Proceedings of the First International Workshop on

Application of Membrane Computing, Concurrency and Agent-based Modelling in Population Biology

(AMCA-POP 2010)

Satellite event of the 11th Conference on Membrane Computing (CMC11)

Edited by Paolo Milazzo and Mario de J. Peréz Jiménez

Jena, Germany – August 25th, 2010

This volume has been printed with the support of a grant from Fondazione Monte dei Paschi di Siena for the research project “Un approccio etologico/computazionale allo studio dei meccanismi dell’evoluzione delle specie animali” hosted by the Museum of Natural History of the University of Pisa.

July, 2010

Contents

Preface

This volume contains the papers presented at the first International Workshop on Applications of Membrane Computing, Concurrency and Agent-based Modelling in Population Biology (AMCA-POP 2010) held in Jena, Germany on August 25th, 2010 as a satellite event of the 11th Conference on Membrane Computing (CMC11).

The aim of the workshop is to investigate whether formal modelling and analysis techniques could be applied with profit to systems of interest for population biology and ecology. The considered modelling notations include membrane systems, Petri nets, agent-based notations, process calculi, automata-based notations, rewriting systems and cellular automata. Such notations enable the application of analysis techniques such as simulation, model checking, abstract interpretation and type systems to study systems of interest in disciplines such as population biology, ecosystem science, epidemiology, genetics, sustainability science, evolution and other disciplines in which population dynamics and interactions with the environment are studied. Papers contain results and experiences in the modelling and analysis of systems of interest in these fields.

The workshop program includes three invited talks by Mats Gyllenberg (University of Helsinki, Finland), Giancarlo Mauri (University of Milano-Bicocca, Italy), and Jamal Hisham Hashim (UNU-IIGH, Malaysia). A paper by Giancarlo Mauri and colleagues related with his invited talk is included in this volume, together with four regular peer-reviewed contributions. The volume contains also two extended abstracts of poster presentations. Mauri's paper and the four regular contributions will be published also in a volume of the EPTCS series (see `http://www.eptcs.org/`).

We wish to thank the invited speakers, the authors of the contributed papers, the members of the program committee, the additional reviewers Thomas Anung Basuki, Federico Buti, Paolo Cazzaniga, Andrzej Mizera, and Ashutosh Trivedi, and the organizers of CMC11 for their contributions and support.

Paolo Milazzo and Mario de J. Peréz Jiménez

AMCA-POP 2010 Program Committee

- Roberto Barbuti (University of Pisa, Italy)
- Antonio Cerone (UNU-IIST, Macao SAR, China)
- Gabriel Ciobanu (Romanian Academy, Iasi, Romania)
- Mariangiola Dezani-Ciancaglini (University of Turin, Italy)
- Gabi Escuela (F.S. University, Jena, Germany)
- Mats Gyllenberg (University of Helsinki, Finland)
- Marta Kwiatkowska (Oxford University, UK)
- Paolo Milazzo (University of Pisa, Italy)
- Mario de J. Peréz Jiménez (University of Sevilla, Spain)
- Ion Petre (Abo Akademi, Finland)
- Michael Sonnenschein (Oldenburg University, Germany)
- Luca Tesei (University of Camerino, Italy)

An Analysis on the Influence of Network Topologies on Local and Global Dynamics of Metapopulation Systems

Daniela Besozzi[a] Paolo Cazzaniga[b]
Dario Pescini[b] Giancarlo Mauri[b]

[a]Università degli Studi di Milano
Dipartimento di Informatica e Comunicazione
Via Comelico 39, 20135 Milano, Italy
besozzi@dico.unimi.it

[b]Università degli Studi di Milano-Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Viale Sarca 336, 20126 Milano, Italy
cazzaniga/pescini/mauri@disco.unimib.it

Metapopulations are models of ecological systems, describing the interactions and the behavior of populations that live in fragmented habitats. In this paper, we present a model of metapopulations based on the multivolume simulation algorithm tau-DPP, a stochastic class of membrane systems, that we utilize to investigate the influence that different habitat topologies can have on the local and global dynamics of metapopulations. In particular, we focus our analysis on the migration rate of individuals among adjacent patches, and on their capability of colonizing the empty patches in the habitat. We compare the simulation results obtained for each habitat topology, and conclude the paper with some proposals for other research issues concerning metapopulations.

1 Introduction

The field of metapopulations ecology deals with the study of spatial systems describing the behavior of interacting populations that live in fragmented habitats [17]. The purpose of these models is to understand how the local and global dynamics of metapopulation systems, usually balanced between local extinctions and new colonizations of unoccupied patches, depend on the spatial arrangement of the habitat. Consequently, relevant insights into related fields of ecological research, such as evolutionary ecology or conservation and landscape management, can be achieved. Indeed, the topology of fragmented habitats potentially holds relevant implications for the persistence of populations, and their robustness against natural or anthropogenic disturbance [36].

Recently, in addition to ever increasing applications of graph-based methods for the analysis of complex networks in cell biology [1, 2], graph theory has also been applied to the study of metapopulations systems. In graph models of metapopulations, nodes are used to represent habitat patches, and graph edges are used to denote some functional connections between patches (typically related to the dispersal of individuals). Attributes can be associated to nodes, describing the quality or dimension of patches, while different types of edges can be exploited to represent the distance between connected patches, the rate of dispersal between a couple of patches, or simply whether two patches are connected or not.

Metapopulation models using graph-based methods [36, 15] are simple to implement and require relatively few data for their definition, while individual-based models implement more detailed aspects,

To appear in:
AMCA-POP 2010 – Electronic Proceedings in Theoretical Computer Science (EPTCS)

concerning the nature and the interaction of populations [34, 4]. Both types of modeling approaches are useful for the analysis of specific features of metapopulations but, while the first focuses on the properties of the habitat topology, the second is more concerned with the emergent dynamics. In this paper, we present a stochastic multivolume model of metapopulations, which integrates the explicit representation of interactions between the individuals of the populations – and therefore allows to simulate the emergent local and global dynamics – with a graph description of the habitat topology – which allows to investigate the influence of distinct spatial structures on the dynamics.

This model, which represents a simplified extension of a previous metapopulation model that we introduced in [7, 6], is based on the multivolume stochastic simulation algorithm tau-DPP [11, 8], a stochastic class of membrane systems. Membrane systems, or P systems, were introduced in [27] as a class of unconventional computing devices of distributed, parallel and nondeterministic type, inspired by the compartmental structure and the functioning of living cells. The basic model consists of a membrane structure where multisets of objects evolve according to given evolution rules. A comprehensive overview of P systems and of its many applications in various research areas, ranging from Biology to Linguistics to Computer Science, can be found in [28, 12, 29].

In tau-DPP, the distinct compartments of any multivolume model can be arranged according to a specified hierarchy (e.g., a membrane structure), under the additional assumption that the topological structure and the volume dimensions do not change during the system evolution (each volume is assumed to satisfy the standard requirements of the classical stochastic simulation algorithm, see [16] and [5] for more details). Inside each volume, two different types of rules can be defined: the *internal rules*, which modify the objects contained inside the volume where they take place (in the case of metapopulation, they describe the growth and death of population individuals according to the Lotka-Volterra model of preys and predators), and the *communication rules*, which are used to move the objects between adjacent volumes (in the case of metapopulation, they describe the migration of population individuals).

In this paper, tau-DPP is exploited to analyze the emergent dynamics of metapopulation systems, where the focus is on the influence that the topology of patches has on the migration of individuals, and their capability to colonize other patches in the habitat. To this purpose, we consider six different habitat topologies, formally described by graph structures, and analyze how the topological structure of patch-to-patch connections, and the rate of individual dispersal between connected patches, influence the local and global dynamics of a metapopulation. In particular, we will first consider how a given topology and a fixed dispersal rate between patches can influence the prey-predators dynamics, and then we will focus on the colonization of empty patches, starting from the dispersal of predators that live in a few patches which occupy peculiar positions in the given network topology.

The paper is structured as follows: in Section 2 we present the concept of metapopulations in Ecology, and then describe the multivolume model of metapopulations by focusing, in particular, to the different habitat topologies. In Section 3 we will show the simulation results concerning the influence of these habitat topologies on the emergent dynamics of metapopulations, considering the effects of predators dispersal and colonization. Finally, in Section 4 we conclude the paper with some final remarks and several proposals for further research issues concerning metapopulations.

2 Metapopulations

In this section, we first provide a brief introduction to the most relevant features of metapopulations, concerning both the topology of the habitats and the emergent dynamics. Then, we describe the modeling approach used in this paper, that is based on a stochastic class of membrane systems, which will be used

in Section 3 to analyze the influence of different network topologies on the dynamics of metapopulations.

2.1 Dynamical models of interacting populations in Ecology

Since its introduction in [22], the concept of metapopulations (also called *multi-patch systems*) has been extensively applied in Ecology to analyze the behavior of interacting populations, to the purpose of determining how fragmented habitats can influence various aspects of these systems, such as local and global population persistence, or the evolution of species [18]. Lately, this topic has been largely employed for other populations species, living in both natural and artificial/theoretical fragmented landscapes [17].

A metapopulation consists of local populations, living in spatially separated habitats called *patches* – which can be characterized by different areas, quality or isolation – connected each other through a *dispersal pool*, which is the spatial place where individuals from a population spend some lifetime during the migration among patches. In multi-patch systems, two principal types of dynamics exist: on the one hand, the individuals of the different populations can have *local* interactions inside each patch (according to a given dynamical model, e.g., the Lotka-Volterra system of interaction between preys and predators [25]); on the other hand, the dispersal of individuals among mutually connected patches can influence the *global* behavior of the whole system [20, 21, 33, 37]. The dispersal of individuals, which is usually dependent on the distance between patches, may reduce the local population growth, and thus increase the extinction risk, which can be due also to environmental and demographical stochasticity. Hence, the persistence of populations is assumed to be balanced between local extinctions and the process of colonization, that is, the establishment of new populations in empty patches [17].

Several theoretical frameworks for metapopulation analysis have been defined up to now, remarking specific properties of multi-patch systems which have been either explicitly or implicitly considered in these modeling methods (see, e.g., [14, 17, 24, 19] for further details). For instance, referring to the landscape, most theoretical models take care of the spatial structure of the habitat, the local quality of the environment, the patch areas and their mutual connectivity (or isolation), in order to capture the effect of habitat fragmentation on species persistence. In fact, good local conditions can determine the growth and the survival of populations inside the patches, and high patch connectivity can decrease local extinction risk. Moreover, as dispersal and colonization are distance-dependent elements, they can be used to account for the importance of real landscape structures. Referring to population interactions and dynamics, colonization can depend or not on the cooperation of migrating individuals (in the first case, it is called “Allee effect”). Models not accounting for within-patch dynamics – but only assuming whether a patch is occupied or not – usually consider local dynamics on a faster time scale with respect to the global dynamics, and also neglect the dependence of colonization and extinction rates on population sizes. Finally, regional stochasticity can account for “bad” or “good” years over the local environmental quality, which depends on, e.g., the weather conditions which affect sustenance resource availability and, once more, they can influence the growth and survival of populations.

Recently, graph-based models for metapopulations have started to be more and more defined because of the intuitive and visual way they hold for the representation of these ecological systems (see [36, 23, 35] and references therein). In these models, nodes represent habitat patches and graph edges denote functional connections between patches (typically related to the dispersal of individuals). In addition, attributes can be associated to nodes, describing the quality or dimension of patches, and different types of edges can be adopted to represent the distance between connected patches, the rate of dispersal between a couple of patches, or simply whether two patches are connected or not. These models allow to make insights into the features of habitat distribution, such as the predominant importance of some nodes or clusters of nodes with respect to other characteristics of metapopulation, like their dynamics, the

vulnerability to disturbance, the persistence of populations according to dispersal, and so on. These results open promising perspective in related research fields as evolutionary ecology, conservation biology, epidemiology, management and design of natural reserves.

2.2 A P system–based model of metapopulations: focusing on network topologies

Most of the issues discussed in Section 2.1 were explicitly considered in our previous model for metapopulations [6, 7]. In those works, metapopulation models were based on a class of membrane systems called DPP [31, 30], which were used to execute qualitative stochastic simulations of the local and global dynamics of metapopulations. In particular, in [7] we introduced a model of metapopulations with predator-prey dynamics, where additional features were used in order to catch and better describe relevant properties of the modeled system. For instance, the regions of the membrane structure were represented as nodes of a weighted graph with attributes, where the weight associated to edges corresponds to the "distance" among connected regions, while attributes specify their surface dimension. These new features are necessary in order to outline the spatial distribution of patches and the relevant additional features associated to them: the dimension of a patch is needed to define the density of the populations living inside that patch, while the distance is needed to identify isolated patches, as well as to define the dispersal rates of migrating individuals. Moreover, by using some rules which do not modify the objects on which they act (the so-called "mute rules"), we modified the classical view of maximal parallelism, by allowing the maximal application of rules but, at the same time, reducing the maximal consumption of objects. The model was applied to investigate some emergent metapopulation behaviors, such as the influence of patch dimension, patch-to-patch distance, stochastic breeding, the dynamics underlying migration and colonization, the effects due to isolated patches, etc. Then, in [6] we extended the analysis of that model by focusing on periodic resource feeding strategies, and compared different systems where either increasing, decreasing, stationary or purely feeding stochastic phases were defined inside each patch. We have shown there, for instance, how the seasonal variance can transform the basic Lotka-Volterra dynamics inside each patch into a more complex dynamics, where the different phases of a feeding cycle can be identified through the effect that they have on the standard oscillations of preys and predators.

In this section, we present a simplified model of metapopulations, which exploits the multivolume stochastic simulation algorithm tau-DPP [11, 5]. With respect to the previous model, here we will not need to use the concept of mute rules, as the probabilistic choice and applications of rules is already embedded in the tau leaping algorithm [10], on which tau-DPP is based. Moreover, we will not consider the presence of the dispersal pool, but we will instead focus our analysis on the direct communication of individuals among interconnected patches, according to some fixed network topologies. In order to compare the influence of each network, we have decided to perform our analysis on a total of 6 patches, spatially arranged in different ways. Namely, we assume that these network topologies can be described by graphs having the same number of nodes, but distinct connections, such as the chain, grid, star, ring, complete or random structure (see graphs a, b, c, d, e, f, respectively, in Fig. 1). From now on, we will refer to the formal data structure by using the term 'graph', and use the term 'network' to denote the topological relationship on each graph.

Formally, each network topology $\nu \in \{a, b, c, d, e, f\}$, can be generally described by a weighted undirected graph $G_\nu = (N_\Delta^\nu, E^\nu, w^\nu)$ where:

- N_Δ^ν is the set of nodes, such that each node $p_i \in N_\Delta^\nu$, i=1, ..., 6, is characterized by a value $\delta(p_i) \in \Delta$ (with Δ being a set of attributes of some kind);
- $E^\nu \subseteq \{(p_i, p_j) \mid p_i, p_j \in N_\Delta^\nu\}$ is the set of (undirected) edges between nodes;

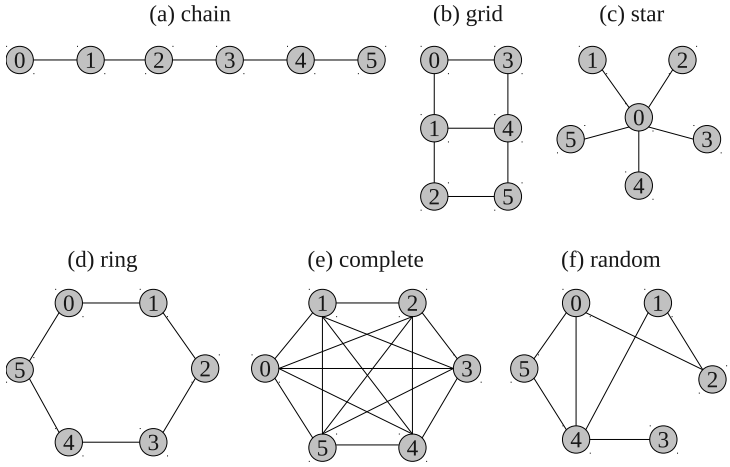


Figure 1: Network topologies.

- $w^\nu : E^\nu \to \mathbb{R}^+$ is the weight function associating a cost to each edge.

In the case of metapopulations, the set of nodes N_Δ^ν coincides with the set of patches, the attribute of a node represents the area of the patch, the edges characterize which patches are directly reachable from any patch (self-edges might exist as well but will not be considered in this work), and the weight $w_{i,j}^\nu$ of an edge (p_i, p_j) represents a cost to measure the effort that individuals have to face when moving from patch p_i to p_j. Given a network topology ν, we denote by $Adj(p_i)^\nu$ the set of nodes that are directly connected to any node p_i, that is, $Adj(p_i)^\nu = \{p_j \in N_\Delta^\nu \mid \exists\,(p_i, p_j) \in E^\nu\}$. We also denote by $deg(p_i)^\nu$ the degree of patch p_i, that is, the number of patches directly connected to p_i (formally, $deg(p_i)^\nu = card(Adj(p_i)^\nu)$). We outline that, in what follows, we will assume that: (1) $w_{i,j}^\nu = 1$ for each $(p_i, p_j) \in E^\nu$ and each $\nu \in \{a, b, c, d, e, f\}$, that is, all edges have the same cost; (2) $\delta(p_i) = 1$ for each $p_i \in N_\Delta^\nu$ and each $\nu \in \{a, b, c, d, e, f\}$, that is, all patches have the same dimension. The rational behind this is that, in this paper, we focus our attention on the influence that different topologies of the habitat network can have on the local and global dynamics of metapopulations, regardless of the local features of each patch, or of the distances between patches. These features might be naturally added in further works related to this model, where real data can be used to define a specific model of some metapopulation systems.

In addition to the chosen network topology, this model of metapopulations also considers the presence of species individuals, which locally interact according to a chosen dynamics, and give rise to global dynamics thanks to the dispersal processes. To this purpose, in this paper we assume that each patch is characterized by the Lotka-Volterra (LV) model describing the interaction between the individuals of two populations, namely preys and predators. Inside each patch, the LV model is described by the following set of internal rules:

$$\begin{aligned} r_1 &: \quad AX \to XX \\ r_2 &: \quad XY \to YY \\ r_3 &: \quad Y \to \lambda \end{aligned}$$

where X denotes the preys, Y denotes the predators, A denotes the sustenance resources and λ is the

empty symbol. Rules r_1 and r_2 model the growth of preys and predators, respectively, while rule r_3 models the death of predators. Each rule is also characterized by a stochastic constants (expressed in $time^{-1}$), that is used – together with the current amounts of individuals occurring in the patch – to evaluate its application probability step by step, according to the tau leaping algorithm (see [10, 11, 8] for more details). All the simulations shown hereafter have been executed using the following values of stochastic constants and of initial amount of preys, predators, and sustenance resources: c_1=0.1, c_2=0.01, c_3=10, X_0=Y_0=1000, A_0=200 (the value of A is fixed for the entire duration of each simulation). The simulations have been performed with the software BioSimWare [5], that implements different stochastic simulation algorithms for both single and multivolume systems. The software is available for free download at http://bimib.disco.unimib.it/index.php/Software.

In Fig. 2 we show the oscillating dynamics (left side) of preys and predators in the single patch, obtained with this choice of parameters, and the corresponding phase space (right side). These figures can be considered as reference to compare and discuss the dynamics obtained in the multi-patch model, as described in Section 3.

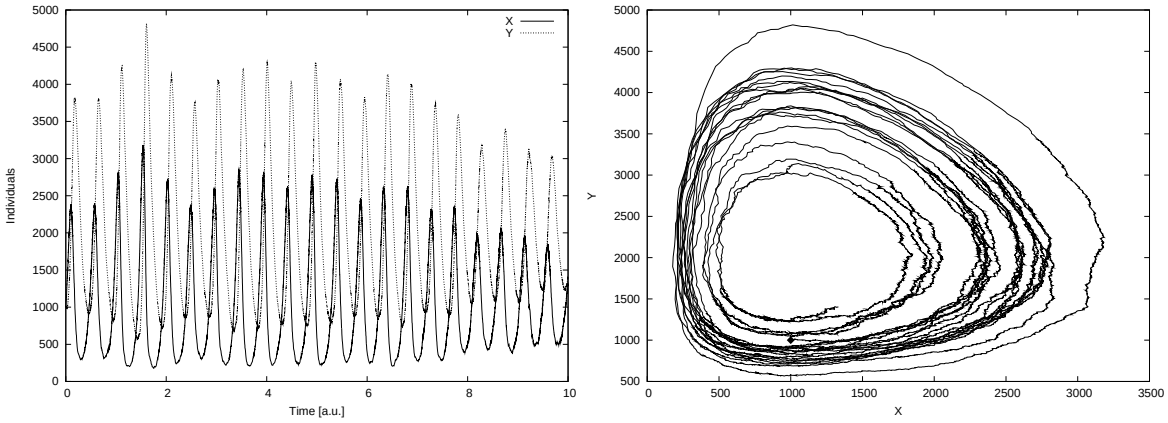


Figure 2: The Lotka-Volterra dynamics in the single patch: oscillations in preys, X, and predators, Y (left side), and corresponding phase space (right side).

The single patch model is then extended to a multi-patch model where, inside each patch p_i of each network topology v, we add as many communication rules as the number of patches connected to p_i (that is, a total of $deg(p_i)^v$ rules inside each patch). These rules are needed to move population individuals among the various patches of the network, thus allowing to analyze the effects of migration and colonization in the metapopulation. This is done by attaching a destination target to each communication rule, specifying the destination patch, as it is usually done in P systems. Formally, in each patch p_i of network v, we add the so-called *dispersal rules*

$$r_{d_{p_j}} : Y \rightarrow (Y, target(p_j)),$$

for each $p_j \in Adj(p_i)^v$. Similarly to the local rules r_1, r_2, r_3, the probability of applying each dispersal rule is determined by using its stochastic constant $c_{d_{p_j}}$, whose values will be given in the next section to consider different migration rates.

3 The influence of network topologies on metapopulation dynamics

In this section we analyze how the topological structure of patch-to-patch connections, and the rate of individual dispersal between connected patches, influence the local and global dynamics of a metapopulation. In particular, in Section 3.1 we consider how a given topology and a fixed dispersal rate can influence the prey-predators dynamics, while in Section 3.2 we focus on the capability of colonization of empty patches, starting from the dispersal of predators living in a few patches which occupy peculiar positions in the given network topology.

3.1 Network topologies and migration

In this section, we analyze the role of migration and compare the six network topologies with respect to four different conditions for the dispersal rules. Namely, we assume that each patch of each topology is initialized with a complete LV model as given in Section 2.2, where the value of the stochastic constant $c_{d_{p_j}}$ for the dispersal of predators, in each patch $p_i \in N_\Delta^\nu$, can assume one of the following values:

1. $c_{d_{p_j}}$=1, for each $p_j \in Adj(p_i)^\nu$;
2. $c_{d_{p_j}}$=10, for each $p_j \in Adj(p_i)^\nu$;
3. $c_{d_{p_j}}$=20, for each $p_j \in Adj(p_i)^\nu$;
4. $c_{d_{p_j}}=\frac{10}{deg(p_i)}$, for each $p_j \in Adj(p_i)^\nu$.

By considering the first condition as reference, the power of dispersal in the second (third) condition is ten-fold (twenty-fold) the first one, irrespective of the position that patch p_i occupies in the considered network. In other terms, the flux of dispersal from each patch, in the first three conditions, results amplified by the number of connections that each patch has with respect to the other patches in the network. On the contrary, the fourth condition corresponds to the situation when, for each patch $p_j \in Adj(p_i)^\nu$, the sum of the values of constants of dispersal rules in p_i is always equal to 10, but the rate of dispersal along each edge from p_i to p_j depends on the degree of p_i. For instance, in the network topology *a* (Fig. 1), the value of $c_{d_{p_j}}$ in patches p_0 and p_5 is equal to 10, while the value of $c_{d_{p_j}}$ in patches p_1, ..., p_4 is equal to 5; in the network topology *c* (Fig. 1), the value of $c_{d_{p_j}}$ in patch p_0 is equal to 2, while the value of $c_{d_{p_j}}$ in all other patches is equal to 10, and so on. So doing, we can weigh the dispersal of predators according to the position of each patch in the network, and simulate a situation where the flux of dispersal from each patch towards its adjacent patches is uniform throughout the whole network.

For space limits, in Fig. 3 we present the phase spaces of all network topologies, obtained from simulations of the fourth condition only. For each network, in particular, we show the phase space of the local dynamics of each patch. The graphics show that, in the case of the chain graph (phase space (a)), the patches having different degrees are characterized by different dynamics: in fact, patches p_0 and p_5 show a different behavior with respect to the other patches. In addition to the role of patch degree, we can see that also the position of patches in the graph plays a central role: despite the fact that patches p_1, p_2, p_3 and p_4 have all the same degree, the dynamics inside p_1 and p_4 differs from that of patches p_2 and p_3. This is due to the different power of dispersal rules of their two neighbors, namely $c_{d_{p_j}} = 10$ in patches p_0, p_5, while $c_{d_{p_j}} = 5$ in patches p_2, p_3, which cause a larger flux of predators dispersal towards patches p_1 and p_4. The global effect is the presence of three different dynamics (one in p_0, p_5, another one in p_1, p_4, and a third one in p_2, p_3), all of which are characterized by oscillations in X and Y with no regular amplitudes (compare these phase spaces with the standard LV phase space in the single patch model

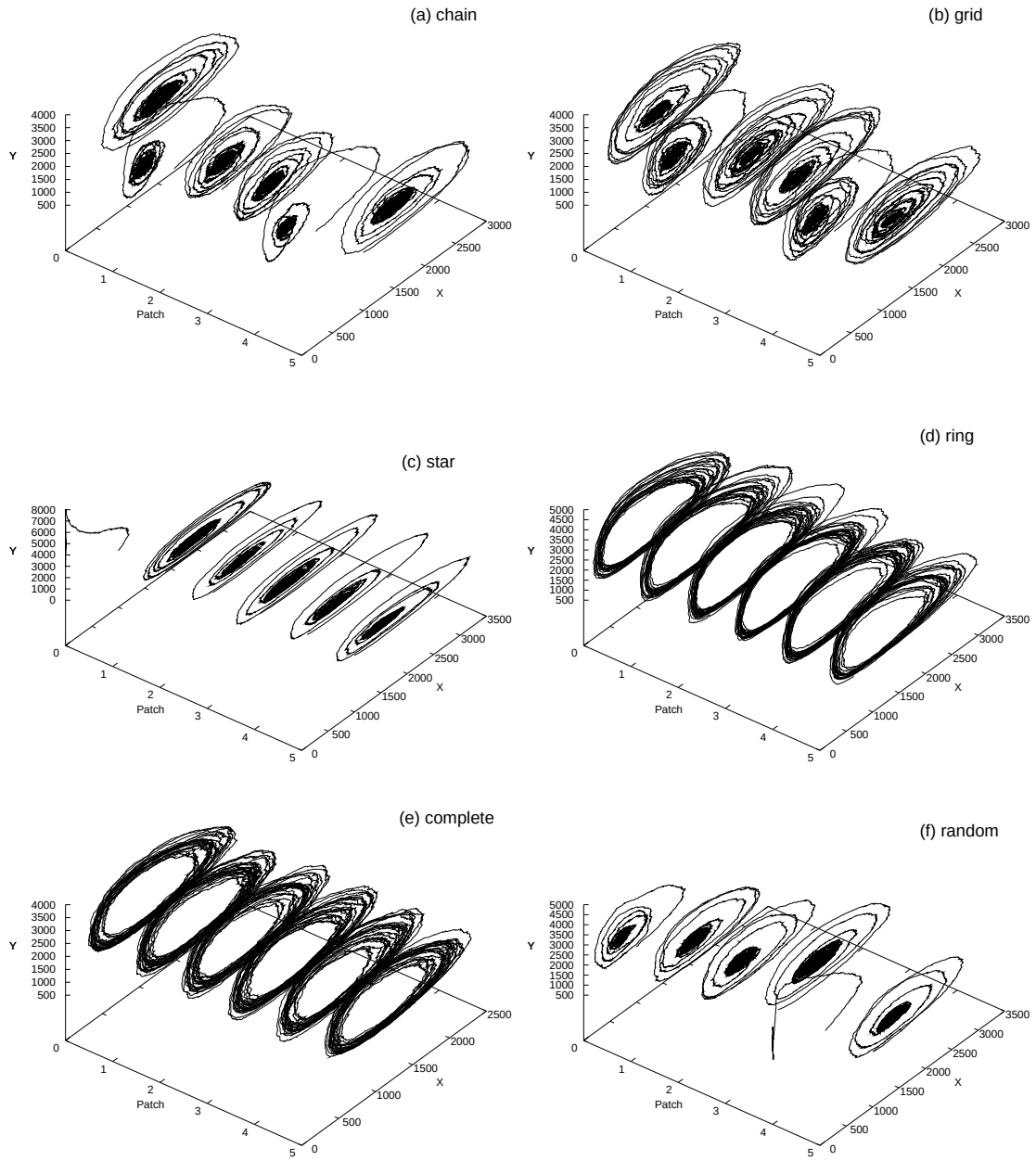


Figure 3: The power of migration: LV dynamics in the phase space of each network topology.

given in Fig. 2, right side, and also with the phase spaces in Fig. 3, graphics (d) and (e)). Furthermore, we can evidence that these oscillations are characterized by an initial wider amplitude, which is reduced during time.

Similarly, the dynamics of the patches in the grid graph (phase space (b)) is influenced only by the number of edges; in this phase space, we can identify two different types of dynamics: one for the patches with three edges (p_1, p_4) and another one for those with two connections.

In the star graph (phase space (c)), the LV dynamics endures in all patches apart from p_0, where the number of preys X collapses to an attractor in zero, and no oscillations according to the LV dynamics in both X and Y can be established. In this patch, the number of predators fluctuates in a certain range, because of their dispersal from/to the other patches. Basically, in this condition patch p_0, that represents the center of the star, becomes a local area of the habitat where only dispersal occurs.

The simulations for the ring and complete graphs (phase spaces (d), (e)) show very similar results: in both cases, all patches in each graph have the same degree (two in the first configuration and five in the second one), leading to regular oscillations in X and Y with almost constant amplitude.

The results concerning the last configuration, the random graph (phase space (f)), show a combination of the effects described above. In particular, the dynamics of the patches differ each other depending on the degree of the patches themselves; moreover, in p_4, which is characterized by the highest degree, the high number of incoming predators (migrating from the four adjacent patches) leads to the extinction of preys (similarly to what happens in patch p_0 of the star graph).

We also tested, for each network topology, the other three conditions listed above. In these cases, the results have shown that the amplification of the power of dispersal with respect to the patch degree gives rise to a balance between the incoming and migrating individuals, which leads to comparable LV dynamics for all networks, with regular oscillations inside each patch (data not shown).

3.2 Network topologies and colonization

In this section, we compare the six network topologies with respect to the capability of colonizing the empty patches that each network contains, starting from the patches that contain a complete LV model and that occupy a peculiar position in that network . We recall that in this work we are considering only the migration of predators, hence the empty patches are hereby assumed to contain no predators but only an initial amount of preys. In each network v, the set of patches initialized with the complete LV model will be denoted as p_{LV}^{v}. To test the feature of colonization, we consider four different initial conditions, hereby denoted as ICk, k=1,...,4, where Y_0=0 and:

1. IC1 is characterized by $c_{d_{p_j}}$=1 and X_0=10;
2. IC2 is characterized by $c_{d_{p_j}}$=1 and X_0=100;
3. IC3 is characterized by $c_{d_{p_j}}$=10 and X_0=10;
4. IC4 is characterized by $c_{d_{p_j}}$=10 and X_0=100.

In each given network, all empty patches are initialized with the same chosen condition ICk, besides the patches in the set p_{LV}^{v} that are initialized with a standard LV model, having the communication constant $c_{d_{p_j}}$ equal to the one given in the chosen ICk, and all other parameters as given in Section 2.2.

With this type of analysis, we expect to determine which features of the network topologies are more relevant with respect to the colonization of empty patches, under a given initial condition. All conditions have been tested for each network and, for each fixed initial condition, different sets of p_{LV}^{v} have been considered. In the following, for space limits, we present only some results of these simulations, and

briefly discuss the results obtained in the other analyzed conditions. In each of the following graph, preys (X) are represented with solid lines, while predators (Y) are represented with dashed lines.

We start by considering the network $v = a$, that is, the chain graph. In this case, we present the results obtained in all the initial conditions IC1, IC2, IC3, IC4, considering three sets of LV patches, namely p^a_{LV}=$\{p_0, p_5\}$, p^a_{LV}=$\{p_2\}$ and p^a_{LV}=$\{p_0\}$. In the first case (p^a_{LV}=$\{p_0, p_5\}$, shown in Fig. 4) we can see that, when the power of dispersal is low (IC1, IC2), the time required by the predators to reach the patches p_2 and p_3, which are at the highest distance from p_0 and p_5, allows an initial uncontrolled growth of the preys in p_2 and p_3, which subsequently undergo extinction as soon as the predators enter the patch. Such "delay" in the local establishment of a population of predators is the effect that prevent the formation of the LV dynamics; this effect, as shown hereafter, is a common aspect of all network topologies. Concerning the chain network, this is more evident in condition IC2, where the initial amount of preys inside the empty patches is higher than IC1: in this case, the LV dynamics can be established only in four of the six patches. On the other hand, with the initial conditions IC3 and IC4, the power of dispersal is sufficient to colonize all of the patches, irrespectively of the numbers of preys that are initially present in the empty patches and of the position of the LV complete patch. Similar results for the chain network have been obtained in the second analyzed case (p^a_{LV}=$\{p_2\}$, shown in Fig. 5) and in the third case (p^a_{LV}=$\{p_0\}$, data not shown).

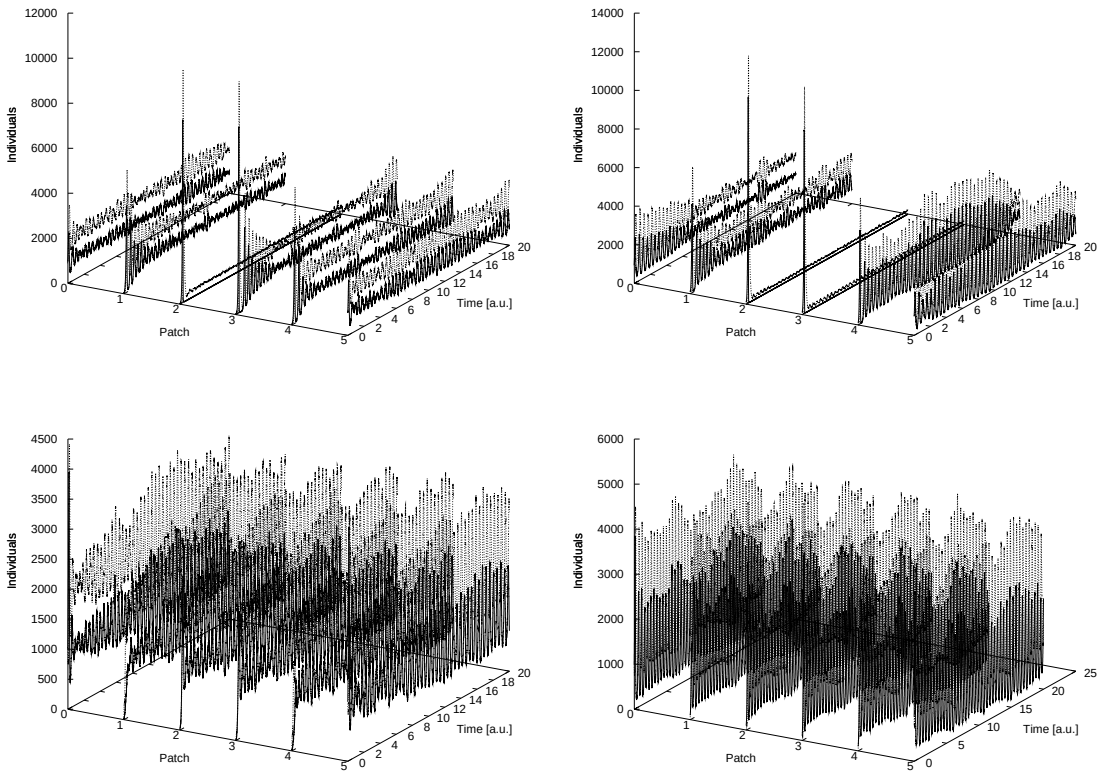


Figure 4: Colonization in the chain topology, with p^a_{LV}=$\{p_0, p_5\}$ and initial conditions IC1 (top left), IC2 (top right), IC3 (bottom left), IC4 (bottom right).

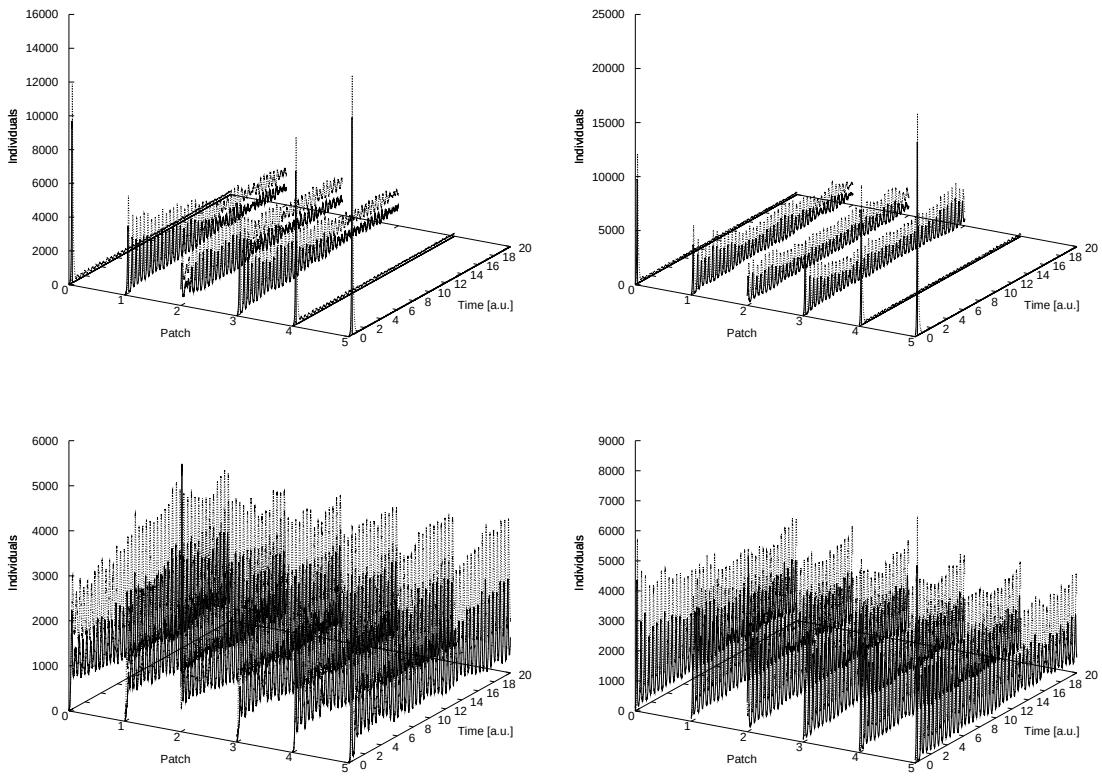


Figure 5: Colonization in the chain topology, with p^a_{LV}=$\{p_2\}$ and initial conditions IC1 (top left), IC2 (top right), IC3 (bottom left), IC4 (bottom right).

For the network topology $v = b$, that is, the grid graph, we show the results obtained in the cases IC1, when p^b_{LV}=$\{p_0\}$ (Fig. 6, left side) and p^b_{LV}=$\{p_1\}$ (Fig. 6, right side). According to the position of the LV complete patches in this network topology, we can see that, in the first case, the predators are capable to colonize patches p_1 and p_3, that are directly connected to p_0, and patch p_4, that is directly connected to both p_1 and p_3. However, patches p_2 and p_5 cannot be colonized. In the second case, the higher degree of the LV complete patch p_1, allows the colonization of all patches. With the initial condition IC2 (data not shown), in the other tested cases p^b_{LV}=$\{p_0\}$ and p^b_{LV}=$\{p_1\}$, only the patches directly connected to p_0 and p_1, respectively, are colonized by the predators.

