Scientific Annals of Computer Science vol. 23 (1), 2013, pp. 119–167
doi: 10.7561/SACS.2013.1.119

Simulation and Verification in a Process Calculus for Spatially-Explicit Ecological Models

Anna PHILIPPOU[1], Mauricio TORO[1,2], Margarita ANTONAKI[1]

Abstract

We propose PALPS, a Process Algebra with Locations for Population Systems. PALPS allows us to produce spatially-explicit individual-based ecological models and to reason about their behavior. PALPS has two abstraction levels: At the first level, we may define the behavior of an individual of a population and, at the second level, we may specify a system as the collection of individuals of various species located in space. In PALPS, the individuals move through their life cycle while changing their location and interact with each other in various ways such as predation, infection or mating. Furthermore, we propose a translation of a subset of PALPS into the probabilistic model checker PRISM. We illustrate our framework via models of dispersal in metapopulations and by applying PRISM on PALPS models for verifying temporal logic properties and conducting reachability and steady-state analysis.

Keywords: Process calculi, ecology, probabilistic model checking, spatially-explicit individual-based models

1 Introduction

Population ecology is a subfield of ecology that deals with the dynamics of species populations and how these populations interact with the environment. Its main aim is to understand how the population sizes of species

[1]Department of Computer Science, University of Cyprus. P.O. Box 20537 1678 Nicosia, Cyprus. Email: {annap,mtoro,cs05ma}@cs.ucy.ac.cy

[2]This work was carried out during the tenure by the second author of an ERCIM "Alain Bensoussan" Fellowship Programme. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 246016.

that live together in groups change over time and space. It has been of special interest to conservation scientists and practitioners who are interested in predicting how species will respond to specific management schemes and in guiding the selection of reservation sites and reintroduction efforts, e.g. [21, 36]. To make such predictions, scientists have been constructing models of ecosystems. These models are abstract representations of the systems in question which are subsequently studied to gain understanding of the real systems. Models enable researchers to simulate large-scale experiments that would be too costly or unethical to perform on a real ecosystem. They also enable the simulation of ecological processes over long periods of time.

One of the main streams of today's theoretical ecology is the *individual-based* approach to modeling population dynamics. In this approach, the modeling unit is that of a *discrete individual* and a system is considered as the composition of individuals and their environment. The concept of an individual does not always need to coincide with that of an individual organism. More general population units, such as ant colonies and bird flocks, can also be employed and the resulting models do not differ technically from models treating individual organisms.

Since individuals usually move from one location to another, it is common in individual-based modeling to represent space explicitly. There are four different frameworks in which *spatially-explicit individual-based models* can be defined [5]. They differ in the way space and time are modeled: each can be treated either discretely or continuously. The four resulting frameworks have been widely studied in population ecology and they are considered to complement as opposed to compete with each other. In fact, in many cases, models of more than one types can be constructed for a system in question, with each approach offering a distinct perspective towards the understanding of the system.

In this work, our aim is to introduce a process-calculus framework to enable spatially-explicit modeling of ecological systems. Indeed, process calculi also known as process algebras, first proposed in [31, 25] to aid the understanding and reasoning about communication and concurrency, provide a number of features that make them suitable for capturing and reasoning about biological as well as ecological systems. To begin with, process calculi enable the construction of models by composing together smaller interacting components. This feature of modularity is especially suited towards the individual-based approach of modeling populations, as it enables one to describe the evolution of each individual of one or more populations as a process and, subsequently, to compose a set of individuals (as well as their environment) via the parallel operator into a complete ecological system. The semantics of the process calculus then generates automatically the complete state space of the system by taking into account all possible interactions and evolutions of the individuals/components of which the systems is composed. Furthermore, process calculi focus on the notion of *observation*, which can be varied according to the needs of the problem under investigation. A modeler can build a system incrementally and introduce less or more detail depending on the desired level of abstraction while, via the use of restriction operators, the modeler may determine the parts of the system to be studied. Finally, the development of process calculi has been investigated into a wide range of directions and accompanied by a variety of analysis techniques. On the one hand, features such as time, probability and stochastic behavior have been extensively studied in the context of process calculi and they can be exploited to provide more accurate models. On the other hand, associated analysis techniques and tools have been developed and they can be used to analyze and predict system behavior. Such techniques include analysis of properties that can be expressed as a temporal logic formula, the use of equivalences for comparing the behaviors of different systems or for minimizing the state space of a system and, more recently, in the domain of biological systems, techniques for deriving mean-field equations that describe the population level dynamics of individual-based models [30].

The process-calculus framework we propose follows the individual-based modeling and, in particular, it falls in the discrete-time, discrete-space class of Berec's taxonomy [5]. Our process calculus, PALPS, associates processes with information about their location and their species. The habitat is defined as a graph consisting of a set of locations and a neighborhood relation. Movement of located processes is then modeled as the change in the location of a process with the restriction that the originating and the destination locations are neighboring locations. In addition to moving between locations, located processes may communicate with each other by exchanging messages upon channels. Communication may take place only between processes which reside at the same location. Furthermore, PALPS may model probabilistic events, with the aid of a probabilistic choice operator, and uses a discrete treatment of time. Finally, in PALPS, each location may be associated with a set of attributes capturing relevant information such as the capacity or the quality of the location. These attributes form the basis of a set of expressions that refer to the state of the environment and are employed within models to enable the enunciation of location-dependent behavior.

The operational semantics of our calculus is given in terms of labeled transition systems which combine probabilistic and nondeterministic behavior. Analysis techniques for this type of models have been widely studied in the formal methods literature. To allow such analysis and take advantage of existing tools, we describe a translation of (a subset of) PALPS into the PRISM language. This encoding can be employed for both simulating and for model checking PALPS systems using the PRISM tool [1], a model checker for probabilistic systems.

In the remainder of this section, we briefly describe discrete-time discrete-space individual-based modeling as discussed in [5]. We continue with an overview of related work on the formal modeling of ecological and biological systems and we summarize the contributions of this article.

Discrete-time, discrete-space, individual-based modeling. Individual-based modeling is an approach to model ecological systems which demands description of the environment and each individual living in it, together with its individual-individual and individual-environment interactions. It assumes that space is divided into discrete locations and that time runs in discrete steps. The main processes that determine the fate of an individual are mortality, reproduction, movement and predation. In the discrete-time approach, engagement in these processes is determined by a probability.

At almost every step of constructing a model, at least a few alternatives are plausible often involving tradeoffs between making the models more realistic though more complex to simulate and analyze. We discuss here two of the parameters that involve the treatment of space and the interleaving of actions.

Regarding space, discrete-space environments require a topology and regular lattices made up of squares are by far the most common choice. The size of the habitat can be considered as infinite, a choice that simplifies the derivation of many mathematical results, or finite, which is the choice taken in most simulations. In finite habitats one needs to specify the *boundary conditions*, that is, how to deal with the situation when an individual reaches the boundary of the habitat. There are three boundary conditions defined in the literature: *absorbing*, *reflecting* and *periodic*. Absorbing boundary conditions do not deal with boundary and edge effects, but focus on populations in a central portion of the environment. In reflecting boundary conditions, individuals that hit the boundary ricochet back into the habitat in a random direction. This seems realistic for animals living on an island or water animals living in a pond, but unrealistic for, e.g., plant seeds dispersal. Finally, in periodic boundary conditions, the opposite edges of the habitat are connected together. The argument towards the adoption of these conditions is that as one individual exits one side of the habitat another may enter from another one.

Finally, another issue that is relevant to modeling ecosystems is that of *process ordering* inside each time unit. In particular, simulations carried out by ecologists impose an order on the events that may take place within a model. For instance, if we consider mortality and reproduction within a single-species model three cases exist: concurrent ordering, reproduction preceding mortality and reproduction following mortality. In *concurrent ordering*, individuals may reproduce and die simultaneously. For *reproduction preceding mortality*, the population first reproduces, then all individuals, including new offspring, are exposed to death. For *reproduction following mortality*, individuals are first exposed to death and, subsequently, surviving individuals are able to reproduce. Ordering can have significant implications on the simulation. Thus, alternatives must be carefully studied before conclusions are drawn.

Formal frameworks. Various formalisms have been proposed in the literature for modeling biological and ecological systems. Similarly to ecosystem modeling, these approaches differ in their treatment of time and space and can be considered as supplements as opposed to rivals of each other as each offers a distinct view and techniques for analyzing systems.

One strand of these formalisms is based, like PALPS, on process calculi, and constitutes extensions of calculi such as CCS [31], the π-calculus [32] and CSP [25]. WSCCS [45] is one of the first proposals made towards modeling ecosystems. It is a probabilistic extension of CCS [31] with synchronous communication that has been employed in various ecological studies by the author and others [43, 29]. It follows the discrete-time approach to modeling but does not include the notion of space. WSCCS was also given a semantics in terms of *mean field equations* in [30]. This semantics captures the average behavior of a system over time, without computing the entire state space, therefore, avoiding the state-space explosion problem.

As far as continuous time is concerned, there are various proposals in the literature on stochastic process calculi. These include *stochastic* CCS [17], the *stochastic pi-calculus* [11] and the stochastic process algebra PEPA [24]. An extension of the latter, BIO-PEPA [16], has been developed for modeling and analyzing biochemical networks and includes features such as compartments, support for stoichiometry and general kinetic laws.

Moving on to space, various process calculi have been proposed in the literature to model space in order to support the reasoning of biological processes. One of the most common approaches involves the notion of a *compartment*. Compartments are closed areas which may contain elements and other compartments and they can be used to model biological membranes or simply to represent different abstract locations. Such proposals include the *calculus of wrapped compartments* (CWC) [9], *BioAmbients* [41] as well as *Brane calculi* [10].

A different approach towards modeling biological and ecological systems is that of *P systems* [40]. P systems were conceived as a class of distributed and parallel computing inspired by the compartmental structure and the functioning of living cells. P-systems have been extended in various directions such as probabilistic P-Systems [38], continuous P-systems [37], *dynamical probabilistic P systems* [38], spatial P systems [35], and *multi-environment P systems* [13]. They have been applied to a wide range of applications in the field of ecology, e.g. [6, 12, 7].

Stochastic simulation for P systems (and other formalisms) has also been actively investigated. Research efforts have been based on the Gillespie algorithm [22] which is currently used as the reference procedure for performing stochastic and discrete simulations of various biological systems. Of great interest is also the approximate algorithm Gillespie introduced in [23]. This consists of an approximate simulation strategy, referred to as the tau leaping method, in which, by using Poisson random numbers, it is possible to leap over many reaction events in a way that approximates the exact stochastic simulation. This strategy has also been extended in the spatial realm. Extensions are considered in [14] and evaluated in [26], whereas in [42] they were used as a basis for introducing a rewriting strategy for stochastic P systems of dynamically changing nested compartments and used for simulating ecological systems.

Finally, we mention the calculus of looping sequences [4], and its spatial extension [3] and *cellular automata* [19, 15].

Contributions. In this paper we present the process calculus PALPS, a process calculus for modeling spatially-explicit individual-based systems which adopts a discrete-time and discrete-space approach. This framework allows the specification of an ecosystem of multiple species residing on a finite topology. We model the *habitat* as a graph consisting of a set of locations and a neighborhood relation that allows the modeler to define the intended boundary conditions. We include a special movement action that allows movement of individuals along the graph.

Individuals are modeled as processes in PALPS possessing a species and a location that may change dynamically as computation proceeds. Individuals may engage in any of the basic processes of reproduction, dispersal, predation or death and they may communicate with other processes which reside at the same location. Probabilistic decisions are implemented through a probabilistic choice operator.

Technically, PALPS can be considered as an extension of CCS with probabilistic choice, locations and location attributes. In particular, it inherits from CCS the notion of *two-way communication* according to which processes may interact with each other whenever two components are able to execute complementary actions. We believe that this type of communication is appropriate for modeling various aspects of population behavior such as predation, infection and feeding: for each of these there exist two components able to engage in complementary behavior (e.g. predator vs. prey). The level of abstraction implemented by this binary (as opposed to synchronous) communication allows us to observe each local interaction and, for instance, to compute the number of births/infections during a time unit. To make this possible, interactions in PALPS, also referred to as internal actions in process calculi such as CCS, are labeled by the type of the action and the location where the action has taken place.

As far as space is concerned, PALPS shares a similar treatment of locations with process algebras developed for reasoning about mobile ad hoc networks such as [20, 27]. A significant factor that distinguishes PALPS from these and other process calculus approaches, is its ability to define location-dependent behavior. This is achieved by associating locations with attributes that capture information relevant to the location and form the basis of a set of expressions which may determine the evolution of individuals. Examples of such attributes are the temperature, the humidity, and the number of preys and predators on a location. Based on such attributes, processes can determine their behavior, e.g., whether to reproduce or to disperse. We illustrate the expressiveness of PALPS by constructing spatially-explicit individual-based models of metapopulation dispersal.

Another contribution of our work regards the development of a methodology for translating PALPS models into the PRISM language, with the prospect of making more advanced analysis of ecological models as opposed to just simulations. Most research on ecological systems models to date, focuses on the development of models of different metapopulation systems. One key question regarding these models is what one can do with the models other than just simulate trajectories. A possible answer to this question is to use model checking tools for automatically analyzing various properties of the model. As an example, the probabilistic model checker PRISM, [1], has been used to investigate continuous-time P systems. Various other formalisms have been translated to PRISM (e.g. [34]). However, as far as we know, there has been no work on model checking of spatially-explicit individual-based models. A contribution of this paper is to make this possible. Specifically, we propose a translation of a subset of PALPS into the Markov-decision-process component of the PRISM language. This subset of PALPS is the fragment of the language where the replication operator is replaced by a *restricted* replication operator. Essentially, this restricted operator places a bound on the number of new individuals than can be created of each species. This is necessary due to the fact that PRISM does not allow for the dynamic creation of modules, thus, the maximum total number of all modules/individuals must be created a-priori. We prove the correctness of our translation and we apply it to perform analysis of some simple examples.

