and *backward bisimulation* (BB). FB is a sufficient condition for FDE; BB, instead, coincides with BDE for this class of ODEs. The main advantage in using these bisimulations is that the more expensive, symbolic checks through SMT are replaced by "syntactic" ones on a *reaction network*, a finitary structure similar to a CRN which encodes the ODE system. This has led in [11] to an efficient partition-refinement algorithm with polynomial space and time complexity. The bisimulations can be seen as liftings of equivalences and minimization algorithms for continuous-time Markov chains (CTMCs). Indeed the well-known notions of CTMC ordinary and exact lumpability [7] correspond to FB and BB, respectively, when the ODEs represent the CTMC's Kolmogorov equations; and, in this case, the complexity of our partition-refinement algorithm collapses to those of the best-performing ones for CTMC minimization [16,42]. As a consequence of this connection, FDE and BDE are not comparable in general.

This paper presents *ERODE* (https://sysma.imtlucca.it/tools/erode/), a fully-featured multi-platform tool implementing the reduction techniques from [9,11,12]. The tool distinguishes itself from the prototypes accompanying [9,11,12] in that: (i) It is not a command-line prototype but a mature tool with a modern integrated development environment; (ii) It collects all the techniques of our framework for ODE reduction in a unified coherent environment; (iii) It offers a language, and an editor, to express the entire class of ODEs supported by the reduction techniques, while the prototypes could reduce only CRNs; (iv) It implements an ODE workflow consisting of numerical solution and graphical visualization of results; (v) It offers importing/exporting facilities for other formats like biochemical models for the well-known tools BioNetGen [4] and Microsoft GEC [21], or ODEs defined in MATLAB.

Paper outline. Section 2 reviews the reduction techniques from [9,11,12]; Section 3.1 describes *ERODE*'s architecture, while Section 3.2 details its functionalities by discussing the components of an *ERODE* specification. *ERODE*'s capabilities are further stated using a collection of large examples in Section 4. Finally, Section 5 concludes.

2 Theory Overview

The theory behind the techniques implemented in *ERODE* has been presented in [9,11,12], while a tutorial-like unifying presentation can be found in [44]. This section provides an overview that emphasizes relevant aspects for explaining *ERODE*'s performance.

Illustrating example. Let us consider an idealized biochemical interaction between molecules A and B; A can be in two states, u (unphosphorylated) and p (phosphorylated) and can bind/unbind with B. This results in a network with five *species*, denoted by A_u, A_p, B, A_uB, and A_pB. The dynamics of the system is described in Fig. 1 (a) through a CRN with six reactions, where r_1, r_2, r_3 and r_4, are the kinetic constants. By applying the well-known law of mass action, each species is associated with one ODE variable which models the evolution of its concentration as a function of time, with reactions that fire at a speed proportional to their rate times the concentrations of their reagents.

$$
\begin{aligned}
A_u &\xrightarrow{r_1} A_p \\
A_p &\xrightarrow{r_2} A_u \\
A_u + B &\xrightarrow{r_3} A_uB \\
A_uB &\xrightarrow{r_4} A_u + B \\
A_p + B &\xrightarrow{r_3} A_pB \\
A_pB &\xrightarrow{r_4} A_p + B
\end{aligned}
$$

(a)

$$
\begin{aligned}
[\dot{A_u}] &= -r_1[A_u] + r_2[A_p] - r_3[A_u][B] + r_4[A_uB] \\
[\dot{A_p}] &= \quad r_1[A_u] - r_2[A_p] - r_3[A_p][B] + r_4[A_pB] \\
[\dot{B}] &= -r_3[A_u][B] + r_4[A_uB] - r_3[A_p][B] + r_4[A_pB] \\
[\dot{A_uB}] &= \quad r_3[A_u][B] - r_4[A_uB] \\
[\dot{A_pB}] &= \quad r_3[A_p][B] - r_4[A_pB]
\end{aligned}
$$

(b)

Fig. 1: CRN model (a) and underlying ODEs (b) of an idealized biochemical interaction.

For example, $A_u + B \xrightarrow{r_3} A_uB$ fires at speed $r_3[A_u][B]$, where $[\cdot]$ denotes the current concentration of a species. Consequently, this term appears with negative sign in the ODEs of its *reagents* (A_u and B), and with positive sign in the ODE of its *product*, A_uB. The resulting ODEs for our sample system are shown in Fig. 1 (b), where the 'dot' operator denotes the (time) derivative. The model is completed by an *initial condition* which assigns the initial concentration $[X](0)$ to each species X in the network.[1]

Differential equivalences. It can be shown that $\{\{[A_u],[A_p]\},\{[B]\},\{[A_uB],[A_pB]\}\}$ is an FDE for our running example. Indeed, exploiting basic properties one can write self-consistent ODEs for the sums of species in each equivalence class:

$$
\begin{aligned}
[\dot{A_u}] + [\dot{A_p}] &= -r_3\big([A_u] + [A_p]\big)[B] + r_4\big([A_uB] + [A_pB]\big), \\
[\dot{B}] &= -r_3\big([A_u] + [A_p]\big)[B] + r_4\big([A_uB] + [A_pB]\big), \\
[\dot{A_uB}] + [\dot{A_pB}] &= \quad r_3\big([A_u] + [A_p]\big)[B] - r_4\big([A_uB] + [A_pB]\big).
\end{aligned}
\tag{1}
$$

By the change of variables $[A] = [A_u] + [A_p]$ and $[AB] = [A_uB] + [A_pB]$, we get:

$$
[\dot{A}] = -r_3[A][B] + r_4[AB], \quad [\dot{B}] = -r_3[A][B] + r_4[AB], \quad [\dot{AB}] = r_3[A][B] - r_4[AB]
$$

This *quotient* ODE model essentially disregards the phosphorilation status of the A molecule. Setting the initial condition $[A](0) = [A_u](0) + [A_p](0)$ and $[AB](0) = [A_uB](0) + [A_pB](0)$ yields that the solution satisfies $[A](t) = [A_u](t) + [A_p](t)$ and $[AB](t) = [A_uB](t) + [A_pB](t)$ at all time points t.

