IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 June 2025, accepted 1 July 2025, date of publication 4 July 2025, date of current version 16 July 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3586078

METHODS

Simulation and Analysis of Distributed Reaction Systems

LINDA BRODO[1], ROBERTO BRUNI[2], MORENO FALASCHI[3], AND ION PETRE[4]

[1]Dipartimento di Scienze Economiche e Aziendali, University of Sassari, 07100 Sassari, Italy
[2]Department of Computer Science, University of Pisa, 56127 Pisa, Italy
[3]Department of Information Engineering and Mathematics, University of Siena, 53100 Siena, Italy
[4]Department of Mathematics and Statistics, University of Turku, 20014 Turku, Finland

Corresponding author: Ion Petre (ion.petre@utu.fi)

This work was supported in part by Italian Ministero dell'Università e della Ricerca (MUR) Progetti di ricerca di Rilevante Interesse Nazionale (PRIN) 2022 Project "MEDICA" under Grant 2022RNTYWZ and Grant CUP_B53D23013170006; in part by the Next Generation EU Programme Project Piano Nazionale di Ripresa e Resilienza (PNRR) ECS00000017—"Tuscany Health Ecosystem (THE)"—Spoke 3 under Grant CUP B63C22000680007; in part by Italian MUR PRIN PNRR 2022 Project "Decentralized Ledgers in Circular Economy (DELICE)" under Grant P20223T2MF; in part by Italian MUR PRIN 2022 PNRR Project "Resource Awareness in Programming: Algebra, Rewriting, and Analysis" under Grant P2022HXNSC; in part by the Next Generation EU Programme Project PNRR "SEcurity and RIghts In the CyberSpace (SERICS)" under Grant PE00000014 and Grant CUP H73C2200089001; and in part by the Istituto Nazionale di Alta Matematica (INdAM)–Gruppo Nazionale per il Calcolo Scientifico (GNCS) Project under Grant CUP_E53C22001930001.

ABSTRACT Reaction systems (RSs) are a computational framework inspired by the interplay between biochemical interactions. Similarly to regulatory networks, reaction products can dynamically activate or inhibit other reactions. This provides a framework for discrete-time, interactive computation, where each state is determined by those reactions enabled in the immediately preceding state and by additional environmental interventions, if any. Since their introduction, RSs have been extended to study many aspects of complex systems: multi-agent collaborations, cause-effect relationships, model checking, and many others. The gap in the literature addressed in this paper is the lack of software tools for simulating and analysing distributed reaction systems (DRSs). We introduce a process algebraic approach to describing, simulating, and analysing DRSs. This allows designing complex models by re-using modular components in a well-structured and compositional way, and analysing them with the BioReSolve modelling software. We demonstrate our approach by experimenting with distributed Lotka-Volterra models, where multiple agents evolve according to their own periodic dynamics, but can also synchronise their cycles through diverse communication topologies.

INDEX TERMS BioReSolve, distributed reaction systems, Lotka-Volterra models, process algebras, reaction systems.

I. INTRODUCTION

Reaction systems (RSs) are a computational modelling framework inspired by biochemical reactions in living cells. Introduced by Ehrenfeucht and Rozenberg [1], they provide a way to model interactions in biological, chemical, or abstract systems where the behaviour is governed by principles of *enabling* and *inhibiting*, depending on the presence or absence of specific resources. A reaction system consists of a finite background set of elements (often called *entities* or *resources*) and a finite set of *reactions*. Each reaction is defined by three sets: the set of *reactants* R (resources that must be present for the reaction to be enabled), the set of *inhibitors* I (resources that must *not* be present for the reaction to be enabled), and the set of *products* P (entities produced if the reaction is enabled). Reaction systems evolve in discrete steps, interpreted as state transitions. The *current state* is defined by the set of entities present at a given moment, possibly supplemented by some *external* entities provided by the *environment* (also called *context*). A reaction is *enabled* in the current state if all entities in R are present and no entities in I are present. At each step, all and only the enabled reactions occur simultaneously, and the union of their product sets are made available in the next state. Entities

The associate editor coordinating the review of this manuscript and approving it for publication was Yizhang Jiang.

from the previous state that are not produced by any enabled reaction decay and disappear in the next state. This abstract model captures key biological phenomena, for example, gene expression and inhibition depending on the presence of transcription factors, enzymatic activity depending on the molecular environment, and regulatory mechanisms that allow cells to respond to external signals.

The reaction systems framework has been widely investigated for its mathematical properties [2] and has been employed in a variety of application areas. In systems and synthetic biology, it has been used for modelling regulatory networks [3], [4], [5], controlling signalling networks [6], and for exploring treatment strategies [7]. In molecular chemistry, it supports abstract modelling of chemical reaction networks [8]. In computer science, it has been applied to formal verification [3], [9], [10], [11], concurrency [12], [13], [14], and membrane computing [15], [16].

Compared to other computational models, the key advantages brought by reaction systems as a modelling framework are its simple and intuitive computational mechanisms—facilitation and inhibition, no counting—making it immediately understandable and widely applicable across all natural sciences. Existing approaches, such as chemical reaction networks [17], reaction-diffusion systems [18], Boolean networks [19], Petri nets [20], and stochastic process algebras [21], typically focus on capturing the emergent dynamical properties of systems, driven by concurrent resource access, quantitative configurations, and stochasticity. In contrast, reaction systems adopt a first-principles approach, where reactions can inhibit one another through their products, which explicitly function as reactants or inhibitors for other reactions. This provides a novel framework for tracing complex, multi-step cause-effect bindings in dynamical systems.

Distributed reaction systems (DRSs) build upon standard reaction systems by introducing the idea of several agents placed in multiple locations (or components), each with their own local context and local reactions, but also capable of communicating or influencing each other. Several variants of distributed (or networks of) reaction systems were introduced in [22] (full version in [11]) and further investigated in [23], [24], [25], [26], [27], [28], [29], and [30]. They link reaction systems to the rich computer science literature on distributed and concurrent systems, bringing them closer to modelling multicellular systems, tissue-level interactions, and compartmentalized systems in biology and ecology. A distributed reaction system consists of several components, with each component having its own local set of reactions and local state. Various communication mechanisms have been proposed, allowing products and even reactions to be moved across components. The context can also be local to each component, or global to the entire system.

The standard mathematical formalism for expressing, analysing, and simulating concurrent communication protocols in distributed systems is process algebra and the many process calculi built on it [31], [32]. They are very effective

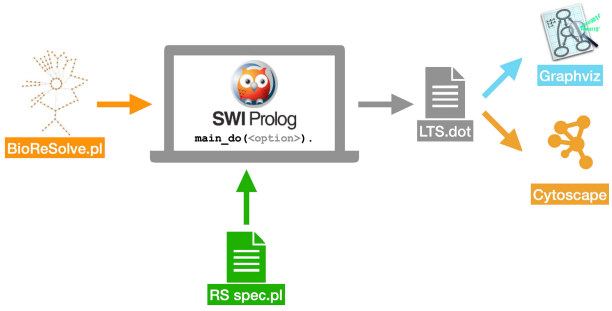

FIGURE 1. The figure illustrates the typical BioReSolve usage workflow: 1) BioReSolve.pl is the main Prolog module that contains all rules and predicates necessary for computing behavioural properties of reaction systems. After being loaded by the Prolog engine, it provides all functionalities of our framework. 2) RS spec.pl is the user-defined file that includes the reaction system specification. It encodes the specific biological model to be analysed (e.g., entities, reactions, contexts, etc.) and must be loaded together with BioReSolve. 3) The user interacts with the system by invoking the main predicate `main_do(⟨option⟩)` in the SWI-Prolog environment. 4) The result is saved to the file system, in a new, plain text, file. For LTS, the output is in DOT format, which can then be visualized using popular tools such as Graphviz (standard graph rendering and layout) or Cytoscape (particularly suited for large networks visualization).

in modelling communication and concurrency features and have been widely applied to modelling distributed systems, including biological compartments [33]. Process algebra has been used to describe standard reaction systems in [7], [13], [14], [34], [35], [36], [37], and [38].

In this paper, we introduce a *process algebraic semantics* for handling communication patterns within reaction systems, extending its applicability to distributed reaction systems. We define a notion of located reaction system, which can be executed on a single node of a network and is allowed to send and receive entities to and from its neighbours. The network is represented as a directed graph, where the direction of each arc constrains the direction of communication. Each node's local environment consists of its set of neighbours and its own context set. Following [35], we adopt an expressive notion of contexts: processes can be with guards, non-deterministic and recursive, allowing for infinite interactive computations over a finite state space. We extend the definition of our processes by adding a user-friendly syntax that supports direct communication between processes, the assignment of names to processes, and the use of a communication operator to send entities to other processes by referring to their names.

To support this framework, we developed a new prototype interpreter implemented in SWI-Prolog, and set it up as part of the BioReSolve tool, freely available at [39] both as an online interface and as downloadable code for stand-alone installation. This addresses an important gap in the blossoming research on distributed reaction systems: the lack of computational tools to support their simulation and analysis under various communication policies and interactions with the environment. BioReSolve is a rapid prototyping tool for the simulation, analysis, and verification of reaction system models, first introduced in [36] and later extended in [40].

BioReSolve implements fully non-deterministic simulation of a reaction systems model, as well as the whole state space generation. Figure 1 shows the way to interact with BioResolve. The SWI-Prolog interpreter takes as input the tool source code (BioResolve.pl) and the system model (RS spec.pl). Then the user can use one of the option for the command **main_do**(...). Different options, each consisting of a Prolog atom, can be used depending on the type of analysis or operation to be performed (e.g., simulation, reachability, trace generation, causal analysis, LTS generation). For example, the query **main_do(digraph)** generates the entire labelled transition system (LTS). The appearance of nodes, arcs and labels is fully customizable by the user, e.g., to automatically colour the nodes depending on their content, or to select which entities to display or hide inside node and arc labels. The output is in DOT format that can be visualized with Graphviz or Cytoscape tool. Besides LTS generation, BioReSolve offers several capabilities, ranging from sanity checks (presence of irrelevant entities, dead reactions, etc.) and state-space exploration of reaction systems to allow for behavioural property verification, such as reachability, cyclicity, and stability analysis. Since the tool has been designed in close connection with the syntax and operational semantics of the process algebraic version of reaction systems, it inherited some important modularity and compositionality features of the process algebraic approach that made it possible to progressively introduce novel features, such as the support for guarded contexts and several model transformations. It is intended for researchers in computational systems biology and formal methods, facilitating reproducible and symbolic analysis of RS-based models. A key usability feature of BioReSolve is its ability to represent multiple biological experiments within a single model, enabled by the rich information encoded in transition labels. It addresses an important gap in the software support for modelling with reaction systems. Despite theoretical connections to other modelling frameworks such as membrane systems [15], Petri nets [20], temporal logic [41], [42], there are only very few tools available for experimenting with reaction systems. They include HERESY [43] (https://github.com/aresio/HERESY, GPU-based simulator, no longer maintained), WebRSim [44] (https://github.com/scolobb/brsim, basic simulator, no longer maintained), ReactICS [41] (https://reactics.org, model checking tool for temporal logic properties), and an FPGA-based implementation [45]. None of them specifically supports distributed reaction systems.

As another important contribution, we have defined, implemented, and analysed the first (Distributed) RS model of a distributed Lotka-Volterra system. The model consists of multiple Lotka-Volterra models, each expressed as a reaction system, communicating with each other using custom communication graphs (defining who communicates to whom) and communication policies (defining what can be communicated). We demonstrate the expressiveness of our implementation by showing that a wide range of communication topologies and policies can be easily specified and explored. This flexibility allows for a deeper understanding of how different configurations may influence the emergent behaviours of the overall distributed system.

The paper is structured as follows. In Section II we recall the standard formulation of RSs, while in Section III we overview its process algebraic version and start introducing the distributed Lotka-Volterra model in Reaction Systems. In Section IV we introduce the principles that govern our novel process algebra for DRSs and we show how to encode them in standard RSs. In Section V we present some examples of computations between communicating agents in the context of a Lotka-Volterra prey-predator system. Related work concerning extensions of RSs with communication capabilities is discussed in Section VI. Some concluding remarks are in Section VII.

II. REACTION SYSTEMS

In the following, we use $X\#Y$ to denote the disjointness requirement between two sets X and Y: $X\#Y \triangleq (X \cap Y) = \varnothing$.

Let S be a finite set of entities called *the background set*. A *reaction* in S is a triple $a = (R, I, P)$, where $R, I, P \subseteq S$ are the sets of *reactants*, *inhibitors*, and *products*, respectively, with $R, I, P \neq \varnothing$ and $R\#I$.

A *state* of the reaction system is a subset of entities $W \subseteq S$ currently available to the system. Reaction $a = (R, I, P)$ is *enabled* in state W if $R \subseteq W$ and $I\#W$, written $en_a(W)$. Then, the *result* of applying reaction a in state W is defined as

$$res_a(W) \triangleq \begin{cases} P, & \text{if } en_a(W), \\ \varnothing & \text{otherwise.} \end{cases}$$

A *reaction system* is a pair $\mathcal{A} = (S, A)$ where S is the set of entities, and A is a finite set of reactions over S. For a state $W \subseteq S$, the result of the reactions A in W, denoted $res_A(W)$, is the union of all products of enabled reactions, i.e., $res_A(W) \triangleq \cup_{a \in A} res_a(W)$.

The dynamic behaviour of a RS $\mathcal{A} = (S, A)$ is formalized as an *interactive process*. An n-step *interactive process* in $\mathcal{A}$, with $n \geq 0$, is a pair $\pi = (\gamma, \delta)$:

- $\gamma = \{C_i\}_{i \in [0,n]}$ is called the *context sequence*, where $C_i \subseteq S$ for any $i \in [0, n]$;
- $\delta = \{D_i\}_{i \in [0,n]}$ is called the *result sequence*, where $D_i \subseteq S$ for any $i \in [0, n]$, with $D_0 = \varnothing$ and $D_{i+1} = res_A(D_i \cup C_i)$ for any $i \in [0, n-1]$.

We let $W_i \triangleq C_i \cup D_i$ for any $i \in [0, n]$ and we call $\tau \triangleq W_0, \ldots, W_n$ the *state sequence*.

A. RUNNING EXAMPLE: A REACTION SYSTEM-BASED LOTKA-VOLTERRA MODEL

The celebrated Lotka-Volterra (LV) model originated from chemistry [46], [47] and ecology [48], and it found applications in competitive dynamical systems of many types, especially in economics [49]. The model describes

the evolution of two species whose dynamics depend on each other's level. It is usually described in terms of a predator-prey population model. Throughout this paper we will use 'foxes' to refer to the predator population and 'rabbits' to refer to the prey population. In the basic variant of the model that we follow in this paper, the prey/rabbit population ρ is assumed to have access to an unlimited amount of food, and their multiplication is unhindered. On the other hand, the multiplication of the predator/fox population ϕ is assumed to depend on the availability of the prey population. The model is typically described using the following chemical reaction network:

$$\begin{aligned}
&\textit{(Prey multiplies)} && \rho \xrightarrow{a} 2 * \rho;\\
&\textit{(Prey being hunted)} && \rho + \phi \xrightarrow{b} \phi;\\
&\textit{(Predator multiplies)} && \rho + \phi \xrightarrow{c} \rho + 2 * \phi;\\
&\textit{(Predator dies)} && \phi \xrightarrow{d} \varnothing,
\end{aligned}$$

where a, b, c, d are the kinetic rate constants of each reaction and ρ, ϕ are the variables for the prey and predator populations, resp.

We introduce a discrete version of the Lotka-Volterra model based on reaction systems. This example will be used throughout this paper.