For the network topology $v = c$, that is, the star graph, we show the results obtained in the cases IC1, when p^c_{LV}=$\{p_1\}$ (Fig. 7, left side) and p^c_{LV}=$\{p_1, p_3\}$ (Fig. 7, right side). According to the position of the LV complete patches in this network topology, we can see that, in the first case, no patches are colonized because of the high degree of p_0 (which is the only patch connected to p_1) that spreads the predators over the other patches, thus preventing the formation of the LV dynamics. In the second case, the combined effect of migration from p_1 and p_3 allows the colonization of patch p_0, which is directly connected with both of them. We then performed other simulations starting with conditions IC3 and IC4: in these cases, the higher value of $c_{d_{p_j}}$ allows the colonization of every patch (except from patch p_0) independently from the initial position of the LV complete patch (data not shown). On the contrary, when

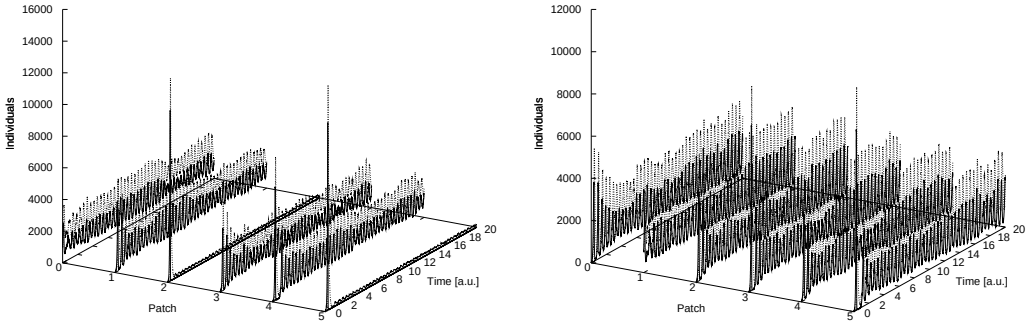


Figure 6: Colonization in the grid topology, with initial condition IC1 and p^b_{LV}=$\{p_0\}$ (left), p^b_{LV}=$\{p_1\}$ (right).

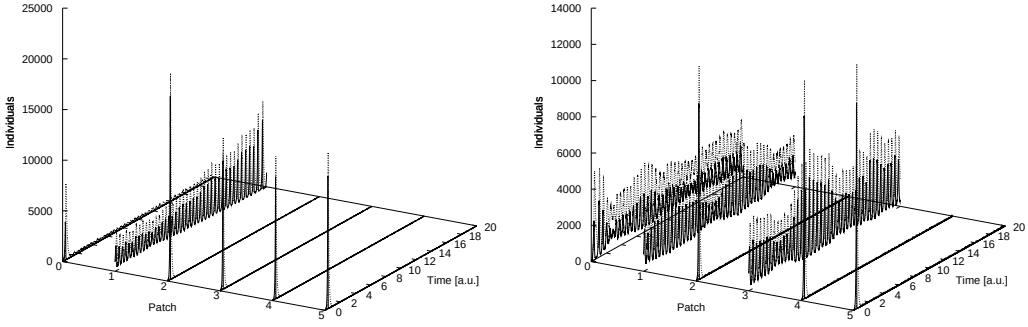


Figure 7: Colonization in the star topology, with initial condition IC1 and p^c_{LV}=$\{p_1\}$ (left), p^c_{LV}=$\{p_1, p_3\}$ (right).

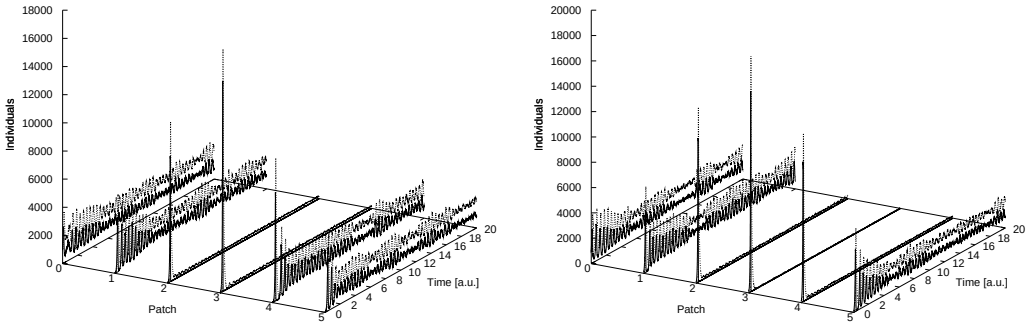


Figure 8: Colonization in the ring topology, with p^d_{LV}=$\{p_0\}$ and initial condition IC1 (left) and IC2 (right).

we assume p^c_{LV}=$\{p_0\}$, that is, the center of the star, then all patches are fully colonized, independently from the considered initial condition.

For the network topology $v = d$, that is, the circular graph, we show the results obtained in the cases IC1 and IC2, when p^d_{LV}=$\{p_0\}$ (Fig. 8, left and right sides, respectively). Starting with the initial condition IC2, the predators are capable of colonizing only the patches directly connected to the LV complete patch p_0, while in the case IC1, also patch p_4 (being at distance 2 from the LV complete patch) is colonized. These results highlight, in particular, another aspect that was more marginal in the other simulations: the stochastic nature of the communication process and of the growth of preys, which leads to the extinction of preys in patch p_2, while in patch p_4 it drives the local behavior to an oscillatory dynamics.

For the network topology $v = e$, that is, the complete graph, we show the results obtained in the cases IC1, when p^e_{LV}=$\{p_0\}$ (Fig. 9, left side) and p^e_{LV}=$\{p_0, p_3\}$ (Fig. 9, right side). While in the second case – where the LV dynamics is initially placed in two patches – the predators can colonize all patches, in the first case the colonization of all empty patches fails. Once more, this is an effect of the stochastic noise combined with the low amounts of predators, which is in turn caused by the fact that the higher the number of adjacent patches, the lower the number of predators that persist inside each patch. In all other simulations performed with initial conditions IC3 and IC4, all patches have always been colonized, as the higher values of dispersal rules assure a more uniform spread of predators throughout the network, and thus flattens the influence of migration delay (data not shown).

For the network topology $v = f$, that is, the random graph, we show the results obtained in the cases IC1, when p^f_{LV}=$\{p_0\}$ (Fig. 10, left side) and p^f_{LV}=$\{p_2\}$ (Fig. 10, right side). According to the position of the LV complete patches in this network topology, we can see that, in the first case, all patches are colonized by predators (similar results are obtained by placing the LV complete model in patch p_4 – data not shown). In the second case, patch p_5 is not colonized because there is only one path of length 2 which connects it to the initial complete LV patch p_2; the same holds for patch p_3, which has distance from p_2 equal to 3. For similar reasons, considering the case of initial condition IC1, with the LV complete model in patch p_3, the only patch that is not colonized by predators is p_2 (data not shown). In all the simulations performed with the initial condition IC2, some of the patches have not been colonized because of the high amount of preys initially occurring in the patches. On the other hand, with the initial conditions IC3, IC4, the power of dispersal allows the colonization of all patches (data not shown).

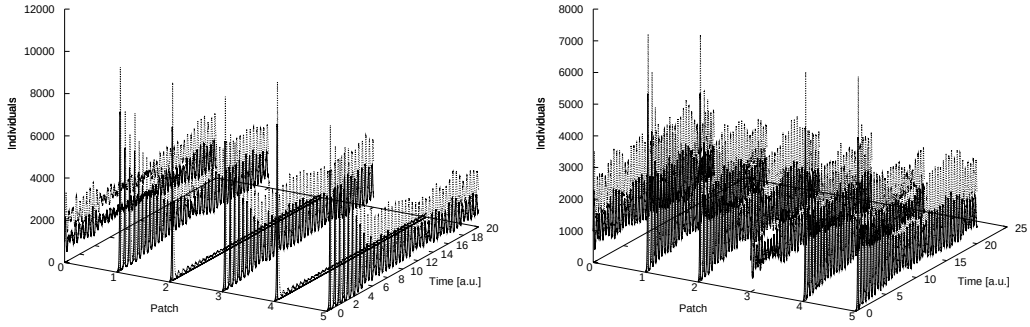


Figure 9: Colonization in the complete topology, with initial condition IC1 and p^e_{LV}=$\{p_0\}$ (left), p^e_{LV}=$\{p_0, p_3\}$ (right).

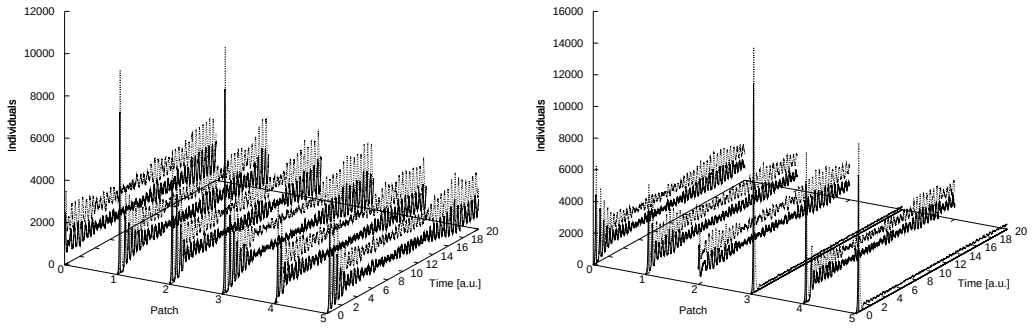


Figure 10: Colonization in the random topology, with initial condition IC1 and $p^f_{LV}=\{p_0\}$ (left), $p^f_{LV}=\{p_2\}$ (right).

4 Discussion

The fragmented habitats of real metapopulations are usually characterized by complex network topologies. In this paper, we have analyzed six small topologies that can be considered representative of local areas in a structured habitat, and we have investigated the influence that the degree and the position of each patch in the topology can have on the migration of individuals, as well as on the capability of colonizing empty patches. Our analysis suggests that, with respect to the power of migration (Section 3.1), we can identify different behaviours that depend on two characteristics of the topology: on a first level, the local behaviour inside each patch is influenced by its degree. This is especially evident if we compare the network topology described by the circular or complete graphs, with the topology described by the star graph: while in the first case (where all nodes have the same degree) all patches are characterized by a similar (regular) oscillating dynamics, in the second case the most critical node is the center of the star (which has a much higher degree than all other nodes in the same graph). In the latter case, this patch is likely to undergo a local modification of its initial dynamics, due to a more higher incoming migration of individuals from all other adjacent patches. On a second level, assuming in this case that the degree of nodes is equal, then also the position of each patch in the topology matters: for instance, we have seen that in the network topology described by the chain graph – where all nodes, besides the ones at the extremes of the chain, have the same degree – the local dynamics is also influenced by the dynamics of the adjacent patches in the graph. Therefore, in hypothetical habitats where there exist many patches connected in a linear way, our results suggest that the length of the chain might have a negative role in the establishment and in the maintenance of local dynamics.

Considering the feature of colonization (Section 3.2), we have evidenced that, in most network topologies, the lack of colonization can be due to the delay of migrating predators with respect to the (uncontrolled) local growth of prey, which then leads to the extinction of preys and the prevention of the LV dynamics. To effectively measure how strong is the power of the delay, it would be interesting to understand whether the local growth of preys can be controlled by inducing their death and thus potentially allowing the establishment of oscillations. Besides this aspect deserving further investigations, our analysis have evidenced that the colonization of empty patches occurs more easily in those patches that are adjacent to the patch(es) initialized with the LV complete model. Once more, this highlights the relevance of the position of the patch(es) where standard oscillations in preys and predators are already

settled at the beginning of the simulation. Indeed, the power of colonization is stronger in the circular and complete networks – where the position of the LV complete patch is irrelevant (as the spread of migrating individuals throughout the network results uniform), and it is weaker in the star network – where the position of the LV complete patch is of primary importance (as the spread of migrating individuals throughout the network strongly depends on whether the patch is placed at the center or at the tips of the star).

In addition to the investigations that we have presented in this work, further types of analysis that we plan to perform on metapopulation systems concern, for instance, the study of the aspects considered in this paper (migration, colonization, network topologies, etc.) by assuming other local and global dynamics, e.g., the population growth according to the logistic function. Moreover, an interesting issue that might be investigated is the synchronization of local population dynamics (e.g. by considering the establishment and decay of oscillations in preys and predators) during migration through a given network topology, or in the process of colonization.

Concerning the use of graphs, other relevant questions regard the analysis of the dynamics with respect to graph properties, such as different measures of habitat connectivity (centrality indexes) [13, 26]. In this context, for example, the star graph can resemble the notion of hub (a node with high degree) in a typical scale-free network, a structure that is known to be robust to random disturbances but highly vulnerable to deliberate attacks on the hubs [32, 3].

Another topic of interest concerns the fact that various populations can coexist in a common habitat, but have distinct (inter)species dynamics or different dispersal capabilities in that habitat [9]. In cases like this, it would be interesting to construct and analyze different metapopulation models, one for each target species, according to both the patch-to-patch connections and to the specific population dynamics. By comparing and intersecting the results obtained on the distinct network topologies of the common habitat derived in this way, it would be possible to determine the locations of the habitat that are most important for each species, and thus aid the design of natural reserve systems where we can have the most appropriate solution for all species in terms of the maximal improvement of dispersal (reduction of species isolation) and the minimal spread of disturbances (diseases, pathogens, invasive species, etc.) [36].

We believe that our modeling approach opens interesting perspectives and can represent an useful tool for the investigation of a wide range of properties in metapopulation systems. We expect that applications of this model to real cases – characterized by complex habitat networks (where each patch possesses its own features of quality, occupancy, connectivity) and different population dynamics – will aid in the achievement of important results and new perspective in Ecology.

References

[1] T. Aittokallio & B. Schwikowski (2006): *Graph-based methods for analysing networks in cell biology. Briefings in bioinformatics* 7(3), pp. 243–255. Available at `http://dx.doi.org/10.1093/bib/bbl022`.

[2] R. Albert (2005): *Scale-free networks in cell biology*. *Journal of Cell Science* 118(21), pp. 4947–4957. Available at `http://dx.doi.org/10.1242/jcs.02714`.

[3] R. Albert & A. L. Barabási (2002): *Statistical mechanics of complex networks*. *Reviews of Modern Physics* 74(1), pp. 47–97. Available at `http://dx.doi.org/10.1103/RevModPhys.74.47`.

[4] L. Berec (2002): *Techniques of spatially explicit individual-based models: construction, simulation, and mean-field analysis*. *Ecological Modelling* 150(1-2), pp. 55–81. Available at `http://dx.doi.org/10.1016/S0304-3800(01)00463-X`.

[5] D. Besozzi, P. Cazzaniga, G. Mauri & D. Pescini (2010): *BioSimWare: A P Systems–based Simulation Environment for Biological Systems*. Accepted for presentation at CMC11, Jena, Germany, 2010.

[6] D. Besozzi, P. Cazzaniga, D. Pescini & G. Mauri (2007): *Seasonal variance in P system models for metapopulations*. *Progress in Natural Science* 17(4), pp. 392–400. Available at http://www.informaworld.com/smpp/content~db=all~content=a790271810~tab=linking.

[7] D. Besozzi, P. Cazzaniga, D. Pescini & G. Mauri (2008): *Modelling metapopulations with stochastic membrane systems*. *BioSystems* 91(3), pp. 499 – 514. Available at http://www.sciencedirect.com/science/article/B6T2K-4PD4XHR-1/2/0903081b39759345708d5fc97aeee6bc.

[8] D. Besozzi, P. Cazzaniga, D. Pescini & G. Mauri (2009): *Algorithmic Bioprocesses*, chapter A Multivolume Approach to Stochastic Modelling with Membrane Systems, pp. 519–542. Springer Verlag. Available at http://www.springer.com/computer/theoretical+computer+science/book/978-3-540-88868-0.

[9] A. Bunn (2000): *Landscape connectivity: A conservation application of graph theory*. *Journal of Environmental Management* 59(4), pp. 265–278. Available at http://dx.doi.org/10.1006/jema.2000.0373.

[10] Y. Cao, D. T. Gillespie & L. R. Petzold (2006): *Efficient step size selection for the tau-leaping simulation method*. *Journal of Chemical Physics* 124, p. 044109. Available at http://www.ncbi.nlm.nih.gov/pubmed/16460151.

[11] P. Cazzaniga, D. Pescini, D. Besozzi & G. Mauri (2006): *Tau leaping stochastic simulation method in P systems*. In: H. J. Hoogeboom, G. Păun, G. Rozenberg & A. Salomaa, editors: *Proc. of the 7th International Workshop on Membrane Computing*, 4361. LNCS, pp. 298–313. Available at http://www.springerlink.com/content/y055172665v12t2k/?p=0a193a302bda4824b742ec37b8cda9d9&pi=18.

[12] G. Ciobanu, G. Păun & M. J. Pérez-Jiménez, editors (2005): *Applications of Membrane Computing*. Springer-Verlag, Berlin.

[13] S. N. Dorogovtsev & J. F. F. Mendes (2002): *Evolution of networks*. *Advances in Physics* 51(4), pp. 1079–1187. Available at http://www.informaworld.com/smpp/content~db=all~content=a713801291~tab=linking.

[14] J. B. Dunning Jr., D. J. Stewart, B. J. Danielson, B. R. Noon, T. L. Root, R. H. Lamberson & E.E. Stevens (1995): *Spatially Explicit Population Models: Current Forms and Future Uses*. *Ecological Applications* 5(1), pp. 3–11. Available at http://www.esajournals.org/doi/abs/10.2307/1942045.

[15] A. Fall, M. Fortin, M. Manseau & D. O'brien (2007): *Spatial Graphs: Principles and Applications for Habitat Connectivity*. *Ecosystems* 10, pp. 448–461. Available at http://dx.doi.org/10.1007/s10021-007-9038-7.

[16] D. T. Gillespie (1977): *Exact stochastic simulation of coupled chemical reactions*. *Journal of Physical Chemistry* 81(25), pp. 2340–2361. Available at http://dx.doi.org/10.1021/j100540a008.

[17] I. Hanski (1998): *Metapopulation dynamics*. *Nature* 396, pp. 41–49. Available at http://www.nature.com/nature/journal/v396/n6706/abs/396041a0.html.

[18] A. Hastings & S. Harrison (1994): *Metapopulation dynamics and genetics*. *Annual Review of Ecology and Systematics* 25, pp. 167–188. Available at http://arjournals.annualreviews.org/doi/abs/10.1146/annurev.es.25.110194.001123.

[19] A. Hastings & C. L. Wolin (1989): *Within-patch dynamics in a metapopulation*. *Ecology* 70(5), pp. 1261–1266. Available at http://www.esajournals.org/doi/abs/10.2307/1938184.

[20] V. A. A. Jansen (2001): *The dynamics of two diffusively coupled predator-prey populations*. *Theoretical Population Biology* 59, pp. 119–131. Available at http://dx.doi.org/10.1006/tpbi.2000.1506.

[21] V. A. A. Jansen & A. L. Lloyd (2000): *Local stability analysis of spatially homogeneous solutions of multi-patch systems*. *Journal of Mathematical Biology* 41, pp. 232–252. Available at http://www.springerlink.com/content/px0an76xh7551jew/.

[22] R. Levins (1969): *Some demographic and genetic consequences of environmental heterogeneity for biological control*. *Bulletin of the Entomological Society of America* 71, pp. 237–240.

[23] E. S. Minor & D. L. Urban (2008): *A Graph-Theory Framework for Evaluating Landscape Connectivity and Conservation Planning*. *Conservation Biology* 22(2), pp. 297–307. Available at http://dx.doi.org/10.1111/j.1523-1739.2007.00871.x.

[24] A. Moilanen (2004): *SPOMSIM: software for stochastic patch occupancy models of metapopulation dynamics*. *Ecological Modelling* 179, pp. 533–550. Available at http://dx.doi.org/10.1016/j.ecolmodel.2004.04.019.

[25] J. D. Murray (2002): *Mathematical Biology. I: An introduction*. Springer-Verlag, New York.

[26] M. E. J. Newman (2003): *The Structure and Function of Complex Networks*. *SIAM Review* 45(2), pp. 167–256. Available at http://scitation.aip.org/getabs/servlet/GetabsServlet?prog=normal&id=SIREAD000045000002000167000001&idtype=cvips&gifs=yes.

[27] G. Păun (2000): *Computing with membranes*. *Journal of Computer and System Sciences* 61(1), pp. 108–143. Available at http://dx.doi.org/10.1006/jcss.1999.1693.

[28] G. Păun (2002): *Membrane Computing. An Introduction*. Springer-Verlag, Berlin.

[29] G. Păun, G. Rozenberg & A. Salomaa, editors (2010): *The Oxford Handbook of Membrane Computing*. Oxford University Press.

[30] D. Pescini, D. Besozzi, G. Mauri & C. Zandron (2006): *Dynamical probabilistic P systems*. *International Journal of Foundations of Computer Science* 17(1), pp. 183–204. Available at http://dx.doi.org/10.1142/S0129054106003760.

[31] D. Pescini, D. Besozzi, C. Zandron & G. Mauri (2006): *Analysis and simulation of dynamics in probabilistic P systems*. In: N. Pierce A. Carbone, editor: *Proc. of 11th International Workshop on DNA Computing, DNA11*, 3892. LNCS, London, ON, Canada, pp. 236–247. Available at http://www.springerlink.com/content/7p7653442111r273/?p=832da88ca13a4d75be122f548c3b0df6&pi=18.

[32] S. H. Strogatz (2001): *Exploring complex networks*. *Nature* 410, pp. 268–276. Available at http://tam.cornell.edu/SS_exploring_complex_networks.pdf.

[33] A. D. Taylor (1990): *Metapopulations, dispersal, and predator-prey dynamics: an overview*. *Ecology* 71(2), pp. 429–433. Available at http://www.esajournals.org/doi/abs/10.2307/1940297.

[34] J. M. J. Travis & C. Dytham (1998): *The evolution of dispersal in a metapopulation: a spatially explicit, individual-based model*. *Proceedings of the Royal Society of London. Series B: Biological Sciences* 265(1390), pp. 17–23. Available at http://dx.doi.org/10.1098/rspb.1998.0258.

[35] D. Urban & T. Keitt (2001): *Landscape Connectivity: A Graph-Theoretic Perspective*. *Ecology* 82(5), pp. 1205–1218. Available at http://dx.doi.org/10.2307/2679983.

[36] D. L. Urban, E. S. Minor, E. A. Treml & R. S. Schick (2009): *Graph models of habitat mosaics*. *Ecology Letters* 12, pp. 260–273. Available at http://dx.doi.org/10.1111/j.1461-0248.2008.01271.x.

[37] W. W. Weisser, V. A. A. Jansen & M. P. Hassell (1997): *The effects of a pool of dispersers on host-parasitoid systems*. *Journal of Theoretical Biology* 189, pp. 413–425. Available at http://dx.doi.org/10.1006/jtbi.1997.0529.

Modelling the Dynamics of an *Aedes albopictus* Population

Thomas Anung Basuki Antonio Cerone
International Institute for Software Technology (UNU-IIST)
United Nations University, Macau SAR China
anung@iist.unu.edu antonio@iist.unu.edu

Roberto Barbuti Andrea Maggiolo-Schettini Paolo Milazzo
Dipartimento di Informatica
Università di Pisa, Pisa, Italy
barbuti@di.unipi.it maggiolo@di.unipi.it milazzo@di.unipi.it

Elisabetta Rossi
Dipartimento di Coltivazione e Difesa delle Specie Legnose
Università di Pisa, Pisa, Italy
erossi@agr.unipi.it

We present a methodology for modelling population dynamics with formal means of computer science. This allows unambiguous description of systems and application of analysis tools such as simulators and model checkers. In particular, the dynamics of a population of *Aedes albopictus* (a species of mosquito) and its modelling with the Stochastic Calculus of Looping Sequences (Stochastic CLS) are considered. The use of Stochastic CLS to model population dynamics requires an extension which allows environmental events (such as changes in the temperature and rainfalls) to be taken into account. A simulator for the constructed model is developed via translation into the specification language Maude, and used to compare the dynamics obtained from the model with real data.

1 Introduction

In the last few years many formalisms have been defined to model biological systems at molecular and cellular levels [3, 9, 13, 22, 23]. These formalisms allow unambiguous description of systems and application of analysis tools, such as simulators and model checkers.

Among these formalisms the Calculus of Looping Sequences (CLS) [3] seems to be applicable to other classes of biological systems. CLS is based on term rewriting, in which terms may represent simple biological structures and compartments, and rewrite rules may represent very general events. Moreover, a stochastic extension of CLS has been defined, called Stochastic CLS [2], which allows the dynamics over time of the described system to be studied [6, 7].

In this paper we deal with the problem of modelling population dynamics with formal means of computer science. Many aspects of population dynamics, such as births, deaths and interaction of individuals, can be modelled by using Stochastic CLS. Other aspects related to environmental events, such as changes in climatic conditions, require an extension of the formalism. In this paper we define such an extension and use it to model the dynamics of a population of *Aedes albopictus*.

Aedes albopictus (Skuse), or Asian tiger mosquito, is a species indigenous to the oriental region, but it is now widespread in many countries throughout the world. It is an aggressive mosquito, which causes nuisance and it is well known as an important disease vector [19].

To appear in:
AMCA-POP 2010 – Electronic Proceedings in Theoretical Computer Science (EPTCS)

A simulator for the constructed model is developed via translation into the specification language Maude [12], and used to compare the dynamics obtained from the model with real data.

There are a number of other approaches to the modelling of population dynamics with formal means of computer science [5,10,20]. Barbuti *et al.* [5] extend P systems with features typical of timed automata with the aim of describing periodic environmental events such as changes of seasons. Cardona *et al.* [10] propose a modelling framework based on P systems and apply it to the modelling of the dynamics of some scavenger birds in the Pyrenees. McCaig, Norman and Shankland [20] present a process algebraic approach to the modelling of population dynamics. With respect to these proposals we believe that our approach allows a finer modelling of environmental events. Moreover, thanks to the extensions of the tools already developed for Stochastic CLS, it offers means for accurate analysis of phenomena.

2 Stochastic CLS

Calculi of Looping Sequences (CLS class) is a class of formalisms introduced in Milazzo's PhD thesis [21] for modelling biological systems. The first formalism of the class to be defined, the Full Calculus of Looping Sequences (Full CLS) uses 4 operators: sequencing, parallel composition, looping and containment. The parallel composition operator has the typical semantics as in other formalisms such as the π-Calculus and Brane Calculi. Sequencing is inspired by the sequential structure of several macromolecules such as DNA. The looping operator is always applied together with the containment operator and supports the modelling of membrane-like structures. An important language of the CLS class is Stochastic CLS, which supports the modelling of quantitative aspects of biological systems such as time and probabilities.

We start by introducing the syntax of sequences and terms, the basic building blocks of Stochastic CLS.

Definition 1. *Sequences S and Terms T are defined as follows:*

$$S ::= \varepsilon \mid S \cdot S \mid a \qquad\qquad T ::= S \mid \left(T\right)^{L} \rfloor T \mid T \mid T$$

where ε represents the empty sequence and $a \in \mathscr{E}$. We denote the set of all terms with $\mathscr{T}$, and the set of all sequences with $\mathscr{S}$.

We assume the existence of a possibly infinite set of symbols $\mathscr{E}$. The parallel composition operator $\mid$ is used to model a mixture of elements. The application of the looping and containment operator to two terms T_1 and T_2, denoted by $\left(T_1\right)^{L} \rfloor T_2$, models structure T_2 within a compartment surrounded by structure T_1. Structure T_1 is called the *loop part* and T_2 is called its *content part*.

The behaviour of a biological system is modeled by means of transitions between terms. This is done by applying *rewrite rules*, that are described by two *patterns*, to be instantiated by terms, and a *rate* that defines the frequency with which the rule is applied.

Definition 2. *Let TV be an infinite set of term variables ranged over by $X, Y, Z, \ldots$ Term Patterns TP and Patterns P are defined as follows:*

$$TP ::= S \mid \left(P\right)^{L} \rfloor P \mid TP \mid TP \qquad\qquad P ::= TP \mid TP \mid X$$

where $X \in TV$. We denote with $\mathscr{P}$ the set of all patterns. We denote with $Var(P)$ the set of variables in P.

Definition 3. *An* instantiation *is a partial function $\sigma : TV \to \mathscr{T}$. We denote with Σ the set of all possible instantiations. Given $P \in \mathscr{P}$, we denote with $P\sigma$ the term obtained by replacing all variables $X \in Var(P)$ with $\sigma(X)$.*

Definition 4. *A* rewrite rule *is a triple (P_L, k, P_R), denoted with $P_L \overset{k}{\mapsto} P_R$, where $P_L, P_R \in \mathscr{P}$, $k \in \mathbb{R}$ and such that $Var(P_R) \subseteq Var(P_L)$.*

Definition 5. *A biological system is a pair $(T, \mathscr{R})$, where T is a term representing the initial state of the system and $\mathscr{R}$ is a set of rewrite rules that represent the potential* events *which may occur in the system.*

Interactions between populations in a biological or ecological system occur through some kind of reactions, which may be biochemical reactions at molecular level or changes in organisms' development in ecosystems. To perform in silico analysis of a biological or ecological system, the behaviour of the system must be simulated (*in silico experiment*). The problem of simulating chemically reacting system was stated by Gillespie [17]. We generalise Gillespie's formulation of the problem to any biological or ecological system as follows.

> A volume or environment V contains a mixture of N species $S_1, \ldots, S_N$ which can interreact through M reaction channels $(R_1, \ldots, R_M)$. Given the initial numbers of individuals (molecules or organisms) of each species, what will these population levels be at any later time?

Gillespie consider time evolution of a reacting system as discrete and stochastic. In Gillespie's Stochastic Simulation Algorithm [16], the state of the system is represented by a vector $\mathbf{x} = \mathbf{X}(t) = (X_1(t), \cdots, X_N(t))$, where $X_i(t)$ represents the number of S_i individuals in V at time t. Gillespie assumed that for every reaction channel R_j, there is a constant c_j such that $c_j dt$ is the average probability that a particular combination of reactant individuals in R_j will react accordingly in the next infinitesimal time interval dt. To calculate the probability that a reaction R_j will occur in V in the next infinitesimal time interval $(t, t+dt)$, we must multiply $c_j dt$ by the total number of distinct combinations of individuals in V at time t that are reactants of R_j. Let us denote such number by $h_j(\mathbf{x})$. Gillespie defines the *propensity function* $a_j(\mathbf{x})$ for reaction R_j as the product of $h_j(\mathbf{x})$ and c_j, such that $a_j(\mathbf{x})\ dt$ is the probability that one R_j reaction will occur in the next infinitesimal time interval $[t, t+dt)$.

Gillespie defines a Direct Method to implement his Stochastic Simulation Algorithm [17]. This version of Gillespie's SSA is defined as follows.

Algorithm 1. Let $\{R_1, \ldots, R_M\}$ be a set of rewrite rules, $X_1, \ldots, X_N$ be numbers of N categories of individuals, *maxtime* be the time limit for the duration of the simulation.

Step 0 Initialise simulation time t to 0. Compute propensity a_i for every rewrite rule R_i.

Step 1 Compute the time increment τ.

Step 2 Increase simulation time t by time increment τ.

Step 3 If $t > maxtime$ then stop. Otherwise select the next rule index μ.

Step 4 Execute rule R_μ and update numbers $X_1, \ldots, X_N$ of N categories of individuals and propensities a_i for all rewrite rules R_i affected by the application of R_μ accordingly. Return to **Step 1**.

Gillespie showed in his paper [17] that the time when next reaction occurs (time $t+\tau$ calculated at Step 2 when reaction R_μ selected at Step 3 occurs) is exponentially distributed with parameter $a_0(\mathbf{x}) =$

$\sum_{i=1}^{M} a_i(\mathbf{x})$. Gillespie used a general Monte Carlo method called *inversion method* to compute the exponentially distributed τ and μ from two uniformly distributed random numbers as follows:

$$\tau = \frac{1}{a_0(\mathbf{x})} ln(\frac{1}{r_1}) \tag{1}$$

$$\mu = \textit{the integer for which } \sum_{\nu=1}^{\mu-1} a_\nu(\mathbf{x}) < r_2 a_0(\mathbf{x}) \leq \sum_{\nu=1}^{\mu} a_\nu(\mathbf{x}) \tag{2}$$

where r_1, r_2 are two real values uniformly distributed over interval [0,1] generated by a random number generator.

In previous work Basuki, Cerone and Carvalho [6] extended Algorithm 1 to handle compartment selection. This is useful when we have to simulate a biological system with multi-compartments as is the case for molecular reactions occurring within cells. Such a modified version of the Direct Method is described as follows.

Algorithm 2. Let $\{R_1, \ldots, R_M\}$ be a set of M rewrite rules, $X_1, \ldots, X_N$ be numbers of N categories of individuals, *maxtime* be the time limit for the duration of the simulation.

Step 0 Initialise simulation time t to 0. Compute propensity a_i for every rewrite rule R_i.

Step 1 Compute the time increment τ.

Step 2 Increase simulation time t by time increment τ.

Step 3 If $t > maxtime$ then stop. Otherwise select the next rule index μ and the index θ of the compartment in which rule R_μ will occur.

Step 4 Execute rule R_μ in the compartment with index θ and update numbers $X_1, \ldots, X_N$ of N categories of individuals and propensities a_i for all rewrite rules R_i affected by the application of R_μ accordingly. Return to **Step 1**.

Since reactions are confined within compartments, we need to extend Gillespie's algorithm to choose in which compartment reaction R_μ should occur. Let C be the number of compartments and X_k^i the number of individuals of kind S_k in the i-th compartment. We define $X_k = \sum_{i=1}^{C} X_k^i$.

Let a_j^i be the propensity of reaction R_j occurring inside the i-th compartment. Then a_j^i is defined as the product of c_j by the number h_j^i of distinct combinations of reacting individuals of reaction R_j within the i-th compartment. We define $a_j = \sum_{i=1}^{C} a_j^i$. If t is the current simulaton time, then $t + \tau$ represents the time at which next reaction occurs, with τ exponentially distributed with parameter $a_0 = \sum_{j=1}^{M} a_j$. Time increment τ is calculated as in Alghorithm 1. The index μ of the reaction that occurs at time $t + \tau$ and the index θ of the compartment in which such reaction occurs are calculated as follows:

$$(\mu, \theta) = \textit{the integers for which } \sum_{j=1}^{\mu} \sum_{i=1}^{\theta-1} a_j^i < r_2 a_0 \leq \sum_{j=1}^{\mu} \sum_{i=1}^{\theta} a_j^i \tag{3}$$

where r_2 is a random real number which is uniformly distributed over interval [0,1].

3 Extending Stochastic CLS

The evolution of a system modelled by using Stochastic CLS is entirely characterised by the rewrite rules, which determine the occurrence of events in the system. In this way the set of rewrite rules

predicts all events that may occur. This works well for biological systems, where all events are caused by biochemical reactions which are governed by precise laws.

In ecological systems, instead, we need to deal with environmental events, whose cause is often unknown or depends on a very complex combination of factors, which are external to the system itself. For example the dynamics of a population of a given species depends not only on the interaction with other species within the same ecosystem, such as predators, preys and competitors, but also on the occurrence of environmental events such as climatic events (i.e. variation of temperature and rainfalls) and events related to habitats (i.e. tree clearing, desiccation of a water container, pollution, hunting and human settlement). Therefore, we assume the existence of a list of external events, with information about the time when these events occur. The occurrence of an external event may modify some environmental information which affects the ecosystem evolution, such as temperature, volume of water, desiccation, level of pollution. Moreover, the list of external events may change dynamically. For instance, an initial desiccation event for a water container will be removed from the list after the occurrence of a rainfall event, and will be replaced with a new desiccation event with a later desiccation time.

We extend Stochastic CLS by introducing a list E of external events. The events in list E are sorted in increasing order based on the time they are scheduled to occur. When an external event occurs it updates information in the system state. The updated information may be then used by rewrite rules.

In general, the environment is organised as several nested compartments, each associated with specific environmental information, which is relevant to the specific ecosystem we are modelling and may be modified by the occurrence of external events. We further extend Stochastic CLS by attaching environmental information to the looping operator. This is similar to the extension of Stochastic CLS to Spatial CLS [4], in which spatial information is added to the looping operator and sequence.

Definition 6. Terms T, Nonparallel Terms C, Sequences S *and* Environmental Information I *are given by the following grammar:*

$$\begin{array}{llll} T & ::= C^n \mid T \mid T & C & ::= S \mid \left(T\right)_I^L \rfloor T \\ S & ::= \varepsilon \mid a \mid S \cdot S & I & ::= \lambda \mid a : V \mid I\, I \end{array}$$

where a is a generic element of $\mathscr{E}$, ε represents the empty sequence, λ represents the empty environmental information, V represents the information value and $n \in \mathbb{N}$. We denote with $\mathscr{T}$, $\mathscr{C}$, $\mathscr{S}$ and $\mathscr{I}$ the infinite set of terms, nonparallel terms, sequences and environmental information, respectively.

Note that in Definition 6 we have introduced a notation to group identical nonparallel terms together. For instance, C^5 is equivalent to $C \mid C \mid C \mid C \mid C$.

Events in event list E update environmental information in the system state. Every element of E is a triple (N_E, V_E, t_E), where N_E is the name of the event, V_E is a value that will be used to update the information field related to this event and t_E is the time at which this event is scheduled to occur. We assume the existence of an event handler algorithm which will handle the update of the term representing the system state due to the occurrence of an event (N_E, V_E, t_E).

To run the simulation using the extended version of Stochastic CLS, we need to modify the version of Gillespie's Direct Method [17] defined in Algorithm 2. In modelling population dynamics we have to deal with the same problem we encounter at cellular level: reactions occur in compartments. Therefore, we extend the Direct Method for multi-compartments described in Algorithm 2 with additional steps to handle the execution of external events from the event list. After computing the time of the next rewrite rule, we need to compare this time with the time of the first event in the list, and execute the event with earlier occurrence time. We propose the modified version of Direct Method as follows.

Algorithm 3. Let $\{R_1,\ldots,R_M\}$ be a set of rewrite rules, $X_1,\ldots,X_N$ be numbers of N categories of organisms, E be a list of events and *maxtime* be the time limit for the duration of the simulation.

Step 0 Initialise simulation time t to 0. Compute propensity a_i for every rewrite rule R_i.

Step 1 Compute the time increment τ. Let (N_E,V_E,t_E) be the first event from E with N_E the name of the event, V_E the value needed to update the system state and t_E the occurrence time of the event.

Step 2 If $t_E < t+\tau$ then set t to t_E and then call the event handler algorithm to handle the new event and return to **Step 1**. Otherwise increase simulation time t by time increment τ.

Step 3 If $t >$ *maxtime* then stop. Otherwise select the next rule index μ and the index θ of the compartment in which rule R_μ will occur.

Step 4 Execute rule R_μ in the compartment with index θ and update numbers $X_1,\ldots,X_N$ of N categories of organisms and propensities a_i for all rewrite rules R_i affected by the application of R_μ accordingly. Return to **Step 1**.

The event handler algorithm is specific to the external events occurring in the system. This algorithm updates system state and list of external events and recomputes the propensities that have been affected by the change of system state.

The simulation is affected by the propensity of every rewrite rule. Propensity depends on the number of individuals in the population and the rule rate constant. External factors from the environment affect propensity values. In general, we cannot associate a rule rate constant with each rewrite rule, because the value of the rule rate depends on environmental information, which changes according to external events. Since environmental information is incorporated in terms, to model the rule rate we associate with that rule a function f ranging over terms.

Definition 7. *Let TV be an infinite set of term variables ranged over by $X,Y,Z,\ldots$, IV be an infinite set of information variables ranged over by $x,y,z,\ldots$ and NV be an infinite set of natural number variables ranged over by $q,r,s,\ldots$ Information Patterns IP, Term Patterns TP and Patterns P are defined as follows:*

$$IP ::= I \mid I \,|\, x \quad CP ::= S \mid \left(T\right)^{L}_{IP} \rfloor P \quad TP ::= CP^{q} \mid TP \,|\, TP \quad P ::= TP \mid TP \,|\, X$$

where $X \in TV$ $x \in IV$ and $q \in NV$. We denote with $\mathscr{P}$ the set of all patterns. We denote with $Var(P)$ the set of variables in P.