Structure of the paper. The structure of the remainder of this paper is as follows. In Section 2 we present the syntax and the semantics of PALPS. We continue to illustrate the expressiveness of the calculus by providing models of metapopulation dispersal in Section 3. In Section 4 we present a translation of PALPS into the Markov-decision-process component of the PRISM language. We establish the correctness of the translation and we overview the types of analysis that this translation makes possible on PALPS models. We then apply these techniques on simple examples and we explore the potential of the approach in Section 5. Finally, in Section 6, we conclude with a discussion of future work.

2 The Process Calculus

In our calculus, *Process Algebra with Locations for Population Systems* (PALPS), we consider a system as a set of individuals operating in space, each belonging to a certain species and inhabiting a location. This location may be associated with attributes which describe characteristics of the location and can be used to define location-dependent behavior of individuals. Furthermore, individuals who reside at the same location may interact with each other by communicating upon channels, e.g. for predation, or they may migrate to a new location where they may continue their computation. PALPS may model probabilistic events with the aid of a probabilistic operator and uses a discrete treatment of time.

2.1 The Syntax

In this section we formalize the syntax of PALPS which is built based on the following basic entities:

- $\mathbf{S}$: a set of species ranged over by $\mathbf{s}$, $\mathbf{s}'$.
- $\mathbf{Loc}$: a set of locations ranged over by ℓ, ℓ'. The habitat of a system is then implemented via a relation $\mathbf{Nb}$, where $(\ell, \ell') \in \mathbf{Nb}$ exactly when locations ℓ and ℓ' are neighbors. For convenience, we use $\mathbf{Nb}$ as a function and write $\mathbf{Nb}(\ell)$ for the set of all neighbors of ℓ.
- $\mathbf{Ch}$: a set of channels ranged over by lower-case strings.
- Ψ: a set of attributes, ranged over by ψ, ψ'. We write ψ_ℓ for the value of attribute ψ at location ℓ.

Species and locations are characteristics associated with every individual in a PALPS system. The species characteristic is static whereas the location characteristic is dynamic: as computation proceeds, an individual may change its location from ℓ to ℓ' with the restriction that $(\ell, \ell') \in \mathbf{Nb}$. In turn, attributes are characteristics associated with locations and they may capture information such as the capacity, the temperature or the quality of the location. They form the basis of the set of expressions of the language which is defined below.

Expressions. PALPS employs two sets of expressions: *logical expressions*, ranged over by e, and *arithmetic expressions*, ranged over by w. These expressions are intended to capture environmental situations which may affect the behavior of individuals. Expressions e and w are constructed as follows:

$$\begin{array}{rcl} e & ::= & \mathit{true} \mid \neg e \mid e_1 \wedge e_2 \mid w \bowtie c \\ w & ::= & c \mid \psi@\ell^\star \mid \mathbf{s}@\ell^\star \mid @\ell^\star \mid \mathbf{op}_1(w) \mid \mathbf{op}_2(w_1, w_2) \end{array}$$

where c is a real number, $\bowtie \in \{=, \leq, \geq\}$ and $\ell^\star \in \mathbf{Loc} \cup \{\mathsf{myloc}\}$.

To begin with, logical expressions e are built using the propositional calculus connectives, as well as comparisons between an arithmetic expression w and a constant c (e.g., $\mathbf{s}_1@\ell + \mathbf{s}_2@\ell > 1$). Arithmetic expressions include three special expressions interpreted as follows: Expression $\psi@\ell^\star$ is equal to the value of attribute ψ at location $\ell^\star$. Expression $(\mathbf{s}@\ell^\star)$ is equal to the number of individuals of species $\mathbf{s}$ at location $\ell^\star$, and expression $@\ell^\star$ denotes the total number of individuals of all species at location $\ell^\star$.

Location $\ell^\star$ can be an arbitrary location or the special location myloc. This latter label is employed to bestow individuals with the ability to express conditions on the status of their current location no matter where that might be, as computation proceeds. Specifically, myloc refers to the actual location of the individual in which the expression appears and it is instantiated to this location when the condition needs to be evaluated (see rule (Cond) in Table 3). In conclusion, arithmetic expressions are the set of all expressions formed by arbitrary constants c, quantities $\psi@\ell^\star$, $\mathbf{s}@\ell^\star$, $@\ell^\star$, and the usual unary and binary arithmetic operations ($\mathbf{op}_1$ and $\mathbf{op}_2$) on real numbers. Logical expressions and arithmetic expressions are evaluated within a system environment (as defined in Tables 1 and 2).

Processes. The syntax of PALPS is given at 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 BNFs:

$$\begin{array}{rcl} P & ::= & \mathbf{0} \mid \sum_{i\in I} \eta_i.P_i \mid \bigoplus_{i\in I} p_i{:}P_1 \mid \mathsf{cond}\ (e_1 \rhd P_1, \ldots, e_n \rhd P_n) \mid C \\ R & ::= & !\mathit{rep}.P \\ S & ::= & \mathbf{0} \mid P{:}\langle \mathbf{s}, \ell\rangle \mid R{:}\langle \mathbf{s}\rangle \mid S_1 \,|\, S_2 \mid S\backslash L \end{array}$$

where $L \subseteq \mathbf{Ch}$, I is an index set, $p_i \in (0,1]$ with $\sum_{i\in I} p_i = 1$, $e_1, \ldots, e_n$, is a set of logical expressions such that $e_1 \vee \ldots \vee e_n = \mathsf{true}$, C ranges over a set of process constants $\mathcal{C}$, each with an associated definition of the form $C \overset{\text{def}}{=} P$, and

$$\eta ::= a \mid \overline{a} \mid go\,\ell \mid \surd.$$

Beginning with the individual level, P can be one of the following: Process $\mathbf{0}$ represents the inactive individual, that is, an individual who has ceased to exist. Process $\sum_{i\in I}\eta_i.P_i$ describes the nondeterministic choice between a set of action-prefixed processes: it can execute any of the activities η_i and proceed as the respective P_i. We write $\eta_1.P_1 + \eta_2.P_2$ to denote the binary form of this operator. In turn, an activity η can be an input action on a channel a, written simply as a, a complementary output action on a channel a, written as $\overline{a}$, a movement action with destination ℓ, $go\,\ell$, or the time-passing action, written as $\surd$. Actions of the form a, and $\overline{a}$, $a \in \mathbf{Ch}$, are used to model arbitrary activities performed by an individual; for instance, eating, predation and reproduction. The tick action $\surd$ measures a tick on a global clock and is used to separate the rounds of an individual's behavior. Essentially, 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 and proceed to their next round.

Process $\bigoplus_{i\in I} p_i{:}P_i$ represents the probabilistic choice between processes P_i, $i \in I$. The process randomly selects an index $i \in I$ with probability p_i, and then evolves to process P_i. We write $p_1{:}P_1 \oplus p_2{:}P_2$ for the binary form of this operator. 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. Note that this choice is deterministic. Finally, process constants provide a mechanism for including recursion in the calculus.

Moving on to the species level, we employ the special *species* process R defined as $!rep.P$. This process is a replicated process which may always receive input through channel rep and create new instances of process P, where P is a new individual of species R. Such inputs will be provided by individuals in the phase of reproduction via the complementary action $\overline{rep}$.

Finally, population systems are built by composing in parallel located individuals and species. An individual is defined as $P{:}\langle \mathbf{s}, \ell\rangle$, where $\mathbf{s}$ and ℓ are the species and the location of the individual, respectively. A species is given by $R{:}\langle \mathbf{s}\rangle$, where $\mathbf{s}$ is the name of the species. Finally, $S\backslash L$ models the restriction of the use of channels in set L within S. As a syntactic shorthand, we write $P{:}\langle \mathbf{s}, \ell, n\rangle$ for the parallel composition of n copies of process $P{:}\langle \mathbf{s}, \ell\rangle$.

Example 1. We consider a simplification of the model presented in [44] which studies the reproduction of the parasitic *Varroa mite*. This mite usually attacks honey bees and it has a pronounced impact on the beekeeping industry. In this system, a set of individuals reside 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 offspring only if they have exclusive use of a location. After reproduction the offspring disperse and continue indefinitely with the same behavior. In PALPS, we may model the described species $\mathbf{s}$ as $R \overset{\mathrm{def}}{=} !rep.P_0$, where

$$\begin{aligned} P_0 &\overset{\mathrm{def}}{=} \sum_{\ell \in \mathbf{Nb}(\mathsf{myloc})} \frac{1}{4} : go\,\ell.\surd.\mathsf{cond}\ (\mathbf{s}@\mathsf{myloc} = 1 \rhd P_1;\ \mathsf{true} \rhd \surd.P_0) \\ P_1 &\overset{\mathrm{def}}{=} p{:}\overline{rep}.\surd.P_0 \oplus (1-p){:}\overline{rep}.\overline{rep}.\surd.P_0 \end{aligned}$$

According to the definition, during the dispersal phase, an individual moves to a neighboring location which is chosen probabilistically among the four neighboring locations on the lattice of the individual. Subsequently, the conditional construct allows to determine the exclusive use of a location by an individual. The special label myloc is used to denote the actual location of an individual once the individual is instantiated within a system definition. In case of exclusive usage, the flow of control process according to P_1 which models the probabilistic production of one or two children of the species. A system that contains two individuals at a location ℓ and one at location ℓ' can be modeled as

$$System \overset{\mathrm{def}}{=} (P_0{:}\langle \mathbf{s}, \ell, 2\rangle | P_0{:}\langle \mathbf{s}, \ell'\rangle | R{:}\langle \mathbf{s}\rangle)\backslash\{rep\}.$$

Let us now extend the example into a two-species system. In particular, consider a competing species $\mathbf{s}'$ of the Varroa mite, such as the pseudo-scorpion, which preys on $\mathbf{s}$. To model this, we may define the process $R' \overset{\mathrm{def}}{=} !rep'.Q_0$, where

$$\begin{aligned} Q_0 &\overset{\mathrm{def}}{=} \overline{prey}.\surd.Q_1 + \surd.Q_2 \\ Q_1 &\overset{\mathrm{def}}{=} \overline{rep'}.\surd.Q_0 \\ Q_2 &\overset{\mathrm{def}}{=} \overline{prey}.\surd.Q_1 + \surd.\mathbf{0} \end{aligned}$$

An individual of species s' looks for a prey. If it succeeds in locating one, then it produces an offspring. If it fails for two consecutive time units it dies.

To implement the possibility of preying on the side of species $\mathbf{s}$, the definition must be extended by introducing the complementary input actions on channel *prey* at the appropriate places:

$$P_0 \overset{\text{def}}{=} \sum_{\ell \in \mathbf{Nb}(\mathsf{myloc})} \frac{1}{4} : (go\,\ell.\surd.\mathsf{cond}\ (\mathbf{s}@\mathsf{myloc} = 1 \rhd P_1;\ \mathsf{true} \rhd \surd.P_0) + prey.\mathbf{0})$$

$$P_1 \overset{\text{def}}{=} p{:}(\overline{rep}.\surd.P_0 + prey.\mathbf{0}) \oplus (1-p){:}(\overline{rep}.\overline{rep}.\surd.P_0 + prey.\mathbf{0})$$

2.2 The Semantics

The semantics of PALPS is defined in terms of a *structural operational semantics* given at the level of configurations of the form (E, S), where E is an *environment* and S is a population system. The environment E is an entity of the form $E \subset \mathbf{Loc} \times \mathbf{S} \times \mathbb{N}$, where each pair ℓ and $\mathbf{s}$ is represented in E at most once and where $(\ell, s, m) \in E$ denotes the existence of m individuals of species s at location ℓ. The environment E plays a central role in evaluating expressions.

The satisfaction relation for logical expressions $\models$ is defined inductively on the structure of a logical expression, as shown in Table 1. It depends on the evaluation function for arithmetic expressions $\mathsf{val}\,(E, w)$ defined in Table 2.

Table 1: **The satisfaction relations for logical and arithmetic expressions**

$E \models \mathsf{true}$	always	
$E \models \neg e$	if and only if	$\neg(E \models e)$
$E \models e_1 \wedge e_2$	if and only if	$E \models e_1 \wedge E \models e_2$
$E \models w \bowtie e$	if and only if	$\mathsf{val}\,(E, w) \bowtie e$

Before we proceed to the semantics we define some additional operations on environments that we will use in the sequel:

Definition 1. Consider an environment E, a location ℓ and a species $\mathbf{s}$.

