Process algebra for epidemiology: evaluating and enhancing the ability of PEPA to describe biological systems.

Soufiene Benkirane

University of Stirling

Abstract

Modelling is a powerful method for understanding complex systems, which works by simplifying them to their most essential components. The choice of the components is driven by the aspects studied. The tool chosen to perform this task will determine what can be modelled, the maximum number of components which can be represented, as well as the analyses which can be performed on the system.

Performance Evaluation Process Algebra (PEPA) was initially developed to tackle computer systems issues. Nevertheless, it possesses some interesting properties which could be exploited for the study of epidemiological systems. PEPA's main advantage resides in its capacity to change scale: the assumptions and parameter values describe the behaviour of a single individual, while the resulting model provides information on the population behaviour. Additionally, stochasticity and continuous time have already proven to be useful features in epidemiology. While each of these features is already available in other tools, to find all three combined in a single tool is novel, and PEPA is proposed as a useful addition to the epidemiologist's toolbox. Moreover, an algorithm has been developed which allows converting a PEPA model into a system of Ordinary Differential Equations (ODEs). This provides access to countless additional software and theoretical analysis methods which enable the epidemiologist to gain further insight into the model. Finally, most existing tools require a deep understanding of the logic they are based on and the resulting model can be difficult to read and modify. PEPA's grammar, on the other hand, is easy to understand since it is based on few, yet powerful concepts. This makes it a very accessible formalism for any epidemiologist.

The objective of this thesis is to determine precisely PEPA's ability to describe epidemiological systems, as well as extend the formalism when required. This involved modelling two systems: the bubonic plague in prairie dogs, and measles in England and Wales. These models were chosen as they exhibit a good range of typical features, allowing to thoroughly test PEPA. All features required in each of these models have been analysed in detail, and a solution has been provided for representing each of these features. While some of them could be expressed in a straightforward manner, PEPA did not provide the tools to express others. In those cases, we determined methods to approach the desired behaviour, and the limitations of said methods were carefully analysed. In the case of models with a structured population, PEPA was extended to simplify their expression and facilitate the writing process of the PEPA model. The work also required the development of an algorithm to derive ODEs adapted to the type of models encountered. Finally, the PEPAdum software was developed to assist the modeller in the generation and analysis of PEPA models, by simplifying the process of writing a PEPA model with compartments, performing the average of stochastic simulations and deriving and explicitly providing the ODEs using the Stirling Amendment.

Acknowledgements

I would like to thank all those who have contributed to the success of this PhD project.

First of all, I would like to thank Dr Carron Shankland, for her unrivalled dedication as principal supervisor of this project.

Her enthusiasm, knowledge as well as her guidance have been greatly appreciated, and the quality of this thesis was greatly improved thanks to her advice.

Next, a big thank you goes to Dr Rachel Norman, who, as second supervisor, subjected each bit of my thesis to further critique and thereby enabled me to further improve my work.

I would also like to thank my fellow PhD students for their helpful advice and support throughout the past three years. In particular, thank you Ros for rekindling the dying flame of our "PG-Tips" advice sessions for postgraduate students; thank you Claire for your organisational talent; and thanks Andy and Jesse for your critical responses to my presentations.

Further thanks go to the members of the administrative team, Grace, Linda and Heather.

Some people, while not being physically present, helped me as best they could. My parents and my sister first of all, who have helped me get to where I am today. Joris, who patiently encouraged

and supported me through good times and bad times; Véro, Alex and Erwann who came to see me despite the distance, cold and Scottish rain.

Last but not least, Gwen has given me her support each and every day of this thesis, no matter how hard it got, with unsurpassed patience. I doubt that I would have achieved this without her.

Contents

Chapter 1

Introduction

According to the World Health Organization [95], in 2002, about 19.1% of worldwide deaths, and 52.7% of deaths in Africa were caused by infectious and parasitic diseases. In order to avoid as many of them as possible, it is important to carry out research on the topic to understand, prevent, cure and reduce the impact of a given disease. To achieve these goals, different areas of expertise must be involved, ranging from medicine to geography, sociology, biology and mathematics. Mathematical modelling is an important part of this process of understanding the disease and its behaviour.

For some, modelling is not a scientifically acceptable technique of describing biological systems. To a certain extent, these critics are right. A model will never be able to describe accurately a system involving living organisms, whether they are animals, plants, bacteria or cells. Yet, as the statistician George Box said [12, p.424]: "essentially, all models are wrong, but some are useful". A model, by definition, is an incomplete representation of a system. The simplification required to construct the model serves a purpose. Complete biological systems are often very complex and their behaviour is influenced by many factors. This has two consequences: the actual influence of

each of the factors is usually not known and the size of the system renders it too complicated to understand and analyse in its entirety. Models can help in both cases. Firstly, different hypotheses concerning the way factors influence the model can be tested and the parameter space can be explored in order to validate a theory. Secondly, in order to simplify the considered system, the model can focus on a reduced set of carefully chosen factors, depending on the questions that it is asked to answer.

The objective of this thesis is to determine to what extent Performance Evaluation Process Algebra (PEPA) is suitable for modelling diseases, as well as extend the formalism when required. This involves recognizing PEPA's strengths and weaknesses, and remedy any issues encountered when possible. The first objective will be achieved by modelling two epidemiological systems, the bubonic plague in prairie dogs, and measles in England and Wales, and analysing each of the features required to describe those systems. The second required the update of the algorithm to generate the set of ordinary differential equations from a PEPA model, and the development of a software, PEPAdum, in order to assist the modeller in the analysis of the model, by simplifying the process of writing a PEPA model with compartments, performing the average of stochastic simulations and deriving and explicitly providing the ODEs using the Stirling Amendment.

Today, many formalisms are available to modellers. Their principal characteristics are their expressivity, the capacity to handle large and complex models and the ease of performing theoretical analysis on the resulting model. Unfortunately, no existing formalism has a very flexible grammar and is able to handle large and/or complex models and offers the possibility to perform an extensive theoretical analysis of the model. In order to have at our disposal tools adapted to different types of analyses, different formalisms with as wide a range of strengths as possible are essential. The following list of formalisms is given as an indication of existing options and a to give some historical context, but this work will not try to compare them directly and in depth to PEPA, as this beyond

the scope of this study.

General-purpose programming languages [76, 77], such as Java or C, are probably the most flexible tool that can be used to model epidemiological systems. They give a total freedom to the modeller, and any computable feature can be expressed. The main drawback of this method is that it requires a high proficiency in the programming language by the modeller. Also, the freedom provided comes with a price. First, the capacity of the model to efficiently describe a high number of individuals or a complex system is entirely in the hands of the programmer. Secondly, everything has to be programmed from scratch, leading to a time-consuming, tedious and error-prone process of building the model. Finally, very few tools exists to check the validity of the model and perform additional analyses of the model.

Cellular automata [4, 93] is a discrete technique that explicitly describes individuals and space. The basic technique divides the world into squared compartments, which have a set of valid states. Simple rules define the evolution of the state of each compartment, depending on the state of its neighbours. For example in the case of an epidemiological model, the compartments could be in a healthy individual, infectious individual or empty state, and a rule could be that if two or more neighbouring compartments are in an infectious individual state, and the considered compartment is in a healthy individual state, it evolves to an infectious individual state. Some more complex variants allow the compartments to have different shapes, explicitly model individuals, or have a three or more dimensional space. Cellular automata are particularly suitable for systems where space and individual level behaviour (as opposed to population level behaviour) are important. The drawbacks of cellular automata lie in their lack of efficiency and the availability of tools to perform additional analyses. Since a cellular automaton explicitly exhibits all compartments and individuals, the processing time increases with the population size and the number of compartments considered, limiting the size the model can reach. Also, the technique is very flexible, and as such,

very little theory exists to help in the analysis of the model. For example, the existing algorithms deriving the average behaviour of the system are limited to a restricted set of cellular automata.

Ordinary differential equations [54, 55] are by far the most commonly used modelling technique in epidemiology. It is a population level technique, in the sense that the rules dictating the evolution of the system are describing the behaviour of groups of individuals rather than the individuals themselves. A model is presented as a system of equations. Each species represents a group of individuals, and the corresponding equation describes its behaviour. In general, the equations are deterministic and can only present the average behaviour of the system. This means that the information related to the inherent stochasticity of biological systems is not available to the modeller. Ordinary differential equations also have an expressivity more limited than either of the two previous techniques presented, and struggle in particular to represent spatial structures. Yet differential equations remain a very powerful modelling technique. First, the processing time remains practically constant when the population size is increased. Where cellular automata are often limited to a population size in the tens or hundreds of thousand individuals, differential equations can handle a virtually infinite population size. Also, since the technique has been used for centuries, very efficient tools have been developed to carry out the analysis of models. Finally, its restrictive grammar allows several types of checks and analyses to be performed easily and efficiently.

Process algebra is the most recent of the modelling techniques presented here. The basic principle behind it is that each individual can be in one of several states. Depending on the state she is in, she can then choose between a set of actions, each of them having a probability or rate and a target state associated with it. In some respects, process algebra has a lot in common with cellular automata. In both cases, the focus is on individual behaviour. Also, change happens in a similar manner: individual agents are in a certain state and simple rules are used to describe how

they evolve. Major differences can also be identified. First, individual agents are tracked during a simulation of a cellular automaton model, which makes it possible to study their evolution, while process algebra formalisms only track the number of agents in a particular state. Also, contrary to process algebra, space is an essential part of a cellular automaton's behaviour. Finally, process algebras usually have a more rigid grammar. While this makes it difficult or even impossible to model some behaviours, it also allows efficient model checking and more analyses can be performed on the model.

The first modern process algebra is the work of Robin Milner: calculus of communicating systems (CCS) [66]. CCS and its extension π-calculus [67] have since been used to model parallel programming [63] and biochemical processes [73]. π-calculus has also been adapted to specifically deal with chemical reactions [14]. Since then, numerous process algebras have been developed, each adapted to different systems or questions studied. Beta-binders [20] is based on π-calculus specifically conceived to describe biological systems. It has been so far mainly used to model signalling pathways [10, 38]. Jane Hillston's PEPA [42] later further extended CCS. The language was particularly adapted to study properties such as throughput, utilisation, deadlock and livelock, particularly important when studying computer systems. More than ten years later, BioPEPA [17], a variant of PEPA , was developed to specifically deal with biochemical networks.

Process algebra has not been studied very extensively in an epidemiological context. McCaig et al. [60, 59, 58] have studied the use of WSCCS [84], a CCS inspired process algebra, for epidemiological systems. He initially analysed how to describe basic disease features [60] and developed an algorithm [61] to derive Mean Field Equations from a WSCCS model. Once he was able to successfully describe a model, he studied the concept of superspreaders and supershedders [57], which is essential to understand the evolution of diseases such as AIDS or SARS. Ciocchetta and Hillston [18] have also analysed how Bio-PEPA could be used to model epidemiological systems,

and developed a model of avian influenza. Finally, Bradley et al. [13] used PEPA to describe the behaviour of internet worm attacks. While they modelled a computer system, some of the key features are very similar of those witnessed in epidemiological systems.

1.1 PEPA

1.1.1 Background

PEPA was originally developed to evaluate performance of computer and network systems. It has been applied to a range of issues, from the optimisation of the bandwidth resource allocation in a cellular phone network [31], to the analysis of queues and server breakdowns [2]. It has also been used to model the performance and correctness of a robotic workcell designed for an automated manufacturing system [34, 47], or quality of service analysis of high-availability web server.

PEPA is a promising candidate as a tool for disease modelling. Its main strength is the ability to "change scale". A PEPA model is written at the individual level. This means that all the assumptions and parameter values are defined for the behaviour of an individual. This makes designing the model simpler and more reliable for two main reasons. First, modellers sometimes have to rely on their intuition to fill the gaps in the knowledge of the system studied, either because the information is not accessible, and because it would be too expensive (in time or money) to be worth the investment. In that case, it is easier to reason at the individual level, granting more likely to be accurate assumptions. The output of the model however is at the population level, for both stochastic simulations or Ordinary Differential Equations (ODEs). This means that the resulting data describe the population behaviour as a whole, ignoring single individual disparities. This allows us to have a general picture of the problem, and test the impact of low level behavioural assumptions on the population level dynamics.

PEPA also allows the modeller to take advantage of the research that has been done on both stochastic simulations and ODEs. Indeed, a full understanding of an issue involves being able to analyse both single simulations and average behaviour. Other formalisms usually only provide one or the other. ODEs for example only the average behaviour of the model. In general, cellular automata only provide individual stochastic simulations. While it can be argued that the average behaviour can be derived by averaging those simulations, no formal analysis can be performed on the average behaviour in the case of a cellular automaton model. PEPA, on the other hand, gives access to both types of analyses with a supporting analytical theory via several available tools. The one that has been used in this thesis is the PEPA Eclipse Plug-in [85], developed by Edinburgh University. This tool allows several kinds of analyses:

- It can derive the Continuous Time Markov Chain from small models: the full state-space of the model is generated, which allows passage-time and steady state analyses [37, Section 11.5]. It also provides information about the utilisation of different processes (what is the probability that it is in a certain state?), the throughput of actions (how many times per time step is this action fired?) and the distribution of the population (how many processes are in a certain state on average?).

- It can perform stochastic simulation: several types of analyses can be performed using stochastic simulations. Population level analysis gives the average number of processes in each state over time of one or more simulations. The throughput analysis counts the average number of times each action has been fired per time step. Finally, the average response time is also available, typically useful for queuing systems.

- It can derive the ODEs from the model: Algorithms [41, 13, 43] have been developed which allow the ODEs to be derived from PEPA models. The same type of analyses can be per-

formed as for the stochastic simulations. The tool however does not explicitly provide the expression of the ODEs, and only the result of the analysis is available.

Yet, PEPA suffers from drawbacks that might prevent it from being an efficient tool in disease modelling. Indeed, as opposed to cellular automata for example, space cannot currently be easily expressed in PEPA. This can be a major issue in some diseases such as measles or bovine tuberculosis [21]. Also, the set of possible rates is restricted to real (positive) numbers. This means that, the PEPA Eclipse Plug-in does not allow rates to be a function of the number of individuals in a certain state, which is a common way of modelling some features. Finally, communication between processes is also limited. Indeed, it is not possible in PEPA to have two processes that belong to the same subgroup communicating directly. More details will be given on this problem in section 2.2.

It is important to note that the version of the PEPA plugin used here is the 0.0.13, which is not the last available version. The main reason behind the choice to keep this version rather than upgrading to a more up-to-date version lies in the fact that its developers mainly use computer systems based models. During the development of version 0.0.13, we had the chance to collaborate with the developers on several occasions, which made that version more stable and reliable when confronted with the kind of models epidemiological systems require. Also, the version 0.0.13 does not check for deadlocks. In some of the models presented in this thesis, deadlocks may occur. It would have been simple to circumvent them, but it would have only resulted in more complicated models, without any additional benefit. For these two reasons, it has been decided to rely on the version 0.0.13 throughout the thesis.

1.1.2 Syntax

PEPA has a similar philosophy to other process algebras: systems are represented as the composition of *components* which undertake *actions*. In PEPA the actions are assumed to have a duration, or delay. Thus the expression $(\alpha, r).P$ denotes a component which can undertake an α action, at rate r (where rate is 1/delay) to evolve into a component P. Here $\alpha \in \mathcal{A}$ where $\mathcal{A}$ is the set of action types and $P \in \mathcal{C}$ where $\mathcal{C}$ is the set of component types.The following section is largely based on a paper from Benkirane et al. [7].

PEPA has a small set of combinators, allowing system descriptions to be built up as the concurrent execution and interaction of simple sequential components. We informally introduce the syntax below. More detail can be found in [42]. The structured operational semantics are shown in Fig. 1.1.

Prefix: The basic mechanism for describing the behaviour of a system with a PEPA model is to give a component a designated first action using the prefix combinator, denoted by a full stop, which was introduced above. As explained, $(\alpha, r).P$ carries out an α action with rate r, and it subsequently behaves as P.

Choice: The component $P + Q$ represents a system which may behave either as P or as Q. The activities of both P and Q are enabled. The first activity to complete distinguishes one of them: the other is discarded. The system will behave as the derivative resulting from the evolution of the chosen component.

Constant: It is convenient to be able to assign names to patterns of behaviour associated with components. Constants are components whose meaning is given by a defining equation. The

Prefix

$$\frac{}{(\alpha, r).E \xrightarrow{(\alpha,r)} E}$$

Choice

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E + F \xrightarrow{(\alpha,r)} E'} \qquad \frac{F \xrightarrow{(\alpha,r)} F'}{E + F \xrightarrow{(\alpha,r)} F'}$$

Constant

$$\frac{E \xrightarrow{(\alpha,r)} E'}{A \xrightarrow{(\alpha,r)} E'} \; (A \stackrel{def}{=} E)$$

Hiding

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\alpha,r)} E'/L} \; (\alpha \notin L) \qquad \frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\tau,r)} E'/L} \; (\alpha \in L)$$

Cooperation

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E' \underset{L}{\bowtie} F} \; (\alpha \notin L) \qquad \frac{F \xrightarrow{(\alpha,r)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E \underset{L}{\bowtie} F'} \; (\alpha \notin L)$$

$$\frac{E \xrightarrow{(\alpha,r_1)} E' \quad F \xrightarrow{(\alpha,r_2)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,R)} E' \underset{L}{\bowtie} F'} \; (\alpha \in L), \qquad R = \frac{r_1}{r_\alpha(E)} \frac{r_2}{r_\alpha(F)} \min(r_\alpha(E), r_\alpha(F))$$

Figure 1.1: Operational Semantics of PEPA

notation for this is $X \stackrel{def}{=} E$. The name X is in scope in the expression on the right hand side meaning that, for example, $X \stackrel{def}{=} (\alpha, r).X$ performs α at rate r forever.

Hiding: The possibility to abstract away some aspects of a component's behaviour is provided by the hiding operator, denoted P/L. Here, the set L identifies those activities which are to be considered internal or private to the component and which will appear as the unknown type τ.

Cooperation: We write $P \underset{L}{\bowtie} Q$ to denote cooperation between P and Q over L. The set which is used as the subscript to the cooperation symbol, the *cooperation set* L, determines those activities on which the *cooperands* are forced to synchronise. For action types not in L, the components proceed independently and concurrently with their enabled activities. We write $P \parallel Q$ as an abbreviation for $P \underset{L}{\bowtie} Q$ when L is empty.

If a component enables an activity whose action type is in the cooperation set it will not be able to proceed with that activity until the other component also enables an activity of that type. The two components then proceed together to complete the *shared activity*. The rate of the shared activity may be altered to reflect the work carried out by both components to complete the activity. The total capacity of a component C to carry out activities of type α is termed the *apparent rate* of α in P, denoted $r_\alpha(P)$. See section 1.1.3 for the definition.

In some cases, when an activity is known to be carried out in cooperation with another component, a component may be *passive* with respect to that activity. This means that the rate of the activity is left unspecified (denoted $\top$) and is determined upon cooperation by the rate of the activity in the other component. All passive actions must be synchronised in the final model.

If more than one activity with a passive rate can be enabled within a sequential component, a weight can be attached to the passive rate. The weight is a natural number, which determines in which proportion one activity will be enabled. For example, if we have $(\alpha, w_1\top).P + (\alpha, w_2\top).Q$,

then the sequential component has a probability $w_1/(w_1 + w_2)$ to subsequently behave as P, and a probability $w_2/(w_1 + w_2)$ to subsequently behave as Q.

The syntax may be formally introduced by means of the following grammar:

$$S ::= (\alpha, r).S \mid S + S \mid C_S$$

$$P ::= P \underset{L}{\bowtie} P \mid P/L \mid C$$

where S denotes a *sequential component* and P denotes a *model component* which executes in parallel. C stands for a constant which denotes either a sequential component or a model component. C_S stands for constants which denote sequential components. The effect of this syntactic separation between these types of constants is to constrain legal PEPA components to be cooperations of sequential processes, a necessary condition for an ergodic underlying Markov process. We denote the set of all sequential components by $\mathcal{S}$, the set of model components by $\mathcal{M}$, and the set of all activities of a process P by $\mathcal{A}ct(P)$.

1.1.3 Apparent Rate

The *apparent rate* at which an action type occurs within a component is of importance when comparing components or when defining how they interact. We assume that the apparent rate of an action type represents the total capacity of a component to carry out activities of that type when it is in its current state. The following section is largely based on a paper from Benkirane et al. [7].

Definition 1 (Apparent Rate) *The* apparent rate *of action of type α in a component P, denoted $r_\alpha(P)$, is the sum of the rates of all activities of type α in $\mathcal{A}ct(P)$.*

1. $r_\alpha((\beta, r).P) = \begin{cases} r & \text{if } \beta = \alpha \\ 0 & \text{if } \beta \neq \alpha \end{cases}$

2. $r_\alpha(P + Q) = r_\alpha(P) + r_\alpha(Q)$

3. $r_\alpha(P/L) = \begin{cases} r_\alpha(P) & \text{if } \alpha \notin L \\ 0 & \text{if } \alpha \in L \end{cases}$

4. $r_\alpha(P \underset{L}{\bowtie} Q) = \begin{cases} \min(r_\alpha(P), r_\alpha(Q)) & \text{if } \alpha \in L \\ r_\alpha(P) + r_\alpha(Q) & \text{if } \alpha \notin L \end{cases}$

Unlike some other stochastic process algebras, PEPA assumes *bounded capacity*: a component cannot be made to perform an activity faster by cooperation, so the rate of a shared activity is the minimum of the rates of the activity in the cooperating components.

The apparent rate will be *undefined* for component expressions containing *unguarded* rates, i.e. rates which are not prefixed by an activity. Consequently we do not allow a component to be defined by such an expression.

Note that in cases of cooperation, the apparent rate of the shared activity will be the minimum of the apparent rates of the components involved, where

$$\begin{aligned} m\top < n\top &: \text{ for } m < n \text{ and } m, n \in \mathbb{Q} \\ n\top + r = n\top &: \text{ for } n \in \mathbb{Q}, r \in \mathbb{R} \\ r < n\top &: \text{ for all } r \in \mathbb{R}, n \in \mathbb{Q} \\ m\top + n\top = (m+n)\top &: \text{ for all } m, n \in \mathbb{Q} \\ \frac{m\top}{n\top} = \frac{m}{n} &: \text{ for all } m, n \in \mathbb{Q} \end{aligned}$$

For passive actions bounded capacity means that the corresponding component does not contribute to the apparent rate of the shared activity.

1.1.4 Additional definitions

Additional definitions will be used throughout this thesis. First however, a small PEPA model will be used to illustrate those definitions.

$$C \stackrel{def}{=} (\alpha, r).C'$$

$$C' \stackrel{def}{=} (\beta, r').C$$

$$B \stackrel{def}{=} (\alpha, \top).B$$

$$C[c] \underset{\alpha}{\bowtie} B[b]$$

In this model, there are initially c instances of C and b instances of B. C and B communicate over the action α at rate r and $\top$ respectively ($\top$ meaning that B is passive). When the communication is performed, an individual in C "moves" to C' while the individual in B remains in B. Finally, individuals in C' "move" back to C at rate r'.

Definition 2 (Component type) *A component type of a PEPA model corresponds to a state an individual can be in. It can also be referred to as component.*

The model presented previously contains three component types: C, C' and B.

Definition 3 (Sequential component) *A sequential component is the representation of an individual in the model. The local state of a sequential component at time t corresponds to the component in which the sequential component is at time t. Sequential components are also referred to as agents.*

The total number of sequential components is constant throughout the simulation, as PEPA does not allow them to be created or deleted. The number of sequential components in each

component type however varies over the course of the simulation. In the model presented earlier, they are $b + c$ sequential components. Initially, b of them are in state B and c of them are in state C.