Backward differential equivalence (BDE) equates variables that have the same solutions at all time points, if initialized equally. It can be shown that $\{\{[A_u],[A_p]\},\{[B]\},\{[A_uB],[A_pB]\}\}$ is also a BDE if $r_1 = r_2$. In this case, we obtain a quotient ODE by keeping only one variable (and equation) per equivalence class, say $[A_u]$, $[B]$ and $[A_uB]$, and rewriting every occurrence of $[A_p]$ and $[A_pB]$ as $[A_u]$ and $[A_uB]$, respectively:

$$
\begin{aligned}
[\dot{A_u}] &= -2r_1[A_u] - r_3[A_u][B] + r_4[A_uB] \\
[\dot{B}] &= -2r_3[A_u][B] + 2r_4[A_uB] \\
[\dot{A_uB}] &= \quad r_3[A_u][B] - r_4[A_uB]
\end{aligned}
$$

[1] Throughout the paper we will work with *autonomous* ODE systems, which are not dependent on time. Also, we will use the terms 'variable' and 'species' interchangeably.

Both FDE and BDE yield a reduced model that can be exactly related to the original one. BDE is lossless, because every variable in the same equivalence class has the same solution, but it is subject to the constraint that variables in the same block be initialized equally. Instead, with FDE one cannot recover the individual solution of an original variable in general, but no constraint is imposed on the initial conditions.

Symbolic minimization algorithms. In [12], establishing that a given partition is a differential equivalence amounts to checking the equality of the functions representing their derivatives. This is encoded in (quantifier-free) first-order logic formulae over the nonlinear theory of the reals. The problem is decidable for a large class of ODEs (and Z3 implements a decision procedure [28]). Such a class is identified by the IDOL language of [12], covering polynomials of any degree, rational expressions, minima and maxima. This captures affine systems, CRNs with mass-action or Hill kinetics [45], and the deterministic *fluid* semantics of process algebra [24,38].

A partition of ODE variables is a BDE if any assignment with equal values in any equivalence class has equal derivatives within each equivalence class. Thus, $\{\{[A_u],[A_p]\},$ $\{[B],[A_uB],[A_pB]\}\}$ is a BDE if and only if the following formula is *valid* (i.e. true for all assignments to the real variables $[A_u]$, $[A_p]$, $[B]$, $[A_uB]$, and $[A_pB]$):

$$[A_u] = [A_p] \ \wedge\ [B] = [A_uB] = [A_pB] \implies f_{[A_u]} = f_{[A_p]} \ \wedge\ f_{[B]} = f_{[A_uB]} = f_{[A_pB]} \quad (2)$$

where $f_{[\cdot]}$ stands for the derivative assigned to the corresponding species in Fig. 1 (b). As usual, the SMT solver will check the satisfiability of its negation.

To automatically find differential equivalences of an ODE model, the SMT checks are embedded in a partition-refinement algorithm that computes the largest differential equivalence which refines a given input partition of variables. In particular, a current partition is refined at each step using the *witness* returned by the SMT solver, i.e. a variable assignment that falsifies the hypothesis that the current partition is a differential equivalence. The algorithm terminates when no witness is found, guaranteeing that the current partition is a differential equivalence. Let us fix the rates $r_1 = r_2 = 1$, $r_3 = 3$ and $r_4 = 4$. Then, $\{\{[A_u],[A_p]\},\{[B],[A_uB],[A_pB]\}\}$ is not a BDE for our running example. Indeed, the assignment $\{[A_u] = 1, [A_p] = 1, [B] = 2, [A_uB] = 2, [A_pB] = 2\}$ is a witness for the negation of Equation 2, since we get $f_{[A_u]} = 2$, $f_{[A_p]} = 2$, $f_{[B]} = 4$, $f_{[A_uB]} = -2$ and $f_{[A_pB]} = -2$ under this assignment. This information is then used to refine the current partition by splitting its blocks into sub-blocks of variables that have the same computation of derivative, obtaining $\{\{[A_u],[A_p]\},\{[B]\},\{[A_uB],[A_pB]\}\}$. No witness can be generated for this partition, ensuring that it is a BDE.

The FDE case is more involved, as discussed in [12]. Considering our running example, we have that $\{\{[A_u],[A_p]\},\{[B],[A_uB],[A_pB]\}\}$ is an FDE if and only if

$$(f_{[A_u]} + f_{[A_p]} = \hat{f}_{[A_u]} + \hat{f}_{[A_p]}) \wedge (f_{[B]} + f_{[A_uB]} + f_{[A_pB]} = \hat{f}_{[B]} + \hat{f}_{[A_uB]} + \hat{f}_{[A_pB]}) \quad (3)$$

is *valid*, where each $\hat{f}_{[\cdot]}$ is obtained from the corresponding derivative $f_{[\cdot]}$ by replacing each variable with the sum of the variables in its block divided by the size of the block. For example, each occurrence of the term $r_4[A_uB]$ is replaced by $r_4\frac{[B]+[A_uB]+[A_uB]}{3}$.

It can be shown that the partition is not an FDE, because a witness falsifying Equation 3 can be found by the SMT solver. However, differently from the BDE case, Equation 3 does not compare single derivatives, but sums of derivatives, hence it cannot be used to decide how to refine the partition. For this, a "binary" characterization of FDE performs SMT checks on each pair of species in the same block of a partition to decide if they have to be split into different sub-blocks.

We remark that the algorithms allow the preservation of user-defined observables. For instance, a variable of interest can be put in an initial singleton block when reducing with FDE. Similarly, in order to meet the constraints on BDE, one can build an initial partition *consistent* with the initial conditions of the original model (that is, two variables are in the same initial block if their initial conditions are the same).

Syntax-driven minimisation. A reaction network (RN) differs from an elementary CRN in that the kinetic constants may be negative. This gives rise to an ODE system with derivatives that are multivariate polynomials of degree at most two [11]. FB and BB are equivalence relations over variables/species in the Larsen-Skou style of probabilistic bisimulation [31]. They are defined in terms of quantities computed by inspecting the set of reactions [31]. In order to check if a given partition of species $\mathcal{H}$ is an FB one computes the *ρ-reaction rate* of a species X, and the *cumulative ρ-production rate* by X of the species in a block $H \in \mathcal{H}$, defined respectively as:

$$\mathbf{crr}[X,\rho] := (\rho(X)+1) \sum_{X+\rho \xrightarrow{\alpha} \pi \in R} \alpha, \quad \mathbf{pr}[X,H,\rho] := (\rho(X)+1) \sum_{X+\rho \xrightarrow{\alpha} \pi \in R} \alpha \cdot \pi(H)$$

where ρ and π are multisets of species, and $\rho(X)$ and $\pi(H)$ denote the multiplicity of X in ρ, and the cumulative multiplicity of species from H in ρ, respectively. We note that ρ is the *reagent partner* of X, which can be either $\emptyset$ for unary reactions, or a species for binary ones. Intuitively, $\mathbf{crr}[X,\rho]$ quantifies the decrease of X's concentration due to reactions where X has partner ρ, while $\mathbf{pr}[X,H,\rho]$ quantifies the increase of its concentration gained by the species in H. In particular, $\mathcal{H}$ is an FB if for any pair of species X, Y in the same block of $\mathcal{H}$ it holds that $\mathbf{crr}[X,\rho] = \mathbf{crr}[Y,\rho]$ and $\mathbf{pr}[X,H,\rho] = \mathbf{pr}[Y,H,\rho]$ for all blocks H of $\mathcal{H}$, and all reagent partners ρ. BB is defined similarly. We refer to [9] for a detailed presentation of FB and BB.