Table 2: **The evaluation relation for arithmetic expressions**

$\mathsf{val}\,(E, c)$	$=$	c
$\mathsf{val}\,(E, \psi@\ell)$	$=$	ψ_ℓ
$\mathsf{val}\,(E, \mathbf{s}@\ell)$	$=$	$n, (\ell, \mathbf{s}, n) \in E$
$\mathsf{val}\,(E, \mathbf{s}@\ell)$	$=$	$0, (\ell, \mathbf{s}, n) \notin E$
$\mathsf{val}\,(E, @\ell)$	$=$	$\sum_{\mathbf{s}\in\mathbf{S}} n_\mathbf{s}, (\ell, \mathbf{s}, n_\mathbf{s}) \in E$
$\mathsf{val}\,(E, \mathbf{op}_1(w))$	$=$	$\mathbf{op}_1(\mathsf{val}\,(E, w))$
$\mathsf{val}\,(E, \mathbf{op}_2(w_1, w_2))$	$=$	$\mathbf{op}_2(\mathsf{val}\,(E, w_1), \mathsf{val}\,(E, w_2))$

- $E \oplus (\mathbf{s}, \ell)$ increases the count of individuals of species $\mathbf{s}$ at location ℓ in environment E by 1:

$$E \oplus (\mathbf{s}, \ell) = \begin{cases} E' \cup \{(\ell, \mathbf{s}, m+1)\} & \text{if } E = E' \cup \{(\ell, \mathbf{s}, m)\} \text{ for some } m \\ E \cup \{(\ell, \mathbf{s}, 1)\} & \text{otherwise} \end{cases}$$

- $E \ominus (\mathbf{s}, \ell)$ decreases the count of individuals of species $\mathbf{s}$ at location ℓ in environment E by 1:

$$E \ominus (\mathbf{s}, \ell) = \begin{cases} E' \cup \{(\ell, \mathbf{s}, m-1)\} & \text{if } E = E' \cup \{(\ell, \mathbf{s}, m)\}, m > 1 \\ E' & \text{if } E = E' \cup \{(\ell, \mathbf{s}, 1)\} \\ \bot & \text{otherwise} \end{cases}$$

We may now define the semantics of PALPS, presented in Tables 3 and 4, and given in terms of two transition relations: the non-deterministic relation $\longrightarrow_n$ and the probabilistic relation $\longrightarrow_p$. A transition of the form $(E, S) \xrightarrow{\mu}_n (E', S')$ means that a configuration (E, S) may execute action μ and become (E', S'). A transition of the form $(E, S) \xrightarrow{w}_p (E', S')$ means that a configuration (E, S) may evolve into configuration (E', S') with probability w. Whenever the type of the transition is irrelevant to the context, we write $(E, S) \xrightarrow{\alpha} (E', S')$ to denote either $(E, S) \xrightarrow{\mu}_n (E', S')$ or $(E, S) \xrightarrow{w}_p (E', S')$. Action μ appearing in the non-deterministic relation may have one of the following forms:

- $a@\ell$ and $\overline{a}@\ell$ denote the execution of a and $\overline{a}$ respectively at location ℓ.
- $\tau_{a@\ell}$ denotes an internal action that has taken place on channel a at location ℓ. This may arise when two complementary actions take place

at the same location ℓ or when a move action take place. Note that recording the location and the channel of a synchronization in internal actions, is a point on which PALPS departs from other process calculi featuring binary communication. The intention behind this design decision is to enable to observe and reason about behavior of a system pertaining to synchronizations.
- $\surd$ denotes the time passing action.

Rules for individuals. The rules of Table 3 prescribe the semantics of located individuals in isolation. The first four rules define non-deterministic transitions. The fifth axiom defines a probabilistic transition, and the last two rules refer to both the non-deterministic and the probabilistic case. All rules are concerned with the evolution of the individual in question and the effect of this evolution to the system's environment. A key issue in the definition of the rules is to preserve the compatibility of P and E as transitions are executed. We consider each rule separately:

- Axiom (Nil) specifies that the $\mathbf{0}$ process may execute the time consuming action $\surd$. This axiom allows for time-progress in a system with inactive individuals.
- Axiom (Tick) specifies that a $\surd$-prefixed process will execute the time consuming action $\surd$ and then proceed as P. The state of the new environment depends on the state of P. If $P = \mathbf{0}$ then the individual has terminated its computation and it is removed from E (see the definition of $E^{P,\mathbf{s},\ell}$). If $P \neq \mathbf{0}$ then E remains unchanged.
- Axiom (Act) specifies that $\eta.P$ executes action $\eta@\ell$ and evolves to P. Note that the action is decorated by the location of the individual executing the transition to enable synchronization of the action with complementary actions taking place at the same location (see rule (Par2), Table 4). This axiom excludes the cases of $\eta = go\,\ell$ and $\eta = \surd$ which are treated separately.
- According to Axiom (Go), an individual may change its location. This gives rise to action $\tau_{go@\ell}$ and has the expected effect on the environment E.
- Rule (NSum) describes the behavior of a nondeterministic choice: any of the available summands may be selected and executed.
- Rule (PSum) expresses the semantics of probabilistic choice: a process is chosen probabilistically leading to the appropriate continuation. If

Table 3: **Transition rules for individuals**

(Nil)	$(E,\mathbf{0}{:}\langle s,\ell\rangle) \xrightarrow{\surd}_n (E,\mathbf{0}{:}\langle s,\ell\rangle)$	
(Tick)	$(E,\surd.P{:}\langle s,\ell\rangle) \xrightarrow{\surd}_n (E^{P,\mathbf{s},\ell}, P{:}\langle s,\ell\rangle)$	
(Act)	$(E,\eta.P{:}\langle s,\ell\rangle) \xrightarrow{\eta@\ell}_n (E^{P,\mathbf{s},\ell}, P{:}\langle s,\ell\rangle)$	$\eta \neq go\,\ell', \surd$
(Go)	$(E, go\,\ell'.P{:}\langle s,\ell\rangle) \xrightarrow{\tau_{go@\ell}}_n (E^{P,\mathbf{s},\ell,\ell'}, P{:}\langle s,\ell'\rangle)$	$(\ell,\ell') \in \mathbf{Nb}$
(NSum)	$\dfrac{(E,\eta_i.P_i{:}\langle s,\ell\rangle) \xrightarrow{\mu}_n (E',P_i{:}\langle s,\ell'\rangle)}{(E,\sum_{i\in I}\eta_i.P_i{:}\langle s,\ell\rangle) \xrightarrow{\mu}_n (E',P_i{:}\langle s,\ell'\rangle)}$	
(PSum)	$(E,\dot{\sum}_{i\in I} p_i{:}P_i{:}\langle s,\ell\rangle) \xrightarrow{p_i}_p (E^{P_i,\mathbf{s},\ell}, P_i{:}\langle s,\ell\rangle)$	
(Const)	$\dfrac{(E,P{:}\langle s,\ell\rangle) \xrightarrow{\alpha} (E',P'{:}\langle s,\ell'\rangle)}{(E,C{:}\langle s,\ell\rangle) \xrightarrow{\alpha} (E',P'{:}\langle s,\ell'\rangle)}$ $\quad C \overset{\text{def}}{=} P{:}\langle s,\ell\rangle$	
(Cond)	$\dfrac{(E,P_i{:}\langle s,\ell\rangle) \xrightarrow{\alpha} (E',P_i'{:}\langle s,\ell'\rangle), E{\models}e_i{\downarrow}\ell, E\not\models e_j{\downarrow}\ell, j<i}{(E,\mathsf{cond}\ (e_1 \rhd P_1,\ldots,e_n \rhd P_n){:}\langle s,\ell\rangle) \xrightarrow{\alpha} (E',P_i'{:}\langle s,\ell'\rangle)}$	

$$\text{where } E^{P,\mathbf{s},\ell} = \begin{cases} E \ominus (\mathbf{s},\ell) & \text{if } P = \mathbf{0} \\ E & \text{otherwise} \end{cases}$$

$$E^{P,\mathbf{s},\ell,\ell'} = ((E \ominus (s,\ell)) \oplus (s,\ell'))^{P,\mathbf{s},\ell'}$$

the resulting state of the individual, namely P_i, is equal to $\mathbf{0}$, then the individual is removed from the environment E.

- Rule (Const) expresses the semantics of process constants.
- Finally, rule (Cond) stipulates that a conditional process may perform an action of continuation P_i assuming that $e_i{\downarrow}\ell$ evaluates to true and all $e_j \downarrow \ell$, $j < i$ evaluate to false. Note that we write $e \downarrow \ell$ for the expression e with all occurrences of myloc substituted by location ℓ.

Rules for systems. We may now move on to Table 4 which defines the semantics of system-level operators. The first two rules define the semantics for the replication operator, the next five rules define the semantics of the parallel composition operator, and the last rule deals with the restriction operator.

According to axioms (R_Tick) and (R_Rep), a species process may idle or

Table 4: **Transition rules for systems**

(R_Tick)	$\dfrac{R = !rep.P{:}\langle \mathbf{s} \rangle}{(E,R) \xrightarrow{\surd}_n (E,R)}$
(R_Rep)	$\dfrac{R = !rep.P{:}\langle \mathbf{s} \rangle,\ \ell \in \mathbf{Loc}}{(E,R) \xrightarrow{rep@\ell}_n (E \oplus (\mathbf{s},\ell), P{:}\langle \mathbf{s},\ell \rangle \mid R)}$
(Par1)	$\dfrac{(E,S_1) \xrightarrow{\mu}_n (E',S_1'), (E,S_2) \not\longrightarrow_p, \mu \neq \surd}{(E,S_1 \mid S_2) \xrightarrow{\mu}_n (E', S_1' \mid S_2)}$
(Par2)	$\dfrac{(E,S_1) \xrightarrow{a@\ell}_n (E_1,S_1'), (E,S_2) \xrightarrow{\overline{a}@\ell}_n (E_2,S_2')}{(E,S_1 \mid S_2) \xrightarrow{\tau_{a@\ell}}_n (E \otimes (E_1,E_2), S_1' \mid S_2')}$
(Par3)	$\dfrac{(E,S_1) \xrightarrow{w_1}_p (E_1,S_1'), (E,S_2) \xrightarrow{w_2}_p (E_2,S_2')}{(E,S_1 \mid S_2) \xrightarrow{w_1 \cdot w_2}_p (E \otimes (E_1,E_2), S_1' \mid S_2')}$
(Par4)	$\dfrac{(E,S_1) \xrightarrow{w}_p (E',S_1'), (E,S_2) \not\longrightarrow_p}{(E,S_1 \mid S_2) \xrightarrow{w}_p (E', S_1' \mid S_2)}$
(Time)	$\dfrac{(E,S_1) \xrightarrow{\surd}_n (E_1,S_1'), (E,S_2) \xrightarrow{\surd}_n (E_2,S_2')}{(E,S_1 \mid S_2) \xrightarrow{\surd}_n (E \otimes (E_1,E_2), S_1' \mid S_2')}$
(Res)	$\dfrac{(E,S) \xrightarrow{\alpha} (E',S'), \alpha \notin \{a@\ell, \overline{a}@\ell \mid a \in L\}}{(E,S \backslash L) \xrightarrow{\alpha} (E',S') \backslash L}$

it may engage in action $rep@\ell$ for any location ℓ and create a new individual P of species $\mathbf{s}$ at location ℓ.

Rules (Par1) - (Par4) specify the semantics of parallel composition. Note that the symmetric versions of these rules are omitted. According to (Par1), if a component may execute a non-deterministic transition and no probabilistic transition is enabled by the other component (denoted by $(E,S_2) \not\longrightarrow_p$), then the transition may take place. If the parallel components may execute complementary actions on some channel a and some location ℓ, then they may synchronize with each other producing action $\tau_{a@\ell}$ (rule (Par2)). If both components may execute probabilistic transitions then they may proceed together with probability the product of the two distinct probabilities (rule (Par3)). If exactly one of them enables a probabilistic transition then this transition takes precedence over any non-deterministic transitions of the other component (rule (Par4)).

Note that in case that the components proceed simultaneously then the environment of the resulting configuration should take into account the changes applied in both of the constituent transitions (rules (Par2), (Par3) and (Time)). This is implemented by $E \otimes (E_1, E_2)$ as follows:

$$\begin{aligned} E \otimes (E_1, E_2) \quad = \quad & \{(\ell, \mathbf{s}, m + i_1 + i_2) \mid (\ell, \mathbf{s}, m) \in E, \\ & (\ell, \mathbf{s}, m + i_1) \in E_1, (\ell, \mathbf{s}, m + i_2) \in E_2, i_1, i_2 \in \mathbb{Z}\} \end{aligned}$$

Rule (Time) defines that parallel processes must synchronize on $\surd$ actions. This allows one tick of time to pass and all processes to proceed to their next round. Finally, rule (Res) defines the semantics of the restriction operator in the usual way.

Initial configuration. Based on this machinery, the semantics of a system S is obtained by applying the semantic rules to the initial configuration. The initial configuration, (E, S), is such that $(\ell, \mathbf{s}, m) \in E$ if and only if S contains exactly m individuals of species $\mathbf{s}$ located at ℓ of the form $P{:}\langle \mathbf{s}, \ell \rangle$, where $P \neq \mathbf{0}$. In general, we say that E is *compatible* with S whenever $(\ell, \mathbf{s}, m) \in E$ if and only if S contains exactly m individuals of species $\mathbf{s}$ located at ℓ. It is possible to prove that the defined semantics preserves compatibility of configurations [2]:

Lemma 1. *Whenever* $(E, S) \xrightarrow{\alpha} (E', S')$ *and* E *is compatible with* S*, then* E' *is also compatible with* S'*.*

3 Examples

During the last few decades, the theory of metapopulations has been an active field of research in Ecology and it has been extensively studied by conservation scientists and landscape ecologists to analyze the behavior of interacting populations and to determine how system parameters may influence various aspects of these systems such as local and global population persistence and species evolution. Population dispersal is one such phenomenon that has been of special interest to ecologists. It affects the long-term persistence of populations, the coexistence of species and genetic differentiation between subpopulations and understanding this process is essential for obtaining a good understanding of the behavior of metapopulations. The evolution of dispersal has received much attention by scientists and it has been studied in connection to various parameters such as the

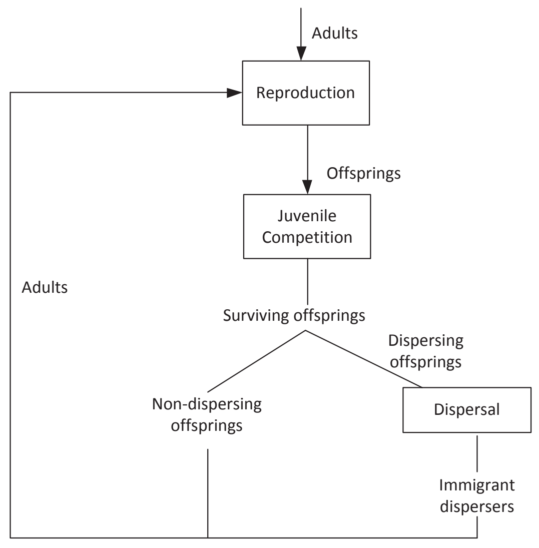

Figure 1: The sequence of events in the lifetime of a dispersing species

connectivity of the habitat on which a metapopulation exists, patch quality and local dynamics.