Definition 4 (Model component) *The model component is the equation which defines the initial state of the model as well as the cooperation sets between the different component types.*

In the example, the model component is $C[c] \underset{\alpha}{\bowtie} B[b]$ and $\underset{\alpha}{\bowtie}$ corresponds to the cooperation set between the sequential components initially in C and the ones initially in B.

Definition 5 (Subgroup) *Two component types C_1 and C_2 belong to the same subgroup if and only if there exist n activities $a_1, ..., a_n$ such as $C_1 \xrightarrow{a_1} ... \xrightarrow{a_n} C_2$ or $C_2 \xrightarrow{a_1} ... \xrightarrow{a_n} C_1$. $SG(S)$ refers to the subgroup to which S belongs.*

This means that a subgroup is a group of components that can "reach" one another. In our example, we have two subgroups: $\{C, C'\}$ and $\{B\}$.

1.2 BioPEPA

BioPEPA [17] is a modification of PEPA developed in 2009 to handle some biological features such as stoichiometry and general kinetic laws. It was specifically aimed at describing biochemical networks. While based on PEPA, BioPEPA has several conceptual differences with PEPA and is considered a different process algebra in the context of this study. The objective of this section is to answer the common question: why did we choose to study PEPA and not BioPEPA.

The major feature of BioPEPA is the introduction of functional rates to express general kinetic laws. This is particularly useful in epidemiology as it allows to reproduce some of the well-studied terms previously modelled (usually using ODEs). For example in the case of births, the Verhulst-Pearl equation dN/dt=$birth\_rate \times N.(1 - N/K)$, with N the number of individuals capable of giving birth, and K the carrying capacity [86] can be directly expressed using BioPEPA, while it is not possible to do so in simple terms using PEPA, as detailed in section 5.2, mainly because it depends on the population size. BioPEPA also added the option to include space in the form of discrete compartments. While discrete space might not be sufficient in some cases, it allows a significant number of systems where space is an essential feature to be described.

At first sight, BioPEPA seems to be an extension of PEPA more adapted to disease modelling in every way. Most of PEPA's drawbacks have been dealt with. Functional rates are included, and a lot of flexibility is allowed when defining them. It is possible to include compartments which could describe cities or burrows, which means that space can be added to a model.

The way rates are expressed however denotes a major difference between PEPA and BioPEPA. In the case of PEPA, the rate represents how long it takes a sequential component to perform a certain action, regardless of the rest of the system. In case cooperation is required, it can only be performed when two sequential components in the right state are ready to perform it. BioPEPA on the other hand explicitly gives the overall rate of the reaction, which corresponds to a population level representation of the rate. Similarly, component types have a concentration rather than a number of sequential components in this particular state. While a modification to the grammar now allows to explicitly represent individuals, the natural way of modelling in BioPEPA remains describing chemical species.

In conclusion, BioPEPA might be a good candidate for modelling diseases. It has many useful features PEPA is lacking, and looks like the perfect candidate to model diseases. It is hard however

to compare the two process algebras: while the model is written at the individual level in the case of PEPA, it is described at the population level in BioPEPA. As a result, BioPEPA cannot be considered an extension of PEPA, and their respective characteristics make them valuable for different types of systems and analyses.

1.3 Presentation of the running examples

The objective of this thesis is to assess to what extent PEPA can be considered a valid tool in the field of epidemiology and to extend the formalism and its available tools when required. This will be achieved by modelling a series of features that are considered important for disease modelling. Based on this first step, the required additions to the formalism will be developed. In order to determine which features are important, and if PEPA can accurately model them, two epidemiological systems have been chosen, which cover the most essential features required. The first disease, the bubonic plague in a population of prairie dogs, is an interesting disease to study as it includes two main vectors of transmission, indirect and direct transmission (more details in section 5.5), and the question of which of these vectors drives the epidemic has been an open question until very recently. While data is not freely available to validate the model, it is still possible to do so by comparing the results to those obtained in previous papers [92], whose results were validated against a (non-available) set of data. The second disease, measles, is a very different disease, as its behaviour is fairly simple, but some features such as immigration or seasonality are important. It is also easier to include space in the measles model as it is less complex. Finally, a large data set is freely available which covers the reported the number of cases of measles in England and Wales over more than 20 years.

1.3.1 The Bubonic Plague in a community of Black-Tailed Prairie Dogs

Prairie dogs are rodents within the squirrel family [90] located exclusively in North America. The black-tailed prairie dog is a very sociable animal, living in towns of hundreds of animals, which are collections of families consisting of 1 male and 1 to 4 females [79]. Contacts between members of a colony are very frequent. Indeed, the town is very well connected, with several corridors in and out of each family burrow. Contact is very close: in order to recognize each other, prairie dogs seem to"kiss": they actually gently touch their front teeth together.

Black-tailed prairie dogs are on the verge of being placed on the list of endangered species. Indeed, since the arrival of the European settlers, the number of prairie dogs in the American prairies has decreased by 98% [79]. They have suffered from a bad reputation for several reasons (and still do [26]). They have been thought to compete with cattle and bison over grass. The holes they dig in the ground are hazardous to horses. In addition to this, the bubonic plague has almost exterminated them. However, recent studies [3] have shown that they are actually a keystone species in the grassland ecosystem, and eradicating them would have a devastating effect on both the fauna and the flora.

The bubonic plague is an infection of the lymphatic system. It is mainly transmitted by the bite of an infectious flea [96, 15, 1]. The bite causes the plague bacteria, *Yersinia pestis*, to replicate in the host's body. This will result in the appearance of swollen lymph nodes, which are very painful and can evolve into a suppurating open sore.

The plague mainly affects wild rodents, but is still an issue in human populations and animal to human transmission is possible in some rare cases. If left untreated, the plague kills its human host in 30 to 60% of cases. According to WHO [96], in 2003, 9 countries reported 2118 cases and 182 deaths. As for prairie dogs, because they had never been exposed to the bubonic plague before

it was introduced at the beginning of the 20th century, they had no natural defences against it, which results in death within a week of infection for almost 100% of the infected animals.

In the context of our analysis, this biological and epidemiological system has been chosen mainly to illustrate different types of transmission, since the two main types play an essential role in the behaviour of the system. Births and deaths are also an important feature in the system, and it will be used to demonstrate how the theory developed can be used in a model.

1.3.2 Measles in England and Wales between 1944 and 1964

Despite the worldwide efforts to vaccinate children against it, measles is still the vaccine-preventable disease of childhood that causes the most deaths [22]. According to UNICEF [88], 875.000 people died of measles in 1999, most of them in Africa and South-East Asia. While it is generally a very benign disease, malnutrition, and in particular vitamin A deficiency, can cause very serious complications [78], ranging from diarrhoea or otitis to pneumonia or blindness, and can even lead to death.

Without any vaccination, measles infects 95-98% of children before they turn eighteen [71], and an infectious person will infect 75%-90% of susceptible household contacts. The incubation lasts for six to nine days [9], then measles begins with increasing fever, cough, coryza and conjunctivitis [71]. This is when the infectivity is highest. A rash then appears on the face and the neck, and may spread to the rest of the body. Usually, the individual recovers after six to seven days and then has a lifelong immunity to the disease.

Measles was probably first described when the so-called *Antonine Plague* ravaged the Roman army and killed one-third of the population of some areas between 165 and 180 AD. It has since

killed, among others, two-thirds of Cuba's native inhabitants in 1529, half the population of Honduras in 1531, a fifth of the Hawaiian population in the 1850s, and, along with influenza and pneumonia, decimated the Adamanese population in the 19th century, as they had not previously been exposed to these diseases. Overall, it is estimated to have killed about 200 million people worldwide over the last 150 years [83].

In 1954, the virus responsible for the disease was isolated, which led to the creation of the vaccine [25] in 1959 that was first licensed in 1963 [83]. This led to a campaign of mass vaccination in developed countries, resulting in the near eradication of the disease. However, it is still very deadly in developed countries [88], with a death ratio ranging from 3% to 34% [71]. Recently, a rise in the number of cases in the last 10 years has been witnessed, resulting from the diminishing vaccination coverage [33, 69, 16]. This is due to parents becoming unwilling to vaccinate their children as the side effects of the vaccine (fever or measles-like rash) currently outweigh the risk of getting the disease. On the other hand, the vaccination coverage has increased quickly in developing countries, reducing the number of deaths caused by the disease by 78% between 2000 and 2008 [62].

Measles has a very different profile compared to the bubonic plague. The transmission mechanism is trivial, which simplifies the model greatly. On the other hand, the population involved is much larger, at the scale of cities or even countries, which reveals different types of problems related to PEPA. Finally, space is a more important feature in this case, as it is believed to be the reason behind the epidemic cycles observed. Lastly, a detailed set of data is freely available, which allows the validation of the results obtained using PEPA. Due to the nature of this data, this thesis will focus on the number of cases of measles in England and Wales between 1944 and 1964. In 1944, national notification of measles patients was made mandatory in England and Wales [27]. This provided some very good data on the number and location of measles cases, with a reporting rate of over 50% [9]. Mass vaccination was introduced in 1968, changing entirely the landscape

of the disease in those areas. However, the administrative creation of "Greater London" changed the boundaries of the city in 1965. For this reason, it is not possible to draw a consistent map of England for the whole 1944-1968 period, so the focus will be on the 1944-1964 period only.

1.4 Structure of the thesis

The objective of this thesis is to determine to what extent PEPA is suitable for disease modelling, and to extend it where necessary. In order to achieve this, several essential epidemiological features will be studied to evaluate the possibility to model them in PEPA, and when necessary and possible, solutions will be provided for each of them. It is important to note that, unless stated otherwise, the parameters are not based on any actual biological system. This is due to the fact that we are only interested in knowing if PEPA can model the studied features, and we do not try to gain any insight into the diseases which are being represented.

For this purpose, this thesis has been divided into six chapters. The first chapter presents the two main types of transmission mechanisms, indirect and direct transmission, and presents the first issues related to PEPA's limitations. The second chapter then explains how to derive ODEs from a PEPA model, and the limitations of the developed algorithm. It then presents software that automatically executes the algorithm on a PEPA model to derive the ODEs. Finally, an example shows how the ODEs can be a useful tool to have in addition to having access to the stochastic simulations, and that the algorithm is also adapted to computer systems models as well as biological ones. The third chapter analyses how frequency and density dependent transmission can be expressed in PEPA. Births and deaths are then dealt with in chapter four, with different methods to model them presented in detail. Since the features required have been studied in a different work (in [92] in the case of the plague and in [9] in the case of measles), the model of the

bubonic plague in a prairie dog town is then presented and analysed. Chapter five deals with timed events, and describes how intervention, the circadian clock and seasonality can be modelled. These features are then used in the examples of control policies and measles in Leeds, the latter being analysed in detail and compared to the data available. Finally, the sixth chapter presents how space, in the form of compartments, can be used in PEPA, and a new grammar is presented which allows the PEPA models with space to be more concisely presented, before software translates it to a valid PEPA model. Space is then added to the examples of the bubonic plague and measles analysed earlier. Finally, we will conclude on PEPA's main strengths and weaknesses, and review the perspectives for PEPA in an epidemiological context.

Chapter 2

First epidemiological models

As a first step in applying PEPA to epidemiology, the different forms of transmission will be modelled, as it is a fundamental mechanism in every disease. These first models, presented in a paper from Benkirane et al. [7] will be based on the SIR model, first described by Kermack and McKendrick [51] in 1927. SIR corresponds to *Susceptible*, *Infectious* and *Recovered*, and corresponds to a general model of disease spread which has since been applied to a wide range of different diseases. The model splits the population into three groups:

Susceptibles represent the people that never had the disease, and may acquire it after exposure to the infection.

Infectious are the people who carry the disease, and may pass it (directly or indirectly) to *susceptible* individuals.

Recovered (or Removed) are immune to the disease. This might be because they have been infectious and recovered from it, because they have been vaccinated, or because they are

naturally immune to the disease. The immunity might be lifelong or temporary, in which case the individuals might become *susceptibles* again.

This basic pattern needs to be adapted to each disease, and additional classes might be necessary. Here are a few common ones:

Exposed : once infected, a *susceptible* is not usually immediately infectious. An incubation period is normally necessary. The infectivity usually coincides with the start of the first symptoms, but it is not the case for every disease.

Superspreaders are *infectious* individuals that are more likely to infect *susceptibles* than "normal" *infectious* individuals. This can be either because they have more contacts with other individuals (children at school, nurses) or because they are more infectious (in the case of the human influenza, some people produce more bioaeorosols [64]), and the probability for a contact to successfully pass the disease is higher. Superspreaders have a significant impact on the behaviour of a disease, such as AIDS [48], or SARS [56].

These classes can be used to create variants of the standard SIR model. SEIR (including an incubation period), SIRS (the *recovereds* may lose their immunity), SEIRS (with both features - this is the one used in this section), SIS (when there is no immunity) are frequently used in epidemiology [52, 53].

The most commonly used formulation of the SEIRS model is:

$$\begin{aligned} \frac{dS}{dt} &= -\beta SI/N + \alpha R \,, \\ \frac{dE}{dt} &= -\delta E + \beta SI/N \,, \\ \frac{dI}{dt} &= -\gamma I + \delta E \,, \\ \frac{dR}{dt} &= -\alpha R + \gamma I \,, \end{aligned} \tag{2.1}$$

where β is the effective transmission rate, δ is the incubation rate, γ is the rate at which the infectious individual recovers and becomes immune, α is the rate of losing immunity and becoming susceptible again, and $N = S + E + I + R$. This standard model will be used in the later sections of this chapter and in the next chapter as a benchmark.

McCaig [57, page 137-138] showed in his work on WSCCS (Weighted Synchronous Calculus of Communicating Systems) [84] and epidemiology that in process algebra, all forms of transmission can be reduced to either *direct* or *indirect* transmission. Being able to model such transmission mechanisms is a necessary condition (but obviously not sufficient) in modelling a disease. It is therefore essential to verify that PEPA has the capacity to perform this task. For this purpose, both transmission mechanisms will be modelled. First, a model including indirect transmission, which can be more intuitively translated into PEPA, will be presented. Then a second model will describe how direct transmission can be approached in PEPA.

2.1 Indirect Transmission

Indirect transmission corresponds to the case where host-to-host contact does not result in passing the infection, but a intermediary is needed for that purpose. This intermediary can have different forms [68]:

Airborne: refers to the case where dust particles containing microorganisms residue from evaporated droplets (droplets nuclei) generated when the host coughs, sneeze or talks, can remain suspended in air for extended periods of time, and may travel over large distance. Diseases, such as influenza, pneumonia, tuberculosis, or polio, rely on airborne transmission [68].

Fecal-oral: corresponds to diseases that infect the digestive system, resulting in infectious faeces. Without proper hygiene measures, this can result in secondary infection via infected water, infected food if handled with the presence of faeces, or directly handling faeces or anything that has been in contact with faeces. Typhoid, for example, mainly relies on this mode of transmission.

Fomite: some organisms are capable of surviving on hard surfaces (fomites) such as door handles, medical instruments, computer keyboards and mice for an extended period of time. Any contact with the infected surface might result in an ingestion of the organism. The list of diseases that can transmit via fomites includes *Norwalk virus*, a virus that causes vomiting and diarrhoea, conjunctivitis or *Escherichia coli* infection.

Vector-borne: Vectors are animals that can transmit the disease from one host to another. Mosquitoes (malaria), fleas (bubonic plague), rats (rabies), ticks (Louping Ill Virus) can be potential vectors in different diseases. The behaviour of the disease largely depends on the vector, but the latter can potentially remain infectious for extended periods of time, and travel over long distances.

Obviously, it is possible for a single disease to spread via several transmission mechanisms. However, the infectiousness or the time before the infectious material decays may vary depending on the mechanism, which may result in in a difference in disease transmission.

The model of indirect transmission presented in Figure 2.1 is a simple SEIRS model, with a surface (similar to the fomite) acting as the agent of transmission. The amount of available environment that can be infected is finite and the infectiousness decays probabilistically. While having limited amount of environment might be considered unrealistic, it is possible to increase it in order to avoid depleting the healthy environment during the stochastic simulation runs. In this

$$
\begin{aligned}
S &\stackrel{def}{=} (contact, \top).E \\
E &\stackrel{def}{=} (infected, ir).I \\
I &\stackrel{def}{=} (infect\_env, ier).I + (contact, \top).I + (recover, rr).R \\
R &\stackrel{def}{=} (contact, \top).R + (lose\_immunity, li).S \\
InfEnv &\stackrel{def}{=} (contact, cr).InfEnv + (decay, dr).Env \\
Env &\stackrel{def}{=} (infect\_env, \top).InfEnv \\
S[990] \parallel I[10] &\underset{\{contact, infect\_env\}}{\bowtie} Env[10000]
\end{aligned}
$$

Figure 2.1: Indirect transmission

case, we can consider the amount of healthy environment infinite, as the model behaves as if it was infinite.

The agents S, E, I and R respectively represent the *Susceptible*, *Exposed*, *Infectious* and *Recovered* individuals, while *InfEnv* and *Env* represent the infectious and the healthy environment. The $\{S,E,I,R\}$ and the $\{InfEnv,Env\}$ subgroups do not communicate within subgroups, as indicated by the $\parallel$ operator. This means that S, E, I and R do not communicate with each other, and neither do *Env* and *InfEnv*. However, they do communicate between subgroups. For example, in a communication, one agent in S can communicate with one agent in *InfEnv*.

Basically, the infection spreads as follows:

- Each I repeatedly (at rate ier) carries out the *infect_env* action, infecting a healthy environment agent, which becomes *InfEnv*. I also recovers at rate rr.
- Each *InfEnv* agent carries out the *contact* action at rate cr to infect S, I or R agents. Obviously, it only succeeds in passing on the infection when an S is contacted. It also loses infectiousness at rate dr.

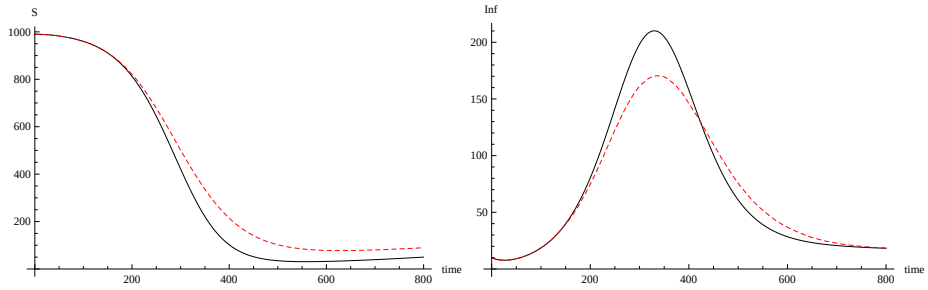

Figure 2.2: Comparison between the standard ODEs and the PEPA model for the indirect transmission example for the susceptible (left) and the infectious (right) individuals. For both graphs, average of 1000 stochastic simulations (dotted line), modified standard ODEs (solid line)

- Once an S is contacted, it has an incubation period of $1/ir$ time steps (the E state), and then becomes infectious (I).
- R is immune to the disease, but loses its immunity at rate li and becomes S again.

The standard system of equations 2.1 cannot be used here. Indeed, its transmission mechanism corresponds to direct transmission, described in the next section. However, this system can be adapted in order to incorporate a vector:

$$\begin{aligned}
\frac{dS}{dt} &= -\beta S.InfEnv/N + \alpha R \ , \\
\frac{dE}{dt} &= -\delta E + \beta S.InfEnv/N \ , \\
\frac{dI}{dt} &= -\gamma I + \delta E \ , \\
\frac{dR}{dt} &= -\alpha R + \gamma I \ , \\
\frac{dInfEnv}{dt} &= -\eta InfEnv + \zeta I
\end{aligned} \tag{2.2}$$

To obtain this model, the standard system (2.1) was modified to describe the new transmission mechanism. In particular, the path of infection is now different: S is infected by $InfEnv$, and

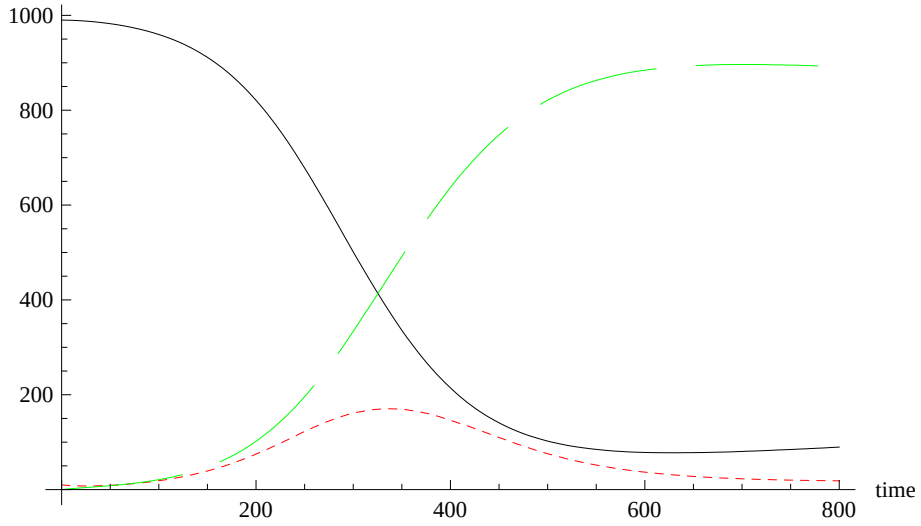

Figure 2.3: Indirect Transmission. S (solid line), I (dotted line), R (dashed line). Parameter values: $cr = 0.08, li = 0.0005, ir = 1, rr = 0.02, ier = 0.01, dr = 0.005$.

InfEnv is infected by I. The output of this system of ODEs is compared to the average of 1000 stochastic simulations in Figure 2.2. The parameters, while not representative of any particular disease, have been chosen to be biologically realistic. Even if both graphs are not exactly similar, the overall behaviour of the two models is very similar. Figure 2.3 represents the evolution of the S, I and R population in the average of 1000 simulations of the model presented in Figure 2.1. The graph output corresponds to the typical behaviour of a disease during an outbreak, as can be seen in the literature [50, 91, 30].

We can now move on to direct transmission.

2.2 Direct Transmission

Direct transmission [68] requires a physical contact between a *susceptible* and a *infectious* individual for the disease to spread. Two different direct transmission mechanisms exist:

Direct contact: the disease requires physical contact of the infectious and susceptible hosts, as the microorganisms responsible for the disease requires a host to live. The infection is usually passed by body contact, as in the case of herpes simplex virus, or via body fluids, as for Sexually Transmitted Diseases (STDs) or conjunctivitis.

Droplets: in this case, the infectious person coughs or sneezes and generates droplets that can enter another organism via the eye, nose or the mouth. Measles, influenza or tuberculosis rely on this type of transmission mechanism. The difference with airborne transmission is the fact that the droplets are short-lived, and will not remain in the air for a long time.

Obviously, a disease can rely on several different mechanisms, direct or indirect, to spread. The Bubonic Plague in prairie dogs [92], for example, can transmit via direct contact, via the animal's faeces, or via fleas.