Definition 8. *The* instantiation *is a partial function $\sigma : TV \cup IV \cup NV \to \mathscr{T} \cup \mathscr{I} \cup \mathbb{N}$ such that $\sigma(TV) \subseteq \mathscr{T}$, $\sigma(IV) \subseteq \mathscr{I}$ and $\sigma(NV) \subseteq \mathbb{N}$. We denote with Σ the set of all possible term instantiations. Given $P \in \mathscr{P}$, we denote with $P\sigma$ the term obtained by replacing all variables $X \in Var(P)$ with $\sigma(X)$.*

Definition 9. *A* rewrite rule *is a 4-tuple (f_c,P_L,P_R,f), usually written as*

$$[f_c]\; P_L \overset{f}{\longmapsto} P_R$$

where $f_c : \Sigma \to \{true, false\}$, $Var(P_R) \subseteq Var(P_L)$, and $f : T \to \mathbb{R}^{\geq 0}$.

The left pattern matches a portion of the term that models the system by using an instantiation function $\sigma \in \Sigma$. This portion of the system must also satisfy the constraint function f_c to enable the rule to be applied. A rate function f associated with the rule will be applied to $P_L\sigma$. After the rule is applied, $P_L\sigma$ is substituted by $P_R\sigma$.

Definition 10. *An ecosystem is a triple $(T,\mathscr{R},E)$, where T is a term representing the initial state of the system, $\mathscr{R}$ is a set of rewrite rules that represent the potential* internal events *which may occur in the system, and E is a list of* external events.

4 Modelling the Population Dynamics of *Aedes albopictus*

We use the formalism developed in the previous section to model *Aedes albopictus* population dynamics.

4.1 Modelling Information about a Mosquito

We model each mosquito by using a looping and containment operator with a parallel composition of symbols representing information about the mosquito in the content part and a symbol a in the loop part. The information in the content part consists of the current development phase of the mosquito and an indicator of whether the mosquito has sucked blood or not. In our approach we only model females, assuming equal numbers of males and females in the population. In this way we do not need to model gender in the information of a mosquito.

Aedes albopictus goes through 4 development phases in its life cycle: egg, larva, pupa, and adult. The larval stage is divided into 4 *instars* [8]. The adult stage is divided into 8 *gonotrophic cycles* [14]. A gonotrophic cycle is a cycle in the adult life which consists of three phases called Beklemishev phases [18]: search for a host and blood-feeding, digestion of the blood and egg maturation, search for a suitable oviposition site and oviposition. We use symbols Egg, $Larva$, $Pupa$ and $Adult$ to denote the 4 development phases. Since larva phase is divided into 4 instars, we use symbols 1, ..., 4 to represent instars. Analogously, we use symbols 1, ..., 8 to represent gonotrophic cycles.

An adult mosquito needs blood before ovipositing eggs. We model this phenomenon by adding symbol $Blood$ to the content part of the looping and containment operator defining the mosquito to represent an adult mosquito that has sucked blood. The number of $Blood$ symbols in the content part indicates how many times that mosquito has sucked blood. For instance we represent 3 adult mosquitoes at gonotrophic cycle 1 that have sucked blood twice and 5 larvae at instar 1 phase using the following term:
$((a)^L_\lambda \rfloor (Adult \mid 1 \mid Blood^2))^3 \mid ((a)^L_\lambda \rfloor (Larva \mid 1))^5$.

4.2 Modelling Compartments

In Stochastic CLS compartments are modelled by using looping-containment operators. As we have seen in Definition 6 compartments play an important role in our approach, because environmental information is attached to them.

Aedes albopictus, like other species of mosquitoes, spends its immature stages in water. In particular, *Aedes albopictus* prefers to lay eggs outdoors [11]. Its natural breeding places are small, restricted, and shaded water collections surrounded by vegetation. In urban areas, many man-made containers such as tin cans, pots, tires and bottles are usually stored outdoors and collect rainfall water, and thus become ideal breeding places [15]. Adult *Aedes albopictus* needs to suck blood before ovipositing. However, *Aedes albopictus* only sucks blood during daytime. Moreover, during immature stages, the duration of the stage is affected by temperature while death rate is affected by population density. We can therefore define an outermost environmental compartment (we call it *environment*), with the value of average daily temperature and daytime/nighttime as relevant environmental information, inside which there are several other compartments where immature mosquitoes live (we call them *containers*). Population density in one container is defined as the number of individuals inside the container divided by the water volume in the container. Therefore, relevant environmental information for a container includes not only temperature but also water volume and desiccation time. Typical external events are sunrise and sunset, which determine switching between daytime and nighttime, temperature changes, which

affect desiccation time by reducing the volume of water inside the containers and, as a result, increases population density, and rainfalls, which increase the level of water in containers where mosquitoes live, so decreasing the population density.

Each kind of compartment has different environmental information. The outermost compartment is the environment, to which we need to attach information about current temperature and daylight. Therefore, environment is modelled by a term

$$(En)^{L}_{Temp:V_{Temp}\ Daylight:V_{Daylight}} \rfloor (T)$$

where V_{Temp} is a real number representing the current temperature, $V_{Daylight}$ is a boolean representing whether it is daylight time and T is the term representing the population of *Aedes albopictus*.

Immature *Aedes albopictus* live in small containers, modelled by using looping-containment operators with symbol C inside the loop part. For each container we attach the following environmental information:

- an index to identify each container, to be used for container selection by Algorithm 3;
- the volume of water inside the container, to be used to compute population density;
- container temperature;
- three population density thresholds, to be used in the computation of death rates of mosquitoes living in the container;
- container desiccation time.

If N_C is the number of containers in our model, we use natural numbers in $[1, N_C]$ to identify containers. We model the volume of water in an abstract way by classifying containers as $full$, $half-full$ and *empty*. Population density thresholds, which are used to classify the population density in a container and set the death rates accordingly will be defined in Section 4.3. Desiccation, or decrease of water, in a container is a process that depends on the characteristic of the container. A desiccation time, which measures how many days are needed to reduce the volume of water in a container, is assigned to each container. Container desiccation time will be defined in Section 4.4. As an example, term

$$\begin{aligned} \mathit{Containers} \ ::= \ & (C)^{L}_{ind:1\ Temp:10\ Vol:empty\ \phi_1:100\ \phi_2:250\ \phi_3:300\ DTime:2.0} \rfloor \varepsilon \mid \\ & (C)^{L}_{ind:2\ Temp:10\ Vol:full\ \phi_1:50\ \phi_2:125\ \phi_3:150\ DTime:1.0} \rfloor \varepsilon \end{aligned}$$

defines two containers, one identified by number 1, with no water, population density thresholds 100, 250 and 300, and desiccation time 2 days, and one identified by number 2, full of water, with population density thresholds 50, 125 and 150, and desiccation time 1 day.

A population of immature and adult *Aedes albopictus* individuals is modelled as a parallel composition of looping and containment operators, each with symbol C inside the loop part to model a specific container and a parallel composition of looping and containment operators (with symbol a inside the loop part) inside the content part to model the immature mosquitoes living inside that container, and looping and containment operators with symbol a inside the loop part to model the adult *Aedes albopictus* individuals living in open space. The whole population is then put inside another looping-containment operator with symbol En inside the loop part, which models the environment in which the population lives. In this way we model the environment in which a population lives as the outermost compartment of the Stochastic CLS term that models the biological system of interest.

Given the two containers defined above, a daytime environment at a temperature of 10° C with a population of 8 adult mosquitoes at the first gonotrophic cycle, 5 of which have sucked blood twice and 3 of which haven't sucked blood, and 2 empty containers is defined as follows.

$$\begin{aligned} Pop &::= (En)^{L}_{Temp:10\ Daylight:true} \rfloor (AdultPop \mid Containers) \\ AdultPop &::= ((a)^{L}_{\lambda} \rfloor (Adult|1|Blood^{2}))^{5} \mid (a)^{L}_{\lambda} \rfloor (Adult|1))^{3} \end{aligned}$$

We assume that the temperature in all containers is the same as the temperature in the environment. Propagations of temperature changes in the environment to the containers are handled by the event handler algorithm as we will explain in Section 4.4.

4.3 Modelling Internal Events

We have seen in Section 4.1 that the lifecycle of *Aedes albopictus* consists of the following 14 stages: egg, larva (instar 1–4), pupa and adult (8 gonotrophic cycles). Internal events describe transitions between some of these stages as well as other events occurring at a specific stage. We identify 29 internal events and we model the effect of each of them on the system by a rewrite rule:

Rule R1 egg hatch

Rules R2–R4 transitions between instars

Rule R5 pupation

Rule R6 adult emergence

Rule R7 blood sucking

Rules R8–R15 oviposition at each gonotrophic cycle

Rules R16–29 death at each stage of the life cycle (14 events)

Rules R1–R5, which model transitions between immature development stages, rule R6, which models the transition from the last immature development stage to the first adult stage, and rules R16–R21, which model the death events in such stages, are shown in Figure 1.

The duration of an immature stage depends on temperature and is measured in degree-days. Degree-days for each immature stage is defined as the number of days it takes for an individual in that stage to develop at 1°C above the minimum temperature for development (MTD) [1]. Following this definition, we define the values of temperature in the environmental information as the difference between the actual temperature and MTD.

If d_i is the average duration of the i-th development stage, then the rate constant of the rule modelling the transition from stage i to the next stage is $1/d_i$. This is true if there are no other events occurring during this stage. For every immature development stage we define one rule for the transition to the next stage and another one for the death event. Rate functions for transitions in immature stages and death events are computed by multiplying $1/d_i$ by survivability rate at i-th stage and by death rate at i-th stage, respectively. We assume that the sum of death rate and survivability rate at one development stage is equal to one. Since the duration of an immature stage depends on temperature, the rate is then multiplied by the difference between the current temperature and MTD. The death rate at an immature stage is defined locally for each container and depends on the population density of the container. We classify the population density in a container into 4 classes of density: sparse, normal, crowded and overcrowded. We define 3 thresholds to be used to classify density: ϕ_1, ϕ_2 and ϕ_3. These three thresholds are part of the

$$(C)^L_x \rfloor \big(Y|(a)^L_\lambda \rfloor (Egg|X)\big) \xmapsto{f_1} (C)^L_x \rfloor \big(Y|(a)^L_\lambda \rfloor (Larva|1|X)\big) \quad \text{(R1)}$$

$$(C)^L_x \rfloor \big(Y|(a)^L_\lambda \rfloor (Larva|1|X)\big) \xmapsto{f_2} (C)^L_x \big(Y|(a)^L_\lambda \rfloor (Larva|2|X)\big) \quad \text{(R2)}$$

$$(C)^L_x \rfloor \big(Y|(a)^L_\lambda \rfloor (Larva|2|X)\big) \xmapsto{f_3} (C)^L_x \rfloor \big(Y|(a)^L_\lambda \rfloor (Larva|3|X)\big) \quad \text{(R3)}$$

$$(C)^L_x \rfloor \big(Y|(a)^L_\lambda \rfloor (Larva|3|X)\big) \xmapsto{f_4} (C)^L_x \rfloor \big(Y|(a)^L_\lambda \rfloor (Larva|4|X)\big) \quad \text{(R4)}$$

$$(C)^L_x \rfloor \big(Y|(a)^L_\lambda \rfloor (Larva|4|X)\big) \xmapsto{f_5} (C)^L_x \rfloor \big(Y|(a)^L_\lambda \rfloor (Pupa|X)\big) \quad \text{(R5)}$$

$$(C)^L_x \rfloor \big(Y|(a)^L_\lambda \rfloor (Pupa|X)\big) \xmapsto{f_6} (C)^L_x \rfloor Y \mid (a)^L_\lambda \rfloor (Adult|1|X) \quad \text{(R6)}$$

$$(C)^L_x \rfloor \big((a)^L_\lambda \rfloor (Egg|X) \mid Y\big) \xmapsto{f_{16}} (C)^L_x \rfloor Y \quad \text{(R16)}$$

$$(C)^L_x \rfloor \big((a)^L_\lambda \rfloor (Larva|1|X) \mid Y\big) \xmapsto{f_{17}} (C)^L_x \rfloor Y \quad \text{(R17)}$$

$$(C)^L_x \rfloor \big((a)^L_\lambda \rfloor (Larva|2|X) \mid Y\big) \xmapsto{f_{18}} (C)^L_x \rfloor Y \quad \text{(R18)}$$

$$(C)^L_x \rfloor \big((a)^L_\lambda \rfloor (Larva|3|X) \mid Y\big) \xmapsto{f_{19}} (C)^L_x \rfloor Y \quad \text{(R19)}$$

$$(C)^L_x \rfloor \big((a)^L_\lambda \rfloor (Larva|4|X) \mid Y\big) \xmapsto{f_{20}} (C)^L_x \rfloor Y \quad \text{(R20)}$$

$$(C)^L_x \rfloor \big((a)^L_\lambda \rfloor (Pupa|X) \mid Y\big) \xmapsto{f_{21}} (C)^L_x \rfloor Y \quad \text{(R21)}$$

Figure 1: Rewrite rules for the immature stages of *Aedes albopictus*

environmental information attached to each container. The rate functions for rules R1–R6 and R16–R21 are computed as follows:

$$f_i((C)^L_I \rfloor (T)) = \begin{cases} \frac{V_{Temp}\cdot(1-DR(i,n,V_{Vol},V_{\phi_1},V_{\phi_2},V_{\phi_3}))}{DD(i)} & \text{if } i \in [1,6] \\ \frac{V_{Temp}\cdot DR(i-15,n,V_{Vol},V_{\phi_1},V_{\phi_2},V_{\phi_3})}{DD(i-15)} & \text{if } i \in [16,21] \end{cases} \quad (4)$$

where

- i is the index of the rewrite rule,
- $I = ind{:}k \;\; Temp{:}V_{Temp} \;\; Vol{:}V_{Vol} \;\; \phi_1{:}V_{\phi_1} \;\; \phi_2{:}V_{\phi_2} \;\; \phi_3{:}V_{\phi_3} \;\; DTime{:}V_{DTime}$ is the environmental information attached to the container to which rule Ri is applied,
- V_{Temp} is the container temperature,
- $DR(j,n,V_{Vol},V_{\phi_1},V_{\phi_2},V_{\phi_3})$ is the death rate function at immature stage j for the container which contains n immature mosquitoes, with density thresholds ϕ_1,ϕ_2,ϕ_3 and contains a volume V_{Vol} of water,
- $DD(j)$ represents the duration of stage j in degree-days.

We use 4 classes of population density (sparse, normal, crowded and overcrowded) to define death rate in our model. We use the following assumptions for all containers:

- threshold values used to classify population density in a container are defined for the case in which the container is full of water,
- the baseline death rate of stage i is the death rate of the population in a container whose population density is normal,

- when population in one container is overcrowded or there is no more water in the container only death events can occur, so the death rate is set to 1,
- death rate increases by 20% above the baseline death rate if population density is crowded,
- death rate decreases by 20% below the baseline death rate if population density is sparse,
- when a container is only half full, the values of thresholds used to classify the population density are divided by 2.

We define the death rate function $DR : \mathbb{N} \times \mathbb{N} \times \mathscr{E} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ as follows:

$$DR(j,n,V,\phi_1,\phi_2,\phi_3) = \begin{cases} 1 & \text{if } V \text{ is } empty \\ 1 & \text{if } V \text{ is } full \text{ and } n \geq \phi_3 \\ 1.2 \cdot BDR(j) & \text{if } V \text{ is } full \text{ and } \phi_2 \leq n < \phi_3 \\ BDR(j) & \text{if } V \text{ is } full \text{ and } \phi_1 \leq n < \phi_2 \\ 0.8 \cdot BDR(j) & \text{if } V \text{ is } full \text{ and } n < \phi_1 \\ DR(j,2n,full,\phi_1,\phi_2,\phi_3) & \text{if } V \text{ is } half-full \end{cases} \tag{5}$$

where $BDR(j)$ is the baseline death rate for phase j of the life cycle, n is the number of immature mosquitoes in the container, ϕ_1, ϕ_2, ϕ_3 are the container density thresholds and V is the volume of water in the container.

The adult life of an *Aedes albopictus* is divided into 8 gonotrophic cycles. Every gonotrophic cycle consists of two internal events: blood sucking and oviposition. The oviposition (the sixth internal event) is also a transition from one gonotrophic cycle to the next gonotrophic cycle. Figure 2 shows the rewrite rules modelling blood-sucking and oviposition events. Rule $R7$ models the blood sucking by adult mosquitoes. All adult mosquitoes have the same probability of sucking blood. We assume that a mosquito always sucks a constant amount of blood. To oviposit, the amount of blood sucked by an adult female must be above a threshold (represented by φ in rules R8–R15).

Rules R8–R15 model the oviposition for the 8 gonotrophic cycles of the mosquito. We assume that all adults die after ovipositing at the 8th gonotrophic cycle. The number of eggs any female can produce in each gonotrophic cycle is between 45 and 80. This number declines over age. We model this by defining function *eggs*, for each gonotrophic cycle j of the mosquito.

$$eggs(j) = \begin{cases} 40 & \text{if } j = 1 \\ 37 & \text{if } j = 2 \\ 35 & \text{if } j = 3 \\ 32 & \text{if } j = 4 \\ 30 & \text{if } j = 5 \\ 27 & \text{if } j = 6 \\ 25 & \text{if } j = 7 \\ 22 & \text{if } j = 8 \end{cases} \tag{6}$$

Although the number of eggs produced by a female mosquito at the j-th gonotrophic cycle is between 45 and 80, $eggs(j)$ only returns half of this value to take into account that we only model female individuals.

Finally figure 3 shows rules R22–R29, which model the death at every adult stage. Rule rates for rules R7–R15 and R22–R29 are defined as follows:

$$f_i = \begin{cases} \frac{1}{d(i)} & \text{if } i = 7 \\ \frac{(1-BDR(i))}{d(i)} & \text{if } i \in [8,15] \\ \frac{BDR(i-14)}{d(i-14)} & \text{if } i \in [22,29] \end{cases} \tag{7}$$

where $d(i)$ is the duration of stage i and $BDR(i)$ is the death rate at stage i.

All rules presented in this section are implemented by using Maude rewrite laws.

$$(En)^L_{Daylight:true\ x} \rfloor (Y|(a)^L_\lambda \rfloor (Adult|X|Blood^q)) \xmapsto{f_7} (En)^L_{Daylight:true\ x} \rfloor (Y|(a)^L_\lambda \rfloor (Adult|X|Blood^{q+1})) \quad \text{(R7)}$$

$$[q > \varphi]\ (En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|1|X|Blood^q)|(C)^L_y \rfloor Z) \xmapsto{f_8} (En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|2|X)|(C)^L_y \rfloor (Z|((a)^L_\lambda \rfloor (Egg|X))^{eggs(1)}) \quad \text{(R8)}$$

$$[q > \varphi]\ (En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|2|X|Blood^q)|(C)^L_y \rfloor Z) \xmapsto{f_9} (En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|3|X)|(C)^L_y \rfloor (Z|((a)^L_\lambda \rfloor (Egg|X))^{eggs(2)}) \quad \text{(R9)}$$

$$[q > \varphi]\ (En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|3|X|Blood^q)|(C)^L_y \rfloor Z) \xmapsto{f_{10}} (En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|4|X)|(C)^L_y \rfloor (Z|((a)^L_\lambda \rfloor (Egg|X))^{eggs(3)}) \quad \text{(R10)}$$

$$[q > \varphi]\ (En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|4|X|Blood^q)|(C)^L_y \rfloor Z) \xmapsto{f_{11}} (En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|5|X)|(C)^L_y \rfloor (Z|((a)^L_\lambda \rfloor (Egg|X))^{eggs(4)}) \quad \text{(R11)}$$

$$[q > \varphi]\ (En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|5|X|Blood^q)|(C)^L_y \rfloor Z) \xmapsto{f_{12}} (En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|6|X)|(C)^L_y \rfloor (Z|((a)^L_\lambda \rfloor (Egg|X))^{eggs(5)}) \quad \text{(R12)}$$

$$[q > \varphi]\ (En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|6|X|Blood^q)|(C)^L_y \rfloor Z) \xmapsto{f_{13}} (En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|7|X)|(C)^L_y \rfloor (Z|((a)^L_\lambda \rfloor (Egg|X))^{eggs(6)}) \quad \text{(R13)}$$

$$[q > \varphi]\ (En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|7|X|Blood^q)|(C)^L_y \rfloor Z) \xmapsto{f_{14}} (En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|8|X)|(C)^L_y \rfloor (Z|((a)^L_\lambda \rfloor (Egg|X))^{eggs(7)}) \quad \text{(R14)}$$

$$[q > \varphi]\ (En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|8|X|Blood^q)|(C)^L_y \rfloor Z) \xmapsto{f_{15}} (En)^L_x \rfloor (Y|(C)^L_y \rfloor (Z|((a)^L_\lambda \rfloor (Egg|X))^{eggs(8)}) \quad \text{(R15)}$$

Figure 2: Rewrite rules for blood-sucking and oviposition events of *Aedes albopictus*

4.3.1 Implementation Strategies

Since we use Maude to implement our model, which mosquito is chosen in the application of rule R7 depends on the strategy implemented in Maude. To guarantee fairness we implement our own strategy in choosing the mosquito with the smallest number of blood sucking times first.

All adult mosquitoes in a given development stage that have sucked enough blood have the same probability of ovipositing. Therefore we consider one rule for each development stage (rules R8–R15). We have to deal with the same problem (of choosing the mosquito to oviposit) as in rule R7. To guarantee fairness we define a strategy of choosing the mosquito, based on how many times the mosquito has sucked blood. As a consequence of the strategy defined for rule R7 the number of times a mosquito sucks blood is proportional to the time spent in adult stages. Our strategy will choose the mosquito with the biggest number of blood sucking times of ovipositing first.

We also implement a strategy for choosing the container in which a mosquito oviposits. This strategy randomly chooses the container in which the mosquito oviposits.

The three strategies we have defined in this section have a different purpose from the strategy defined by Basuki, Cerone and Milazzo [7], which was used to choose which rewrite rule to apply during a

$$(En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|1|X)) \xmapsto{f_{22}} (En)^L_x \rfloor Y \quad \text{(R22)}$$
$$(En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|2|X)) \xmapsto{f_{23}} (En)^L_x \rfloor Y \quad \text{(R23)}$$
$$(En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|3|X)) \xmapsto{f_{24}} (En)^L_x \rfloor Y \quad \text{(R24)}$$
$$(En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|4|X)) \xmapsto{f_{25}} (En)^L_x \rfloor Y \quad \text{(R25)}$$
$$(En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|5|X)) \xmapsto{f_{26}} (En)^L_x \rfloor Y \quad \text{(R26)}$$
$$(En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|6|X)) \xmapsto{f_{27}} (En)^L_x \rfloor Y \quad \text{(R27)}$$
$$(En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|7|X)) \xmapsto{f_{28}} (En)^L_x \rfloor Y \quad \text{(R28)}$$
$$(En)^L_x \rfloor (Y|(a)^L_\lambda \rfloor (Adult|8|X)) \xmapsto{f_{29}} (En)^L_x \rfloor Y \quad \text{(R29)}$$

Figure 3: Rewrite rules for death events in adult phases of *Aedes albopictus* life cycle

simulation. Instead, the strategies defined in this section are used to choose which portion of the term that models the system state matches the lefthand side of a rewrite rule.

4.4 Modelling External Events

External events are events that cannot be controlled by the system. These events are usually used to model changes in the environment that affect the population. Every event is modelled as a triplet (N,V,t), where the event name N is used to distinguish the kind of event, the event value V is used to update the environmental information in the system state and the event time t is the time when the event is scheduled to occur. Event names and values will be explained in the next paragraphs. Event time t is a non-negative real number and measures time in days. The integer part of t represents the day and the fractional part represents the time of the day at which an event should occur. For instance $t = 1.5$ means that the event is scheduled to occur on day 1 at 12 pm, and $t = 4.125$ means that the event is scheduled to occur on day 4 at 3 am.

As explained in Section 4.2, for each container there are seven kinds of environmental information in our model: container index, container temperature, volume of water in the container, three container thresholds for population density and container desiccation time. External events must deal with these kinds of environmental information. We define four kinds of event: light change event, change of temperature, desiccation, and rainfall. Light change events are scheduled twice a day, one at sunrise and another at sunset.

A sunrise event changes the *Daylight* information associated with the environment from $false$ to $true$. A sunset event changes the *Daylight* information from $true$ to $false$. A change of temperature event updates temperature in all compartments. A desiccation event updates the volume of water in a specific container. A rainfall event updates the volume of water in all containers. Container indices are used by the event handler algorithm to handle all events that occur. Population density thresholds are used to compute propensity after population density in one container is updated due to the occurrence of a desiccation or a rainfall event. Container desiccation time is used to schedule new desiccation events due to the occurrence of a desiccation or a rainfall event.

A light change event is modelled as a triplet $(Light,V,t)$. The time when the sun rises and the time

when the sun sets depend on the position of a place on the earth and the time of the year. Value V determines whether the event is sunrise ($V = sunrise$) or sunset ($V = sunset$) event. For instance in a place where in a winter day the sun rises at 8 am and sets at 5 pm, the sunrise event on day 1 is modelled as a triplet $(Light, sunrise, 1.33)$ and the sunset event on the same day is modelled as $(Light, sunset, 1.71)$.

Temperature affects the duration of immature phases of the mosquito development. We model a temperature change as a triplet $(Temp, V_{Temp}, t)$, which is interpreted as the event of setting the temperature to a new value V_{Temp} starting from time t. We consider only the average daily temperature. We schedule one temperature change event every day at midnight. So a triplet $(Temp, 10, 3.0)$ means that the average temperature on day 3 is 10°C above the MTD of *Aedes albopictus*.

The desiccation event is modelled as a triplet $(Desic, i, t)$ which is interpreted as a desiccation in the container with index i at time t. We assume that the desiccation time depends on container type and measure this time as the number of days needed to reduce the water volume by one level (from $full$ to $half{-}full$ or from $half{-}full$ to *empty*). Initially, we introduce one event for each container in list L scheduled according to the desiccation time of the container to which it refers. Every time a desiccation event occurs and the container is not yet empty, another desiccation event is scheduled to reduce the water volume to the next level. For instance, if the system state is represented as:

$$(En)^L_I \rfloor ((C)^L_{ind:1\ Vol:empty\ DTime:2.0\ I'} \rfloor T' \,|\, (C)^L_{ind:2\ Vol:full\ DTime:1.5\ I''} \rfloor T'')$$

where I, I' and I'' represent part of environmental information which is not relevant for desiccation events, T', T'' are terms representing population of Aedes albopictus inside container 1 and 2, respectively, and the first event in list E is $(Desic, 2, 1.0)$, then at time 1.0 the system state becomes

$$(En)^L_I \rfloor ((C)^L_{ind:1\ Vol:empty\ DTime:2.0\ I'} \rfloor T' \,|\, (C)^L_{ind:2\ Vol:half-full\ DTime:1.5\ I''} \rfloor T'')\,.$$

The event $(Desic, 2, 1.0)$ is removed from and a new desiccation event $(Desic, 2, 2.5)$ is added to list E.

In our model we only consider containers stored outdoors. In this way, rainfalls are scheduled events that increase the water volume level in all containers. Rainfalls are assumed to be prescheduled initially. Every time a rainfall event occurs, all desiccation events have to be removed from the list and new desiccation events should be added. We classify rainfalls as *heavy* and $light$. A heavy rainfall increases the water volume level of all containers to $full$. A light rainfall increases the water volume level of all containers from *empty* to $half{-}full$ or from $half{-}full$ to $full$. The rainfall event is modelled as a triplet $(Rain, lev, t)$ which represents a rainfall event with level lev (*heavy* or $light$) starting at time t. For instance, if the system state is represented as:

$$(En)^L_I \rfloor ((C)^L_{ind:1\ Vol:empty\ DTime:2.0\ I'} \rfloor T' \,|\, (C)^L_{ind:2\ Vol:half-full\ DTime:1.5\ I''} \rfloor T'')$$

and list E contains three events $(Rain, light, 1.25)$, $(Desic, 2, 1.5)$ and $(Desic, 1, 2.0)$, then at time 1.25 the system state becomes

$$(En)^L_I \rfloor ((C)^L_{ind:1\ Vol:half-full\ DTime:2.0\ I'} \rfloor T' \,|\, (C)^L_{ind:2\ Vol:full\ DTime:1.5\ I''} \rfloor T'')$$

The three events are removed from the list and two new desiccation events $(Desic, 2, 2.75)$ and $(Desic, 1, 3.25)$ are added to the list.

The event handler algorithm is very simple. Given a list E of events, N_C containers and a term T that represents the system state, the algorithm removes the first event (N_E, V_E, t_E) from E and performs the different actions decribed above according to the value of $N_E \in \{Light, Temp, Desic, Rain\}$. The removal of the first event from the list and the subsequent actions are implemented by using Maude rewrite laws.

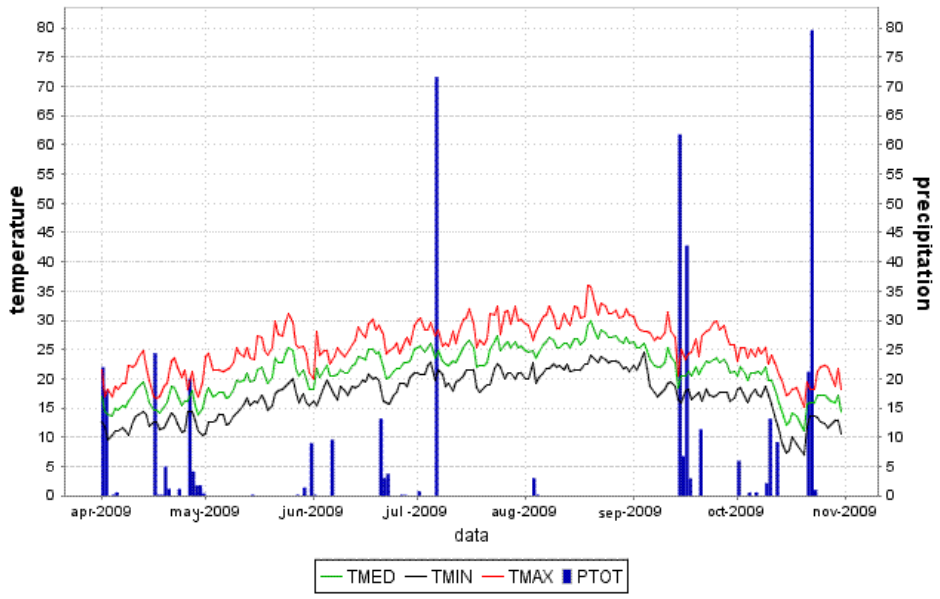


Figure 4: Temperature and Rainfall in Massa Carrara, Italy

4.5 In silico Experiment and Analysis

As already mentioned, we have implemented our model in Maude. We have then run a simulation by using data collected during May–November 2009 in the province of Massa-Carrara (Tuscany, Italy) in 11 CO_2 mosquito traps. The 11 traps have captured a total of 3535 *Aedes albopictus* individuals, and have been checked at the following dates: 8 May (4 Aedes a.), 15 May (25 Aedes a.), 19 May (81 Aedes a.), 5 June (33 Aedes a.), 18 June (167 Aedes a.), 3 July (360 Aedes a.), 14 July (561 Aedes a.), 29 July (381 Aedes a.), 19 August (486 Aedes a.), 3 September (471 Aedes a.), 19 September (276 Aedes a.), 23 September (292 Aedes a.), 14 October (398 Aedes a.). Note that traps need to be charged with CO_2 in order to work, and that the charge allows the trap to work for one day. Hence, data refer to captures of mosquitoes in one day for each considered date. This way of sampling mosquito populations follows standard practice.

Figure 4 shows the climatic data (temperature in °C and rainfalls in *mm*) during May–November 2009 in Massa-Carrara province. In our simulation we use 8.8°C as MTD [24] and 11 containers. Each container has carrying capacity of 100–250 organisms and desiccation time between 4.5 and 9.0 days.

In our simulation we initialise the population with 4 adult mosquitoes (which equals the number of adult mosquitoes collected on 8 May 2009) and 10 immature mosquitoes in each of the 11 containers, 6 eggs, 2 instar-1 larvae, 1 instar-2 larva and 1 instar-3 larva. The water volume level in each container is initially set to half-full. We also set initial desiccation events according the desiccation times of the containers. Let t_0 be the time when the simulation starts and DT_i be the desiccation time of container identified by index i, then we set initial desiccation events at time $t_0 + DT_i$ for $i = 1$ to 11.

Figure 5 shows the result of our simulation compared with the population sampling produced by using the 11 traps. We can notice some differences between the simulation results and the field sampling. For example, the number of mosquitoes in the sampling decreases between 19 May and 5 June, whereas in the simulation such number rapidly increases. This probably happens because of the coarse classification of rainfalls in our model: a very tiny rainfall, with neglectable effect in reality, which occurs just

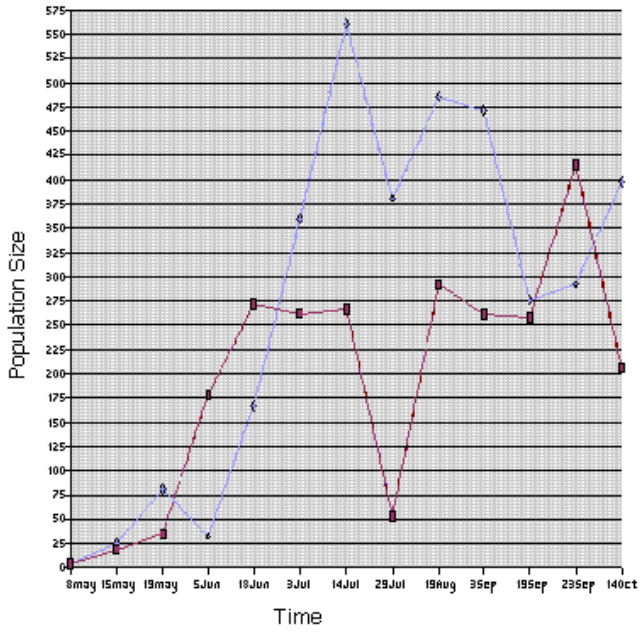


Figure 5: Comparison of in silico simulation (dark line) with data sampled from mosquito traps (light line)

before 19 May, is classified as light rain and, as a result, increases the level of water of the containers in the simulation. This may indicate that we need to improve our model by using a finer classification of rainfalls.

The number of mosquitoes captured in traps rapidly increases from 18 June to 14 July, probably due to rainfalls. However, no population growth is shown by the simulation during that period. This may be due to an overweighed effect that temperature decrease has in our model on immature stage duration and death rates. It may also be due to too small values for dessication times used in our model.

In the simulation the effect of the heavy rainfalls that occur just before 19 September immediately causes a population increase on 23 September, but the subsequent decrease in rainfalls and heavy decrease in temperature lead to a population decrease on 14 October. In the field sampling, instead, population samples keep increasing from 19 September to 14 October. This difference might indicate that decrease of rainfalls and temperature take a longer time in reality to affect the population growth than in our simulation. This might be again due to too small values for dessication times used in our model. Moreover, a decrease in temperature might cause a slower desiccation, a phenomenon that is not considered in our model.

5 Conclusions

We have presented an extension of the Calculus of Looping Sequences aimed at describing population dynamics and ecosystems. The extension consists in allowing a list of external events to be provided by the modeller in order to describe environmental events such as changes in the climatic conditions. The modeller has also to provide an event handler algorithm that is used by the simulation algorithm

associated with the extended formalism. The event handler algorithm is invoked every time an external event is planned to occur and it changes the simulation state in accordance with the type of the considered event.

We have used the extended formalism to give a model of a population of *Aedes albopictus*, an aggressive mosquito that is well known as an important disease vector. A simulator for this model has been developed via translation into the specification language Maude. We have compared results of simulations of our model with real data obtained from the sampling of mosquitoes during May-November 2009 in the province of Massa-Carrara (Tuscany, Italy). Since changes in the temperature and rainfalls have a significant effect on the mosquito population dynamics, we have exploited data on such environmental events (in the same area and the same period of the sampling of mosquitoes) to construct a list of external events for the model.

The results of our simulations show some differences from the real data. However, these differences seem to be motivated by some restrictive modelling choices that could be revised in order to construct an improved and finer model. Improvements to the model are hence part of our future work, which includes also

- modelling of populations of other disease vector mosquitos such as *Aedes aegypti*;
- study of dynamics of populations in other geographic areas;
- study of different control policies to the mosquito population.

It would be particularly interesting to study the effects of events such as periodic cleaning of containers and use of pesticides on the mosquito population to choose the most promising control policy. Such a policy could then be experimented in the field, and the results obtained could be used to further validate the model. A method to choose the best mosquito population control policy would be of interest in particular in those areas in which such mosquitoes act as vectors of diseases.

Acknowledgment

Antonio Cerone would like to thank Syed Mohamed Aljunid and Jamal Hisham Hashim for their hospitality and helpful discussion at UNU-IIGH in Kuala Lumpur and for further email discussions which inspired this work.

This work has been supported partly by UNU-IIST core funding, partly by a grant from Fondazione Monte dei Paschi di Siena for the project “Un approccio etologico/computazionale allo studio dei meccanismi dell’evoluzione delle specie animali” and partly by a grant for international inter-university collaborations from the Italian Ministry for Education and Scientific Research (MIUR).

References

[1] Jorge A. Ahumada, Dennis Lapointe & Michael D. Samuel (2004): *Modeling the Population Dynamics of Culex quinquefasciatus (Diptera: Culicidae), along An Elevational Gradient in Hawaii*. *Journal of Medical Entomology* 41(6), pp. 1157 – 1170.

[2] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, P. Tiberi & A. Troina (2008): *Stochastic CLS for the Modeling and Simulation of Biological Systems*. In: *Transactions on Computational Systems Biology ix*, pp. 86 – 113.

[3] Roberto Barbuti, Giulio Caravagna, Andrea Maggiolo-Schettini, Paolo Milazzo & Giovanni Pardini (2008): *The Calculus of Looping Sequences*. In: M. Bernardo, P. Degano & G. Zavattaro, editors: *Formal Methods for Computational Systems Biology, LNCS 5016*, Springer, pp. 387 – 423.

[4] Roberto Barbuti, Andrea Maggiolo-Schettini, Paolo Milazzo & Giovanni Pardini (2008): *Spatial Calculus of Looping Sequences*. *Electronic Notes in Theoretical Computer Science* 229(1), pp. 21 – 39.

[5] Roberto Barbuti, Andrea Maggiolo-Schettini, Paolo Milazzo & Luca Tesei (2009): *Timed P Automata*. *Electronic Notes in Theoretical Computer Science* 227, pp. 21 – 36.

[6] Thomas Anung Basuki, Antonio Cerone & Rafael V. Carvalho (2009): *Modelling Cell Cycle Using Different Levels of Representation*. *EPTCS* 11, pp. 51 – 69.

[7] Thomas Anung Basuki, Antonio Cerone & Paolo Milazzo (2009): *Translating Stochastic CLS into Maude*. *Electronic Notes on Theoretical Computer Science* 227, pp. 37 – 58.

[8] Carrie Marie Bradford (2005): *Effects of Weather on Mosquito Biology, Behavior, and Potential for West Nile Virus Transmission on the Southern Hihg Plains of Texas*. Ph.D. thesis, Texas Tech. University.

[9] Laurence Calzone, François Fages & Sylvain Soliman (2006): *BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge*. *Bioinformatics* 22(14), pp. 1805–1807.

[10] Mónica Cardona, M. Angels Colomer, Antoni Margalida, Ignacio Pérez-Hurtado, Mario J. Pérez-Jiménez & Delfí Sanuy (2010): *A P system based model of an ecosystem of some scavenger birds* LNCS 5957, pp. 182–195.

[11] K. L. Chan, B. C. Ho & Y. C. Chan (1971): *Aedes aegypti (L.) and Aedes albopictus (Skuse) in Singapore City, 2. Larval Habitats*. *Bulletin of World Health Organization* 44, pp. 629–633.

[12] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer & Carolyn Talcott (2003): *The Maude 2.0 System*. In: Robert Nieuwenhuis, editor: *Rewriting Techniques and Applications (RTA 2003)*, number 2706 in Lecture Notes in Computer Science, Springer-Verlag, pp. 76–87.

[13] V. Danos & C. Laneve (2004): *Formal Molecular Biology*. *Theoretical Computer Science* 325, pp. 69 – 110.