Example 1 (A Simple RS for Lotka-Volterra): We discretize the rabbit and the fox populations into only two levels, one denoting a "low" level (entities r and f, resp.) and the other denoting a "high" level (entities R and F, resp.) The model consists of the following four inhibitor-free reactions over the background set $S = \{\mathsf{r}, \mathsf{f}, \mathsf{R}, \mathsf{F}\}$:

$$\begin{aligned}
&\textit{(Prey multiplies)} && a_1 \triangleq (\{\mathsf{r}, \mathsf{f}\}, \varnothing, \{\mathsf{R}, \mathsf{f}\});\\
&\textit{(Prey being hunted)} && a_2 \triangleq (\{\mathsf{R}, \mathsf{F}\}, \varnothing, \{\mathsf{r}, \mathsf{F}\});\\
&\textit{(Predator multiplies)} && a_3 \triangleq (\{\mathsf{R}, \mathsf{f}\}, \varnothing, \{\mathsf{R}, \mathsf{F}\});\\
&\textit{(Predator dies)} && a_4 \triangleq (\{\mathsf{r}, \mathsf{F}\}, \varnothing, \{\mathsf{r}, \mathsf{f}\}).
\end{aligned}$$

The model indeed exhibits the expected Lotka-Volterra periodic behaviour, as seen in its context-free state-transition graph Figure 2(a): starting from any combination of low/high levels of rabbits and foxes, the model cycles through 4 states in which the two variables switch between low and high.

III. A PROCESS ALGEBRA OF REACTION SYSTEMS

In this section we briefly recall the algebraic syntax and the formal semantics for RSs based on SOS inference rules, as introduced in [36]. We adopt the algebraic variant of reaction systems that supports the use of guarded contexts [50]. We provide only an intuitive overview, as a detailed presentation of the SOS rules and their induced semantics is not essential for the purposes of this work. A comprehensive account is in [36].

Definition 1 (RS Processes): Let S be a finite background set. An *RS process* P is any term defined by the following grammar:

$$\begin{aligned}
\mathsf{P} &:= [\mathsf{M}]\\
\mathsf{M} &:= (R, I, P) \mid D \mid \mathsf{K} \mid \mathsf{M}|\mathsf{M}\\
\mathsf{K} &:= \mathbf{0} \mid (R, I, C).\mathsf{K} \mid \mathsf{K} + \mathsf{K} \mid \mathsf{A}
\end{aligned}$$

where $R, I, P, D, C \subseteq S$, $R\#I$ and A are identifiers belonging to a predefined family $\Delta = \{\mathsf{A}_i \triangleq \mathsf{K}_i\}_{i\in I}$, called the *environment*, of recursive context definitions.

Without loss of generality, we allow the sets R, I, P to be empty in Definition 1, as auxiliary symbols can always be introduced to represent entities that are permanently present or absent, if required.

An RS process P encloses a *mixture* process M composed of the parallel composition of some reactions (R, I, P), some sets of entities D (possibly empty), and some *context* K. We write $\prod_{j\in J} \mathsf{M}_j$ for the parallel composition of all M_j with $j \in J$.

A context process K is a non-deterministic and recursive system. The nil context $\mathbf{0}$ stops the computation. Given the current set of entities D, a guarded prefix $(\mathsf{R}, \mathsf{I}, \mathsf{C}).\mathsf{K}$ is used to check if $R \subseteq D$ and $I\#D$, in other words, whether all entities in R are present and none from I are present in D. In this case, the process contributes the entity set C and the continuation K will represent the context at the next step. Otherwise, it behaves as the nil context. The syntax of the guarded prefix is intentionally similar to that of reactions, as it checks for the presence of reactants R and absence of inhibitors I with respect to the current entities D, in order to provide the entities in C. When both R and I are empty, we use the shorthand $C.\mathsf{K}$. The non-deterministic choice $\mathsf{K}_1 + \mathsf{K}_2$ allows the context to behave as either K_1 or K_2. Finally, the constant process A behaves as determined by K if its definition $\mathsf{A} \triangleq \mathsf{K}$ is present in the environment Δ.

We say that P and P' are structurally equivalent, written $\mathsf{P} \equiv \mathsf{P}'$, when they denote the same term up to the laws of commutative monoids (unit, associativity and commutativity) for parallel composition $\cdot|\cdot$, with $\varnothing$ as the unit, and the laws of idempotent and commutative monoids for choice $\cdot + \cdot$, with $\mathbf{0}$ as the unit. We also assume $D_1|D_2 \equiv D_1 \cup D_2$ for any $D_1, D_2 \subseteq S$. Under these assumptions, any process M can always be written in a standard form

$$D \mid (\prod_{k\in K} \mathsf{K}_k) \mid (\prod_{j\in J} (R_j, I_j, P_j))$$

for some suitable sets of reactions $\{(R_j, I_j, P_j) \mid j \in J\}$ and contexts $\{\mathsf{K}_k \mid k \in K\}$.

Definition 2 (RS Processes): Let $\mathcal{A} = (S, A)$ be a RS, and $\pi = (\gamma, \delta)$ an n-step interactive process in $\mathcal{A}$, with $\gamma = \{C_i\}_{i\in[0,n]}$ and $\delta = \{D_i\}_{i\in[0,n]}$. For any step $i \in [0, n]$, the corresponding RS process $[\![\mathcal{A}, \pi]\!]_i$ is defined as follows:

$$[\![\mathcal{A}, \pi]\!]_i \triangleq \left[D_i \mid \mathsf{K}_{\gamma^i} \mid \prod_{a\in A} a \right]$$

where $\mathsf{K}_{\gamma^i} \triangleq C_i.C_{i+1}.\cdots.C_n.\mathbf{0}$ is the serialization of the entities offered by γ^i (the shifting of γ at the i-th step). We write $[\![\mathcal{A}, \pi]\!]$ as a shorthand for $[\![\mathcal{A}, \pi]\!]_0$.

The operational semantics of RS processes is compositionally defined by SOS inference rules in Table 6 in

the Appendix. This semantics induces a labelled transition system, whose nodes are RS processes P and whose transition labels ℓ carry some information about the conditions under which the transition, from the source state to the target states, is possible. Each rule defines the behaviour of a single syntax operator: the premises in the upper part of the inference rule describe the conditions to be verified inductively for the operator to act, while the lower part describes the transition that can occur if the premises are satisfied. As an example, the rule (*Par*) is applied when two sub-processes are executed concurrently and verifies that their transition labels are compatible (written $\ell_1 \frown \ell_2$). Axioms, like rules (*Ent*) and (*Pro*), have no premises to be verified in the upper part and are thus always applicable. Rule (*Sys*) is responsible for verifying that all the rules associated with each operator that is present in the current state configuration have been executed and that all reactants that were required by subprocesses were indeed available. Roughly, a transition label ℓ records all information related to the underlying step, like the sets D of entities currently in the system and the overall set of entities C that are provided by context processes. For the purposes of this paper, it is not relevant to know the details of the label syntax, thus we skip their description and in the following examples we report just their underlying set of entities provided by the context or just omit them. Moreover, when the whole LTS of a RS process is shown, we further simplify the notation: by noticing that reactions are persistent we omit them from node names.

From [36, Theorem 19] it is known that the SOS semantics of an RS process matches the set-theoretic dynamics of its underlying Reaction System. More precisely, for any Reaction System $\mathcal{A} = (S, A)$ and any interactive process $\pi = (\gamma, \delta)$ in $\mathcal{A}$, with $\gamma = \{C_i\}_{i\in[0,n]}$ and $\delta = \{D_i\}_{i\in[0,n]}$, any outgoing transition from $[\![\mathcal{A}, \pi]\!]_i$ leads to $[\![\mathcal{A}, \pi]\!]_{i+1}$. According to the above simplifications about transition labels and node naming, the corresponding transitions are written

$$\left[D_i \mid \mathsf{K}_{\gamma^i}\right] \xrightarrow{C_i} \left[D_{i+1} \mid \mathsf{K}_{\gamma^{i+1}}\right].$$

The following example illustrates how the LTS semantics works exploiting the already mentioned Lotka-Volterra (LV) model for describing the evolution of two species whose dynamics depend on the number of each other.

Example 2: Following our running Example 1, the background set $S = \{\mathsf{f}, \mathsf{F}, \mathsf{r}, \mathsf{R}\}$ contains the four entities for describing the evolution of a system populated by foxes and rabbits, where f and r denote a low level of foxes and rabbits, respectively, and F and R denote a high level of foxes and rabbits, respectively. Here the context gives no entities: $\mathsf{K} \triangleq \varnothing.\mathsf{K}$, thus the dynamics is only regulated by the reactions. In the following we let $\mathsf{Rct} = a_1 \mid a_2 \mid a_3 \mid a_4$ denote the parallel composition of the four reaction of the model in Example 1. Assuming our initial state contains a few foxes and rabbits, then its configuration is: $\mathsf{P}_0 \triangleq [\{\mathsf{f}, \mathsf{r}\} \mid \mathsf{K} \mid \mathsf{Rct}]$. The execution of P_0 generates a four-state cycle:

$$\mathsf{P}_0 \xrightarrow{\varnothing} \mathsf{P}_1 \xrightarrow{\varnothing} \mathsf{P}_2 \xrightarrow{\varnothing} \mathsf{P}_3 \xrightarrow{\varnothing} \mathsf{P}_0 \xrightarrow{\varnothing} \cdots$$

where $\mathsf{P}_1 \triangleq [\{\mathsf{f}, \mathsf{R}\} \mid \mathsf{K} \mid \mathsf{Rct}]$, $\mathsf{P}_2 \triangleq [\{\mathsf{F}, \mathsf{R}\} \mid \mathsf{K} \mid \mathsf{Rct}]$, and $\mathsf{P}_3 \triangleq [\{\mathsf{F}, \mathsf{r}\} \mid \mathsf{K} \mid \mathsf{Rct}]$. Starting in P_0, where there is a low number foxes (i.e., the predators), the population of rabbits can grow in P_1 while the population of foxes remains small. Then, in P_1 there is plenty of food for foxes, i.e., a high number of rabbits, their population grows in P_2, while the population of rabbits remains large. In P_3 the abundance of the predators will decrease the population of rabbits. This is the cause of the decrement of the fox population that brings the system back to P_0, and from now on the dynamics will be cyclic. Since the recursive context K and the set of reactions Rct do not change over time we further abbreviate the notation and write (see Figure 2(a), for a graphical view):

$$[\{\mathsf{f}, \mathsf{r}\}] \to [\{\mathsf{f}, \mathsf{R}\}] \to [\{\mathsf{F}, \mathsf{R}\}] \to [\{\mathsf{F}, \mathsf{r}\}] \to [\{\mathsf{f}, \mathsf{r}\}] \to \cdots$$

In the next example, we show what happens when a different context recursively provides a small number of rabbits at each step.

Example 3: Consider the RS of the Example 2. We model the context that recursively introduces some extra number of rabbits at each step: $\mathsf{K} \triangleq \{\mathsf{xr}\}.\mathsf{K}$, where we introduce a new element xr to be distinguished from the reaction product r. Introducing additional rabbits induces the configuration of the system to change to a high level of rabbits, regardless of what their level was to start with. In other words, having $\{\mathsf{r}, \mathsf{xr}\}$ in the current state is considered equivalent to having $\{\mathsf{R}\}$. To account for this conceptual equivalence, the reactions are extended as follows:

(Prey multiplies)	$a_1 \triangleq (\{\mathsf{r}, \mathsf{f}\}, \{\mathsf{xr}\}, \{\mathsf{R}, \mathsf{f}\})$;
(Prey being hunted)	$a_2 \triangleq (\{\mathsf{R}, \mathsf{F}\}, \varnothing, \{\mathsf{r}, \mathsf{F}\})$;
	$a_2' \triangleq (\{\mathsf{r}, \mathsf{xr}, \mathsf{F}\}, \varnothing, \{\mathsf{r}, \mathsf{F}\})$;
(Predator multiplies)	$a_3 \triangleq (\{\mathsf{R}, \mathsf{f}\}, \varnothing, \{\mathsf{R}, \mathsf{F}\})$;
	$a_3' \triangleq (\{\mathsf{r}, \mathsf{xr}, \mathsf{f}\}, \varnothing, \{\mathsf{R}, \mathsf{F}\})$;
(Predator dies)	$a_4 \triangleq (\{\mathsf{r}, \mathsf{F}\}, \{\mathsf{xr}\}, \{\mathsf{r}, \mathsf{f}\})$.

Now, let $\mathsf{Rct}' = a_1 \mid a_2 \mid a_2' \mid a_3 \mid a_3' \mid a_4$ and $\mathsf{P}_0 \triangleq [\{\mathsf{r}, \mathsf{f}\} \mid \mathsf{K} \mid \mathsf{Rct}']$. Due to the activity of the context, in P_0 only reaction a_3' is enabled as the combination of r and xr amounts to the presence of a large number of rabbits. The following evolution of the system is characterised by the constant presence of a large number of rabbits, either as a direct product of the reaction execution or as an effect of the provision by context (see Figure 2(a) for a graphical view of the system evolution):

$$[\{\mathsf{f}, \mathsf{r}\}] \xrightarrow{\{\mathsf{xr}\}} [\{\mathsf{F}, \mathsf{R}\}] \xrightarrow{\{\mathsf{xr}\}} [\{\mathsf{F}, \mathsf{r}\}] \xrightarrow{\{\mathsf{xr}\}} [\{\mathsf{F}, \mathsf{r}\}] \xrightarrow{\{\mathsf{xr}\}} \cdots$$

Our context can also behave non-deterministically, and in the next example we exploit this possibility letting the context non-deterministically provide the set $\{\mathsf{xr}\}$ or the empty set.

Example 4: Consider the RS of the Example 3 where we change the context behaviour such that at each step it can either provide some extra rabbits or none: $\mathsf{K}' \triangleq \{\mathsf{xr}\}.\mathsf{K}' + \varnothing.\mathsf{K}'$. Given the non-deterministic behaviour of the context, from $\mathsf{P}_0 \triangleq [\{\mathsf{r}, \mathsf{f}\} \mid \mathsf{K}' \mid \mathsf{Rct}']$ there will now be two possible branches, depending on what the context

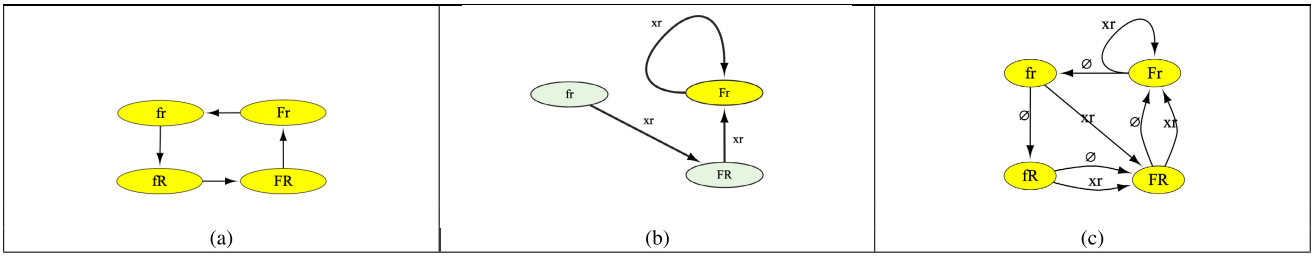

FIGURE 2. The dynamics of a single-agent reaction system-based Lotka-Volterra model indicated through its state-transition graph. Each node in the graph represents a model state, defined by the levels of foxes ('f' for low, 'F' for high) and rabbits ('r' for low, 'R' for high). Each edge represents a state transition. In yellow we indicate the nodes corresponding to cycles in the graph. (a) Under a context-free evolution, i.e., with no contribution from the environment, the model cycles through its four states, switching between low/high levels of rabbits and foxes. This is the expected Lotka-Volterra periodic model behaviour. (b) When the environment offers a constant surplus of rabbits (Example 3), indicated by xr on the state transitions, the model converges into a state where the surplus of rabbits is turned into a high level of foxes and a low level of rabbits. This is in accordance with the model rule 'Predator multiplies': from a state with high levels of both rabbits and foxes, the system transitions into a state with high levels of foxes and low levels of rabbits. The surplus of rabbits is not visible in the state because the model rules transforms it immediately into a high level of foxes and a low level of rabbits. (c) The non-deterministic dynamic of the model under an environment that may occasionally introduce a surplus of rabbits (Example 4). For example, from state 'fr' the model will transition to state 'fR' with an empty environment and to state 'FR' if the environment introduces a surplus of rabbits 'xr'.

will provide: the set $\{\mathsf{xr}\}$ or the empty set. In the first case, the system will evolve in the state $[\{\mathsf{R},\mathsf{F}\}\,|\,\mathsf{K}'\,|\,\mathsf{Rct}']$ by firing reaction a_3'; in the second case the context introduces the empty set, and the system will evolve in the state $[\{\mathsf{R},\mathsf{f}\}\,|\,\mathsf{K}'\,|\,\mathsf{Rct}']$ by firing reaction a_1. The subsequent evolution is depicted graphically in Figure 2(b), where we label the edges with $\{\mathsf{xr}\}$ or with $\varnothing$ depending on what the context provides and where the always present components K' and Rct' are omitted from node names. Please note that in the graph in Figure 2(b), the transitions labelled with the symbol $\varnothing$ draw a graph that corresponds to the graph in Figure 2(a).