Direct transmission is less natural and intuitive to model in PEPA than indirect transmission. Indeed, the PEPA syntax does not allow only two agents in the same subgroup to communicate. If the communication happens between an agent of component type A and an agent of component type B, with A and B in the same subgroup, then **all** agents of component type A and B at time t are required to participate in the communication. This is obviously not what is needed here. Instead, one I should communicate with one S at any given time step. One solution would be to change the way PEPA behaves in this situation. The choice has however been made to operate within the constraints of standard PEPA. Therefore, it was necessary to "split" the I agent in two. On the one hand, we have I as an individual. It can be contacted, or recover. On the other hand, we have the *transmitter* side of I, the side of it that is infectious, and is trying to contact other individuals. At all times, the number of I and the number of *transmitter* are equal, which guarantees that the model remains consistent. Because the number of I changes throughout the

lifetime of the model, the number of *Transmitter* must also change. However, it is not possible to create (or delete) new agents in PEPA. For this reason, a new component type, *Dormant* has been created. Effectively, the *Dormant* are waiting to be activated as *Transmitter*. A side effect of the constraint that $I = Transmitter$ is that $Dormant = S + E + R$. The components *Transmitter* and *Dormant* constitute what will be referred to in section 4.1 as the *mirror subgroup* of the $\{S, I, R\}$ subgroup.

The PEPA model for direct transmission can be seen in Figure 2.4. S, E and R have the same behaviour they had in Figure 2.1. The disease spreads as a *Transmitter* contacts an S, which becomes exposed. Once the incubation is over, E and *Dormant* communicate over the action *infected* and E becomes I, while *Dormant* becomes *Transmitter*. I recovers at rate rr, and while doing so, communicates with *Transmitter*, so they respectively move to R and *Dormant*. These activities synchronise the number of I and *Transmitter*.

$$
\begin{aligned}
S &\stackrel{def}{=} (contact, \top).E \\
E &\stackrel{def}{=} (infected, ir).I \\
I &\stackrel{def}{=} (contact, \top).I + (recover, rr).R \\
R &\stackrel{def}{=} (contact, \top).R + (lose\_immunity, li).S \\
Transmitter &\stackrel{def}{=} (contact, cr).Transmitter + (recover, \top).Dormant \\
Dormant &\stackrel{def}{=} (infected, \top).Transmitter \\
S[990] \parallel I[10] &\underset{\{infected, contact, recover\}}{\bowtie} Transmitter[10] \parallel Dormant[990]
\end{aligned}
$$

Figure 2.4: Direct transmission

The model resembles the one presented in Figure 2.1. However, the role of *Transmitter* is very different from the one of *InfEnv*. Indeed, in the indirect transmission model, the environment is infected by the infectious individual regularly. The amount of *InfEnv* is usually different from

the number of I, as a single I can infect several pieces of environment, and the environment can remain infectious after the agent in I that infected it has recovered. In the case of *Transmitter*, only one agent is moved from *Dormant* to *Transmitter* each time an agent moves from E to I, and it is moved from *Transmitter* to *Dormant* again when I recovers, leading to a perfect match between the number of agents in I and *Transmitter*.

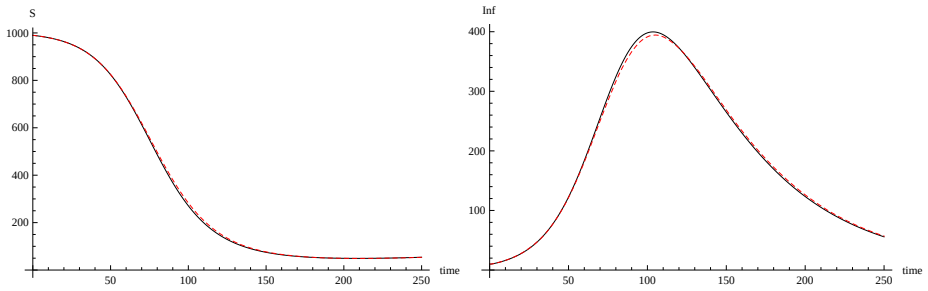

Figure 2.5: Comparison between the standard ODEs and the PEPA model for the direct transmission example. For both graphs, average of 1000 stochastic simulations (dotted line), modified standard ODEs (solid line)

Figure 2.5 shows the comparison between the average behaviour of the model in Figure 2.4 and the standard ODEs of equation (2.1) for S and I. The graph shows a nearly perfect match between the two methods. Figure 2.6 represents the evolution of the S, I and R population in the average of 1000 simulations of the model presented earlier, under the same parameter settings as indirect transmission. The parameters have been chosen with the same mindset as in section 2.1: they do not strictly represent a disease, but have been chosen to be biologically realistic. And once again, the results show the kind of behaviour expected.

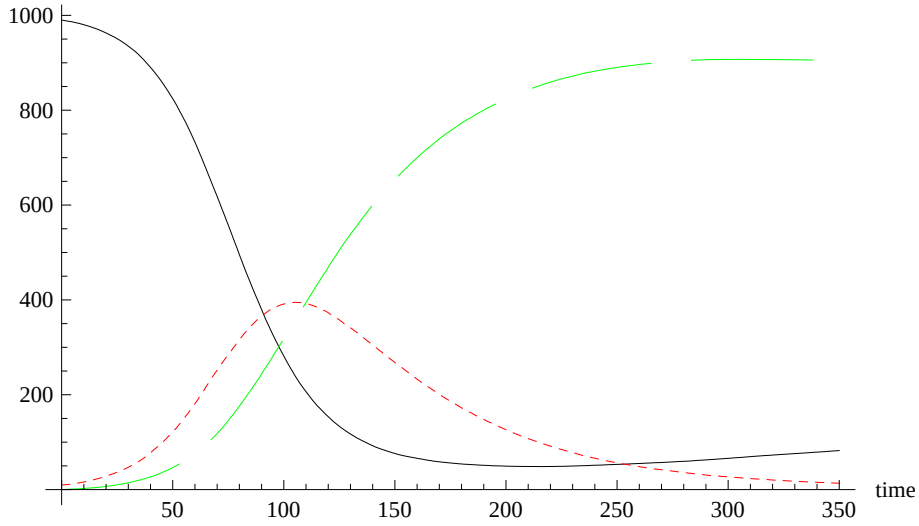

Figure 2.6: Direct Transmission. S (solid line), I (dotted line), R (dashed line). Parameter values: $cr = 0.08, ir = 1, li = 0.0005, rr = 0.02$.

2.3 Summary

This chapter has demonstrated that PEPA can describe the two essential types of transmission in epidemiology, direct and indirect. It is however already obvious here that PEPA's limitations might affect the systems that can be expressed using this formalism. As seen with the direct transmission example, some features might require the use of counter-intuitive solutions to be described.

Chapter 3

Deriving ODEs from a PEPA model

Before studying the bubonic plague and measles, a need will be addressed in this chapter. Indeed, while stochastic simulations are very powerful and often required to the study of some issues, having access to the average behaviour of the model can also provide important information on the disease.

Ordinary Differential Equations (ODEs) have been used for a long time in modelling, and especially for disease modelling. This has resulted in a lot of experience and expertise in using the formalism for this purpose, and makes it the perfect complementary tool to modelling in PEPA. However, the changes in a PEPA model are inherently discrete. Indeed, when an activity takes place at time t, one or several agents change from one state to another. For example, if a component type C allows the activity $(action\_name, rate).TargetComponent$, the action $action\_name$ for an

agent in C happens at a specific time t_0. This means that the number of agents in C decreases by one between $t - \epsilon$ and $t + \epsilon$, $\forall \epsilon \in \mathbb{R}+$, with ϵ sufficiently small. This does not seem to be compatible with continuous Ordinary Differential Equations (ODEs). A hypothesis on the PEPA model allows the approximation to hold: if the number of agents is large enough, the stochastic simulations exhibit a behaviour resembling the one of a continuous model. Finally, using ODEs to study PEPA models brings scalability to the formalism. Indeed, process algebras in general cannot deal with models with a large population. As the computation is performed at the population level, the ODEs, on the other hand, do not suffer from this drawback.

In this chapter, the existing algorithm, the Hillston Method [43], is first presented. An improvement of the method that I developed, the Stirling Amendment, is then described, and the two algorithms are compared. These first two sections have been presented in the paper from Benkirane et al. [7]. In section 3.3, the Dining Philosophers Problem is translated to PEPA and analysed using the ODEs derived from the model, showing a possible use of the ODEs. Finally, software, PEPAdum, automatically deriving ODEs from a PEPA model using the Stirling Amendment is presented.

3.1 Existing methods

3.1.1 Jane Hillston Method

The first algorithm to generate ODEs was provided by Hillston [43]. It was originally created to deal with systems with a large number of replicated components, where stochastic simulations could not be performed, and specifically designed to deal with computer systems. This section will present the key elements of this algorithm, as presented in the paper by Hillston [43].

Hillston restricts the set of PEPA models to which the algorithm can be applied. The permitted models need to feature certain criteria:

1. Cooperating components must share the same local rate (no passive components).
2. Actions may not be hidden.
3. Cooperation must include all common actions.
4. Components cannot cooperate if they belong to the same subgroup.

Through the models presented in chapter 2 we discovered a fifth limitation:

5. There should not be implicit choice between independent but distinct components offering the same action for synchronisation. In other words, two sequential components that belong the same subgroup cannot allow the same action to be fired.

First, Hillston defines what an entry and an exit activity are:

Definition 6 (Exit Activity) *An activity (α, r) is an exit activity of D if D enables (α, r), i.e. there is a transition $D \xrightarrow{(\alpha,r)}$ in the derivation graph of D. We denote the set of exit activities of D by $Ex(D)$.*

Definition 7 (Entry Activity) *Similarly, an activity (β, s) is an entry activity of D if there is a derivative D' which enables (β, s), and D is the one-step β-derivative of D', i.e. $D' \xrightarrow{(\beta,r)} D$ is in the labelled transition system of D'. We use $En(D)$ to denote the set of entry activities of D.*

These definition are necessary as the impact of an activity on each component type needs to be recorded. Finally, the *activity matrix* is defined:

Definition 8 (Activity Matrix) *For a model with N_A activities and N_D distinct local derivatives, the activity matrix M_a is an $N_D \times N_A$ matrix, and the entries are defined as follows:*

$$(n_i, a_j) = \begin{cases} +1 & \textit{if } a_j \textit{ is an entry activity of } n_i \\ -1 & \textit{if } a_j \textit{ is an exit activity of } n_i \\ 0 & \textit{otherwise.} \end{cases}$$

Each row of the activity matrix represents a component type, while each column represents an activity. A cell records whether the activity is an entry or exit activity for the component type. In the final system of ODEs, there is one equation of each component type. The algorithm for generating those ODEs is detailed in Figure 3.1.

The Hillston algorithm has been written with computer systems based models in mind. This implies that some features vital to epidemiology have been left out. In particular, as detailed in [7], because self-loops are uncommon in computer systems, the Hillston algorithm has been written ignoring the influence they can have on the system and the actions involving a self-loop are dealt with as if they were absent from the model. In the context of a single agent, this makes sense: in a indirect transmission system for example as presented in Figure 3.2, the number of agents in I remains constant when the action *infect_env* is performed. However, when the impact of the action is considered at the scale of the whole system, this does not make sense anymore: indeed, as it is involved in the communication with *Env*, the number of times the infectious agents perform the action *infect_env* have an impact on the number of agents moving from *Env* to *InfEnv*.

The model presented in Figure 3.2 is a modified version of the model presented in Figure 2.1 of section 2.1, in order to ensure that the restrictions required by the Hillston algorith are met. Two changes have been necessary for this. First, in order not to violate condition (5), the action *contact* now only involves *InfEnv* and S. Also, the passive rates have all been replaced in order

//Form one ODE for each local derivative/state variable

`For` $i = 1...N_D$

//Find the activities involving this derivative

`For` $j = 1....N_A$

`If` $M_a(i,j) \neq 0$

`//Form exit set` $Ex(j)$ `for activity j`

$Ex(j) = \emptyset$

`For` $k = 1...N_D$

`If` $M_a(k,j) = -1$

$Ex(j) = Ex(j) \cup \{k\}$

//Record the impact of each such activity

`If` $M_a(i,j) = +1$

Add

$$+r_j \times \min_{k \in Ex(j)}(n_k(t))$$

to the equation.

`If` $M_a(i,j) = -1$

Add

$$-r_j \times \min_{k \in Ex(j)}(n_k(t))$$

to the equation.

Figure 3.1: Pseudo-code of the Hillston Algorithm as presented in [43].

$$
\begin{aligned}
S &\stackrel{def}{=} (contact, cr).E \\
E &\stackrel{def}{=} (infected, ir).I \\
I &\stackrel{def}{=} (infect\_env, ier).I + (recover, rr).R \\
R &\stackrel{def}{=} (lose\_immunity, li).S \\
InfEnv &\stackrel{def}{=} (contact, cr).InfEnv + (decay, dr).Env \\
Env &\stackrel{def}{=} (infect\_env, ier).InfEnv \\
S[990] \parallel I[10] &\underset{contact, infect\_env}{\bowtie} Env[10000]
\end{aligned}
$$

Figure 3.2: Indirect transmission model adapted to comply to the restrictions required by the Hillston algorithm.

to make sure that all the activities involved in a cooperation share the same rate (condition (1)). The ODEs of this model have been derived using the Hillston Method:

$$
\begin{aligned}
dS/dt &= -cr.S + li.R \\
dE/dt &= -ir.E + cr.S \\
dI/dt &= -rr.I + ir.E \\
dR/dt &= -li.R + rr.I \\
dInfEnv/dt &= -dr.InfEnv + ier.Env \\
dEnv/dt &= -ier.Env + dr.InfEnv
\end{aligned}
$$

Even before plotting the ODEs against the average of stochastic simulations, some terms intuitively seem not to represent the behaviour of the model properly. Indeed, each subgroup behaves independently of the other. This is due to actions involved in a cooperation between the SEIR and the environment subgroups to be a self-loop on one side, and thus the influence of the SEIR subgroup and the environment subgroup is ignored for the *infect_env* and the *contact* actions

respectively. This is further shown by direct comparison of the ODEs with the average of 1000 stochastic simulations for I presented in Figure 3.3. Because the term generating Is, that is $cr \times S$, only depends on the number of susceptible individuals, and does not depend on the number of infectious environments, the number of Is quickly rises in the case of the ODEs, while the change is slower in the case of the simulations, as it depends on both the number of S and the number of $InfEnv$, which is initially zero.

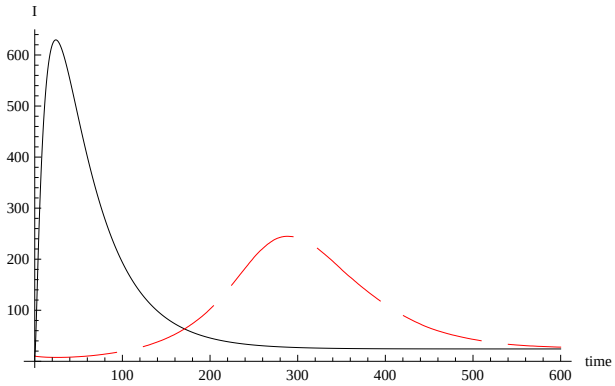

Figure 3.3: Indirect transmission model adapted to comply to the restrictions imposed by the Hillston Method: average of 1000 simulation (dashed line) and ODEs derived with the Hillston Method (solid line)

The Hillston Method is the first existing algorithm designed to derive ODEs from a PEPA model. After that, the Stirling Amendment was published, and several additional methods have since been developed, that are more general than the Hillston method. The Stirling Amendment is presented in the next section, and the other existing algorithms are then briefly presented in the following section to allow comparison with the Stirling Amendment.

3.1.2 The Stirling Amendment

The Hillston Method is not suitable for deriving ODEs of epidemiological models. First, as shown in section 3.2.1, the ODEs obtained do not provide a good match to the average behaviour of the model in the case of self-loops. Also, the conditions imposed by the Hillston Method are too constricting and prevent a number of useful features in epidemiological systems to be modelled. In particular, condition (5) only allows communication with two processes to be performed, and condition (1) forces the actions in the two to share the same rate. In the first models written in chapter 2, it is already obvious why this is a handicapping limitation: the source of infection (whether it is *InfEnv* or *Transmitter*) communicates with S, I and R. Even though the communication with I and R does not result in a change of state, it reduces the number of contacts with S and thus influences on overall rate of the action.

This section presents a modification of the Hillston Method. The resulting method generates a set of ODEs which are at least as good as the Hillston Method's ODEs in matching the average of stochastic simulations under the restrictions imposed by the Hillston Method. It also extends the scope of the Hillston Method by releasing condition (1) and modifying (5). The new set of limitations of the PEPA model for the Stirling Amendment are:

1. Actions may not be hidden.
2. Cooperation must include all common actions.
3. Components do not cooperate if they belong to the same subgroup.
4. Cooperation can only involve components belonging to exactly two different subgroups.

The last condition is a modification of condition (5) detailed in section 3.1.1. Whereas the condition as expressed by the Hillston Method restricted cooperations to involve only two component types

in two different subgroups, condition 4 as presented in the Stirling Amendment only restricts the component types to belong to two different subgroups. The number of component types involved in each subgroup is not restrained.

Two modifications have been made to the original algorithm. First, the activity matrix, initially presented in section 3.1.1 has been modified to account for the new required information. The entry and exit activities were represented with $+1$ and -1. In the Stirling Amendment, it is also necessary to record to which subgroup a component type belongs to, in order to know the component types it competes against in the cooperation. An entry and an exit activity are now described with $+1_S$ and -1_S respectively, with S representing the subgroup the component type belongs to. A third type of entry, $\pm 1_S$, has also been added. It records an action which is both an entry and an exit activity for the component type. This can occur in two cases. The first case corresponds to the situation where the component type C allows the action to be fired, and a different component type also allows it, with the target component type being C. This situation could not occur in the Hillston Method, as condition (5) did not allow two components within the same subgroup to fire the same action to be fired. The second case is a particular situation of the first case. It corresponds to the situation where the activity involving the action is actually a self-loop. While this situation could occur in the PEPA models accepted by the Hillston Method, the algorithm simply ignored the activity. In the Stirling Amendment however, while the self-loop does not add a term to the ODEs, their influence is taken into account when considering the dynamics of the other component types.

The second modification concerns the rate of the activity. The Hillston Method uses the local rate, assumed to be the same within a communication. In the Stirling Amendment, the rate is the overall apparent rate $r_\alpha(Sys)$ multiplied by *component contribution*. The component contribution corresponds to the proportion of demand for cooperation from agents of the other

subgroup actually met by agents of the considered subgroup. For example, if S, I and R compete over the action contact in an indirect transmission model, each time an agent from the environment subgroup is ready to perform the action, the probability that an agent from S performs the action is $S/(S+I+R)$. More generally, for a row n_i, if the matrix entry subscript is S_l (say) then the contribution of n_i to the activity is $n_i(t)/\sum_{k\in Ex_{S_l}(j)\cup Loop_{S_l}(j)} n_k(t)$. That is, the sum of all components tagged -1_{S_l} or $\pm 1_{S_l}$ in that column. This change is specified in the amended algorithm of Figure 3.4. As before there will be one ODE in the system for each row of the matrix.

Definition 9 (Activity Matrix) *For a model with N_A activities and N_D distinct local derivatives, the* activity matrix M_a *is an $N_D \times N_A$ matrix, and the entries are defined as follows:*

$$(n_i, a_j) = \begin{cases} +1_S & \text{if } a_j \text{ is an entry activity of } n_i, S = \mathcal{SG}(n_i) \\ -1_S & \text{if } a_j \text{ is an exit activity of } n_i, S = \mathcal{SG}(n_i) \\ \pm 1_S & \text{if } a_j \text{ is both an entry and an exit activity of } n_i, S = \mathcal{SG}(n_i) \\ 0 & \text{otherwise.} \end{cases}$$

Using the Stirling Amendment, the following ODEs are obtained when derived from the model of direct transmission presented in section 2.2:

//Form one ODE for each local derivative/state variable

`For` $i = 1...N_D$

//Find the activities involving this derivative

`For` $j = 1....N_A$

//Record the impact of each such activity

`If` $M_a(i,j) = +1_S$ *or* $\pm 1_S$

`Add`

$$+r_j(Sys) \times \frac{r \times n_i(t)}{\gamma(r_j(S))}$$

`to the equation of` n_i `for all` $n_z \xrightarrow{(j,r)} n_i$`.`

`If` $M_a(i,j) = -1_S$ *or* $\pm 1_S$

`Add`

$$-r_j(Sys) \times \frac{r \times n_i(t)}{\gamma(r_j(S))}$$

`to the equation of` n_i `for all` $n_i \xrightarrow{(j,r)} n_z$`.`

`with`

$$\gamma(t) = \begin{cases} 1 & \texttt{if t = 0} \\ t & \texttt{otherwise.} \end{cases}$$

Figure 3.4: Pseudo-code for generating the set of ODEs

$$
\begin{aligned}
dS/dt &= -\frac{\top.S}{\top.S+\top.I+\top.R} \times min(\top.S, cr.Transmitter) + li.R \\
dE/dt &= \frac{\top.S}{\top.S+\top.I+\top.R} \times min(\top.S, cr.Transmitter) \\
&\quad -min(ir.E, \top.Dormant) \\
dI/dt &= min(ir.E, \top.Dormant) - min(rr.I, \top.Transmitter) \\
dR/dt &= min(rr.I, \top.Transmitter) - li.R \\
dTransmitter/dt &= -min(rr.I, \top.Transmitter) + min(ir.E, \top.Dormant) \\
dDormant/dt &= min(rr.I, \top.Transmitter) - min(ir.E, \top.Dormant)
\end{aligned}
\tag{3.1}
$$

A few simplifications can be made to this system, in order to make it more readable. First of all, at all times, $N(Transmitter) = N(I)$, so $Transmitter$ can be replaced by I. This means that $min(rr.I, \top.Transmitter) = rr.I$. Also, $N(Dormant) = N(S) + N(E) + N(R) > N(E)$ as the number of sequential components in a particular component type is always positive. This means that $N(Dormant) = 0$ implies that $N(E) = 0$ and so, $min(ir.E, \top.Dormant) = ir.E$. Finally, $\frac{\top.S}{\top.S+\top.I+\top.R} \times min(\top.S, cr.Transmitter)$ can be simplified to $\frac{cr.S.I}{S+I+R}$. Indeed:

- if $N(S) = 0$, then $\frac{\top.S}{\top.S+\top.I+\top.R} \times min(\top.S, cr.Transmitter) = 0$ and $\frac{cr.S.I}{\top.S+\top.I+\top.R} = 0$.
- if $N(S) > 0$, then $min(\top.S, cr.Transmitter) = cr.I$, as $\top.S > cr.Transmitter$ (by definition of the apparent rate) and $N(Transmitter) = N(I)$ and $\top$ can be taken out in the top and the bottom of the fraction. We finally obtain $\frac{\top.S}{\top.S+\top.I+\top.R} \times min(\top.S, cr.Transmitter) = \frac{cr.S.I}{S+I+R}$

Once these simplifications have been made, the resulting set of ODEs becomes:

$$
\begin{aligned}
dS/dt &= -\frac{cr.S.I}{S+I+R} + li.R \\
dSC/dt &= \frac{cr.S.I}{S+I+R} - ir.E \\
dI/dt &= ir.E - rr.I \\
dR/dt &= rr.I - li.R
\end{aligned} \tag{3.2}
$$

Using the same reasoning to simplify them [1], the Stirling Amendment produces this set of ODEs for the indirect transmission model presented in section 2.1:

$$
\begin{aligned}
dS/dt &= -\frac{cr.S.InfEnv}{S+I+R} + li.R \\
dE/dt &= \frac{cr.S.InfEnv}{S+I+R} - ir.E \\
dI/dt &= ir.E - rr.I \\
dR/dt &= rr.I - li.R \\
dInfEnv/dt &= ier.I - dr.InfEnv \\
dEnv/dt &= -ier.I + dr.InfEnv
\end{aligned} \tag{3.3}
$$

The graphs of Figure 3.5 and Figure 3.6 compare the output of the ODEs and the average behaviour of the models presented in chapter 2. In Figure 3.5, the derived ODEs fit well to the average behaviour of the direct transmission model. For indirect transmission however, Figure 3.6 shows that the match between the ODEs and the average of the stochastic simulation is rather poorer. This difference can be noticed in several models. Usually, the outbreak has a bigger amplitude in the case of the ODEs. This is caused by the continuous nature of the ODEs. For example, in the case of an SIR model, the number of infectious agents is typically low at the start of

[1] It has also been assumed that the healthy environment is abundant enough never to be depleted. In other words, $\forall t.N(Env,t) > 0$. In this case, $min(ier.I, \top.Env) = ier.I$

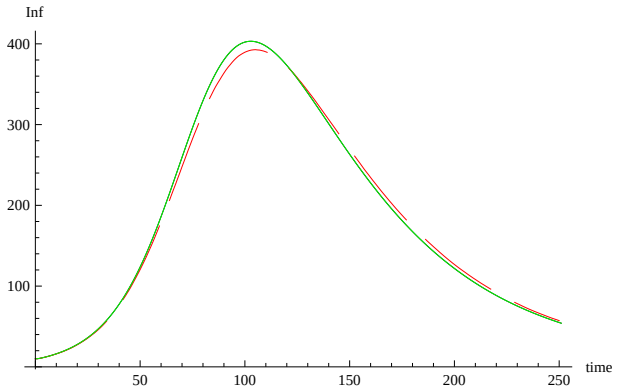

Figure 3.5: Direct Transmission: average of 1000 stochastic simulations (dashed line), Stirling Amendment ODE. Parameters as in Figure 2.5 of chapter 2. RMS = 7.