[14] H. Delatte, G. Gimonneau, A. Triboire & D. Fontenille (2009): *Influence of Temperature on Immature Development, Survival, Longevity, Fecundity, and Gonotrophic Cycles of Aedes albopictus, Vector of Chikungunya and Dengue in the Indian Ocean*. *Journal of Medical Entomology* 46(1), pp. 33 – 41.

[15] Roger Eritja, Raúl Escosa, Javier Lucientes, Eduard Marquès, Ricardo Molina & Santiago Ruiz (2005): *Worldwide Invasion of Vector Mosquitoes: Present European Distribution and Challenges for Spain*. *Biological Invasions* 7(1), pp. 87 – 97.

[16] Daniel T. Gillespie (1976): *A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions*. *The Journal of Computational Physics* 22(4), pp. 403 – 434.

[17] Daniel T. Gillespie (1977): *Exact Stochastic Simulation of Coupled Chemical Reactions*. *The Journal of Physical Chemistry* 81(25), pp. 2340 – 2361.

[18] Frédéric J. Lardeux, Rosenka H. Tejerina, Vicente Quispe & Tamara K. Chavez (2008): *A Physiological Time Analysis of the Duration of the Gonotrophic Cycle of Anopheles pseudopunctipennis and Its Implications for Malaria Transmission in Bolivia*. *Malaria Journal* 7.

[19] M.B. Madon, J.E. Hazelrigg, M.W. Shaw, S. Kluh & M.S. Mulla (2003): *Has Aedes albopicuts Established in California?* *Journal of the American Mosquito Control Association* 19(4), pp. 297 – 300.

[20] Chris McCaig, Rachel Norman & Carron Shankland (2008): *Process Algebra Models of Population Dynamics*. In: *Algebraic Biology (AB 2008), LNCS 5147*, Springer, pp. 139–155.

[21] Paolo Milazzo (2007): *Qualitative and Quantitative Formal Modeling of Biological Systems*. Ph.D. thesis, University of Pisa.

[22] Corrado Priami & Paola Quaglia (2005): *Beta Binders for Biological Interactions*. In: V.Danos & V.Schachter, editors: *Computational Methods in Systems Biology. International Conference CMSB 2004, LNCS 3082*, Springer, pp. 20 – 33.

[23] Aviv Regev, Ekaterina M. Panina, William Silverman, Luca Cardelli & Ehud Shapiro (2004): *BioAmbients: An Abstraction for Biological Compartments*. *Theoretical Computer Science* 325(1), pp. 141–167.

[24] Hwa-Jen Teng & Charles S. Apperson (2000): *Development and Survival of Immature Aedes albopictus and Aedes triseriatus (Diptera: Culicidae) in the Laboratory: Effects of Density, Food, and Competition on Response to Temperature*. *Journal of Medical Entomology* 37(1), pp. 40 – 52.

An Individual-based Probabilistic Model for Fish Stock Simulation

Federico Buti
School of Science and Technology
University of Camerino, Italy
federico.buti@unicam.it

Flavio Corradini
School of Science and Technology
University of Camerino, Italy
flavio.corradini@unicam.it

Emanuela Merelli
School of Science and Technology
University of Camerino, Italy
emanuela.merelli@unicam.it

Elio Paschini
CNR - Institute of Marine Sciences
Ancona, Italy
e.paschini@ismar.cnr.it

Pierluigi Penna
CNR - Institute of Marine Sciences
Ancona, Italy
p.penna@an.ismar.cnr.it

Luca Tesei
School of Science and Technology
University of Camerino, Italy
luca.tesei@unicam.it

We define an individual-based probabilistic model of a sole (Solea solea) behaviour. The individual model is given in terms of an Extended Probabilistic Discrete Timed Automaton (EPDTA), a new formalism that is introduced in the paper and that is shown to be interpretable as a Markov decision process. A given EPDTA model can be probabilistically model-checked by giving a suitable translation into syntax accepted by existing model-checkers. In order to simulate the dynamics of a given population of soles in different environmental scenarios, an agent-based simulation environment is defined in which each agent implements the behaviour of the given EPDTA model. By varying the probabilities and the characteristic functions embedded in the EPDTA model it is possible to represent different scenarios and to tune the model itself by comparing the results of the simulations with real data about the sole stock in the North Adriatic sea, available from the recent project SoleMon. The simulator is presented and made available for its adaptation to other species.

1 Introduction

Ecosystems are composed of living animals, plants and non-living structures that exist together and interact with each other. Fish are part of the marine ecosystem and interact closely with their physical, chemical and biological environment. They are inter-dependent with the ecosystem that provides the right conditions for their growth, reproduction and survival. Conversely, they are a source of food for other animals and form an integral part of the marine food web.

The fishing activity impacts both on the fish stocks and on the ecosystem within which they live. The Ecosystem Approach to Fisheries (EAF) [17] recognises that fisheries have to be managed as part of their ecosystem and that the impact on the environment should be limited as much as possible. Part of this approach is the fish stock assessment. A "stock" is a population of a species living in a defined geographical area with similar biological parameters (e.g. growth, size at maturity, fecundity etc.) and a shared mortality rate. Its aim is to provide information to managers on the state and life history of the stocks. This information is used into the decision making process. Stock assessment can be made using mathematical and statistical models to examine the history of the stock and to make quantitative predictions in order to address the following questions: 1) What is the current state of the stock? 2) What has happened to the stock in the past? 3) What will happen to the stock in the future if different management choices are made? Fisheries employs a wide variety of recognised assessment models and statistical methods to assess the stocks of fish. If we know about the stock size (biomass) and the biology

To appear in:
AMCA-POP 2010 – Electronic Proceedings in Theoretical Computer Science (EPTCS)

of the fish stock, we can estimate how many fish can be safely removed from the stock in order to ensure a sustainable resource.

Using the data from a recent research project [15, 16] on the common sole (*Solea solea*), it was possible to obtain new information on the biology of this fish. These data are the input of mathematical models based on equations determining the stock assessment of this species. This makes possible to regulate the fishing effort in order to avoid overfishing. In this work we want to introduce a somewhat new way of addressing fish dynamic population modelling and fish stock assessment. The main characteristic of our approach is that it is individual-based, that is to say, every single individual of the population under study is considered as an independent entity and the dynamics of the overall population living in a given environment *emerges* from the individual interactions and behaviours. Every aspect of the population can then be observed and measured in a simulation environment. This last aspect permits the tuning and the validation of the individual model by using existent experimental data.

Since systems biology was proposed as a challenge of a new way of understanding biology, it has involved biologists, physicians, mathematicians, physicists, computer scientists and engineers. In particular, in the computer science community a lot of models, languages, approaches and methodologies have been applied in a biological context, and several formalisms have been specifically developed for describing different aspects of biological systems. In [7], authors extend P systems with features typical of timed automata with the aim of describing periodic environmental events such as seasons or periodical hunts/harvests. In [13] it is proposed a modelling framework based on P systems and it is applied to the modelling of the dynamics of some scavenger birds in the Pyrenees. This model considers information about the feeding of the population. In [6], a spatial extension of P systems is introduced and an example of the evolution of ring species, based on small changes between geographically contiguous populations, is modelled. Authors of [25] present a process algebraic approach to the modelling of population dynamics. Currently no time characterisation can be provided of the modelled biological environment because the calculus has not a notion of time. Stamatopoulou et al. provide, in [28] and [29], models based on X-machines and P systems for biological-inspired systems such as colony of ants or bees, flocks of birds and so on. Besozzi et al. [10] model metapopulations (which are ecological models describing the interactions and the behaviour of populations living in fragmented habitats) by means of dynamical probabilistic P systems where additional structural features have been defined (e.g., a weighted graph associated with the membrane structure and the reduction of maximal parallelism). Such a work effectively uses many regions to model an ecological system, thus it really exploits the advantages of the membrane structure. In [18] authors model the behaviour of a bee colony as a society of communicating agents acting in parallel and synchronising their behaviour. Two models are provided: one is based on P systems while the other is based on X-machines but no tool, thus no actual results, are available to compare the behaviour of the two models. Finally, in an older work of Bahr and Bekoff [5] authors model a flock in terms of cellular automata; although interesting, theirs work concentrates only on the *vigilance* of the flock and how it is affected by internal and external factors (such as flock size, number of obstacles and so on).

The individual-based vision is quite natural in the computer science world, since notions such as process, component, activity, flow, interaction all can be easily related to an executor or a virtual entity. When adapting these notions to a biological scenario it is natural to reason in terms of entities that "do something" and probably collaborate to make the whole system function well. On the other side, biologists have often a view that abstracts from single entities preferring to reason in terms of aggregated variables, for which a continuous domain can be adopted and that are related by differential equations (ODE, PDE). Consequently, the available data from observations and experiments follow this way of thinking and are not directly interpretable in an individual-based setting. To bridge the gap there is the

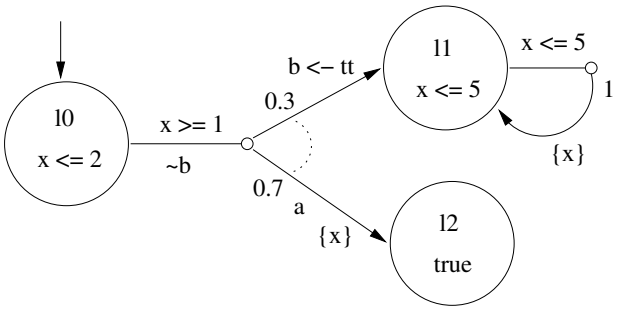


Figure 1: A simple EPDTA.

need to develop methodologies and software systems that make the two worlds interact and somehow work in synergy to transport the advantages of each view into the other and vice versa.

In this paper we define a formalism called Extended Probabilistic Discrete Timed Automata (EPDTA) that is a variant of probabilistic timed automata [23]. It simplifies the time domain, that is discrete, but introduces integer and boolean variables in the state. The formalism is shown to be interpretable as a Markov decision process and also easily translatable to a syntax that is accepted by the probabilistic model-checker PRISM [20, 22]. A model of the behaviour of the common sole (*Solea solea*) living in the North Adriatic sea is then given in terms of an EPDTA by using available data from a recent project [15, 16]. After the individual behaviour is defined we introduce a simulation environment that is agent-based and derives from the one developed in [26]. Essentially, it creates a Multi-Agent System (MAS) in which every agent represents a sole whose internal (probabilistic) behaviour is given by the individual model. The MAS permits a precise monitoring of all the events occurring in the virtual square kilometre of sea that is simulated. The simulator, called FIShPASs (FIshing Stock Probabilistic Agent-based simulator), is available [2] and easily adaptable to simulate other species.

The paper is organised as follows: Section 2 defines EPDTAs and gives their semantics as a Markov decision process. Section 3 shows a particular EPDTA representing the individual behaviour of a sole living in the North Adriatic sea. Section 4 introduces the simulator as a MAS in which each agent implements the individual probabilistic behaviour described in Section 3 and shows the preliminary simulation results that have to be tuned/validated using the real observation data available form the SoleMon project. Section 5 concludes, describing some future work.

2 Extended Probabilistic Discrete Timed Automata

In this section we introduce EPDTAs, a variant of probabilistic timed automata that we need for our purpose. We then show that an automaton of this kind can be interpreted as a Markov decision process and that can be translated easily into one handled by the model checker PRISM [20, 22].

Briefly, a timed automaton (TA) [4] is an automaton equipped with real-valued clock variables and such that transitions are guarded by clock constraints. The control flows from a state to another instantaneously if and only if the guard of that transition is enabled. Each transition has an action associated and can reset some clocks. While the control stays in a state, time elapse i.e. clock values increase. Possible conditions in states, called invariants, can prevent the passage of time forcing the control to exit from that state with one of the enabled transitions.

Following the approach of [11], probability has been introduced into timed automata yielding prob-

abilistic timed automata (PTA) [8, 23]. In this case every transition from a state has a clock constraint as a guard, but then the action, reset and destination state is given by a finite probability distribution. Thus, every step of a PTA consists in a resolution of non-determinism among different enabled transitions, passage of time included, followed by a probabilistic choice of the action, reset and destination according to the given distribution.

An EPDTA is essentially a PTA with the set $\mathbb{N}$ of natural numbers as time domain and in which the locations are enriched with a finite set of boolean and finite-range integer variables. Clocks, that are considered similar to integer variables with range $[0,\infty]$, grow at discrete steps of length 1. We also add a subset of actions that are called *urgent* and that must be executed as soon as they are enabled. The motivation of these variants are essentially given by the peculiarities of the models of individuals in ecosystems: their state-changes are typically modelled in terms of transitions that happen at a time scale of years, months and, in the finer grain, weeks. Thus, there is no need of continuous time. Moreover, each individual has some characteristics, e.g. age, sex, length, weight, fertility, last time of reproduction, etc. that are easily representable by integer (with finite range) and boolean variables and that influence its behaviour. Last characteristic of this meta-model is a constant $\texttt{MAX\_TIME} \in \mathbb{N}$ that represents the maximum number of time steps that the automaton can perform. This means that each clock has, actually, a range $[0,\texttt{MAX\_TIME}]$. Since this constant can be chosen arbitrarily large, this requirement does not limit the generality of the meta-model both if it is used in a simulation environment and if it is used for model checking. On the other hand it permits to give a finite range to all variables of the model, clocks included.

Now we define in detail the syntax and the semantics of EPDTAs. This part is quite technical and can be skipped by the non-familiar reader. Section 3 will present the model of the Solea solea behaviour as a particular EPDTA that can be quite intuitively understood even without knowing all the technical details. The idea of clock variables is central in the framework of timed automata and it is imported in our meta-model. A *clock* is a variable that takes values from the set $\mathbb{N}$. Clocks measure time as it elapses, all clocks of a given system advance at the same pace and clock variables are ranged over by $x, y, z, \ldots$ We use $X, X', \ldots$ to denote finite sets of clocks. A *clock valuation* over X is a function assigning a natural number to every clock. The set of valuations of X, denoted by V_X, is the set of total functions from X to $\mathbb{N}$. Clock valuations are ranged over by $\nu, \nu', \ldots$. Given $\nu \in V_X$ and $n \in \mathbb{N}$, we use $\nu + n$ to denote the valuation that maps each clock $x \in X$ into $\nu(x) + n$.

Clock variables, like other variables, can be assigned during the evolution of the system when certain actions are performed. The assignment consists in instantaneously set the value of a variable to a new value. Clock variables are always assigned to 0, i.e. they are reset. Immediately after this operation a clock restarts to measure time at the same pace as the others. The reset is useful to measure the time elapsed since the last action/event that reset the clock. Given a set X of clocks, a *reset* γ is a subset of X. The set of all resets of clocks in X is denoted by Γ_X and reset sets are ranged over by $\gamma, \gamma', \ldots$ Given a valuation $\nu \in V_X$ and a reset γ, we let $\nu[\gamma$ be the valuation that assign the value 0 to every clock in γ and assign $\nu(x)$ to every clock $x \in X \setminus \gamma$.

We need also to consider a finite set B of boolean variables, ranged over by $b, b', \ldots$, a finite set I of integer variables, ranged over by $v, v', \ldots$, together with a range assignment function $\texttt{range} : I \leftarrow \mathbb{Z} \times \mathbb{Z}$ such that if $\texttt{range}(v) = (z_1, z_2)$ then $z_1 \leq z_2$. Finally, we need a finite set F of totally specified functions, ranged over by $f, f', \ldots$, that we use as tables in which constants values are collected and where then they can be retrieved by applying each function to values in its domain (essentially they are tables of probability values or array of constants). Such tables can contain rational numbers. If they are involved in integer operations they are rounded to the closest integer.

The grammars introduced in the following define the syntax of a first-order language in which very usual functions and relations are present. The language can express boolean and arithmetic expres-

sions. Moreover, we define a syntax for expressing assignments to variable of corresponding type, clock constraints and guards. $Bexp ::= tt \mid ff \mid b \mid Bexp \wedge Bexp \mid \sim Bexp \mid Aexp = Aexp \mid Aexp <= Aexp \mid Aexp < Aexp \mid (Bexp)$, $Aexp ::= z \mid v \mid f(Aexp,Aexp,...)$[1] $\mid Aexp + Aexp \mid Aexp * Aexp \mid Aexp - Aexp \mid Aexp/Aexp$[2] $\mid Aexp\%Aexp$[3] $\mid (Aexp)$ where $z \in \mathbb{Z}$. Assignments are of the form $Assign ::= b \leftarrow Bexp \mid v \leftarrow Aexp \mid Assign, Assign$. Boolean expressions are ranged over by $\beta, \beta', \ldots$, arithmetic expressions are ranged over by $\alpha, \alpha', \ldots$ and assignments are ranged over by $\eta, \eta', \ldots$. The timed behaviour of the system is expressed using constraints on the actual values of the clocks. Given a set X of clocks, the set Ψ_X of *clock constraints* over X is defined by the following grammar: $\psi ::= true \mid false \mid x \# c \mid x - y \# c \mid \psi \wedge \psi$ where $x, y \in \mathscr{X}$, $c \in \mathbb{N}$, and $\# \in \{<, >, \leq, \geq, =\}$. Finally, guards, ranged over by $g, g', \ldots$ are defined as $Guard ::= \psi \mid Bexp \mid Guard \wedge Guard$. As usual, we use the name of the syntactic category to denote the set of the generated objects. Thus, for instance, *Guard* represents the set of all strings that are well-formed guards.

Given sets B, I, we define an interpretation ι as a function assigning a value to every variable in B and I. By means of an interpretation ι we can evaluate a boolean expression β or an arithmetic expression α in the standard sense; we denote with $\mathscr{E}_\iota(\beta)$ the boolean value of β and with $\mathscr{E}_\iota(\alpha)$ the integer value of α both using the interpretation ι. Moreover, we can define a satisfaction relation $\models$ such that $\nu \models \psi$ if the values of the clocks in ν satisfy the constraint ψ in the natural interpretation. Finally, the satisfaction relation can be extended naturally on guards: $\iota, \nu \models g$.

An assignment η is evaluated as a change in the interpretation ι. We denote with $\mathscr{A}(\iota, \eta)$ a new interpretation ι' in which the variables that are assigned in η are all[4] changed with the corresponding values, evaluated from ι in the above sense.

Given a set H let us denote by $\mu(H)$ the set of *finite* probability distributions over H i.e. $\mu(H)$ contains functions $p\colon H \to [0,1]$ such that $\sum_{h \in H} p(h) = 1$ and the set $\{h \in H \mid p(h) > 0\}$ is finite. A probability distribution p can be represented as follows: $p = [h_1 \mapsto p_1, \ldots, h_n \mapsto p_n]$ where the h_i's are exactly all elements of H that have $p(h_i) = p_i > 0$.

Definition 2.1 (EPDTA)**.** *An* extended probabilistic discrete timed automaton T *is a tuple* $(Q, \Sigma, B, I, X, \mathbb{E}, \mathbb{U}, q_0, \iota_0, Inv)$*, where:* Q *is a finite set of locations,* Σ *is a finite alphabet of actions,* B *is a finite set of boolean variables,* I *is a finite set of finite-range integer variables,* X *is a finite set of clocks,* $\mathbb{E}$ *is a finite set of edges,* $\mathbb{U}$ *is a finite set of* urgent *edges,* q_0 *is the initial location,* ι_0 *is the initial interpretation of the variables of* $B \cup I$*,* `MAX_TIME` *is the maximum time of evolution and* Inv *is a function assigning to every* $q \in Q$ *an* invariant*, i.e. a clock constraint* ψ *such that for each clock valuation* $\nu \in V_X$ *and for each* $n \in \mathbb{N}^{>0}$*,* $\nu + n \models \psi \Rightarrow \nu \models \psi$*. Constraints having this property are called* past-closed*.*

Each edge $e \in \mathbb{E} \cup \mathbb{U}$ *is a tuple in* $Q \times Guard \times \mu(\Sigma \times Assign \times \wp(X) \times Q)$*. If* $e = (q, g, prob)$ *is an edge,* q *is the* source*,* g *is the* guard *and* $prob$ *is the* distribution*. If* $prob((a, \eta, \gamma, q')) > 0$ *then there is a possibility for the automaton to reach the* target *location* q' *performing the* action a*, the* assignment η *and the* reset γ*.*

Figure 1 shows an EPDTA with three locations $l0, l1, l2$. The set of clocks is $\{x\}$, the alphabet is $\{a\}$, $l0$ is the initial state, and the invariant of state $l0$ is $x \leq 2$. There is an edge starting from location $l0$ with a guard that is the conjunction of the clock constraint $x \geq 1$ and the boolean expression $\sim b$, where $b \in B$. At the edge it is associated a distribution $[(l1, a, \varepsilon, \{x\}) \mapsto 0.7, (l2, \varepsilon, b \leftarrow tt, \{\}) \mapsto 0.3]$, where ε is the

[1] According to what said above, this can be considered a constant. Of course the arguments of the function must be of the right number and of the right type.

[2] Integer division.

[3] Rest of integer division.

[4] Note that we suppose that η does not contain more than one assignment for each variable.

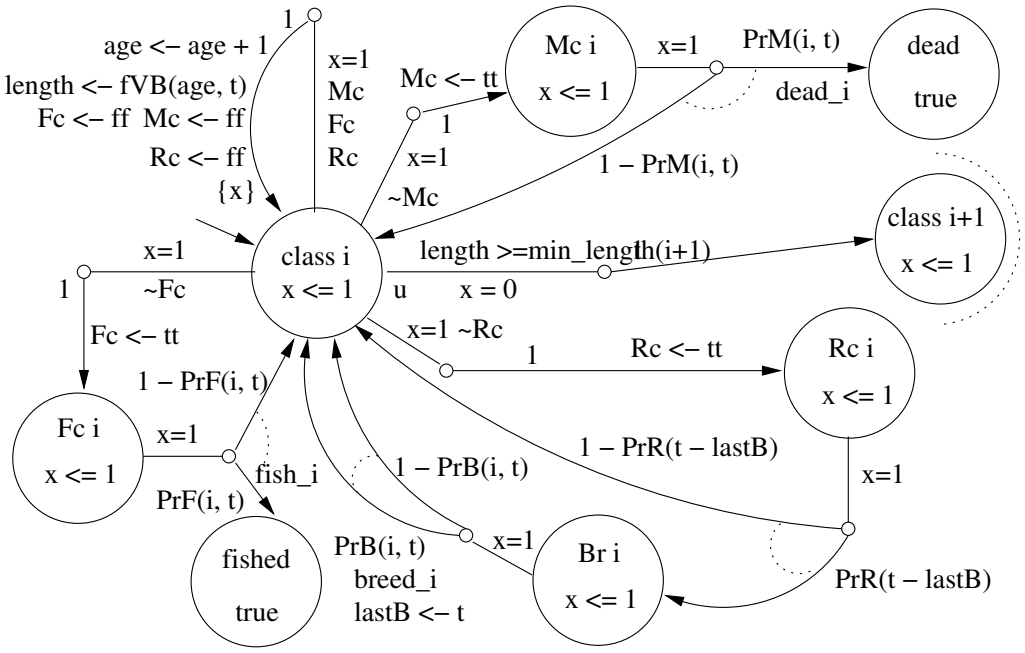


Figure 2: EPDTA representing the behaviour of the sole when it is in class *i*. The other classes are equal.

empty string. From $l1$ there is an edge in which the probability distribution is trivial. This transition is equivalent to a "classical" one.

The semantics of an EPDTA is a Markov decision process. A Markov decision process (MDP) is a pair $(\mathcal{S}, \texttt{Steps})$ where $\mathcal{S}$ is a set of states and `Steps` is a function giving for each state s a set of probability distributions. Each $p \in \texttt{Steps}(s)$ is a discrete probability distribution in $\mu(\mathcal{S})$, saying the probability of each state of being the next state s' in the process. A given MDP evolves as follows: at each step it is in a state s. Firstly it performs a non-deterministic choice to decide which distribution $p \in \texttt{Steps}(s)$ it will apply. Then it performs a probabilistic choice to go to a new state s' according to the chosen p. The process then cycles again.

The semantics of $T = (Q, \Sigma, B, I, X, \mathbb{E}, \mathbb{U}, q_0, \iota_0, Inv)$ is a MDP $(S, \texttt{Steps})$ where the set of states S is the set of all the tuples of the form (q, ν, ι) where $q \in Q \cup \{stop\}$[5], $\nu \in V_{X \cup \{t\}}$ is a valuation of the set of clocks X augmented with a fresh clock t that is never reset[6] and ι is an interpretation of the variables in $B \cup I$. Note that if we fix a `MAX_TIME` as the maximum time step for the system evolution then the set of states if finite as Q is finite and all variables, clock included, have a finite range.

For every state $s = (q, \nu, \iota)$ the set of distributions $\texttt{Steps}(s)$ is determined by the following rules:

Stop **if** $\nu(t) = \texttt{MAX\_TIME}$ **then** $[(stop, \nu, \iota) \mapsto 1] \in \texttt{Steps}(s)$

Time **if** $\nu + 1 \models Inv(q)$ **and** $\nu(t) + 1 \leq \texttt{MAX\_TIME}$ **and** $(\forall (q', g', prob') \in \mathbb{U}(q' = q \Rightarrow \iota, \nu \not\models g'))$ **then** $[(q, \nu + 1, \iota) \mapsto 1] \in \texttt{Steps}(s)$

Urgent **if** $\nu(t) \neq \texttt{MAX\_TIME}$ **and** $(q, g, prob) \in \mathbb{U}$ **and** $\iota, \nu \models g$ **then** $[(q'_1, \nu \backslash \gamma_1, \mathscr{A}(\iota, \eta_1)) \mapsto p_1, \ldots, (q'_n, \nu \backslash \gamma_n, \mathscr{A}(\iota, \eta_n)) \mapsto p_n] \in \texttt{Steps}(s)$ **where** $prob = [(a_1, \eta_1, \gamma_1, q'_1) \mapsto p_1, \ldots, (a_n, \eta_n, \gamma_n, q'_n) \mapsto p_n]$

[5]The fresh location *stop* is added to terminate the activity of the automaton when the maximum time is reached.

[6]This clock is needed to count the global elapsed time.

Non-Urgent **if** $\nu(t) \neq \texttt{MAX\_TIME}$ **and** $(q,g,prob) \in \mathbb{E}$ **and** $\iota,\nu \models g$ **and** $(\forall (q',g',prob') \in \mathbb{U}(q' = q \Rightarrow \iota,\nu \not\models g'))$ **then** $[(q'_1,\nu\backslash\gamma_1,\mathscr{A}(\iota,\eta_1)) \mapsto p_1,\ldots,(q'_n,\nu\backslash\gamma_n,\mathscr{A}(\iota,\eta_n)) \mapsto p_n] \in \texttt{Steps}(s)$ **where** $prob = [(a_1,\eta_1,\gamma_1,q'_1) \mapsto p_1,\ldots,(a_n,\eta_n,\gamma_n,q'_n) \mapsto p_n]$

When rule **Stop** is applicable then no other rule is applicable. The process goes unconditionally to the *stop* location in which it stops. Rule **Time** lets one time unit to elapse, provided that the invariant of the current location will be satisfied at the reached state, that the maximum time was not reached and that no urgent transitions (i.e. that do not permit time passing) are enabled. Rule **Urgent** inserts in $\texttt{Steps}(s)$ all possible distributions that derive from urgent transitions. Note that in case of more than one urgent transitions enabled all are inserted in $\texttt{Steps}(s)$ and thus a non-deterministic choice is done among them by the MDP. The resulting distributions are essentially the same of the original automaton, but here all the operations are performed on the clocks and on the variables to calculate the resulting state. The last rule **Non-Urgent** is applicable only if there are not urgent transitions enabled. The effect is the same of the urgent case and among different enabled non-urgent transitions a non-deterministic choice is done by the MDP.

Proposition 2.2. *Given an EPDTA T and a natural number* MAX_TIME *it is possible to construct a Markov decision process Π in the syntax readable by the model checker PRISM such that Π and the semantics of T are the same Markov decision process.*

We plan to provide an automatic tool for this translation inside our simulator FIShPASs (see Section 4.3). This is very important because having the PRISM equivalent model improves the tests that the biologists can do against the probabilities put in the model itself. This is because quantitative questions can be asked to the model checker to test hypothesis made about the model or to validate it with available real data. A very powerful and useful logic language, Probabilistic CTL (PCTL) [20, 21], is suitable for expressing such questions.

3 Sole Characteristics and Behaviour: an EPDTA Model

The body of the common sole (*Solea solea*) is egg-shaped and flat [1, 16, 12]. The maximum body height is equal to 1/3 of the total length. The eyes are on the right side, the upper one slightly anterior to the lower. Both pectoral fins are well developed, the left one being somewhat smaller than the right one. The dorsal fin begins anterior to the eyes, by the mouth. The last rods of the dorsal and the anal fins are connected to the caudal fin, which is round. The colour on the eyed side of the body is greyish-brown to reddish-brown, with large and diffused dark spots. The pectoral fin has a blackfish spot at its distal half. The posterior margin of the caudal fin is generally dark. This common sole species lives in the eastern Atlantic, from Scandinavia to Senegal and in the entire Mediterranean. It is rare in the Black Sea.

Here we present an individual model of a sole living in the North Adriatic sea as an EPDTA. This model is quite adaptable for other species of fish or soles of different environments by varying the different characteristic functions and probability tables that are embedded in the model itself. The quantitative information (lengths, probabilities, offspring estimation) was elaborated in collaboration with the Institute of Marine Sciences of Ancona, Italy and members of their project **SoleMon** [15, 16]. As usually for this kind of fish, sole are categorized into so called *classes* which represents soles of similar age/length and thus with similar behaviour and subject to similar natural mortality or fishing. In the Adriatic sea, according to the project *SoleMon*, sole of age class 0+ aggregates inshore along the Italian coast, mostly in the area close to the Po river mouth; age class 1+ gradually migrates off-shore and adults concentrate in the deepest waters located at South West from Istria peninsula. Growth analyses on this species have

been made using otoliths, scales and tagging experiments. An otolith is a structure in the saccule or utricle of the inner ear, specifically in the vestibular labyrinth [24], whose section presents several concentric rings, very much like those of the tree trunks. By measuring the thickness of individual rings, it has been assumed (at least in some species) to estimate fish growth because fish growth is directly proportional to otolith growth. However, some studies disprove a direct link between body growth and otolith growth.

A great variability in the growth rate was noted: some specimens had grown 2 cm in one month, while others, of the same age group, needed a whole year. The von Bertalanffy growth function [9] (VBGF) introduced by von Bertalanffy in 1938 predicts the length of a fish as a function of its age:

$$f_{VB}(age) = L_\infty \left[1 - e^{-K(age-t_0)}\right]$$

The length ($f_{VB}(age)$) obtained is expressed in centimetres while *age* and t_0 are in months; the different parameters that occur in the function are partly constants and partly calculated for our specific sole case study. L_∞ is not the maximum length of the animal but the asymptote for the model of average length-at-age , K is the so-called *Brody growth rate coefficient* which, if varied, allow to manipulate the growing function in order to represent periods of low food or abundance of food (so the soles grow less or more having the same age) and t_0 is the time or age when the average size is zero. There parameters of VBGF for the Adriatic sole have been calculated using various methods. Within the framework of the SoleMon project, growth parameters of sole were estimated through the length-frequency distributions obtained from surveys. The results are $L_\infty = 39,6\,cm$, $K = 0.44$ and $t_0 = -0.46$. With this correspondence we can calculate the length of a sole of age *age* and consequently put it in one of the length classes. In the following table the ranges of the classes are shown:

Class	**Minimum length** (cm)	**Maximum length** (cm)
0	0	18.3
1	18.4	25.8
2	25.9	30.7
3	30.8	33.9
4+	34	39.6

Considering the relevance of K for the purpose of the growth function we defined in general a function $f_{VB}(age,t)$ where the parameter t is an absolute month such that different periods could have a different K. The absolute month $t = 0$ can be linked to a particular month of a particular year: in this way known past periods of low food or other environmental events can be represented in the growing function of the model. In the simulation of Section 4.4 we used always the same K along time. Knowing the length, it is possible to estimate the weight using the length-weight relationship:
$Weight(l) = a \cdot l^b$. The parameters have been estimated in SoleMon: $a = 0.007$, $b = 3.0638$. Using this relationship we can determinate, at every instant during our simulations, the biomass of the whole stock under simulation.

Natural mortality (not including fishing) has been estimated through a mortality index M available from the SoleMon project. From this annual index a probability distribution has been derived: $\mathrm{Pr}_M(i,t)$ is the probability that in a given month t a sole in class i dies for natural mortality. Fixing a specific month for $t = 0$ the values of the function are cyclic of a period of 12 months. However, in a simulation of several years the index can be varied in different years and months with a very fine granularity. This permits to represent in a simulation catastrophic periods or particularly favourable ones. The mortality probabilities $\mathrm{Pr}_M(i,t)$ (on a month basis per class) used in the simulations whose results are shown in Section 4.4 are the following:

Class	Jan.	Feb.	Mar.	Apr.	May	Jun.	Jul.	Aug.	Sep.	Oct.	Nov.	Dec.
0	0.083	0.078	0.073	0.068	0.063	0.058	0.058	0.053	0.048	0.043	0.038	0.033
1	0.032	0.031	0.030	0.023	0.030	0.028	0.028	0.028	0.027	0.026	0.026	0.025
2	0.024	0.024	0.023	0.023	0.023	0.023	0.023	0.022	0.022	0.022	0.021	0.021
3	0.021	0.021	0.021	0.021	0.021	0.021	0.021	0.021	0.021	0.021	0.021	0.021
4+	0.020	0.020	0.020	0.020	0.019	0.019	0.019	0.019	0.019	0.019	0.018	0.018

Mortality for fishing is estimated by a fishing index F. With the same reasoning done for the natural mortality a probability has been derived: $\Pr_F(i,t)$ is the probability that in a given month t a sole in class i is fished. In this case the periods of no fishing can be represented in the model. Similarly to the previous case the probability table can be cyclic over years or can be personalised month per month. The fishing index can be $F = 0$, meaning that there is no fish, can be moderate (estimated $F = 0.2$) or can be so strong that a situation of overfishing may occur (typically $F > 1$). For instance, the fishing probabilities $\Pr_F(i,t)$ (on a month basis per class) corresponding to a fishing index $F = 0.2$ are the following:

Class	Jan.	Feb.	Mar.	Apr.	May	Jun.	Jul.	Aug.	Sep.	Oct.	Nov.	Dec.
0	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65
1	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65
2	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65
3	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65
4+	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65	1.65

Breeding is another important aspect of the life of soles that has been embedded in the model. In this case two estimations are needed. The first one is the probability of being reproductive after m months since the last breed, that we denote $\Pr_R(m)$. For simplicity, this probability has been estimated as 0 for $m = 0,1,\ldots,11$ and as 1 for all $m \geq 12$. However, this can be changed and refined in future versions of the model. If a sole passes this check of fertility then there is the probability of breeding $\Pr_B(i,t)$. In this case, of course, soles in class 0 have this probability equals to 0. Soles of higher classes have higher probability to breed, but only in the appropriated months, which are from November to March of every year. This probability is then spread along these months. The table used in our simulations is the following:

Class	Jan.	Feb.	Mar.	Apr.	May	Jun.	Jul.	Aug.	Sep.	Oct.	Nov.	Dec.
0	0.3	0.25	0.1	0	0	0	0	0	0	0	0.1	0.25
1	0.3	0.25	0.1	0	0	0	0	0	0	0	0.1	0.25
2	0.3	0.25	0.1	0	0	0	0	0	0	0	0.1	0.25
3	0.3	0.25	0.1	0	0	0	0	0	0	0	0.1	0.25
4+	0.3	0.25	0.1	0	0	0	0	0	0	0	0.1	0.25

Note that in the case of breeding there is a potential artificial situation in a simulation. It can happen that one sole of the higher classes do not breed at all along one year. It is unlikely, but in the simulation may happen. This is absolutely not possible in reality. We plan to adapt the model in the future in order to avoid this situation. Another weakness of the model (and obviously of the simulator) is the lack of information about the offspring of an individual. This is a forced missing because no real information is known about the number of eggs that are fecundated nor the number of eggs that hatch, becoming new individuals. This issue is managed from the tool for the purposes of the simulation and will be treated in details in Section 4.3.

The resulting overall model is shown in Figure 2. Two clocks are used, x for counting the passage of one month and t, never reset, for measuring the absolute time since the beginning. Note that only

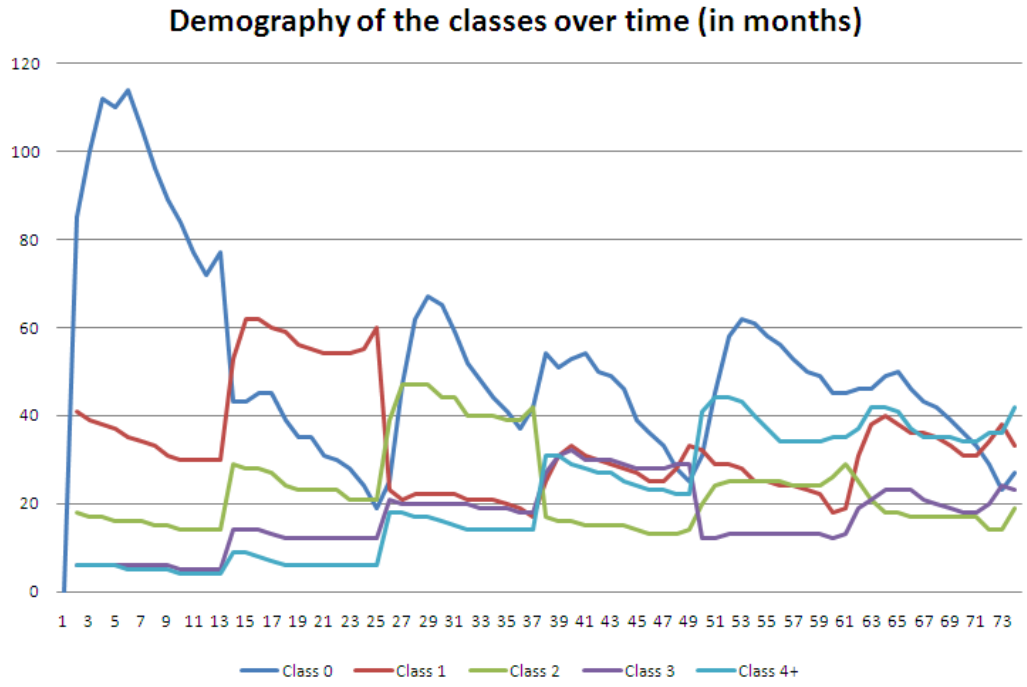


Figure 3: Demography over time without fishing ($F = 0.0$).

the transitions from one generic location "class i" are shown. The whole model is simply the resulting EPDTA considering the locations for all the classes, while the locations "dead" and "fished" are the same for all the classes. Every class has its particular fishing, mortality and breeding probability (e.g. a smaller, thus younger sole, has less probability to die/be fished than a older one). The boolean variables M_c, F_c, R_c are used to assure that every month the sole makes a mortality check (M_c), a fishing check (F_c) and a reproduction/breeding check (R_c). Note that time can advance of one month only if all these checks are done. The urgency[7] is used for forcing the class change of the class as soon as the sole reaches the minimum length for that class. The labels dead_i, fish_i and breed_i are used to communicate to the simulation environment that a particular sole (the one sending the signal) of class i died, was fished or breed. The meaning of the integer variables *age*, *length* is obvious and the variable *lastB* counts the months elapsed since the last breeding of this sole. The probability tables and the functions used in Figure have been all described above.

4 Simulation

As shown at the end of Section 2 the probability model can be automatically checked to discover interesting properties or to make consistency checks on the given probabilities. Another important way to use the same model is for simulating a sole in a population of them. The idea is not new, but it very naturally fits in the fish stock monitoring. If we want to predict what happens to a population after several years of fishing of a certain strength under normal or particular conditions, all we have to do is to instruct a

[7] In the picture the urgent transitions are indicated by a little "u" attached at the beginning of their arrows.