In this section, we describe two examples relating to metapopulation dispersal through which we illustrate how our calculus can be used to construct models of this phenomenon.

Example 2. This example is motivated by the spatially-explicit individual-based model of [46]. In this work the authors construct a fairly simple model of metapopulation dispersal which departs from previous works in that, unlike previous models of metapopulation dispersal which tended to be deterministic and at the level of population densities, the model constructed is stochastic and individual-based.

According to this study, a set of genotypes co-exist within a habitat and differ only in their propensity to disperse. The metapopulation is composed of $n \times n$ subpopulations inhabiting a set of patches arranged on a square lattice with cyclic boundaries, so that individuals leaving the “top” or “right-side” of the world reappear on the “bottom” or “left-side” respectively and vice versa. Each patch is associated with a so-called patch quality related to the capacity of the patch.

The behavior of an individual of the genotypes under study is illustrated diagrammatically in Figure 1. To begin with, an adult individual produces λ descendants. Subsequently, a phase of competition takes place between the juveniles of the population of which a fraction survives. Each surviving offspring may disperse according to a probability of dispersal distinct to its genotype. In case it disperses, the neighboring patch it moves to is selected with equal probability among all neighbors. We point out that the percentage of offspring surviving juvenile competition at patch ℓ is given by $\gamma_\ell = (1 + \alpha_\ell \cdot N_\ell)^\beta$, where α_ℓ is the measure of the patch quality, N_ℓ is the number of individuals residing at patch ℓ and β is a constant that relates to the degree of competition.

This metapopulation can be modeled in PALPS as follows. We consider the set of locations (i, j), $1 \le i, j \le n$, where two locations (i, j) and (k, l) are neighbors if they are adjacent on the grid. Further, we use the location attribute α_ℓ as a measure of the quality of the patch at ℓ. Then, genotype i with some constant probability of dispersal p_i and $\lambda = 3$ can be defined as the species process $R_i = !rep_i.J_i$, where

$$\begin{array}{lcll} A_i & \stackrel{\mathrm{def}}{=} & \overline{rep}_i.\overline{rep}_i.\overline{rep}_i.0 & \text{Adult Individual} \\ J_i & \stackrel{\mathrm{def}}{=} & q_i{:}S_i \oplus (1-q_i){:}\mathbf{0} & \text{Juvenile} \\ S_i & \stackrel{\mathrm{def}}{=} & p_i{:}D_i \oplus (1-p_i){:}\surd.A_i & \text{Surviving Juvenile} \\ D_i & \stackrel{\mathrm{def}}{=} & \sum_{\ell \in \mathbf{Nb}(\mathsf{myloc})} \frac{1}{4} : go\,\ell.\surd.A_i & \text{Dispersing Juvenile} \end{array}$$

and q_i the probability of survival of juvenile competition is given by $q_i = (1 + \alpha_\ell \cdot @\ell)^\beta$. Then a system can be modeled as the composition of the various genotypes as well as the individuals of the initial population under study:

$$System \stackrel{\mathrm{def}}{=} [(R_1{:}\langle 1\rangle \mid \dots \mid R_k{:}\langle k\rangle \mid \prod_{1 \le i \le k, \ell \in \mathbf{Loc}} A_i{:}\langle i, \ell, m_i\rangle)\backslash\{rep_1, \dots rep_k\}.$$

Analysis in this model may focus on the effect that the dispersal rates, the degree of competition and/or the patch quality may have on population dispersal.

Example 3. As another more complex example, let us consider a model of wood thrush dispersal, initially proposed in [47] and expanded upon in [33]. This model considers three types of birds: adult breeders, adult floaters, and juveniles which are birds in their first year of life.

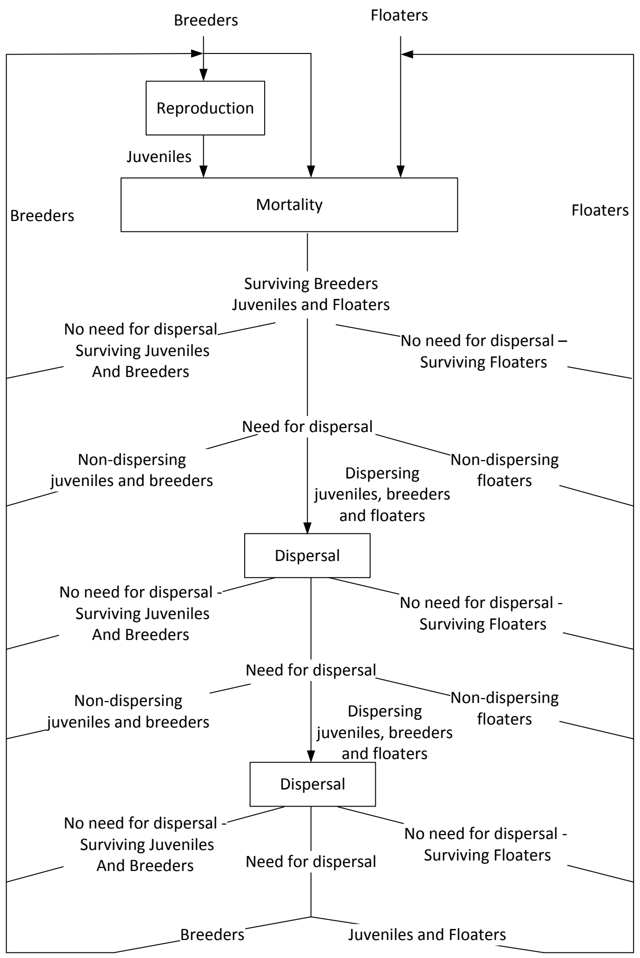
Breeders
Floaters
Reproduction
Juveniles
Mortality
Breeders
Floaters
Surviving Breeders Juveniles and Floaters
No need for dispersal Surviving Juveniles And Breeders
No need for dispersal – Surviving Floaters
Need for dispersal
Non-dispersing juveniles and breeders
Dispersing juveniles, breeders and floaters
Non-dispersing floaters
Dispersal
No need for dispersal - Surviving Juveniles And Breeders
No need for dispersal - Surviving Floaters
Need for dispersal
Non-dispersing juveniles and breeders
Dispersing juveniles, breeders and floaters
Non-dispersing floaters
Dispersal
No need for dispersal - Surviving Juveniles And Breeders
No need for dispersal - Surviving Floaters
Need for dispersal
Breeders
Juveniles and Floaters

Figure 2: A cycle in the lifetime of the metapopulation

According to this model, adult breeders produce an offspring at a rate dictated by various system parameters such as clutch size, nest predation and parasitism rates which we denote as r_b. Following reproduction, each individual has a probability of dying before the next time step which is higher in juveniles and adult floaters in comparison to adult breeders. We write q_b, q_j and q_f for the mortality rates of breeders, juveniles and floaters, respectively. If following mortality a habitat patch has more birds than its capacity allows, then dispersal will occur according to a probability determined by the size of the patch and the distance between neighboring patches. This probability is higher in floaters and juveniles in comparison to adult breeders who exhibit a high site fidelity. We write p_b, p_j and p_f for the dispersion rates of breeders, juveniles and floaters, respectively. If a bird reaches a patch with available capacity then it will settle. If not, then it will either attempt to disperse to another patch or it will become a floater depending on whether it has reached its maximum number of dispersal events. Once dispersal has occurred, the juveniles become adults and the model begins another cycle. This sequence of events in the behavior of the populations is presented diagrammatically in Figure 2.

This metapopulation can be modeled in PALPS as follows. We consider the set of locations **Loc** and an associated predefined neighbor function. We also assume the existence of a set of probabilities $\{p_{i,j}\}_{i,j\in\mathbf{Loc}}$ where $p_{i,j}$ represents the probability of dispersal from patch i to patch j. Finally, we introduce the location attribute c_ℓ as a measure of the capacity of patch ℓ. Then, the wood thrush species can be modeled by the process $R =!rep.Juv$, where the behavior of a juvenile individual J_i is described by the following equations:

$$
\begin{array}{lcll}
Juv & \stackrel{\text{def}}{=} & q_j : JC_0 \oplus (1-q_j) : \mathbf{0} & \text{Juvenile survival} \\
JC_0 & \stackrel{\text{def}}{=} & \mathsf{cond}\ (@\mathsf{myloc} > c_{\mathsf{myloc}} \rhd JD_0, \mathsf{true} \rhd \surd.AB) & \text{Check patch capacity} \\
JD_0 & \stackrel{\text{def}}{=} & p_j : JA_1 \oplus (1-p_j) : \surd.AB & \text{Decide whether to disperse} \\
JA_1 & \stackrel{\text{def}}{=} & \sum_{\ell\in\mathbf{Nb}(\mathsf{myloc})} p_{\mathsf{myloc},\ell} : go\,\ell.JC_1 & \text{Dispersal attempt 1} \\
JC_1 & \stackrel{\text{def}}{=} & \mathsf{cond}\ (@\mathsf{myloc} > c_{\mathsf{myloc}} \rhd JD_1, \mathsf{true} \rhd \surd.AB) & \text{Check patch capacity} \\
JD_1 & \stackrel{\text{def}}{=} & p_j : JA_2 \oplus (1-p_j) : \surd.AB & \text{Decide whether to disperse} \\
JA_2 & \stackrel{\text{def}}{=} & \sum_{\ell\in\mathbf{Nb}(\mathsf{myloc})} p_{\mathsf{myloc},\ell} : go\,\ell.JC_2 & \text{Dispersal attempt 2} \\
JC_2 & \stackrel{\text{def}}{=} & \mathsf{cond}\ (@\mathsf{myloc} > c_{\mathsf{myloc}} \rhd \surd.Fl, \mathsf{true} \rhd \surd.AB) & \text{Become floater or adult}
\end{array}
$$

AB	$\stackrel{\text{def}}{=}$	$r_b : \overline{rep}.BS \oplus (1 - rb) : BS$	Breeder reproduction
BS	$\stackrel{\text{def}}{=}$	$q_b : BC_0 \oplus (1 - q_b) : \mathbf{0}$	Breeder survival
BC_0	$\stackrel{\text{def}}{=}$	$\mathsf{cond}\ (@\mathsf{myloc} > c_{\mathsf{myloc}} \rhd BD_0, \mathsf{true} \rhd \surd.AB)$	Check patch capacity
BD_0	$\stackrel{\text{def}}{=}$	$p_b : BA_1 \oplus (1 - p_b) : \surd.AB$	Decide whether to disperse
BA_1	$\stackrel{\text{def}}{=}$	$\sum_{\ell \in \mathbf{Nb}(\mathsf{myloc})} p_{\mathsf{myloc},\ell} : go\,\ell.BC_1$	Dispersal attempt 1
BC_1	$\stackrel{\text{def}}{=}$	$\mathsf{cond}\ (@\mathsf{myloc} > c_{\mathsf{myloc}} \rhd BD_1, \mathsf{true} \rhd \surd.AB)$	Check patch capacity
BD_1	$\stackrel{\text{def}}{=}$	$p_b : BA_2 \oplus (1 - p_b) : \surd.AB$	Decide whether to disperse
BA_2	$\stackrel{\text{def}}{=}$	$\sum_{\ell \in \mathbf{Nb}(\mathsf{myloc})} p_{\mathsf{myloc},\ell} : go\,\ell.BC_2$	Dispersal attempt 2
BC_2	$\stackrel{\text{def}}{=}$	$\mathsf{cond}\ (@\mathsf{myloc} > c_{\mathsf{myloc}} \rhd \surd.Fl, \mathsf{true} \rhd \surd.AB)$	Floater or adult
Fl	$\stackrel{\text{def}}{=}$	$q_f : FC_0 \oplus (1 - q_f) : \mathbf{0}$	Floater survival
FC_0	$\stackrel{\text{def}}{=}$	$\mathsf{cond}\ (@\mathsf{myloc} > c_{\mathsf{myloc}} \rhd FD_0, \mathsf{true} \rhd \surd.Fl)$	Check patch capacity
FD_0	$\stackrel{\text{def}}{=}$	$p_f : FA_1 \oplus (1 - p_f) : \surd.Fl$	Decide whether to disperse
FA_1	$\stackrel{\text{def}}{=}$	$\sum_{\ell \in \mathbf{Nb}(\mathsf{myloc})} p_{\mathsf{myloc},\ell} : go\,\ell.FC_1$	Dispersal attempt 1
FC_1	$\stackrel{\text{def}}{=}$	$\mathsf{cond}\ (@\mathsf{myloc} > c_{\mathsf{myloc}} \rhd FD_1, \mathsf{true} \rhd \surd.Fl)$	Check patch capacity
FD_1	$\stackrel{\text{def}}{=}$	$p_f : FA_2 \oplus (1 - p_f) : \surd.Fl$	Decide whether to disperse
FA_2	$\stackrel{\text{def}}{=}$	$\sum_{\ell \in \mathbf{Nb}(\mathsf{myloc})} p_{\mathsf{myloc},\ell} : go\,\ell.\surd.Fl$	Dispersal attempt 2

As before, the system can be modeled as the composition of the species as well as the various individuals that form the study:

$$\begin{aligned} System \;\stackrel{\text{def}}{=}\;& (R{:}\langle 1\rangle \mid \prod_{\ell \in \mathbf{Loc}} AB{:}\langle 1, \ell, n_b^\ell\rangle \mid \prod_{\ell \in \mathbf{Loc}} Juv{:}\langle 1, \ell, n_j^\ell\rangle \\ & \mid \prod_{\ell \in \mathbf{Loc}} Fl{:}\langle 1, \ell, n_f^\ell\rangle)\backslash\{rep\} \end{aligned}$$

Varying the model parameters (such as the habitat topology and the patch quality) may allow an analysis of the effects of the parameters on patch and metapopulation persistence.