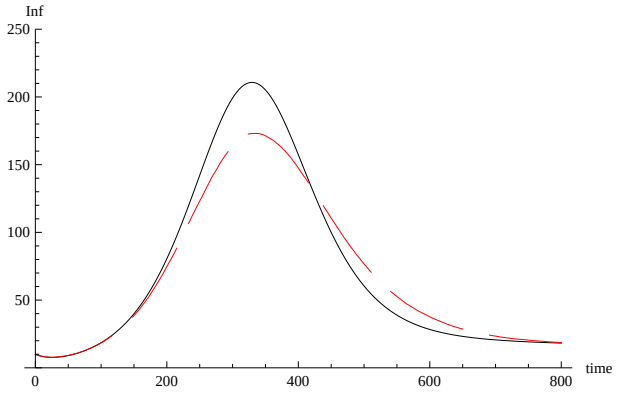

Figure 3.6: Indirect Transmission: average of 1000 stochastic simulations (dashed line), Stirling Amendment ODE (solid line). Parameters as in Figure 2.2 of chapter 2. RMS = 15.

the simulation. During the stochastic simulations, this leads to some runs witnessing the number of infectious individuals dropping to zero. This results in all the communications involving I being blocked, potentially leading to a completely different behaviour. In the indirect transmission model, if the number of I drops to zero too early, before the environment has been contaminated, it leads to the infection fading out and the number of I remaining zero for the rest of the run. However, if this does not happen, the outbreak will take place, with the number of I rising quickly. However, in the case of the ODEs, the number of I never drops to zero. Indeed, it always remains positive, allowing the communication to happen at all time. This has been confirmed in the work of Robertson [75]. In his work, Robertson performed a large amount of stochastic simulations of the prairie dog model presented in Figure 5.4 of section 5.5.1. He then removed all the runs where the infection died out in the first days of the simulation, and then compared the average of the remaining runs with the derived ODEs. He found that the ODEs had a much better match with the stochastic simulations without fade-out, compared to the entirety of the simulations.

3.1.3 Other existing methods

Since the paper by Benkirane et al. [7] was published, two notable algorithms have independently been published. The paper by Bradley et al. [13] takes an approach very similar to the one presented in the next section: based on the Hillston Method, it generalises the algorithm to be able to release some of the conditions, and applies it to a direct transmission SIR model. However, the resulting algorithm completely ignores self-loops. We will see on the first model presented in this paper is presented in Figure 3.7 how it can lead to an erroneous interpretation of the model.

Using the algorithm presented by Bradley et al. [13], the main infection term is:

$$\frac{dS}{dt} = -\beta I_s Net$$

$$
\begin{aligned}
S &\stackrel{def}{=} (infectS, \top).I \\
I &\stackrel{def}{=} (infectI, \beta).I + (infectS, \top).I + (patch, \gamma).R \\
R &\stackrel{def}{=} stop \\
Net &\stackrel{def}{=} (infectI, \top).Net' \\
Net' &\stackrel{def}{=} (infectS, \beta).Net + (fail, \delta).Net \\
S[1000] &\parallel I \underset{\{infectS, infectI\}}{\bowtie} Net[200]
\end{aligned}
$$

Figure 3.7: Model of transmission of a worm in a computer network, as presented in [13]

where

$$
I_s(t) = \begin{cases} 0 & \text{if } S(t)=0 \\ 1 & \text{otherwise.} \end{cases}
$$

However, the infection involves a competition between S, I and R. This means that when an individual infects another individual, she can potentially infect either a susceptible, infectious or recovered individual. Obviously, when the individual infected is infectious or recovered, the infection does not have any impact on the state of the individual. It does however have an impact on the overall system. The number of infections per unit of time only depends on the number of agents of infection (I in our example). If only susceptible individuals can be targeted, this means that the infectious individuals somehow manage to only contact them, whereas if everyone can be infected (as in the original model), the susceptible individuals only get a fraction of the infectious contacts.

Hayden et al. [41] also proposed an algorithm to derive ODEs from a PEPA model. A different approach is taken here. Instead of comparing the ODEs to the average of stochastic simulations, the CTMC is used as a benchmark to test the validity of the ODEs. The algorithm is more general than the Stirling Amendment presented in the next section. Indeed, the only remaining

restriction to the PEPA model is condition (4), which proscribes cooperation within a subgroup. However, even though the ODEs can be derived, some of the results, and in particular the explicit expression of the ODEs only apply to "split-free" models, which corresponds to models complying to condition (5), which prohibits any implicit choice in the model. This condition is very limiting when it comes to epidemiology, as components often compete over an action, like *contact* in the case of the model presented in Figure 2.1 in section 2.1.

Even though these algorithms can be used on a greater set of PEPA models than the Stirling Amendment, this algorithm will still be used in the subsequent sections. The algorithm from Bradley et al. [13] was clearly not ready to deal with some of the issues, and self-loops in particular. The ODEs expression from Hayden et al. [41] does however tackle the points necessary to model disease transmission models. However, it is geared toward computer systems, and the focus is not on the features that epidemiology requires. In the next section, a new algorithm will be presented, specifically designed to handle epidemiological systems.

3.2 The Hillston Method and the Stirling Amendment compared

The Stirling Amendment attempts to generalise the Hillston Method, in order to be able to apply it to PEPA models of epidemiological systems. The objective of this section is to formally show that the Stirling Amendment is indeed an extension of the Hillston Method, in order to benefit from the results Hillston obtained in [43]. This is achieved by proving that the two algorithms produce the same set of ODEs for PEPA models meeting the following conditions:

Conditions

1. Cooperating components must share the same local rate (no passive components).
2. Actions may not be hidden.
3. Cooperation must include all common actions.
4. Components of the same type do not cooperate.
5. There should not be implicit choice between independent but distinct components offering the same action for synchronisation.
6. No self-loops.

The first five conditions are the ones imposed by the Hillston Method. Condition 6 was not explicitly expressed in [43], but is rather implicit as self-loops are uncommon in computer systems. As shown in section 3.2.1, the output from the Hillston Method and the Stirling Amendment are different for models with self-loops. More details will be given in that section.

The objective of this section is to prove that the Hillston Method and the Stirling Amendment are equivalent under the conditions above. This is achieved by applying the five conditions to the algorithm and the Activity matrix of the Stirling Amendment, respectively presented in Figure 3.4 and in Definition 9 and showing that these are equivalent to the definitions of Hillston presented in section 3.1.1. The new expression of the algorithm presented in Figure 3.1.2 is presented in Figure 3.8, while the Activity matrix is reformulated below:

Activity matrix The activity matrix is now exactly identical to its definition in the Hillston Method, detailed in Definition 8. Indeed, the entries of the activity matrix for a model are defined

as follows.

$$(n_i, a_j) = \begin{cases} +1 & \text{if } a_j \text{ is an entry activity of } n_i \\ -1 & \text{if } a_j \text{ is an exit activity of } n_i \\ 0 & \text{otherwise.} \end{cases}$$

Two changes can be noticed in the activity matrix, compared to the one seen in section 3.1.2. First, the information about the side is not needed any more. Indeed, it was only needed in the "$r \times n_i(t)/r_j(S)$" part of the algorithm, which is shown in (A) to be equal to 1. Also, $\pm 1_S$ is not necessary anymore. It was only needed for the case where an action was both an entry and an exit activity for a component type. Condition (5) assures that there is no more implicit competition, hence $\forall a_j, \exists!(n_1, n_2)/\ n_1 \xrightarrow{(a_j, r)} n_2$. Also, condition (6) ensures that self-loops are not permitted either. The resulting definition corresponds exactly to the definition presented in section 3.1.1.

Concerning the algorithm itself, its formulation varies slightly when compared to the one presented in Figure 3.1. The set of exit components of activity j is not explicitly calculated in the algorithm presented in Figure 3.8. The definition however is exactly the same. The expression of the term added to the ODEs is presented differently. Its value is however the same, as shown by the proof (B). Overall, for all the models satisfying the set of conditions presented in this section, both algorithms produce the same set of ODEs.

Additional proofs

(A)Why is $r \times n_i(t)/\gamma(r_j(S)) = 1$

As S corresponds to a subgroup, condition (4) guarantees that the components within the subgroup do not cooperate. This means that if $S = n_1||n_2||...||n_i||...||n_l$, then,

$$r_j(S) = \sum_{k=1}^{l} r_j(n_k)\ .$$

//Form one ODE for each local derivative/state variable

`For` $i = 1...N_D$

//Find the activities involving this derivative

`For` $j = 1....N_A$

//Record the impact of each such activity

`If` $M_a(i,j) = +1$

Add

$$+r_j(Sys) \tag{A}$$

to the equation for each (n_z, n_i) / $n_z \xrightarrow{(j,r)} n_i$. However, there will one such transition due to condition (5).

If $M_a(i,j) = -1$

Add

$$-r_j(Sys) \tag{A}$$

to the equation for each (n_z, n_i) / $n_i \xrightarrow{(j,r)} n_z$. However, there will one such transition due to condition (5).

and

$$r_j(Sys) = r_j \times \min_{k \in Ex(j)}(n_k(t)) \tag{B}$$

Figure 3.8: Pseudo-code of the algorithm under Hillston Method's restrictions

Also, because of condition (5), we know that $\exists!(n_i, n_z)$ / $n_i \xrightarrow{(j,r)} n_z$. This means that $\forall k \in [1, l], k \neq i, r_j(n_k)$=0, which results in $r_j(S) = r \times n_i(t)$, and thus, eventually $r \times n_i(t)/\gamma(r_j(S)) = 1$.

(B)Value of $r_j(Sys)$

Notation

In this proof, we will need new notation:

- P is used to represent a single component.
- For all component P, $N(P)$ is the number of instances of the component P.

Definition 10 The set of possible components is $\mathcal{C}$.

Definition 11 The set of system equations is $\mathcal{S}$. It is defined by $\forall Q \in \mathcal{S}$, $Q ::= Q \underset{L}{\bowtie} Q|P$, *with* $P \in \mathcal{C}$.

Definition 12 The length of the system equation is the number of single components it contains. It will be noted $N(Sys)$.

For example $Sys = (P||P) \underset{L}{\bowtie} Q$ is of length 3, assuming P and Q are sequential components.

Definition 13 The set of exit components of an activity j is defined as $Ex(j) = \{P \in \mathcal{C}, \exists P_x \in \mathcal{C}, \exists r \in \mathbb{R}^+, P \xrightarrow{(j,r)} P_x\}$.

Lemma 3.2.1 $\forall Q_1, Q_2 \in \mathcal{S},\ \forall L,\ N(Q_1 \underset{L}{\bowtie} Q_2) = N(Q_1) + N(Q_2)$.

We want to show that, for every system equation of the form $Sys ::= Q \underset{L}{\bowtie} Q|P$, with $N(Sys) = n$ and for every action j, we have

$\exists i_0, i_1, ..., i_k \in \mathbb{N}, \exists r \in \mathbb{R}$

$$r_j(Sys) = \begin{cases} r \times min(N(P_{i_0}), ..., N(P_{i_k})) \text{ where } \{P_x,\ x \in \{i_0, i_1, ..., i_k\}\} = Ex(j) \\ 0 \text{ if } \forall P, Q \in \mathcal{S}, \forall r \in \mathbb{R}^+,\ P \overset{(j,r)}{\not\longrightarrow} Q \end{cases}$$

Inductive reasoning will be used to prove this statement.

Proof 3.2.2 ***Basis****: Show that the statement holds for a system equation of length 1.*

This means that $\exists P \in \mathcal{C}$, $Sys = P$. *Then, two cases appear:*

First case: $\exists Q \in \mathcal{C}$, $\exists r \in \mathbb{R}\ P \overset{(j,r)}{\longrightarrow} Q$

In this case, $r_j(Sys) = r\ = r \times N(P)$ *as* $N(P) = 1$, *with* $P = Ex(j)$ *as it is the only component in the system such that* $\exists Q \in \mathcal{C}$, $\exists r \in \mathbb{R}\ P \overset{(j,r)}{\longrightarrow} Q$.

Second case: $\forall Q \in \mathcal{C}$, $\forall r \in \mathbb{R}\ P \overset{(j,r)}{\not\longrightarrow} Q$

In this case, $r_j(Sys) = 0$.

So the statement holds for a system equation of length 1.

Inductive step*: Show that if* $\forall k \in [\![1, n]\!]$, *the property holds for a system of length* k, *then it also holds for a system of length* $n + 1$.

Let's call R *a system equation of length* $N(R) = (n + 1)$. *As* $n \geq 2$, *we know that* $\exists Q_1,\ Q_2,\ \exists L\ R = Q_1 \underset{L}{\bowtie} Q_2$. *This means that* $r_j(R) = r_j(Q_1 \underset{L}{\bowtie} Q_2)$. *The length of a*

system verifies the Lemma 3.2.1, so $N(R) = N(Q_1) + N(Q_2)$. *The length of* Q_1 *and the length of* Q_2 *are both greater or equal to 1, which means that their respective lengths are also smaller or equal to* n.

Here again, this can be divided into two cases:

First case: $j \in L$

In this case, according to the rules for the calculation of the apparent rate, we know that $r_j(R) = min(r_j(Q_1), r_j(Q_2))$. *As both* $N(Q_1)$ *and* $N(Q_2)$ *are smaller or equal to* n, *we can use the induction hypothesis on* Q_1 *and* Q_2*:*

$\exists P_{1_0}, P_{1_1}, ..., P_{1_{Q_1}}, P_{2_0}, P_{2_1}, ..., P_{2_{Q_2}} \in \mathcal{C}$, *with* $\{P_x, x \in \{1_0, ..., 1_{Q_1}, 2_0, ..., 2_{Q_2}\}\} = Ex(j), \exists r_1, r_2 \in \mathbb{R}$,

$$\begin{aligned} r_j(R) &= min(r_j(Q_1), r_j(Q_2)) \\ &= min(r_1 \times min(N(P_{1_0}), ..., N(P_{1_{Q_1}})), r_2 \times min(N(P_{2_0}), ..., N(P_{2_{Q_2}}))) \\ &= r_1 \times min(min(N(P_{1_0}), ..., N(P_{1_{Q_1}})), min(N(P_{2_0}), ..., N(P_{2_{Q_2}}))) \qquad \textit{cond. (1)} \\ r_j(R) &= r_1 \times min(N(P_{1_0}), ..., N(P_{1_{Q_1}}), N(P_{2_0}), ..., N(P_{2_{Q_2}})) \textit{ because min is associative} \end{aligned}$$

In this case, the property holds for a system of length $n + 1$.

Second case: $j \notin L$

Here, we have $r_j(R) = r_j(Q_1) + r_j(Q_2)$. *However, condition (3) ensures that if* Q_1 *and* Q_2 *had common actions, they would have to cooperate on them. So they do not share any common action. This means that:*

$$r_j(R) = \begin{cases} r_j(Q_1) & \textit{if } \exists P_y \in Q_1, \exists P_x \; P_y \xrightarrow{(j,r)} P_x \\ r_j(Q_2) & \textit{if } \exists P_y \in Q_2, \exists P_x \; P_y \xrightarrow{(j,r)} P_x \\ 0 & \textit{otherwise} \end{cases}$$

If $r_j(R) = 0$, then the statement holds. If $\exists i \in \{1,2\}\ r_j(R) = r_j(Q_i)$, then as $N(Q_i) \leq n$, we can apply the induction hypothesis to Q_i and so:

$\exists P_{i_0}, P_{i_1}, ..., P_{i_{Q_i}} \in \mathcal{C}$, *with* $\{P_x, x \in \{1_0, ..., 1_{Q_1}\}\} = Ex(j), \exists r \in \mathbb{R}$, $r_j(R) = r \times min(N(P_{i_0}), ..., N(P_{i_{Q_i}}))$

In this case again, the property holds for a system of length $n+1$.

Thereby, we have shown that in all cases, the property holds for a system of length $n+1$.

□

Since both the basis and the inductive step have been proved, it has now been proved by mathematical induction that for all system equations:

$\exists i_0, i_1, ..., i_k \in \mathbb{N}, \exists r \in \mathbb{R}$

$$r_j(Sys) = \begin{cases} r \times min(N(P_{i_0}), ..., N(P_{i_k})) \text{ where } \{P_x,\ x \in \{i_0, ..., i_k\}\} = Ex(j) \\ 0 \text{ if } \forall P, Q \in \mathcal{S}, \forall r \in \mathbb{R}^+,\ P \overset{(j,r)}{\not\rightarrow} Q \end{cases}$$

3.2.1 Condition (6): an illustrative example

The Stirling Amendment has a different behaviour compared to the Hillston Method when it comes to self-loops. In order to compare the output of the sets of ODEs generated by both algorithms, this section shows an example of a model which complies with the first five conditions presented in the previous section, but does not satisfy condition (6).

The model here is a very simple SI model with indirect transmission, with a fixed amount of infected environment. *InfEnv* only tries to infect S individuals, but always stays in the same state. This example shows how the Hillston Method and our algorithm have different outputs when a self-loop is involved in a cooperation.

$$
\begin{aligned}
S &\stackrel{def}{=} (contact, cr).I \\
I &\stackrel{def}{=} (recover, rr).S \\
InfEnv &\stackrel{def}{=} (contact, cr).InfEnv \\
S[990] &\parallel I[10] \underset{contact}{\bowtie} InfEnv[50]
\end{aligned}
$$

Using the algorithm presented in [43], the ODEs obtained from this model are:

$$
\begin{aligned}
\frac{dS}{dt} &= -cr \times S + rr \times I \ , \\
\frac{dI}{dt} &= -\frac{dS}{dt} \ , \\
\frac{dInfEnv}{dt} &= 0 \ ,
\end{aligned} \tag{3.4}
$$

When the Stirling Amendment is used, the ODEs are:

$$
\begin{aligned}
\frac{dS}{dt} &= -cr \times min(S, InfEnv) + rr \times I \ , \\
\frac{dI}{dt} &= -\frac{dS}{dt} \ , \\
\frac{dInfEnv}{dt} &= 0 \ ,
\end{aligned} \tag{3.5}
$$

The difference between the two sets of ODEs lies in how the *contact* activity has been dealt with. Indeed, in the case of Hillston Method, as *InfEnv* does a self-loop, it is completely ignored. The same results would be obtained for a model without *InfEnv*. However, the way the model was written implies that without it, no cooperation is possible. In the Stirling Amendment's set of ODEs, the case of the *contact* activity is dealt differently. Indeed, the S can only be contacted

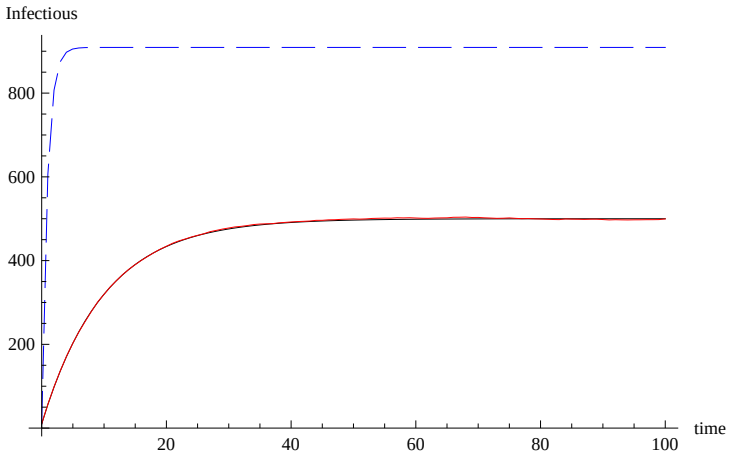

Figure 3.9: Graph of the model illustrating the difference in handling self-loops. Average of 200 simulations and the Stirling Amendment ODEs (superposed, solid line), the Hillston Method ODEs (dashed line). Parameter values: $cr = 1, rr = 0.1$.

if there is some *InfEnv* available. This is why the rate at which S becomes I depends on the minimum between the number of S and *InfEnv*.

Figure 3.9 further illustrates this point. While the ODEs obtained with the Stirling Amendment provide a very good match to the average of 200 simulations, the ODEs obtained with the Hillston Method exhibit a very different behaviour, due to the number of S decreasing without *InfEnv* limiting it.

3.3 The Dining Philosophers Problem

The motivation for the Stirling Amendment came from examples in epidemiology; however, it has a wider application. In this section, we present an example from computer science to illustrate that, and to demonstrate the type of further analysis possible with the ODEs.

3.3.1 Presentation of the problem

$$
\begin{aligned}
Pthink &\stackrel{def}{=} (hungry, h).P0 \\
P0 &\stackrel{def}{=} (acquire, \top).P1 \\
P1 &\stackrel{def}{=} (acquire, \top).Peat + (release, n).P0 \\
Peat &\stackrel{def}{=} (eating, e).PR2 \\
PR2 &\stackrel{def}{=} (release, r).PR1 \\
PR1 &\stackrel{def}{=} (release, r).Pthink \\
Fork\_A &\stackrel{def}{=} (acquire, a).Fork\_U \\
Fork\_U &\stackrel{def}{=} (release, \top).Fork\_A \\
Pthink[p] &\underset{\{acquire, release\}}{\bowtie} Fork\_A[f]
\end{aligned}
$$

Figure 3.10: PEPA model for the Philosophers' problem.

The dining philosophers problem is a very famous problem in the history of computer science. It was first presented by Hoare [46], and gives a simple and appealling presentation of a number key features in concurrency, such as resource allocation, deadlock, livelock, starvation and fairness. Moreover, solutions to these problems can be modelled in the same framework. In a generalisation of the problem, we have p philosophers around a table, with f forks and a bowl of spaghetti in the middle. Each philosopher can spend his time thinking or eating. When a philosopher is hungry, he needs two forks to eat his spaghetti. The features mentioned above arise when the number of forks is smaller then the number of philosophers.

The PEPA model presented in Fig. 3.10 captures the behaviour of the philosophers in a variant of the problem, as the original problem is not very suited to PEPA. In the most commonly used version of the problem, each philosopher has two forks, one on his left and one on his right, that she shares with her neighbour. To represent it in PEPA, the model would need to explicitly represent

each philosopher and its associated forks, as there is no way to distinguish between agents of the same state otherwise.