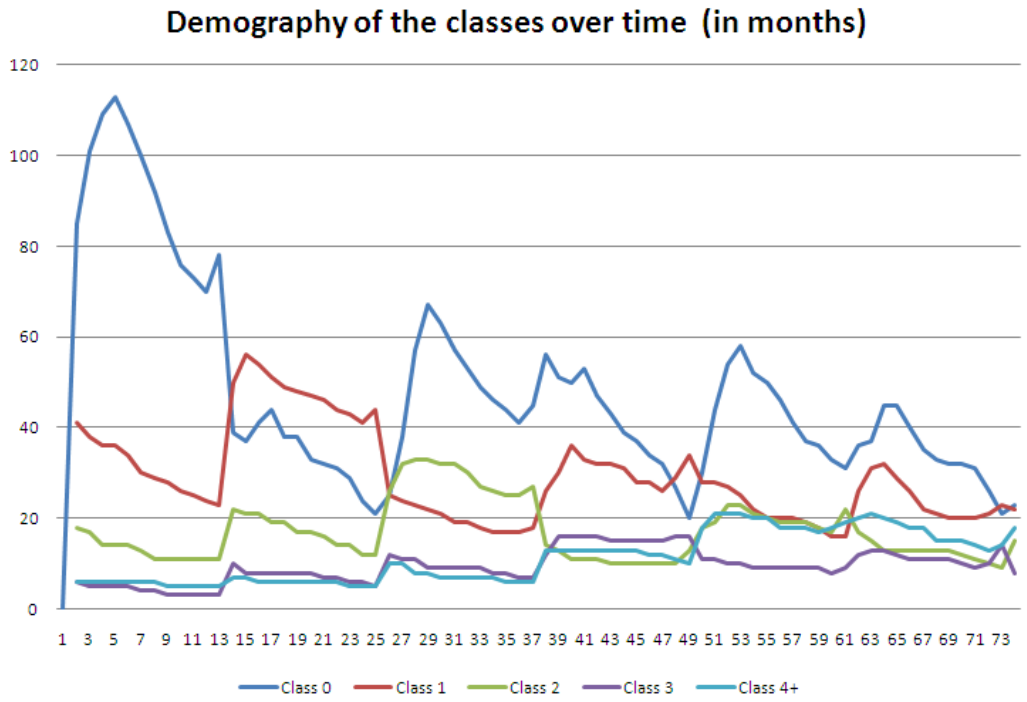


Figure 4: Demography over time with light fishing ($F = 0.2$).

population of virtual soles that uses the given model as probabilistic behaviour and let them evolve over time. In every moment, our virtual environment can show to us all the statistics we want to know about the whole stock but also about every single sole.

Naturally the hard thing to do is to precisely tune the model with the most possible available real data. This work has to be done before the simulations on the model can be considered to have a certain degree of reliability.

In this section we show the simulator that we have developed for reaching this goal. It uses agent technology, as we discuss in the following. We are currently at the very initial phase of the tuning of probabilities and other values using real data. The implementation is quite stable and the adaptability to other species can be done quite easily. More information is available at [2].

4.1 Multi-Agent Systems

Several definitions have been given for the term "agent" during the last decades, the most suited of which is the one given from Russel: an agent is something that can retrieve information from the environment through its sensor and can perform actions with its actuator [27]. Alternatively, Woldrige and Jenning [19, 30] define agent as hardware or software-based computer system that have the following properties: autonomy, reactivity, pro-activeness, and social ability. A **Multi-Agent System** (MAS) is a collection of autonomous agents that communicate, cooperate, share knowledge and solve their own problem.

In a MAS, each agent can be either cooperative or selfish; in other words the single agent can share a common goal with the others (e.g. an ant colony), or they can pursue their own interests (as in the free market economy). MAS are usually exploited when the problem considered cannot be solve (efficiently)

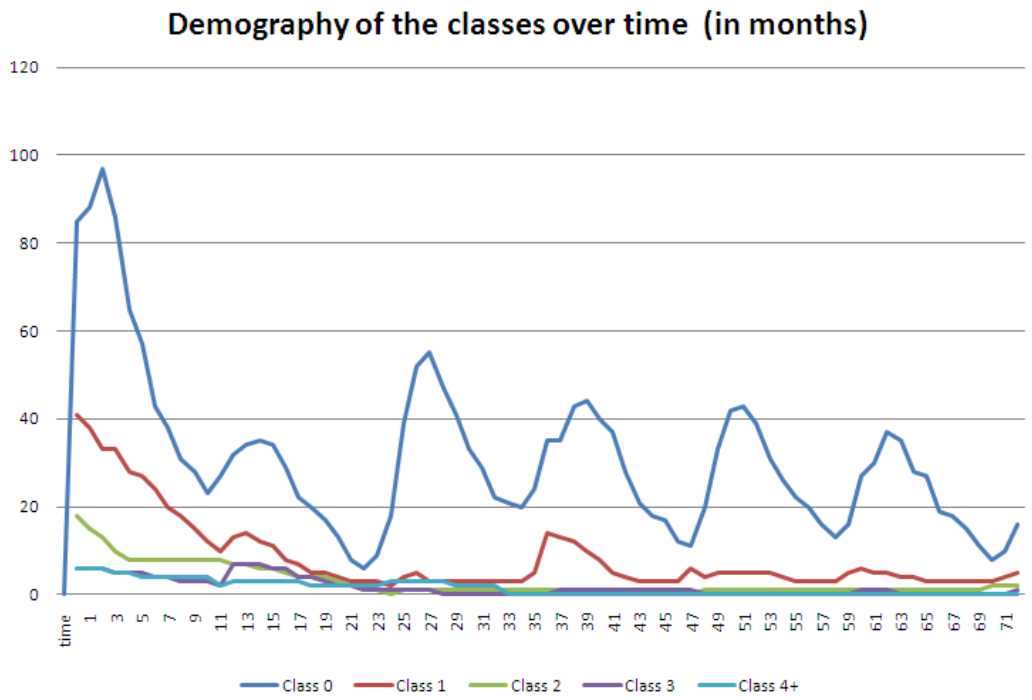


Figure 5: Demography over time with overfishing ($F = 1.2$).

by an individual agent or a monolithic system. They are used to model coordinated defence systems but also for disaster response models, social network modelling, transportation, logistics, graphics as well as in many other fields when the problem is non-linear or the interaction with flexible individual participants have to be represented or again when in-homogeneous space is relevant. Finally, MAS are widely used in networking and mobile technologies, to achieve automatic and dynamic load balancing, high scalability, and self-healing networks.

In the context of a MAS, an agent needs to *communicate* its information to the others and after that it needs to *coordinate* its activities (which is important to prevent conflicts between the agent belonging to the MAS) and *negotiate* its interest to solve a problem without conflicts. This need of interaction and exchange of information between agents is the basic characteristic that differentiate MASs from traditional artificial intelligence which work only as a single agent.

4.2 Hermes middleware

Hermes [14, 3] is an agent-based middleware for designing and execution of activity-based applications in distributed environments. It provides an integrated environment where users can focus on the design of the particular activity of interest ignoring the topological structure of the distributed environment. Hermes consists of a 3-layer software architecture: the Agent layer, the BasicServices layer and the Core layer.

An Hermes execution consists of a creation of a MAS in which the agents are of two kinds: user agents and service agents. The Agent Layer is the upper layer of the mobile platform that contains both kinds of agents. A service agent accesses to local place resources such as data and tools (which, for

security reason, are not directly accessible) while a user agent executes complex tasks and implement part of the logic of the application. Hermes is based on the concept of *place*: a place is a well defined node of a network where service agents are located. When a service agent is created on a place and bound to it, there is no way for it to migrate to another place of the network. User agents can instead be copied to another place (weak mobility) and their execution can continue on the migration place.

4.3 FIShPASs: FIshing Stock Probabilistic Agent-based Simulator

FIShPASs [2] is a simulator based on Hermes. It exploits the agent paradigm to simulate, as a MAS, the evolution of a population of fish of a certain species. At the moment a model for a sole (Solea solea) population living in the North Adriatic sea is available, but the simulator can be easily adapted for other species or soles of different environments

The basic elements of the simulator are: the `SoleaAgent`, the `SquareKilometreSea`, the `Registry` and the `Randomizer`. The `SquareKilometreSea` is exactly the Hermes place where the simulation is started. This first version of the simulator is quite simple since the sea is not spatially simulated; it is more like a *simple container* for the population of soles (exactly what is an Hermes place for its agents). In the near future we point to improve the overall model with support to space and thus displacement of sole in the simulated sea, water currents, temperature and so on.

The `SoleaAgent` is a user agent and represents a single sole in the simulated sea. While the spatial model is not so accurate, the sole *behavioural* model is quite complex. Indeed the `SoleaAgent` implements the EPDTA presented in Section 3 and thus can be fished, can die for natural mortality, can reproduce with the given probabilities, and naturally grows as time passes. Note that also the non-deterministic choices that are presented to the agent (as its behaviour is essentially a Markov decision process) are resolved probabilistically with a uniform distribution on all the enabled non-deterministic choices. As briefly discussed in Section 3, since the simulation is managed on a month basis, we can arrange theoretical probability so that certain behaviour cannot occur or can occur rarely in certain periods of the year. For instance, we can suppose that the fishing period goes (hypothetically) from October to February while in the other months fishing is prohibited and fix the fishing probability to 0 for the prohibited period. Since the probability values are given also on the basis of the length of the sole, the model can be easily adapted to different scenarios to simulate, for example, overfishing of some classes or sudden reduction of the population of some other classes because of a disease and so on.

The third element of the simulation is the `Randomizer`. It is a service agent which generates random numbers in $[0,1]$ for the sole. Every time a sole checks the probability of doing something (death, reproduction, etc.) it requests a new number to the `Randomizer`. The service returns a new generated value that is contrasted with the probability of the individual to decide whether the action occurs or not.

The last main element of the simulator is the `Registry`. Like the `Randomizer` it is a service agent and it simply keeps track of the sole available in the simulated sea. Its main purposes are the consistency control over the simulation and the generation of statistics about the simulated months (in particular, population per different class, weight of the biomass, number of death/fished soles in the last month). At the beginning of the simulation the `Registry` reads the input data and computes the number of individuals of the initial population then waits for them to communicate their status to the registry itself.

`SoleAgents` are programmed to communicate their status to the `Registry` every month in any case (even in case of death), which means that the `Registry` can always know if all the soles have communicated with it during the current interaction. Moreover, it always knows the exactly demography of the population. This communication acts also as a synchronization that ensure time consistency on the population. It is the `Registry` that manages time increments and that enables the sole to execute their

internal behaviour (setting the `SoleaAgent` variable corresponding to the local clock "x" of the model of Figure 2 to 1). Before each increment the `Registry` waits for a communication from all the population of soles and then it increments the time. In this way the `Registry` ensure that at each simulation step (i.e. each month) no sole is out of simulation time range (behind or beyond the current time).

Finally, the `Registry` generates new soles if some of the existing ones reproduced during the last month. Having the generation in the `Registry` is a strategic choice to be sure that the new born sole will be correctly set in the current time frame. In reality, a female sole produces, depending on its class, between 150000 and 250000 eggs and spreads them in the water. The number of them that will grow at least until class 1 is very low. Currently there is not a direct known relation between the number of females that breed in a month or in a season and the number of surviving and developing eggs in the following months/season. Thus, the `Registry` creates every month a number of newborn soles in class 0 that corresponds to the number observed in reality (data from SoleMon project). One challenge for the future will be trying to find a relation between the signals of breed (breed$_i$) given by the `SoleAgents` to the `Registry` in a certain period and the number of surviving newborn to introduce after some month(s).

Given the real data about individuals in the different classes from 2005 to 2008, we taken the first column, which represents the newborns (males + females) of every year, halved the values (since we consider only female soles) and distributed the newborns so obtained along the year, according to the previous fertility table.

population km^2	**0**	**1**	**2**	**3**	**4+**
2005	169	82	36	12	4
2006	92	179	43	10	1
2007	205	138	72	18	1
2008	117	123	61	10	6

In such a way we obtain the *birth rate* table below. It represents the amount of newborns that are *automatically* generated from the simulator every simulated month. Since the table covers only 4 years it is used cyclically in the subsequent years, thus the fifth year the generated newborns come from the first row of the table and so on.

Year	**Jan.**	**Feb.**	**Mar.**	**Apr.**	**May**	**Jun.**	**Jul.**	**Aug.**	**Sep.**	**Oct.**	**Nov.**	**Dec.**
1	26	21	9	0	0	0	0	0	0	0	8	21
2	14	12	4	0	0	0	0	0	0	0	4	12
3	30	25	11	0	0	0	0	0	0	0	11	25
4	16	15	6	0	0	0	0	0	0	0	6	15

Summing up, the FIShPASs simulation steps are the following:

1. the sea (place) is launched and the service agents (`Randomizer` and `Registry`) are generated on it. The `Registry` calculates the initial population
2. the sole population is generated from the place basing on the SoleMon project data
3. the soles register to the `Registry`
4. once all population has signed to the `Registry`, it generates statistics and starts their behavioural simulation by sending them a message to update their internal clock x
5. the soles execute all their operations for the current month, reset the clock x and send an ack to the `Registry`

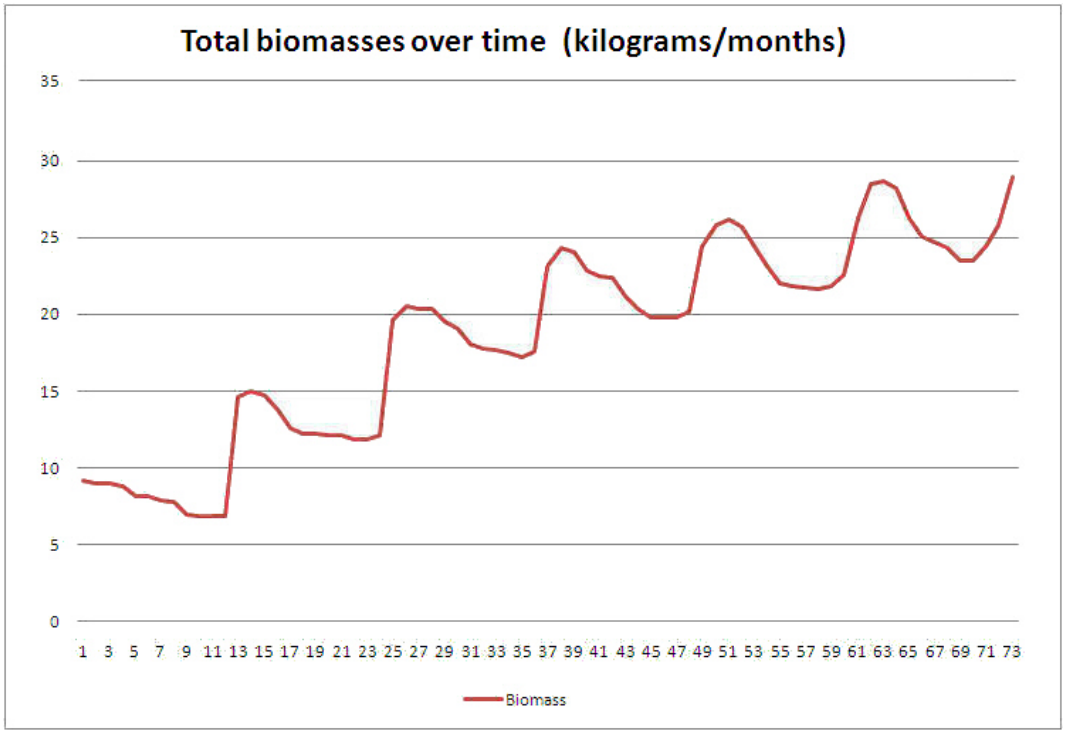


Figure 6: Biomass over time without fishing ($F = 0.0$.)

6. once all the acks are received by the `Registry`, it generates statistics for the elapsed month, creates newborn soles and then sends a new message to increment the clock x

The last two points are repeated until the desired period of time has passed.

4.4 Results

We set up a series of simulations to test our model. Every simulation has a timestep one month long and lasts for 72 months (6 years) considering always a virtual square kilometre of sea. The three charts of Figures 3, 4 and 5 show the trend of population that can occur, along the simulation, varying the fishing probability (i.e. the probability for a sole to be fished) while mortality and reproduction probabilities remain the same. The charts concentrate on three very different scenarios. The first one considers a case of no fishing (fishing index F=0.0). In this case the population remains stable, spreading on the different classes. It is particularly notable an increment of soles in the third class as well as a gradual and steady increment of individual in the fourth class.

The second case considers a light fishing activity (F=0.2). The scenario rapidly changes with classes third and fourth that grow slower than in the previous chart. Moreover all of them has difficulty to get over 20 individuals (whereas in the previous plot all of the bigger classes where around 40 individuals). The trend is that of a decreased population of soles composed of less mature individuals and some small ones (see the biomasses charts for more details).

The third and last chart about the population shows another possible scenario. In this case we suppose that the sea undergoes an overfishing activity (F= 1.2). This situation has obviously an extreme impact

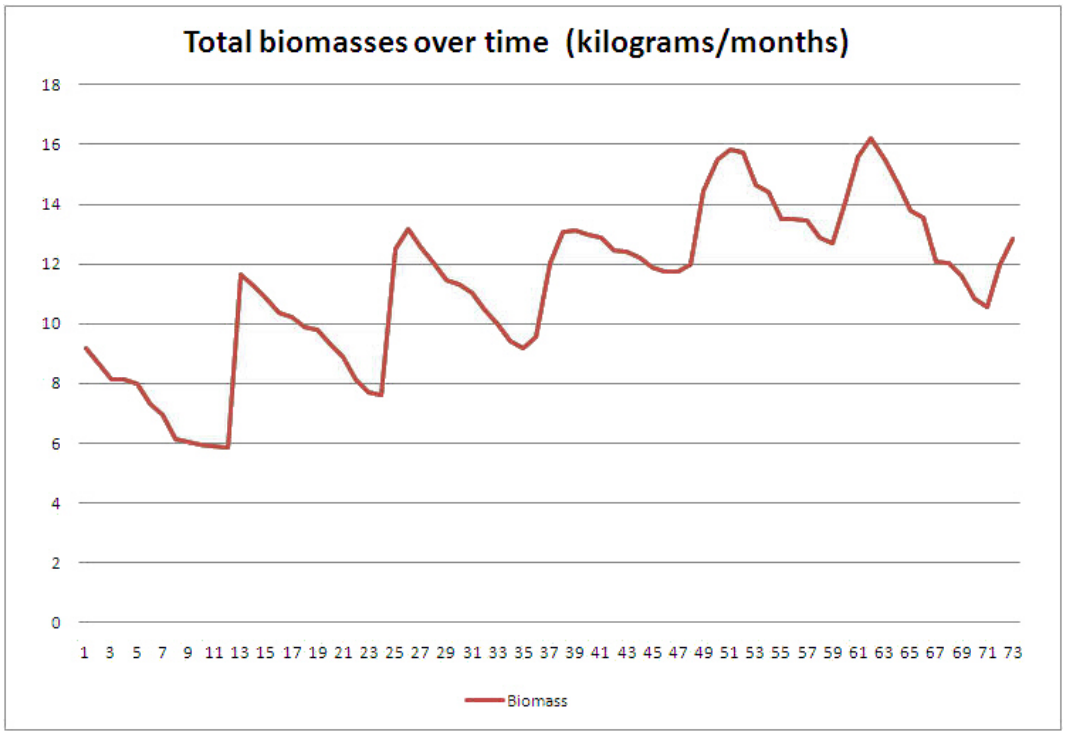


Figure 7: Biomass over time with overfishing ($F = 1.2$).

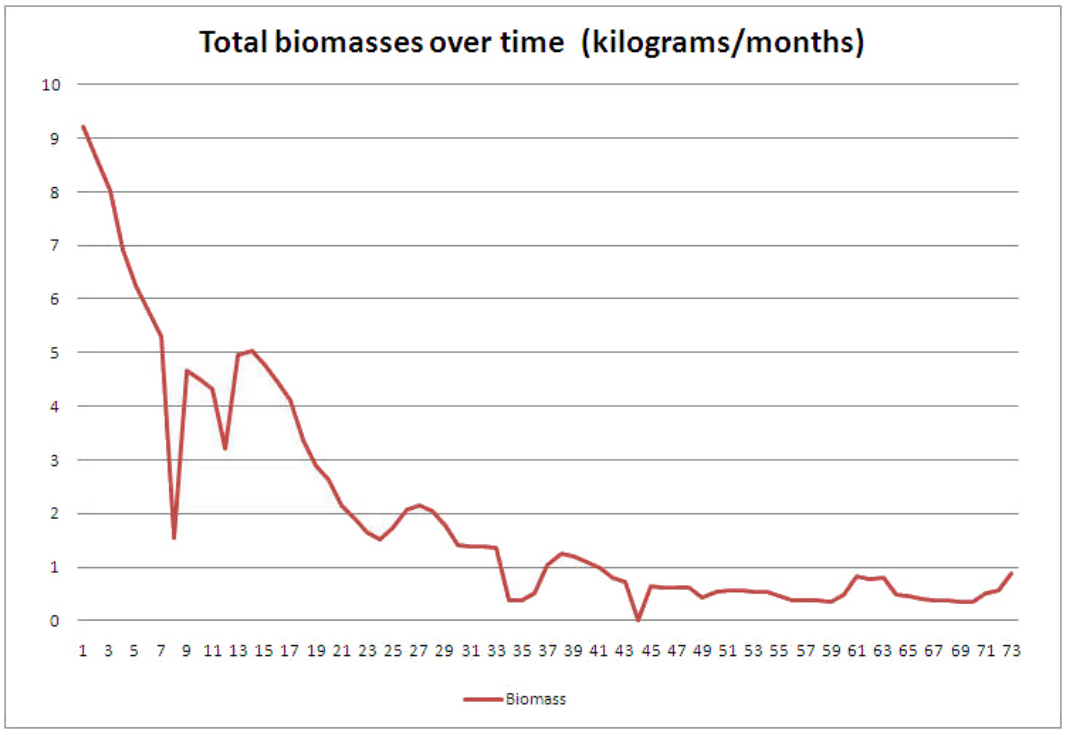


Figure 8: Biomass over time with overfishing ($F = 1.2$).

on the population. As can be seen, after two years (23-24 on the x-axis), the population is simply gone with only the 0 class of individuals available (due to their automatic generation, as explained above).

The three charts of Figures 6, 7 and 8 represent the biomass trend, i.e. the total amount (in kilograms) of soles for the different classes in the three scenarios described above. In the case of no fishing (F=0.0) the sole population tends to spread over all the classes and the biomass grows accordingly. The biomass increment is constant and at the end of the 6 years the total biomass is around 28 kg (against 8 kg at the beginning of the simulation).

When we introduce light fishing (F=0.2) the biomass tendency is similar to the previous scenario but the values are totally different. In particular the population is impoverished in the higher classes (see previous plots) and the overall biomass grows slower at the beginning with a more marked decreasing tendency during the last year (months 60 - 73) when soles grow, reaching class fourth and thus are easier to be fished. At the end of the simulation the total biomass is around 13 kg which means less than an half of the soles biomass without fishing.

With the introduction of overfishing (F=1.2) the scenario changes drastically. The fishing activities has a great impact on the population biomass that is halved at the beginning of the second year (13-15 on the x-axis) and then runs fast under 1 kg in the middle of the forth year (41-42 on the x-axis). All the bigger soles, more subject to fishing, have been caught or are dead and only the smaller ones (i.e. with a small biomass) remain (also because they are auto-generated). Again, as seen in the corresponding classes chart, the population is decimated.

More details and charts about these simulations can be found on the simulator website [2] along with contacts to request a copy of the current version of the tool.

5 Conclusions and Future Work

We have defined an individual-based model of the behaviour of a common sole (*Solea solea*) living in North Adriatic sea. The model has been specified as an Extended Probabilistic Discrete Timed Automaton (EPDTA), a formalism that is a variant of probabilistic timed automata. We have defined the semantics of an EPDTA as a Markov decision process and we have observed that an EPDTA can be translated to a syntax acceptable by the model-checker PRISM. The estimation of the probabilities and of the characteristic function of the species has been done by using the real data of the SoleMon project. The individual probabilistic behaviour then has been embedded into an agent of a MAS. The MAS simulates the population of soles over time and can provide information on the evolution of the stock by monthly statistics of the individual states. We have presented the simulator FIShPASs (FIshing Stock Probabilistic Agent-based Simulator) that implements the presented model and is easily adaptable for other species.

There are a lot of interesting things to do as future work. First, we want to tune the model, working in team with specialized biologists, in order to increase the confidence on its predictions. The translation of the model into a PRISM acceptable syntax can be made available inside the simulation environment. Having the PRISM equivalent model can highly improve the tests that the biologists can do against the probabilities put in the various tables embedded in the model. This is because quantitative questions can be asked to the model checker to test hypothesis made about the model itself or to validate it with available real data. In the MAS part a huge number of improvements are possible. For instance, soles can be given a geometrical space to occupy and can move in the simulated square kilometre. They can also emigrate and immigrate from/to the simulated space. A 3D environment, i.e. a cube kilometre, instead of a 2D one could be more appropriate because other species could be simulated simultaneously and made

interact with the soles (towards a more predator-prey approach). Moreover, a physical conformation of the territory can be added to the model possibly influencing the interactions (of different kind, to be introduced in the model too) between the individuals (the formation of an isolated population, the impossibility to meet, etc.). Finally, the effects of the passage of a particular fishing device can be modelled; for this we know there are available data for tuning/validation.

References

[1] *FAO-Adriamed, Scientific Cooperation to Support Responsible Fisheries in the Adriatic Sea*. Available at `http://www.faoadriamed.org/`.

[2] *FIShPASs: FIshing Stock Probabilistic Agent-Based Simulator*. Available at `http://cosy.cs.unicam.it/fishpass/`.

[3] *Hermes: a Middleware for Agent-based Distributed Applications and Mobile Computing*. Available at `http://hermes.cs.unicam.it/`.

[4] R. Alur & D. L. Dill (1994): *A Theory of Timed Automata*. *Theor. Comput. Sci.* 126, pp. 183–235.

[5] D. B. Bahr & M. Bekoff (1999): *Predicting flock vigilance from simple passerine interactions: modelling with cellular automata*. *Animal Behaviour* 4, pp. 831–839.

[6] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, G. Pardini & L. Tesei (2010): *Spatial P Systems*. *Natural Computing* In press. Received: 26 October 2009 Accepted: 24 February 2010 Published online: 24 March 2010.

[7] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo & L. Tesei (2009): *Timed P Automata*. *Fundamenta Informaticae* 94(1), pp. 1–19.

[8] D. Beauquier (2003): *On probabilistic timed automata*. *Theor. Comput. Sci.* 292(1), pp. 65–84.

[9] L. von Bertalanffy (1938): *A quantitative theory of organic growth (inquiries on growth laws II)*. *Human biology* 10(2), pp. 181–213.

[10] D. Besozzi, P. Cazzaniga, D. Pescini & G. Mauri (2007): *Seasonal variance in P system models for metapopulations*. *Progress in Natural Science* 17, pp. 392–400.

[11] A. Bianco & L. De Alfaro (1995): *Model Checking of Probabilistic and Nondeterministic Systems*. In: *Foundations of Software Technology and Theoretical Computer Science*, Springer-Verlag, pp. 499–513.

[12] L. Bolognini (2008): *Distribuzione spaziale e valutazione dello stock di Solea solea (Quensel 1806) in Medio e Alto Adriatico*. Master's thesis, Università Politecnica delle Marche.

[13] M. Cardona, M. Angels Colomer, A. Margalida, I. Pérez-Hurtado, M. J. Pérez-Jiménez & D. Sanuy (2010): *A P system based model of an ecosystem of some scavenger birds* 5957, pp. 182–195. WMC 2009.

[14] F. Corradini & E. Merelli (2005): *Hermes: Agent-Based Middleware for Mobile Computing*. In: *Formal Methods for Mobile Computing, SFM-Moby 2005*, *LNCS* 3465, pp. 234–270.

[15] G. Fabi, O. Giovanardi, F. Grati, I. Isajlovič, S. Raicevich, G. Scarcella & N. Vrgoč (2005-2009): *SoleMon Project. Assessmet of Sole (Solea solea) from rapido trawl surveys in GSA 17 (Northern and Central Adriatic)*. Technical Report, CNR Institute of Marine Science, Ancona, Italy.

[16] G. Fabi, O. Giovanardi, F. Grati, I. Isajlovič, S. Raicevich, G. Scarcella & N. Vrgoč (2007): *Assessmet of Sole (Solea solea) from rapido trawl surveys in GSA 17 (Northern and Central Adriatic)*. In: *9th Meeting of the Sub-Committee on Stock Assessment (SCSA), Working Group on Demersals, General Fisheries Commission for the Mediterranean (GFCM)*, Athens, Greece, pp. 9–10. (2005-2006; SoleMon Project).

[17] S. M. Garcia (2003): *The ecosystem approach to fisheries: issues, terminology, principles, institutional foundations, implementation and outlook*. Food & Agriculture Org.

[18] M. Gheorghe, M. Holcombe & P. Kefalas (2001): *Computational models of collective foraging*. *BioSystems* 61, pp. 133–141.

[19] N. R. Jennings, K. Sycara & M. Wooldridge (1998): *A Roadmap of Agent Research and Development*. *Autonomous Agents and Multi-Agent Systems* 1(1), pp. 7–38.

[20] M. Kwiatkowska, G. Norman & D. Parker (2002): *PRISM: Probabilistic Symbolic Model Checker*. In: T. Field, P. Harrison, J. Bradley & U. Harder, editors: *TOOLS '02*, *LNCS* 2324, Springer, pp. 200–204.

[21] M. Kwiatkowska, G. Norman & D. Parker (2008): *Using probabilistic model checking in systems biology*. *ACM SIGMETRICS Performance Evaluation Review* 35(4), pp. 14–21.

[22] M. Kwiatkowska, G. Norman & D. Parker (2009): *PRISM: Probabilistic Model Checking for Performance and Reliability Analysis*. *ACM SIGMETRICS Performance Evaluation Review* 36(4), pp. 40–45.

[23] M. Kwiatkowska, G. Norman, R. Segala & J. Sproston (2002): *Automatic Verification of Real-time Systems with Discrete Probability Distributions*. *Theor. Comput. Sci.* 282, pp. 101–150.

[24] F. Lagardere & H. Troadec (1997): *Age estimation in common sole Solea solea larvae: validation of daily increments and evaluation of a pattern recognition technique*. *Marine Ecology Progress Series* 155(1988), pp. 223–237.

[25] C. McCaig, R. Norman & C. Shankland (2008): *Process Algebra Models of Population Dynamics*. In: *AB '08: Proc. of the 3rd Int. Conf. on Algebraic Biology*, Springer-Verlag, Berlin, Heidelberg, pp. 139–155.

[26] P. Penna (2010): *Virtual Population Simulation with MAS (Multi-Agent System). Case study: Marine Stock of Solea Solea*. Master's thesis, School of Science and Technology, University of Camerino.

[27] S. J. Russell & P. Norvig (2003): *Artificial Intelligence: A Modern Approach*. Pearson Education.

[28] I. Stamatopoulou, M. Gheorghe & P. Kefalas (2005): *Modelling dynamic configuration of biology inspired Multi-Agent Systems with Communicating X-machines and Population P Systems*. *Lecture Notes in Computer Science* 3365, p. 389401.

[29] I. Stamatopoulou, P. Kefalas & M. Gheorghe. *OPERAS: a Framework for the Formal Modelling of Multi-Agent Systems and its Application to Swarm-based Systems*.

[30] M. Wooldridge & N. R. Jennings (1994): *Intelligent Agents: Theory and Practice*. *Knowledge Engineering Review* Submitted to Revised.

Celer: an Efficient Program for Genotype Elimination

Nicoletta De Francesco Giuseppe Lettieri
Luca Martini
Dipartimento di Ingegneria dell'Informazione: Informatica, Elettronica, Telecomunicazioni
Università di Pisa
Largo Lucio Lazzarino, 2
56122 Pisa - Italy
{n.defrancesco,g.lettieri,luca.martini}@iet.unipi.it

This paper presents an efficient program for checking Mendelian consistency in a pedigree. Since pedigrees may contain incomplete and/or erroneous information, geneticists need to pre-process them before performing linkage analysis. Removing superfluous genotypes that do not respect the Mendelian inheritance laws can speed up the linkage analysis. We have described in a formal way the Mendelian consistency problem and algorithms known in literature. The formalization helped to polish the algorithms and to find efficient data structures. The performance of the tool has been tested on a wide range of benchmarks. The results are promising if compared to other programs that treat Mendelian consistency.

keywords: abstract interpretation

1 Introduction

Geneticists employ the so-called *linkage analysis* to relate genotypic information with their corresponding phenotypic information. Genotypes are organized in data structures called *pedigrees*, that besides genetic data, record which individuals mate and their offspring. Since pedigrees may contain incomplete and/or erroneous information, geneticists need to pre-process them before performing linkage analysis. Moreover, in many cases, we cannot know any genetic information for some individuals (for instance because they refuse to or cannot be analyzed) and we would like to know which are their possible genotypes. Therefore, we would like to pre-process the pedigree by removing some candidate genotypes, in such a way that the remaining genotypes respect the classical Mendelian laws. When the pedigree is composed by thousands of individuals, this consistency checking need to be automated. The first notable contribution in the pedigree consistency check is the algorithm proposed by Lange and Goradia in 1987 [6]. The algorithm takes as input a pedigree with a list of genotypes associated to every individual, and perform genotypes elimination by removing from the lists the genotypes that lead to Mendelian inconsistencies. The algorithm performs a fixpoint iteration by processing one nuclear family at a time. This algorithm is optimal (in the sense that it removes all the genotypes that lead to Mendelian inconsistencies, and only them) when the pedigree has no loops. An example of loop in a pedigree is when two individuals that mate have an ancestor in common. An algorithm that is optimal even in the presence of loops has been proposed by O'Connell and Weeks in 1999 [8]. In brief, the algorithm selects the loop breakers (that is the individuals that, if duplicated, remove the loop) and perform the Lange Goradia algorithm for every combination of the genotypes of the loop breakers. Unfortunately, it has been proven [1] that the consistency check on pedigrees with marker data containing at least three alleles is a NP-hard problem.

The remainder of the paper is organized as follows. Firstly, we formalize the problem of genotype elimination (Section 2) and the algorithms of Lange-Goradia (Section 2.1) and O'Connell and Weeks

To appear in:
AMCA-POP 2010 – Electronic Proceedings in Theoretical Computer Science (EPTCS)

(2.2). Then, we describe the implementation of Celer (Section 3). Section 4 describes the performances of Celer on a large set of benchmarks. Then, we compare our program with other existing software (Section 5). Finally, we conclude and suggest some directions for future works.

2 Mendelian consistency algorithms

A pedigree contains parental and genetic information about a set of individuals. Pedigrees are usually represented in a graphical way by drawing a circle for every female individual and a box for every male individual. Inside the circle (or the box) there can be some data regarding the individual (for instance genetic information, or affection status). Parental relations are represented by lines that connect to a node (the so called marriage node). Arrows depart from the marriage nodes to the children of the couple. In Figure 1 we report the graph of a pedigree composed by 11 individuals. For each individual, we report his/her identification number (id from now on) and his/her possible genotypes.

We collect the parental structure in a triple $\langle I, f, m\rangle$ where I is the set of individuals and f and m are two partial functions from I into I mapping a subset $\mathrm{dom}\, f = \mathrm{dom}\, m \subset I$ of individuals to their father and mother, respectively. The individuals that do not have parents in the pedigree are called *founders*. For the pedigree of Fig. 1, the founders are the individuals with id in $\{1,2,3,6\}$.

We suppose that we are looking at a single locus. The possible alleles in the locus are in the set $\mathscr{A}$, ranged by uppercase case letters $A, B, C, \ldots$. Let $\mathscr{G}$ be the set of unordered pairs of elements in $\mathscr{A}$. Since we consider the genotypes (A,B) and (B,A) as equivalent, the genotype of each individual will be an element of the set $\mathscr{G}$. A fully specified genetic map of a pedigree $\langle I, f, m\rangle$ is an element h of $I \to \mathscr{G}$. We say that a fully specified map (fsmap from now on) is Mendelian if the genotypes of every non-founder individual is such that one of its allele is derived from the mother and the other from the father. It is often useful to check for Mendelian consistency in a subset of the individuals in the pedigree. Since the Mendelian conditions involve an individual and both his parents, it makes sense to consider those subsets that contain either both or none of the parents of each individual in the subset. Given a pedigree $\langle I, m, f\rangle$ we say that $S \subseteq I$ is a *regular* subset of I if, for each $i \in \mathrm{dom}\, f \cap S$, we have that $f(i) \in S \iff m(i) \in S$. Intersections and unions of regular subsets are again regular subsets. For instance, in the pedigree of Fig. 1, the set $\{3,4,7,8,9,11,12\}$ is an an example of a regular subset of the individuals.

We can also define a function $\mathsf{mate}\colon \mathscr{G} \times \mathscr{G} \to \wp(\mathscr{G})$ that, given two genotypes, returns the set of Mendelian genotypes that can be generated by selecting one allele from each one. We have (remember that we use unordered pairs):

$$\mathsf{mate}((A,B)(C,D)) = \{(A,C),(A,D),(B,C),(B,D)\}$$

With the help of function mate, we can now express more precisely when a fsmap is Mendelian on a regular subset of individuals:

Definition 1 (Mendelian consistency)**.** Let $P = \langle I, f, m\rangle$ be a pedigree and let S be a regular subset of I. The fully specified map h is *Mendelian on S* if and only if for every individual $i \in S$ such that $f(i) \in S$ and $m(i) \in S$, we have $h(i) \in \mathsf{mate}(h(f(i)), h(m(i)))$.

We say that an fsmap h on a pedigree $P = \langle I, f, m\rangle$ is *Mendelian* if it is Mendelian on I. The reader can verify that the fsmap in Fig. 1 is Mendelian.

Since in general we do not know precisely the genotype of each individuals, only partially specified maps will be available. A partially specified map H (psmap from now on) records for every individual of the pedigree the genotypes it may have according to our information (e.g. because we have collected

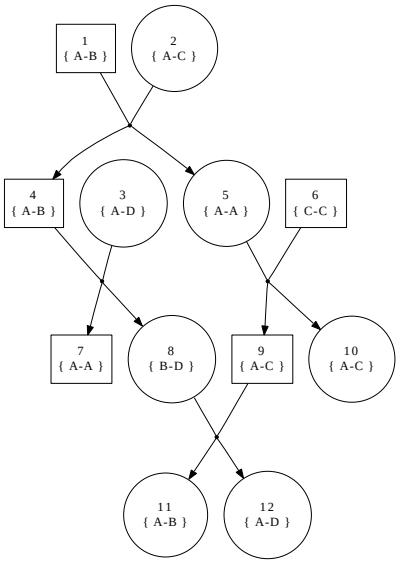


Figure 1: An example of a pedigree

some genetic data or we have observed the phenotype). A psmap H is an element of the set $I \to \wp(\mathscr{G})$. We can introduce a partial order relation $\sqsubseteq$ on set $\mathscr{M}$. We say that map H_1 is more precise than or equal to map H_2, and we write $H_1 \sqsubseteq H_2$, if and only if, for every individual $i \in I$, $H_1(i) \subseteq H_2(i)$. With an abuse of notation we identify any fully specified map h with the partially specified map that maps $\{h(i)\}$ to every individual $i \in I$. Thus we write $h \sqsubseteq H$ to mean that, for every individual i, $h(i) \in H(i)$. All psmaps such that $H(i) = \varnothing$ for any $i \in I$ describe an inconsistent situation where no possible assignment of genotypes is compatible with the available information. We identify all these psmaps and denote them by $\bot$, the psmap that maps $\varnothing$ to all individuals in I. We denote by $\mathscr{M} = (I \to \wp(\mathscr{G}))/\bot$ the set obtained by this identification. The set $\mathscr{M}$ is a complete lattice, with least upper bound $\bigsqcup$ given by pointwise union. The greatest lower bound $\bigsqcap$ is obtained in two steps: first, the pointwise intersection is computed; then, if any individual is mapped to $\varnothing$ in the previous step, the result is taken to be $\bot$.

In psmaps we are interested in those genotypes, taken from the sets of each individual, that can be used to build a Mendelian fsmap.

Definition 2 (Consistent genotype)**.** Let $P = \langle I, f, m\rangle$ be a pedigree and let S be a regular subset of I. Given a psmap H and an individual $i \in I$, we say that genotype $g \in H(i)$ is *consistent on S* iff there exists an fsmap $h \sqsubseteq H$ with $h(i) = g$ such that h is Mendelian on S.