4 Translating PALPS into PRISM

In this section we turn to the problem of model checking PALPS models. To begin with, we observe that the operational semantics of PALPS gives rise to transition systems that can be easily translated to Markov decision processes (MDPs). We recall that Markov decision processes are a type of transition systems that combine probabilistic and non-deterministic behavior. As such, model checking approaches that have been developed for MDPs can also be applied to PALPS models.

Given this observation, in this section we explore the possibility of employing the probabilistic model checker PRISM to perform analysis of the semantic models derived from PALPS processes. Specifically, we propose a translation scheme for encoding PALPS systems into PRISM models on which model checking can be performed. In the next section, we will illustrate our approach by performing analysis on the PALPS system presented in Example 1.

PRISM is a probabilistic model checker for Markov decision processes, discrete time Markov chains, and continuous time Markoc chains. Using PRISM it is also possible to generate random sample paths of execution for simulation. For our study we are interested in the MDP support of the tool which supports model checking though probabilistic computational time logic [1]. Specifically, we will propose a method for translating PALPS systems into the MDP subset of the PRISM language yielding models on which simulation and analysis can be performed via the PRISM model checker.

There are two alternatives for generating an input for PRISM given a PALPS model. First, it is possible to explicitly construct the Markov decision process corresponding to the model and directly specifying the transition matrix to the model checker. Second, it is possible to reproduce/translate the model in question into the PRISM language. According to PRISM's manual, the latter method is typically more efficient than the first alternative. This is because PRISM is a symbolic model checker and the underlying data structures used to represent the model, function better when there is a high-level structure and regularity to exploit. Therefore, we chose to encode PALPS models into the PRISM language. In the remainder of this section, we will give a brief presentation of the PRISM language, present an encoding of (a subset of) PALPS into PRISM and prove its correctness.

4.1 The PRISM Language

The PRISM language is a simple, state-based language, based on guarded commands. A PRISM model consists of a set of *modules* which can interact with each other on shared actions following the CSP-style of communication. Each module possesses a set of *local variables* which can be written by the module and read by all modules. In addition, there are *global variables* which can be read and written by all modules. The behavior of a module is described by a set of *guarded commands*. When modeling Markov decision processes, these commands take the form:

`[act] guard` p_1 `:` u_1 `+ ... +` p_m `:`u_m`;`

where `act` is an optional action label, guard is a predicate over the set of variables, $p_i \in (0, 1]$ and u_i are updates of the form:

`(`x'_1 `=` $u_{i,1}$`) & ... & (`x'_k `=` $u_{i,k}$`)`

where $u_{i,j}$ is a function over the variables. Intuitively, such an action is enabled in global state s if s satisfies `guard`. If a command is enabled then it may be executed in which case, with probability p_i, the update u_i is performed by setting the value of each variable x_j to $u_{i,j}(s)$ (where x'_j denotes the new value of variable x_j).

A model is constructed as the parallel composition of a set of modules. The semantics of a complete PRISM model is the parallel composition of all modules using the standard CSP parallel composition. This means that all the modules synchronize over all their common actions (i.e., labels). For a transition arising from synchronization between multiple processes, the associated probability is obtained by multiplying those of each component transition. Whenever, there is a choice of more than one commands, this choice is resolved non-deterministically. We refer the reader to [1] for the full description and the semantics of the PRISM language.

4.2 Encoding PALPS into the PRISM Language

As observed in [34], the main challenge of translating a CCS-like language (like PALPS) into PRISM is how to map binary CCS-style communication over channels to PRISM's multi-way (CSP style) communication. Our approach for dealing with this challenge, similarly to [34], is to introduce a distinct action for each possible binary, channel-based, communication which captures the channel as well as the sender/receiver pair. The two other actions in PALPS, namely the tick action and the movement action, can be easily handled as follows: In the case of the tick action, the encoding is straightforward as its semantical treatment in PALPS is via synchronous communication which can be represented directly in PRISM. In the case of movement actions, it is possible to introduce a PRISM command that performs the necessary updates on the state of the model.

To translate PALPS into the PRISM language, we translate each process into a module. The execution flow of a process is captured with the use of a local variable within the module whose value is updated in every command in such a way as to guide the computation through the states of the process.

Then, each possible construct of PALPS is modeled via a set of commands. For example, the probabilistic summation is represented by encoding the probabilistic choices into a PRISM guarded command. Non-deterministic choices are encoded by a set of simultaneously enabled guarded commands that capture all nondeterministic alternatives, whereas the conditional statement is modeled as a set of guarded commands where the guard of each command is determined by the expressions of the conditional process.

Unfortunately, the replication operator cannot be directly encoded into PRISM since the PRISM language does not support the dynamic creation of modules. To overcome this problem, we consider a bounded replication construct of the form $!^m P$ in which we specify the maximum number of P's, namely m, that the can be created during computation. We note that, in practice, the value of m can be selected by estimating a bound of the maximum size of the population, or it can be determined by the size of the state-space of the resulting model.

We now consider the main ideas of translating PALPS into the PRISM language via an example.

Example 4. Consider a habitat consisting of four patches $\{1, 2, 3, 4\}$, where $\mathbf{Nb}$ is the symmetric closure of the set $\{(1,2),(1,3),(2,4),(3,4)\}$. Furthermore, suppose that each location ℓ has an attribute c_ℓ determining the capacity of the location in terms of the number of individuals it may support. Let $\mathbf{s}$ be a species residing on this habitat defined according to the bounded replication R:

$$\begin{aligned}
R &\stackrel{\text{def}}{=} !^m rep.P \\
P &\stackrel{\text{def}}{=} \mathsf{cond}\ (s@\mathsf{myloc} > c@\mathsf{myloc} \rhd P_1, s@\mathsf{myloc} \leq c@\mathsf{myloc} \rhd P_2) \\
P_1 &\stackrel{\text{def}}{=} \sum_{\ell \in \mathbf{Nb}(\mathsf{myloc})} \frac{1}{2} : go\,\ell.\surd P \\
P_2 &\stackrel{\text{def}}{=} \overline{rep}.\surd.P
\end{aligned}$$

According to the definition, an individual of species $\mathbf{s}$ begins by checking whether the location has exceeded its capacity. If so then it migrates with equal probability to one of the neighboring locations, otherwise, it produces one offspring and it returns to its initial state. Now, consider a system initially consisting of two individuals, one at location 1 and one at location 4:

$$System \stackrel{\text{def}}{=} (P{:}\langle \mathbf{s}, 1\rangle \mid P{:}\langle \mathbf{s}, 4\rangle \mid R{:}\langle \mathbf{s}\rangle)\backslash\{rep\}$$

In order to translate *System* in the PRISM language we first need to encode global information relating to the system. This consists of four global variables that record the initial populations of each of the locations and the values of attributes c_ℓ. For simplicity, let us assume that $c = 2$ for all locations. We also include a global variable i that measures the inactive individuals still available to be triggered. Initially $i = m$.

```
const int c = 2;
global s1: [0,m+2] init 1;
global s2: [0,m+2] init 0;
global s3: [0,m+2] init 0;
global s4: [0,m+2] init 1;
global i: [0,m+2] init m;
```

We continue to model the two individuals $P{:}\langle \mathbf{s}, 1\rangle$ and $P{:}\langle \mathbf{s}, 4\rangle$. Each individuals will be described by a module. In Figure 3, we may see the translation of individual $P{:}\langle \mathbf{s}, 1\rangle$. We observe, that its species variable $sp1$ is set to 1, a constant that identifies the species, its location variable $loc1$ is set to 1 and variable $st1$, recording its state, is set to 1, the initial state of the module. Overall, the module has 5 different states. From state 1 two commands are enabled: one leads to state 2 and the other to state 4. From state 2, reproduction is implemented in two steps: in the first step the number of individuals of the current location is increased by one and $st1$ becomes 3, and in the second step, the offspring is created by synchronization of the module and an inactive individual module via action rep_1_i. Note that these two moves cannot be merged into one due to the restriction of PRISM that commands which synchronize with other modules (such as rep_1_i) cannot modify global variables (such as s_i). On the other hand, from state 4, a probabilistic transition is enabled during which the module changes its location (variable $loc1$) to one of the neighboring locations and updates the populations of the relevant locations accordingly. In both cases, the flow of control is determined by the assignment $st = 5$. Whenever $st = 5$, the module is willing to synchronize via action *tick*, which we assume to be an action shared by all modules and via which a tick on the global clock is measured. Then the module resumes the initial state ($st = 1$).

Individual $P{:}\langle \mathbf{s}, 4\rangle$ may be defined similarly. Note that the module encoding this process is identical to module $P1$ with the exception that we rename the variables and the initial value of the location variable.

This leaves us with the encoding of R, the component that implements replication of individuals. As we have already discussed, we achieve this via

```
module P1

st1 : [1..5] init 1;
loc1: [1..4] init 1;
const int sp1 = 1;

[] (st1=1)&(s1<c) -> (st1'=2);

[] (st1=2)&(loc1=1)&(i>0) -> (s1'=s1+1)&(i'=i-1)&(st1'=3);
[rep_1_3] (st1=3) -> (st1'=5); // Activate module 3
... // All activation possibilities are enumerated

[] (st1=2)&(loc1=2)&(i>0) -> ...
[] (st1=2)&(loc1=3)&(i>0) -> ...
[] (st1=2)&(loc1=4)&(i>0) -> ...

[] (st1=1)&(s1>=c) -> (st1'=4);
[] (st1=4)&(loc1=1) ->
           0.5:(st1'=5)&(loc'=2)&(s1'=s1-1)&(s2'=s2+1)
           + 0.5:(st1'=5)&(loc'=3)&(s1'=s1-1)&(s3'= s3+1);
[] (st1=4)&(loc1=2) -> ...
[] (st1=4)&(loc1=3) -> ...
[] (st1=4)&(loc1=4) -> ...

[tick] (st1=5) -> (st1'=1);
endmodule
```

Figure 3: PRISM code for an active individual

bounded replication which makes an assumption on the maximum number of new individuals that can be created in a system. Given this assumption, our model must be extended by an appropriate number of inactive individuals awaiting for a trigger via a *rep_i_j* action as illustrated in Figure 4.

Thus, the inactive individual modeled by module P3, awaits to synchronize with any of the remaining modules 1,...,max, in which case it inherits the location of the synchronizing module and it sets $st3 = 1$ so that it may begin to execute the code of an active individual, presented in Figure 3, with the variables appropriately renamed.

4.3 Formal Translation

In this section, we will formalize the intuitions of the previous example into a formal translation of PALPS into PRISM and we will prove its correctness.

```
module P3

st3 : [0..5] init 0;
loc3: [1..4] init 1;
const int sp3 = 1;

[tick] (st3=0) -> (st3'=0);
[rep_1_3] (st3=0) -> (st3'=1)&(loc3'=loc1);
...
[rep_max_3] (st3=0) -> (st3'=1)&(loc3'=loc_max);
...
// Here we append the code of an active P individual
// with the variables appropriately renamed

endmodule
```

Figure 4: PRISM code for an inactive individual

Consider a PALPS model. This consists of a set of locations, $L = \{1, \ldots, k\}$, a set of attributes, $\Theta = \{\theta_1, \ldots, \theta_m\}$, and a value of each attribute at each location, the neighborhood relation **Nb** and a process *System*. We assume, for the reasons already discussed in the previous example, that all replication processes are bounded and have the form $R = !^n rep.P$, thus allowing the creation of up to n individuals of the specific species. We also, assume that all *rep* channels are restricted within *System*. Then, the PRISM model is constructed as follows:

- For each species $\mathbf{s}$, the model contains the k global integer variables $s_1, \ldots, s_k$, capturing the number of individuals of species $\mathbf{s}$ for each of the locations. The variables are appropriately initialized based on the definition of *System*.

- For each attribute θ and each location ℓ, the model contains a constant that records the values of θ_ℓ.

- Each (active) process $P{:}\langle \mathbf{s}, \ell \rangle$ of *System* becomes a PRISM module with a constant $sp_P = \mathbf{s}$, a variable $loc_P = \ell$ and a variable st_P, with range $1, \ldots, |P|$, which records the current state of the individual and where $|P|$ is the number of states that P may evolve to. The body of the module is the translation of process P into guarded commands.