To show how guarded contexts can be exploited, in the next example, at each step, the context will introduce xr only when the current state has a low level of rabbits $\{r\}$.

Example 5: Consider the RS of the Example 3, where we replace the purely non-deterministic context K' by the guarded context $\mathsf{K}'' \triangleq (\{\mathsf{r}\},\varnothing,\{\mathsf{xr}\}).\mathsf{K}'' + (\varnothing,\{\mathsf{r}\},\varnothing)).\mathsf{K}''$. The new context behaves in a slightly different way: it recursively checks at each step if in the system there are few rabbits, and only in that case it provides the set $\{\mathsf{xr}\}$. Letting $\mathsf{P}_0 \triangleq [\{\mathsf{f},\mathsf{r}\}\,|\,\mathsf{K}''\,|\,\mathsf{Rct}']$, we get the (deterministic) transition system

$$[\{\mathsf{f},\mathsf{r}\}] \xrightarrow{\{\mathsf{xr}\}} [\{\mathsf{F},\mathsf{R}\}] \xrightarrow{\varnothing} [\{\mathsf{F},\mathsf{r}\}] \xrightarrow{\{\mathsf{xr}\}} [\{\mathsf{F},\mathsf{r}\}] \xrightarrow{\{\mathsf{xr}\}} \cdots$$

Notice that, when we omit the transition labels, the LTS is the same as the one in Figure 2(a), Example 3.

IV. A PROCESS ALGEBRA OF DISTRIBUTED RS

The distributed version of reaction systems we intend to model consists of a static network where each node hosts a single, local RS process whose reactions can send products to neighbour nodes. Communication is asynchronous and transmitted products will be available at the next time instant. At the level of syntax, we assume each node of the network is uniquely identified by its *location name* n, drawn from a given set $\mathcal{N}$. Location names can be seen as some sort of networking address and are attached to RS processes: the term $n[\mathsf{M}]$ represents a located process identified by $n \in \mathcal{N}$. Location names are also used to specify the destination of some products of reactions: the notation $n[\mathsf{e}]$ denotes the product e assigned for delivery to location n. As a shorthand, we write $n[P]$ for the located set of entities $\{n[\mathsf{e}] \mid \mathsf{e} \in P\}$.

Definition 3 (DRS Processes): Let S be a finite background set and $\mathcal{N}$ be a finite set of location names. We let $\mathcal{N}[S] \triangleq \{n[\mathsf{e}] \mid n \in \mathcal{N}, \mathsf{e} \in S\}$ be the corresponding set of located entities and $\hat{S} \triangleq S \cup \mathcal{N}[S]$ the set of admissible products. A *DRS process* P is any term defined by the following grammar:

$$\begin{aligned}\mathsf{P} &:= n[\mathsf{M}] \mid \mathsf{P}|\mathsf{P}\\ \mathsf{M} &:= (R,I,P) \mid D \mid \mathsf{K} \mid \mathsf{M}|\mathsf{M}\\ \mathsf{K} &:= \mathbf{0} \mid (R,I,C).\mathsf{K} \mid \mathsf{K}+\mathsf{K} \mid \mathsf{A}\end{aligned}$$

where $R, I, C \subseteq S, D, P \subseteq \hat{S}$ and A is an identifier belonging to a predefined family $\Delta = \{\mathsf{A}_i \triangleq \mathsf{K}_i\}_{i\in I}$ of recursive context definitions.

A located RS process P is defined as a parallel composition of located RSs, $n[\mathsf{M}]$, each one composed by the parallel composition of some local reactions, some local sets of entities D (possibly empty), and some local context K. There are three key differences w.r.t Definition 1: first, each local process M is executed within a separate container $n[\cdot]$; second, the products of reactions can contain both ordinary entities e and their located versions $n[\mathsf{e}]$ for some destination n; third, located entities are also allowed to appear in the local state, because $D \subseteq \hat{S}$. The third extension is just a technical convenience for allowing the local production of a located element $n[\mathsf{e}]$ within the current container $m[\cdot]$ even if it is targeted to a different destination $n \neq m$. In fact, DRS processes are subject to the following congruences:

$$n[\mathsf{M}|\mathsf{M}'] \equiv n[\mathsf{M}]|n[\mathsf{M}'], n[m[D]] \equiv m[D],$$

with $n, m \in \mathcal{N}$.

We remark that both reactions (R, I, P) and guarded context prefixes $(R, I, C).\mathsf{K}$ can only inspect local states, because $R, I \subseteq S$ cannot involve located entities, contrary to $P \subseteq \hat{S}$. Similarly, local contexts can only produce local entities, because $C \subseteq S$ in $(R, I, C).\mathsf{K}$.

As usual, any process M can always be written in a standard form

$$D \mid (\prod_{k \in K} \mathsf{K}_k) \mid (\prod_{j \in J} (R_j, I_j, P_j))$$

and we call M a *plain* process when $D \subseteq S$. Correspondingly, any DRS process P can always be written in a standard form $\prod_{n \in \mathcal{N}} n[\mathsf{M}_n]$, where each process M_n is plain.

The semantics is defined by the inference rules in Table 6 together with the inference rules in Table 7, where rule (*Sys*) in Table 6 is replaced by rule (*Loc*) in Table 7. The inference rules induce a labelled transition system; as before we leave the full technical details about transition labels to the appendix, as they are not relevant for the discussion of our main case study in Section V.

A. COMMUNICATION TOPOLOGY IN DRS

Given a DRS process $\mathsf{P} = \prod_{n \in \mathcal{N}} n[\mathsf{M}_n]$ where each M_n is plain, we denote by $msg(\mathsf{M}_n)$ the set of possible outgoing *messages* of M_n. Formally:

$$\begin{aligned} msg((R, I, P)) &\triangleq P \cap \mathcal{N}[S] \\ msg(\mathsf{K}) &\triangleq \varnothing \\ msg(D) &\triangleq \varnothing \\ msg(\mathsf{M}|\mathsf{M}') &\triangleq msg(\mathsf{M}) \cup msg(\mathsf{M}') \end{aligned}$$

Then, the *network* of P is the labelled directed graph $(\mathcal{N}, \rightarrow)$ whose set of nodes is $\mathcal{N}$ and whose transition relation $\rightarrow \subseteq \mathcal{N} \times S \times \mathcal{N}$ is such that:

$$n \xrightarrow{\mathsf{e}} m \qquad \text{iff} \qquad m[\mathsf{e}] \in msg(\mathsf{M}_n)$$

To some extent, the presence of the arc $n \xrightarrow{\mathsf{e}} m$ means that the location n can act as a context for location m by producing some additional entities for m through reactions in n and thus n can influence the behaviour of m. When n and m are connected in both directions it means that a mutual influence is possible. In the next example, we deploy two copies of the same reaction system in two different nodes that can communicate each other. As a matter of notation, in the following we denote by $\mathsf{P}[m/n]$ the DRS process obtained by replacing with m all the occurrences of name n that are in P.

Example 6: Let us consider the reactions introduced in Example 3. We extend them in such a way that whenever a large number of rabbits is produced, some extra rabbits will reach a neighbour location n. Correspondingly, the reactions can be rewritten as follows:

(Prey multiplies)	$a_1 \triangleq (\{\mathsf{r}, \mathsf{f}\}, \{\mathsf{xr}\}, \{\mathsf{R}, \mathsf{f}, n[\mathsf{xr}]\});$
(Prey being hunted)	$a_2 \triangleq (\{\mathsf{R}, \mathsf{F}\}, \varnothing, \{\mathsf{r}, \mathsf{F}\});$
	$a_2' \triangleq (\{\mathsf{r}, \mathsf{xr}, \mathsf{F}\}, \varnothing, \{\mathsf{r}, \mathsf{F}\});$
(Predator multiplies)	$a_3 \triangleq (\{\mathsf{R}, \mathsf{f}\}, \varnothing, \{\mathsf{R}, \mathsf{F}, n[\mathsf{xr}]\});$
	$a_3' \triangleq (\{\mathsf{r}, \mathsf{xr}, \mathsf{f}\}, \varnothing, \{\mathsf{R}, \mathsf{F}, n[\mathsf{xr}]\});$
(Predator dies)	$a_4 \triangleq (\{\mathsf{r}, \mathsf{F}\}, \{\mathsf{xr}\}, \{\mathsf{r}, \mathsf{f}\}).$

The above reactions can be used to define a local process $\mathsf{M} \triangleq \{\mathsf{r}, \mathsf{f}\} \mid \mathsf{Rct}'$, where let $\mathsf{Rct}' = a_1 \mid a_2 \mid a_2' \mid a_3 \mid a_3' \mid a_4$. Since the context has not a role in this example, we just omit it. In order to deploy two copies of M in two different locations 1 and 2, let us define $\mathsf{Rct}_i' \triangleq \mathsf{Rct}'[i/n]$ and $\mathsf{M}_i \triangleq \mathsf{M}[i/n] = \{\mathsf{r}, \mathsf{f}\} \mid \mathsf{Rct}_i'$ for $i \in \{1, 2\}$. Then, we let the DRS process

$$\mathsf{P}_0 \triangleq 1[M_2] \mid 2[M_1] = 1[\{\mathsf{r}, \mathsf{f}\} \mid \mathsf{Rct}_2'] \mid 2[\{\mathsf{r}, \mathsf{f}\} \mid \mathsf{Rct}_1'].$$

At the beginning, only reaction a_1 is enabled in both locations, and therefore we have the transition

$$1[\{\mathsf{r}, \mathsf{f}\}] \mid 2[\{\mathsf{r}, \mathsf{f}\}] \xrightarrow{\varnothing} 1[\{\mathsf{R}, \mathsf{f}, 2[\mathsf{xr}]\}] \mid 2[\{\mathsf{R}, \mathsf{f}, 1[\mathsf{xr}]\}]$$

where, as usual, we omit to mention Rct_1' and Rct_2' because they are always present and unchanged. Exploiting structural congruence, the target process can be rewritten in the form

$$\begin{aligned} &1[\{\mathsf{R}, \mathsf{f}, 2[\mathsf{xr}]\}] \mid 2[\{\mathsf{R}, \mathsf{f}, 1[\mathsf{xr}]\}] \\ &\quad \equiv 1[\{\mathsf{R}, \mathsf{f}\}] \mid 1[\{2[\mathsf{xr}]\}]] \mid 2[\{\mathsf{R}, \mathsf{f}\}] \mid 2[\{1[\mathsf{xr}]\}]] \\ &\quad \equiv 1[\{\mathsf{R}, \mathsf{f}\}] \mid \{2[\mathsf{xr}]\}] \mid 2[\{\mathsf{R}, \mathsf{f}\}] \mid \{1[\mathsf{xr}]\}] \\ &\quad \equiv 1[\{\mathsf{R}, \mathsf{f}, \mathsf{xr}\}] \mid 2[\{\mathsf{R}, \mathsf{f}, \mathsf{xr}\}] \end{aligned}$$

At the next step, only reaction a_3 is enabled in both locations. Thus:

$$1[\{\mathsf{R}, \mathsf{f}, \mathsf{xr}\}] \mid 2[\{\mathsf{R}, \mathsf{f}, \mathsf{xr}\}] \xrightarrow{\varnothing} 1[\{\mathsf{R}, \mathsf{F}, \mathsf{xr}\}] \mid 2[\{\mathsf{R}, \mathsf{F}, \mathsf{xr}\}]$$

where in the target configuration we have already exploited structural congruence to embed directly the extra rabbits produced in one location to their destination.

At the next step, only reaction a_2 is enabled in both locations:

$$1[\{\mathsf{R}, \mathsf{F}, \mathsf{xr}\}] \mid 2[\{\mathsf{R}, \mathsf{F}, \mathsf{xr}\}] \xrightarrow{\varnothing} 1[\{\mathsf{r}, \mathsf{F}\}] \mid 2[\{\mathsf{r}, \mathsf{F}\}]$$

This time, no extra rabbits are supplied to neighbour locations, so that only reaction a_4 is now enabled, leading to the transition

$$1[\{\mathsf{r}, \mathsf{F}\}] \mid 2[\{\mathsf{r}, \mathsf{F}\}] \xrightarrow{\varnothing} 1[\{\mathsf{r}, \mathsf{f}\}] \mid 2[\{\mathsf{r}, \mathsf{f}\}]$$

that brings the system back to its original configuration P_0. The behaviour is summarised by the central loop in Figure 3(a), where it is also shown what does it happen if any possible combination of initial configurations are chosen for the two systems. Notably, no matter which initial configuration is considered, after a few transitions the populations of preys and predators in both locations will "synchronise", reaching the central loop we have just described.

B. FLATTENING

Although DRSs provide a suitable framework for composing communicating RSs along different patterns, and to study their emergent behaviours, it can be shown that they are as much expressive as ordinary RSs. More precisely, for every DRS process P over the set of names $\mathcal{N}$ and entities S, we can define an RS process $\mathcal{F}(\mathsf{P})$ over the background set $\hat{S}$ that exhibits an isomorphic LTS. Vice versa, the behaviour of any RS process $\mathsf{P} = [\mathsf{M}]$ is isomorphic to the behaviour of the DRS process $n[\mathsf{P}]$ (up to the obvious renaming of entities that sends e to $n[\mathsf{e}]$). These correspondences are formalized by Theorem 8 in the Appendix.

Showing that a distributed reaction system can be mapped to an equivalent reaction systems process is important to show that interactions between agents may be embedded in a reaction systems process. However, it also shows two practical remarks: (1) distributed reaction systems allow for more compact model specifications and are more readable for communication than reaction systems processes, and (2) flattening can be useful for implementing distributed reaction systems on top of an existing implementation of reaction systems processes.