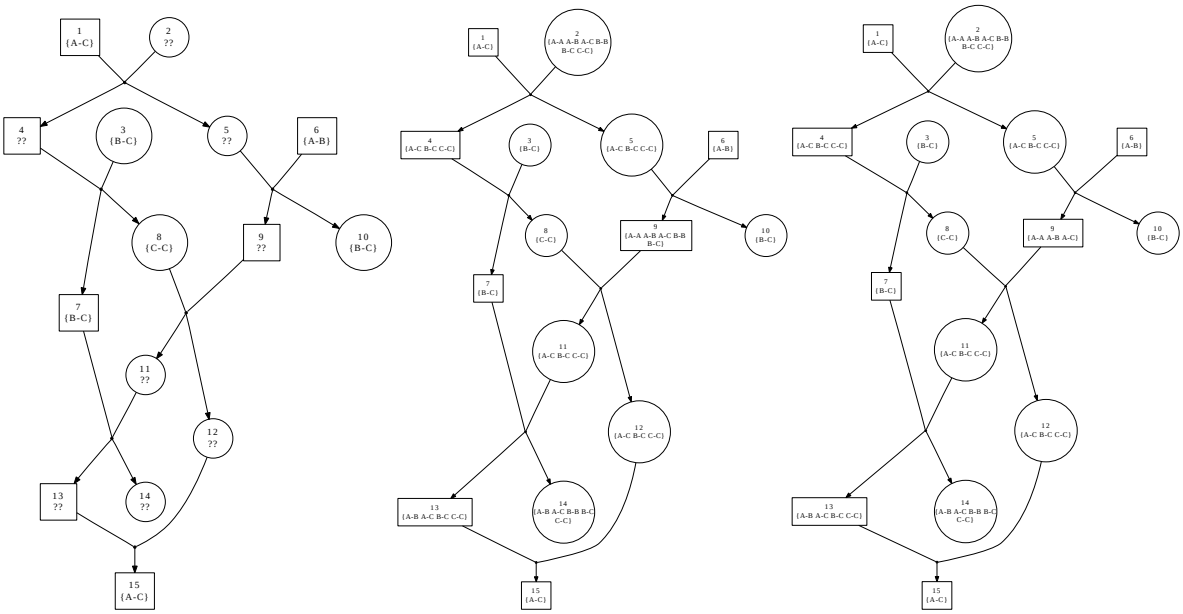
A psmap H is consistent on S if all $g \in H(i)$, for all $i \in I$, are consistent on S.

A pedigree consistency algorithm can be seen as a function that takes a psmap and returns another psmap where some inconsistent genotypes have been removed. More precisely, we define function $\mathsf{filter}_S \colon \mathscr{M} \to \mathscr{M}$ such that $\mathsf{filter}_S(H) = H' \sqsubseteq H$ and H' is consistent on S.

We say that a psmap H on a pedigree $\langle I, f, m\rangle$ is *fixed* on a set $S \subseteq I$ if $H(i)$ is a singleton set for all $i \in S$.

Example 3. Let $i \in I$ be a non-founder in the pedigree $\langle I, f, m\rangle$ and assume the psmap H is fixed on $\{f(i), m(i)\}$. Thus $H(f(i)) = \{g_f\}$ and $H(m(i)) = \{g_m\}$. Let us compute $H' = \mathsf{filter}_{\{f(i),m(i),i\}}(H)$. Consider $G = H(i) \cap \mathsf{mate}(g_f, g_m)$. If $G \neq \varnothing$ then $H' = H[G/i]$, otherwise $H' = \bot$.

Let S and T be two regular subsets of I. We may want to obtain $\mathsf{filter}_{S\cup T}(H)$ from $\mathsf{filter}_S(H)$ and $\mathsf{filter}_T(H)$, which may be simpler to compute. A candidate composition is $\mathsf{filter}_S(H) \sqcap \mathsf{filter}_T(H)$, since



(a) (b) (c)

Figure 2: An example of the applications of the genotype elimination algorithms: the initial pedigree 2(a), after the application of Lange-Goradia algorithm 2(b), and after the application of O'Connell and Weeks 2(c). In the initial pedigree, we have marked with "??" the untyped individuals.

this operation keeps the genotypes which are consistent on both S and T. However, in general, we only have $\mathsf{filter}_{S\cup T}(H) \sqsubseteq \mathsf{filter}_S(H) \sqcap \mathsf{filter}_T(H)$, and the relation may be strict. Nonetheless, it can be easily seen that the equality holds whenever H is fixed on $S \cap T$.

A useful function in the definition of consistency check algorithms is function $\mathsf{split}_S \colon \mathscr{M} \to \wp(\mathscr{M})$. Given any $S \subseteq I$, $\mathsf{split}_S(H)$ is the set of all psmaps $F \sqsubseteq H$ such that F is equal to H on $I \setminus S$ and is fixed on S. Thus, if $S = \{x_1, \ldots, x_n\}$, then for each $(g_1, \ldots, g_n) \in H(x_1) \times \cdots \times H(x_n)$ we have a psmap $F \in \mathsf{split}_S(H)$ such that $F(x_i) = g_i$ for all $1 \leqslant i \leqslant n$ and $F(x) = H(x)$ for all $x \notin S$. If $P = \langle I, f, m\rangle$ is a pedigree and H is a psmap on it, we have the following relation for all $T, S \subseteq I$ (where S is regular)

$$\bigsqcup_{F \in \mathsf{split}_T(H)} \mathsf{filter}_S(F) = \mathsf{filter}_S(H). \tag{1}$$

2.1 The Lange-Goradia algorithm

The idea of the Lange-Goradia algorithm is to remove all the genotypes of an individual i that are inconsistent on any nuclear family to which i belongs. This is accomplished by looking at one nuclear family at a time. Let H be a psmap for a pedigree $\langle I, f, m\rangle$. If $S = \{x, y, k_1, \ldots, k_n\} \subseteq I$ is a nuclear family where x and y are the parents and $k_1, \ldots, k_n$ are the children, then each pair (g_x, g_y) of genotypes in $H(x) \times H(y)$ is examined in turn, checking that $\mathsf{mate}(g_x, g_y) \cap H(k_i) \neq \varnothing$ for all the children k_i with $i = 1, \ldots, n$. If this is the case, then g_x, g_y and all genotypes in $\mathsf{mate}(g_x, g_y) \cap H(k_i)$ for each children k_i are consistent on S. All genotypes that are not found to be consistent after all pairs of genotypes in $H(x) \times H(y)$ have been examined are certainly inconsistent on S and, thus, also inconsistent, so they can be safely removed. More formally, we can say that the algorithm computes $\bigsqcup_{F \in \mathsf{split}_{\{x,y\}}(H)} \mathsf{filter}_S(F)$ (note that a nuclear family is

a regular subset of I), which is equal to $\mathsf{filter}_S(H)$ according to (1). For each $F \in \mathsf{split}_{\{x,y\}}(H)$, $\mathsf{filter}_S(F)$ is computed as $\bigsqcap_{i=1}^{n} \mathsf{filter}_{\{x,y,k_i\}}(F)$. This is equal to $\mathsf{filter}_S(F)$ since F is fixed on $\{x,y\} = \bigcap_{i=1}^{n}\{x,y,k_i\}$. Finally, $\mathsf{filter}_{\{x,y,k_i\}}(F)$ is computed as in Example 3, for each $1 \leqslant i \leqslant n$.

The algorithm is iterated on all nuclear families until no new genotypes are removed. If H' is the psmap obtained at the end of the algorithm and $g \in H(i)$ for any $i \in S$, then g is consistent on all nuclear families to which i belongs. Let us call $\mathsf{LG}\colon \mathscr{M} \to \mathscr{M}$ the function that maps an input psmap H to the output psmap $\mathsf{LG}(H)$ according to the Lange-Goradia algorithm. In general, $\mathsf{filter}_I(H) \sqsubseteq \mathsf{LG}(H)$ and the relation may be strict, i.e., the algorithm may not eliminate all inconsistent genotypes. As shown by Lange and Goradia [6], a sufficient condition for $\mathsf{filter}_I(H) = \mathsf{LG}(H)$ is the absence of loops in the pedigree. As an example, consider the pedigree of Figure 2. The pedigree contains loops, since there are individuals that mate that have an ancestor in common (for instance individuals 12 and 13 are both descendant of individual 8). Therefore, it is not guaranteed that the result of Lange-Goradia (Figure 2(b)) contains only consistent genotypes. In fact, consider individual 9. Although the genotype (B,B) is not consistent, the Lange-Goradia algorithm cannot eliminate it. To see that it is not possible to find a Mendelian fsmap that is $\sqsubseteq$ of that depicted in Figure 2(a), consider individual 15. One of his alleles is A. Since his alleles must come from individuals 7, 8, and 9, at least one of those individuals must have allele A. Individuals 7 and 8 do not contain it, thus 9 must have A as allele, and we can eliminate (B,B). We will see in the next subsection that the O'Connell and Weeks algorithm is able to eliminate (B,B) from individual 9.

2.2 The O'Connell and Weeks algorithm

The O'Connell and Weeks algorithm [8] is able to remove all inconsistent genotypes from a psmap. The algorithm has the same input of the Lange-Goradia algorithm: a pedigree $P = \langle I, f, m\rangle$ and a psmap $H \in \mathscr{M}$. Let us call $\mathsf{OCW}\colon \mathscr{M} \to \mathscr{M}$ the function that maps an input psmap H to the output psmap $\mathsf{OCW}(H)$ according to the O'Connell and Weeks algorithm.

First, a suitable set $B \subseteq I$ of *loop breakers* is found. A loop breaker is an individual that is involved in a loop in the pedigree and set B must contain such an individual for each loop in the pedigree.

A new pedigree $\overline{P} = \langle I \cup \overline{B}, \overline{f}, \overline{m}\rangle$ is built, where $\overline{B}$ contains a new individual $\overline{b}$ for each $b \in B$, $\overline{f}$ is undefined for all $\overline{b} \in B$, is equal to f for all x such that $f(x) \notin B$, and $\overline{f}(x) = \overline{f(x)}$ for all $f(x) \in B$ (and similarly for $\overline{m}$). Thus, $\overline{P}$ is obtained from P by breaking all loops. Then, for each $F \in \mathsf{split}_B(H)$ a psmap $\overline{F}$ on $\overline{P}$ is built, where $\overline{F}(x) = F(x)$ for all $x \in I$ and $\overline{F}(\overline{b}) = F(b)$ for all $b \in B$. Finally, $\mathsf{LG}(\overline{F})$ is computed for all $\overline{F}$ and all output psmaps thus obtained are joined. Since $\overline{P}$ contains no loops, we have $\overline{F}' = \mathsf{LG}(\overline{F}) = \mathsf{filter}_{I\cup\overline{B}}(\overline{F})$ for all $\overline{F}$. It is easy to see that it is $\overline{F}'(b) = \overline{F}'(\overline{b})$ for all $b \in B$ and that this implies that the restriction of $\overline{F}'$ to I is consistent on I. Indeed, if F' is the restriction of $\overline{F}'$ to I we have $F' = \mathsf{filter}_I(F)$.

We note that there is no need to actually build pedigree $\overline{P}$, since $\mathsf{LG}(F)$ will produce the same result as $\mathsf{LG}(\overline{F})$ whenever F is fixed on B. Thus we can simply define

$$\mathsf{OCW}(H) = \bigsqcup_{F \in \mathsf{split}_B(H)} \mathsf{LG}(F). \tag{2}$$

For each $F \in \mathsf{split}_B(H)$ we have $\mathsf{LG}(F) = \mathsf{filter}_I(F)$, thus we obtain $\mathsf{OCW}(H) = \mathsf{filter}_I(H)$ from (1).

Fig. 3 shows a block-diagram representation of the O'Connell and Weeks algorithm. Note that eq.(2) corresponds to the part of the diagram from the split block onwards. The initial LG block is not necessary for the completeness of the algorithm, but is introduced in order to try to reduce the cost of the rest of

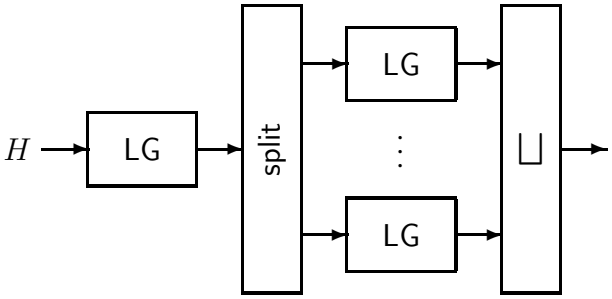


Figure 3: The O'Connell and Weeks algorithm.

the algorithm, since the number of Lange-Goradia invocations depends combinatorially on the number of genotypes assigned to each loop breaker.

As an example, consider the pedigree depicted in Figure 2(b). The pedigree contains various loops that can be broken, for instance, by choosing individuals 8 and 12 as loop breakers. This choice leads to three applications of the Lange-Goradia algorithm to the pedigree of Fig. 2(b) in which the individual 12 is typed as (A,C), (B,C), and (C,C), respectively. The three runs have as results the psmaps depicted in Figure 4. The union of these three psmaps gives as results the psmap depicted in Figure2(c). We can note that genotypes (B,B) and (B,C) have been eliminated from individual 9.

3 The Celer tool

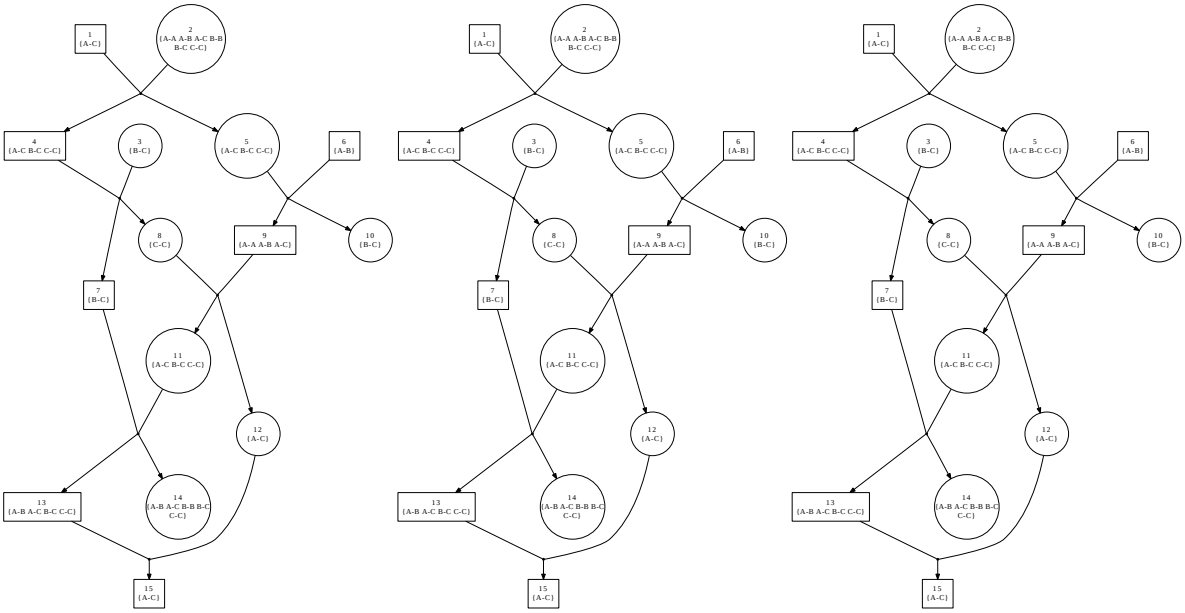
We have implemented the O'Connell and Weeks algorithm in a tool named Celer. Celer has been developed in C++ and is able to perform genotype elimination. Using a command-line switch, it is possible to select either Lange-Goradia or O'Connell and Weeks's algorithm. Celer receives as input a pedigree in pre-LINKAGE format, and writes the processed pedigree in a human-readable form. Moreover, it is also possible to have a DOT-file as output, that can be processed with Graphviz [3] to obtain a graphical representation of the resulting pedigree.

3.1 Parental information

In the design of our application, we kept the genotypic information separated from the parental information. During the parsing of the file, parental relations are stored in a redundant set of data structures (list of nuclear families in the pedigree, list of partners of each individual, list of families each individual belongs to, etc.). These data structures allow to recover all the parental relations needed by the consistency algorithms in a fast way. For instance, during the Lange-Goradia algorithm, to avoid unnecessary iterations, we set up a working list of the families to be processed. When the genotypes set of an individual changes, we insert in the working list only the families the individual belongs to.

3.2 Genotypes set as bitmaps

Our efficient implementation uses bitmaps to represent elements of $\wp(\mathscr{G})$ (individual of a psmap). When the set contains few genotypes, a bitmap needs more space than other alternatives such as binary search trees. On the contrary, this slight drawback is counter-balanced by many advantages. First of all, the



(a) (b) (c)

Figure 4: An example of the applications of the O'Connell and Weeks: from the pedigree in Fig.2(b) the individuals 8 and 12 are chosen as loop breakers, leading to three applications of the Lange-Goradia algorithms whose results are depicted in this figure.

operations of search, insertion and deletion from subset of $\mathscr{G}$ can be completed in constant time. Moreover, when the maximum number of alleles is known in advance, bitmaps can avoid the use of dynamic memory, thus speeding up the operations of copy and allocation/deallocation. Union and intersection of set of genotypes can be implemented with bitwise logical operations. Even the iteration of all the genotypes in a set can be implemented efficiently by calculating the least significant bit in a word.

We chose to represent alleles with unsigned integers in the range $[0, N-1]$, where N is the maximum number of alleles. With this choice, elements of $\wp(\mathscr{G})$ are triangular bitmaps with N rows. When $N = 32$, the n-th word of the matrix represents the subset of $\mathscr{G}$ composed by genotypes with n as the first allele, and $k <= n$ as the second allele. In this way, it is easy to build bit masks for manipulating sets of genotypes.

As an example, consider the optimization suggested in [8]. To speed up the initial application of the Lange-Goradia algorithm, O'Connell and Weeks suggest to pre-process the pedigree by removing those genotypes that can be easily identified as superfluous by looking at a single parent-child pair. For instance, when a child is fully specified with alleles (A,B), it is possible to remove from its parents all the genotypes that do not contain at least one from A and B. With the genotype set represented as a bitmap, it is sufficient to clear all the bits that are not in words A,B and in columns A,B. The `C++` code of this operation can be found in Figure 5.

Concluding, the bitmap has been a key choice for speeding up all the consistency algorithms.

3.3 Loop breakers selection

We have seen that the O'Connell and Weeks algorithm executes the Lange-Goradia algorithm once for every combination of the genotypes of the loop breakers. Therefore, the selection of loop breakers greatly

```
void bitmap::reduce_parent_child(int A, int B) {
  // A is always less or equal than B
  unsigned int allele_mask;
  if (A == B) { // homozygous individual
    allele_mask = (1 << A);
    unsigned int i;
    for (i=0; i<A;++i) {
      data[i] &= allele_mask;
    }
    i++; // leave A–th word untouched
    for (; i<32;++i) {
      data[i] &= allele_mask;
    }
  } else {
    allele_mask = (1 << A) | (1 << B);
    unsigned int i;
    for (i=0; i<A;++i) {
      data[i] &= allele_mask;
    }
    ++i; // leave A–th word untouched
    for (; i<B;++i) {
      data[i] &= allele_mask;
    }
    ++i; // leave B–th word untouched
    for (; i<32;++i) {
      data[i] &= allele_mask;
    }
  }
}
```

Figure 5: The C++ code for the optimization suggested by O'Connell and Weeks. When an individual is typed we remove from his/her children (and parents) the genotypes that do not contain at least one of his/her alleles. In the code, A and B are the alleles of the typed individual.

affects the total running time of the O'Connell and Weeks's algorithm. In Celer, we chose to apply the selection strategy suggested by Becker et al. [2]. The idea of the selection algorithm is to prefer to choose the individuals that break more loops at a time, and to avoid the ones that have a long list of genotypes. Becker et al. show that this problem is equivalent to the calculus of the minimum spanning tree of a directed graph. The graph to be analyzed can be obtained from the parental graph by removing all the individuals (and corresponding marriage nodes) that do not belong to any loop. This reduction of the graph must be put in place whenever a new loop breaker is chosen. The individuals in this graph are labelled with the result of a function f that estimates the cost of the selection of the corresponding loop breaker. The function $f\colon \mathscr{M} \times I \to \mathbb{Z}^+$ is defined as $f(H,i) = \log(\sharp H(i))/d(i)$, where $\sharp H$ denotes the cardinality of set H and $d(i)$ is the number of neighbours of individual i in the graph. The intended meaning of the function d is to be a heuristic estimate of the number of loop the individual belongs to. We implemented the spanning tree calculus with a modified version of the classical algorithm by Kruskal [5]. In fact, in this case, the function f (and in particular d) must be recalculated because the graph is reduced whenever a new loop breaker is found. However, since the cost of selection is only increasing, the greedy methodology of the spanning tree algorithm can be preserved.

It is easy to see that, by definition of split, given $S,T \in I$ and $H \in \mathscr{M}$, with $T \subseteq S$ and H fixed on T, it holds $\mathsf{split}_S(H) = \mathsf{split}_{S\setminus T}(H)$. Therefore, in the split phase, we discard all the loop breakers that have a single genotype.

3.4 Recursive vs non recursive reduction

To reduce the number of Lange-Goradia reductions (one for every combination of the genotypes of the loop breakers), O'Connell and Weeks suggest to use a recursive version of their algorithm. Instead of calculating all the combinations and applying the Lange-Goradia reduction, they adopt a backtracking methodology and execute a Lange-Goradia reduction whenever a loop breaker genotype is fixed. The algorithm can be expressed by the following pseudo-code. In the pseudo-code, given a function f, we denote with $f[x/y]$ the function f' defined as $f'(z) = f(z)$ if $z \neq x$, and y otherwise. This notation is used for updating the The rationale behind this approach is to avoid a brute-force exploration of the results of

Algorithm 1 The recursive version of the O'Connell and Weeks algorithm

```
1:  OCWR(P, B, H)
2:  if B = ∅ then
3:     return H
4:  else
5:     R ← ⊥
6:     select an individual i ∈ B
7:     for g ∈ H(i) do
8:        H' ← H[i/g]
9:        R ← R ⊔ OCWR(P, B \ i, LG(H'))
10:    end for
11:    return R
12: end if
```

the split function in (2). However, our experiments show that this approach does not pay off when coping with large pedigrees and few combinations to explore. In fact, all the psmaps that are on the recursion call stack must be initialized and copied, thus leading to an increased use of memory. When the number

Name	Individuals	Generations	%Founders	Avg Family size
HOPS	221	12	21.72%	1.52
APE	4921	15	3.23%	1.82
QMSIM	8420	10	4.99%	2.00

Table 1: The three benchmarks used

of individuals is not high and there are many combinations to explore, the recursive version is better than the non recursive one.

4 Performances of Celer

We have tested Celer with three different pedigrees. Following the methodology described in [10], we have simulated genetic data by picking founder alleles from the uniform distribution, applying randomly the Mendelian laws down the pedigree to calculate non-founder alleles, and, finally, deleting the genotype information of some individuals.

The first pedigree we considered is composed by 221 individuals. It is a human pedigree that traces the ancestors of two individuals affected by hypophosphatasia (HOPS). The pedigree comes from the Hutterite population living in North America, and it has been used previously in [7, 10].

We analyzed 100 datasets for each combination of the number of alleles (5, 7, 10, 12, 15, 17, 20, 25, 30), and of the ratio of untyped individual (5, 10, 20, 30, and 50 percent), for a total of 4500 datasets.

Then, we tested a larger pedigree composed by 4921 individuals. This pedigree was also studied in [10] and has been simulated with the method of Gasbarra et al. [4]. It has been used as a benchmark for the tool Allelic Path Explorer (APE). The pedigree contains 159 founders, and 75 percent of individuals were inbred. Again, simulating genetic data, we have created 100 datasets for each combination of number of alleles and each ratio of untyped individuals.

The last pedigree we tested is even bigger. It is composed by 8420 individuals and has been generated with the tool QMSIM [12]. It is composed of 10 generations. The founders are 420 individuals (400 females and 20 males). We have tested the performance of Celer on a Intel Core 2 Duo 3.00 GHz machine equipped with 2GB of RAM and running Ubuntu Linux 9.10 (kernel version 2.6.31-21).

Figure 6 shows the execution time of Celer when the Lange-Goradia algorithm is executed. We have put the number of alleles on the x axis and there is a line for every percentage of untyped individuals in the pedigree. Every dot in the graph refers to the average execution time of the 100 datasets for each combination number of alleles-ratio of untyped individual. We have used a logarithmic scale on the y axis, and therefore the linear trend corresponds to an exponential growth of the execution time when the number of alleles is raised. We can note that, even though the QMSIM pedigree is composed by a larger number of individuals than APE, the execution times are significantly lower. This could be due to its simple and regular parental structure (see Table 1 for a comparison). We have measured a very low variance among the same 100 datasets, except when the number of alleles is high and the percentage of untyped individual is set to 50%. This effect is particularly evident in benchmark APE. We reported in Figure 6(d) the mean, and the first three quartiles of the execution times of Celer, when the ratio of untyped individuals is 50% and the alleles are between 20 and 30.

We have also tested the same benchmarks when Celer executes the O'Connell and Weeks algorithm. However, in many cases, the loop breakers selection algorithm is able to find only loop breakers that have a single genotype. In this case, as we have seen in Section 3.3, the O'Connell and Weeks algorithm

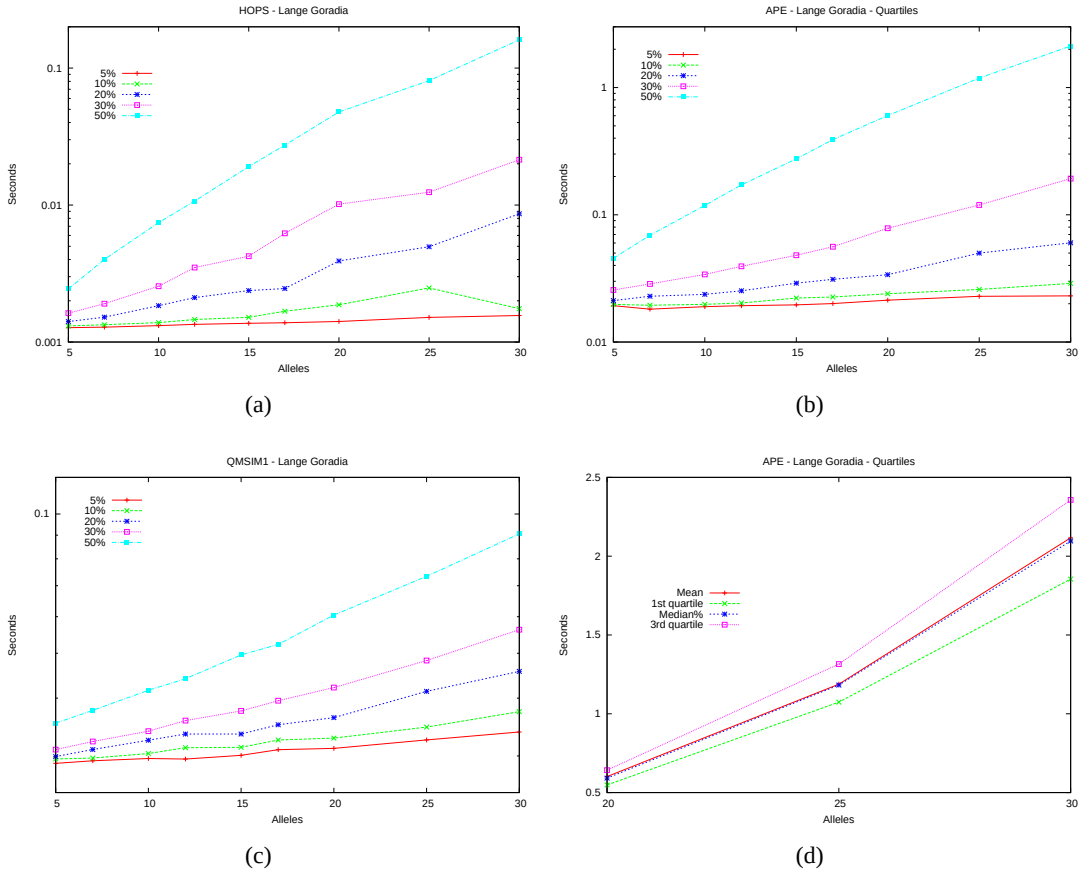
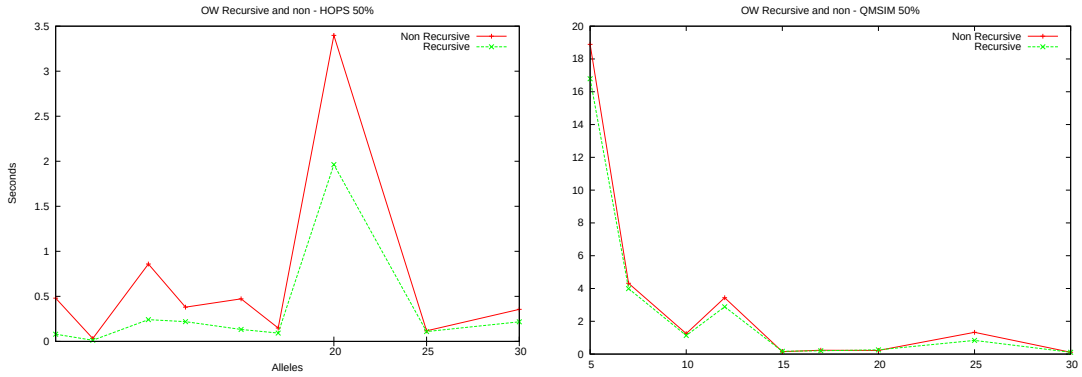


Figure 6: The performance of Celer when the Lange-Goradia algorithm is applied: HOPS 6(a), APE 6(b), and QMSIM 6(c). In 6(d) we show the quartile for the benchmark APE when the ratio of untyped individuals is set to 50%, where we noticed a significant variance.

Benchmark	% unknown	Avg LB	Max LB	Avg Cases	Max Cases
HOPS	<50%	0.0200	2	0.04	10
	50%	1.7622	10	686*	$4.66 \cdot 10^6$
QMSIM	<50%	0.1175	6	0.18	240
	50%	2.6978	19	955*	$2.24 \cdot 10^8$
APE	<50%	0.1175	4	0.19	32
	50%	8.398	172	$4.02 \cdot 10^{67}$	$3.61 \cdot 10^{70}$

Table 2: The number of loop breakers and the number of cases generated by the split functions. The mean marked with (*) have been calculated excluding testcases with combinatorial explosion (4 for HOPS, 8 for QMSIM).



(a) (b)

Figure 7: The execution times of the O'Connell and Weeks algorithm for HOPS 7(a), and QMSIM 7(b), when only half of the individuals in the pedigree are typed.

is equivalent to the Lange-Goradia. With a low rate of untyped individuals, the number of loop breakers (from now on we consider only the loop breakers with more than one genotype) is different from zero only in some sporadic cases, and thus the average execution time of the O'Connell and Weeks is very similar to the Lange-Goradia one (the only difference being the loop breaker selection procedure). We report in Table 2 the average number of loop breakers and the number of cases generated by the split function. As shown in the table, there is the risk of a combinatorial explosion. When the ratio of unknown individuals has been set to 50%, we could not complete the O'Connell and Weeks analysis within 30 minutes of computations for 3 out of the 900 pedigrees of the HOPS benchmark and 8 out of 900 of the QMSIM benchmark, and for all the pedigrees of the APE benchmark.

In Figure 7 we have plotted the average executions of the O'Connell Weeks algorithm run on the benchmarks HOPS and QMSIM when only half of the individuals are typed. We can note that the recursive version of the algorithm dominates the non-recursive version. However, the gap between the twos is very small in QMSIM, due to the overhead of the backtracking procedure that nearly counterbalances the advantage of executing fewer Lange-Goradia iterations.

5 Comparison with other software

O'Connell and Weeks have implemented their algorithms in the Pedcheck program [9]. Pedcheck is able to check Mendelian consistency in pedigree with different levels of accuracy (and therefore with different computational requirements). Level 1 analysis is able to discover simple errors related to a single nuclear family (a child and parent's alleles are incompatible, more than 4 alleles in a sibship, or 3 if there is a homozygous child). Level 2 correspond to Lange-Goradia algorithm. Level 3 and 4 provide a basic support to error correction. Level 3 identifies the so-called critical genotypes (that is the individuals that, if left untyped, make the pedigree consistent). Level 4 requires to know the frequencies of the alleles to estimate the most probable corrections.

At this time Celer is more precise than Pedcheck as regards to genotype elimination, but it does not offer error correction capabilities. Celer is more precise because it can also perform O'Connell and Weeks algorithm that we have seen is more precise than the Lange-Goradia algorithm. Moreover, when Pedcheck is applied to large pedigrees, even the Level 2 (Lange-Goradia) phase, takes a considerable amount of time. For example, consider the QMSIM benchmarks (8420 individuals and 4000 families). Even with only 10% of untyped individuals and 5 alleles, Pedcheck needs about 10 minutes of computation, while our program executes the Lange-Goradia algorithm in less than 20 milliseconds. We performed the same tests that we used on our tool and we found that Pedcheck could complete the analysis in times comparable with ours only on the HOPS benchmark . We report in Figure 8 the average execution times of Celer (with the Lange-Goradia algorithm) and Pedcheck (level 2 analysis) for the HOPS benchmarks and ratio of untyped individuals varying from 10 to 50%. We can see that Celer always outperforms Pedcheck.

Mendelsoft [11] is another tool that is able to check Mendelian consistency and perform error correction. Sanchez et al. model the Mendelian consistency problem with soft constraint networks and use a generic weighted constraint network (WCN) solver. In this way, they are not limited to a single error and can also correct pedigree with multiple errors. They evaluate their tool with random and real pedigrees composed of thousands of individuals and containing many errors. Even if we cannot directly compare Mendelsoft with Celer (that does not have error correction capabilities), we can note that the memory requirements of the WCN solver are very high. We have tested Mendelsoft with a machine equipped with 2GB of RAM and in many cases the program crashed because the amount of virtual memory was not sufficient. In particular, for the HOPS pedigree, Mendelsoft do not complete with this amount of RAM when the number of alleles is above 12.

6 Conclusions and future works

We have described the design and implementation of Celer, a program that performs genotype elimination. The design of the program has been aided by a formal description of the problem that highlighted the critical aspects of the algorithms and helped us to find the best data structures. We have measured the performances of the program and we have found that Celer is able to cope with large pedigrees. In the future, we would like to improve the working list selection algorithm of the Lange-Goradia elimination procedure and to test different loop breakers selection algorithms on highly-looped pedigrees, such as the one found in the APE test cases.

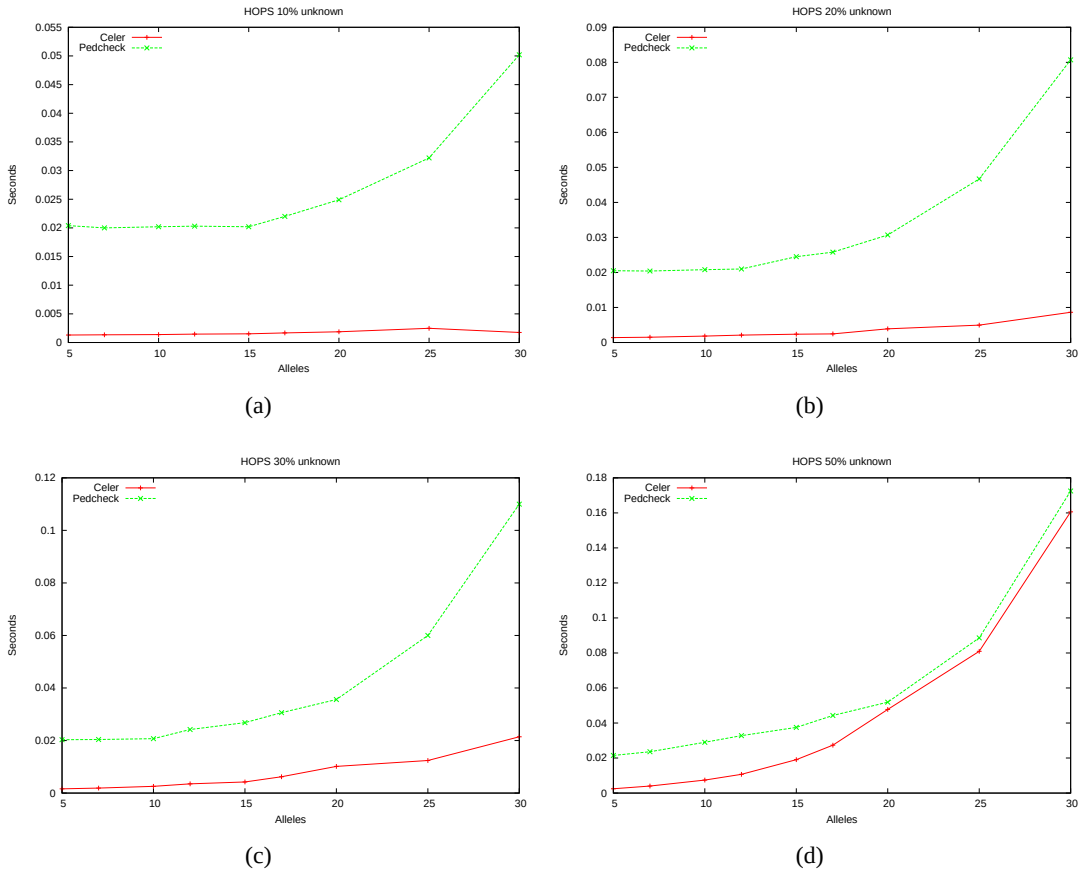


Figure 8: Comparison with Pedcheck

References

[1] Luca Aceto, Jens A. Hansen, Anna Ingólfsdóttir, Jacob Johnsen & John Knudsen (2004): *The complexity of checking consistency of pedigree information and related problems*. *J. Comput. Sci. Technol.* 19(1), pp. 42–59.

[2] Ann Becker, Dan Geiger & Alejandro A Schäffer (1998): *Automatic selection of loop breakers for genetic linkage analysis*. *Human heredity* 48(1).

[3] J. Ellson, E.R. Gansner, E. Koutsofios, S.C. North & G. Woodhull (2003): *Graphviz and Dynagraph – Static and Dynamic Graph Drawing Tools*. In: M. Junger & P. Mutzel, editors: *Graph Drawing Software*, Springer-Verlag, pp. 127–148.

[4] Dario Gasbarra, Mikko J. Sillanpää & Elja Arjas (2005): *Backward simulation of ancestors of sampled individuals*. *Theoretical Population Biology* 67(2), pp. 75 – 83. Available at `http://www.sciencedirect.com/science/article/B6WXD-4F6F67C-1/2/134b19fb4e742340bb5b97813e0308b8`.

[5] Jr. Kruskal, Joseph B. (1956): *On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem*. *Proceedings of the American Mathematical Society* 7(1), pp. 48–50. Available at `http://www.jstor.org/stable/2033241`.

[6] K. Lange & T.M. Goradia (1987): *An Algorithm for Automatic Genotype Elimination*. *Am. J. Human Genetics* 40, pp. 250–256.

[7] Y. Luo & S. Lin (2003): *Finding starting points for Markov chain Monte Carlo analysis of genetic data from large and complex pedigrees*. *Genet. Epidemiol.* 25(1), pp. 14–24.

[8] J. R. O'Connell & D. E. Weeks (1999): *An optimal algorithm for automatic genotype elimination*. *Am. J. Human Genetics* 65(6), pp. 1733–1740.

[9] Jeffrey R. O'Connell & Daniel E. Weeks (1998): *PedCheck: A Program for Identification of Genotype Incompatibilities in Linkage Analysis*. *The American Journal of Human Genetics* 63(1), pp. 259 – 266. Available at `http://www.sciencedirect.com/science/article/B8JDD-4R1WP1V-17/2/b556d7a79c50d44c4f200e65a4eac506`.

[10] Matti Pirinen & Dario Gasbarra (2006): *Finding Consistent Gene Transmission Patterns on Large and Complex Pedigrees*. *IEEE/ACM Trans. Comput. Biol. Bioinformatics* 3(3), pp. 252–262.

[11] Marti Sanchez, Simon Givry & Thomas Schiex (2008): *Mendelian Error Detection in Complex Pedigrees Using Weighted Constraint Satisfaction Techniques*. *Constraints* 13(1-2), pp. 130–154.