- Each species definition $R{:}\langle \mathbf{s} \rangle = !^n rep.P$ of *System* becomes a sequence of n PRISM modules, $P_{x+1}, \ldots, P_{x+n}$, where x is the number of individuals of species $\mathbf{s}$ in the initial state of *System*. In our model, we introduce a variable $i_{\mathbf{s}}$ that records the current number of inactive individuals of species $\mathbf{s}$:

$$i_s \quad : \quad [0..n] \;\; init \;\; n;$$

Each inactive module P_y possesses a constant $sp_y = \mathbf{s}$, a variable loc_y which is not initialized, and another variable st_y, with range $0, \ldots, |P|$, which corresponds to the current state of the individual and where $|P|$ is the number of states that P may evolve to. Note that $st_y = 0$ corresponds to the state where the inactive individual is awaiting activation by one of the active individuals of species $\mathbf{s}$. To capture this, we include the following commands:

```
module $P_y$
$st_y : [0..|P|]$ init 0;
$loc_y : [1..m]$;
const int $s_y = s$;
$[tick](st_y = 0) \longrightarrow (st'_y = 0)$
$[rep_{1,y}](st_y = 0) \longrightarrow (st'_y = 1)\&(loc_y = loc_1)$;
...
$[rep_{x+n,y}](st_y = 0) \longrightarrow (st'_y = 1)\&(loc_y = loc_x)$;
...
//Here we append the translation of P
endmodule
```

Thus, P_y may be activated by any of the modules $P_1, \ldots, P_{x+n}$ and then proceed according to the translation of process P into guarded commands.

We now continue to describe how a process at the individual level of PALPS can be translated to a sequence of PRISM commands. We denote the translation of a process P as $[\![P]\!]$ and we define it inductively on the structure of P. Note that, for convenience, we write s_Q to for the state (integer value) associated with process Q. Finally, in the translations below, we assume that we are working within a module with identifier x and variables st, and loc.

Case 1: $Q = go\,l.P$. We translate the process by including the command

$$[]\qquad (st = s_Q)\&(loc = a) \longrightarrow (st' = s_P)\&(loc' = l)\&(st'_a = st_a - 1)\&(st'_l = st_l + 1);$$

for each $(a, l) \in \mathbf{Nb}$ and then appending the translation of P.

Case 2: $Q = a.P$. We translate the process by including the command

$$[a_{y,x}]\qquad (st = s_Q)\&(loc = loc_y) \longrightarrow (st' = s_P);$$

for each location l and each module y that may perform an output on channel a. We then append the translation of P.

Case 3: $Q = \overline{a}.P$. We translate the process by including the command

$$[a_{x,y}]\qquad (st = s_Q)\&(loc = loc_y) \longrightarrow (st' = s_P);$$

for each location l and each module y that may perform an input on channel a. We then append the translation of P.

Case 4: $Q = \surd.P$. We translate the process by including the command

$$[tick]\qquad (st = s_Q) \longrightarrow (st' = s_P);$$

and appending the translation of P.

Case 5: $Q = rep_s.P$. We translate the process by including the command

$$[rep_{y,x}]\qquad (st = s_Q) \longrightarrow (st' = s_P)\&(loc' = loc_y);$$

for each module y that may output on channel rep. We then append the translation of P.

Case 6: $Q = \overline{rep_{\mathbf{s}}}.P$. We translate the process by including the commands

$$\begin{array}{rl} [] & (st = s_Q)\&(i_s > 0) \longrightarrow (i'_s = i_s - 1)\&(st' = st_{Q_2}); \\ [rep_{x,y}] & (st = st_{Q_2}) \longrightarrow (st' = s_P); \end{array}$$

for each inactive module y of species $\mathbf{s}$, and then appending the translation of P.

Case 7: $Q = \alpha_1.P_1 + \alpha_2.P_2$. We translate the process by computing the translations $[\![\alpha_1.P_1]\!]$ and $[\![\alpha_2.P_2]\!]$ and replacing all commands of the form

$$[act]\qquad (st = st_{a_1.P_1})\&guard \longrightarrow updates;$$

by

$$[act]\qquad (st = st_Q)\&guard \longrightarrow updates$$

and similarly for the commands of $[\![\alpha_2.P_2]\!]$.

Case 8: $Q = p_1 : P_1 + \ldots + p_n : P_n$. We translate the process by appending $[\![P_1]\!], \ldots, [\![P_n]\!]$ to the command:

$$[] \qquad (st = s_Q) \longrightarrow p_1 : (st' = s_{P_1}) + ... + p_n : (st' = s_{P_n});$$

Case 9: $Q = \mathsf{cond}\ (e_1 \rhd P_1, \ldots, e_n \rhd P_n)$. We translate the process by constructing $[\![P_1]\!], \ldots, [\![P_n]\!]$ and replacing each command of the form:

$$[act] \qquad (st = s_{P_i})\&guard \longrightarrow updates;$$

by the command

$$[act] \qquad (st = s_Q)\&[\![e_1@loc]\!]\&...\&[\![e_{i-1}@loc]\!]\&[\![e_i@loc]\!]\&guard \longrightarrow updates;$$

where $[\![e@l]\!]$ is the translation of the PALPS expression $e@\ell$ into the PRISM language. Note that although all the transitions are simultaneously enabled from state s_Q, the use of the condition assigned to each of the conditional alternatives implies that at most one P_i can be selected and this is such that i is the smallest i with e_i evaluating to true.

Case 10: C, $C \overset{\text{def}}{=} P$. We translate the process by computing $[\![P]\!]$ and replacing each command in $[\![P]\!]$ of the form

$$[] \qquad (st = s_P)\&guard \longrightarrow updates;$$

by

$$[] \qquad (st = s_C)\&guard \longrightarrow updates;$$

Case 11: $Q = 0$. We translate the process as

$$\begin{array}{rl} [] & (st = s_0)\&(loc = l) \longrightarrow (s_l' = s_l - 1)\&(st' = s_{inactive}); \\ [tick] & (st = s_{inactive}) \longrightarrow (st = s_{inactive}); \end{array}$$

The translation above is mostly self-explanatory. It is worth pointing out, however, that, in Case 6, the translation of the activation on a new individual is conducted in two steps: in the first step, it is confirmed that there is still an inactive individual to be activated, $(i_s > 0)$ before reserving this individual via the assignment $i_s' = i_s - 1$ and, in the second step, module x synchronizes with one such individual y via action $rep_{x,y}$.

4.4 Correctness of the Translation

We now turn to consider the correctness of the proposed translation. This is demonstrated via the following two theorems. In what follows, given a PRISM model M, we write $M \longrightarrow M'$ if M contains an action `[α] guard -> updates;` where `guard` is satisfied in model M and execution of the `updates` gives rise to model M'.

Theorem 1. For any configuration (E, Sys), where E is compatible with Sys, whenever $(E, Sys) \xrightarrow{\alpha} (E', Sys')$ then either $[\![(E, Sys)]\!] \longrightarrow [\![(E', Sys')]\!]$ or $[\![(E, Sys)]\!] \longrightarrow\longrightarrow [\![E', Sys')]\!]$.

Theorem 2. For any configuration (E, Sys), where E is compatible with Sys, whenever $[\![(E, Sys)]\!] \longrightarrow M$ then $(E, Sys) \xrightarrow{\alpha} (E', Sys')$ and one of the following holds:

1. either $M = [\![(E', Sys')]\!]$, or
2. $M \longrightarrow M'$, $M' = [\![(E', Sys')]\!]$ and for all $M \longrightarrow M_1$ we have $(E', Sys') \xrightarrow{\beta} (E'', Sys'')$ and $M_1 \longrightarrow M_2$ where $M_2 = [\![(E'', Sys'')]\!]$.

Theorem 1 establishes that each transition of (E, Sys) can be mimicked in one or two steps by its translation module. Theorem 2 considers the other direction of the correctness: Given a transition of a PRISM module there are two possibilities. First, it is possible that the transition of the module corresponds to exactly one transition of the associated PALPS process. Second, it is possible that the transition of the module has resulted in an intermediate step of establishing the transition of a PALPS process. In this case, according to the theorem, each transition $M \longrightarrow M_1$ of this intermediate state M can also be executed on the side of the PALPS process, $(E', Sys') \xrightarrow{\beta} (E'', Sys'')$, and the pending move of the first transition can still be executed yielding a state M_2 which is in fact the translation of (E'', Sys'').

Sketch of the proof of Theorem 1: The proof consists of a case analysis of all possible ways in which the transition $(E, Sys) \xrightarrow{\alpha} (E', Sys')$ can be produced. Three cases exist:

- If the transition involves a single process participant $P{:}\langle \mathbf{s}, \ell\rangle$, then we may verify by induction on the structure of P that any action of P can also be performed by its translation and the resulting PRISM model is the translation of (E', Sys'). In all cases this can be established in a single move of module P.

- If instead the transition has arisen via the communication of two components of Sys then it is possible to establish that the two modules corresponding to the two components share the action in question and can thus execute the synchronization. This may take one or two actions at the PRISM level depending on whether process activation is involved.

- Finally, if $\alpha = \surd$, then it must be that all components of Sys enable the transition $\surd$. We may then observe that the PRISM translations of the components enable the *tick* action and thus the transition can be performed in exactly one move. In any case, the resulting PRISM model is the translation of (E', Sys'). □

Sketch of the proof of Theorem 2: The proof consists of a case analysis of all possible ways in which the transition $[\![(E, Sys)]\!] \longrightarrow M$ can be produced. It follows along similar lines with the proof of Theorem 1, with the exception of the first step implementing the activation of an inactive module. The important point to note here is that the intermediate state M captures correctly environment E' in the transition $(E, Sys) \longrightarrow (E', Sys')$. Thus, any further transition from this intermediate state M is in fact also enabled at the PALPS level. The resulting states are equivalent, pending the second step of the activation of the inactive individual. □

4.5 Verification in PRISM

In this section we briefly describe the types of analysis that can be performed on PALPS models via the PRISM model checker. For details, we refer the reader to [8].

Model Checking To begin with, PALPS models may be model checked in PRISM against properties specified in the PCTL logic. The syntax of the PCTL logic is given by the following grammar where Φ and ϕ range over PCTL state and path formulas, respectively, $p \in [0, 1]$ and $k \in \mathbb{N}$.

$$\begin{array}{rcl} \Phi & := & \mathit{true} \;\mid\; e \;\mid\; \neg\Phi \;\mid\; \Phi \wedge \Phi' \;\mid\; \mathsf{P}_{\bowtie p}[\phi] \\ \phi & := & \mathsf{X}\Phi \;\mid\; \Phi \mathsf{U}^k \Phi \;\mid\; \Phi_1 \mathsf{U} \Phi \end{array}$$

In the syntax above, we distinguish between state formulas Φ and path formulas ϕ, which are evaluated over states and paths, respectively. A state formula is built over logical expressions e and the construct $\mathsf{P}_{\bowtie p}[\phi]$.

Intuitively, a configuration s satisfies property $\mathsf{P}_{\bowtie p}[\phi]$ if for any possible execution beginning at the configuration, the probability of taking a path that satisfies the path formula ϕ satisfies the condition $\bowtie p$.

Path formulas include the X (next), U^k (bounded until) and U (until) operators, which are standard in temporal logics. Intuitively, $\mathsf{X}\Phi$ is satisfied in a path if the next state satisfies path formula Φ. Formula $\Phi_1\mathsf{U}^k\Phi_2$ is satisfied in a path if Φ_1 is satisfied continuously on the path until Φ_2 becomes true within k time units steps. Finally, formula $\Phi_1\mathsf{U}\Phi_2$ is satisfied if Φ_2 is satisfied at some point in the future and Φ_1 holds up until then.

As an example, consider a population $\mathbf{s}$ in danger of extinction. A property that one might want to check for such a population is that the probability of extinction of the population in the next ten years is less than a certain threshold p_e. This can be expressed in PCTL by the property $\mathsf{P}_{\leq p_e}[true\mathsf{U}^{10}\sum_{\ell\in\mathbf{Loc}}\mathbf{s}@\ell = 0]$. Alternatively, one might express that a certain central location ℓ will be re-inhabited with at least some probability p_r by $\mathbf{s}@\ell = 0 \to \mathsf{P}_{\geq p_r}[true\mathsf{U}(\mathbf{s}@\ell > 0)]$.

It is also possible to study the relation within a model between the size of the initial population and the probability of extinction of the population, by checking properties of the form $\mathbf{s}@\ell \geq m \to \mathsf{P}_{\geq p_e}[true\mathsf{U}(\mathbf{s}@\ell = 0)]$ or to explore the dynamics between two (or more) competing populations $\mathbf{s}$ and $\mathbf{s}'$. As an example, expressing that, within the next 20 years with some high probability, members of the population $\mathbf{s}$ will outnumber the members of population $\mathbf{s}'$: $\mathsf{P}_{\geq p}[true\mathsf{U}^{20}(\sum_{\ell\in\mathbf{Loc}}\mathbf{s}'@\ell \leq \sum_{\ell\in\mathbf{Loc}}\mathbf{s}@\ell)]$.

PRISM also enables to take a more *quantitative* approach for model checking PCTL properties: it supports the verification of the constructs $P_{min=?}[\phi]$ and $P_{max=?}[\phi]$ via which the minimum and maximum probabilities of satisfying ϕ are computed.

Steady-state behavior. PRISM also supports reasoning about the *steady-state behavior* of a model, that is, the behavior in the long-run or when an equilibrium is reached. Steady-state properties are only available for discrete-time and continuous-time Markov chains. These properties are expressed by $S_{bound}[prop]$. Such a property is true in a state s of a discrete-time or a continuous-time Markov chain if, starting from s, the steady-state (long-run) probability of being in a state which satisfies the (boolean-valued) property *prop*, meets the bound *bound*. For example, the steady-state property $S_{bound}[s@2 = 4]$ expresses that the long-run probability that there will be 4 individuals of species s at location 2 meets the bound.