The bisimulation style enabled in [11] the adaptation of Paige and Tarjan's coarsest refinement problem [33] to compute the largest FB/BB. This is done by generalizing algorithms for Markov chain lumping [16,42], obtaining algorithms with $\mathcal{O}(m \cdot n \cdot \log n)$ and $\mathcal{O}(m \cdot n)$ time and space complexity, respectively, with m being the number of monomials appearing in the underlying ODE system, and n the number of ODE variables.

Let us fix $r_1 = 1$, $r_2 = 2$, $r_3 = 3$ and $r_4 = 4$ in our running example. Then, $\{\{A_u, A_p\}, \{B, A_uB, A_pB\}\}$ is not an FB. The algorithm from [11] proceeds in two steps.

In the first step, $\mathbf{crr}[X,\rho]$ is computed for each species X and partner ρ. This information is used to refine the input partition, obtaining $\{\{A_u\},\{A_p\},\{B\},\{A_uB, A_pB\}\}$. The first block is split because we have $\mathbf{crr}[A_u,\emptyset] = r_1$ and $\mathbf{crr}[A_u,\emptyset] = r_2$. Similarly, B is singled out because $\mathbf{crr}[B,\emptyset] = 0$, while $\mathbf{crr}[A_uB,\emptyset] = \mathbf{crr}[A_pB,\emptyset] = r_4$.

In the second step, the algorithm iteratively refines the current partition by selecting one of its blocks, H_{sp}, as a *splitter* in the current iteration: $\mathbf{pr}[X,H_{sp},\rho]$ is computed

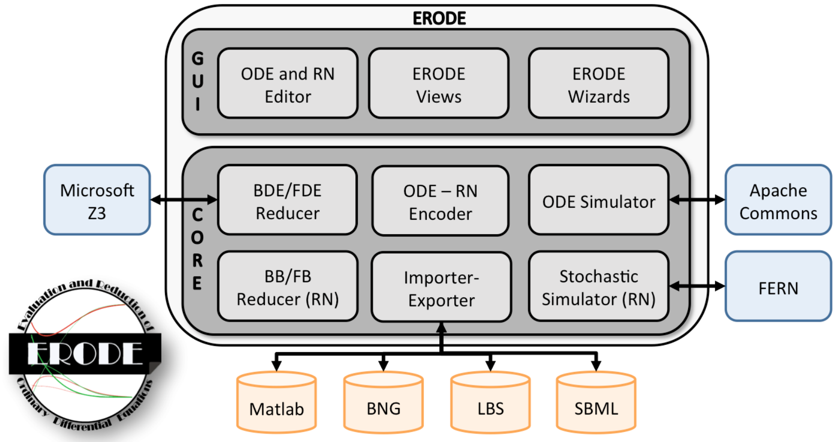

Fig. 2: *ERODE*'s Architecture.

for each X and ρ. This can be done efficiently by considering only reactions with species of H_{sp} in their products. Let us assume that $\{A_u\}$ is the splitter used in the first iteration. Only two reactions have A_u in their products, leading to the computation of $\mathbf{pr}[A_p, \{A_u\}, \emptyset] = r_2$ and $\mathbf{pr}[A_uB, \{A_u\}, \emptyset] = r_4$. Any other production rate of $\{A_u\}$, like $\mathbf{pr}[A_pB, \{A_u\}, \emptyset]$, has value 0. This information is used to refine the partition, obtaining $\{\{A_u\}, \{A_p\}, \{B\}, \{A_uB\}, \{A_pB\}\}$. No further refinement is possible in the following iterations, hence the partition, which is an FB, is returned.

3 *ERODE*

ERODE is an application based on the Eclipse framework for Windows, Mac OS and Linux. It does not require any installation process, and it is available, together with a manual and sample models, at http://sysma.imtlucca.it/tools/erode.

3.1 Architecture

Figure 2 provides a pictorial representation of the architecture of *ERODE*. It is organized in the presentation layer, with the graphical user interface, and the core layer. The main components of the GUI layer are depicted in the screenshot of *ERODE* in Fig. 3, including a fully-featured text editor based on the *xText* framework which supports syntax highlighting, content assist, error detection and fix suggestions (top-middle of Fig. 3). This layer also offers a number of views, including a *project explorer* to navigate among different *ERODE* files (top-left of Fig. 3); an *outline* to navigate the parts of the currently open *ERODE* file (bottom-left of Fig. 3); a *plot view* to display ODE solutions (top-right of Fig. 3); and a *console view* to display diagnostic information like warnings and model reduction statistics (bottom-right of Fig. 3). Finally, the GUI layer offers a number of wizards for: (i) updating *ERODE* to the latest distribution; (ii) creating new *ERODE* files and projects; and (iii) importing models provided in third-party languages.

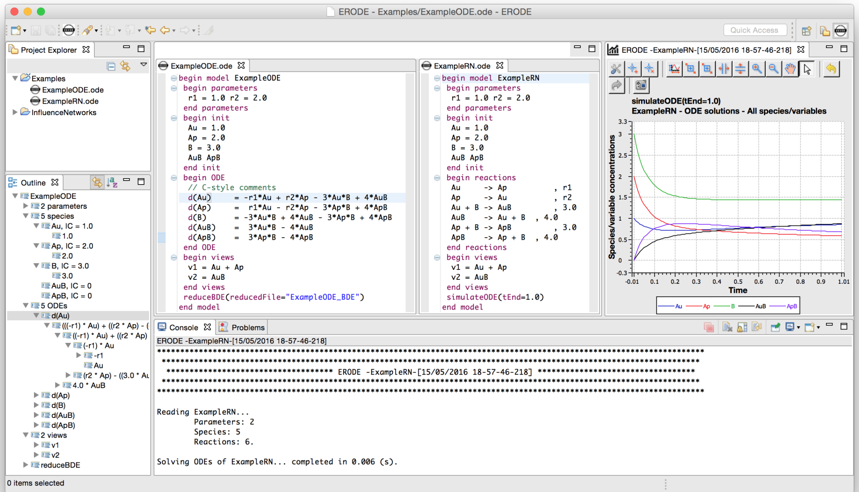

Fig. 3: A screenshot of *ERODE*.