The mapping $\mathcal{F}(\cdot)$, from DRS processes to RS processes, is called the *flattening* and it is defined as follows:

$$\mathcal{F}(\prod_{n\in\mathcal{N}} n[\mathsf{M}_n]) \triangleq \left[\prod_{n\in\mathcal{N}} \mathcal{F}(n, \mathsf{M}_n)\right]$$

The auxiliary function $\mathcal{F}(\cdot,\cdot)$ is in turn defined as follows:

$$\begin{aligned}
\mathcal{F}(n, \mathsf{M}_1|\mathsf{M}_2) &\triangleq \mathcal{F}(n, \mathsf{M}_1)|\mathcal{F}(n, \mathsf{M}_2)\\
\mathcal{F}(n, D) &\triangleq n[D]\\
\mathcal{F}(n, (R, I, P)) &\triangleq (n[R], n[I], n[P])\\
\mathcal{F}(n, \mathbf{0}) &\triangleq \mathbf{0}\\
\mathcal{F}(n, (R, I, C).\mathsf{K}) &\triangleq (n[R], n[I], n[C]).\mathcal{F}(n, \mathsf{K})\\
\mathcal{F}(n, \mathsf{K} + \mathsf{K}') &\triangleq \mathcal{F}(n, \mathsf{K}) + \mathcal{F}(n, \mathsf{K}')\\
\mathcal{F}(n, \mathsf{A}) &\triangleq \mathsf{A}_n\\
\mathcal{F}(\{\mathsf{A}_i \triangleq \mathsf{K}_i\}_{i\in I}) &\triangleq \bigcup_{n\in\mathcal{N}} \{\mathsf{A}_{i,n} \triangleq \mathcal{F}(n, \mathsf{K}_i)\}_{i\in I}
\end{aligned}$$

where, as usual, we assume that $m[n[\mathsf{e}]] = n[\mathsf{e}]$ and that for each definition $\mathsf{A} \triangleq \mathsf{K}$ in the environment Δ, there is a corresponding definition $\mathsf{A}_n \triangleq \mathcal{F}(n, \mathsf{K})$ in the environment $\mathcal{F}(\Delta)$.

Roughly, the flattening transforms a generic DRS process $\prod_n n[\mathsf{M}_n]$ to the RS process $\left[\prod_n \mathcal{F}(n, \mathsf{M}_n)\right]$, where all located processes M_n can be executed together within the same group $[\cdot]$ because the names of their locations are now embedded in the names of the entities.

Example 7: Let us consider the DRS P_0 from Example 6. The flattened process $\mathcal{F}(\mathsf{P}_0)$ is the RS process $[\{1[\mathsf{r}], 1[\mathsf{f}], 2[\mathsf{r}], 2[\mathsf{f}]\} \mid \mathsf{Rct}_{1,2}]$, where $\mathsf{Rct}_{1,2}$ is the parallel composition of all the following reactions.

(Prey mult.) $a_{1,1} \triangleq (\{1[\mathsf{r}], 1[\mathsf{f}]\}, \{1[\mathsf{xr}]\}, \{1[\mathsf{R}], 1[\mathsf{f}], 2[\mathsf{xr}]\})$;
$a_{1,2} \triangleq (\{2[\mathsf{r}], 2[\mathsf{f}]\}, \{2[\mathsf{xr}]\}, \{2[\mathsf{R}], 2[\mathsf{f}], 1[\mathsf{xr}]\})$;
(Prey hunt.) $a_{2,1} \triangleq (\{1[\mathsf{R}], 1[\mathsf{F}]\}, \varnothing, \{1[\mathsf{r}], 1[\mathsf{F}]\})$;
$a'_{2,1} \triangleq (\{1[\mathsf{r}], 1[\mathsf{xr}], 1[\mathsf{F}]\}, \varnothing, \{1[\mathsf{r}], 1[\mathsf{F}]\})$;
$a_{2,2} \triangleq (\{2[\mathsf{R}], 2[\mathsf{F}]\}, \varnothing, \{2[\mathsf{r}], 2[\mathsf{F}]\})$;
$a'_{2,2} \triangleq (\{2[\mathsf{r}], 2[\mathsf{xr}], 2[\mathsf{F}]\}, \varnothing, \{2[\mathsf{r}], 2[\mathsf{F}]\})$;
(Pred. mult.) $a_{3,1} \triangleq (\{1[\mathsf{R}], 1[\mathsf{f}]\}, \varnothing, \{1[\mathsf{R}], 1[\mathsf{F}], 2[\mathsf{xr}]\})$;
$a'_{3,1} \triangleq (\{1[\mathsf{r}], 1[\mathsf{xr}], 1[\mathsf{f}]\}, \varnothing, \{1[\mathsf{R}], 1[\mathsf{F}], 2[\mathsf{xr}]\})$;
$a_{3,2} \triangleq (\{2[\mathsf{R}], 2[\mathsf{f}]\}, \varnothing, \{2[\mathsf{R}], 2[\mathsf{F}], 1[\mathsf{xr}]\})$;
$a'_{3,2} \triangleq (\{2[\mathsf{r}], 2[\mathsf{xr}], 2[\mathsf{f}]\}, \varnothing, \{2[\mathsf{R}], 2[\mathsf{F}], 1[\mathsf{xr}]\})$;
(Pred. dies) $a_{4,1} \triangleq (\{1[\mathsf{r}], 1[\mathsf{F}]\}, \{1[\mathsf{xr}]\}, \{1[\mathsf{r}], 1[\mathsf{f}]\})$;
$a_{4,2} \triangleq (\{2[\mathsf{r}], 2[\mathsf{F}]\}, \{2[\mathsf{xr}]\}, \{2[\mathsf{r}], 2[\mathsf{f}]\})$.

The above example witnesses the conceptual advantages offered by the concept of DRS in terms of modularity, conciseness, re-usability, understandability and flexibility w.r.t. a direct modeling of complex scenarios in terms of ordinary RSs.

V. CASE-STUDY: THE LOTKA-VOLTERRA MODEL

In this section we investigate the applicability of DRS to the modelling of multi-agent systems of increasing complexity. To this aim we build on the running example of Lotka-Volterra models for predator-prey systems, by considering a variety of node distributions and communication patterns among neighbours. More concretely, we consider a distributed, multi-agent Lotka-Volterra model, where each agent evolves according to its own LV model, and whose state may be influenced by contributions from neighbouring agents. We investigate whether through communication the agents get to synchronize their LV cycle regardless of their initial states. We demonstrate that, depending on the communication topology and each agent's communication policy, the different agents may or may not eventually synchronize their cycles. The predator-prey metaphor that we follow is that where rabbits and/or foxes can escape from one forest to a neighbouring forest if their level is high. The details of the model depend on the topology of the communication (defining who may send to whom) and on the policy of the communication (defining what may be sent). Getting an influx of either rabbits or foxes will set the corresponding variable on 'high', even if its local status without the influx would have been 'low'. In this section we discuss the process algebraic formulation of the model. The reaction system formulation of the model is in the Appendix.

Our model consists of a family of $k \geq 2$ agents, numbered from 1 to k. Each agent is modelled through its own reaction system, but all reaction systems rely on the same background set and their reactions differ only with the aspects related to communications of entities to neighbouring nodes. As a preliminary step towards the final model, let us remind the entities considered so far:

- we use r and R to represent low/high levels of rabbits, respectively;
- we use f and F to represent low/high levels of foxes, respectively;

- we use xr and xf to represent extra levels of rabbits/foxes coming from surrounding contexts, respectively.

Since we want to study the behaviour of the network according to any initial configuration of each agent, the initial ingredient is a non-deterministic context that provides the initial levels of rabbits and foxes.

$$\mathsf{K}_{rR} \triangleq \{\mathsf{r}\}.\mathsf{Emp} + \{\mathsf{R}\}.\mathsf{Emp}$$
$$\mathsf{K}_{fF} \triangleq \{\mathsf{f}\}.\mathsf{Emp} + \{\mathsf{F}\}.\mathsf{Emp}$$
$$\mathsf{Emp} \triangleq \varnothing.\mathsf{Emp}$$

The use of the recursive vacuous process Emp is needed as a continuation because the use of $\mathbf{0}$ in its place would stop the computation immediately.

In our full formulation of the distributed LV model, we must account for the idea that having xr in the current state contributes in the same way in determining the next state as having R, and that xf has the same influence as having F. In the previous sections, we have addressed this issue by increasing the number of reactions to handle the possible combinations of r/xr and f/xf. For example, when both r and xr are present, the behaviour of the agent should correspond to the situation where a large number of rabbits is available, but when r is present and xr is absent, it means that only a low number of rabbits is available. When R is present, then the presence/absence of xr is not essential because there is already a large number of rabbits. Similarly for the population of foxes. Of course, we tacitly assume the invariant that guarantees r and R are never both present at the same time, and similarly for f and F.

(Prey multiplies)	$a_1 \triangleq (\{\mathsf{r},\mathsf{f}\},\{\mathsf{xr},\mathsf{xf}\},\{\mathsf{R},\mathsf{f}\})$;
(Prey being hunted)	$a_2 \triangleq (\{\mathsf{R},\mathsf{F}\},\varnothing,\{\mathsf{r},\mathsf{F}\})$;
	$a_2' \triangleq (\{\mathsf{r},\mathsf{xr},\mathsf{F}\},\varnothing,\{\mathsf{r},\mathsf{F}\})$;
	$a_2'' \triangleq (\{\mathsf{R},\mathsf{f},\mathsf{xf}\},\varnothing,\{\mathsf{r},\mathsf{F}\})$;
	$a_2''' \triangleq (\{\mathsf{r},\mathsf{xr},\mathsf{f},\mathsf{xf}\},\varnothing,\{\mathsf{r},\mathsf{F}\})$;
(Predator multiplies)	$a_3 \triangleq (\{\mathsf{R},\mathsf{f}\},\varnothing,\{\mathsf{R},\mathsf{F}\})$;
	$a_3' \triangleq (\{\mathsf{r},\mathsf{xr},\mathsf{f}\},\varnothing,\{\mathsf{R},\mathsf{F}\})$;
(Predator dies)	$a_4 \triangleq (\{\mathsf{r},\mathsf{F}\},\{\mathsf{xr}\},\{\mathsf{r},\mathsf{f}\})$;
	$a_4' \triangleq (\{\mathsf{r},\mathsf{f},\mathsf{xf}\},\{\mathsf{xr}\},\{\mathsf{r},\mathsf{f}\})$.

To mitigate the case explosion, we adopt here a different approach based on local contexts, which can be used to determine the overall level of rabbits and foxes before the application of reactions. The idea is to acknowledge that the products of reactions do not necessarily determine themselves the overall numbers of preys and predators, because external interventions are possible. This leads us to introduce some auxiliary versions of the background set and leave the task to determine the introduction of r/R and f/F to suitable guarded contexts. Consequently, we consider the following background sets, where primed entities will only appear in product sets of reactions and in guarded prefixes of context processes:

- we use r and R to represent low/high levels of rabbits, respectively;
- we use r' and R' to represent that low/high levels of rabbits are at least produced, respectively;
- we use f and F to represent low/high levels of foxes, respectively;
- we use f' and F' to represent that low/high levels of foxes are at least produced, respectively;
- we use xr' and xf' to represent extra levels of rabbits/foxes coming from surrounding contexts, respectively.

Given primed entities, the following guarded contexts compute the right level of preys and predators:

$$\begin{aligned}\mathsf{K}_r \triangleq\ & (\{\mathsf{r}',\mathsf{xr}'\},\varnothing,\{\mathsf{R}\}).\mathsf{K}_r + (\{\mathsf{R}'\},\varnothing,\{\mathsf{R}\}).\mathsf{K}_r \\ & + (\{\mathsf{r}'\},\{\mathsf{xr}'\},\{\mathsf{r}\}).\mathsf{K}_r + (\varnothing,\{\mathsf{r}',\mathsf{xr}',\mathsf{R}'\},\varnothing).\mathsf{K}_r \\ \mathsf{K}_f \triangleq\ & (\{\mathsf{f}',\mathsf{xf}'\},\varnothing,\{\mathsf{F}\}).\mathsf{K}_f + (\{\mathsf{F}'\},\varnothing,\{\mathsf{F}\}).\mathsf{K}_f \\ & + (\{\mathsf{f}'\},\{\mathsf{xf}'\},\{\mathsf{f}\}).\mathsf{K}_f + (\varnothing,\{\mathsf{f}',\mathsf{xf}',\mathsf{F}'\},\varnothing).\mathsf{K}_f\end{aligned}$$

The last option handles the initial configuration of the agent, where none of the entities $\{\mathsf{r}',\mathsf{xr}',\mathsf{R}',\mathsf{f}',\mathsf{xf}',\mathsf{F}'\}$ are present, because the initial level of rabbits and foxes has not yet been determined by K_{rR} and K_{fF}. If these options were omitted, no choice would be possible and the computation would stop immediately.

In turn, this tweak allows to reuse a much simpler version of reactions, where extra levels of preys and predators need not be taken into account as reactants and inhibitors.

(Prey multiplies)	$a_1 \triangleq (\{\mathsf{r},\mathsf{f}\},\varnothing,\{\mathsf{R}',\mathsf{f}'\})$;
(Prey being hunted)	$a_2 \triangleq (\{\mathsf{R},\mathsf{F}\},\varnothing,\{\mathsf{r}',\mathsf{F}'\})$;
(Predator multiplies)	$a_3 \triangleq (\{\mathsf{R},\mathsf{f}\},\varnothing,\{\mathsf{R}',\mathsf{F}'\})$;
(Predator dies)	$a_4 \triangleq (\{\mathsf{r},\mathsf{F}\},\varnothing,\{\mathsf{r}',\mathsf{f}'\})$.

Depending on the communication topology, when large populations of rabbits and foxes are produced, a provision of extra rabbits and foxes must be communicated to neighbour nodes. Therefore, the above reactions must be specialized for each agent, depending on whether the network is designed to allow migration of preys and predators to neighbour nodes. This is accomplished by defining the following sets of located entities for each node n:

$$\mathcal{XR}_n \triangleq \{m[\mathsf{xr}'] \mid n \xrightarrow{\mathsf{xr}'} m \text{ is a communication edge}\},$$
$$\mathcal{XF}_n \triangleq \{m[\mathsf{xf}'] \mid n \xrightarrow{\mathsf{xf}'} m \text{ is a communication edge}\},$$
$$\mathcal{X}_n \triangleq \mathcal{XR}_n \cup \mathcal{XF}_n.$$

To make this scheme concrete, consider a 2-agent LV model where the first agent can send out rabbits to the second, but not foxes, while the second cannot send out anything to the first. This variant of the model corresponds to the following communication sets: $\mathcal{X}_1 = \mathcal{XR}_1 = \{2[\mathsf{xr}']\}$, $\mathcal{XF}_1 = \mathcal{X}_2 = \mathcal{XR}_2 = \mathcal{XF}_2 = \varnothing$.

The reactions of the agent n are thus specialised by adding the product sets $\mathcal{XR}_n$ and $\mathcal{XF}_n$ whenever a large number of preys/predators is produced.

(Prey mult.) $a_{n,1} \triangleq (\{\mathsf{r}, \mathsf{f}\}, \varnothing, \{\mathsf{R}', \mathsf{f}'\} \cup \mathcal{XR}_n)$;
(Prey hunted) $a_{n,2} \triangleq (\{\mathsf{R}, \mathsf{F}\}, \varnothing, \{\mathsf{r}', \mathsf{F}'\} \cup \mathcal{XF}_n)$;
(Pred. mult.) $a_{n,3} \triangleq (\{\mathsf{R}, \mathsf{f}\}, \varnothing, \{\mathsf{R}', \mathsf{F}'\} \cup \mathcal{X}_n)$;
(Pred. dies) $a_{n,4} \triangleq (\{\mathsf{r}, \mathsf{F}\}, \varnothing, \{\mathsf{r}', \mathsf{f}'\})$.