Rewards. PRISM models can also be augmented with information about rewards: It is possible to assign a reward (a positive real number) to any command or state of a PRISM model. Every time a command is executed or a state is visited, the rewards associate with the command or state is accumulated. It is then possible to reason about reward-based properties for discrete-time Markov chains, by extending the logic PCTL with the following additional operators [18]:

$$R_{\bowtie r}[C^{\leq k}] \mid R_{\bowtie r}[I^{\leq k}] \mid R_{\bowtie r}[F\Phi] \mid R_{\bowtie r}[S]$$

where $\bowtie \in \{<, \leq, \geq, >\}, r \in \mathbb{R}_{\geq 0}, k \in \mathbb{N}$ and Φ is a PCTL formula. The R operator defines properties about the *expected* value of rewards. The formula $R_{\bowtie r}[\psi]$, where ψ denotes one of the four possible operators in the grammar above, is satisfied in a state s if, from s, the expected value of reward ψ meets the bound $\bowtie r$. Operator $C^{\leq k}$ refers to the reward accumulated over k time steps; $I^{\leq k}$ the state reward at time instant k; $F\Phi$, the reward accumulated before a state satisfying Φ is reached; and S, the long-run rate of reward accumulation. Properties of the form $R_{=?}[\psi]$ means "what is the expected reward for operator ψ?".

5 A Case Study in PRISM

In this section, we apply our methodology for model checking PALPS systems using the PRISM tool and we explore the size of models on which model checking can be performed. As an example we consider a number of variations of the system in Example 1, described in Section 2.

In our model we will assume a lattice of locations of size $n \times n$ (for $n = 2, 3, 4$). To begin with, we considered a lattice of 2×2 with reflecting boundary conditions. This implies that, in case of migration, an individual may move to one of two other locations. Thus, the PALPS definition of an individual takes the following form:

$$P \stackrel{\text{def}}{=} \sum_{\ell \in \mathbf{Nb}(\mathsf{myloc})} \frac{1}{2} : go\,\ell.\surd.\mathsf{cond}\ (\mathsf{s@myloc} = 1 \rhd P_1;\ \mathsf{true} \rhd \surd.P)$$

$$P_1 \stackrel{\text{def}}{=} p : \overline{rep}.\surd.P_1 \oplus (1-p) : \overline{rep}.\overline{rep}.\surd.P_1$$

The PRISM encoding of the system follows the translation methods presented in Section 4.3. It contains four global variables, `s1`, `s2`, `s3`, `s4`, which represent the number of individuals at each location. It also includes the constant p which characterizes the probability that an individual produces one or two children. The PRISM model contains a module for each active individual and a module for each inactive individual. All the tests were performed on a G46VW Asus laptop with an Intel i5 2.50 GHz processor and 8 GB of RAM. We ran the tests under Linux Ubuntu 13.04 (Kernel 3.8.0_17), using PRISM 4.0.3 (with the MTBDD engine) and Java 7.

As a first experiment we explored the size of a system that can be handled for *model checking* in PRISM by setting a limit on the maximum number of individuals that can be created. It turns out that the size of a model with seven or more individuals exceeds the amount of RAM memory available. Furthermore, we created a variation of the model with one predator in addition to the individuals of type P above. In this case it turns out that the size of the model with 1 predator and 5 preys, exceeds the size that can be treated in PRISM on personal computers which, according to the PRISM manual, is $10^7 - 10^8$ states.

Case study size	Number of States	Number of Transitions	Construction time (sec.)	RAM (GB)
3 PALPS individuals	130397	404734	8	0.5
2 PALPS individuals*	148360	250964	13	0.8
4 PALPS individuals	210736	7312132	101	1.9
5 PALPS individuals	18647917	85022112	601	2.8
6 PALPS individuals	87332123	116457321	5094	4.5
4 PALPS individuals*	113450511	411434511	3521	5.1
7 PALPS individuals				out of RAM

Table 5: Performance of building probabilistic models in PRISM.

In Table 5 we summarize the results we obtained in our experiments. In the models, we fixed $p = 0.4$. Results marked by $*$ refer to models extended with a predator.

We believe that the reason for the state-space explosion is the large number of possible interleaving orders among all the commands of all the modules, which, in the case of model checking have to be computed in their totality. In next section we discuss how to overcome this problem in the future.

As a second experiment, we proceeded to determine the limits for *simulating* PALPS models. We constructed PRISM models with various numbers of modules of active and inactive individuals and we run them on PRISM. In Table 6, we summarize the results. It turns out that for models with more than 5000 individuals simulation requires at least 12 hours (which was the time limit we set for our simulations).

Individuals	File Size (MB)	RAM (GB)	Simulation Time (s)
10	0.1	0.18	1
100	0.4	0.3	8
500	2.0	0.5	45
1000	4.2	1.0	300
1500	6.2	0.7	454
2000	8.2	0.9	820
5000	20.1	2.0	$>$ 12 hours
10000	44.1	3.4	$>$ 12 hours

Table 6: Performance of simulating probabilistic systems in PRISM.

Subsequently, we looked into the restriction imposed by our assumption of bounded replication. This restriction may lead to deadlocks when an active individual attempts to reproduce but no inactive module is available for synchronization. To explore this, we used the *simulation* environment of PRISM and searched for deadlocks by repeating 100 simulations of the model of maximum path length 1000 time steps. Although, this procedure is not complete, we may consider it sufficient as it looks into a fairly large number of life cycles of the population. In Table 7 we summarize the results obtained.

As a further experiment with PRISM, we considered a simplification of Example 1 in which the model consists of exactly 3 individuals and in which reproduction does not produce new processes but only updates the number of individuals at the reproduction location. Therefore, there will always be 3 active modules, but the population size at each location will be updated every time an individual performs the reproduction action. The PRISM model of this example contains 4 modules, one for each of the three individual, and one module for modeling reproduction. As an example, we present some excerpts of the module to define one of the individuals, namely $P1$. First, the individual has to choose to move to a new location. This is modeled as follows where the variables `st1` and `loc1` represent the local state of the module and the location of the individual, respectively.

Active individuals	Inactive individuals	Deadlock (x, y, z)
3	18	(No,Yes,Yes)
4	24	(No,Yes,Yes)
5	30	(No,Yes,Yes)
6	36	(No,Yes,Yes)
7	42	(No,No,Yes)
8	48	(No,No,Yes)
9	56	(No,No,Yes)
10	60	(No,No,No)

Table 7: Occurrence of deadlock in various instances of the model. The values (x, y, z) refer to the presence of deadlock in the case of 4 locations (x), 9 locations (y), and 16 locations (z).

```
st1 : [0..40] init 0;
// For location 1
[] (st1=0)&(loc1=1) ->
          0.5: (st1'=17)& (s1'=s1-1)&(s2'=s2+1)&(loc1'=2)
          + 0.5: (st1'=27)&(s1'=s1-1)&(s3'=s3+1)&(loc1'=3);
```

Subsequently, the process checks if it has absolute control of the location. If it does, then it reproduces; otherwise, it synchronizes with the clock action.

```
 // For location 1
 [] (st1 = 17) & (loc1=1) & (s1 = 1) -> (st1' = 19);
 [] (st1 = 17) & (loc1=1) & (s1 != 1) -> (st1' = 15);
```

In the case that reproduction is selected, the individual chooses to reproduce one or two times.

```
 // For location 1
 [] st1 = 19 -> p : (st1' = 12) + (1-p) : (st1' = 11);
 [rep1] (st1 =11) -> (st1' = 12);
 [rep1] (st1 = 12) -> (st1' = 15);
```

Whether the individual reproduces or not, it synchronizes with the other modules with a clock action and, finally, returns to the initial state.

```
[tick] (st1 = 15) -> (st1' = 0);
```

The reproduction module is defined as follows. Local variable `st4` represents the local state of the process. At state 0, the module can synchronize with the other modules via the clock action or the module can synchronize with any individual at any location with the respective reproduction action. After synchronizing with the reproduction action, the module goes to a state in which it updates the number of individuals at the respective location by one and then returns to the initial state.

```
st4 : [0..34] init 0;
[tick] (st4 = 0) -> (st4'=0);
// For individual 1
[rep1] (st4=0) -> (st4'=1);
[] (st4=1) & (loc1=1) -> (st4'=0) & (s1'=s1+1);
[] (st4=1) & (loc1=2) -> (st4'=0) & (s2'=s2+1);
[] (st4=1) & (loc1=3) -> (st4'=0) & (s3'=s3+1);
[] (st4=1) & (loc1=4) -> (st4'=0) & (s4'=s4+1);
```

As a first analysis of this simplified example, we studied the effect of the value of the probability `p` on the size of the population by performing a reachability analysis of the model. Specifically, Figure 5 records the probability to reach a state with a certain total number of individuals for various values of the constant `p`.

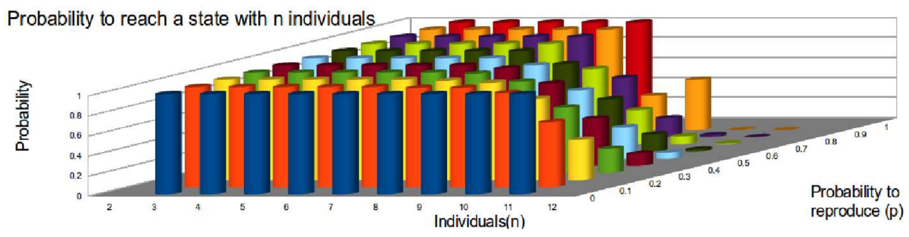

Figure 5: Probability that the population reaches a certain size, for various values of p.

Using PRISM it is also possible to study the steady-state behavior of the model. For instance, one might wonder about the effect of the initial placement of individuals on the grid for the long-run behavior of the system. To this effect, we have enunciated and checked the following property `S =? [s`$_q$` = `r`]`, where $q \in [1, 4]$ is the identifier of the location and $r \in [0, 6]$ the number of individuals. This property allows us to evaluate the probability to have r individuals at location q in the long run. For example, for

$q = 1, r = 2$, this property means "what is the probability that, in the long run, we reach a state in which the number of individuals at location 1 is equal to 2?". Interestingly, our experiments have shown that the initial placement of individuals has no effect on the long-run behavior of the model: in the long run all locations have the same probability of being populated by a population of r individuals for any value of r. Intuitively, this is so because as there are no predators in the system, in the long run the population densities of the locations balance out. In Table 8 we present these probabilities for $r \in [0, 6]$.

Number of individuals	0	1	2	3	4	5	6
Probability	0.0	0.191	0.418	0.261	0.109	0.021	0.0

Table 8: Steady-state probability for reaching a state with a certain number of individuals at a location

Finally, in Table 9, we present some results that we obtained by assigning rewards to certain actions of the model. In particular we assigned rewards to (1) the clock action and (2) the reproduction actions. We then used PRISM with the property $R = ?[C^{\leq k}]$ to compute the average reward accumulated within the first k execution steps. In the case that rewards are associated with the clock action, this property calculates the average number of clock steps taken within the first k execution steps. Similarly, in the case that rewards are associated with the reproduction action, this property calculates the average number of reproductions within the first k execution steps. In the latter case, we observed that the results were the same for each individual. That is, each individual engages in the same average number of reproductions within the first k execution steps. Intuitively, this is so because there are no predators in the system and, following the first move action and thereafter, each individual is equally likely to reproduce.

As an example of how this is implemented in PRISM, to count the average size of the offspring of individual $P1$, we add a reward of 1 each time individual $P1$ synchronizes with the reproduction module by a reproduction action. This reward structure is defined as follows:

```
rewards  "repP1"
[rep1] true : 1;
endrewards
```

Thus, as reported in Table 9, the expected number of tick-synchronizations within the first 100 steps of the model is 10.558, whereas the expected number of reproductions an individual engages in during the first 100 steps of the model is 2.117.

Simulation steps	Expected time units	Expected size of offspring
100	10.558	2.117
1000	139.049	2.133

Table 9: Expected number of clock steps and the expected number of reproductions of an individual

6 Concluding Remarks and Future Work

This paper reports on work towards the development of a process-calculus framework for the spatially-explicit and individual-based modeling of ecological systems which follows a discrete-time, discrete-space approach. The main distinguishing feature of PALPS is that it associates locations with attributes which allow the enunciation of location-dependent behavior. For example, "an individual will migrate from its current location only if this location has exceeded its capacity". This is achieved by enriching the syntax with a set of expressions and extending the notion of a *process* by the notion of a *configuration.* A configuration is an entity that combines the process/code of a system with an environment which records information regarding the state of the system, based on which expressions are evaluated.

Furthermore, we have described a methodology for translating PALPS models into the PRISM language. The proposed encoding can be employed for both simulating and model checking PALPS systems using the PRISM tool. In this paper we experimented with the model-checking capability and we demonstrated types of model-checking analysis that can be performed on PALPS models via simple examples. This exercise has confirmed that model checking of population systems defined in PALPS is not viable, at least at this stage. This is because, the large number of modules and the high degree of nondeterminism that occurs within PRISM encodings of PALPS modules results in a state-space explosion that becomes apparent even in systems with a very small number of individuals.

As far as simulation is concerned, we have observed that PRISM allows the simulation of realistic PALPS models. Nonetheless, the individual-based approach implemented by PALPS and its translation into PRISM impose limits on the size of systems that can be simulated that are not as prominent in stochastic models: As we discussed in Section 1, it is indeed the case that continuous time frameworks can benefit from Gillespie's stochastic simulation methods, which speed up discrete stochastic simulation by leaping over many reaction events in a way that approximates exact stochastic simulation.