The core layer implements the minimization algorithms and related data structures for FDE, BDE, FB and BB (not detailed here because already addressed in [11,12,44]). A wrapper to Z3 via Java bindings is included for FDE/BDE reduction. The core layer also provides functionalities to encode an RN specification in its corresponding explicit ODE (or IDOL) format, and vice versa, as well as export/import functionalities for third-party languages. Finally, this layer provides support for numerical ODE solvers, using the Apache Commons Maths library [3]. When the input is a CRN (i.e. an RN with only positive rates) it can also be interpreted as a CTMC, following an established approach [22]. Using the *FERN* library [17], *ERODE* features CTMC simulation.

3.2 Language

This section details *ERODE*'s features by discussing the parts composing an *ERODE* file. We do this referring to the two alternative specification formats of our running example from Fig. 1, expressed in *ERODE* in Listings 1 and 2. There are six components of an *ERODE* specification: (i) parameter specification; (ii) declaration of variables and (optional) initial conditions; (iii) initial partition of variables; (iv) ODE system, either in plain format or as an RN; (v) observables, called *views*, tracked by the numerical solver; (vi) commands for ODE numerical solution, reduction, and exporting into other formats.

Parameter specification. An *ERODE* specification might start with an optional list of parameters enclosed in the `parameters` block, each is specified as:

```
<parameter> = expression
```

where `expression` is an arithmetic expression involving parameter names and reals through the following operators: +, −, ⋆, /, ^, `abs`, `min`, and `max`. Parameters can be used to specify values of initial conditions, kinetic rates, or views.

```
begin model ExampleODE
 begin parameters
  r1 = 1.0
  r2 = 2.0
 end parameters
 begin init
  Au = 1.0 Ap = 2.0 B = 3.0
  AuB = 0 ApB = 0
 end init
 begin partition
  {Au,Ap}, {AuB}, {B,ApB}
 end partition
 begin ODE
  // C-style comments
  d(Au)    = -r1*Au + r2*Ap - 3*Au*B + 4*AuB
  d(Ap)    =  r1*Au - r2*Ap - 3*Ap*B + 4*ApB
  d(B)     = -3*Au*B + 4*AuB - 3*Ap*B + 4*ApB
  d(AuB)   =  3*Au*B - 4*AuB
  d(ApB)   =  3*Ap*B - 4*ApB
 end ODE
 begin views
  v1 = Au + Ap
  v2 = AuB
 end views
 reduceBDE(reducedFile="ExampleODE_BDE.ode")
end model
```

Listing 1: Direct ODE specification.

```
begin model ExampleRN
 begin parameters
  r1 = 1.0
  r2 = 2.0
 end parameters
 begin init
  Au = 1.0 Ap = 2.0 B = 3.0
  AuB ApB
 end init
 begin partition
   {Au,Ap}, {AuB}
  end partition
 begin reactions
  Au      -> Ap       , r1
  Ap      -> Au       , r2
  Au + B  -> AuB      , 3.0
  AuB     -> Au + B   , 4.0
  Ap + B  -> ApB      , 3.0
  ApB     -> Ap + B   , 4.0
 end reactions
 begin views
  v1 = Au + Ap
  v2 = AuB
 end views
 simulateODE(tEnd=1.0)
end model
```

Listing 2: Reaction network.

Variable declaration. The mandatory `init` block defines all ODE variables of the model, each specified as:

```
<variable> [= expression]
```

where `expression` is an arithmetic expression as above that evaluates to the initial condition assigned to the variable (defaulting to zero if not specified).

Initial partition of variables. Optionally, a partition of variables can be specified in the `partition` block. This can then be used as the initial partition of the partition-refinement algorithms, as described later. (The user is required to specify only the partition blocks of interest, while all variables not mentioned explicitly are assigned to an implicit additional block.) For instance, Listings 1 and 2 represent the same initial partition {{Au,Ap},{AuB},{B,ApB}}.

ODE Definition. In the direct declaration format (Listing 1) the derivatives are specified within the `ODE` block. Each equation is specified as:

```
d(<variable>) = derivative
```

where `derivative` is an arithmetic expression, possibly containing also ODE variables. This allows to express ODEs belonging to IDOL [12].

In the reaction network format (Listing 2), the ODEs are inferred from reactions of the form:

```
reagents -> products, rate
```

where `reagents` and `products` are two multisets of variables. The multiplicity of a variable in a multiset can be defined through the + operator or with the * operator in the obvious way; that is, `A + A` is equivalent to `2*A`. If `rate` is a variable-free expression that evaluates to a real number (as in all reactions of Listing 2), then the reaction represents a dynamics akin to the law of mass action, discussed in Section 2. In addition, *ERODE* supports more general arithmetical expressions for rates through the `arbitrary` keyword. In this case, the reaction firing rate is explicit. For instance, the two following reactions are equivalent:

```
Au + B -> AuB, arbitrary 3.0*Au*B        Au + B -> AuB, 3.0
```

Views. Views are the observations of interest. As for ODEs, each view can be specified as an arithmetic expression involving variables, parameters and reals. In Listings 1 and 2 the intent is to collect the total concentration of the `A`-molecules, regardless of their phosphorylation state (view `v1`), and the concentration of the species `AuB` (view `v2`).

For a CRN specification, views can also contain terms of form `var(s1)` and `covar(s1,s2)`, to compute the variance of the variable `s1` and the covariance of `s1` and `s2`, respectively. To do so, *ERODE* implements the so-called *linear noise approximation* (e.g., [6]) to be able to study approximations of higher order moments of the concentrations of species in a CRN.

ODE Solution. The ODEs can be numerically solved using the command:

```
simulateODE(tEnd=<value>, steps=<value>, csvFile=<filename>)
```

It numerically integrates the ODE system starting from the specified initial conditions up to time point `tEnd`, interpolating the results at `steps` equally spaced time points. Two plots are generated, one for the the trace of each ODE variable and one for the trace of each specified view, respectively. If the optional argument `csvFile` is present, the plots are exported into a comma-separated values format.

Conversion options. An explicit ODE specification can be converted in the RN format (and vice versa) using

```
write(fileOut=fileName,format=<ODE|RN|MA-RN>)
```

If `format` is set to `ODE`, then the target file will be in explicit ODE format, while with `RN` an RN with possibly arbitrary rates will be generated. If the *ERODE* input to be exported is an explicit ODE with derivatives given by multivariate polynomials of degree at most two, the `MA-RN` will use the encoding of [11] to output a mass-action RN.