In order to have a more PEPA-friendly problem, we have p philosophers are collaborating with f forks, with the forks available for everyone to use. At the beginning, all the philosophers are thinking, and all the forks are available. The forks have no special placement: effectively, we have a pile of forks in the middle of the table from which any philosopher can choose. When a philosopher wishes to eat, he asks for a fork. Once he manages to get one, he asks for a second fork. Then, he spends some time eating, releases the first fork, and the second fork, and finally thinks again. In this simple situation, deadlock may arise when $p \geq f$: f philosophers have one fork and cannot eat without a second fork, but there are no free forks available. Therefore, we allow hungry philosophers with one fork (*P1*) to release it and return to the state *P0*. In other words, he just stops waiting for the second fork, releases the one he has in his hand, but as he is still hungry, comes back to the state where he is waiting for his first fork. By allowing the hungry philosophers to release their fork, we provide the opportunity for another to obtain two forks.

We have then derived the ODEs from the PEPA model, using the Stirling Amendment:

$$
\begin{aligned}
dPthink(t)/dt &= -h.Pthink(t) + \frac{r.PR1(t)}{R(t)} min(R(t), \top.Fork\_U(t)) \ , \\
dP0(t)/dt &= h.Pthink(t) + \frac{n.P1(t)}{R(t)} min(R(t), \top.Fork\_U(t)) \\
&\quad - \frac{P0(t)}{P(t)} min(\top.P(t), a.Fork\_A(t)) \ , \\
dP1(t)/dt &= -\frac{n.P0(t)}{R(t)} min(R(t), \top.Fork\_U(t)) \\
&\quad + \frac{P0(t) - P1(t)}{P(t)} min(\top.P(t), a.Fork\_A(t)) \ , \\
dPeat(t)/dt &= -e.Peat(t) + \frac{P1(t)}{P(t)} min(\top.P(t), a.Fork\_A(t)) \ , \\
dPR2(t)/dt &= e.Peat(t) - \frac{r.PR2(t)}{R(t)} min(R(t), \top.Fork\_U(t)) \ ,
\end{aligned}
$$

$$
\begin{aligned}
dPR1(t)/dt &= \frac{r.(PR2(t) - PR1(t))}{R(t)} min(R(t), \top.Fork\_U(t)) \,, \\
dFork\_A(t)/dt &= min(R(t), \top.Fork\_U(t)) - min(\top.P(t), a.Fork\_A(t)) \,, \\
dFork\_U(t)/dt &= min(\top.P(t), a.Fork\_A(t)) - min(R(t), \top.Fork\_U(t)) \,, \qquad (3.6)
\end{aligned}
$$

where $R(t) = r(PR2(t) + PR1(t)) + n.P1(t)$ and $P(t) = P0(t) + P1(t)$. Simulations show that both the simulations and the ODEs quickly reach an equilibrium, which will be exploited in the next section. Since this model has a steady state, the Markov analysis of the PEPA tool can be used. The steady state value of $Peat$ can be computed easily for twelve philosophers and nine forks [2].

3.3.2 Optimisation of the service

What added benefit can be gained from having ODEs for the philosophers model? Recall that n is the rate at which a hungry philosopher releases the single fork already acquired. Given values for h, e, a, r, and of course p and f, what would be the optimal value of n, call it n_{max}, in order to serve as many philosophers as possible?

An analysis of the Markov chain has two drawbacks. First, the system becomes quickly unmanageable as its size increases. Indeed, the size of the Markov chain increases exponentially with the number of processes, and the limit of the computer power is quickly reached. Second, the analysis is dependent on the values of the parameters, and must be repeated in its entirety when parameter values change. Finally, it is only possible to have an approximate value for n_{max} by trying several different values and finding which one gives the biggest $Peat$ at steady state.

Simulation may also be used to answer this question; however, as for the Markov Chain method, it is only possible to get an approximate value for n_{max} using a try and fail method. This approach

[2] There are 1837 states in the model. The calculation takes only twelve seconds on a computer with a P4 2.80GHz processor and 1Gb of RAM, but larger values of p and f are not possible without more memory.

is time consuming as tens of sets of a few hundred simulations are needed, which results in thousands of simulations [3]. As above, if one or more of the other parameters are modified, all the simulations must be done again, which makes this approach both time consuming and approximate.

ODEs provide a better approach to solving this problem. The solution will be exact, and n_{max} can be calculated for new parameter values in a few seconds. Once the system is solved, the solution does not change.

To answer the question and find n_{max}, the ODEs (3.6) are considered in the steady state, when all derivatives are equal to zero. This yields the following equations (dropping the (t) parameter):

$$0 = -h.Pthink + r.PR1 \tag{3.7}$$

$$0 = h.Pthink + n.P1 - \frac{P0}{P0+P1}a.Fork\_A \tag{3.8}$$

$$0 = -n.P0 + \frac{P0-P1}{P0+P1}a.Fork\_A \tag{3.9}$$

$$0 = -e.Peat + \frac{P1}{P0+P1}a.Fork\_A \tag{3.10}$$

$$0 = e.Peat - r.PR2 \tag{3.11}$$

$$0 = r.(PR2 - PR1) \tag{3.12}$$

$$0 = r.(PR2 + PR1) + n.P1 - a.Fork\_A \tag{3.13}$$

$$0 = a.Fork\_A - r.(PR2 + PR1) - n.P1 \tag{3.14}$$

$$p = Peat + P0 + P1 + PR2 + PR1 + Pthink \tag{3.15}$$

$$f = Fork\_U + Fork\_A \tag{3.16}$$

$$Fork\_U = P1 + 2.Peat + 2.PR2 + PR1 \tag{3.17}$$

[3] To obtain a smooth mean, around 10,000 simulations are required for each set of parameter values. To give an idea, 10,000 simulations takes about 41min on a computer with a P4 2.80GHz processor and 1Gb of RAM.

Equations (3.7) to (3.14) come directly from the ODEs. The *min* terms of (3.6) have been simplified using the knowledge that *P0*, *P1* and *Fork_U* are strictly positive. Equations (3.15) and (3.16) use the fact that the number of philosophers and the number of forks is constant over time. Finally, the last equation states that the number of forks used, i.e. those unavailable for others, is equal to the sum of the number of philosophers with at least a fork, multiplied by the number of forks they have.

It seems here that there are more equations than unknowns. Actually, some of the equations can be derived from others. Equations (3.13) and (3.14) are identical. Equation (3.9) can be derived by adding (3.8) and (3.10) (and using (3.7), (3.11) and (3.12) to prove that $h.Pthink = e.Peat$). Finally, (3.14) can be obtained by subtracting (3.8) from (3.10) (and using (3.7), (3.11), and (3.12) to prove that $2.h.Pthink = 2.e.Peat = r(PR2 + PR1)$). So we eventually end up with eight equations with eight unknowns. This system has been solved with the help of Mathematica, in order to get a function for $Peat(n)$ in terms of the rates, p and f which allows us to solve the question orignally posed. For example, by setting the parameters to $h = e = 1$ and $r = a = 10$, we get:

$$Peat(n) = \frac{50(n^2 - n(-380 + \sqrt{25 + 245n + n^2}) - 10(5 + \sqrt{25 + 245n + n^2}))}{-1400 + 3095n + 11n^2} .$$

Once again using Mathematica, we can obtain n_{max} such that $Peat(n_{max})$ is maximal:

$$n_{max} = \frac{10}{13}(17 - \sqrt{94}) = 5.62 \text{ with } Peat(n_{max}) = 4.67 .$$

In other words, with $h = e = 1$, $r = a = 10$, twenty philosophers and fifteen forks, the value for n to ensure a maximum of philosophers eating is $n = 5.62$, and this value allows 4.67 philosophers

to eat on average. Although this example is simplistic, the principle can be translated to, for example, a resource allocation problem in an operating system. In this case, the value of n_{max} would tell us how to set the back off to allow maximum throughput of clients.

3.4 PEPAdum: automatically deriving the ODEs from a PEPA model

The PEPA Eclipse plugin [85] has the option to derive the ODEs of a PEPA model and gives the graph that those ODEs generate. However, the algorithms the program uses to generate the ODEs only gives a numerical approximations of the ODEs, and more importantly, the ODEs are not explicitly given, which prevents any theoretical analysis.

To overcome these issues, software, PEPAdum, has been developed in Java (more details in section 3.4.2). The objective here is to facilitate the analysis of the model by giving as much information to the modeller about the system. PEPAdum allows both the average of stochastic simulations, based on the PEPA Eclipse code, and the derivation of ODEs, using the Stirling Amendment, to be performed. It is also possible to obtain the standard deviation of the simulations, in order to get more insight on the variability of the model. Finally, the average of the simulations and the ODEs can be compared using the Root Mean Square (RMS) value. The output is written in a Mathematica [94] file, which allows any further analysis to be performed and in particular in the generation of graphs.

3.4.1 PEPAdum description

PEPAdum itself is fairly intuitive to use and tries to address the issues of the PEPA Eclipse plugin, which is tailored for computer systems models. Once the file itself has been chosen in the window

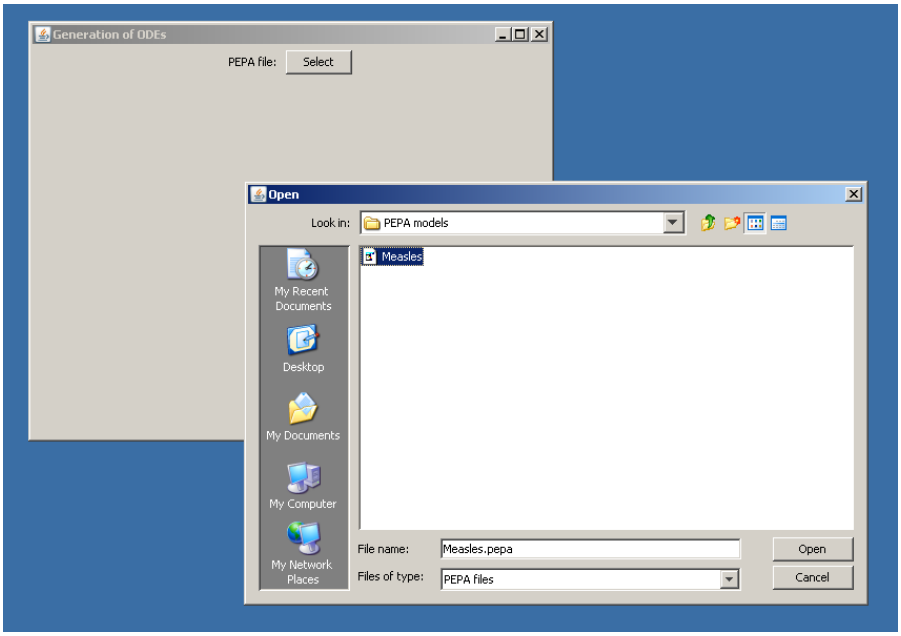

Figure 3.11: PEPAdum main page

shown in Figure 3.11, the user can choose in the next window shown in Figure 3.12 :

- the stop time: the time at which the simulations stop. The experiments always start at $t = 0$.
- the number of experiments: the number of experiments to be averaged. This number is ignored in the case of ODE derivation.
- the number of data points: this corresponds to the number of points at which the number of agents in each of the requested component type's state is recorded. This number is also ignored in the case of ODE derivation.
- the recorded component types: this is actually done in a different way compared to the PEPA Eclipse plugin. In the plugin, several occurrence of each component might appear. If the initial system equation is $S[s]||I[i]$, and one susceptible individual becomes infectious, the system should become $S[s-1]||I[i+1]$. The PEPA Eclipse plugin however differentiates the

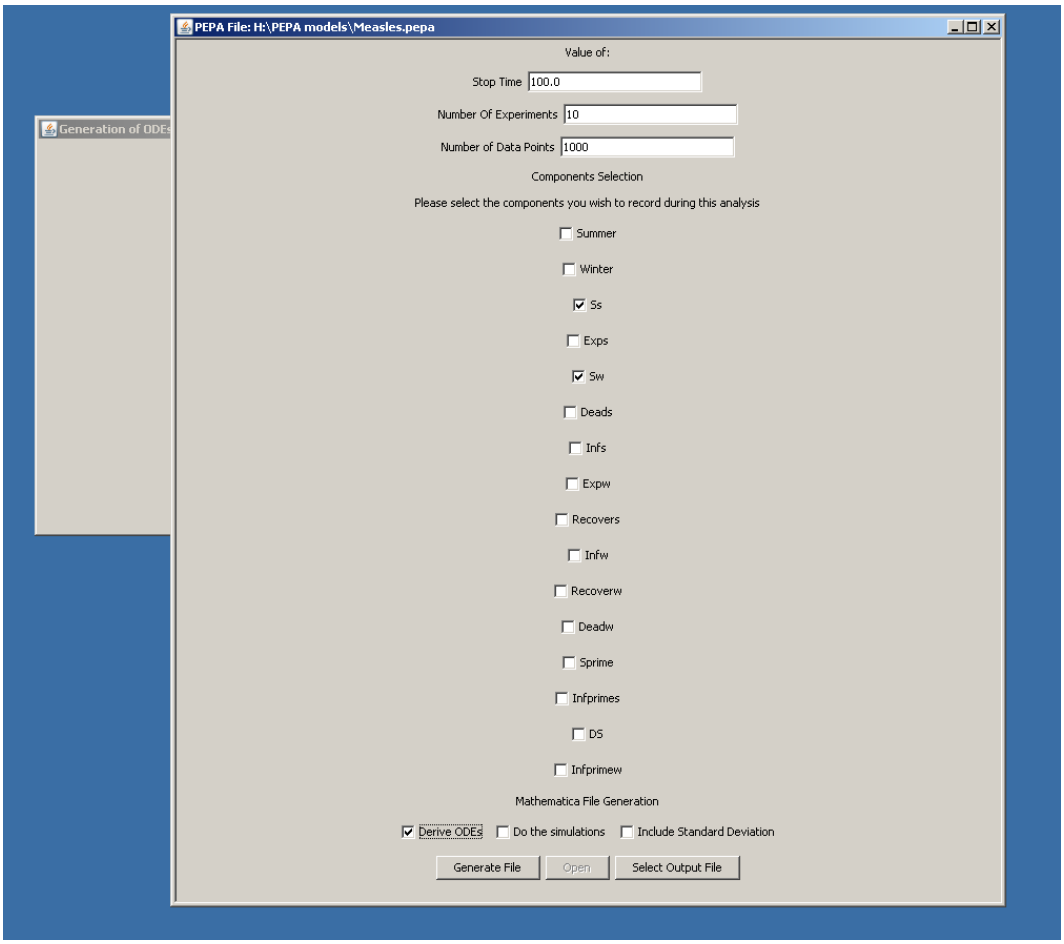

Figure 3.12: PEPAdum available options

between agents in the same state but with a different initial state. In this case, we obtain $S[s-1]||I\_1[1]||I\_2[i]$. In most cases, and in particular in epidemiology, this information is not required as the only difference between the two is the initial state the agents were in. In PEPAdum, this differentiation is not made, and we obtain $S[s-1]||I[i+1]$ as expected.

- Derive ODEs: the user can choose whether she wants the ODEs to be derived. In the resulting Mathematica file, the actual ODEs are given, as well as the code to plot the graph of each component type.

- Do the simulation: the user can choose whether or not she wants the simulations to be performed, using the stop time, number of experiments, number of data points, recorded component types given earlier. The resulting Mathematica file includes the actual data points and the time it took for the experiments to run (ignoring all the processing happening before and after them). The code is given to plot the graph of each selected component type.

- Include Standard Deviation: If the simulations are to be run, the standard deviation can be included in the data.

- Generate File: generates the Mathematica file including all the selected features. The default file name is the same as the PEPA file used as input, with the ".nb" extension.

- Open: this option is only available after the file has been generated, and opens the Mathematica file with the default software on the machine.

- Select Output File: allows the user to choose the name of the generated file.

When both the simulations and the ODEs are requested, additional information is added in the final Mathematica file. For each component type, the instructions for a graph comparing the ODEs and the stochastic simulations (and the standard deviation if requested) as well as for the

Root Mean Square (RMS) value of the difference between the average of the simulations and the ODEs, are added. The objective here is to allow the modeller to have access to all the tools with minimal knowledge of Mathematica.

3.4.2 PEPAdum architecture

The PEPA plugin does actually perform ODE derivation and analysis. A few ODES solvers are available to choose from (from the Runge-Kutta family of solvers and adaptive step-size 5th-order Dormand Prince ODE solver), but they only provide a numerical approximation of the solution of the system, but cannot explicitly provide the expression of the equations. For these reasons, it was decided to directly parse the PEPA model, using JavaCC [49], with a Java-based underlying platform. Basically, once the file has been chosen, it is parsed immediately, with all the relevant information retrieved. Then, once the options described in section 3.4.1 are given by the user, the information required is fed to the function in the PEPA Eclipse plugin that runs stochastic simulations and the ODEs are generated when the option has been chosen. Because of the possibility to include the standard deviation, the simulations are requested one by one. Indeed, the PEPA Eclipse plugin only gives the values for the average of the number of simulations requested, while the values of all simulations are needed for the standard deviation.

Some basic error handling has also been included, either reporting the Java error messages, some PEPA errors (which are usually quite general and unhelpful), or some restriction linked to Mathematica (for example, I and E are not allowed, as Mathematica interprets them as the mathematical number).

Once the simulation results and/or the ODEs are obtained, the output is written in a Mathematica 5 (and usually Mathematica 6 and 7) compatible file.

3.4.3 Limitations

PEPAdum has been developed to deal with the models accepted by the Stirling Amendment, both for the stochastic simulations and the derivation of the ODEs. In particular, the set of limitations presented in section 3.1.2 are still relevant for PEPAdum. Also, some models require a series of actions to be performed in order for the state to change. In the model presented in Figure 6.6, the action *go_winter* must be performed p times before an agent in *Summer* can go in the state *Winter*. This feature is not allowed by PEPAdum, both for stochastic simulations and the derivation of the ODEs.

3.5 Summary

Being able to derive ODEs from a PEPA model is an asset to epidemiological systems study. Even though it requires the different populations in the model to be large, it brings scalability to process algebra, as well as a long experience of being used as a tool in epidemiology. The existing Hillston Method was not adapted for epidemiological systems, so an extension, the Stirling Amendment, was developed to deal with issues preventing the ODEs of models based on those systems to be derived. We have shown that the Stirling Amendment deals with a superset of the models dealt with by Hillston Method. In addition to this, the models including self-loops did not match the simulations in the case of the Hillston Method. The way self-loops are represented have been changed in the Stirling Amendment, and the match improved significantly. Finally, it has been proved that for all the models accepted by the Hillston Method that do not have self-loops, the resulting ODEs are identical, whichever algorithm is chosen.

The example of the Dining Philosophers has then shown that the Stirling Amendment can be used to find the mean behaviour of PEPA models not only of biologically inspired problems, but of

computer science inspired problems as well. The ODEs derived from the model correctly represent the mean behaviour of the model and have also shown to facilitate quick calculation of properties not easily or accurately addressed using only the Markov chain or stochastic simulations. Finally, a piece of software, PEPAdum, has then been presented. The aim of PEPAdum is to provide the modeller tools to derive ODEs using the Stirling Amendment, and then be able to compare the result to stochastic simulations.

Chapter 4

Modelling disease transmission

As mentioned in [70, 57], disease transmission usually exhibits one of two different transmission terms: frequency and density dependent transmission. Both types of transmission are essential in epidemiology, as they commonly appear in different types of diseases. Frequency and density dependent transmission is also used for other features, such as births and deaths.

It has been shown by Norman et al. [70] that frequency dependent transmission terms arise naturally from process algebra models. This remains true in PEPA, as seen in the two examples presented in chapter 2. Even then however, the direct transmission model, where a cooperation within a subgroup was needed, already showed the limitations in PEPA's expressivity. The objective of this chapter is to generalise the methods presented in chapter 2 for any kind of frequency or density dependent cooperation.

In this chapter, the mirror subgroup, briefly mentioned in section 2.2, will be described in details. Frequency dependent transmission will then be studied, in order to give a general method to express it. The case of density dependent transmission, which cannot be directly expressed in PEPA, will then be examined.

4.1 The mirror subgroup

Before the frequency and density dependent terms are presented, an important feature, the mirror subgroup, is defined, as it will be extensively used in this chapter and the next. The notion of the mirror subgroup will be formally defined, a method to generate it will be described, and an example will show how it can be used, as well as how to simplify it.

The mirror subgroup has been briefly mentioned in section 2.2, where the *Transmitter* and the *Dormant* component types were mirroring the number of agents in the *Inf* component type and the S, E and R component types respectively. This notion can be further formalised:

Definition 14 (Mirror subgroup) *A subgroup S' with component types $\{C'_i\}_{i\in I}$ is a mirror subgroup of a subgroup S with component types $\{C_j\}_{j\in J}$ if and only if there exist two partitions with the same number of elements $\chi'=\{C'[i]\}_{i\in I}$ and $\chi=\{C[j]\}_{j\in J}$ such that all reachable states verify:*

$$N(C'[i_0])=N(C[j_0])$$

In other words, this means that a group of components in the mirror subgroup will always have the same number of agents as the group of components it represents in the original subgroup.

A method to construct the mirror subgroup of a subgroup can be described in two steps. This method is inspired from epidemiological systems, but can be applied to any models. In the first step, every component type C of the original subgroup needs to be replicated as a component C'. The required cooperations can then be added in both subgroups. The next step consists in ensuring that the number of agents in the original component and its mirror component is identical at all times. In order to achieve this, for all the activities in the form $(\alpha, r).C_{target}$ of a component C in the original subgroup, six cases arise:

- the action α is a self-loop and not involved in a communication: the model does not need to be modified.

- the action α is a self-loop and involved in a communication with a component type not in the mirror subgroup: the model does not need to be modified.

- the action α is a self-loop and involved in a communication with a component type C'_0 belonging to the mirror subgroup: in this case, the action is not a self-loop in the component type in the mirror subgroup[1]. If the activity is $(\alpha, r').C'_1$, it is necessary to update the original subgroup to reflect this change in C_0. To do so, the activity is changed to $(\alpha, r').C'_{0_{temp}}$ and the component type $C'_{0_{temp}}$ is added, allowing the activity $(\alpha', big).C'_1$, with big a rate high relatively to the other rates of the model. The component C_0 also allows the activity $(\alpha', \top).C_1$, to update simultaneously the number of agents in both the original and the mirror subgroup.

- the action α is not a self-loop and is not involved in a communication: in this case, C' will allow the action α with a passive rate and the target component will be C'_{target}, and both the initial and replicated component will communicate over α.

- the action α is not a self-loop and is involved in a communication with a component type which is not in the mirror subgroup: this time, it is not possible for the replicated component to communicate over α with the component C. In order to make sure the number of agents in each component remains the same, a temporary state will be added to the original subgroup. The original activity is modified to $(\alpha, r).C_{temp}$ and a component type C_{temp} is added to the subgroup, which allows one activity $(\alpha', big).C_{target}$ with big a rate relatively bigger than the other rates used in the model. On the other hand, in the component C' in the mirror

[1] If the action is a self-loop in the mirror subgroup as well, it has no impact on the behaviour of the model and can be removed entirely.

subgroup, the activity $(\alpha', \top).C'_{target}$ is added.

- the action α is not a self-loop and is involved in a communication with a component type C'_0 belonging to the mirror subgroup: this case simply requires both C' and C_0 to be updated using the method described in the previous point, creating the two component types C_{temp} and $C'_{0_{temp}}$.