We let $\mathsf{React}_n \triangleq a_{n,1} \mid a_{n,2} \mid a_{n,3} \mid a_{n,4}$ denote the parallel composition of the above reactions.

The whole network is described by the DRS process $\prod_n \mathsf{P}_n$, where each agent $n \in [1, k]$ is modelled as $\mathsf{P}_n \triangleq n[\mathsf{K}_{rR} \mid \mathsf{K}_{fF} \mid \mathsf{K}_r \mid \mathsf{K}_f \mid \mathsf{React}_n]$.

We investigated the behaviour of the model with 2 and with 3 agents in many different variants of the communication topology and of the communication policy. We explored the context-free interactive processes of the models starting from all initial states where each agent A_k is in any of the four possible initial states $k[\mathsf{fr}]$, $k[\mathsf{fR}]$, $k[\mathsf{Fr}]$, or $k[\mathsf{FR}]$, $\forall 1 \leq k \leq n$. This leads to 16 initial states for the 2-agent model and to 64 initial states for the 3-agent model. It is obvious from the model formulation that starting from these states, in all subsequent states each agent contains either r' or R', and either f' or F'. However, the states may also include a combination of xr' and xf', depending on the communication policy/topology of the model and on the states of the other agents. We reduce the number of states in the figures (but not in the BioReSolve simulations) so that the extra rabbits and the rabbits variables are combined into a single rabbits variable, and similarly for the fox variables. For example, to display the state transition graphs, we noticed the equivalence between the sets $\{\mathsf{r}', \mathsf{xr}'\}$, $\{\mathsf{R}', \mathsf{xr}'\}$ and $\{\mathsf{R}'\}$ (all denoting a large number of rabbits) and between the sets $\{\mathsf{f}', \mathsf{xf}'\}$, $\{\mathsf{F}', \mathsf{xf}'\}$ and $\{\mathsf{F}'\}$ (all denoting a large number of foxes). Based on this observation, we merged in the figures all equivalent states, leading to the graphs in Figure 3 with 64 nodes. The full set of reaction systems models that we analysed and the summary of the results are available at [51]. The idea of exploring transition graphs to explore the equivalence of reaction systems was discussed in a different setup in [52].

Our analyses of the distributed Lotka-Volterra model with reaction systems show a surprisingly diverse behaviour, that may be worthy of further investigation. For example, characterizing the conditions under which the agents eventually synchronize their cycles regardless of the initial states (equivalently, the reduced state transition graph collapses into a single cycle) seems to be quite complex, as it depends both on the communication topology as well as on the communication policy for each agent. Interestingly, the synchronised cycles appear as an emergent behaviour in all networks of Figure 3, even if in some of the cases additional stable cycles are also possible.

To further explore the scalability of the BioReSolve environment, we tested distributed Lotka-Volterra models with up to 7 agents with a circular communication topology, where agent i communicates to agent $i + 1$ (agent n sends to agent 1) both rabbits and foxes. As before, we considered all initial states where each agent A_k is in any of the four possible initial states $k[\mathsf{fr}]$, $k[\mathsf{fR}]$, $k[\mathsf{Fr}]$, or $k[\mathsf{FR}]$, $\forall 1 \leq k \leq n$. Each agent in the distributed Lotka-Volterra model has 2^4 possible states, because each of the variables for foxes, rabbits, extra foxes and extra rabbits can have 2 possible values. This means that the n-agent model has 2^{4n} possible states. However, depending on the communication topology of the model, not all possible states are necessarily reachable from the initial states. With 7 agents, the state-transition graph has 32.769 reachable states out of $(2^4)^7 = 2^{28} = 268.435.456$ admissible states and it was produced by BioReSolve in about 10 minutes on a standard laptop.

VI. RELATED WORK

The literature on distributed, multi-agent reaction systems is very rich. They were introduced in [22] (full version in [11]) and consisted of several reaction systems working in parallel, with a context automaton providing a conditional generation of contexts based on the current state of each reaction system agent, but with the agents unable to communicate directly with each other. In addition, the authors introduced rsCTLK, a logic for reaction systems, allowing for model checking of temporal logic properties.

A related notion of a network of reaction systems was introduced in [23]. The network is specified through a graph, with the reaction system agents placed in its nodes, and with the edges specifying the communication channels between them. Each reaction system in this model contributes all products of all its enabled reactions to the context of its neighbours. The authors show that it is possible to define a transformation from a network to a standard RS in order to simulate the behaviour of the network. Our system is slightly more flexible, as a RS can send just a subset of products to each target RSs, while the whole set of products is dispatched in [23].

A more targeted mode of communication is defined in [24], where the networks of reaction systems are extended with the ability of individual agents to communicate products or reactions to each other. The system is more flexible than in [23], as one RS can communicate a subset of products to any RSs in the network. There is also a variant in which agents will send all of its enabled reactions to its neighbours, while losing all of the disabled reactions, thus offering a dynamic, continuously evolving structure of the system. The model was further extended in [29] with communication by request: individual agents are equipped in this version with the ability to request the state of other components. Our definitions are similar to those in [24] for the communication of products; however, [24] does not consider an explicit notion of context, while we work with extended contexts, which can be non-deterministic and recursive. We also provide an algebraic compositional semantics, which facilitates the definition of extensions of our framework. Moreover, we have an implementation which allows us to perform in-silico experiments with graphical representations of the computations, which results useful also for studying the attractors of the modelled system. Another

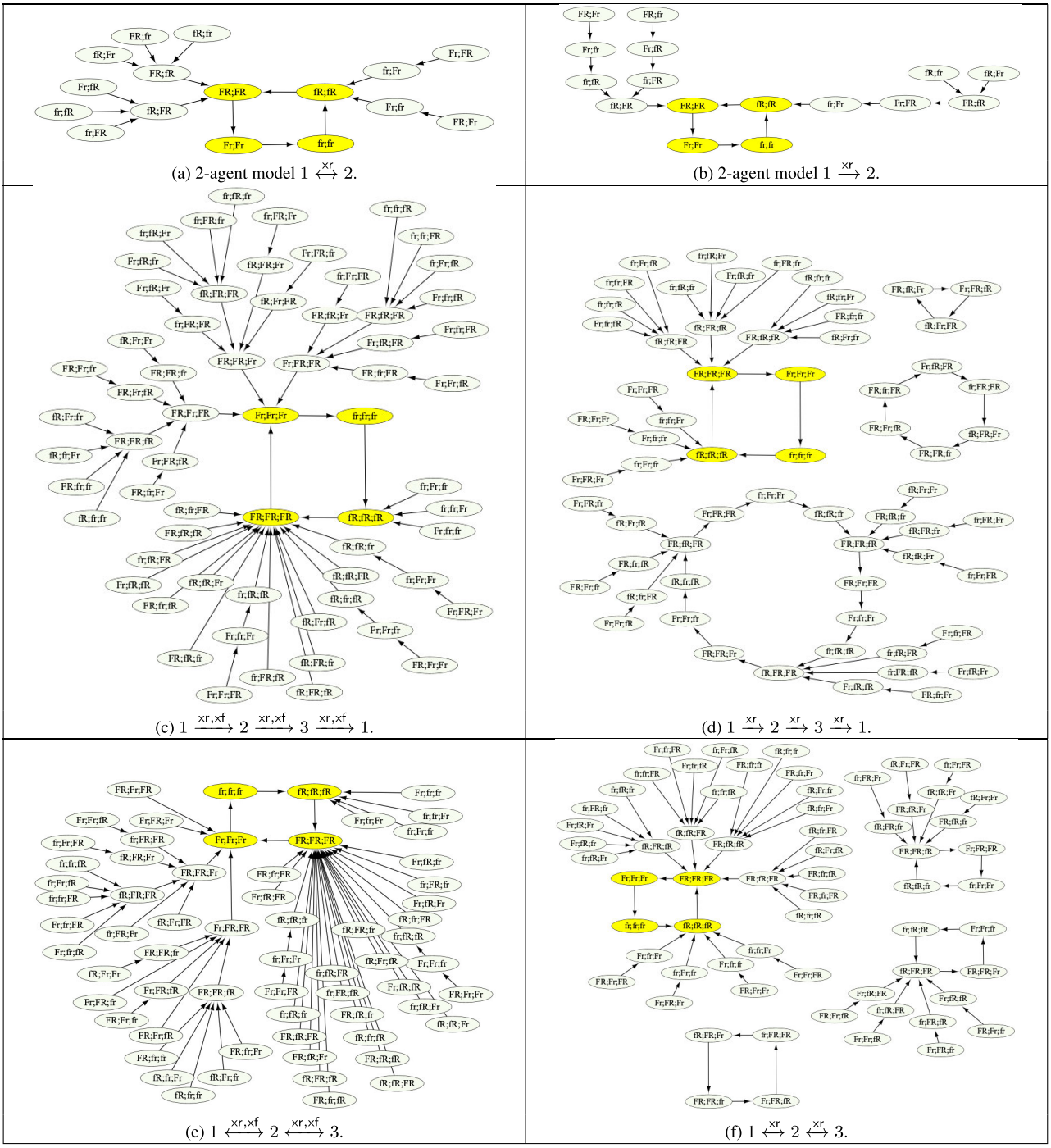

FIGURE 3. Several variants of reaction system, multi-agent Lotka-Volterra models exhibit diverse behaviour: depending on the topology of the system and on the communication policy, the agents may eventually synchronize their cycles or not. The node labels indicate the state of each of the agents (separated with semi-colon), while the edges indicate the state transitions. Each node in the graph represents a model state, and present its configurations as a list of agents, separated by semicolon. For each agent, the configuration is defined by the levels of foxes ('f' for low, 'F' for high) and rabbits ('r' for low, 'R' for high). Each edge represents a state transition. The synchronized cycles are in yellow. The state-transition graphs were obtained with BioResolve [12], [39] and the visualization with Cytoscape [53]. (a) A two-agent model where the agents send out rabbits to each other when they transition to a state with a high level of rabbits. (b) Similar as in (a), but the communication is only from agent one to agent two. (c) A three-agent model with the communication cyclic from agent one to agent two to agent three to agent one. Each agent sends out rabbits (foxes, resp.) when they transition to a state with a high level of rabbits (foxes, resp.) (d) Same topology as in (c), but only rabbits are being sent out from one agent to the next. (e) A three-agent model with bi-directional communication between agents one and two and between agents two and three. Both rabbits and foxes are being sent out. (g) Same topology as in (f), but only rabbits are being sent out.

important novel contribution of our system is that contexts themselves are processes and hence they can also exchange information directly between them, and this improves much the expressivity and flexibility of our framework.

The concepts of networks of RS and communicating (distributed) RS were compared in [25] and [28]. The work on distributed reaction systems was continued in [26] and [27] where languages were assigned to the computations of distributed reaction systems and their place in the Chomsky hierarchy was investigated. In a recent study, a generalized distributed reaction system model was introduced in [30] where reactions can check in all agents for the presence and absence of various resources, and the products may be communicated to other agents. The authors prove that all these variants of generalized distributed reaction systems can be transferred into a standard reaction system.

VII. CONCLUSION

We introduced in this paper a process algebra approach to distributed reaction systems. Our main objective for doing so was to gain access to a rich literature on analysis and simulation tools for concurrent systems, making them applicable to distributed reaction systems. We also implemented this method in the software tool BioResolve [39], making it possible to describe, analyse and simulate distributed reaction systems. This may be significant to modellers and theoreticians in reaction systems, allowing for easy prototyping of various models and concepts.

We introduced the reaction systems description of a distributed model of Lotka-Volterra as a standard model of co-dependent species. This adds to the library of models available for reaction systems, which will be of independent interest as a prototype of a complex interactive behaviour described through simple regulation via inhibition. We highlighted how the communication policy adds a considerable amount of complexity to the behaviour of this classical model, to the extent that it seems a difficult problem to characterize under which conditions the Lotka-Volterra agents may synchronize their cycles.

The bridge to process algebras that we established in this paper may be significant to exploring a wider range of topics of research in reaction systems: quantitative reaction systems [13], [35], [54], [55], causality analysis, model checking [11] and different notions of equivalence [38], [52], [56], [57], [58]. Over the past two decades, reaction systems have been extensively studied as a non-quantitative, set-based mathematical formalism. While quantitative extensions were proposed in [10], [13], [54], and [59], such approaches received limited attention, possibly due to their inherent mathematical intricacies. Our work builds a bridge to process algebra, providing an alternative framework that is well suited to quantitative analysis. This enables the study of resource competition, concurrency, stochastic dynamics, numerical convergence, and other key concepts. These are well-established topics in mathematical modelling and are highly relevant to the study of biological systems. Addressing them within the reaction systems framework, particularly with its focus on cause-effect relationships, is both promising and exciting.

TABLE 1. List of acronyms.

Acronym	Meaning
RS	Reaction System
DRS	Distributed Reaction System
LV	Lotka-Volterra model

TABLE 2. General symbols for reaction systems.

Symbol	Description
S	Background set of entities in a reaction system
e	Generic entity
D	Current set of entities in a reaction system
C	Contextual set of entities in a reaction system
W	State in which reactions are applied
R	Set of reactants in a reaction
I	Set of inhibitors in a reaction
P	Set of products in a reaction
$R\#I$	Disjoint sets
a	Generic reaction
(R, I, P)	A reaction tuple: reactants, inhibitors, products
A	Set of reactions
$en_a(W)$	Enabling condition of reaction a in state W
$res_a(W)$	Result of reaction a in state W
$res_A(W)$	Result of all reactions in set A in state W

TABLE 3. General symbols for RS processes.

Symbol	Description
K	Generic context process
$\mathbf{0}$	Termination
A	Identifier
$(R, I, C).\mathsf{K}$	Guarded prefix
$C.\mathsf{K}$	Prefix (shorthand for $(\varnothing, \varnothing, C).\mathsf{K}$)
$\mathsf{K}_1 + \mathsf{K}_2$	Nondeterministic choice
$\mathsf{A} \triangleq \mathsf{K}$	Identifier def. (possibly recursive)
Δ	Environment: set of identifier defs
M	Generic mixture process
a	A generic single-reaction process
(R, I, P)	A single-reaction process
$\mathsf{M}_1 \mid \mathsf{M}_2$	Binary parallel composition
$\prod_{j\in J} \mathsf{M}_j$	Generic parallel composition
P	Generic RS process
$[\mathsf{M}]$	RS process
$\mathsf{P}_1 \equiv \mathsf{P}_2$	Structural equivalence
$[D \mid (\prod_{k\in K} \mathsf{K}_k) \mid (\prod_{j\in J} a_j)]$	RS process in standard form

APPENDIX
GLOSSARY AND NOTATION

Tables 1–5 provide the key references for acronyms and symbols used throughout the main text. To facilitate readability and contextual clarity, different tables have been employed depending on the specific context in which the terms appear.

APPENDIX
SOS SEMANTICS OF RS PROCESSES

The SOS semantics of RS processes is defined by the SOS rules in Table 6 (see [7]). A transition label ℓ, written $\langle\langle D \rhd R', I', C\rangle \rhd R, I, P\rangle$ or just $\langle\langle D \rhd a'\rangle \rhd a\rangle$ for short, records: the set D of entities currently in the system; the set C of entities provided by the context given the presence of all entities in R' and the absence of all entities in I'; and the set P of reaction products given the presence of all entities in R and

TABLE 4. General symbols for DRS processes.