Export to third-party languages. The command:

```
export<format>(fileOut=fileName)
```

exports *ERODE* files into four different target third-party languages:

`Matlab` : a Matlab function representing an ODE system (extension `.m`).

BNG : a CRN generated with the well-established tool BioNetGen version 2.2.5-stable [4] (extension `.net`). This is available for CRN specifications only.

LBS : format of the Microsoft's tool GEC [21] (extension `.lbs`), available for CRN specifications only.

SBML : the well-known `SBML` interchange format (http://sbml.org) (extension `.sbml`).

Reduction commands. All ODE reduction commands share the common signature

```
reduce<kind>(prePartition=<NO|IC|USER>, reducedFile=<name>)
```

where `kind` can be `FDE`, `BDE`, `FB`, or `BB`. The ODE input format affects which reduction options are available. For an ODE system defined directly, only FDE and BDE are enabled. FB and BB are additionally available for RNs representing polynomial ODE systems of degree at most two [11]. This is imposed by having reagents multisets of size at most two in each reaction and restricting to mass-action type rate expressions.

The option `prePartition` defines the initial partition for the minimization algorithm. The maximal aggregation is obtained with the `NO` option. If it is set to `IC`, the initial partition is built according to the constraints given by the initial conditions: variables are in the same initial block whenever their initial conditions are equal. If the option is set to `USER`, then the partition specified in the `partition` block will be used.

If `reducedFile` is present, then a reduced model will be generated according to the computed partition following the model-to-model transformation from [9] (for FB and BB) and [12] (for FDE and BDE). This will have the same format as the input, and will contain one variable for each equivalence class. The name of the variable is given by the first variable name in that block, according to a lexicographical order.

Considering our running example, no reduction is found running `reduceFDE` on Listing 1 if pre-partitioning is set to `USER`. Instead, when it is set to `NO` we find the FDE $\{\{A_u, A_p\}, \{B\}, \{A_uB, A_pB\}\}$ discussed in Section 2, implying that it is the maximal one of the model. The output file for the case without pre-partitioning is provided in Listing 3, which also shows that the association between the original ODE variables and those in the reduced model is maintained by annotating the output file with comments alongside the new variables.[2] This information can be useful for visually inspecting the reduced model in order to gain insights into the physical interpretation of the reduction [9]. Finally, we note that each reduced species has initial concentration equal to the sum of those in the corresponding block.

In Section 2 we have shown that the partition $\{\{A_u, A_p\}, \{B\}, \{A_uB, A_pB\}\}$ is also a BDE provided that $r_1 = r_2$. However, this reduction is not found if running `reduceBDE` with pre-partitioning set to `IC`, as it violates the initial conditions for `Au` and `Ap`. Instead, if the pre-partitioning is disabled, then the above partition is the coarsest refinement, but the user is warned about the inconsistency with the initial conditions. The BDE reduction without pre-partitioning for `r1=r2=1.0` is given in Listing 4. The initial condition for the ODE of each representative is equal to that of the corresponding original variable.

[2] Here output files have been typographically adjusted to improve presentation.

```
begin model ExampleODE_FDE
 begin parameters
  r1 = 1.0
  r2 = 2.0
 end parameters
 begin init
  Au   = 1.0 + 2.0
  B    = 3.0
  AuB
 end init
 begin ODE
  d(Au)    = - 3*Au*B + 4*AuB
  d(B)     = - 3*Au*B + 4*AuB
  d(AuB)   =   3*Au*B - 4*AuB
 end ODE
 //Comments associated to the species
 //Au:  Block {Au, Ap}
 //B:   Block {B}
 //AuB: Block {AuB, ApB}
end model
```

Listing 3: FDE reduction.

```
begin model ExampleODE_BDE
 begin parameters
  r1 = 1.0
  r2 = 1.0
 end parameters
 begin init
  Au   = 1.0
  B    = 3.0
  AuB
 end init
 begin ODE
  d(Au)    = - 3*Au*B + 4*AuB
  d(B)     = - 6*Au*B + 8*AuB
  d(AuB)   =   3*Au*B - 4*AuB
 end ODE
 //Comments associated to the species
 //Au:  Block {Au, Ap}
 //B:   Block {B}
 //AuB: Block {AuB, ApB}
end model
```

Listing 4: BDE reduction.

The model of Listing 2 is not reduced by FB, independently on the pre-partitioning choice. This is consistent with FB being only a sufficient condition for FDE (although it is effective on many meaningful models from the literature, as discussed in [11]). The result of the BB reduction is instead provided in the right inset. As for BDE, we considered the case `r1=1.0` and `r2=1.0` without pre-partitioning. It can be shown that the underlying ODEs of the reduced model correspond to those of Listing 4, as expected. (The placeholder species `SINK` is created to rule out reactions that have no products.)

```
 begin parameters
  r1 = 1.0   r2 = 1.0
 end parameters
 begin init
  Au = 1.0   B = 3.0   AuB
  SINK
 end init
 begin reactions
  Au -> 2*Au , r2
  Au -> SINK , r1
  Au + B -> Au , 3.0
  Au + B -> AuB , 3.0
  AuB -> Au + B , 4.0
  AuB -> B + AuB , 4.0
 end reactions
 //Comments associated to the species
 //Au:  Block {Au, Ap}
 //B:   Block {B}
 //AuB: Block {AuB, ApB}
end model
```

4 Evaluation

Prototypal versions of *ERODE*'s reduction algorithms have been evaluated in [9,11,12,44] against a number of models from the literature. The main outcomes of these analyses are: (i) Our reduction techniques are effective, as we found reductions in many large-scale models that enjoy substantial speed-ups for the numerical ODE solution [9,11]; (ii) Our forward and backward notions are not comparable in general, as there are models which can be reduced by the former but not by the latter, and vice versa [9]; (iii) In some cases, observables of interest specified by the modeller can be used to automatically generate initial partitions that lead to forward reductions preserving them [44]; (iv) FDE and BDE are less efficient than FB and BB, but are more general and lead to better reductions in the forward case [12]. (v) FB and BB correspond to the notions of ordinary and exact CTMC lumpability [7], respectively [11]; in particular FB has been validated in [11] against the ordinary CTMC lumping algorithm [16] implemented in MRMC [29].