These conclusions lead to the challenge of defining new approaches for analysis of MDPs that arise from the modeling of population systems. A number of directions may be employed for tackling this challenge. The first method is related with the idea of sequentializing execution of models by imposing an ordering between actions and processes. As we have already discussed, this idea has stemmed from ecologists concerned with the construction of discrete-time ecological models: According to [5], in discrete-time, individual-based models, ordering of processes and actions can be applied on models and different policies on these orderings can be explored and studied separately.

Applying such orderings on PALPS would greatly simplify the state-space of models as multiple interleaving orders that are causing the models to explode will be pruned away and nondeterminism will be significantly reduced. We also expect that sequentializing the execution of different types of actions will enable the grouping of sequences of actions into single macro-actions thus yielding smaller models while preserving the initial behaviors. For example, given a sequence of movement actions by a set of individuals we can simply retain the first and last state while pruning away the intermediate states of the separate individuals moves. We are currently exploring this approach by enriching PALPS models with *policies* that determine externally the orderings of different actions within a model. To this end, we are also interested in developing the notion of (partial) confluence [39] for PALPS and applying it for reducing the state space of PALPS systems.

Other possible approaches we plan to investigate for reducing the state-space of PALPS models include enhancing the semantics of PALPS to enable a more succinct presentation of systems especially in terms of the multiplicity of individuals. Such a semantics would have as its basic entity the process $P{:}\langle \mathbf{s}, \ell, n\rangle$, which represents n individuals of species s at location ℓ, and it would ease both the specification as well as the verification of system models. In addition, we are interested to explore the possibility of giving symbolic semantics on PALPS systems by applying a symbolic treatment of environments. Furthermore, an interesting alternative would be to extend the work of [30] in the spatial domain for defining mean-field semantics for PALPS.

As far as the PRISM tool is concerned, we believe that it would be worth to consider alternative approaches for producing PRISM input from PALPS models. We recall that there are two possibilities for achieving this: using the PRISM language or constructing the transition matrix of a Markov decision process. Adopting the second approach could prove beneficial for a number of reasons: (1) In PALPS, we use CCS communication instead of the CSP communication employed in PRISM language. This is necessary for modeling phenomena such reproduction and preying where the communication is binary and retaining the asymmetry between the roles of *input* and *output* is important for implementing the task (e.g. we must distinguish between the actions of a prey and a predator). (2) Direct encoding of PALPS into a transition matrix would avoid the bounded replication and subsequent modeling of inactive individuals in a PRISM model, deemed necessary due to PRISM not allowing the dynamic creation of modules. Note that the translation of recursion for the probabilistic pi-calculus into PRISM proposed in [34] is not sufficient for our purposes because the language considered for the translation concerns a restricted kind of recursion and no replication. (3) Finally, another source of complexity in the translation concerns the modeling of reproduction, where an intermediate step is required to model the process due to the necessity of updating a global variable in a synchronization command, something that is not allowed within the PRISM language. These observations suggest that representing PALPS models as a Markov decision process, instead of encoding them into the PRISM language, could result in simpler models.

Another possible direction for future work which aims to enhance the expressiveness of PALPS for capturing more complex systems includes the adoption of continuous time as well as the use of dynamic attributes to allow exploring the system while, for example, patch quality degrades, temperatures increase, etc. This could be made possible via the combination of continuous and discrete time behaviors similarly to the *behavioral hybrid process calculus* of [28]. Finally, we are currently involved in the application of our methodology to case studies from the local habitat, the intention being to explore properties such as species extinction and persistence.

References

[1] Online PRISM documentation. http://www.prismmodelchecker.org/doc/.

[2] M. Antonaki. A probabilistic process algebra and a simulator for modeling population systems. Master's thesis, University of Cyprus, 2012.

[3] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and G. Pardini. Spatial calculus of looping sequences. *Theoretical Computer Science*, 412(43):5976–6001, 2011. doi:10.1016/j.tcs.2011.01.020.

[4] Roberto Barbuti, Andrea Maggiolo-Schettini, Paolo Milazzo, and Angelo Troina. A calculus of looping sequences for modelling microbiological systems. *Fundamenta Informaticae*, 72(1-3):21–35, 2006.

[5] L. Berec. Techniques of spatially explicit individual-based models: construction, simulation, and mean-field analysis. *Ecological Modeling*, 150:55–81, 2002. doi:10.1016/S0304-3800(01)00463-X.

[6] D. Besozzi, P. Cazzaniga, D. Pescini, and G. Mauri. Modelling metapopulations with stochastic membrane systems. *BioSystems*, 91(3):499–514, 2008. doi:10.1016/j.biosystems.2006.12.011.

[7] D. Besozzi, P. Cazzaniga, D. Pescini, and G. Mauri. An analysis on the influence of network topologies on local and global dynamics of metapopulation systems. In *Proceedings of AMCA-POP'10*, pages 1–17, 2010. doi:10.4204/EPTCS.33.1.

[8] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proceedings of FSTTCS'95*, LNCS 1026, pages 499–513. Springer, 1995. doi:10.1007/3-540-60692-0_70.

[9] Livio Bioglio, Cristina Calcagno, Mario Coppo, Ferruccio Damiani, Eva Sciacca, Salvatore Spinella, and Angelo Troina. A Spatial Calculus of Wrapped Compartments. *CoRR*, abs/1108.3426, 2011.

[10] L. Cardelli. Brane calculi - interactions of biological membranes. In *Proceedings of CMSB'04*, LNCS 3082, pages 257–278. Springer, 2005. doi:10.1007/978-3-540-25974-9_24.

[11] Luca Cardelli and Radu Mardare. Stochastic Pi-Calculus Revisited. Technical report, Microsoft Research, Cambridge, UK, 2012.

[12] M. Cardona, M. Colomer, A. Margalida, I. Pérez-Hurtado, M. J. Pérez-Jiménez, and D. Sanuy. A P system based model of an ecosystem of

the scavenger birds. In *Proceedings of WMC'09*, LNCS 5957, pages 182–195. Springer, 2009. doi:10.1007/978-3-642-11467-0_14.

[13] Mónica Cardona, M. Angels Colomer, Antoni Margalida, Antoni Palau, Ignacio Pérez-Hurtado, Mario J. Pérez-Jiménez, and Delfí Sanuy. A computational modeling for real ecosystems based on P systems. *Natural Computing*, 10(1):39–53, 2011. doi:10.1007/s11047-010-9191-3.

[14] P. Cazzaniga, D. Pescini, D. Besozzi, and G. Mauri. Tau leaping stochastic simulation method in P systems. In *Proceedings of WMC'06*, LNCS 4361, pages 298–313. Springer, 2006. doi:10.1007/11963516_19.

[15] Qiuwen Chen, Fei Ye, and Weifeng Li. Cellular-automata-based ecological and ecohydraulics modelling. *Journal of Hydroinformatics*, 11(3-4):252–265, 2009. doi:10.2166/hydro.2009.026.

[16] Federica Ciocchetta and Jane Hillston. Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theoretical Computer Science*, 410(33–34):3065–3084, 2009. doi:10.1016/j.tcs.2009.02.037.

[17] Rocco De Nicola, Diego Latella, Michele Loreti, and Mieke Massink. Rate-based transition systems for stochastic process calculi. In *Proceedings of ICALP (2)*, LNCS 5556, pages 435–446. Springer, 2009. doi:10.1007/978-3-642-02930-1_36.

[18] V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker. Automated verification techniques for probabilistic systems. In *Proceedings of SFM'11*, LNCS 6659, pages 53–113. Springer, 2011. doi:10.1007/978-3-642-21455-4_3.

[19] S. C. Fu and G. Milne. A flexible automata model for disease simulation. In *Proceedings of ACRI'04*, LNCS 3305, pages 642–649. Springer, 2004. doi:10.1007/978-3-540-30479-1_66.

[20] V. Galpin. Modelling network performance with a spatial stochastic process algebra. In *Proceedings of AINA'09*, pages 41–49. IEEE Computer Society, 2009. doi:10.1109/AINA.2009.75.

[21] L. R. Gerber and G. R. VanBlaricom. Implications of three viability models for the conservation status of the western population of steller sea lions (Eumetopias jubatus). *Biological Conservation*, 102(3):261–269, 2001. doi:10.1016/S0006-3207(01)00104-5.

[22] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81:2340–2361, 1977. doi:10.1021/j100540a008.

[23] D. T. Gillespie and L.R. Petzold. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115:1716–1733, 2001. doi:10.1063/1.1378322.

[24] Stephen Gilmore and Jane Hillston. The PEPA workbench: A tool to support a process algebra-based approach to performance modelling. In *Proceedings of Computer Performance Evaluation Modelling Techniques and Tools'94*, LNCS 794, pages 353–368, 1994. doi:10.1007/3-540-58021-2_20.

[25] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[26] M. Jeschke, R. Ewald, and A. Uhrmacher. Exploring the performance of spatial stochastic simulation algorithms. *Journal of Computational Physics*, 230(7):2562–2574, 2011. doi:10.1016/j.jcp.2010.12.030.

[27] D. Kouzapas and A. Philippou. A process calculus for dynamic networks. In *Proceedings of FMOODS/FORTE'11*, LNCS 6722, pages 213–227. Springer, 2011. doi:10.1007/978-3-642-21461-5_14.

[28] Tomas Krilavičius and Helen Schonenberg. Discrete Simulation of Behavioural Hybrid Process Calculus. In *Proceedings of IFM 2005*, pages 33–38, 2005. URL: http://doc.utwente.nl/54779/.

[29] C. McCaig, R. Norman, and C. Shankland. Process algebra models of population dynamics. In *Proceedings of AB'08*, LNCS 5147, pages 139–155. Springer, 2008. doi:10.1007/978-3-540-85101-1_11.

[30] Chris McCaig, Rachel Norman, and Carron Shankland. From individuals to populations: A mean field semantics for process algebra. *Theoretical Computer Science*, 412(17):1557–1580, 2011. doi:10.1016/j.tcs.2010.09.024.

[31] R. Milner. *A Calculus of Communicating Systems*. Springer, 1980.

[32] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts 1 and 2. *Information and Computation*, 100:1–77, 1992. doi:10.1016/0890-5401(92)90008-4.

[33] E. S. Minor, R. I. McDonald, E. A. Treml, and D. L. Urban. Uncertainty in spatially explicit population models. *Biological Conservation*, 141(4):956–970, 2008. doi:10.1016/j.biocon.2007.12.032.

[34] Gethin Norman, Catuscia Palamidessi, David Parker, and Peng Wu. Model Checking Probabilistic and Stochastic Extensions of the π-Calculus. *IEEE Transactions on Software Engineering*, 35(2):209–223, March 2009. doi:10.1109/TSE.2008.77.

[35] G. Pardini. *Formal Modelling and Simulation of Biological Systems with Spatiality*. PhD thesis, University of Pisa, 2011.

[36] R. G. Pearson and T. P. Dawson. Long-distance plant dispersal and habitat fragmentation: identifying conservation targets for spatial landscape planning under climate change. *Biological Conservation*, 123:389–401, 2005. doi:10.1016/j.biocon.2004.12.006.

[37] Mario J. Pérez-Jiménez and Francisco José Romero-Campero. A study of the robustness of the EGFR signalling cascade using continuous membrane systems. In *Proceedings of IWINAC'05*, LNCS 3561, pages 268–278, 2005. doi:10.1007/11499220_28.

[38] D. Pescini, D. Besozzi, G. Mauri, and C. Zandron. Dynamical probabilistic P-systems. *International Journal of Foundations of Computer Science*, 17(1):183–204, 2006. doi:10.1142/S0129054106003760.

[39] Anna Philippou and David Walker. On confluence in the π-calculus. In *Proceedings of ICALP '97*, LNCS 1256, pages 314–324. Springer, 1997. doi:10.1007/3-540-63165-8_188.

[40] Gheorghe Păun. Computing with Membranes (P Systems): An Introduction. In *Current Trends in Theoretical Computer Science*, pages 845–866. World Scientific, 2001.

[41] Aviv Regev, Ekaterina M. Panina, William Silverman, Luca Cardelli, and Ehud Shapiro. BioAmbients: an abstraction for biological compartments. *Theoretical Computer Science*, 325(1):141–167, September 2004. doi:10.1016/j.tcs.2004.03.061.

[42] Antoine Spicher, Olivier Michel, Mikolaj Cieslak, Jean-Louis Giavitto, and Przemyslaw Prusinkiewicz. Stochastic P systems and the simulation of biochemical processes with dynamic compartments. *Biosystems*, 91(3):458–472, 2008. doi:10.1016/j.biosystems.2006.12.009.

[43] D. J. T. Sumpter, G. B. Blanchard, and D. S. Broomhear. Ants and agents: a process algebra approach to modelling ant colony behaviour. *Bulletin of Mathematical Biology*, 63(5):951–980, 2001. doi:10.1006/bulm.2001.0252.

[44] D. J. T. Sumpter and D. S. Broomhead. Relating individual behaviour to population dynamics. *Proceedings of of Royal Society B: Biological Sciences*, 268(1470):925–932, 2001. doi:10.1098/rspb.2001.1604.

[45] C. Tofts. Processes with probabilities, priority and time. *Formal Aspects of Computing*, 6:536–564, 1994. doi:10.1007/BF01211867.

[46] J. M. J. Travis and C. Dytham. The evolution of disperal in a metapopulation: a spatially explicit, individual-based model. *Proceedings: Biological Sciences*, 265(1390):17–23, 1998. doi:10.1098/rspb.1998.0258.

[47] D. L. Urban and H. H. Shugart. Avian demography in mosaic landscapes: modeling paradigm and preliminary results. *Wildlife 2000: Modeling Habitat Relationships of Terrestrial Vertebrates*, pages 273–279, 1986.