[12] Mehdi Sargolzaei & Flavio S. Schenkel (2009): *QMSim: a large-scale genome simulator for livestock*. *Bioinformatics* 25(5), pp. 680–681. Available at `http://dx.doi.org/10.1093/bioinformatics/btp045`.

Modelling of Multi-Agent Systems: Experiences with Membrane Computing and Future Challenges

Petros Kefalas and Ioanna Stamatopoulou
Dept. of Computer Science CITY College, Thessaloniki, Greece
International Faculty of the University of Sheffield
{kefalas, istamatopoulou}@city.academic.gr

Formal modelling of Multi-Agent Systems (MAS) is a challenging task due to high complexity, interaction, parallelism and continuous change of roles and organisation between agents. In this paper we record our research experience on formal modelling of MAS. We review our research throughout the last decade, by describing the problems we have encountered and the decisions we have made towards resolving them and providing solutions. Much of this work involved membrane computing and classes of P Systems, such as Tissue and Population P Systems, targeted to the modelling of MAS whose dynamic structure is a prominent characteristic. More particularly, social insects (such as colonies of ants, bees, etc.), biology inspired swarms and systems with emergent behaviour are indicative examples for which we developed formal MAS models. Here, we aim to review our work and disseminate our findings to fellow researchers who might face similar challenges and, furthermore, to discuss important issues for advancing research on the application of membrane computing in MAS modelling.

1 Multi-Agent Systems and Formal Methods

Software artefacts are characterised as agents if they can exhibit autonomous, reactive, proactive and social behaviour [37]. Autonomy is a property that allows agents to carry out their own thread of computation, without (much) intervention. Reactivity is not classified as an intelligent behaviour, however, it is essential to provide immediate response to the percepts from the environment. Sometimes, reactivity alone is more than enough to develop an emergent behaviour of a system [5]. The operation of intelligent agents is driven by goals that are achieved through a sequence of actions planned. Such goal-oriented (proactive) behaviour often involves a rather complex deliberation process. Finally, agents are able to communicate with other agents, a behaviour which leads to interaction between agents.

Multi-agent systems (MAS) consist of independent agents that can collaborate, negotiate, compete etc. towards the achievement of personal or shared goals. MAS are rather complex, highly interactive, highly parallel and highly dynamic systems. Agents play different roles in a MAS but they need to exchange and share information and knowledge in order to engage in a common problem solving activity. This, apart from the need for certain communication and interaction protocols, requires an effective organisation between agents. Organisation in a MAS, such as agent roles, communication structure, number of participating agents etc., is not static; it changes all the time throughout its operation. These dynamics make MAS a challenging software development activity. The more complex a MAS is, the more difficult the modelling process turns out to be and, in consequence, the less easy it is to ensure correctness at the modelling and implementation level. Correctness implies that all desired safety properties are verified at the end of the modelling phase and that an appropriate testing technique is applied to prove that the implementation has been built in accordance to the verified model [13].

To appear in:
AMCA-POP 2010 – Electronic Proceedings in Theoretical Computer Science (EPTCS)

In software engineering the term formal methods is used to classify mathematical approaches to all stages of software development. The main arguments in favour of formal methods are rigour, expressiveness and the ability to reason. The latter led to the promise of delivering correct software, i.e. software that is developed based on formal specifications and proofs (verified and tested), such that it performs in a desired manner under all circumstances. There is a dispute whether formal methods have delivered what they promised, i.e. correctness, but no one can argue that research and practice have shown a considerable number of successes.

With MAS the issue of correctness is much more complicated, since MAS are open systems, often exhibiting unpredictable emergent behaviour, and organised around a rather complex structure, with agents that intensively communicate, continuously change roles, interact etc. Therefore, formal modelling and verification that lead to implementation, testing, and simulation are challenging issues in MAS.

In our area of interest, formal modelling is particularly appealing as it raises many issues that cannot be tackled in a straightforward manner and leave many open challenges. More specifically we have been investigating, among others, the suitability of classes of Membrane Computing systems [28] as means of formal modelling of agents and MAS. We have mainly focused our efforts to developing models for biological systems with emergent behaviour or biology inspired systems. In this paper we record our research experience on formal modelling for MAS. We review the last decade work, by describing problems and their solutions. This is aimed at disseminating our findings to fellow researchers who might face similar challenges. We also focus on important issues for advancing research on formal methods in MAS further.

2 Case studies for Multi-Agent Systems

During the last years we have been researching on the formal modelling of MAS. Before we start describing our cumulative experience, we will briefly summarise the kind of MAS we have been dealing with.

One class of MAS that was thought to be of particular interest was the biological systems and mostly systems of social insects. Colonies of ants and bees as well as cell grouping fall within this category [35; 17; 23; 20; 11]. Both consist of relatively simple reactive agents that if left alone there is nothing much they can do. Organised as colonies, however, with roles and direct or indirect communication exhibit an emergent intelligent behaviour. For example, Pharaoh ants [35; 17], apart from the pheromone trail they produce to be used for foraging, they have a very effective way of organising their work within the society as well as help each other to survive by exchanging food. Japanese bees [23] contribute as individuals to increase the temperature in a hive and burn the attacking hornet. Foraging bees communicate the destination of the food source by the famous dance of the returning bee [11]. Similar emergent behaviour but with less direct communication among agents can be achieved in flocks, schools and herds [31; 27].

The other class of MAS that we have been modelling are those characterised as biology-inspired. Such a system is NASA ANTS [8], a MAS which deploys unmanned spacecrafts with a variety of specialisations to explore asteroids. More particularly, spacecrafts are organised in groups, each one consisting of a leader, multiple workers, and one or more messengers. Workers are responsible for gathering measurements, messengers for coordinating communication among all involved spacecrafts, and leaders for gathering measurements from workers, setting goals and coordinating group formations. We chose ANTS because it provides a good testbed for applying formal methods [29]. The important feature of ANTS is that there is a rather strict organisation which however is not affected even though there might be a large number of spacecrafts out of order or destroyed [33]. Robustness is a property of

all swarm intelligence systems.

3 Key issues in the modelling of agents and multi-agent systems

Individual agents operate based on the following:

- they perceive their environment by receiving stimuli as input which they filter and accept for further processing;
- they receive messages from other agents;
- they update their beliefs based on both the percepts as well as the information encoded in the received messages, by revising their temporary knowledge about the environment and others;
- they react based on a specific set of rules that describe individual behaviours;
- they engage in a deliberation process, which allows them to revise their goals and plans, and decide what is the next action to be performed;
- they compile and send messages to other agents;
- they act and the effects of their action appear in the environment.

Not all the above are present in every agent. For example, reactive agents do not deliberate, while "smarter" proactive agents do. Also, communication between simple biological agents is rather primitive and mostly done through the environment, in contrast to more elaborated direct communication that may follow a strict protocol. Therefore, in order to create a model of an agent, one would require:

- non-trivial data structures, e.g. set of n-tuples, sequences, lists, terms, with a set of their corresponding operations, to represent beliefs, goals, plans, messages, percepts etc.;
- means of encoding rules that express reactive behaviour, which can also be arranged in a strict order, e.g. avoid collisions, follow trail, forage;
- means of encoding the functionality that corresponds to their proactive behaviour (if such must be present), such as revision of goals and plan generation;
- representation of the internal state of the agent.

In a multi-agent system:

- each agent operates in parallel with others;
- the mode of interaction imposes the way in which agents exchange messages;
- the roles and organisation define the structure of the communication flow;
- new agents may come into play while other cease to exist.

At MAS level, modelling would require:

- ways to deal with exchange of messages between agents, either direct or indirect through the environment, deterministic broadcast, peer-to-peer or non-deterministic etc.;
- a way to express the interaction with the environment, that is, perception and action;
- a method for expressing the asynchronous computation of individuals;
- the addition and removal of agent instances "on the fly";

- means for structuring and restructuring the organisation "on the fly" (structure mutation).

If the modelling method used is formal, then there are a number of consequences that accompany this choice. First of all, formal reasoning on the model can be performed. This can be through formal verification, either proofs or model checking. Formal verification [10] will check whether desired properties of individual agents, or ideally of the whole system, stand. This is rather crucial before someone proceeds with implementation. Secondly, one can employ formal testing techniques. A set of test cases can be produced from the model to check whether the implementation is correct with respect to the model [16]. Thirdly, if the formal method is accompanied by tools, prototype animation or simulation may be possible [3]. This would facilitate the identification of misconceptions in the model which can then be fixed before proceeding to the implementation. Finally, a set of refinement transformations could safely lead to an implementation of the system that matches the original specification and model.

Table 1: Comparison of features of CXS, tPS, PCol and PPS with respect to modelling.

Modelling feature	**CXS**	**tPS**	**PCol**	**PPS**
Individual Agents				
Agent internal state representation	√	×	×	×
Rules to describe reactive behaviour	√	×	√	×
Rules to describe proactive behaviour	×	×	×	×
Non-trivial data structures for beliefs, goals, messages, stimuli etc	√	×	×	×
Formal verification of individual agents	√	×	×	×
Test case generation for individual agents	√	×	×	×
Communication				
Direct communication and message exchange	√	√	×	×
Non-deterministic communication	×	√	√	√
Indirect communication through the environment	×	√	√	√
Environmental stimuli (input)	√	√	√	√
Perception	×	×	×	×
MAS Structure				
Definition of agent roles	√	√	×	√
Addition of agent instances on the fly	×	×	×	√
Removal of agent instances on the fly	×	×	×	√
Communication network restructuring	×	×	×	√
MAS Operation				
Maximal parallelism	√	√	√	√
Arbitrary parallelism	√	√	√	√
MAS verification and testing	×	×	×	×
Tool support	√	×	×	√
Environment				
Modelling of the environment	×	√	√	√

In our experimentation, we tried out two types of formal methods, namely state-based methods and membrane computing. For the modelling process we have investigated a number of instances of those methods, such as X-Machines (XM) [9; 12] and Communicating X-Machines (CSX) [21] for the former, as well as tissue P systems (tPS) [26], P Colonies (PCol) [24] and Population P Systems [4] with active cells (PPS) for the latter. Table 1 shows a comparison between all methods, as to whether they can satisfy —directly, not through implicit means— the key issues in MAS modelling mentioned above. The comparison refers to the most widely used definitions of the models. There are actually numerous extensions that one way or another try to enhance the existing definitions with additional features. It should also be noted that X-Machines are not included in the comparison as an X-Machine model may only represent a single agent whereas in the table we compare formalisms that may be used for the modelling of MAS.

4 Methods employed for modelling of MAS

After carefully considering the aforementioned alternatives and based on the comparison presented above we selected to work with Communicating X-Machines and Population P Systems with active cells, and attempted a number of modelling approached for all the MAS mentioned above, e.g. biological cells, flocks, ants, Pharaoh ants, foraging bees, Japanese bees, and NASA ANTS. The reason for selecting CXMs is due its advantages in regards to the modelling of an individual agent's behaviour. Out of the three membrane computing formalisms we selected PPSs with active cells due to the fact that they best support operations on the MAS structure, such as addition and removal of agents as well as communication network restructuring.

A Communicating X-Machine System $CXMS = (XM_i, R),\ 1 \leq i \leq n$ is a collection of n X-machines XM_i able to communicate through channels, as they defined in the communication relation R. More particularly, an XM_i is a deterministic stream X-machine [13] defined as follows:

$$X = (\Sigma, \Gamma, Q, M, \Phi, F, q_0, m_0)$$

where:

- Σ and Γ are the input and output alphabets, respectively.
- Q is the finite set of states.
- M is the (possibly) infinite set called memory.
- Φ is a set of partial functions φ; each such function maps an input and a memory value to an output and a possibly different memory value, $\varphi : \Sigma \times M \rightarrow \Gamma \times M$.
- F is the next state partial function, $F : Q \times \varphi \rightarrow Q$, which given a state and a function from the type Φ determines the next state. F is often described as a state transition diagram.
- q_0 and m_0 are the initial state and initial memory respectively.

Note that the definition of an XM_i that belongs to a communicating system is slightly different, but for reasons of exposition it is not appropriate to elaborate here.

As an example consider the case of the Pharaoh ants. A more complete model of this case study may be found in [30] but it is partially included here for demonstration purposes.

The ants spend much of their in-nest time doing nothing, thus staying inactive. An ant may become active when its food reserves drop below a defined minimum threshold.

The assumptions that are made in this study are the following:

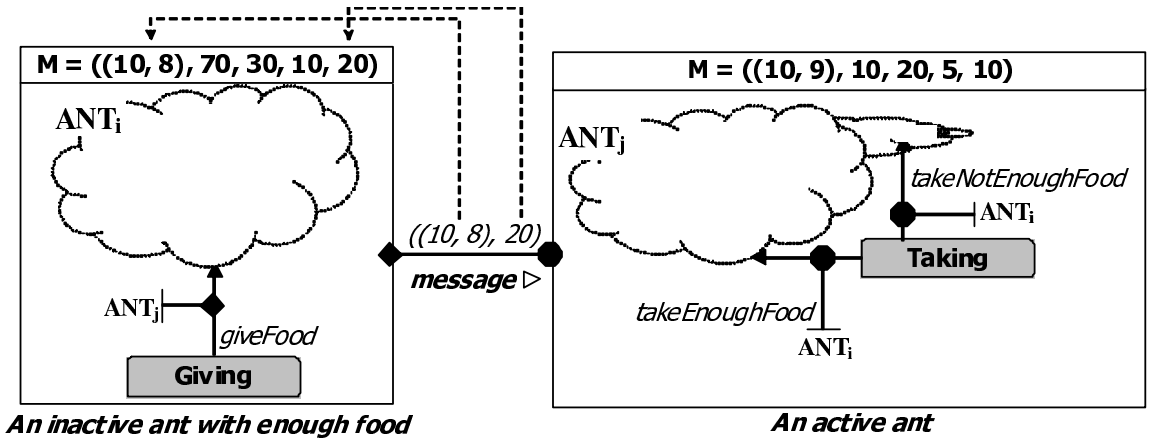


Figure 1: An example of a Communicating X-machine modelling the exchange of food between two ants.

- the colony only consists of workers;
- the nest, in which the colony is situated, is a rectangular environment (2D grid);
- the ants are either inactive or move around looking for food. If no food is found, they go outside the nest to forage and identify (new) locations for food;
- when two ants meet they might share food, if one is actively searching for food and the other has food reserves;
- the ants go out to forage when they do not have sufficient food reserves (according to the food quantity threshold), no food source is identified and a pheromone trail leading to an exit of the nest is discovered;
- ants that are outside may enter the nest at any time.

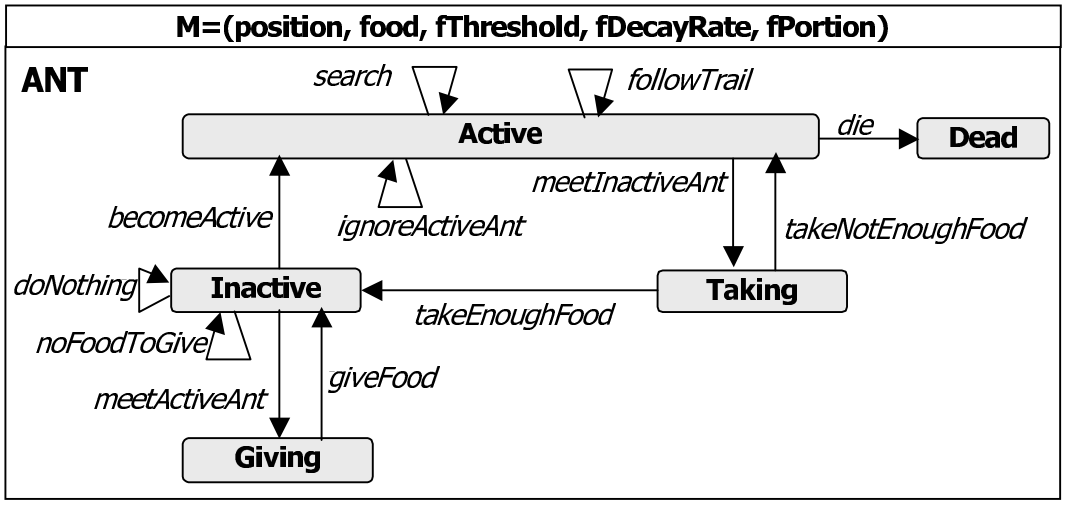


Figure 2: An example of an X-machine modelling a Pharaoh ant.

An example of a Communicating X-machine in regards to the above description is depicted in Fig. 1 and shows how two ants communicate by sharing food. The inactive ant's function *giveFood* sends as output the food amount it is willing to share to be received as input by the *takeEnoughFood* function of the active ant. Fig. 2 shows the state transition diagram of the XM model of one individual Pharaoh ant.

A Population P System with Active Cells [4] is defined as a construct:

$$P = (V, K, \gamma, \alpha, w_E, C_1, C_2, \ldots, C_n, R)$$

where:

- V is a finite alphabet of symbols called objects;
- K is a finite alphabet of symbols, which define different types for the cells;
- α is a finite set of bond-making rules of the general form $(t, x_1; x_2, p)$, with $x_1, x_2 \in V^*$, and $t, p \in K$;
- $\gamma = (\{1, 2, \ldots n\}, A)$, with $A \subseteq \{\{i, j\} \mid 1 \leq i \neq j \leq n\}$, is a finite undirected graph;
- $w_{\mathrm{E}} \in V^*$ is a finite multiset of objects initially assigned to the environment;
- $C_i = (w_i, t_i)$, for each $1 \leq i \leq n$, with $w_i \in V^*$ being a finite multiset of objects, and $t_i \in K$ the type of cell i;
- R is a finite set of:
  - communication rules of the form $r : (a; b, in)_t$, $r : (a; b, enter)_t$, $r : (b, exit)_t$, for $a \in V \cup \{\lambda\}$, $b \in V$, $t \in K$, which allow the moving of objects between neighbouring cells or a cell and the environment according to the cell type and the existing bonds among the cells;
  - object transformation rules of the form $r : (a \to b)_t$, for $a \in V$, $b \in V^+$, $t \in K$, meaning that an object a is replaced by an object b within a cell of type t;
  - cell differentiation rules of the form $r : (a)_t \to (b)_p$, with $a, b \in V$, $t, p \in K$ meaning that consumption of an object a inside a cell of type t changes the cell, making it become of type p. All existing objects remain the same besides a which is replaced by b;
  - cell division rules of the form $r : (a)_t \to (b)_t (c)_t$, with $a, b, c \in V$, $t \in K$ meaning that a cell of type t containing an object a is divided into two cells of the same type. One of the new cell has a replaced by b while the other by c;
  - cell death rules of the form $r : (a)_t \to \dagger$, with $a \in V$, $t \in K$ meaning that an object a inside a cell of type t causes the removal of the cell from the system.

An example of a Population P System modelling tumour growth is depicted in Fig. 3 (model description borrowed from the NetLogo models library [36]; a complete system definition using PPS may be found in [30]).

A tumour consists of two kinds of cells: stem and transitory cells. It is a stem cell that is required for the formation of the tumour to begin. At each time unit all cells divide thus doubling the size of the tumour. Stem cells may divide in two ways: (a) *asymmetrically*, thus breeding a transitory cell that moves outward, and (b) *symmetrically*, breeding another stem cell which also moves outward and settles in another location thus creating a metastasis of the original tumour. In effect, a stem cell never dies as during division one of the two daughter cells is always a stem cell.

Transitory cells divide only symmetrically up to a certain age; after that age they mature and eventually die. Finally, transitory cells that have originated from a metastatic stem cell, called metatransitory, as well as all their offsprings die younger.

Each cell type has its own objects and rules. For instance, in Fig. 3 C_6 is a *stem* cell with two rules according to the given example: a transformation rule that represents the cell's ageing by increasing it by 1, and a division rule that divides the cell in two creating a new cell in position pos' with an age of 0.

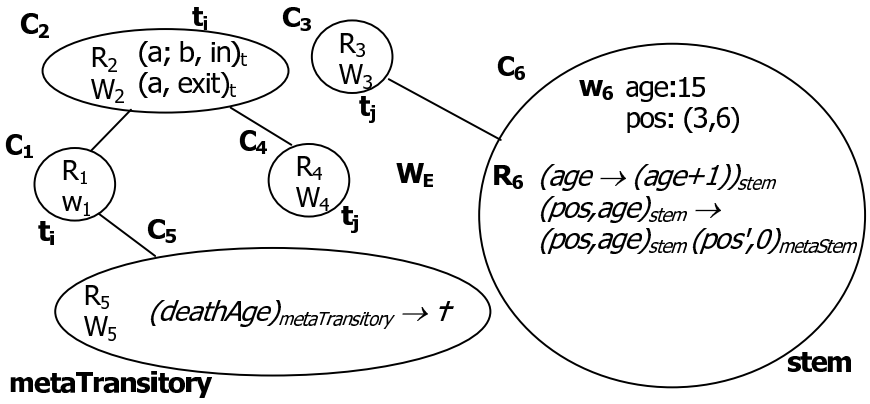


Figure 3: An example of a Population P System; R_i: set of rules related to cell C_i; w_i: multiset of objects associated to cell C_i.

5 Synthesised Methods and Transformations

It is apparent that the two types of methods (state-based and membrane) are complementary to MAS modelling needs, something which led us to some kind of synthesis of the two. We attempted a potential integration of the two as an instance of the *OPERAS* framework [30].

The *OPERAS* formal framework for defining a dynamic multi-agent system, is defined by the 6-tuple (O,P,E,R,A,S) where:

- O contains the reconfiguration operations (or rules, e.g. of the general form $condition \Rightarrow action$). Each operation involves the application of one or more of the operators that create or remove a communication channel between agents or introduce/remove an agent in/from the system;
- P is the distributed union of the percepts of all the types of agents involved in the system;
- the communication relation $R : A \times A$ with $(A_i,A_j) \in R$, $A_i,A_j \in A$ conveys the information that agents A_i,A_j can communicate by exchanging messages;
- E is a model of the environment;
- A is the set of agent instances $A = \{A_1,\ldots A_n\}$ where A_i is an agent instance defined in terms of (a) its individual behaviour, and (b) its local structural mutation mechanism for reconfiguring the system structure in its proximity;
- the set $S = \{(Behaviour_t, StrMut_t) \,|\, t \in Types\}$ holds the definitions of agent types (*Types* being a set of identifiers of the types of agents).

In OPERAS the behaviour of an agent can be modelled separately from its control. In principle, this means that one can employ two different formal methods for each, thus taking advantage of both state-based models and membrane computing ideas. It is therefore implied that there are several options which could instantiate OPERAS into concrete modelling methods. As mentioned above, we have long experimented with Communicating X-machines and Population P Systems with active cells, thus resulting into hybrid models such as $OPERAS_{CC}$ [34] and $OPERAS_{XC}$ [33]. The former uses PPS features for both modelling the dynamics and the behaviour of the agents, as is abstractly depicted in Fig. 4.

$OPERAS_{XC}$ on the other hand is a combination that uses state machines for the behaviour and PPS-like rules for the organisation of the system, as is abstractly depicted in Fig. 5. This gave us the opportunity to combine the advantages that XMs have in terms of modelling the behaviour of an agent with

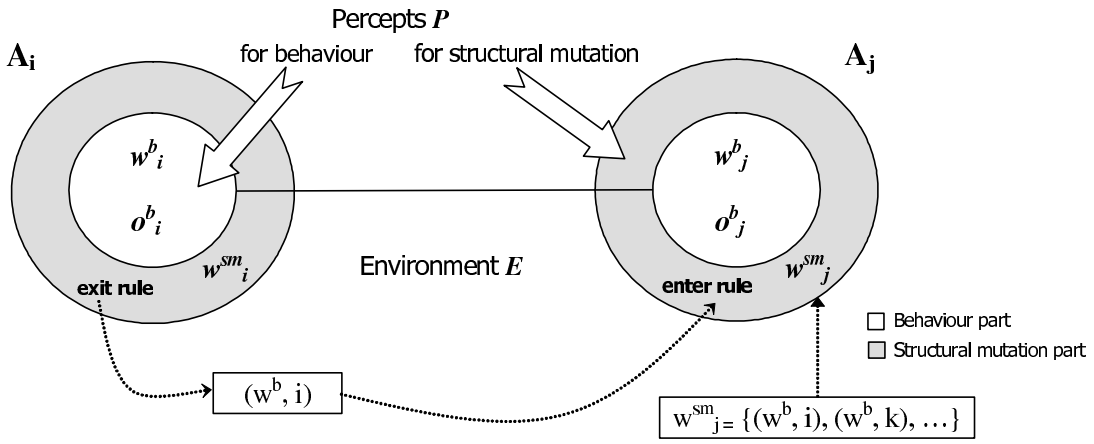


Figure 4: An abstract example of an $OPERAS_{CC}$ model that uses Population P System concepts for the representation of both the agent's behaviour and structure dynamics.

the advantages that PPSs have in terms of defining the control over the structure of the system. Also, the computation is driven by either method, which leads to a variety of interesting overall MAS computation. For a complete case study modelled using $OPERAS_{XC}$ the interested reader is referred to [30].

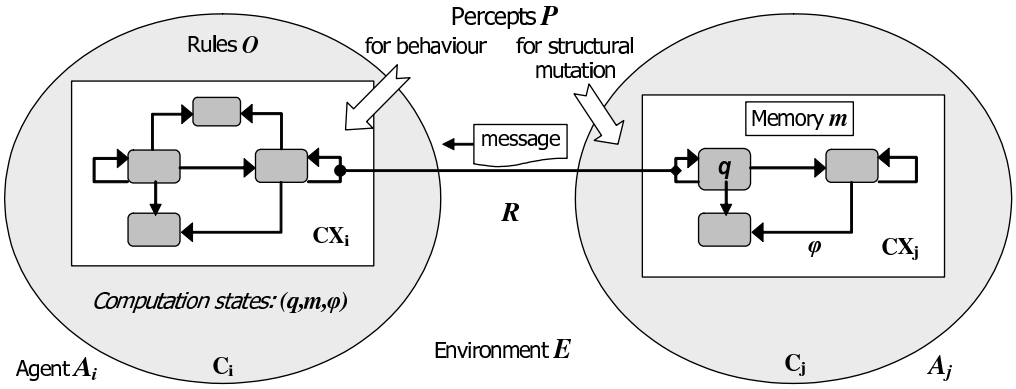


Figure 5: An abstract example of an $OPERAS_{XC}$ model that uses state machines for the behaviour and PPS-like rules for the organisation of the system.

We have also attempted a number of transformations between Communicating X-Machines and P Systems and vice-versa [19; 23]. Others in the P system research community showed more interest in other transformations, such as from P Systems to Petri nets [25], process algebra [6], cellular automata [7] etc. These have not only demonstrated the equivalence of these methods but created a temptation to try out several techniques known for one method to the other.

In the development of the MAS models, two notations have been created, namely X-Machine Definition Language (XMDL) [18] and Population P System Definition Language (PPSDL) [32]. The accompanied tools assisted us in understanding in depth the overall behaviour of the developed models through textual animation.

6 Discussion and Open Issues

There are several open issues that are left for further research on MAS development, namely formal modelling, verification, testing and tool support.

With respect to modelling, the effort to define a special class of P Systems should continue. The new class must employ all features that will facilitate agent and multi-agent system modelling. The computation within cells must be enhanced so that it can offer some complex reasoning required for goal-oriented agents. Such an attempt may inevitably restrict maximal parallelism for the sake of correctness of the behaviour of agents as well as of the whole system. An initial first attempt can be found at [22]. There is also room for development of the perception of the environment, since inserting an object into the cell may not be enough in realistic systems. MAS researchers would also like to see built-in features towards formal modelling of interaction between cells.

So far, there have not been useful results reported for verification of MAS models. Although verification of individual agents is possible, verification of the complete MAS seems like an insurmountable obstacle. The obvious reason is the combinatorial explosion problem due to the number of interactions that increase the state space in an exponential manner. It would be worth investigating whether there exist methods that work under specific harmless assumptions that could omit non-safety properties and reduce the search space. Finally, model checking techniques for P systems is an interesting area open for research developments. A direction towards model checking would be the automatic or semi-automatic translation of models into code, for model checkers such as SPIN [14] or SMV [2].

If formal verification seems hard to achieve, informal techniques, such as simulation have a lot to offer [1]. Suitable and correct transformations of formal models could lead to executable models that simulate MAS. In turn, simulation can facilitate the discovery of erroneous situations or undesired behaviour of the system. For numerous types of MAS, biology inspired included, visual animation is highly desirable [36; 35].

Finally, although there exist testing techniques that can identify all faults in the implementation of an individual agent developed based on state-based models, there is little work done towards the testing of membrane systems [15]. Inevitably, it would appear to be a challenge to invent equivalent techniques for the whole multi-agent system.

7 Conclusion

We have presented a review on the use of Population P Systems with active cells in Multi-Agent System modelling. In the process of developing formal models of MAS, we discovered a number of challenging issues that could partly be addressed by state based models and partly by membrane computing models. These characteristics were pinned down together with the available features of various methods that could make modelling possible. The synthesized solution gives space to a number of challenges, such as verification, testing and simulation. With this review, we attempted to disseminate our findings, initialise discussion that will set up directions of future research.

References

[1] *The P Systems Webpage — Software*. http://ppage.psystems.eu/index.php/Software.

[2] *SMV: Symbolic Model Verifier*. http://www-2.cs.cmu.edu/ modelcheck/smv.html.

[3] (2007). *FLAME: Flexible Large-scale Agent Modelling Environment*. http://www.flame.ac.uk/.

[4] F. Bernardini & M. Gheorghe (2004): *Population P Systems*. *Journal of Universal Computer Science* 10(5), pp. 509–539.

[5] R. A. Brooks (1991): *Intelligence Without Reason*. In: J. Myopoulos & R. Reiter, editors: *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91)*, Morgan Kaufmann publishers Inc., pp. 569–595.

[6] G. Ciobanu & B. Aman (2007): *On the relationship between membranes and ambients*. *BioSystems* 91(3), pp. 515–530.

[7] D. Corne & P. Frisco (2007): *Dynamics of HIV Infection Studied with Cellular Automata and Conformon-P systems*. *BioSystems* 91(3), pp. 531–544.

[8] S. Curtis, J. Mica, J. Nuth, G. Marr, M. Rilee & M. Bhat. (2000): *ANTS (Autonomous Nano Technology Swarm): An Artificial Intelligence Approach To Asteroid Belt Resource Exploration*. In: *Proceedings of 51st International Astronautical Congress*, International Austronautical Federation.

[9] S. Eilenberg (1974): *Automata, Languages and Machines*. Academic Press.

[10] E. A. Emerson & E. M. Clarke (1981): *Characterising correctness properties of parallel programs as fixpoints*. In: *Proceedings of the 7th International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science* 85, Springer-Verlag, New York, pp. 169–181.

[11] M. Gheorghe, M. Holcombe & P. Kefalas (2001): *Computational Models of Collective Foraging*. *BioSystems* 61, pp. 133–141.

[12] M. Holcombe (1988): *X-machines as a Basis for Dynamic System Configuration*. *Software Engineering Journal* 3(2), pp. 69–76.

[13] M. Holcombe & F. Ipate (1998): *Correct Systems: Building a Business Process Solution*. Springer-Verlag, London.

[14] G.J. Holzmann (2004): *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley.

[15] F. Ipate & M. Gheorghe (2009): *Testing non-deterministic stream X-machine models and P systems*. *Electronic Notes in Theoretical Computer Science* 227, pp. 113–126.

[16] F. Ipate & M. Holcombe (1997): *An Integration Testing Method that is proved to find all faults*. *International Journal of Computer Mathematics* 63(3), pp. 159–178.

[17] D. Jackson, F. Ratnieks & M. Holcombe (2004): *Coupled computational simulation and empirical research into the foraging system of Pharaoh's ants (Monomorium Pharaonis)*. *Biosystems* 76, pp. 101–112.

[18] E. Kapeti & P. Kefalas (2000): *A Design Language and Tool for X-machines Specification*. In: D. I. Fotiadis & S. D. Spyropoulos, editors: *Advances in Informatics*, World Scientific Publishing Company, pp. 134–145.

[19] P. Kefalas, G. Eleftherakis, M. Holcombe & M. Gheorghe (2003): *Simulation and Verification of P Systems through Communicating X-Machines*. *BioSystems* 70(2), pp. 135–148.

[20] P. Kefalas, G. Eleftherakis, M. Holcombe & I. Stamatopoulou (2005): *Formal Modelling of the Dynamic Behaviour of Biology-Inspired Agent-Based Systems*. In: M. Gheorghe, editor: *Molecular Computational Models: Unconventional Approaches*, Idea Publishing Inc., pp. 243–276.

[21] P. Kefalas, G. Eleftherakis & E. Kehris (2001): *Modular Modelling of Large-Scale Systems Using Communicating X-machines*. In: Y. Manolopoulos & S. Evripidou, editors: *Proceedings of the 8th Panhellenic Conference in Informatics*, Livanis Publishing Company, pp. 20–29.

[22] P. Kefalas & I. Stamatopoulou (2010): *Towards modelling of reactive, goal-oriented and hybrid intelligent agents using P Systems*. In: *11th International Conference on Membrane Computing.* Submitted.

[23] P. Kefalas, I. Stamatopoulou, I. Sakellariou & G. Eleftherakis (2009): *Transforming Communicating X-Machines into P Systems. Natural Computing* 8(4), pp. 817–832.

[24] J. Kelemen, A. Kelemenova & G. Paun (2004): *Preview of P colonies: A biochemically inspired computing model*. In: M. Bedau et al., editor: *Proceedings of the 9th Intern. Conference on the Simulation and Synthesis of Living Systems (Alife IX)*, pp. 82–86.

[25] J. Klein & M. Koutny (2007): *Synchrony and asynchrony in membrane systems*. In: H. J. Hoogeboom, G. Paun, G. Rozenberg & A. Salomaa, editors: *Membrane Computing, 7th International Workshop, Leiden, Holland, Lecture Notes in Computer Science* 4361, Springer, pp. 66–85.

[26] C. Martin-Vide, Gh. Păun, J. Pazos & A. Rodriguez-Paton (2003): *Tissue P Systems. Theoretical Computer Science* 296, pp. 295–326.

[27] O. Paunovski, G. Eleftherakis & A.J. Cowling (2008): *Framework for empirical exploration of emergence using multi-agent simulation*. In: S. Stepney, F. Polack & P. Welch, editors: *Proceedings of the 1st Workshop on Complex Systems Modelling and Simulation (COSMOS'08)*, pp. 1–31.

[28] G. Păun, G. Rozenberg & A. Salomaa (2010): *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., New York, NY, USA.

[29] C. Rouf, A. Vanderbilt, W. Truszkowski, J. Rash & M. Hinchey (2004): *Verification of NASA Emergent Systems*. In: *Proceedings of the 9th IEEE International Conference on Engineering Complex Computer Systems (ICECCS'04)*, pp. 231–238.

[30] I. Stamatopoulou (2008): *A Formal Framework for the Modelling of Multi-Agent Systems with Dynamic Structure*. Ph.D. thesis, University of Sheffield, UK.

[31] I. Stamatopoulou, M. Gheorghe & P. Kefalas (2005): *Modelling dynamic configuration of biology-inspired Multi-Agent Systems with Communicating X-machines and Population P Systems*. In: *Membrane Computing: 5th International Workshop, Lecture Notes in Computer Science* 3365, Springer, pp. 389–401.

[32] I. Stamatopoulou, P. Kefalas, G. Eleftherakis & M. Gheorghe (2005): *A modelling language and tool for Population P Systems*. In: *Proceedings of the 10th Panhellenic Conference in Informatics (PCI'05).*

[33] I. Stamatopoulou, P. Kefalas & M. Gheorghe (2007): *OPERAS: a Formal Framework for Multi-agent Systems and its Application to Swarm-based Systems*. In: A. Artikis, G. O'Hare, K. Stathis & G. Vouros, editors: *Proceedings of the 8th International Workshop on Engineering Societies in the Agents World (ESAW'07)*, pp. 208–223.

[34] I. Stamatopoulou, P. Kefalas & M. Gheorghe (2007): *OPERAS$_{CC}$: An instance of a Formal Framework for MAS Modelling based on Population P Systems*. In: G. Eleftherakis, P. Kefalas & G. Păun, editors: *Proceedings of the 8th Workshop on Membrane Computing (WMC'07)*, South-East European Research Centre, pp. 551–566.

[35] I. Stamatopoulou, I. Sakellariou, P. Kefalas & G. Eleftherakis (2008): *OPERAS for Social Insects: Formal Modelling and Prototype Simulation*. Special Issue of *Romanian Journal of Information Science and Technology (ROMJIST)* on *Natural Computing — from biology to computer science and back to applications* 11(3), pp. 267–280.

[36] U. Wilensky (1999). *NetLogo. http://ccl.northwestern.edu/netlogo. Center for Connected Learning and Computer-based Modelling. Northwestern University, Evanston, IL.*

[37] M. Wooldridge & N. R. Jennings (1995): *Intelligent Agents: Theory and Practice*. *Knowledge Engineering Review* 10(2), pp. 115–152.

A Process Calculus for Spatially-explicit Ecological Models (Extended Abstract)

Xenia Efthymiou Anna Philippou
Department of Computer Science
University of Cyprus

cs06ep@cs.ucy.ac.cy annap@cs.ucy.ac.cy

In this paper we propose PALPS, a Process Algebra with Locations for Population Systems. PALPS allows us to produce spatially-explicit, individual-based models and to reason about their behavior. Our calculus has two levels: at the first level we may define the behavior of an individual of a population while, at the second level, we may specify a system as the collection of individuals of various populations located in space, moving through their life cycle, traveling autonomously in space and interacting with each other in various ways such as preying on each other. We describe the syntax and the semantics of PALPS and we illustrate its applicability via simple examples.

1 Introduction

During the last years we have witnessed an increasing trend towards the use of formal frameworks for reasoning about biological as well as ecological systems, including process algebras [13, 14, 10], cellular automata [5] and *P*-systems [3]. Process algebras, first proposed in [12, 7], to aid the understanding and reasoning about concurrent systems, have proved to provide a number of features that make them amenable towards capturing biological processes. In particular, process algebras are especially suited towards the so-called individual-based approach of modeling populations as they enable one to describe the evolution of each individual of the population as a process and, subsequently, to compose a set of individuals (as well as their environment) into a complete ecological system. Features such as time, probability and stochastic behavior, which have been extensively studied within the context of process algebras, can be exploited to provide more accurate models, while associated analysis tools can be used to analyze and predict their behavior.

In this work, our aim is to introduce a process-algebraic framework equipped with the notion of a *location* to enable spatially-explicit modeling of ecological systems. In particular, we propose a domain-specific process algebra which associates individuals with information about their position and thus allows to explore location-dependent behavior of a population system. There exists a variety of existing work which introduces location behavior into formal frameworks. Amongst them, we mention [2, 8, 1] which introduce the concept of a location into frameworks developed for reasoning about biological processes, whereas relevant proposals that introduce locations in process algebras for reasoning about ad hoc networks can be found in [11, 6, 9]. The novelty of our proposal is that it associates location information with population-system specific behavior such as reproduction and preying. In the next section we present our process calculus and in the final section we conclude with remarks on future work.

2 The Process Calculus

In our calculus, PALPS (Process Algebra with Locations for Population Systems), we consider a system as a set of individuals operating in space, each possessing a species and a location identifier. Movement in the calculus is modeled via a specialized action whose effect is to change the location of an individual, with the restriction that the originating and the destination locations are neighboring locations. The notion of neighborhood is implemented via a relation **Nb** where $(\ell,\ell') \in \mathbf{Nb}$ exactly when locations ℓ and ℓ' are neighbors.

2.1 The Syntax

We continue to formalize the syntax of PALPS. We begin by describing the basic entities of the calculus. We assume a set of channels **Ch**, ranged over by a, b, as well as a set of locations **Loc** ranged over by ℓ, ℓ'. Furthermore, we assume a set of special labels **S** corresponding to the species under consideration, ranged over by $\mathbf{s}$, $\mathbf{s}'$. To model preying, we also assume the existence of a relation **Prey** $\subseteq \mathbf{S}\times\mathbf{S}$, where $(\mathbf{s},\mathbf{s}') \in \mathbf{Prey}$ if individuals of species $\mathbf{s}$ prey on individuals of species $\mathbf{s}'$.