Configuration		*FB reduction (s)*			*BB reduction (s)*		
$\lvert R\rvert$	$\lvert S\rvert$	*Min*	*Avg*	*Max*	*Min*	*Avg*	*Max*
1.00E+6	1.00E+5	2.34E+0	2.35E+0	2.38E+0	4.98E+0	5.40E+0	6.17E+0
5.00E+6	5.00E+5	1.95E+1	1.96E+1	1.98E+1	3.91E+1	3.96E+1	3.98E+1
1.00E+7	1.00E+6	3.89E+1	3.91E+1	3.92E+1	9.59E+1	9.77E+1	9.95E+1
1.50E+7	1.50E+6	9.62E+1	9.71E+1	9.86E+1	1.67E+2	1.68E+2	1.69E+2
2.00E+7	2.00E+6	1.58E+2	1.59E+2	1.62E+2	3.30E+2	3.31E+2	3.33E+2
2.50E+7	2.50E+6	3.42E+2	3.46E+2	3.52E+2	8.72E+2	8.92E+2	9.24E+2
3.00E+7	3.00E+6	Out of memory			Out of memory		

Table 1: FB and BB reductions for random RNs with 30% of binary reactions.

With *ERODE* we could confirm all these previously reported results. In this section, we carry out a systematic evaluation of *ERODE*'s capabilities in terms of scalability as a function of: the input model size (Section 4.1), its degree of non-linearity (Section 4.2), and its degree of aggregability (Section 4.3). For this, we considered a collection of synthetic benchmarks to be able to gain full control on the model parameters to be changed for performing these studies.

All experiments were run on a 3.2 GHz Intel Core i5 machine with 16 GB of RAM. In order to avoid interferences, each single model was tested on a fresh Java Virtual Machine, with assigned 10 GB of RAM. For each reduction we used initial partitions with one block only containing all variables. Information on how to replicate the experiments is available at http://sysma.imtlucca.it/tools/erode/benchmarks.

4.1 Scalability

We begin by studying the scalability of the partition-refinement algorithms in terms of the model size. Such an assessment has been conducted already in [12] for BDE/FDE, where it has been shown that BDE can handle models up to 786,432 reactions and 65,538 species, while FDE handled up to 8,620 reactions and 745 species. For larger models Z3 issued out-of-memory errors. Here we confirm these figures when using *ERODE*.

Instead, to study the scalability of FB and BB, we consider a number of random RNs underlying degree-two polynomials. The set-up is as follows. First, we fixed 7 different configurations with increasing number of reactions and species (columns $|R|$ and $|S|$ of Table 1, respectively). For each configuration, we generated five random RNs, each having 70% unary reactions in the form $A \to B$, leading to degree-one monomials in the ODEs for species A and B, and 30% binary reactions in the form $A+B \to C$, leading to degree-two monomials for A, B, and C. (Here the percentage of binary reactions was fixed arbitrarily — it will be studied in more detail in the next subsection.) The species involved in each reaction were sampled uniformly (with re-insertion), while the kinetic rates were drawn uniformly from the interval [1;10,000]. We ensured that none of the RNs could be reduced in order to stress the algorithm by forcing it to evaluate the maximum number of partition-refinement iterations. To reduce noise, the measurements for each RN were repeated three times, for a total of 15 experiments per configuration.

Table 1 summarizes the results. The columns *Min*, *Avg* and *Max* provide, respectively, the minimum, average, and maximum reduction times obtained per configuration. FB and

	Percentage of binary reactions					
	0%	20%	40%	60%	80%	100%
Syntactic reductions $\lvert R\rvert$=3.50E+6 and $\lvert S\rvert$=2.50E+5						
FB (s)	6.40E+0	1.31E+1	1.58E+1	2.03E+1	2.18E+1	2.65E+1
BB (s)	1.51E+1	2.53E+1	3.06E+1	3.61E+1	4.44E+1	4.98E+1
Symbolic reductions $\lvert R\rvert$=1.50E+3 and $\lvert S\rvert$=2.50E+2						
FDE (s)	—	2.81E+2	3.63E+2	4.86E+2	1.06E+3	2.50E+3
BDE (s)	2.87E–1	2.89E–1	2.90E–1	2.92E–1	2.95E–1	2.96E–1

Table 2: Reductions of random elementary RNs with varying ratio of binary reactions.

BB reductions succeeded for models up to 25,000,000 reactions and 2,500,000 species, requiring about 5 and 15 minutes, respectively. Larger RNs led to out-of-memory errors. The first and sixth row show that an increment of factor 25 in both the number of species and reactions leads to about two order of magnitude larger runtimes, consistently with the algorithms' complexities (Section 2). Finally, we note that BB reductions were performed twice as slow as the corresponding FB ones This is consistent with [11], which shows that for BB the inner loops of the partition-refinement algorithm execute about twice as many instructions as for FB (see Algorithms 4 and 5 from [11]).

4.2 Degree of Nonlinearity

We now study how the reduction runtimes are affected by the nonlinearity in the model, here measured as the percentage of monomials of degree greater than one in the ODE.

For FB and BB we fixed a configuration with $|R|$=3,500,000, and $|S|$=250,000, similarly to the largest CRN in [9,11], and considered models with increasing percentage of binary reactions. For each percentage, we generated five RNs similarly to Section 4.1. Table 2 gives the reduction runtimes. We note an increase in the runtimes as a function of the percentage of binary reactions. This is consistent with the time complexity of FB and BB (Section 2). In fact, RNs with higher ratio of binary reactions have more monomials in the underlying ODEs (see Section 4.1). However we note that in practice the runtimes at worst only quadruplicates respect to the linear case (column 0%).

Table 2 also reports the evaluation for FDE/BDE considering RNs of size $|R|$ = 1,500 and $|S|$ = 250. We note that BDE requires much less time than FDE, as expected from the discussion in Section 2. In addition, we find that the BDE runtimes are essentially not affected. The same does not hold for FDE: incrementing the percentage of binary reactions by 20 leads to an increment of factor between 1.3 and 2.3 in the runtimes. The different impact on the performance of BDE and FDE can be explained by the algebraic transformations required by FDE to compute the $\hat{f}_{[\cdot]}$ terms shown in Equation (3). Consider for example a partition $\mathcal{H}$ and a species X belonging to a block H of $\mathcal{H}$. Then, terms of form X^2 are substituted with terms of form $(\sum_{Y \in H} Y)^2/|H|^2$, with an explosion in the number of monomials appearing in the derivatives. We do not provide the FDE runtime for the 0% case, because it can be shown that, akin to CTMC lumpability, partitions with one block only are FDE for RNs with unary reagents and products only.