Symbol	Description
$\mathcal{N}$	Set of location names
n	Generic location name
$\mathcal{N}[S]$	Set of located entities
$n[\cdot]$	Container
$n[\mathsf{e}]$	Located entity
$n[\mathsf{M}]$	located RS process
$\prod_{n\in\mathcal{N}} n[\mathsf{M}_n]$	Located process in standard form
$msg(\mathsf{M}_n)$	Outgoing messages
$\mathcal{F}(\cdot)$ and $\mathcal{F}(\cdot,\cdot)$	Flattening functions

TABLE 5. Entities in Lotka-Volterra models.

Symbol	Description
r	Low population of rabbits
R	High population of rabbits
xr	Extra population of rabbits
f	Low population of foxes
F	High population of foxes
xf	Extra population of foxes
r′, f′, ...	Auxiliary (temporary) versions

the absence of all entities in I. The SOS rules guarantee that whenever $\mathsf{P} \xrightarrow{\langle\langle D\triangleright R',I',C\rangle\triangleright R,I,P\rangle} \mathsf{P}'$ it holds that $en_{(R',I',C)}(D)$ and $en_{(R,I,P)}(D\cup C)$ or, equivalently but more synthetically, that whenever $\mathsf{P} \xrightarrow{\langle\langle D\triangleright a'\rangle\triangleright a\rangle} \mathsf{P}'$ it holds that $en_{a'}(D)$ and $en_a(D\cup res_{a'}(D))$. Since the various components of the label are computed inductively while traversing each component of P (i.e., any set of entities, any reactions, and any context in P, independently of the order in which they appear) we must make sure that the provisions of separate components are always coherent. This amounts to requiring any label $\ell = \langle\langle D \triangleright R', I', C\rangle \triangleright R, I, P\rangle$ to satisfy the invariant

$$inv(\ell) \triangleq I'\#(D\cup R') \wedge I\#(D\cup R'\cup C\cup R).$$

To this aim, anytime we conjoin two parallel computations $\mathsf{M}_1 \xrightarrow{\ell_1} \mathsf{M}_1'$ and $\mathsf{M}_2 \xrightarrow{\ell_2} \mathsf{M}_2'$ we need to guarantee that such an invariant is preserved by $\mathsf{M}_1|\mathsf{M}_2 \xrightarrow{\ell_1\cup\ell_2} \mathsf{M}_1'|\mathsf{M}_2'$, where $\ell_1\cup\ell_2$ is the label obtained by the component-wise union of its constituents, Formally, we denote this property by writing $\ell_1 \frown \ell_2 \triangleq inv(\ell_1\cup\ell_2)$. The exact definitions of both $\ell_1\cup\ell_2$ and $\ell_1\frown\ell_2$ are spelled out in Table 6.

We now give a brief account of each rule. The rule (*Ent*) records the set of current entities D.

By rule (*Cxt*), a prefixed context process $(R, I, C).\mathsf{K}$ records in the label that the entities R must be present, I must be absent and C are made available and then reduces to K. Rules (*Suml*) and (*Sumr*) select a move of either the left or the right context, resp., discarding the other process. By rule (*Const*), a context identifier A defined as $\mathsf{A} \triangleq \mathsf{K}$ in the environment Δ can behave according to its defining process K.

The rule (*Pro*) assumes the reaction (R, I, P) is enabled: it records its reactants, inhibitors, and products in the label, and leaves the reaction available at the next step, together with its products P. The rule (*Inh*) records in the label the reasons why the reaction (R, I, P) should not be executed: possibly some inhibiting entities ($J \subseteq I$) are present or some reactants ($Q \subseteq R$) are missing, with $J\cup Q \neq \varnothing$, as at least one cause is needed. The rule (*Par*) puts two processes in parallel by pooling their labels and joining all labels components. The sanity check $\ell_1 \frown \ell_2$ is required to guarantee that labels of reactants and inhibitors are consistent.

Finally, the rule (*Sys*) checks that all the needed reactants are available in the system, for both applied guarded contexts (i.e., $R' \subseteq D$) and reactions (i.e., $R \subseteq D\cup C$). Checking the absence of inhibitors ($I'\#D$ and $I\#(D\cup C)$) is not necessary, thanks to the sanity check $\ell_1 \frown \ell_2$ in rule (*Par*).

APPENDIX
SOS SEMANTICS OF DRS PROCESSES AND THE FLATTENING THEOREM

The extension to DRSs requires just three minor tweaks of the SOS rules:

- first, the background set of labels is now $S\cup\hat{S}$, i.e., located entities can appear in the labels;
- second, the rule (*Sys*) is replaced by the new rule (*Loc*), introduced to handle located reaction systems;
- third, the rule (*Par*) should now be applicable to DRS processes P and not just mixture processes M. To avoid any misunderstanding we have preferred to introduce a corresponding rule called (*DPar*).

The new inference rules (*Loc*) and (*DPar*) are defined in Table 6. The next Theorem 8 formalises the correspondence between DRSs and RSs.

Theorem 8: For any RS process P it holds that $\mathsf{P} \xrightarrow{\ell} \mathsf{P}'$ iff $\mathcal{F}(\mathsf{P}) \xrightarrow{\ell} \mathcal{F}(\mathsf{P}')$.

Proof: Actually, we prove a stronger statement, namely that all the following properties are valid:

1) for any RS process P, if $\mathsf{P} \xrightarrow{\ell} \mathsf{P}'$ then $\mathcal{F}(\mathsf{P}) \xrightarrow{\ell} \mathcal{F}(\mathsf{P}')$;
2) for any RS process P, if $\mathcal{F}(\mathsf{P}) \xrightarrow{\ell} \mathsf{P}''$ then $\mathsf{P} \xrightarrow{\ell} \mathsf{P}'$ with $\mathsf{P}'' = \mathcal{F}(\mathsf{P}')$;
3) for any mixture process M, if $\mathsf{M} \xrightarrow{\ell} \mathsf{M}'$ then $\mathcal{F}(n,\mathsf{M}) \xrightarrow{n[\ell]} \mathcal{F}(n,\mathsf{M}')$;
4) for any mixture process M, if $\mathcal{F}(n,\mathsf{M}) \xrightarrow{\ell'} \mathsf{M}''$ then $\mathsf{M} \xrightarrow{\ell} \mathsf{M}'$ with $\ell' = n[\ell]$ and $\mathsf{M}'' = \mathcal{F}(n,\mathsf{M}')$.

where, given $\ell = \langle\langle D \triangleright R', I', C\rangle \triangleright R'I, P\rangle$ we let $n[\ell] \triangleq \langle\langle n[D] \triangleright n[R'], n[I'], n[C]\rangle \triangleright n[R], n[I], n[P]\rangle$.

Similarly to what has been done for the statement of Theorem 8, properties 3 and 4 subsume the following (much more intuitive) correspondence:

- $\mathsf{M} \xrightarrow{\ell} \mathsf{M}'$ iff $\mathcal{F}(n,\mathsf{M}) \xrightarrow{n[\ell]} \mathcal{F}(n,\mathsf{M}')$.

The proof follows from a straightforward structural induction on the syntax of processes. Some cases are illustrated below. Whenever possible we illustrate directly the double implication, instead of discussing the two implications separately.

Base case $\mathsf{M} = D$ and thus $\mathcal{F}(n,\mathsf{M}) = n[D]$. It is obvious that, in both cases, the only applicable axiom is

TABLE 6. SOS semantics of RS processes with guarded contexts (cf. [7]).

$\dfrac{}{D \xrightarrow{\langle\langle D \rhd \varnothing,\varnothing,\varnothing\rangle \rhd \varnothing,\varnothing,\varnothing\rangle} \varnothing}$ (Ent)	$\dfrac{}{(R,I,C).\mathsf{K} \xrightarrow{\langle\langle \varnothing \rhd R,I,C\rangle \rhd \varnothing,\varnothing,\varnothing\rangle} \mathsf{K}}$ (Pfx)
$\dfrac{\mathsf{K}_1 \xrightarrow{\ell} \mathsf{K}_1'}{\mathsf{K}_1 + \mathsf{K}_2 \xrightarrow{\ell} \mathsf{K}_1'}$ $(Suml)$	$\dfrac{\mathsf{K}_2 \xrightarrow{\ell} \mathsf{K}_2'}{\mathsf{K}_1 + \mathsf{K}_2 \xrightarrow{\ell} \mathsf{K}_2'}$ $(Sumr)$
$\dfrac{\mathsf{M}_1 \xrightarrow{\ell_1} \mathsf{M}_1' \quad \mathsf{M}_2 \xrightarrow{\ell_2} \mathsf{M}_2' \quad \ell_1 \frown \ell_2}{\mathsf{M}_1 \mid \mathsf{M}_2 \xrightarrow{\ell_1 \cup \ell_2} \mathsf{M}_1' \mid \mathsf{M}_2'}$ (Par)	$\dfrac{\mathsf{A} \triangleq \mathsf{K} \in \Delta \quad \mathsf{K} \xrightarrow{\ell} \mathsf{K}'}{\mathsf{A} \xrightarrow{\ell} \mathsf{K}'}$ $(Const)$
$\dfrac{}{(R,I,P) \xrightarrow{\langle\langle \varnothing \rhd \varnothing,\varnothing,\varnothing\rangle \rhd R,I,P\rangle} (R,I,P) \mid P}$ (Pro)	$\dfrac{J \subseteq I \quad Q \subseteq R \quad J \cup Q \neq \varnothing}{(R,I,P) \xrightarrow{\langle\langle \varnothing \rhd \varnothing,\varnothing,\varnothing\rangle \rhd J,Q,\varnothing\rangle} (R,I,P)}$ (Inh)
$\dfrac{\mathsf{M} \xrightarrow{\langle\langle D \rhd R',I',C\rangle \rhd R,I,P\rangle} \mathsf{M}' \quad R' \subseteq D \quad R \subseteq D \cup C}{[\mathsf{M}] \xrightarrow{\langle\langle D \rhd R',I',C\rangle \rhd R,I,P\rangle} [\mathsf{M}']}$ (Sys)	
Where, in the above, given $\ell_1 = \langle\langle D_1 \rhd R_1', I_1', C_1\rangle \rhd R_1, I_1, P_1\rangle$ and $\ell_2 = \langle\langle D_2 \rhd R_2', I_2', C_2\rangle \rhd R_2, I_2, P_2\rangle$ we let:	
$\ell_1 \cup \ell_2 \triangleq \langle\langle D_1 \cup D_2 \rhd R_1' \cup R_2', I_1' \cup I_2', C_1 \cup C_2\rangle \rhd R_1 \cup R_2, I_1 \cup I_2, P_1 \cup P_2\rangle$	
$\ell_1 \frown \ell_2 \triangleq (I_1' \cup I_2') \# (D_1 \cup D_2 \cup R_1' \cup R_2') \wedge (I_1 \cup I_2) \# (D_1 \cup D_2 \cup R_1' \cup R_2' \cup C_1 \cup C_2 \cup R_1 \cup R_2)$	

TABLE 7. The new inference rules of DRS processes.

$\dfrac{\mathsf{M} \xrightarrow{\langle\langle D \rhd R',I',C\rangle \rhd R,I,P\rangle} \mathsf{M}' \quad R' \subseteq D \quad R \subseteq D \cup C}{n[\mathsf{M}] \xrightarrow{\langle\langle n[D] \rhd n[R'],n[I'],n[C]\rangle \rhd n[R],n[I],n[P]\rangle} n[\mathsf{M}']}$ (Loc)	$\dfrac{\mathsf{P}_1 \xrightarrow{\ell_1} \mathsf{P}_1' \quad \mathsf{P}_2 \xrightarrow{\ell_2} \mathsf{P}_2' \quad \ell_1 \frown \ell_2}{\mathsf{P}_1 \mid \mathsf{P}_2 \xrightarrow{\ell_1 \cup \ell_2} \mathsf{P}_1' \mid \mathsf{P}_2'}$ $(DPar)$

(Ent) and therefore $\ell = \langle\langle D \rhd \varnothing,\varnothing,\varnothing\rangle \rhd \varnothing,\varnothing,\varnothing\rangle$ and $n[\ell] = \langle\langle n[D] \rhd \varnothing,\varnothing,\varnothing\rangle \rhd \varnothing,\varnothing,\varnothing\rangle$. In fact: $D \xrightarrow{\ell} \varnothing$ iff $n[D] \xrightarrow{n[\ell]} \varnothing$

Base case $\mathsf{M} = \mathbf{0}$ and thus $\mathcal{F}(n,\mathbf{0}) = \mathbf{0}$. It is obvious that, in both cases, there are no outgoing transitions and thus the property trivially holds.

Base case $\mathsf{M} = (R,I,P)$ and thus $\mathcal{F}(n,\mathsf{M}) = (n[R], n[I], n[P])$. By applying the axiom (Pro) to M we derive the transition $\mathsf{M} \xrightarrow{\ell} \mathsf{M}|P$ with $\ell = \langle\langle \varnothing \rhd \varnothing,\varnothing,\varnothing\rangle \rhd R,I,P\rangle$ and, by applying the same axiom to $\mathcal{F}(n,\mathsf{M})$ we derive the transition $\mathcal{F}(n,\mathsf{M}) \xrightarrow{n[\ell]} \mathsf{M}|n[P] = \mathcal{F}(n,\mathsf{M}|P)$, since $n[\ell] = \langle\langle \varnothing \rhd \varnothing,\varnothing,\varnothing\rangle \rhd n[R],n[I],n[P]\rangle$. Alternatively, we can apply the axiom (Inh) by taking suitable sets $J \subseteq I$ and $Q \subseteq R$ such that $J \cup Q \neq \varnothing$, in which case we can still derive both transitions $\mathsf{M} \xrightarrow{\ell} \mathsf{M}$ and $\mathcal{F}(n,\mathsf{M}) \xrightarrow{n[\ell]} \mathcal{F}(n,\mathsf{M})$, where $\ell = \langle\langle \varnothing \rhd \varnothing,\varnothing,\varnothing\rangle \rhd J,Q,\varnothing\rangle$ and $n[\ell] = \langle\langle \varnothing \rhd \varnothing,\varnothing,\varnothing\rangle \rhd n[J], n[Q], \varnothing\rangle$.

Inductive case $\mathsf{K} = \mathsf{K}_1 + \mathsf{K}_2$ and thus $\mathcal{F}(n,\mathsf{K}) = \mathcal{F}(n,\mathsf{K}_1) + \mathcal{F}(n,\mathsf{K}_2)$. By inductive hypotheses, we assume that $\mathsf{K}_i \xrightarrow{\ell} \mathsf{K}_i'$ iff $\mathcal{F}(n,\mathsf{K}_i) \xrightarrow{n[\ell]} \mathcal{F}(n,\mathsf{K}_i')$ for any $i \in [1,2]$. Therefore, the rule $(Suml)$ is either applicable to both K and $\mathcal{F}(n,\mathsf{K})$ or to none of them. If the rule $(Suml)$ is applicable, then trivially $\mathsf{K} \xrightarrow{\ell} \mathsf{K}_1'$ iff $\mathcal{F}(n,\mathsf{K}) \xrightarrow{n[\ell]} \mathcal{F}(n,\mathsf{K}_1')$, because $\mathsf{K}_1 \xrightarrow{\ell} \mathsf{K}_1'$ iff $\mathcal{F}(n,\mathsf{K}_1) \xrightarrow{n[\ell]} \mathcal{F}(n,\mathsf{K}_1')$. Analogous considerations apply to rule $(Suml)$.