To illustrate this method, if we consider the following PEPA model:

Original subgroup

$$C_1 \stackrel{def}{=} (\alpha, \top).C_1 + (\beta, r_1).C_2$$

$$C_2 \stackrel{def}{=} (\zeta, r_2).C_3$$

$$C_3 \stackrel{def}{=} (\delta, r_3).C_1$$

Rest of the model

$$R_1 \stackrel{def}{=} (\beta, rate_1).R_2$$

$$R_2 \stackrel{def}{=} (\alpha, rate_2).R_1$$

$$C_1[c_1] \parallel C_2[c_2] \parallel C_3[c_3] \underset{\{\alpha,\beta\}}{\bowtie} R_1[r_1] \parallel R_2[r_2]$$

Now the action γ on which C_1 and C_3 are to cooperate over is to be added to the model. The obvious way to achieve this is to directly add γ to the expression of C_1 and C_3:

Original subgroup

$$C_1 \stackrel{def}{=} (\alpha, \top).C_1 + (\beta, r_1).C_2 + (\gamma, r'_\gamma).C_2$$

$$C_2 \stackrel{def}{=} (\zeta, r_2).C_3$$

$$C_3 \stackrel{def}{=} (\delta, r_3).C_1 + (\gamma, r_\gamma).C_1$$

Rest of the model

$$R_1 \stackrel{def}{=} (\beta, rate_1).R_2$$

$$R_2 \stackrel{def}{=} (\alpha, rate_2).R_1$$

$$(C_1[c_1] \underset{\gamma}{\bowtie} C_2[c_2] \underset{\gamma}{\bowtie} C_3[c_3]) \underset{\{\alpha,\beta\}}{\bowtie} (R_1[r_1] \parallel R_2[r_2]\ )$$

This, unfortunately, does not result in the expected behaviour. In this case, the action γ is fired only if every agent in the $\{C_1, C_2, C_3\}$ subgroup is involved in it.

In order to have only one agent in C_1 and one in C_3 cooperating over the action γ as required, a mirror subgroup is to be added to the first subgroup. So first, the component types C_1', C_2' and C_3' are added to the model, as well as the action γ.

Original subgroup

$$C_1 \stackrel{def}{=} (\alpha, \top).C_1 + (\beta, r_1).C_2$$

$$C_2 \stackrel{def}{=} (\zeta, r_2).C_3$$

$$C_3 \stackrel{def}{=} (\delta, r_3).C_1 + (\gamma, r_\gamma).C_1$$

Replicated subgroup

$$C_1' \stackrel{def}{=} (\gamma, r_\gamma').C_2'$$

$$C_2' \stackrel{def}{=}$$

$$C_3' \stackrel{def}{=}$$

Rest of the model

$$R_1 \stackrel{def}{=} (\beta, rate_1).R_2$$

$$R_2 \stackrel{def}{=} (\alpha, rate_2).R_1$$

$$(C_1[c_1] \parallel C_2[c_2] \parallel C_3[c_3] \underset{\gamma}{\bowtie} C_1'[c_1] \parallel C_2'[c_2] \parallel C_3'[c_3]) \underset{\{\alpha,\beta\}}{\bowtie} R_1[r_1] \parallel R_2[r_2]$$

Two actions, β, ζ and δ, result in agents in either C_1, C_2 or C_3 going to a different component type. α does not need any particular treatment as it is a self-loop for C_1. ζ and δ are not involved in any cooperation, which means it is simply required to add the action to C_2' and C_3' respectively. β on the other hand, is involved in a cooperation with R_1. In this case, an additional component type $C_{1_{temp}}$ is added to the model. Finally, γ is involved in a cooperation between components in the original (C_3) and the mirror (C_1') subgroups, neither of which are self-loops. Both C_3' and C_1 need to be updated to account for the change in number of agents in C_3 and C_1' due to the action

γ.

Original subgroup

$$C_1 \stackrel{def}{=} (\alpha, \top).C_1 + (\beta, r_1).C_{1_{temp}} + (\gamma'', \top).C_2$$

$$C_{1_{temp}} \stackrel{def}{=} (\beta', big).C_2$$

$$C_2 \stackrel{def}{=} (\zeta, r_2).C_3$$

$$C_3 \stackrel{def}{=} (\delta, r_3).C_1 + (\gamma, r_\gamma).C_{3_{temp}}$$

$$C_{3_{temp}} \stackrel{def}{=} (\gamma', big).C_1$$

Replicated subgroup

$$C'_1 \stackrel{def}{=} (\gamma, r'_\gamma).C'_2 + (\beta', \top).C'_2$$

$$C_{1_{temp}} \stackrel{def}{=} (\gamma'', big).C'_2$$

$$C'_2 \stackrel{def}{=} (\zeta, \top).C'_3$$

$$C'_3 \stackrel{def}{=} (\delta, \top).C'_1 + (\gamma', \top).C'_1$$

Rest of the model

$$R_1 \stackrel{def}{=} (\beta, rate_1).R_2$$

$$R_2 \stackrel{def}{=} (\alpha, rate_2).R_1$$

$$(C_1[c_1] \parallel C_2[c_2] \parallel C_3[c_3] \underset{\{\gamma,\gamma',\gamma'',\beta',\zeta,\delta\}}{\bowtie} C'_1[c_1] \parallel C'_2[c_2] \parallel C'_3[c_3]) \underset{\{\alpha,\beta\}}{\bowtie} R_1[r_1] \parallel R_2[r_2]$$

It can however be noticed that the model can be simplified and the component types C'_2 and C'_3 can be merged. The reason the mirror is required is to have C_1 and C_3 communicate over γ. In the model presented above, C'_1 and C_3, which results in the same behaviour as C_1 and C'_1 have the same number of agents. C'_2 and C'_3 are not involved in any cooperation apart from the ones that ensure that C_2 and C'_2, and C_3 and C'_3 respectively, have the same number of agents. It is then possible to have a component C'_23 which would represent both C_2 and C_3. The model can

then be simplified to:

Original subgroup

$$C_1 \stackrel{def}{=} (\alpha, \top).C_1 + (\beta, r_1).C_{1_{temp}} + (\gamma'', \top).C_2$$

$$C_{1_{temp}} \stackrel{def}{=} (\beta', big).C_2$$

$$C_2 \stackrel{def}{=} (\zeta, r_2).C_3$$

$$C_3 \stackrel{def}{=} (\delta, r_3).C_1 + (\gamma, r_\gamma).C_{3_{temp}}$$

$$C_{3_{temp}} \stackrel{def}{=} (\gamma', big).C_1$$

Replicated subgroup

$$C'_1 \stackrel{def}{=} (\gamma, r'_\gamma).C'_2 + (\beta', \top).C'_2$$

$$C_{1_{temp}} \stackrel{def}{=} (\gamma'', big).C'_{23}$$

$$C'_{23} \stackrel{def}{=} (\delta, \top).C'_1 + (\gamma', \top).C'_1$$

Rest of the model

$$R_1 \stackrel{def}{=} (\beta, rate_1).R_2$$

$$R_2 \stackrel{def}{=} (\alpha, rate_2).R_1$$

$$(C_1[c_1] \parallel C_2[c_2] \parallel C_3[c_3] \underset{\{\gamma,\gamma',\gamma'',\beta',\delta\}}{\bowtie} C'_1[c_1] \parallel C'_2[c_2] \parallel C'_3[c_3]) \underset{\{\alpha,\beta\}}{\bowtie} R_1[r_1] \parallel R_2[r_2]$$

It can be noticed that the action ζ does no longer appear on the replicated subgroup, or in the cooperation set. This is due to the fact that this action was resulting in an agent in C'_2 going to C'_3. As both components have been merged, it would have resulted in a self-loop with a passive rate. In this case, only if $N(C'_2) = 0$ does ζ in C'_2 have an influence in the model. However, if $N(C'_2) = 0$, then $N(C_2) = 0$ as they have the same number of agents, so the action could not be fired anyway. This means that ζ does not have any influence in the model's behaviour, and can therefore be removed entirely.

4.2 Frequency dependent transmission

As shown by Norman et al. [70], frequency dependent transmission is the most natural way of expressing contact in process algebra. The model in section 2.1 highlights this fact, and gives a first example. In this section, a more general expression is given.

Frequency dependent transmission assumes that the number of contacts made by a single individual does not change with the population size. Begon et al. [5] details how it can be translated for epidemiological disease modelling purposes. In the situation where a *Susceptible* contacts other individuals at rate cr, with cr constant, the probability of this *Susceptible* to meet an infectious individual is I/N with I the number of infectious individuals and N the total number of individuals alive [2]. If the probability for a contact to actually result in an infection is p, then the rate at which a susceptible individual is infected is $cr \times p\frac{I}{N}$. So the overall number of new infections per time step is $cr \times p\frac{S \times I}{N}$, which corresponds to the $\beta\frac{SI}{N}$ use in mathematical biology.

Consider the case where a set $A = C_{i_0}, ..., C_{i_m}$ communicates over an action β with a set $B = C_{j_0}, ..., C_{j_k}$, with $A \cap B$ potentially non-empty. This could correspond to the case where the individuals in A represent different types of infectious individuals, and B individuals that can be contacted (it does not necessarily contain the whole population, as strictly quarantined people might not be available for contact for example). An illustrative example is shown in Figure 4.1, where $\{C_2, C_4\}$=A and $\{C_3, C_4\}$=B, and will be used to illustrate the different cases in the next paragraph.

In this case again, a mirror subgroup will be needed. Just as in the case of direct transmission, the original subgroup, called S, needs to be replicated to allow the C_i's to communicate with the C_j's. Basically, the model will be changed, with S mostly staying unchanged, S' with the

[2] In the next chapter, a way of representing births and deaths will allow the number of individuals alive to change.

Original subgroup

$$
\begin{aligned}
C_0 &\stackrel{def}{=} (\alpha_0, r_1).C_1 \\
C_1 &\stackrel{def}{=} (\alpha_1, r_2).C_2 \\
C_2 &\stackrel{def}{=} (\beta, r_3).C_{2_{temp}} \\
C_{2_{temp}} &\stackrel{def}{=} (quick_2, big).C_3 \\
C_3 &\stackrel{def}{=} (\alpha_3, r_4).C_4 \\
C_4 &\stackrel{def}{=} (\beta, r_5).C_{4_{temp}} + (quick'_4, \top).C_2 \\
C_{4_{temp}} &\stackrel{def}{=} (quick_4, big).C_0
\end{aligned}
$$

Replicated subgroup

$$
\begin{aligned}
C'_{012} &\stackrel{def}{=} (quick_2, \top).C'_3 \\
C'_3 &\stackrel{def}{=} (\beta, r_6).C'_3 + (\alpha_3, \top).C'_4 \\
C'_4 &\stackrel{def}{=} (\beta, r_7).C'_{4_{temp}} + (quick_4, \top).C'_1 \\
C'_{4_{temp}} &\stackrel{def}{=} (quick'_4, big).C'_{012}
\end{aligned}
$$

Figure 4.1: An example of frequency dependent transmission in a model. The action the agents communicate over is β

component types of S replicated (C' correspond to the replicated component type of C), and the remainder of the model remaining unchanged. An exception however would be that when an agent evolves to a new state in S, its counterpart in S' needs to change as well, and vice versa. How to do this is detailed next.

Overall in S and S', we have four types of component types:

$C \notin A$ and $C \notin B$:

When an agent in C moves to a new state C_{target} over an action δ, the replicated agent in C' needs to move to C'_{target} too. Two cases can arise:

- the action δ did not involve a communication: in this case, C simply needs to communicate over that same action with C'.

$$\begin{aligned} C &= (\delta, r_C).C_{target} \\ C' &= (\delta, \top).C'_{target} \end{aligned}$$

- the action δ involves a communication: in this case, C' cannot be included in the communication, as it would change its overall rate.

$$\begin{aligned} C &= (\delta, r_C).C_{temp} \\ C_{temp} &= (\delta', big).C_{target} \\ C' &= (\delta', \top).C'_{target} \end{aligned}$$

  The agent in C first needs to move to a temporary state C_{temp}, that will communicate with C' in order for both the agent in C and the one in C' to respectively move to C_{target} and C'_{target} and make sure S' still replicate the behaviour of S.

$C \in A$ and $C \notin B$:

In this case, we need to include the communication and update both C and C'.

$$\begin{aligned} C &= (\beta, r_C).C_{temp} \\ C_{temp} &= (\delta, big).C_{target} \\ C' &= (\delta, \top).C'_{target} \end{aligned}$$

So here, the communication with C happens at rate r_C. Because C goes to a different state, its replicated component type C' needs to be notified, in order to change as well. As both cannot happen at the same time, C first goes to a temporary state C_{temp} which quickly communicates with C' (the rate *big* corresponds to a big number, as both sides of the communication cannot use $\top$), so that the latter goes to state C'_{target}.

$C \notin A$ and $C \in B$:

In this case, the communication mechanism needs to be included in C', while C needs to be updated as well.

$$\begin{aligned} C &= (\delta', \top).C_{target'} \\ C' &= (\beta, r'_C).C'_{temp} \\ C'_{temp} &= (\delta', big).C'_{target'} \end{aligned}$$

This is the same as the previous case, the only difference being the S' is updated rather than S. It might not seem important in the case of one component type. However, when several component

types are involved, as in the example of Figure 4.1, it is important to make sure that the right subgroup is updated.

$C \in A$ and $C \in B$:

Finally, this case is merging the two previous cases:

$$\begin{aligned} C &= (\beta, r_C).C_{temp} + (\delta', \top).C_{target'} \\ C_{temp} &= (\delta, big).C_{target} \\ C' &= (\beta, r'_C).C'_{target} + (\delta, \top).C'_{target} \\ C'_{temp} &= (\delta', big).C'_{target'} \end{aligned}$$

It should be noted here that the rate at which the action β happens and the state to which the agent evolves can be different for C and C'. Examples can be seen in section 2.2 or in Figure 4.1.

Finally, these general rules need to be adapted to the situation. In particular, it is not necessarily needed to include all the component types in the replicated subgroup S'. As seen in Figure 4.1, it is possible to group all the component types of S that do not belong to B in one single component type. This simplifies the process of updating S' and reduces the model's size and complexity, resulting in shorter processing time. This is illustrated in Figure 4.1, where C'_0, C'_1 and C'_2 are merged as C'_{012}.

It had been already shown that process algebra more naturally expressed frequency dependent transmission [70] and PEPA is no exception. However, detailed instructions are given on how to implement frequency dependent communication in a PEPA model. Now, the case of density dependent transmission will be dealt with.

4.3 Density dependent transmission

Density dependent transmission cannot be naturally expressed in PEPA. However in this section, we will give a method to express it.

Density dependent transmission corresponds to the case where the number of contacts an individual deals is proportional to the population size. It is, for example, frequently used for droplets based diseases, such as influenza. Again, Begon et al. [5] details how it can be expressed. The term is exactly the same as in the case of frequency dependent transmission, the exception being that the contact rate cr varies with the population size. So this time, the number of new infections per time step is $cr(N) \times p\frac{S \times I}{N}$. In the particular case where cr is proportional to N, the rate can be written $cr' \times N$, which gives a number of new infections of $cr' \times \times p\frac{S \times I}{N} = cr' \times p \times SI$.

In PEPA, it will be modelled by adapting a particular case of the frequency dependent transmission model shown in the previous section. However, the resulting method will be less general than the one for expressing frequency dependent transmission. Indeed, in the case of density dependent transmission, the contact rate will be assumed to be proportional to the population size, and only the agents in the mirror subgroup will regularly contact the ones in the original subgroup, the latter ones being passively contacted.

The principle of this method is to make use of the fact that the number of agents in the original subgroup is constant. Indeed, PEPA does not allow the creation or deletion of any processes at any time during a simulation, which guarantees this. Consider the case where C_j needs to contact C_i over the action β at rate r_{C_j}. If an agent a_j in C_j can contact all N agents in the original subgroup, this means only 1 over N contacts made will be with the agent a_i in C_i. Now, if the rate is changed to $r_{C_j} \times N$, the rate at which a_j contacts a_i is $\frac{r_{C_j} \times N}{N} = r_{C_j}$. The overall rate of this action would then be $r_{C_j} \times C_j \times C_i$, which corresponds to the one described by Begon [5].

However, one question may arise: why not simply establish a communication between C_j and C_i? Well, the final rate would not be the same, and would not correspond to what we try to achieve here. Indeed, if the model was [3]:

$$\begin{aligned} C_i &= (\beta, \top).C_{target} \\ C'_j &= (\beta, r_{C_j}).C'_j \end{aligned}$$

the overall rate of the action would be $min(\top \times C_i, r_{C_j} \times C'_j) = r_{C_j} \times C_j$ (if we ignore the case where C_i=0), which means that each agent in C_j communicates with agents in C_i at rate r_{C_j}, but the number of contacts made by each agent in C_j is r_{jC} per time step, **regardless of the number of C_i's** (unless there are none), whereas we need to have r_{jC} contacts between each agent in C'_j and each agent of C_i.

In practice, the four component types described in the previous section require the following activities:

$C \notin A$ and $C \notin B$:

These component types will need to allow the communication over β to happen, but the agents remain in the same state. The communication needs to be instantaneous.

$$C = (\beta, \top).C$$

[3] to simplify the model, we ignore the update of S' here. However, it would not change the overall rate of the action β anyway

$C \in A$ and $C \notin B$:

As in the previous section, C needs to be able to communicate over β and update S'. However, the rate has to be $\top$ this time.

$$\begin{aligned} C &= (\beta, \top).C_{temp} \\ C_{temp} &= (\delta, big).C_{target} \\ C' &= (\delta, \top).C'_{target} \end{aligned}$$

$C \notin A$ and $C \in B$:

Again, this case is very similar to what has been done in frequency dependent transmission. However, like in the first case, C has to allow the communication over β while remaining in the same state. The rate at which C' communicates has also been updated:

$$\begin{aligned} C &= (\beta, \top).C + (\delta, \top).C_{target'} \\ C' &= (\beta, r'_C \times N).C'_{temp} \\ C'_{temp} &= (\delta, big).C'_{target'} \end{aligned}$$

$C \in A$ **and** $C \in B$**:**

Finally in this case, we merge the two previous cases.

$$\begin{aligned} C &= (\beta, \top).C_{temp} + (\delta', \top).C_{target'} \\ C_{temp} &= (\delta, big).C_{target} \\ C' &= (\beta, r'_C \times N).C'_{target} + (\delta, \top).C'_{target} \\ C'_{temp} &= (\delta', big).C'_{target'} \end{aligned}$$

Obviously, the different component types in B can have different rates at which they fire β, and as in frequency dependent transmission, the mirror subgroup does not have to include all of the component rates of the original subgroup, some of them can often be merged.

An example is given in Figure 4.2, where the different cases are illustrated. It is very similar to the one given in section 4.2. Again here, C_0 and C_1 are not involved initially in the communication over β, $\{C_2, C_4\} \in A$ and $\{C_3, C_4\} \in B$. There are two differences though. Some of the rates have been updated: the component types in the original subgroup are now passive, while the ones in the mirror subgroup fire at rate r_4 and r_5 for respectively C_3 and C_4, which is translated in a rate of $r_4 \times N$ and $r_5 \times N$. The second change concerns C_0 and C_1: even though they are not directly involved in the communication, they now passively allow it to happen.

4.4 Summary

Frequency and density dependent transmission can now be expressed in PEPA. There are some limitations on the density dependent methods, but most cases are covered, and most epidemiological disease terms can now be translated.

Original subgroup

$$
\begin{aligned}
C_0 &\stackrel{def}{=} (\beta, \top).C_0 + (\alpha_0, r_1).C_1 \\
C_1 &\stackrel{def}{=} (\beta, \top).C_1 + (\alpha_1, r_2).C_2 \\
C_2 &\stackrel{def}{=} (\beta, \top).C_{2_{temp}} \\
C_{2_{temp}} &\stackrel{def}{=} (quick_2, big).C_3 \\
C_3 &\stackrel{def}{=} (\beta, \top).C_3 + (\alpha_3, r_3).C_4 \\
C_4 &\stackrel{def}{=} (\beta, \top).C_{4_{temp}} + (quick'_4, \top).C_2 \\
C_{4_{temp}} &\stackrel{def}{=} (quick_4, big).C_0
\end{aligned}
$$

Replicated subgroup

$$
\begin{aligned}
C'_{012} &\stackrel{def}{=} (quick_2, \top).C'_3 \\
C'_3 &\stackrel{def}{=} (\beta, r_4 \times N).C'_3 + (\alpha_3, \top).C'_4 \\
C'_4 &\stackrel{def}{=} (\beta, r_5 \times N).C'_{4_{temp}} + (quick_4, \top).C'_1 \\
C'_{4_{temp}} &\stackrel{def}{=} (quick'_4, big).C'_{012}
\end{aligned}
$$

Figure 4.2: An example showing how density dependent transmission can be expressed in PEPA

These methods will form the basis of the next section, that will study how births and deaths can be modelled.

Chapter 5

Modelling births and deaths

The birth and death feature is essential to model populations over long time-spans. It might be acceptable for simulations over a short period of time to ignore natural population fluctuations; however, their influence on population behaviour gets noticeable as the timespan studied increases. Usually, when a simulation run is comparable to the life expectancy of an individual, births and deaths have a significant impact on the behaviour of the model. Also, when the immunity to the disease is lifelong, as in the case of measles, described in Chapter 6, the only source of susceptible individuals is usually newborns. In this case, it is also essential to include births and deaths in the model. PEPA, however, is very limited when expressing births and deaths. The main reason is that agents, that would correspond to individuals being born and dying respectively, cannot be created or deleted. Considering that this feature is essential for disease modelling, and more generally to population modelling, this chapter is devoted to detailing different methods to include births and deaths to a PEPA model.

This chapter will be separated in four sections. First some early issues and simple solutions to this problem will be examined. Then, two methods will be presented. The first one assumes

that the birth rate decreases with the population size, until reaching a carrying capacity K and that the death rate remains constant. The second method assumes a constant birth rate, but each individual in the population has to compete for a limited resource. As the amount of available resource remains constant, the population cannot grow over a threshold that depends on the amount of resource. Finally, now that all the necessary tools and methods are available, a full model of the bubonic plague in a prairie dog town will be presented and analysed.

5.1 Introducing births and deaths to a PEPA model

PEPA's grammar does not allow processes to be created or deleted. This means that an agent cannot be created to represent a newborn individual, and the agent representing an individual dying cannot be deleted.

5.1.1 Including births and deaths in a constant population

In a population that remains constant at all times, it is possible to introduce births and deaths by assuming that a birth occurs each time an individual dies. In other words, an agent representing an individual dying will immediately then represent a newborn. This means that the dying agent goes directly to the state in which individuals are born in this particular model.

While this solution might be suitable for some models, in particular if the population is big enough for the approximation to be made, and the simulation time is short enough, it is too big an assumption for most models.

5.1.2 Introducing Ghosts

In order to overcome the limitation on the creation and deletion of processes, a new component type is created: the ghosts. The ghosts correspond to available souls, ready to be born when needed. Also, when an individual dies, it becomes a ghost. The initial number of ghosts has to be chosen carefully: too many means longer processing times for the model, too few and the pool of ghosts might run out and the number of births will be blocked as long as it is empty, leading to a behaviour not reflecting what the modeller had in mind.

A method to describe births and deaths arises naturally from the introduction of ghosts, as detailed in the following example:

$$Alive \stackrel{def}{=} (death, death\_rate).Ghost$$
$$Ghost \stackrel{def}{=} (birth, birth\_rate).Alive$$

In this example, the *Alive* individuals die at a rate *death_rate* and the *Ghost* individuals give birth at a rate *birth_rate*. This basic solution needs to be modified to be used practically. If the death rate is smaller than the birth rate, then the population size will increase until the pool of ghosts runs out, and if the birth rate is bigger, the population size will decrease until everyone is dead. Furthermore, the number of births depends on the number of agents in *Ghost*, which has no biological justification.