Our calculus also employs a set of logical expression ranged over by e. One of our main aims being to facilitate reasoning about spatial-dependent behavior, these conditions are intended to capture environmental (location-relevant) situations which may affect the behavior of individuals. Since, in the present form of PALPS, locations are not associated with any environmental factors (e.g. temperature), the only useful properties individuals may observe concern the number of individuals of the same or another species co-existing within the same location. Thus, we consider expressions e, to be built using the logical connectives $\wedge$ and $\neg$ and the basic expressions $(\mathbf{s}@\ell) \bowtie c$, where c is a natural number and $\bowtie \in \{=,\leq,\geq\}$, the intention being that $(\mathbf{s}@\ell) \bowtie c$ expresses that the number of individuals of species $\mathbf{s}$ are equal to / less than / greater than c. We also write $@\ell \bowtie c$ to denote that the total number of individuals of all species are $\bowtie c$. We then write $S \models e$ for a population system S and an expression e, exactly when S satisfies e. The relation $\models$ is defined by induction on e in the natural way.

The syntax of PALPS consists of three levels: (1) the individual level (ranged over by P), (2) the species level (ranged over by R) and (3) the system level (ranged over by S). Their syntax is defined via the following BNF's

$$\begin{array}{lcl} P & ::= & \mathbf{0} \mid \eta.P \mid P_1 + P_2 \mid \ \mathsf{cond}\ (e_1 \rhd P_1, \ldots, e_n \rhd P_n) \mid C \\ R & ::= & !a.P \\ S & ::= & \mathbf{0} \mid P{:}[\![\mathbf{s},\ell]\!] \mid R{:}[\![\mathbf{s}]\!] \mid S_1 \,|\, S_2 \mid S\backslash L \mid [S] \end{array}$$

where $L \subseteq \mathbf{Ch}$, C ranges over a set of process constants $\mathscr{C}$, each with an associated definition of the form $C \stackrel{\text{def}}{=} P$, where the node P may contain occurrences of C, as well as other constants, and

$$\eta ::= a \mid \overline{a} \mid \mathit{move} \mid \mathit{prey} \mid \surd.$$

Beginning with the *individual* level P, the $\mathbf{0}$ process represents the inactive individual. $\eta.P$ describes the individual who first engages in activity η and then behaves as P. Activity η can be an (input) action on a channel a, written simply as a, a complementary (output) action on a channel a, $\overline{a}$, a movement action, *move*, a preying action, *prey*, or a time-passing action, $\surd$. Actions of the form a, and $\overline{a}$, $a \in \mathbf{Ch}$, are arbitrary actions performed by an individual e.g. *eat*, and they are also used to model reproduction. A $\surd$ action measures a tick on a clock and is used to separate the phases/rounds of an individual's behavior. Essentially, given a system, the intention is that in any given time unit all individuals perform their

available actions, possibly synchronizing as necessary, until they synchronize on their next $\surd$ action. $P_1 + P_2$ represents the nondeterministic choice between P_1 and P_2. The conditional process $\mathsf{cond}\ (e_1 \rhd P_1, \ldots, e_n \rhd P_n)$ presents the conditional choice between a set of processes: it behaves as P_i, where i is the smallest integer for which e_i evaluates to true. Finally, process constants provide a mechanism for including recursion in the calculus.

Moving on to the *species* level, we note that during their life cycle, individuals may produce offsprings. To capture the creation of new individuals, we employ the special *species* processes R. R, defined as $!a.P$, is a replicated process which may continuously receive input through channel a. This will result in the creation of a new individual P. Such inputs will be provided by individuals in the phase of reproduction.

Finally, population systems are built on the basis of located individuals, $P{:}[\![\mathbf{s}, \ell]\!]$, where $\mathbf{s}$ and ℓ are the species and the location of the individual, and species $C{:}[\![\mathbf{s}]\!]$, where $\mathbf{s}$ is the name of the species. Furthermore, $S\backslash L$ models the restriction of the use of channels in set L within S and $[S]$ is the closure operator. This operator is applied at the highest level of a population system and its semantic significance is that it allows us to select the valid behavior of the system based on the conditions that the system satisfies, as expressed in the semantics of the calculus.

As an example, consider the model described in [2] where a set of individual live on an $n \times n$ lattice of resource sites and go through phases of reproduction and dispersal. Specifically, the studied model considers a population where individuals disperse in space while competing for a location site during their reproduction phase. They produce an offspring only if they have exclusive use of a location. After reproduction the offsprings disperse and continue indefinitely the same behavior. In PALPS, we may model the described species $\mathbf{s}$ as $!rep.P$, where

$$\begin{aligned} P &\stackrel{\text{def}}{=} move.\surd.\mathsf{cond}\ (\mathbf{s}@\ell = 1 \rhd P_1; \mathsf{else}\ \surd.P) \\ P_1 &\stackrel{\text{def}}{=} \overline{rep}.\surd.P_1 + \overline{rep}.\overline{rep}.\surd.P_1 \end{aligned}$$

We point out that the conditional construct allows us to determine the exclusive use of location ℓ by an individual where else is used as a shorthand to $\neg(\mathbf{s}@\ell = 1)$. Furthermore, note that P_1 models the nondeterministic production of one or two offsprings of the species. During the dispersal phase, an individual moves to a neighboring location which is chosen nondeterministically, as prescribed in the semantics of the next Section. Then a system containing of two individuals at a location ℓ and one in location ℓ can be modeled as $[P{:}[\![\ell, \mathbf{s}]\!] | P{:}[\![\ell, \mathbf{s}]\!] | P{:}[\![\ell', \mathbf{s}]\!] | (!rep.P){:}[\![\mathbf{s}]\!]]$.

To model a competing species $\mathbf{s}'$ which preys on $\mathbf{s}$, we may define the process $!rep'.Q$, where

$$\begin{aligned} Q &\stackrel{\text{def}}{=} \mathsf{cond}\ (\mathbf{s}@\ell \geq 1 \rhd prey.\surd.Q_1; \mathsf{else}\ \surd.Q_2) \\ Q_1 &\stackrel{\text{def}}{=} \overline{rep'}.\surd.Q \\ Q_2 &\stackrel{\text{def}}{=} \mathsf{cond}\ (\mathbf{s}@\ell \geq 1 \rhd prey.\surd.Q_1; \mathsf{else}\ \surd.die.\mathbf{0}) \end{aligned}$$

This species looks for a prey. If it succeeds it produces an offspring. If it fails for two consecutive time units its dies.

The notion of food at a location may also be modeled in PALPS. A channel *eat* is employed to model eating and, for example, a food source at location ℓ of amount n which replenishes every t time units can be described as $Food_{n,t}{:}[\![f, \ell]\!]$, where

$$Food_{i,j} \stackrel{\text{def}}{=} \begin{cases} eat.Food_{i-1,j} + \surd.Food_{i,j-1} & \text{if } i > 0, j > 0 \\ \surd.Food_{i,j-1} & \text{if } i = 0 \\ Food_{n,t} & \text{if } j = 0 \end{cases}$$

2.2 The Semantics

The semantics of PALPS is defined in terms of a structural operational semantics, which is given at two levels in Tables 1 and 2. The rules of Table 1 describe the behavior of individuals in isolation whereas the rules in Table 2 the behavior of complete systems. A transition of P has the form $P \xrightarrow{e,\eta} P'$, specifying that P can perform action η under condition e and evolve into P'.

Table 1: **Transition rules for individuals**

(Nil)	$\mathbf{0} \xrightarrow{true,\surd} \mathbf{0}$	(Act)	$\eta.P \xrightarrow{true,\eta} P$
(Sum)	$\dfrac{P_i \xrightarrow{e,\alpha} P_i', i \in \{1,2\}}{P_1 + P_2 \xrightarrow{e,\alpha} P_i'}$	(Const)	$\dfrac{P \xrightarrow{e,\alpha} P'}{C \xrightarrow{e,\alpha} P'} \quad C \stackrel{\text{def}}{=} P$
(Cond)	$\dfrac{P_i \xrightarrow{e,\alpha} P_i', e' = e_i \wedge (\bigwedge_{j<i} \neg e_i)}{\mathsf{cond}\ (e_1 \rhd P_1, \ldots, e_n \rhd P_n) \xrightarrow{e \wedge e',\alpha} P_i'}$		

Axiom (Nil) specifies that the inactive process may allow time to pass while Axiom (Act) states that $\eta.P$ can always execute action η and evolve to P. Rules (Sum) and (Const) express the semantics of nondeterministic choice and process constants in the expected way, where (Cond) stipulates that a conditional process may perform an (conditional) action of continuation P_i assuming that e_i evaluates to True and all e_j, $j < i$ are false.

Moving on to the higher level of the semantics, a transition of S has the form $S \xrightarrow{e,\alpha} S'$, signifying that S can perform action α under condition e and evolve into P', where actions α embed in them information about the location and the species taking part in the transition. Precisely, α can have one of the following forms:

- $\eta@l$ denotes the execution of action η at location ℓ.
- $prey,\mathbf{s}@l$ denotes the execution of a prey action at location ℓ by an individual belonging to the species $\mathbf{s}$.
- $\tau@l$ denotes the internal action, which arises when two complementary actions take place at the same location ℓ.
- $\surd$ denotes the time passing action.

Note that in the rules below we write β to range over all η actions with the exception of the specialized actions *move* and *prey* which are treated separately.

To begin with, rule (Loc) embeds location information to actions of a located process. Next, rule (Move) specifies that a located process may nondeterministically move to any neighboring location. Rules (Par1) and (Par2) stipulate the semantics of the parallel composition construct (their symmetric versions are omitted. Rule (Rep), illustrates the semantics of the replication construct. Here we may observe how the generator of new individuals may create a new located individual of a species while itself remaining in the environment for further use. Note that $(!a.P){:}[\![\mathbf{s}]\!]$ can communicate with individuals at all locations and the newly-instantiated individual acquires the location of its parent. Moving on to rule (Prey) we may see how a preying individual may kill an individual of a species on which it preys. Rule (Hide) implements restriction of the set of channels in L and rule (Close) distills only those transitions

Table 2: **Transition rules for systems**

(Loc)	$\dfrac{P \xrightarrow{e,\beta} P'}{P{:}[\![\mathbf{s},\ell]\!] \xrightarrow{e,\beta@\ell} P'{:}[\![\mathbf{s},\ell]\!]}$	(Move)	$\dfrac{P \xrightarrow{e,move} P', (\ell,\ell') \in \mathbf{Nb}}{P{:}[\![\mathbf{s},\ell]\!] \xrightarrow{e,move@\ell} P'{:}[\![\mathbf{s},\ell']\!]}$
(Par1)	$\dfrac{S_1 \xrightarrow{e,\eta@\ell} S_1'}{S_1 \mid S_2 \xrightarrow{e,\eta@\ell} S_1' \mid S_2}$	(Par2)	$\dfrac{S_1 \xrightarrow{e_1,a@\ell} S_1', S_2 \xrightarrow{e_2,\overline{a}@\ell} S_2'}{S_1 \mid S_2 \xrightarrow{e_1 \wedge e_2,\tau} S_1' \mid S_2'}$
(Rep)	$\dfrac{\ell \in \mathbf{Loc}}{(!a.P){:}[\![\mathbf{s}]\!] \xrightarrow{true,a@l} P{:}[\![\mathbf{s},\ell]\!] \mid (!a.P){:}[\![\mathbf{s}]\!]}$	(Prey)	$\dfrac{S \xrightarrow{e,prey,s@\ell} S', \mathbf{s}' \in Preys(\mathbf{s})}{P{:}[\![\mathbf{s}',\ell]\!] \mid S \xrightarrow{e,\tau} S'}$
(Hide)	$\dfrac{S \xrightarrow{e,\eta@\ell} S', \eta \notin \{a,\overline{a} \mid a \in L\}}{S\backslash L \xrightarrow{e,\eta@\ell} S'\backslash L}$	(Close)	$\dfrac{S \xrightarrow{e,\eta@\ell} S', S \models e}{[S] \xrightarrow{true,\eta@\ell} S'}$
(Time)	$\dfrac{S_1 \xrightarrow{e_1,\surd} S_1', S_2 \xrightarrow{e_2,\surd} S_2'}{S_1 \mid S_2 \xrightarrow{e_1 \wedge e_2,\surd} S_1' \mid S_2'}$		

of a system whose conditions are satisfied in the current state of the population system. Finally, (Time) imposes a synchronous nature to the time-passing action $\surd$.

3 Concluding remarks

This paper reports on work in progress towards the development of a process calculus for the spatially-explicit and individual-based modeling of ecological systems. In future work we intend to extend our study by developing the theory of the calculus and introducing probabilistic behavior. Most importantly, we plan to implement a tool to accompany our language for performing simulations and possibly analysis of modeled systems. In related work, we have in fact implemented a prototype tool for a variant of the calculus containing probabilistic choice, locations, movement and reproduction [4]. As our experiments have shown, and as one would expect, the notion of location increases the burden of evaluating systems. Thus, our future work will also have to concentrate on providing optimizations for system analysis.

References

[1] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo & G. Pardini (2009): *Spatial Calculus of Looping Sequences*. *Electronic Notes in Theoretical Computer Science* 229(1), pp. 21–39.

[2] A. Brännström & D. J. T. Sumpter (2005): *Coupled map lattice approximations for spatially explicit individual-based models of ecology*. *Bulletin of Mathematical Biology* 67(4), pp. 663–682.

[3] M. Cardona, M. Colomer, A. Margalida, I. Pérez-Hurtado, M. J. Pérez-Jiménez & D. Sanuy (2010): *A P System Based Model of an Ecosystem of the Scavenger Birds*. In: *Proceedings of WMC 2009*, LNCS 5957, Springer, pp. 182–195.

[4] X. Efthymiou (2010): *A Process Algebraic Framework and Tool for Reasoning about Spacially Explicit Population Systems*. Undergraduate thesis, University of Cyprus.

[5] S. C. Fu & G. Milne (2004): *A Flexible Automata Model for Disease Simulation*. In: *Proceedings of ACRI 2004*, LNCS 3305, Springer, pp. 642–649.

[6] J. Ch. Godskesen (2009): *A Calculus for Mobile Ad-hoc Networks with Static Location Binding*. *Electronic Notes in Theoretical Computer Science* 242(1), pp. 161–183.

[7] C. A. R. Hoare (1985): *Communicating Sequential Processes*. Prentice-Hall.

[8] M. John, R. Ewalda & A. M. Uhrmacher (2008): *A Spatial Extension to the π-Calculus*. *Electronic Notes in Theoretical Computer Science* 194, pp. 133–148.

[9] D. Kouzapas & A. Philippou (2010): *A Process Calculus for Systems with Dynamic Topology*. Technical Report TR-10-01, University of Cyprus.

[10] C. McCaig, R. Norman & C. Shankland (2008): *Process Algebra Models of Population Dynamics*. In: *Proceedings of AB 2008*, LNCS 5147, Springer, pp. 139–155.

[11] M. Merro (2009): *An Observational Theory for Mobile Ad Hoc Networks*. *Information and Computation* 207(2), pp. 194–208.

[12] R. Milner (1980): *A Calculus of Communicating Systems*. Springer.

[13] C. Tofts (1993): *Using process algebra to describe social insect behaviour*. *Transaction of the Society for Computer Simulation* 9, pp. 227–283.

[14] C. Tofts (1994): *Processes with probabilities, priority and time*. *Formal Aspects of Computing* 6, pp. 536–564.

An Application of Model Checking to Epidemiology (Extended abstract)

Peter Drábik
Dipartimento di Informatica, Università di Pisa
Largo B. Pontecorvo 3, 56127, Pisa, Italy
drabik@di.unipi.it

Guido Scatena
IMT Lucca Institute for Advanced Studies
Piazza San Ponziano 6, 55100, Lucca, Italy
g.scatena@imtlucca.it

The aim of this work is evaluate the applicability of probabilistic model checking as an analytic tool for understanding the dynamics of ecological models. We reason that a specialist can obtain useful insights, as the results of the analyses is exact as opposed to classical deterministic models. However, in order to combat the high computational costs additional research is necessary.

The formalism used to model the population by individual-based approach is stochastic sync-automata. We employ the PRISM probabilistic model checker to evaluate the logic-based properties. As case studies we considered the models for infectious disease with droplet contact route of transmission and also vector-borne transmitted diseases.

1 Introduction

Mathematical modelling of the progress of infectious diseases gives the means to discover the likely outcomes of epidemics or helps manage them by vaccination. In case of large populations deterministic approach using differential equations can be employed. More recently, individual-based methodology has been applied to study the epidemic dynamics. Although computationally quite expensive, it has an ambition to account for stochastic effects characterising such dynamics in small populations. Individual-based models, thanks to their similarity to systems of interacting agents, allow benefitting from analysis methods originally developed in computer science.

In this work we attempt to apply such a technique, probabilistic model checking [10], to study compartmental population models. These are utilised for many common childhood diseases that confer long-lasting immunity.

In particular, we start by presenting a modelling language called sync-programs [3] originally developed for description of biological systems such as signalling pathways. It can well serve as agent description language, where each individual is modelled by a finite-state automaton. To represent interactions, synchronisation is utilised. The approach permits multi-way synchronisation if needed. For the purpose of the application to epidemiology, the language has been stochastically extended: each interaction is enriched with rates determining the likelyhood of the related event.

We present the framework on two models. The first is a compartmental model *SIR* from the literature and where only hosts are modelled – each individual by one automaton. This model describes well the progress of infectious diseases with droplet contact route of transmission such as measles, mumps and rubella [12]. The other model, *VectSIR*, demonstrates the dynamic aspect of the description language namely of creation of new automata in the runtime. This approach serves for investigating epidemics of diseases with vector-borne transmission. Vectors are organisms that do not cause disease themselves but that transmit infection by conveying pathogens from one host to another. Even though not supported by exact data from field studies, we believe these models can faithfully be employed for studying tick-borne

encephalitis, Chikungunya (vector mosquitoes), Pappataci fever (vector sandfly) and diseases caused by Rickettsia bacteria like rickettsialpox, Boutonneuse fever and various spotted fevers (transmitted by ticks, fleas and lice).

The analysis is done via probabilistic model checking, a formal verification technique for analysis of systems that exhibit stochastic behaviour. It consists in verification, based on exhaustive exploration of the constructed model, of quantitative properties specified in probabilistic logic.

We are able to check properties regarding the behaviour of each population over time, as for instance to identify conditions for the outbreak of the infection or to demonstrate the retreat of the epidemic. Note than in contrast to simulation approaches with a limited number of traces we obtain exact results based on inspection of all possible behaviours of the system, giving strong formal guarantees.

2 Stochastic dynamic sync-programs

First we describe dynamic sync-programs [3]. Then, for the purpose of modelling the progress of an epidemic in a population, we introduce a stochastic extension of these programs.

2.1 Dynamic sync-programs

In order to model an epidemic in a population, we use a component-based approach. Each component represents an individual, e.g. a host (human, animal) or a vector.

We assume a finite *set of component types* CT. With every component type i a set AP_i of *atomic propositions* is associated, encoding the state of component of that type. The sets of atomic propositions are pairwise disjoint for all the types, i.e. if $i \neq j$ then $AP_i \cap AP_j = \emptyset$. We assume a function $type : AP \to CT$ that for an atomic proposition from AP_i gives its type i. We override the function with its lifting to a set of atomic propositions.

A component is modelled by a finite-state machine called sync-automaton.

Definition 1. A *sync-automaton* A_i, where i is a type, is a tuple $(S_i, S_i^0, SC_i, CC_i, R_i)$ where:

- $S_i \subseteq \mathscr{P}(AP_i)$ is the set of *states*
- $S_i^0 \subseteq S_i$ is the set of *initial states*
- SC_i is a set of labels of the form $\wedge_{k\in L} U_k{:}V_k$ where $L \subseteq \mathbb{N}$ and for every $k \in L$ there is a j such that U_k, V_k are sets of atomic propositions drawn from AP_j or their negations. A label from SC_i is called a *synchronisation condition*
- CC_i is a set of labels of the form $\wedge_{k'\in K} W_{k'}$ where $K \subseteq \mathbb{N}$ and for every $k' \in K$ there is a $j \in I(i)$ such that W_j is a set of atomic propositions drawn from AP_j or their negations. A label from CC_i is called a *creation condition*
- $R_i \subseteq S_i \times SC_i \times CC_i \times S_i$ are the *moves* between states.

Each state of a sync-automaton A_i^I is a truth value assignment to atomic propositions of component type i. Each move is labelled by a synchronisation condition sc and by a creation condition cc. We denote a move from state s_i to state t_i with labels sc and cc by $s_i \xrightarrow[cc]{sc} t_i$. This move intuitively means that automaton A_i^I can move from s_i to t_i if there are concurrently performed moves of sync-automata satisfying condition sc. Note that in this way multi-way synchronisation can be obtained. Moreover, by performing the move, automata described by cc are created.

The synchronisation condition is a label in the form of $\wedge_{k\in L} U_k{:}V_k$ and specifies requirements for automata to synchronise with. For every k in L, the sets of propositions U_k and V_k are to be satisfied in

the starting and ending state, respectively, of the concurrently performed move of a sync-automaton of type $type(U_k)$.

We remark that in the synchronisation condition of a move of a sync-automaton of type i there can be multiple references to the components of a type j, referring to different instances of sync-automata of such a type. References to other instances of the same type i are also allowed. Moreover, note that it is possible for L to be empty. Intuitively, this means that the sync-automaton A_i^I does not have any requirements on other sync-automata. We write a synchronisation condition of this form, i.e. $\wedge_{j\in\emptyset} U_k{:}V_k$, as $NOSYNC$. Move $s_i \xrightarrow{NOSYNC} t_i$ represents an autonomous move of A_i^I. We use an abbreviation $U_k \circlearrowleft$ for a condition of the form $U_k{:}U_k$.

The creation condition is a label in form $\wedge_{k\in K} W_k$ that specifies sync-automata that are to be created. For every k in K, the set of atomic propositions (or negations) W_k encodes an initial state of sync-automaton A_j^I for some $j \in I(i)$ which will be created having this initial state. In case the multiset K is empty, the creation condition is not displayed.

By running in parallel sync-automata, we obtain a sync-program. Note that a program can contain multiple sync-automata of the same type.

Definition 2. A *sync-program* is a tuple $P = (s_{i_1}||\ldots||s_{i_n})$, where each s_i is an initial state of sync-automaton A_j with type $j = type(s_i) \in I$. The creation conditions on the moves of all sync-automata are well formed, i.e. for every creation condition $\wedge_{k'\in K} W_{k'}$ every W_k describes a unique state s' of sync-automaton with type in I and s' is initial.

Now we define the semantics of dynamic sync-programs. Here we describe it intuitively, formal definitions can be found in [3].

The semantics of a dynamic sync-program can be given in terms of a labelled transition system. The states are multisets of states of all automata types, initial states correspond to multisets of initial states admissible by automata in the program.

Transition between two states s and t with label l in the semantics corresponds to a synchronisation of a minimal set l of automata such that the synchronisation is composed of moves of automata from their local states described in s to their local states described in t. Moreover the synchronisation conditions of all the moves are satisfied, in that each precondition is satisfied in s and each postcondition in t. All automata not participating on the synchronisation stay idle. State t also includes initial states of newly created automata described in creation conditions of synchronising moves.

2.2 Stochastic extension

The standard way of extending a formalism to model quantitative aspects [5] of systems is by incorporating a collision-based stochastic framework on the lines of the one presented by Gillespie [4]. The idea is that a rate constant is associated with each considered reaction. Following the law of mass action, such a constant is obtained by multiplying the kinetic constant of the reaction by the number of possible combinations of reactants that may occur in the system. The resulting rate is then used as the parameter of an exponential distribution modelling the time spent between two occurrences of the considered reaction.

The use of exponential distributions to represent the (stochastic) time spent between two occurrences of chemical reactions allows describing the system as a Continuous Time Markov Chain (CTMC), and consequently allows verifying properties of the described system by probabilistic model checking.

In the case of stochastic sync-programs, the transition in the semantics represents a reaction and automata moves the reactants. Hence stochastic rates need to be associated with every move. A move $s_i \xrightarrow[cc]{sc} t_i$ with rate r is denoted as $s_i \xrightarrow[cc]{sc[r]} t_i$.

In the semantics, the exact rate of a transition is equal to the product of the rates of all participating moves multiplied by the number of possible combinations of automata moves. More precisely rate of a transition tr is $r_{tr} = \Pi_{m \in tr}(r_m * \#_m)$ where for each move m participating on the transition r_m is the rate of m and $\#_m$ is the number of automata able and ready to perform this move.

In order to make sure that the rate of a synchronised transition is meaningful, a common technique is to make one move active, which actually defines the rate for the synchronised transition, and the other moves passive, with rates 1.

3 Probabilistic model checking

Model checking is a fully automatic verification method based on the exhaustive search of the state space of the system. Probabilistic model checking enables checking quantitative properties regarding time and probabilities.

Traditional model checking aims at checking the validity of a temporal logic formula ϕ (in LTL, CTL, CTL* , or the like) on a given Kripke structure M, i.e., it checks whether $M \models \phi$. Kripke structures are transition systems, where states are labeled with propositions, and the transition relation is total. In the probabilistic setting, however, different models exist, and their appropriateness is mainly determined by the application, e.g., is continuous time needed, is there a need for nondeterminism, and so forth.

In setting of the present work, as our stochastic semantics is a Continuous Time Markov Chain (CTMC), we are interested in model checking of these models. The logic used is Probabilistic Computation Tree Logic (PCTL) [6]. PCTL is a quantitative variant of CTL where the path quantifiers A and E are replaced by a probabilistic operator P that allows querying the probability of a path formula. Another logic, Continuous Stochastic Logic (CSL)[1] extends PCTL's path operators with time bounds.

Efficient probabilistic model checking tools exist and have been applied by a large number of users from different areas. We concentrate on the model checker PRISM [7]. PRISM supports model checking of CTMCs and Markov decision processes for the logics PCTL and CSL. Other probabilistic model checkers include MRMC, LiQuor and YMER.

4 Application

4.1 Compartmental models in epidemiology

In order to represent the development of an epidemic a model needs to just the characteristic aspects that are relevant to the infection in consideration. In case of SIR model (fig. 1), the population is divided into three compartments: those who are susceptible (S) to the disease, those who are infected (I) and those who have recovered and are immune (R). In the diseases under consideration a single epidemic outbreak is far more rapid than the vital dynamic, we might neglect the birth/death processes.

The typical progress of each host is S to I to R. We model this with a sync-automaton representing each individual. Atomic propositions used are three S, I and R. Each automaton has three states: $\{S, \neg I, \neg R\}$, $\{\neg S, I, \neg R\}$ and $\{\neg S, \neg I, R\}$. We display only the atomic propositions that are true in a state.

The move from S to I occurs by getting an infection from an individual that is infected. This is modelled by a synchronisation, where this move can only be concurrently with a move of another sync-automaton that goes from state satisfying I to state satisfying I (denoted as $I \circlearrowleft$). The rate of this move r_1 represents which the probability of getting the disease in a contact between a susceptible and an

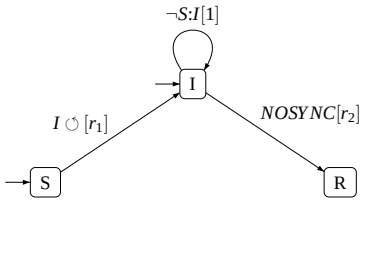
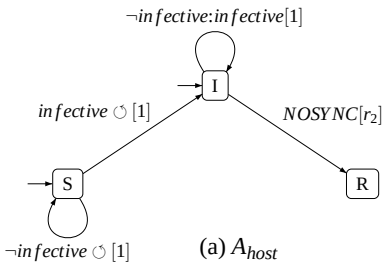


Figure 1: SIR model, A_{SIR}.



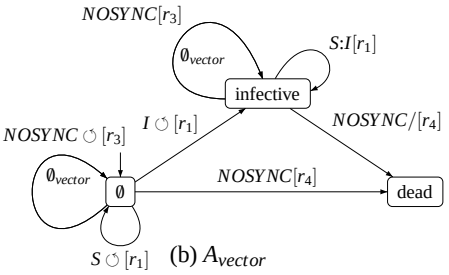


Figure 2: VectSIR model

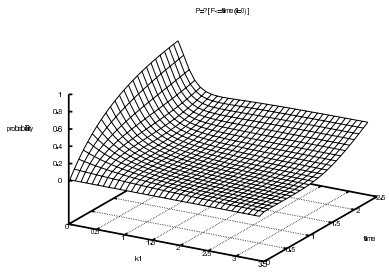


Figure 3: Reach. $(I = 0)$ vs. time.

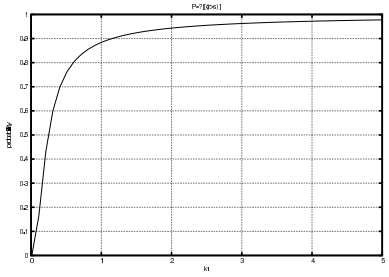


Figure 4: Reach. $(R > S)$.

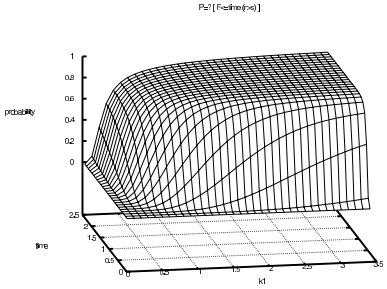


Figure 5: Reach. $(R > S)$ vs. time.

infectious subject. The synchronising partner loop move on state I is passive and thus has rate 1. The recovery from the disease occurs autonomously for each individual, hence the *NOSYNC* move. Its rate is in general dependent on the recovery time D, in particular $r_2 = 1/D$. The key role in determining the dynamic of our model plays the ratio of r_1 to r_2.

In the second model we consider disease with vector-borne transmission. The hosts are modelled by using the SIR approach (fig. 2a), the change is that the infection occurs by a vector (fig. 2b). By feeding on blood of an infected host the vector gets infected (move to state $infective$ with rate r_1) transmits the infection to all successive hosts. We consider a fixed population of hosts. On the other hand, since the reproduction cycle of most vectors (mainly insects) tends to be considerably shorter, we model it by creating new individuals (*NOSYNC* loops in states $\emptyset$ and $infective$ at rate r_3). Moreover, vectors, not depending, on their infectiveness die at rate r_4.

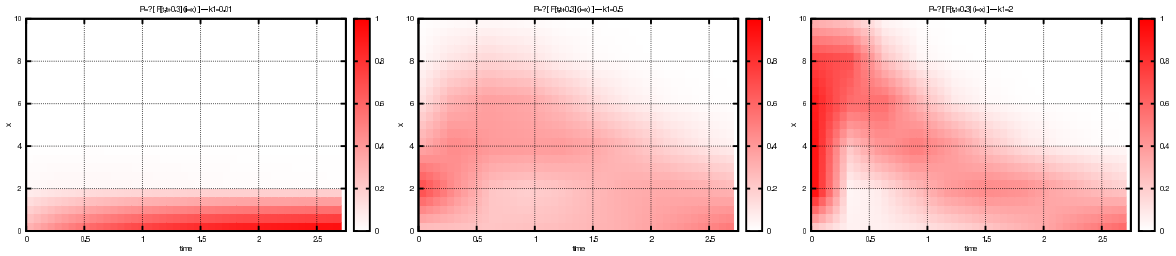
In order to be able to perform the analysis described in the following subsection, we perform an ad hoc translation of the stochastic sync-program to the PRISM input language. The translation preserves the CTMC semantics of stochastic sync-programs.

Note that for obtaining a model that is amenable to probabilistic model checking, that is a finite-state model, we need to restrict the number of instances of automata of each type. Thus, we set a limit of number of individuals to 10, both for hosts and vectors.

4.2 Analysis via probabilistic model checking

We performed probabilistic model checking of the two models, varying rates related to the infection process.

The first model, SIR, is represented by program $SIR = I||S||S||S||S||S||S||S||S||S$ with one automaton



(a) $r_3 = 0,01$ (b) $r_3 = 0,5$ (c) $r_3 = 2$

Figure 6: SIR: probability distribution of I vs. time

in state I and nine in state S. In this model we expected different behaviour depending on the ratio $R_0 = r_1/r_2$. From the deterministic model analysis, if $R_0 > 1/S(0)$ then there is an outbreak with an increase of infectious population; if $R_0 < 1$ then no epidemic outbreak occurs, independently of the initial population in S. This conjecture is checked by fixing rate r_2 and varying r_1. We check the PCTL formula $P =?[((R > S))]$ representing that the population of recovered individuals is bigger that population of susceptibles. Since before a susceptible becomes recovered, necessarily spends some time as infected, the property means that at least half of the population was infected. On fig. 4, result of plotting the evaluation of the above formula with different values of r_1 it can be seen, that if r_1 is small, the probability of reaching a point, where at least half of the population was infected, is low. When increasing r_1, probability of such event increases towards 1. On figure fig. 5 this progress is plotted against time as the evaluation of CSL formula $P =?[F <= t((R > S))]$ again varying r_1.

Probability of the retreat of the epidemic, or reaching state $(I = 0)$ is 1. On fig. 3 it is shown how the retreat is likely to happen in time t.

That the epidemic with small r_1 does not occur is clear from fig. 6a in which the probability of being in a state $I = X$ at time Y is expressed by colour intensity. With higher values of r_1 number of infected individuals is likely to increase and then due to constant population size decrease to 0. The higher r_1 is, the more rapidly the epidemic occurs (fig. 6b and 6c).

In the second model VectSIR vector-borne transmission is considered. The program *VectSIR* consists of nine susceptible hosts, one infected and five non infected vectors. The decisive rate for the speed of epidemic outbreak is the creation rate of new vectors. This is witnessed in fig. 8 and fig. 9 where r_3 is varying with r_1, r_2 and r_4 fixed. Similarly as in the previous model, the retreat of the infection is unavoidable (probability of reaching $(I = 0)$ is 1) and fig. 7 details is progress over time. The probability distributions of values of I in time for r_3 equal to $0,01$, 1 and 2 are shown in figures 10a, 10b and 10c, respectively.

5 Discussion

In this work we presented modelling of a progress of an epidemic via an stochastic individual-based approach. Moreover, analysis technique called probabilistic model checking was applied to study the properties of the model and sensitivity to parameters.

Stochastic approach, as opposed to the classical deterministic one, considers all possible evolutions of the system and thus provides exact results. In case of small populations the model can witness presence of stochastic effects in the system.

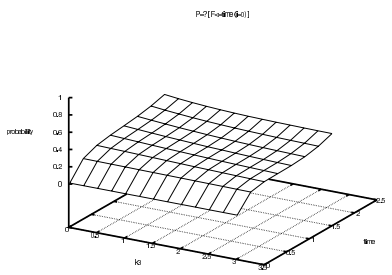


Figure 7: Reach. $(I = 0)$ vs. time.

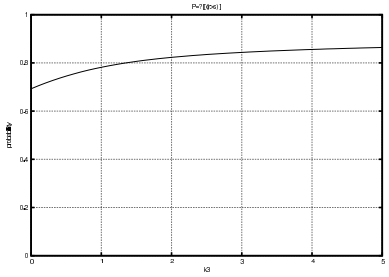


Figure 8: Reach. $(R > S)$.

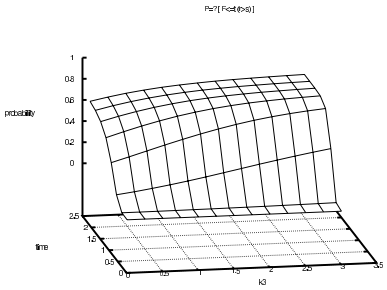
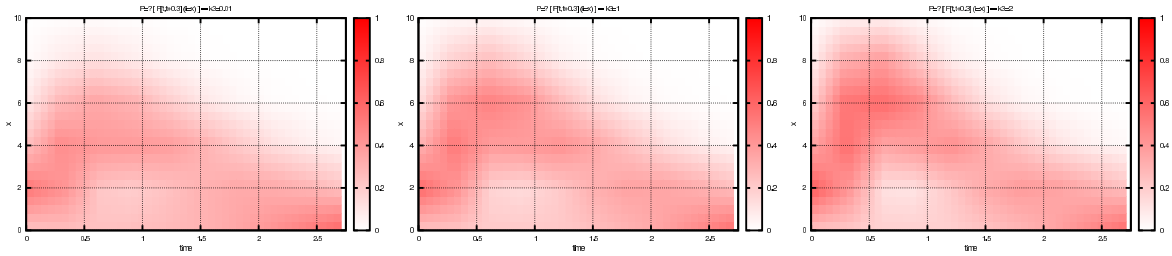


Figure 9: Reach. $(R > S)$ vs. time.



(a) $r_3 = 0,01$ (b) $r_3 = 1$ (c) $r_3 = 2$

Figure 10: VectSIR: probability distribution of I vs. time

We used an automata based formalism to model individuals, where each agent is represented by an finite-state automaton. The formalism seems to be a suitable means for description of these systems, also because of the powerful possibility of specifying interactions of individuals as synchronisation. Another necessary aspect that allows for dynamicity is the runtime automata creation. Since arbitrarily complex behaviour of one agent is expressible by a finite-state automaton, large scale of epidemics can be modelled.

As for the analysis method, probabilistic model checking provides useful insight into the dynamics of the modelled system. Complex queries can be evaluated over the models considering probabilities of values of variables in question and, in turn, plot the results in graphs.

A serious drawback, however, are the computational costs of the procedure. In particular, in order to evaluate the queries in reasonable time (in the order of hours), we needed to limit the analysis to the order of tens of individuals. A possible resolution of this impediment is to use approximate model checking that admits errors as long as they can be bound [9].

As regards related work, probabilistic model checking has recently been applied to study epidemiological models by Huang [8] who focuses the analysis to preventative and controlling measures to limit the effects of the diseases. Ciocchetta and Hilston [2] apply the formalism and toolkit Bio-PEPA to modelling and analysis of avian influenza. More often, stochastic models in epidemiology are used in connection with analysis by simulation [11].

Probabilistic model checking shows signs of being an useful tool in analysis of dynamics of ecological models, and is worth further investigation.

References

[1] Adnan Aziz, Kumud Sanwal, Vigyan Singhal & Robert Brayton (1996): *Verifying continuous time Markov chains*. *Computer Aided Verification* , pp. 269–276.

[2] Federica Ciocchetta & Jane Hillston (2010): *Bio-PEPA for Epidemiological Models*. *Electron. Notes Theor. Comput. Sci.* 261, pp. 43–69.

[3] Peter Drábik, Andrea Maggiolo-Schettini & Paolo Milazzo (2010): *Dynamic Sync-programs for Modular Verification of Biological Systems*. *NCMA10. In press* .

[4] Daniel T. Gillespie (1977): *Exact stochastic simulation of coupled chemical reactions*. *The Journal of Physical Chemistry* 81(25), pp. 2340–2361. Available at `http://dx.doi.org/10.1021/j100540a008`.

[5] John Haigh (2007): *Stochastic Modelling for Systems Biology by D. J. Wilkinson*. *Journal Of The Royal Statistical Society Series A* 170(1), pp. 261–261. Available at `http://ideas.repec.org/a/bla/jorssa/v170y2007i1p261-261.html`.

[6] Hans Hansson & Bengt Jonsson (1994): *A Logic for Reasoning about Time and Reliability*. *Formal Aspects of Computing* 6, pp. 102–111.

[7] A. Hinton, M. Kwiatkowska, G. Norman & D. Parker (2006): *PRISM: A Tool for Automatic Verification of Probabilistic Systems*. In: H. Hermanns & J. Palsberg, editors: *Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06), LNCS* 3920, Springer, pp. 441–444.

[8] Samuel Huang (2010): *Probabilistic Model Checking of Disease Spread and Prevention*. Technical Report, Computer Science Department, University of Maryland.

[9] Joost-Pieter Katoen (2007): *Abstraction of Probabilistic Systems*. In: *FORMATS*, pp. 1–3. Available at `http://dx.doi.org/10.1007/978-3-540-75454-1_1`.

[10] M. Kwiatkowska (2007): *Quantitative Verification: Models, Techniques and Tools*. In: *Proc. 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*, ACM Press, pp. 449–458.

[11] Charles J. Mode & Candace K. Sleeman (2000): *Stochastic Processes in Epidemiology: HIV/AIDS, Other Infectious Diseases and Computers*. World Scientific Publishing Company. Available at `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/981024097X`.

[12] Charles Nunn & Sonia Altizer (2006): *Infectious Diseases in Primates: Behavior, Ecology and Evolution*. Oxford University Press, Oxford. Available at `http://dx.doi.org/10.1093/acprof:oso/9780198565857.001.0001`.