Inductive case $\mathsf{P} = \prod_{n\in\mathcal{N}} n[\mathsf{M}_n]$ and thus $\mathcal{F}(\mathsf{P}) = \left[\prod_{n\in\mathcal{N}} \mathcal{F}(n,\mathsf{M}_n)\right]$. By inductive hypothesis, we assume that, for any $n \in \mathcal{N}$, $\mathsf{M}_n \xrightarrow{\ell_n} \mathsf{M}_n'$ iff $\mathcal{F}(n,\mathsf{M}_n) \xrightarrow{n[\ell_n]} \mathcal{F}(n,\mathsf{M}_n')$. Here we prove the two implications of the thesis separately. Now, suppose that $\mathsf{P} \xrightarrow{\ell} \mathsf{P}'$. Then, by rule $(DPar)$, it must be the case that, for any $n \in \mathcal{N}$, $n[\mathsf{M}_n] \xrightarrow{\ell_n'} n[\mathsf{M}_n']$ with $\mathsf{P}' = \prod_{n\in\mathcal{N}} n[\mathsf{M}_n']$, all labels in the family $\{\ell_n'\}_{n\in\mathcal{N}}$ pairwise compatible (i.e., $\ell_n' \frown \ell_m'$ for any $n \neq m$) and $\ell = \bigcup_{n\in\mathcal{N}} \ell_n'$. Moreover, by rule (Loc), it must be the case that, for any $n \in \mathcal{N}$, $\mathsf{M}_n \xrightarrow{\ell_n} \mathsf{M}_n'$ with $\ell_n' = n[\ell_n]$ and where each label $\ell_n = \langle\langle D_n \rhd R_n', I_n', C_n\rangle \rhd R_n, I_n, P_n\rangle$ satisfies the applicability conditions of rule (Loc), i.e., $R_n' \subseteq D_n$ and $R_n \subseteq D_n \cup C_n$. By inductive hypotheses, it follows that $\mathcal{F}(n,\mathsf{M}_n) \xrightarrow{n[\ell_n]} \mathcal{F}(n,\mathsf{M}_n')$. Therefore, by repeated applications of rule (Par), we derive the transition $\prod_{n\in\mathcal{N}} \mathcal{F}(n,\mathsf{M}_n) \xrightarrow{\ell} \prod_{n\in\mathcal{N}} \mathcal{F}(n,\mathsf{M}_n')$. We note that the applicability conditions of rule (Sys) are satisfied, because $\ell = \bigcup_{n\in\mathcal{N}} n[\ell_n]$ and each ℓ_n satisfies the analogous applicability conditions of rule (Loc) separately. This allows us to conclude that, by rule (Sys), $\mathcal{F}(\mathsf{P}) \xrightarrow{\ell} \left[\prod_{n\in\mathcal{N}} \mathcal{F}(n,\mathsf{M}_n')\right] = \mathcal{F}(\mathsf{P}')$. Vice versa, let us suppose that $\mathcal{F}(\mathsf{P}) \xrightarrow{\ell} \mathsf{P}''$. Since $\mathcal{F}(\mathsf{P}) = \left[\prod_{n\in\mathcal{N}} \mathcal{F}(n,\mathsf{M}_n)\right]$, by rules (Sys) and (Par), it must be the case that, $\mathcal{F}(n,\mathsf{M}_n) \xrightarrow{\ell_n'} \mathsf{M}_n''$ for any $n \in \mathcal{N}$, with $\mathsf{P}'' = \left[\prod_{n\in\mathcal{N}} \mathsf{M}_n''\right]$, all labels in the family

TABLE 8. The multi-agent Lotka-Volterra reaction systems model. The model consists of n independent agents, $1 \leq k \leq n$, with the reaction set for agent A_k shown in the table.

Label	Reactant set	Inhibitor set	Product set
(Prey multiplies)	$\{\mathsf{r}_k, \mathsf{f}_k\}$	$\{\times\mathsf{r}_k, \times\mathsf{f}_k\}$	$\{\mathsf{R}_k, \mathsf{f}_k\} \cup \mathcal{XR}_k$
(Prey being hunted)	$\{\mathsf{R}_k, \mathsf{F}_k\}$	$\varnothing$	$\{\mathsf{r}_k, \mathsf{F}_k\} \cup \mathcal{XF}_k$
	$\{\mathsf{r}_k, \times\mathsf{r}_k F_k\}$	$\varnothing$	$\{\mathsf{r}_k, \mathsf{F}_k\} \cup \mathcal{XF}_k$
	$\{\mathsf{R}_k, \mathsf{f}_k, \times\mathsf{f}_k\}$	$\varnothing$	$\{\mathsf{r}_k, \mathsf{F}_k\} \cup \mathcal{XF}_k$
	$\{\mathsf{r}_k, \times\mathsf{r}_k, \mathsf{f}_k, \times\mathsf{f}_k\}$	$\varnothing$	$\{\mathsf{r}_k, \mathsf{F}_k\} \cup \mathcal{XF}_k$
(Predator multiplies)	$\{\mathsf{R}_k, \mathsf{f}_k\}$	$\{\times\mathsf{f}_k\}$	$\{\mathsf{R}_k, \mathsf{F}_k\} \cup \mathcal{XR}_k \cup \mathcal{XF}_k$
	$\{\mathsf{r}_k, \times\mathsf{r}_k, \mathsf{f}_k\}$	$\{\times\mathsf{f}_k\}$	$\{\mathsf{R}_k, \mathsf{F}_k\} \cup \mathcal{XR}_k \cup \mathcal{XF}_k$
(Predator dies)	$\{\mathsf{r}_k, \mathsf{F}_k\}$	$\{\times\mathsf{r}_k\}$	$\{\mathsf{r}_k, \mathsf{f}_k\}$
	$\{\mathsf{r}_k, \mathsf{f}_k, \times\mathsf{f}_k\}$	$\{\times\mathsf{r}_k\}$	$\{\mathsf{r}_k, \mathsf{f}_k\}$

TABLE 9. The simplified multi-agent Lotka-Volterra reaction systems model using guarded context to obtain the equivalence-reduced interactive processes and state transition graph.

Label	Reactant set	Inhibitor set	Product set
(Prey multiplies)	$\{\mathsf{r}_k, \mathsf{f}_k\}$	$\{\times\mathsf{r}_k, \times\mathsf{f}_k\}$	$\{\mathsf{R}'_k, \mathsf{f}'_k\} \cup \mathcal{XR}_k$
(Prey being hunted)	$\{\mathsf{R}_k, \mathsf{F}_k\}$	$\{\}$	$\{\mathsf{r}'_k, \mathsf{F}'_k\} \cup \mathcal{XF}_k$
(Predator multiplies)	$\{\mathsf{R}_k, \mathsf{f}_k\}$	$\{\times\mathsf{f}_k\}$	$\{\mathsf{R}'_k, F'_k\} \cup \mathcal{XR}_k \cup \mathcal{XF}_k$
(Predator dies)	$\{\mathsf{r}_k, \mathsf{F}_k\}$	$\{\times\mathsf{r}_k\}$	$\{\mathsf{r}'_k, \mathsf{f}'_k\}$

$\{\ell'_n\}_{n\in\mathcal{N}}$ are pairwise compatible (i.e., $\ell'_n \frown \ell'_m$ for any $n \neq m$) and $\ell = \bigcup_{n\in\mathcal{N}} \ell'_n$. Moreover, by rule (*Sys*), it must be the case that the label $\ell = \langle\langle D \rhd R', I', C\rangle \rhd R, I, P\rangle$ satisfies the applicability conditions $R' \subseteq D$ and $R \subseteq D \cup C$. By inductive hypotheses, it must be the case that, for any $n \in \mathcal{N}$, there exist a transition $\mathsf{M}_n \xrightarrow{\ell_n} \mathsf{M}'_n$ such that $\ell'_n = n[\ell_n]$ and $\mathsf{M}''_n = \mathcal{F}(n, \mathsf{M}'_n)$. Letting $\ell_n = \langle\langle D_n \rhd R'_n, I'_n, C_n\rangle \rhd R_n, I_n, P_n\rangle$, this means that $\bigcup_n n[R'_n] \subseteq \bigcup_n n[D_n]$ and $\bigcup_n n[R_n] \subseteq \bigcup_n n[D_n \cup C_n]$. Since, by construction, all the sets D_n, R'_n, I'_n, C_n, R_n, and I_n can only contain entities in S (not in $\hat{S}$, contrary to P_n), this guarantees that, for any $n \in \mathcal{N}$, we have $R'_n \subseteq D_n$ and $R_n \subseteq D_n \cup C_n$. Therefore, for any $n \in \mathcal{N}$, we can apply the rule (*Loc*) to derive $n[\mathsf{M}_n] \xrightarrow{n[\ell_n]} n[\mathsf{M}'_n]$. We conclude by repeatedly applying the rule (*DPar*) to derive the transition $\mathsf{P} \xrightarrow{\ell} \prod_{n\in\mathcal{N}} n[\mathsf{M}'_n]$ such that $\mathcal{F}(\prod_{n\in\mathcal{N}} n[\mathsf{M}'_n]) = \left[\prod_n \mathcal{F}(n, \mathsf{M}'_n)\right] = \left[\prod_n \mathsf{M}''_n\right] = \mathsf{P}''$. Note that the applicability conditions for (*DPar*) are satisfied because we know that all labels in the family $\{\ell'_n\}_{n\in\mathcal{N}} = \{n[\ell_n]\}_{n\in\mathcal{N}}$ are pairwise compatible.

APPENDIX
A REACTION SYSTEM IMPLEMENTATION OF THE DISTRIBUTED LOTKA-VOLTERRA MODEL

The main text describes the first reaction system-based model of a distributed, multi-agent Lotka-Volterra model. In addition to the process algebraic presentation throughout the paper, we describe the model here using the standard reaction systems notation, as this may be of independent interest to researchers in this field. In this model, we have several communicating, independent agents evolving according to their own Lotka-Volterra model, and whose state may be influenced by contributions from neighbouring agents. We assume the model to have a fixed communication topology in the form of a directed graph, describing which agent can communicate to whom. We also assume a fixed communication policy in the form of labels on the edges of the communication graph, describing whether rabbits or foxes (or both) may be sent along that edge. The predator-prey metaphor that we follow is that where rabbits and/or foxes can escape from one forest to a neighbouring forest if their level is high. In other words, the models we investigate here are such that rabbits (and foxes) can only be communicated along an edge from agent A to agent B in the states where A includes entity R, i.e., a high level of rabbits (F for a high level of foxes, respectively).

Our model consists of agents $A_1, A_2, \ldots, A_n$, $n \geq 2$, each modelled through its own reaction system, each using a distinct set of variables. We use variables $\{\mathsf{r}_k, \mathsf{R}_k, \mathsf{f}_k, \mathsf{F}_k\}$ to denote the (main) variables of agent A_k, $1 \leq k \leq n$, standing for low/high levels of rabbits/foxes. We construct its reaction set starting from the following generic form:

(Prey multiplies) $(\{\mathsf{r}_k, \mathsf{f}_k\}, \{\}, \{\mathsf{R}_k, \mathsf{f}_k\} \cup \mathcal{XR}_k)$,
(Prey being hunted) $(\{\mathsf{R}_k, \mathsf{F}_k\}, \{\}, \{\mathsf{r}_k, \mathsf{F}_k\} \cup \mathcal{XF}_k)$,
(Predator multiplies) $(\{\mathsf{R}_k, \mathsf{f}_k\}, \{\}, \{\mathsf{R}_k, \mathsf{F}_k\} \cup \mathcal{XR}_k \cup \mathcal{XF}_k)$,
(Predator dies) $(\{\mathsf{r}_k, \mathsf{F}_k\}, \{\}, \{\mathsf{r}_k, \mathsf{f}_k\})$.

where sets $\mathcal{XR}_k$ and $\mathcal{XF}_k$, as defined in Section IV, include the information about the topology and the communication policy.

In the first reaction ('*prey multiplies*'), A_k will get in its next state a high level of rabbits (denoted as R_k) and, as such, it may also send out a surplus of rabbits to its neighbours, depending on the model's communication policy (e.g., whether rabbits are allowed to escape from A_k and where to). Similarly, in the second reaction ('*prey being hunted*'), A_k will get in its next state a high level of foxes, and it may send out a surplus of foxes. In the third reaction ('*predator multiplies*') both rabbits and foxes get to a high level in the next state, and a surplus of both may be sent out. Nothing is being sent out in the fourth reaction because both rabbits and foxes get to a low level in the next state.

To make this scheme concrete, consider a 2-agent LV model where A_1 can send out rabbits to A_2, but not foxes, while A_2 cannot send out anything to A_1. This variant of

the model corresponds to the following communication sets: $\mathcal{XR}_1 = \{\mathsf{xr}_2\}$, $\mathcal{XF}_1 = \mathcal{XR}_2 = \mathcal{XF}_2 = \emptyset$.

In our full formulation of the distributed LV model, we must account for the idea that having xr_k in the current state contributes in the same way in determining the next state as having R_k, and that xf_k has the same influence as having F_k. This leads to the full formulation of our multi-agent LV model in Table 8. The same model can be given the simplified presentation in Table 9 using guarded contexts.

REFERENCES

[1] A. Ehrenfeucht and G. Rozenberg, "Reaction systems," *Fundamenta Informaticae*, vol. 75, no. 1–4, pp. 263–280, Jan. 2007. [Online]. Available: https://content.iospress.com/articles/fundamenta-informaticae/fi75-1-4-15

[2] A. Ehrenfeucht, I. Petre, and G. Rozenberg, "Reaction systems: A model of computation inspired by the functioning of the living cell," in *Role Theory Comput. Science: Essays Dedicated To Janusz Brzozowski*, Jan. 2017, pp. 1–32, doi: 10.1142/9789813148208_0001.

[3] S. Azimi, B. Iancu, and I. Petre, "Reaction system models for the heat shock response," *Fundamenta Informaticae*, vol. 131, nos. 3–4, pp. 299–312, Jun. 2014, doi: 10.3233/fi-2014-1016.

[4] L. Corolli, C. Maj, F. Marini, D. Besozzi, and G. Mauri, "An excursion in reaction systems: From computer science to biology," *Theor. Comput. Sci.*, vol. 454, pp. 95–108, Oct. 2012, doi: 10.1016/j.tcs.2012.04.003.

[5] R. Barbuti, R. Gori, F. Levi, and P. Milazzo, "Investigating dynamic causalities in reaction systems," *Theor. Comput. Sci.*, vol. 623, pp. 114–145, Apr. 2016, doi: 10.1016/j.tcs.2015.11.041.

[6] S. Ivanov and I. Petre, "Controllability of reaction systems," *J. Membrane Comput.*, vol. 2, no. 4, pp. 290–302, Dec. 2020, doi: 10.1007/s41965-020-00055-x.

[7] J. K. F. Bowles, L. Brodo, R. Bruni, M. Falaschi, R. Gori, and P. Milazzo, "Enhancing reaction systems with guards for analysing comorbidity treatment strategies," *Comput. Methods Syst. Biol.*, vol. 14971, pp. 27–44, Sep. 2024, doi: 10.1007/978-3-031-71671-3_3.

[8] F. Okubo and T. Yokomori, "The computational capability of chemical reaction automata," *Natural Comput.*, vol. 15, no. 2, pp. 215–224, Jun. 2016, doi: 10.1007/s11047-015-9504-7.

[9] S. Azimi, C. Gratie, S. Ivanov, and I. Petre, "Dependency graphs and mass conservation in reaction systems," *Theor. Comput. Sci.*, vol. 598, pp. 23–39, Sep. 2015, doi: 10.1016/j.tcs.2015.02.014.