	FB reduction			BB reduction		
Sym.	*Red. (s)*	*Iter.*	$\lvert\mathcal{H}\rvert$	*Red. (s)*	*Iter.*	$\lvert\mathcal{H}\rvert$
9	3.61E+0	223	222	7.60E+0	224	222
8	3.96E+0	663	662	8.12E+0	664	662
7	4.18E+0	1,923	1,922	8.63E+0	1,924	1,922
6	4.51E+0	5,379	5,378	8.73E+0	5,380	5,378
5	4.51E+0	14,339	14,338	8.77E+0	14,340	14,338
4	4.71E+0	35,849	35,842	8.97E+0	35,844	35,842
3	5.29E+0	81,959	81,922	9.58E+0	81,924	81,922
2	5.56E+0	163,910	163,842	9.71E+0	163,845	163,842
0	6.29E+0	262,147	262,146	1.12E+1	262,157	262,146

(a) 9 binding sites, $|R|$=3,538,944, $|S|$=262,146

	FDE reduction			BDE reduction		
Sym.	*Red.(s)*	*Iter.*	$\lvert\mathcal{H}\rvert$	*Red.(s)*	*Iter.*	$\lvert\mathcal{H}\rvert$
4	1.39E+2	13,284	37	4.10E–1	42	37
3	2.66E+2	38,355	82	6.00E–1	81	82
2	3.52E+2	50,517	162	7.75E–1	113	162
0	2.54E+2	37,022	258	2.22E–1	9	258

(b) 4 binding sites, $|R|$=1,536, $|S|$=258

Table 3: Reductions for variants of M1 of [9,11] by decreasing binding sites' symmetries.

We further study the behavior of FDE/BDE as a function of the maximum degree of the polynomials. For this, we constructed RNs with 60% unary reactions and 40% n-ary reactions (leading to degree n monomials in the underlying ODEs), with $n = 20, 40, 60, 80, 100$. The RNs have size $|R|$=1,500, $|S|$=250, as in the last rows of Table 2. The runtimes, averaged over 5 random RNs, are given in the bottom inset. The BDE runtime for $n = 20$ is five times that of the corresponding one for degree-two polynomials (third column of Table 2), and it further increases of factor 20 for $n = 100$. FDE succeeded for up to $n = 40$, despite the discussed highly demanding algebraic manipulations required, while Z3 returned "unknown" for larger values of n, suggesting an out of memory error.

	Maximum degree of the polynomial n				
	20	40	60	80	100
BDE (s)	1.46E+0	8.30E+0	9.881E+0	1.42E+1	3.34E+1
FDE (s)	7.00E+2	2.00E+3	– *"unknown"* –		

4.3 Number of Iterations vs Runtime

Finally, we study how the number of performed iterations of the partition-refinement algorithms affects the runtime. For FB and BB this is done using variants of model M1 of [9,11], with 3,538,944 reactions and 262,146 species. It is the largest of a family of synthetic benchmarks used in [36] to study the scalability of a network-free simulator for CRNs. It models an idealized binding/unbiding interaction between two molecules, A and B, which can take place through A's nine *binding sites*. Symmetries in the model are introduced through the assumption that such binding sites are equivalent, in the sense that the rate of binding/unbinding does not depend on the identity of the binding site.

Table 3 (a) studies increasingly less symmetric variants of the model, obtained by changing the binding/unbinding rates of each site; the first column shows the number of equivalent sites in the model. The columns *Red.* provide the runtimes of our algorithms. Columns *Iter.* and $|\mathcal{H}|$ show the number of iterations performed and the blocks for the coarsest partitions obtained. Decreasing the number of symmetric binding sites by one leads to an increment of factor between 2 and 3 in the number of iterations and blocks in the partitions. Instead, the runtime increases only slightly: the number of iterations between the first and the last experiment are separated by three orders of magnitude while their respective runtimes at most only double for both FB and BB. This can be explained by the fact that, at each iteration, one block of the current partition is chosen as a potential *splitter*. Therefore only the reactions that have species belonging to the splitter in their products will be inspected. As a result, the smaller is the current splitter, the fewer reactions are scanned in the iteration. More importantly, as discussed in detail in [11], the FB/BB algorithms follow Paige and Tarjan's approach of *ignoring the largest sub-part* [33]. This means that, whenever a block is split, one of its sub-blocks with maximal size will not be further used as splitter. This guarantees that each species will appear in at most $\log |S|$ splitters, with S being the species in the model.

Table 3 (b) reports a similar analysis for FDE and BDE. We use a simplification of M1 where A has only four binding sites, obtaining 1,536 reactions and 258 species, to which both FDE and BDE can be successfully applied. The table has the same structure of Table 3 (a), however here *Iter.* counts the number of performed SMT checks. The table also shows that our symbolic algorithms are strongly affected by the number of performed iterations: the nature of the FDE/BDE algorithms does not allow for advanced optimizations like those discussed for FB/BB. Lastly, it is interesting to note that the number of necessary iterations decreases in the case when no reduction is found (last row of Table 3 (b)). Here, the computation of the largest BDE required nine SMT checks: the SMT solver was able to split the initial block in 250 blocks in the first iteration, then one new block has been created in the following eight iterations until reaching the final partition with one block per species. For FDE, instead, 37,022 SMT checks were necessary. We note that this is relatively close to the number of binary comparisons among 258 elements, i.e. $\binom{258}{2} = 33153$, as expected from the discussion in Section 2.

5 Conclusion

We presented *ERODE*, a tool for the analysis and reduction of ODEs. The main novelty is in the implementation of partition-refinement algorithms that compute the largest equivalence over ODE variables that refine an initial partition, using both syntactic criteria as well as symbolic SMT ones. However, currently *ERODE* does not support algorithms required when the modeler is interested in equivalences that satisfy constraints that are not expressible as initial partitions. An example is the notion of emulation used for model comparison between two CRNs [8], where each BDE partition block must contain at least one species of the *source* CRN, and exactly one of the *target*. We plan to integrate *ERODE* with the algorithm for computing all the BDEs of a CRN from [10].

ERODE is concerned with exact aggregations. These may be too strong in some cases, as small perturbations in the parameters might prevent reductions for ODE variables with nearby trajectories in practice. This motivated the development of approximate notions of equivalence [34,1,43,23]. Preliminary work is treated in [25,41]. However these approaches lack an algorithm for automatic reduction, and they provide error bounds that tend to grow fast with time. In the future we aim at tackling these two issues.

Acknowledgments. This work was partially supported by the EU project QUANTICOL, 600708. L. Cardelli is partially funded by a Royal Society Research Professorship.