One solution to avoid population explosion is to use the number of available ghosts to regulate the population size. If the birth and death rate are roughly constant, the number of agents initially in *Ghost* can be chosen such that $Ghost+Alive=K$ with K the carrying capacity of the population. This way, the population can never increase past the carrying capacity, and the overall number of births naturally decreases with the number of ghosts. In general, this solution is biologically unrealistic, and may only be acceptable for cases where the population does not fluctuate much.

5.1.3 Vertical transmission

Vertical transmission corresponds to the situation where an infectious mother gives birth to a infected child. While it is a transmission mechanism, it was not dealt with in the transmission section as it is included in the birth mechanism. Indeed, vertical transmission requires the agent in *Ghost* corresponding to the newly born individual to go in the exposed/infectious (depending on the disease) state. This may imply, depending on the mechanism, a second birth action, in order to differentiate between the healthy mother giving birth and the infectious one.

5.2 Modelling births and deaths directly

This method of modelling births and deaths is based on the principle that both the birth and the death rates are constant with the population size, but newborns are more likely to die if the population density is high. The rate at which the newborns die is density dependent, and this section will extensively use the results obtained in section 4.3. The biological reason behind this lies in the fact that the mortality of children before the age of five is usually higher than the rest of the population, and one can imagine that when the population approaches its carrying capacity, and resources (food, access to medical care, etc...) becomes scarce, children are more vulnerable than the rest of the population.

A few notations will be used in this section. In the considered model, the subgroup which contains the individuals giving birth/dying will be referred to as $\mathcal{S}$ and its replicated subgroup as $\mathcal{S}'$. If required, the replicated subgroup can be constructed using the method presented in section 4.1. The number of agents in $\mathcal{S}$, which is constant over time, is noted s. $\mathcal{B}$, $\mathcal{D}$ and $\mathcal{N}$ correspond to the groups of component types that can give birth, die and are alive respectively,

and the corresponding replicated groups of components will be noted $\mathcal{B}'$, $\mathcal{D}'$ and $\mathcal{N}'$. It can be noticed that $\mathcal{B} \subseteq \mathcal{N}$, $\mathcal{D} \subseteq \mathcal{N}$ and $\mathcal{N} \subseteq \mathcal{S}$. The number of agents in $\mathcal{N}$ is N. K is the population carrying capacity, which means that at all times, $N \leq K$.

Three actions are to be considered for this method: *birth*, *infant_death* and *death*. As the birth rate is constant over time, the addition of *birth* is straightforward: for each component type $C \in \mathcal{B}$, the activity $(birth, birth\_rate).C$ is added to the equation of C. *death* is modelled exactly the same way. For *infant_death*, the method used in section 4.3 can be adapted to this case: the overall rate at which infants die has to increase with the population size. Furthermore, the population must not exceed the carrying capacity K, which means that the rate at which *infant_death* happens must be equal to the birth rate when the population size is equal to K. This can be obtained by having an overall rate equal to $\beta SN/K$. This can be achieved as described in the following steps:

- for every component $C \notin \mathcal{B}$ (including *Ghost*): add $(infant\_death, \top).C$ to the original subgroup. These activities are required in order to obtain the desired rate, as detailed in section 4.3.
- for every component $C \in \mathcal{B}$: add $(infant\_death, \top).C_{temp}$ to the original subgroup. A component C_{temp} is also required, which allows the action $(quick_C, big).Ghost$. The action $(quick_C, \top).Ghost'$ also needs to be added in C', the replicated component of C. The action $quick_C$ and the component type C_{temp} are required to update the replicated subgroup, as detailed in section 4.1.
- for every component $C \in \mathcal{N}$: the action $(infant\_death, birth\_rate \times s/K).C$ has to be added, with s the total number of agents in the original subgroup (including *Ghost*).

Figure 5.1 illustrates this method. This model focuses on births and deaths, and ignores all

other mechanisms. In this example, $\{S, R\}$=$\mathcal{B}$, $\{S, R, I\}$=$\mathcal{D}$, s is the total number of agents $(S + I + R + Ghost)$ and K the carrying capacity.

Original subgroup

$$
\begin{aligned}
S &\stackrel{def}{=} (birth, birth\_rate).S + (infant\_death, \top).S_{temp} \\
&\quad +(death, death\_rate).Ghost \\
S_{temp} &\stackrel{def}{=} (quick_{sr}, big).Ghost \\
I &\stackrel{def}{=} (infant\_death, \top).I + (death, death\_rate).Ghost \\
R &\stackrel{def}{=} (birth, birth\_rate).R + (infant\_death, \top).R_{temp} \\
&\quad +(death, death\_rate).Ghost \\
R_{temp} &\stackrel{def}{=} (quick_{sr}, big).Ghost \\
Ghost &\stackrel{def}{=} (newborn, big).S + (infant\_death, \top).Ghost
\end{aligned}
$$

Replicated subgroup

$$
\begin{aligned}
SR' &\stackrel{def}{=} (infant\_death, birth\_rate \times s/K).SR' + (quick_{sr}, \top).Ghost' \\
&\quad +(death, \top).Ghost \\
I' &\stackrel{def}{=} (infant\_death, birth\_rate \times s/K).I' + (death, \top).Ghost \\
Ghost' &\stackrel{def}{=} (birth, big).Ghost_{temp} \\
Ghost_{temp} &\stackrel{def}{=} (newborn, big).SR'
\end{aligned}
$$

Figure 5.1: An example of how birth can be directly modelled.

This method, while not being very intuitive, is quite simple to set up. The overall population growth derived from this model corresponds to the Verhulst logistic equation, also referred as as the Verhulst-Pearl equation widely used in population growth modelling [86, 8, 50]. It also corresponds to the term used by Webb et al. [92] in the system used for the bubonic plague model presented in section 5.5. When the two terms corresponding to the actions *birth* and

infant_death are put together, the resulting overall number of effective births for a component type C is $birth\_rate \times C(1 - N/K)$.

5.3 Modelling births and deaths using limited resources

Another way to model births and deaths and avoid population explosion, is to force the individuals in the population to compete for a resource. While the resulting methods are not explicitly density dependent, the resulting population growth exhibits a density dependent behaviour.

In all this section, the notation introduced in section 5.2, and in particular $\mathcal{S}$, s, $\mathcal{B}$, $\mathcal{D}$, $\mathcal{N}$, N and K will be used in the description of the two methods. In addition to this notation, the number of agents in $\mathcal{B}$ and $\mathcal{D}$ will be noted b and d respectively.

Birth Token

In this first method, the fight for resource is necessary to give birth. This means that the average number of births per individual per time steps depends both on the birth rate and on the number of available resources. The latter in turn depends on the overall number of individuals trying to give birth, the average number of births will therefore decrease with the population size, until the threshold set by the amount of available tokens is reached.

A new subgroup is first added to the model, corresponding to the available resource. The resource can be in two different states: available, or not available. Then in the original subgroup $\mathcal{S}$, each of the components in $\mathcal{B}$ will fire an action *may_birth*, and try this way to contact an agent in *Resource*. If the contact is made, the agent in *Resource* temporarily moves to a state *BirthGiven*

that ensures a new individual is created by cooperating with an agent in *Ghost*, which in turn goes to a component type in $\mathcal{N}$. The agent in *BirthGiven* then moves to the state *ResourceUnavailable*, where it remains, unavailable for contact, until it comes back to *Resource*, where it can cooperate again. Concerning death, it is similar to what has been presented in the previous section. Each component capable of dying from natural causes can fire an action *death* at the rate *death_rate*. A simple example is presented in Figure 5.2.

Resource subgroup

$$Resource \stackrel{def}{=} (may\_birth, available\_rate).BirthGiven$$

$$BirthGiven \stackrel{def}{=} (birth, big).ResourceUnavailable$$

$$ResourceUnavailable \stackrel{def}{=} (wait, unavailable\_rate).Resource$$

Original subgroup

$$C \stackrel{def}{=} (may\_birth, birthrate).C + (death, death\_rate).Ghost$$

$$Ghost \stackrel{def}{=} (birth, \top).C$$

Figure 5.2: A resource based birth and death model based on the first method.

One question still remains: how are the parameters and initial conditions for the resource chosen? The objective here is to make sure the population remains under, or close to, the carrying capacity K. This means that the overall number of births must be smaller than or equal to the overall number of deaths when the carrying capacity is reached. Before we treat this question, some notation is necessary. For this question, *PotentialResource* will be the number of agents in the subgroup dealing with the resources, which means that *PotentialResource*=*Resource* + *BirthGiven* + *ResourceUnavailable*, and *available_rate* and *unavailable_rate* will have the same meaning they have in Figure 5.2. We will also assume that everyone in the population can die, which means that $\mathcal{D}=\mathcal{N}$ and $d=n$. Two cases arise here:

- $birth\_rate < death\_rate$ and $\mathcal{B}{=}\mathcal{D}$: in this case, the population size can only decrease, and will never reach the carrying capacity.

- otherwise: in this case, the overall number of deaths at carrying capacity is $death\_rate \times d{=}death\_rate \times K$. On the other hand, the overall number of births must be limited by the resource. And the resource only allows $(available\_rate + unavailable\_rate) \times PotentialResource$ births. The reason behind this term is that a resource can only be used every $(available\_rate + unavailable\_rate)^{-1}$ time steps, and there is $PotentialResource$ of them. So in the end, the parameters and initial conditions must be chosen such that:

$$(available\_rate + unavailable\_rate) \times PotentialResource{=}death\_rate \times K$$

Given that the death rate and the carrying capacity are given by the biological system, two of the other three rates can be chosen, and this equation then determines the third one.

Competition for Food

This method is a lot more biologically intuitive than the first one. In this case, every living individual in the population is required to eat at rate eat_rate. If the individual succeeds, it lives. Otherwise, it dies. Both the birth and the death processes are straightforward actions occurring at their respective apparent rate.

In order to model this principle, the resource has first to be modelled. This is identical to the Birth Token method above. One difference concerns the names. Obviously now, the individuals *eat* rather than *may_birth*. The second more important difference is the absence of the component *BirthGiven*. The reason it was needed in the first method was to make sure the information about the successful birth was passed to the *Ghost*, so a new individual was created. In this case, no

individual is born as a result of the consumption of the resource, so it is not necessary. Finally, the *eat_rate* is now very big. The reason is that when the individual attempts to eat, it either succeeds, or dies if it does not after $1/$*starve_rate* time steps. This means that if the eat rate was small, there would be a chance for this individual to die, even though some food is available.

Concerning the original model, some additional components have to be added. Each component included in the calculation of the carrying capacity K needs to regularly fight for food. This implies that each of these components needs to be able to fire two new actions. The first one is the *eat* activity, which is passive, as explained previously. The second activity is *starve*: after an average of $1/$*starve_rate* time steps, if the individual does not manage to have access to the resource, it dies. If these two actions are added directly to the components, two problems arise. First, because the activity *eat* is fired at a *big* rate, this means that it will be fired in very rapid succession until the *Resource* runs out. This could not be solved by replacing the *big* rate by the rate at which the agent needs to eat, as it would compete with the other components, and in particular *starve_rate*. The other issue concerns *starve_rate*. Indeed, even if the agent eats regularly, because of the way the rate resets, the agent might fire *starve* anyway. In order to avoid this, a new action has been created: *need_to_eat*. This action is fired regularly, and when this happens, an agent in C goes to C'. This component allows the two actions *eat* and *starve* to be fired, respectively at *big* and *starve_rate*. Once the agent has eaten, it comes back to C. If it does not eat, it fires *starve* and becomes a *Ghost*.

Finally, birth and death have also to be added to the model. While including death is very straightforward, and only requires a *death* action to be added to the component types, the birth process will require the addition of two additional components. When an agent gives birth, it needs to communicate with a *Ghost* agent, so a new individual is generated. As they are in the same subgroup, it cannot be done directly. In a similar fashion to what has been done previously in the

Resource subgroup

$$Resource \stackrel{def}{=} (eat, \top).ResourceUnavailable$$

$$ResourceUnavailable \stackrel{def}{=} (wait, unavailable\_rate).Resource$$

Original subgroup

$$C_{1_{food}} \stackrel{def}{=} (eat, big).C_1 + (starve, starve\_rate).Ghost$$

$$C_{2_{food}} \stackrel{def}{=} (eat, big).C_2 + (starve, starve\_rate).Ghost$$

$$C_1 \stackrel{def}{=} (give\_birth, birth\_rate).C_1 + (need\_to\_eat, n2e\_rate).C_{1_{food}} + (death, death\_rate).Ghost$$

$$C_2 \stackrel{def}{=} (need\_to\_eat, n2e\_rate).C_{2_{food}} + (death, death\_rate).Ghost$$

$$Ghost \stackrel{def}{=} (birth, \top).C_1$$

Messenger subgroup

$$Receiver \stackrel{def}{=} (give\_birth, \top).Stork$$

$$Stork \stackrel{def}{=} (birth, big).Receiver + (no\_ghost, 100.0).Receiver$$

Figure 5.3: A resource based birth and death model based on the second method.

replicated subgroup, a *Receiver* communicates with the agent over the action *give_birth*, becomes a *Stork* which in turn communicates with a *Ghost* to create a new individual. In the case where no agents are in *Ghost*, the *Stork* would have to wait for agents to be available, and no additional births would be possible. For this reason, an action *no_ghost* is added to *Stork* with a rate much bigger than all the rates in the model, but still much smaller than *big*, in order to avoid this situation. If the number of agents in *Ghost* is sufficient, the action *no_ghost* can be ignored. One agent initially in *Receiver* is sufficient to perform this task, as the time it remains in *Stork* is very short. *Receiver* and *Stork* can however be included in the *replicated subgroup* if there is one, for a simpler model.

Once again, the parameters and initial conditions of the resource subgroup need to be determined. As in the first method, if $birth\_rate < death\_rate$ and all the components capable of giving birth can also die, the resource is irrelevant as the carrying capacity cannot be reached. In all other cases, we need to make sure that the population can only grow until the food is not sufficient to sustain it. When the carrying capacity K is attained, the population requires $n2e\_rate \times K$ resources per time step to survive. The amount of resource available per time step is $unavailable\_rate \times PotentialResource$, with $PotentialResource = Resource + ResourceUnavailable$, which is constant over time. Overall, this means that:

$$unavailable\_rate \times PotentialResource = n2e\_rate \times K$$

Once again, either *unavailable_rate* or *PotentialResource* can be chosen without constraint, the second one will then be derived from the equation.

5.4 Conclusion on births and deaths techniques

In the sections 5.2 and 5.3, we have presented three different methods to include births and deaths in a PEPA model. The first method considers the population size by considering the possibility for infants to die, the probability depending on the overall population size. This method usually generates the simplest model, and does not usually (granted that the replicated subgroup is already needed) require extra component types. However, the terms used are unusual in PEPA, and the philosophy behind them is not intuitive, and does not have any biological justification. The methods based on competition for resources are more intuitive. The first method in that category is not completely natural. Even though a new subgroup is required, it remains simple, and scales well with the number of component types in the original model. Finally, the second method is the most intuitive: the resource corresponds to food, which is in limited supply. Unfortunately the model's complexity increases proportionally to the number of component types, which in turns increases the processing time. The choice between these methods will ultimately depend on the biology of the disease as well as the complexity of the problem.

Now that the tools required to build the model are available, we can move on to modelling the bubonic plague in a population of prairie dogs.

5.5 Modelling a prairie dog town during a bubonic plague outbreak

The methods presented in the first chapters provided all the necessary tools to build our first epidemiological model based on a real biological system. This model consists of a town of a hundred prairie dogs, two of which are infectious, a thousand fleas, 10 of which are infectious and

a healthy environment reservoir, big enough to be considered infinite. The model itself is available in Figure 5.4.

5.5.1 Description of the model

This model is based on the paper by Webb et al. [92]. According to this paper, the bubonic plague has three main routes of transmission. First of all, it can be transmitted from one infectious animal to another by direct contact. Indeed, prairie dogs being very sociable animals, close contacts between animals from the same town are very frequent. The second vector of transmission is indirect: the bodies of dead infectious prairie dogs remain infectious for an extended period of time. The last vector is the flea. When it bites a prairie dog, a flea can become infected, and might pass the disease to another prairie dog. This last vector has long been considered the main source of transmission [87, 72], but recent papers [92, 24] suggested that the long incubation period and the short infectious window of the bubonic plague in the flea indicated that another important source of transmission was involved. In this section, we will analyse the relative influence of each of the vectors, in order to validate the conclusion presented in the paper by Webb et al. [92], that indirect transmission is the main contributor to disease transmission, and that fleas could not sustain the disease alone.

The model itself is composed of five subgroups. The first two subgroups represent the original prairie dogs subgroup and its replicated subgroup. The three types of contacts are modelled (direct via the *direct_contact* action, indirect via the *indirect_contact* action and vector-borne via the *contact* action), with methods similar to the ones described in Chapter 2. Direct transmission uses a frequency dependent term, while indirect transmission via the infected soil is using a density dependent term, as proposed in [92]. The term for the vector-borne infection differs from the one in the article. Indeed, the term used in the paper's ODE model is $transmission\_rate \times Iq \times S(1 -$

$$
\begin{aligned}
S &\stackrel{def}{=} (contact\_direct, big).Exposed + (indirect\_contact, big).Exposed + \\
&\quad (contact, big).Exposed + (birth, birth\_rate).S + (die\_1, death\_rate\_PD).Ghost + \\
&\quad (infant\_death, big).Stemp \\
Stemp &\stackrel{def}{=} (quick\_S, big).Ghost \\
Exposed &\stackrel{def}{=} (contact\_direct, big).Exposed + (indirect\_contact, big).Exposed + \\
&\quad (contact, big).Exposed + (infected, incubation\_rate).Infectious + \\
&\quad (die\_1, death\_rate\_PD).Ghost + (infant\_death, big).Exposed \\
Infectious &\stackrel{def}{=} (contact\_direct, big).Infectious + (indirect\_contact, big).Infectious + \\
&\quad (infect\_flea, inf\_flea\_rate).Infectious + (infect\_environment, inf\_env\_rate).Infectious \\
&\quad + (contact, big).Infectious + (die\_inf, death\_rate\_inf\_PD).Ghost + \\
&\quad (birth, birth\_rate).Infectious + (die\_2, death\_rate\_PD).Ghost + \\
&\quad (infant\_death, big).Infectious \\
Ghost &\stackrel{def}{=} (indirect\_contact, big).Ghost + (alive, big).S + (infant\_death, big).Ghost \\
SainPD &\stackrel{def}{=} (die\_1, big).GhostPD + (infected, big).InfectiousPD + (quick\_S, big).GhostPD + \\
&\quad (infant\_death, birth\_rate * n/K).SainPD \\
InfectiousPD &\stackrel{def}{=} (die\_inf, big).GhostPD + (die\_2, big).GhostPD + \\
&\quad (contact\_direct, contact\_direct\_rate).InfectiousPD + \\
&\quad (infant\_death, birth\_rate * n/K).InfectiousPD \\
GhostPD &\stackrel{def}{=} (birth, big).TempPD \\
TempPD &\stackrel{def}{=} (alive, big).SainPD \\
Sq &\stackrel{def}{=} (death, fdeath\_rate).GhostF + (findHost, eff * meanNbHost).Sh \\
Sh &\stackrel{def}{=} (flea\_birth, fbirth\_rate).Sh + (death, fdeath\_rate).GhostF + \\
&\quad (leaveHost, leave\_Host\_Rate).Sq + (infect\_flea, big).Eh \\
Eq &\stackrel{def}{=} (death, fdeath\_rate).GhostF + (findHost, eff * meanNbHost).Eh + \\
&\quad (infectiousF, flea\_incubation\_rate).Iq \\
Eh &\stackrel{def}{=} (flea\_birth, fbirth\_rate).Eh + (death, fdeath\_rate).GhostF + \\
&\quad (leaveHost, leave\_Host\_Rate).Eq + (infect\_flea, big).Eh + \\
&\quad (infectiousF, flea\_incubation\_rate).Ih \\
Iq &\stackrel{def}{=} (death, finf\_death\_rate).GhostF + (findHost, eff * meanNbHost).Ih \\
Ih &\stackrel{def}{=} (death, finf\_death\_rate).GhostF + (leaveHost, leave\_Host\_Rate).Iq + \\
&\quad (infect\_flea, big).Ih + (contact, transmission\_rate * eff * meanNbHost).Ih \\
GhostF &\stackrel{def}{=} (flea\_alive, big).Sq \\
AliveFlea &\stackrel{def}{=} (death, big).GhostFlea \\
GhostFlea &\stackrel{def}{=} (flea\_birth, big).TempFlea \\
TempFlea &\stackrel{def}{=} (flea\_alive, big).AliveFlea \\
InfEnv &\stackrel{def}{=} (indirect\_contact, contact\_env\_rate * n).InfEnv + (decay, decay\_rate).HealthyEnv \\
HealthyEnv &\stackrel{def}{=} (infect\_environment, big).InfEnv
\end{aligned}
$$

$$
\begin{aligned}
&((S[98] \parallel Infectious[2] \parallel Ghost[150]) \underset{<*>}{\bowtie} (SainPD[98] \parallel InfectiousPD[2] \parallel GhostPD[150])) \underset{<*>}{\bowtie} \\
&((Sq[990] \parallel Iq[10] \parallel GhostF[10000]) \underset{<*>}{\bowtie} (AliveFlea[1000] \parallel GhostFlea[10000])) \underset{<*>}{\bowtie} HealthyEnv[100000]
\end{aligned}
$$

Figure 5.4: PEPA model of a prairie dog town infection by the bubonic plague

$e^{-eff.N/B}$) with Iq the number of infectious fleas questing for a host, N the number of living prairie dogs,*eff* the searching efficiency of questing fleas and B the number of burrows a host enters. Because the effect of the exponential cannot be reproduced in PEPA, as only constant rates are allowed, it had to be approximated. Begon [6] showed that this term can be approximated by $transmission\_rate \times Iq \times S \times eff.N/B$. Birth is modelled as described in section 5.2. The introduction of death however, required a few modifications to be made to the method presented in section 5.2. First of all, two different causes of death had to be considered: death from natural causes (*die_1* and *die_2*) and death from the disease (*die_inf*). The reason why two different actions had to be used for natural death, is to make sure the correct component in the replicated subgroup is updated when a prairie dog dies. If a susceptible prairie dog dies, for example, we need to ensure that a *SainPD* dies, rather that an *InfectiousPD*. Finally, the action *infect_flea* correspond to the infection of a flea by a prairie dog.

The next two subgroups correspond to the flea. Each flea is either questing, or on a host. This results in 7 components in the subgroups: *Sq* (susceptible questing), *Sh* (susceptible, on the host), *Eq* (exposed questing), *Eh* (exposed, on the host), *Iq* (infectious questing), *Ih* (infectious, on the host), and *GhostF* (the pool of ghosts for the fleas). Each questing agent tries to find a host, while each agent on a host might leave the host. The fleas on a prairie dog host might be infected via the action *infect_flea*. It has to be noted that there is no way in PEPA to track which flea is on which prairie dog. This means that it is possible that a flea might have a contact with two different prairie dogs even though it has not gone through a questing phase in between the two contacts. The approximation is however also made in the ODE model proposed by Webb, and it is assumed that the rates provided take it into account. The same approximation has been made for the action *contact*, for the same reasons. Here again, in a similar fashion to the prairie dog subgroup, fleas can die either naturally, or from the disease. However, the action is the same, as all the living fleas belong to the same component type (*AliveFlea*) in the replicated subgroup. Finally, birth has not

been modelled the way it is by Webb et al. [92]. Indeed, the method used in that paper could not be replicated in PEPA, and a simple action *flea_birth* with a constant rate was chosen, in order to keep to model simple. As the actual rate was not given by Webb, its value has been chosen within biologically realistic bounds.