[10] A. Męski, M. Koutny, and W. Penczek, "Towards quantitative verification of reaction systems," in *Unconventional Computation and Natural Computation* (Lecture Notes in Computer Science), vol. 9726, M. Amos and A. Condon, Eds., Cham, Switzerland: Springer, Jul. 2016, pp. 142–154, doi: 10.1007/978-3-319-41312-9_12.

[11] A. Męski, M. Koutny, Ł. Mikulski, I. Petre, W. Penczek, and M. Piatkowski, "Model checking for distributed reaction systems with temporal-epistemic properties," Tech. Rep., 2025.

[12] L. Brodo, R. Bruni, and M. Falaschi, "SOS rules for equivalences of reaction systems," in *Functional and Constraint Logic Programming* (Lecture Notes in Computer Science), vol. 12560. Cham, Switzerland: Springer, 2021, pp. 3–21, doi: 10.1007/978-3-030-75333-7_1.

[13] L. Brodo, R. Bruni, M. Falaschi, R. Gori, F. Levi, and P. Milazzo, "Quantitative extensions of reaction systems based on SOS semantics," *Neural Comput. Appl.*, vol. 35, no. 9, pp. 6335–6359, Mar. 2023, doi: 10.1007/s00521-022-07935-6.

[14] L. Brodo, R. Bruni, and M. Falaschi, "Dynamic slicing of reaction systems based on assertions and monitors," in *Proc. 25th Int. Symp. Practical Aspects Declarative Lang.*, 2023, pp. 107–124, doi: 10.1007/978-3-031-24841-2_8.

[15] G. Păun and M. J. Pérez-Jiménez, "Towards bridging two cell-inspired models: P systems and R systems," *Theor. Comput. Sci.*, vol. 429, pp. 258–264, Apr. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397511010127

[16] G. Păun, M. J. Pérez-Jiménez, and G. Rozenberg, "Bridging membrane and reaction systems-further results and research topics," *Fundamenta Informaticae*, vol. 127, nos. 1–4, pp. 99–114, Jan. 2013, doi: 10.3233/FI-2013-898.

[17] C. Conradi and C. Pantea, *Multistationarity in Biochemical Networks: Results, Analysis, and Examples*. New York, NY, USA: Academic, 2019, pp. 279–317, doi: 10.1016/B978-0-12-814066-6.00009-X.

[18] K.-J. Lee, W. D. McCormick, J. E. Pearson, and H. L. Swinney, "Experimental observation of self-replicating spots in a reaction–diffusion system," *Nature*, vol. 369, no. 6477, pp. 215–218, May 1994, doi: 10.1038/369215a0.

[19] R. Barbuti, R. Gori, P. Milazzo, and L. Nasti, "A survey of gene regulatory networks modelling methods: From differential equations, to Boolean and qualitative bioinspired models," *J. Membrane Comput.*, vol. 2, no. 3, pp. 207–226, Oct. 2020, doi: 10.1007/s41965-020-00046-y.

[20] J. Kleijn, M. Koutny, and G. Rozenberg, "Modelling reaction systems with Petri nets," in *Proc. Int. Workshop Biol. Processes Petri Nets (BioPPN)*, 2011, pp. 36–52. [Online]. Available: https://ceur-ws.org/Vol-724/

[21] A. Clark, S. Gilmore, J. Hillston, and M. Tribastone, *Stochastic Process Algebras*. Berlin, Germany: Springer, 2007, pp. 132–179, doi: 10.1007/978-3-540-72522-0_4.

[22] A. Męski, M. Koutny, and W. Penczek, "Model checking for temporal-epistemic properties of distributed reaction systems," School Comput., Univ. Newcastle Upon Tyne, Tyne, U.K., Tech. Rep. 1526, 2019.

[23] P. Bottoni, A. Labella, and G. Rozenberg, "Networks of reaction systems," *Int. J. Found. Comput. Sci.*, vol. 31, no. 1, pp. 53–71, Jan. 2020, doi: 10.1142/S0129054120400043.

[24] E. Csuhaj-Varjú and P. K. Sethy, "Communicating reaction systems with direct communication," in *Proc. 21st Int. Conf. Membrane Comput.*, Sep. 2021, pp. 17–30, doi: 10.1007/978-3-030-77102-7_2.

[25] B. Aman, "From networks of reaction systems to communicating reaction systems and back," in *Machines, Computations, and Universality*, vol. 13419. Cham, Switzerland: Springer, 2022, pp. 42–57, doi: 10.1007/978-3-031-13502-6_3.

[26] L. Ciencialová, L. Cienciala, and E. Csuhaj-Varjú, "Languages of distributed reaction systems," in *Proc. Int. Conf. Mach., Computations, Universality*. Cham, Switzerland: Springer, 2022, pp. 75–90, doi: 10.1007/978-3-031-13502-6_5.

[27] L. Ciencialová, L. Cienciala, and E. Csuhaj-Varjú, "Language classes of extended distributed reaction systems," *Int. J. Found. Comput. Sci.*, vol. 34, pp. 1–24, Jul. 2023, doi: 10.1142/S0129054123460024.

[28] B. Aman, "Relating various types of distributed reaction systems," *Int. J. Found. Comput. Sci.*, vol. 34, pp. 1–16, Oct. 2023, doi: 10.1142/S0129054123470044.

[29] E. Csuhaj-Varjú and G. Vaszil, "Variants of distributed reaction systems," *Natural Comput.*, vol. 23, no. 2, pp. 269–284, Jun. 2024, doi: 10.1007/s11047-024-09974-5.

[30] A. Alhazov, E. Csuhaj-Varjú, and P. K. Sethy, "Generalized distributed reaction systems," *Comput. Sci. J. Moldova*, vol. 32, no. 3, pp. 315–331, Nov. 2024, doi: 10.56415/csjm.v32.17.

[31] C. A. R. Hoare, *Communicating Sequential Processes*. Upper Saddle River, NJ, USA: Prentice-Hall, 1985.

[32] R. Milner, J. Parrow, and D. Walker, "A calculus of mobile processes, I," *Inf. Comput.*, vol. 100, no. 1, pp. 1–77, Sep. 1992, doi: 10.1016/0890-5401(92)90008-4.

[33] F. Ciocchetta and J. Hillston, "Bio-PEPA: A framework for the modelling and analysis of biological systems," *Theor. Comput. Sci.*, vol. 410, nos. 33–34, pp. 3065–3084, Aug. 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397509001662

[34] L. Brodo, R. Bruni, and M. Falaschi, "A process algebraic approach to reaction systems," *Theor. Comput. Sci.*, vol. 881, pp. 62–82, Aug. 2021, doi: 10.1016/j.tcs.2020.09.001.

[35] L. Brodo, R. Bruni, M. Falaschi, R. Gori, F. Levi, and P. Milazzo, "Exploiting modularity of SOS semantics to define quantitative extensions of reaction systems," in *Proc. TPNC*, vol. 13082. Berlin, Germany: Springer, 2021, pp. 15–32, doi: 10.1007/978-3-030-90425-8_2.

[36] L. Brodo, R. Bruni, and M. Falaschi, "A logical and graphical framework for reaction systems," *Theor. Comput. Sci.*, vol. 875, pp. 1–27, Jul. 2021, doi: 10.1016/j.tcs.2021.03.024.

[37] D. Ballis, L. Brodo, and M. Falaschi, "Modeling and analyzing reaction systems in Maude," *Electronics*, vol. 13, no. 6, p. 1139, Mar. 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/6/1139

[38] G. Pardini, R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and S. Tini, "Compositional semantics and behavioural equivalences for reaction systems with restriction," *Theor. Comput. Sci.*, vol. 551, pp. 1–21, Sep. 2014, doi: 10.1016/j.tcs.2014.04.010.

[39] (2025). *BioReSolve, a Prolog Interpreter for Reaction Systems Analysis*. Accessed: Jun. 2025. [Online]. Available: https://doi.org/10.5281/zenodo.15696275
[40] L. Brodo, R. Bruni, and M. Falaschi, "A framework for monitored dynamic slicing of reaction systems," *Natural Comput.*, vol. 23, no. 2, pp. 217–234, Jun. 2024, doi: 10.1007/s11047-024-09976-3.
[41] A. Męski, W. Penczek, and G. Rozenberg, "Model checking temporal properties of reaction systems," *Inf. Sci.*, vol. 313, pp. 22–42, Aug. 2015, doi: 10.1016/j.ins.2015.03.048.
[42] A. Męski, M. Koutny, and W. Penczek, "Verification of linear-time temporal properties for reaction systems with discrete concentrations," *Fundamenta Informaticae*, vol. 154, nos. 1–4, pp. 289–306, Aug. 2017, doi: 10.3233/fi-2017-1567.
[43] M. S. Nobile, A. E. Porreca, S. Spolaor, L. Manzoni, P. Cazzaniga, G. Mauri, and D. Besozzi, "Efficient simulation of reaction systems on graphics processing units," *Fundamenta Informaticae*, vol. 154, nos. 1–4, pp. 307–321, Aug. 2017. [Online]. Available: https://journals.sagepub.com/action/showAbstract
[44] S. Ivanov, V. Rogojin, S. Azimi, and I. Petre, "WEBRSIM: A web-based reaction systems simulator," in *Enjoying Natural Computing* (Lecture Notes in Computer Science), vol. 11270, C. G. Díaz, A. Riscos-Núñez, G. Paun, G. Rozenberg, and A. Salomaa, Eds., Berlin, Germany: Springer, 2018, pp. 170–181, doi: 10.1007/978-3-030-00265-7_14.
[45] Z. Shang, S. Verlan, J. Lu, Z. Wei, and M. Zhou, "FPGA implementation of reaction systems," *Electronics*, vol. 13, no. 24, p. 4929, Dec. 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/24/4929
[46] A. Lotka, "Undamped oscillations derived from the law of mass action," *J. Amer. Chem. Soc.*, vol. 42, no. 8, pp. 1595–1599, Aug. 1920, doi: 10.1021/ja01453a010.
[47] A. J. Lotka, "Elements of physical biology," *Sci. Prog. 20th Century*, vol. 21, no. 82, pp. 341–343, Jun. 1925. [Online]. Available: http://www.jstor.org/stable/43430362
[48] V. Volterra, "Variazioni e fluttuazioni del numero d'individui in specie animali conviventi," *Memoria della Reale Accademia Nazionale dei Lincei*, vol. Ser., 6, no. 2, pp. 31–113, 1926.
[49] H.-K. Chao, "Three kinds of the Lotka–Volterra model transfer from biology to economics," *Synthese*, vol. 202, no. 4, p. 124, Oct. 2023, doi: 10.1007/s11229-023-04341-w.
[50] L. Brodo, R. Bruni, M. Falaschi, R. Gori, and P. Milazzo, "Attractor and slicing analysis of a T-Cell differentiation model based on reaction systems," in *From Data to Models Back* (Lecture Notes in Computer Science), vol. 14618, G. Broccia and A. Cerone, Eds., Cham, Switzerland: Springer, 2025, pp. 69–89, doi: 10.1007/978-3-031-87217-4_4.
[51] *Repository for the Lotka-Volterra Models Analyzed in This Paper*. Accessed: May 2025. [Online]. Available: https://github.com/greenAsrai/flat_communication_RS_IEEE/
[52] D. Genova, H. J. Hoogeboom, and N. Jonoska, "A graph isomorphism condition and equivalence of reaction systems," *Theor. Comput. Sci.*, vol. 701, pp. 109–119, Nov. 2017, doi: 10.1016/j.tcs.2017.05.019.
[53] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, "Cytoscape: A software environment for integrated models of biomolecular interaction networks," *Genome Res.*, vol. 13, no. 11, pp. 2498–2504, Nov. 2003, doi: 10.1101/gr.1239303.
[54] V. Mitrana, M. Paun, I. Petre, and A.-M. Prelipcean, "Quantitative reaction systems," in *Proc. 5th Int. Conf. Innov. Res. Appl. Sci., Eng. Technol. (IRASET)*, May 2025, pp. 1–6.
[55] P. Bottoni, V. Mitrana, and I. Petre, "Multiset computing with reaction systems," in *Festschrift Dedicated to Erzsébet Csuhaj-Varjú* Cham, Switzerland: Springer, 2025.
[56] J. Kleijn, M. Koutny, Ł. Mikulski, and G. Rozenberg, *Reaction Systems, Transition Systems, and Equivalences* (Lecture Notes in Computer Science), vol. 11011. Cham, Switzerland: Springer, 2018, pp. 63–84, doi: 10.1007/978-3-319-98355-4_5.
[57] D. Genova, H. J. Hoogeboom, and J. Kleijn, "Comparing reactions in reaction systems," *Theor. Comput. Sci.*, vol. 881, pp. 83–96, Aug. 2021, doi: 10.1016/j.tcs.2020.11.050.
[58] D. Genova, H. J. Hoogeboom, and J. Kleijn, "Functional equivalence and a cover relation for reaction systems," *Theor. Comput. Sci.*, vol. 1004, Jul. 2024, Art. no. 114633, doi: 10.1016/j.tcs.2024.114633.
[59] A. Ehrenfeucht and G. Rozenberg, "Introducing time in reaction systems," *Theor. Comput. Sci.*, vol. 410, nos. 4–5, pp. 310–322, Feb. 2009, doi: 10.1016/j.tcs.2008.09.043.

LINDA BRODO is currently an Associate Professor in computer science at the University of Sassari. Her studies include the theory of abstract parallel programming languages for describing and verifying distributed systems using theoretical tools such as process algebras and rewriting systems. The formal rigour and expressiveness of such theoretical tools enable the treatment of systems in a wide range of applications. She is the author of more than 16 articles in international journals and has presented more than 45 papers at international conferences. Her research interests include formal methods for modelling and analyzing complex systems.

ROBERTO BRUNI is currently a Professor at the Computer Science Department, University of Pisa. He has contributed to several areas, including Petri nets, rewriting logic, process calculi, reaction systems, service-oriented computing, multiparty interactions, abstract interpretation, and program logics. He has co-authored over 170 publications, including textbooks and research monographs. His research interests include modeling and verification of concurrent, distributed, adaptive, and open systems to natural computing and bio-inspired models of computation.

MORENO FALASCHI received the Ph.D. degree in computer science from the University of Pisa, in 1988. He is currently a Full Professor of computer science at the Department of Information Engineering and Mathematics, University of Siena, Italy. He teaches computer science courses at the B.A., M.Sc., and Ph.D. levels. He has been an advisor to several Ph.D. students (12) and postdoctoral researchers and is the leader of the research group on computational models for biomedical applications. He has published more than 100 papers in major scientific international conferences and journals, mainly published by ACM, Elsevier, Springer, and IEEE. His research interests include artificial intelligence (computational symbolic reasoning) with a focus on dynamic computational models of biological systems; in-silico simulations; computational systems biology; and design, analysis, and implementation of (concurrent) constraint languages. He has been the local PI of several national and international research projects funded by Italian MUR or the European Union. He is the Proposer and the Director of the Master Program in Bioinformatics and Data Science at the University of Siena. He is a Reviewer of the major international conferences and journals in his research areas. He has been a PC member (or the chair) of some of the major conferences in his research areas. He has been an invited Visiting Scientist at the Weizmann Institute of Science, Israel; IBM Watson Research Center, USA; and École Polytechnique, France.

ION PETRE is currently a Professor of mathematics at the University of Turku, Finland. He combines methods from systems biology, the theory of computation, and data-driven modelling to investigate emergent behaviour in complex biological and medical systems. He has published more than 150 papers in international journals, conference proceedings, and collective books, and has edited 26 books and special issues of journals. His research interests include the intersection of mathematics, computer science, and biology, with a broad scope that includes bio-inspired computing, network-based models of disease, and the use of artificial intelligence in medical diagnosis and treatment selection.