References

1. Alessandro Aldini, Mario Bravetti, and Roberto Gorrieri. A process-algebraic approach for the analysis of probabilistic noninterference. *JCS*, 12(2):191–245, 2004.
2. Masanao Aoki. Control of large-scale dynamic systems by aggregation. *IEEE Trans. Autom. Control*, 13(3):246–253, 1968.
3. Apache Commons Mathematics Library, http://commons.apache.org/proper/commons-math/.
4. Michael L. Blinov, James R. Faeder, Byron Goldstein, and William S. Hlavacek. Bionetgen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20(17):3289–3291, 2004.
5. Luca Bortolussi, Jane Hillston, Diego Latella, and Mieke Massink. Continuous approximation of collective system behaviour: A tutorial. *Performance Evaluation*, 70(5):317–349, 2013.
6. Luca Bortolussi and Roberta Lanciani. *QEST*, chapter Model Checking Markov Population Models by Central Limit Approximation, pages 123–138. 2013.
7. Peter Buchholz. Exact and Ordinary Lumpability in Finite Markov Chains. *Journal of Applied Probability*, 31(1):59–75, 1994.
8. Luca Cardelli. Morphisms of reaction networks that couple structure to function. *BMC Systems Biology*, 8(1):84, 2014.
9. Luca Cardelli, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. Forward and backward bisimulations for chemical reaction networks. In *CONCUR*, 2015.
10. Luca Cardelli, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. Comparing chemical reaction networks: A categorical and algorithmic perspective. In *LICS*, 2016.
11. Luca Cardelli, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. Efficient syntax-driven lumping of differential equations. In *TACAS*, volume 9636, pages 93–111, 2016.
12. Luca Cardelli, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. Symbolic computation of differential equivalences. In *POPL*, 2016.
13. Vincent Danos, Jérôme Feret, Walter Fontana, Russell Harmer, and Jean Krivine. Abstracting the differential semantics of rule-based models: Exact and automated model reduction. In *LICS*, pages 362–381, 2010.
14. Vincent Danos and Cosimo Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.
15. Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *TACAS*, pages 337–340, 2008.
16. Salem Derisavi, Holger Hermanns, and William H. Sanders. Optimal state-space lumping in Markov chains. *Inf. Process. Lett.*, 87(6):309–315, 2003.
17. Florian Erhard, Caroline C. Friedel, and Ralf Zimmer. FERN - a Java framework for stochastic simulation and evaluation of reaction networks. *BMC Bioinformatics*, 9(1):356, 2008.
18. Jérôme Feret, Vincent Danos, Jean Krivine, Russ Harmer, and Walter Fontana. Internal coarse-graining of molecular systems. *Proceedings of the National Academy of Sciences*, 106(16):6453–6458, 2009.
19. Jerome Feret, Thomas Henzinger, Heinz Koeppl, and Tatjana Petrov. Lumpability abstractions of rule-based systems. *Theoretical Computer Science*, 431:137–164, 2012.
20. Jasmin Fisher and Thomas A. Henzinger. Executable cell biology. *Nature Biotechnology*, 25(11):1239–1249, 2007.
21. Microsoft GEC, http://research.microsoft.com/en-us/projects/gec/.
22. Daniel Thomas Gillespie. Exact Stochastic Simulation of Coupled Chemical Reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.
23. Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Approximate reasoning for real-time probabilistic processes. *Logical Methods in Computer Science*, 2(1), 2006.

24. Richard A. Hayden and Jeremy T. Bradley. A fluid analysis framework for a Markovian process algebra. *Theor. Comput. Sci.*, 411(22-24):2260–2297, 2010.
25. Giulio Iacobelli and Mirco Tribastone. Lumpability of fluid models with heterogeneous agent types. In *DSN*, pages 1–11, 2013.
26. Giulio Iacobelli, Mirco Tribastone, and Andrea Vandin. Differential bisimulation for a Markovian process algebra. In *MFCS*, 2015.
27. Yoh Iwasa, Viggo Andreasen, and Simon Levin. Aggregation in model ecosystems. I. Perfect aggregation. *Ecological Modelling*, 37(3-4):287–302, 1987.
28. Dejan Jovanovic and Leonardo Mendonça de Moura. Solving non-linear arithmetic. In *IJCAR*, pages 339–354, 2012.
29. Joost-Pieter Katoen, Maneesh Khattri, and Ivan S. Zapreev. A Markov Reward Model Checker. In *QEST*, pages 243–244, 2005.
30. Matthias Kowal, Max Tschaikowski, Mirco Tribastone, and Ina Schaefer. Scaling Size and Parameter Spaces in Variability-Aware Software Performance Models. In *ASE*, 2015.
31. Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.
32. Genyuan Li and Herschel Rabitz. A general analysis of exact lumping in chemical kinetics. *Chemical Engineering Science*, 44(6):1413–1430, 1989.
33. Robert Paige and Robert Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.
34. Alessandra Di Pierro, Chris Hankin, and Herbert Wiklicky. Quantitative relations and approximate process equivalences. In *CONCUR*, pages 498–512, 2003.
35. Aviv Regev and Ehud Shapiro. Cellular abstractions: Cells as computation. *Nature*, 419(6905):343–343, 09 2002.
36. Michael W. Sneddon, James R. Faeder, and Thierry Emonet. Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. *Nature Methods*, 8(2):177–183, 2011.
37. Mirco Tribastone, Stephen Gilmore, and Jane Hillston. Scalable differential analysis of process algebra models. *IEEE Transactions on Software Engineering*, 38(1):205–219, 2012.
38. Mirco Tribastone, Stephen Gilmore, and Jane Hillston. Scalable differential analysis of process algebra models. *IEEE Trans. Software Eng.*, 38(1):205–219, 2012.
39. Max Tschaikowski and Mirco Tribastone. Exact fluid lumpability for Markovian process algebra. In *CONCUR*, LNCS, pages 380–394, 2012.
40. Max Tschaikowski and Mirco Tribastone. A unified framework for differential aggregations in Markovian process algebra. *J. Log. Algebr. Meth. Program.*, 84(2):238–258, 2015.
41. Max Tschaikowski and Mirco Tribastone. Approximate reduction of heterogenous nonlinear models with differential hulls. *IEEE Transactions on Automatic Control*, 2016.
42. Antti Valmari and Giuliana Franceschinis. Simple $o(m \log n)$ time Markov chain lumping. In *TACAS*, 2010.
43. Franck van Breugel and James Worrell. Approximating and computing behavioural distances in probabilistic transition systems. *Theoretical Computer Science*, 360(1–3):373–385, 8 2006.
44. Andrea Vandin and Mirco Tribastone. Quantitative abstractions for collective adaptive systems. In *SFM*, volume 9700 of *Lecture Notes in Computer Science*, pages 202–232. Springer, 2016.
45. Eberhard O. Voit. Biochemical systems theory: A review. *ISRN Biomathematics*, 2013:53, 2013.