The last subgroup corresponds to the soil. The method used here is the one described in section 2.1.

All the parameters, apart from *fbirth_rate*, *meanNbHost* and *inf_env_rate*, are taken from the paper from Webb et al. [92], and reported in Figure 6.7. It has however to be noted that the number of potential hosts n is greater that the carrying capacity K. The reason is that while K is the carrying capacity, and the overall number of births is equal to the number of infant deaths, the stochasticity of PEPA can result in a population higher than K at some point in the simulation. So having n higher than K limits the likelihood that the pool of ghosts is exhausted in one of the stochastic simulation runs.

5.5.2 Results

Initially, the objective was to compare the results of our model (both the average of a hundred stochastic simulations and the ODEs derived from the model) to the data gathered on the Pawnee National Grasslands and used by Webb et al. [92]. This data was unfortunately not publicly available, and the point of comparison of this study will be the ODEs provided by Webb et al., who are said to give "a good qualitative and quantitative fit to the data".

The average of a hundred simulations, the ODEs derived from the model and the original ODEs presented in the paper from Webb et al. [92] are compared in Figures 5.6, 5.7 and 5.8, respectively

Parameter	Value	Description
birth_rate	0.0866	Intrinsic rate of increase (host)
K	200	Carrying capacity (host)
n	250	Total number of host (including the ghosts)
death_rate_PD	0.0002	Natural mortality rate (host)
transmission_rate	0.09	Flea transmission rate
contact_direct_rate	0.073	Airborne transmission rate
contact_env_rate	0.073/20	Transmission rate from reservoir
inf_env_rate	0.5	Transmission rate of the infectious prairie dogs to the reservoir
incubation_rate	0.21	Incubation period^{-1} (host)
death_rate_inf_PD	0.5	Infected host mortality rate
decay_rate	0.006	Reservoir decay rate
leave_Host_Rate	0.05	Rate of leaving hosts
eff	0.004	Searching efficiency of questing fleas
meanNbHost	100	Average number of prairie dogs alive
fdeath_rate	0.07	Natural mortality rate (vector)
inf_flea_rate	0.28	Transmission rate: hosts to vector
flea_incubation_rate	0.009	Incubation period^{-1} (vector)
finf_death_rate	0.33	Disease-induced mortality rate (vector)
fbirth_rate	0.14	Intrinsic rate of increase (vector)

Figure 5.5: The parameters used in the model in Figure 5.4.

representing the number of infectious prairie dogs, the number of susceptible prairie dogs, and the number of infectious fleas.

The differences between the three graphs are fairly small. In particular, both sets of ODEs give very similar results, with a Root Mean Square (RMS) value of between 28.05 for the whole population of prairie dogs, 2.36 in the case of infectious prairie dogs, and 0.17 in the case of the infectious fleas. However, it can be noticed that, in the case of the graphs of the infectious prairie dogs and whole town, the difference between the output of the Stochastic Simulation Algorithm (SSA) and the ODEs is larger. This is caused by the phenomenon explained at the end of section 3.1.2: in some simulations, the disease simply dies out, and the average number of prairie dogs increases, while the average of infectious prairie dogs decreases. In the case of the ODEs, this cannot happen. This feature can be used to estimated the percentage of times when the disease will successfully eradicate the prairie dog town. Indeed, if the disease fades, the number of prairie dogs will grow until the population reaches the carrying capacity K. However, if the disease persists, every prairie dog is killed, and the population size is zero. This means that if the number of prairie dogs at steady state is s on average, the disease success rate is:

$$1 - s/K \tag{5.1}$$

We now consider the relative influence of each vector of transmission.

First of all, the model was analysed with the exact same parameters as in Figure 6.7, but *contact_direct_rate* and flea contact rate (which is *transmission_rate* $\times$ *eff* $\times$ *meanNbHost*) are set to zero. This means that only indirect transmission via the environment was active in that model. The population size over time is presented in Figure 5.9. The resulting graph is very similar to the one including the three vectors shown in Figure 5.6. However, one difference is to be noticed: while the steady population size was around 40 in the case of the three vectors, it is around 60 now.

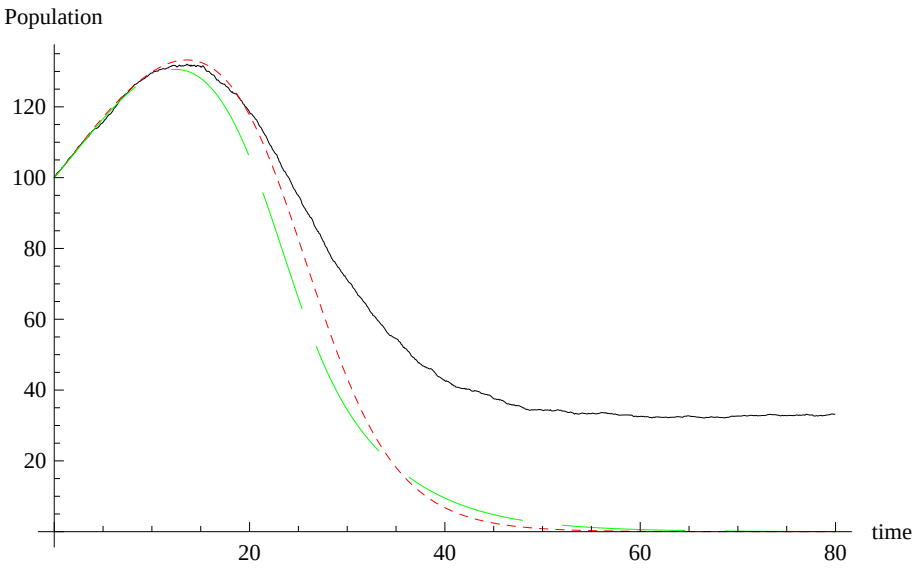

Figure 5.6: Total population of prairie dogs. SSA (solid line), ODEs derived from the PEPA model (dashed line), Webb's ODEs (dotted line). RMS=28.05

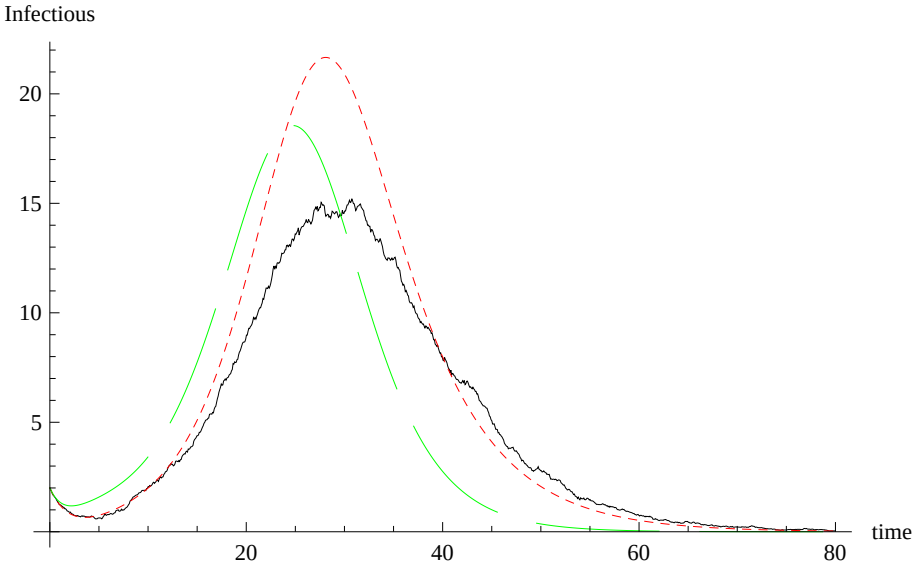

Figure 5.7: Infectious prairie dogs. SSA (solid line), ODEs derived from the PEPA model (dotted line), Webb's ODEs (dashed line). RMS =2.36

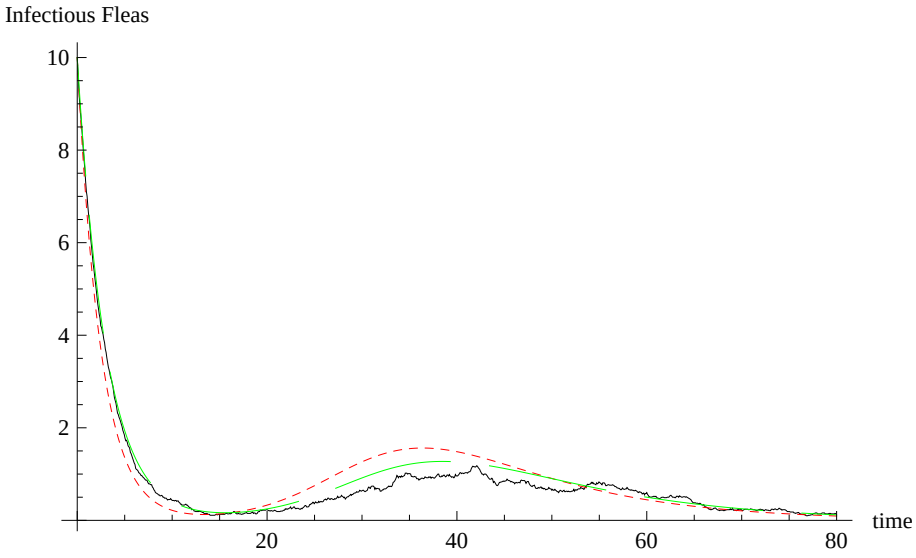

Figure 5.8: Infectious fleas. SSA (solid line), ODEs derived from the PEPA model (dotted line), Webb's ODEs (dashed line). RMS=0.17

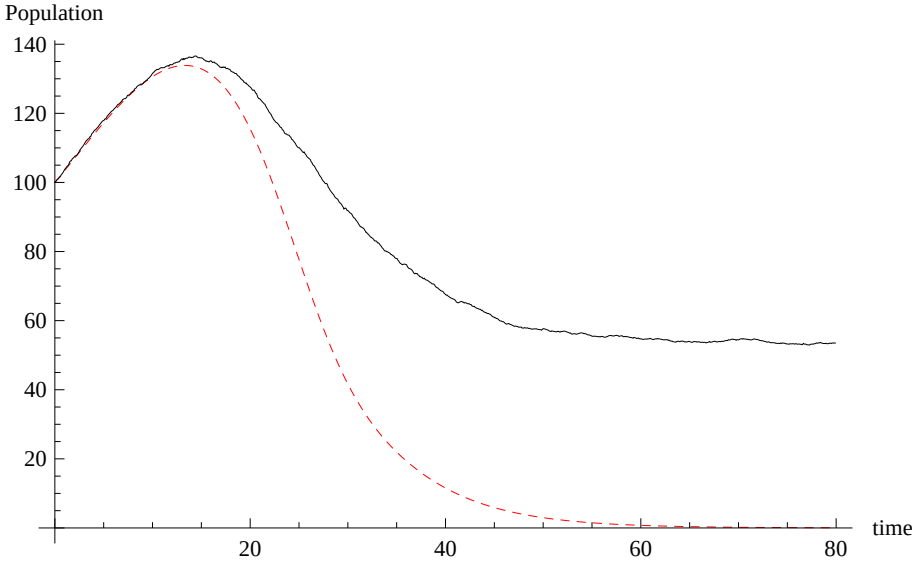

Figure 5.9: PEPA model of a town of prairie dogs infected with the bubonic plague, where the infection can only be passed by the infectious prairie dogs. SSA (solid line), ODES derived from the PEPA model (dotted line).

Using the formula (5.1), we conclude that the prairie dog town is eradicated by the bubonic plague 70% of the time now, whereas it is wiped 80% of the time when the three vectors are present.

The model has then been run with only direct transmission and the transmission via fleas activated. The graph in Figure 5.10 shows that even both vectors together do not make the bubonic plague infectious enough to kill all prairie dogs in the town. The fact that the average of the stochastic simulations reaches the carrying capacity K and that the ODEs successfully describe the behaviour of the model shows that total elimination did not happen in any of the 100 simulations.

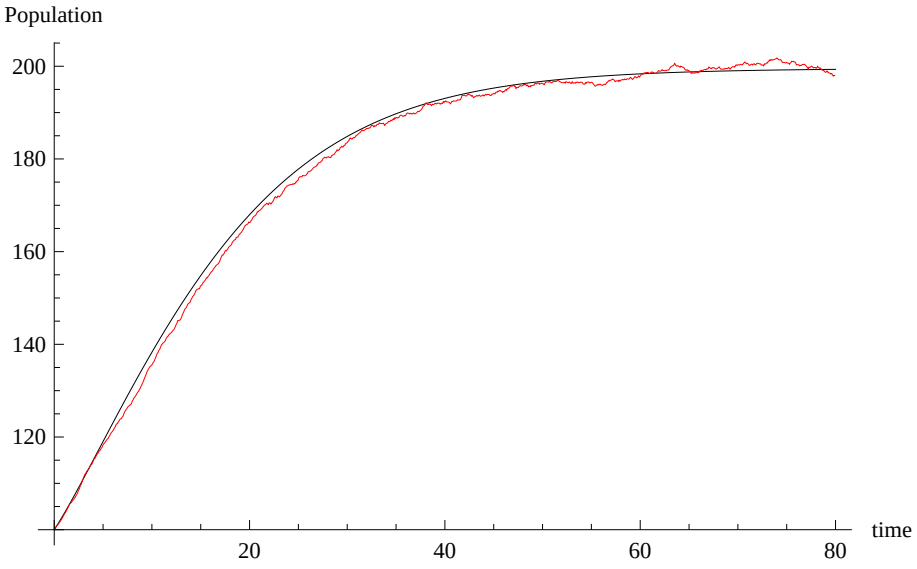

Figure 5.10: PEPA model of a town of prairie dogs infected with the bubonic plague, where the infection can only be passed by the fleas and prairie dogs directly.

5.5.3 Conclusion

Modelling the outbreak of bubonic plague in a prairie dog town provided a lot of insight on the capacity of PEPA to describe epidemiological problems. Indeed, all the essential features could

be integrated in the model. The results have also been validated by confirming the findings on previous models. Finally, stochastic simulations have shown that they could be an essential element of the study of an outbreak, as ODEs do not give the whole picture.

Overall, PEPA has been used to reproduce the results previously obtained by Webb et al. [92], and to show some flaws in the previous study, highlighting some of the issues that ODEs have when used for disease modelling. A few elements however could not be expressed exactly as the actual behaviour of the species. Indeed, the choices made in the original model from Webb et al. [92] for features such as flea to prairie dog transmission, or flea birth, could not be directly translated into PEPA, and an approximation had to be used. While it did not result in a significant difference in the resulting model, as seen in Figure 5.10, it might only be a consequence of the small impact of the flea infection and the short duration of the simulations.

Chapter 6

Introducing Timed Events to PEPA

Timed events correspond to events that happen at fixed times, either once (introduction of the vaccine), or regularly (circadian clock, seasonality, immigration). These are essential in many biological and epidemiological systems. Different types of interventions are essential to describe control measures such as vaccination, vector culling, or quarantine, as it is important to take into account the delay in applying such measures in the event of a disease outbreak. Seasonality is an essential feature to the behaviour of measles, and the dynamics changes completely when it is not included in the model, as the transmission rate is strongly influenced by the aggregation of children at school [9]. Immigration also has an influence on the dynamics of measles, and in particular provides new infectious individuals to a city where the disease had faded out, especially in the case of small and isolated cities.

This chapter will first present how to model intervention, both for a one-time event, such as the introduction of a vaccination or as a regular occurrence, such as immigration. Section 6.2 will then describe the particular cases of the circadian clock and seasonality. Finally, the methods presented will be used to present how control policies can be described, and the model describing the occurrence of measles in Leeds will be thoroughly investigated.

6.1 Intervention

PEPA's grammar does not allow timed events in a model. It is not possible to specify precisely when an action α allowed by a component type P should be fired, as the rate r at which the action happens follows the cumulative distribution function of the exponential distribution $F_\alpha(t)=1-e^{-rt}$ which only ensures that on average, α will be fired after $1/r$ time steps. It is, however, still possible to approximate the happening of a timed event by having a single agent fire the action either once or at regular intervals, depending on need:

- the action α has to be fired once: this corresponds to the case of a sudden change in the behaviour of the model, such as the introduction of a vaccine. The one agent in P, once firing the action α, goes to a state P' for the rest of the simulation.
- the action α has to be fired every t time steps: this corresponds to the case of immigration for example, where a new immigrant arrives every t days. The agent in P remains in the state P after having fired the action α. The rate here would be $1/t$.

Using this method to approximate an intervention does not completely solve the problem. While on average, the action will indeed be fired after $1/t$ time steps, the actual moment at which the intervention takes place varies. For example, an action firing at rate 1 has actually only a

38% probability of happening between 0.5 and 1.5 time steps. In order to reduce this variability, one solution is to split the action into several steps. In other words, an expression $(\alpha, r).P$ will be rewritten $(\alpha, r \times n).(\alpha, r \times n).....(\alpha, r \times n).P$ with n the number of steps, that we will denote $(\alpha, r \times n)^n.P$ for readability purposes. The distribution of the resulting action can be calculated using the following theorem [23]:

Theorem 6.1.1 *If X_1,X_2,...X_n are independent following an exponential distribution $Exp(\alpha)$ then $\sum_{i=1}^{n} X_i$ follows a Gamma distribution $Gamma(n, 1/\alpha)$*

As the rate of each of the activities is actually rn, the resulting probability density function is:

$$f_{n,r}(x) = \frac{x^{n-1}.e^{-rnx}.(rn)^n}{(n-1)!} \tag{6.1}$$

The cumulative density function can be simplified, in the special case where $n \in \mathbb{N}$ to the following expression:

$$F_{n,r}(x) = \sum_{i=n}^{\top} \frac{(rnx)^i}{i!} e^{-rnx} \tag{6.2}$$

This formula allows us to estimate the probability for an event to happen between time $t{=}a$ and $t{=}b$ (with $b > a$) as:

$$F_{n,r}(b) - F_{n,r}(a) = \sum_{i=n}^{\top} \frac{(rnb)^i}{i!} e^{-rnb} - \sum_{i=n}^{\top} \frac{(rna)^i}{i!} e^{-rna} \tag{6.3}$$

The methods presented here also present a major drawback: the ODEs that would be derived from the model are not accurate any more. Indeed, as explained in chapter 3, the approximation of the mean behaviour of PEPA models by ODEs is only valid when the number of agents is large enough. However, in the methods presented in this section, only one agent is in the subgroup

describing the timed event. The ODEs can be derived from the model using the Stirling Amendment, as presented in section 3.1.2, but the resulting set of equations will not describe the system accurately, as the hypothesis the ODEs derivation is based on is invalid. In the case of a single intervention in particular, it is however possible to describe the model using two sets of ODEs, corresponding to the behaviour of the model before and after the intervention. For the moment though, there is no algorithm for deriving the ODEs in such a way. The modeller has to either rely on her intuition, or write two PEPA models corresponding to the two situations, derive the ODEs using the Stirling Amendment, and then add what would be necessary to "link" both sets of ODEs.

This problem, however, is not specific to PEPA models. The same method would need to be applied to a model written directly using ODEs. Indeed, the issues lies in the capacity to switch behaviour instantaneously. In order to achieve this, a value, a series of values, or a function for the time of the switch need to be predefined. In each set of equations representing on of the states (before or after the intervention), an "link" statement allows to switch from one set to the other.

6.2 Cyclic behaviour

Two common time driven features in epidemiology and biological systems in general are the circadian clock and seasonality. Most organisms have a different behaviour in daylight and at night time, or depending the season. In disease modelling in particular, this can have an important impact on the dynamics of the system. The two features are very similar: both correspond to a cyclic change, whether it is two or four states cycle. The method presented in this section, based on the technique presented in the previous section, can be adapted to any number of states the cycle might have.

For the case of the circadian clock, for example, two component types need to be added to the model, each representing a state of the cycle: *Day* and *Night*. The change of states is ruled by the actions *go_day* and *go_night* for the transitions from night to day and from day to night respectively, at the rate r in both cases. As detailed in the previous section, in order to reduce the variability of the length of the cycles, both actions will have to be performed n times consecutively. The resulting component types are:

$$\begin{aligned} Day &\stackrel{def}{=} (go\_night, n \times r)^n.Night \\ Night &\stackrel{def}{=} (go\_day, n \times r)^n.Day \end{aligned}$$

Additional activities will be necessary in order to communicate with the rest of the model, so it can adjust its behaviour. The most likely case is when all agents in a particular state in a different subgroup need to go to another state which describes their new behaviour, as in the case of Inf_s' and Inf_w' in the measles model presented in Figure 6.6. In those cases, an additional activity *during_day* and *during_night* are added to the component types, as well as to the component types affected by the change in the cycle. If C_{day} and C_{night} are the two component types representing the behaviour of the agents during the day and the night respectively, the resulting model would be:

$$\begin{aligned} Day &\stackrel{def}{=} (go\_night, n \times r)^n.Night + (during\_day, big).Day \\ Night &\stackrel{def}{=} (go\_day, n \times r)^n.Day + (during\_night, big).Night \\ C_{day} &\stackrel{def}{=} (during\_night, \top).C_{night} \\ C_{night} &\stackrel{def}{=} (during\_day, \top).C_{day} \end{aligned}$$

Here again, equation (6.3) can be used to evaluate the probability of the duration of a cycle to remain within certain boundaries. The restriction on ODEs also applies in this case, as once again, the assumption that the number of agents is large is not fulfilled in the subgroup representing the cycle. The method which consists in writing a set of ODEs for each state of the cycle, and

then making sure the switch from one behaviour to the other happens at the expected time, can however also be used in this situation.

Now that the methods incorporating timed events to PEPA models are available, a series of examples will be presented in the next sections. First, three different control policies, which are heavily relying on timed-events, will be modelled. Then, the model of measles in Leeds will be thoroughly investigated, with seasonality and immigration playing a central role in the dynamics of the disease.

6.3 Control policies

Control policies are an essential part of the epidemiologist's toolbox, that allows her to determine the best way to react in the case of an outbreak, or the best preventive measure to take. However, the description of control policies usually requires a sudden change in the behaviour of the model. In the case of culling, the death rate of the vector increases dramatically when the culling takes place. For vaccination, susceptible individuals move the the recovered/immune class massively when it happens. Finally for quarantine, the contact rate decreases massively for some infected individuals when the policy is enforced.

6.3.1 Vector culling

The first model, shown in Figure 6.1 corresponds to a vector-borne disease, such as malaria [36]. The context corresponds to an outbreak of the disease, where as soon as it is detected, a vector culling measure is enforced. The model itself is a simple vector-borne transmission SEIR model.

There is no birth nor death, which means the population remains constant. The recovered individuals do not lose their immunity, which means the initial pool of susceptibles only decreases in size over time. The vector can be either healthy or infected. The infectious vector can however recover from the disease and come back to a healthy state. It cannot die naturally or give birth either. The vector can only die when the culling is performed. In the model, only the successful contacts from vector to host are represented by the action *contact*.

The culling is divided into three steps, each of them with a corresponding component type. First, *BeforeCulling* corresponds to the situation before the culling takes place. The amount of time before it happens depends on $(\mathit{firstculling}, \mathit{fcr})^n$. The period before the culling starts is n/fcr time steps on average. Then the agent responsible for the culling goes to the state *Culling*. What happens then is that the action *culling* is fired in quick succession. The agents in *InfVector* and *HealthyVector* cooperate over the action *culling* and move respectively to the state *VectorCullingI* and *VectorCullingH*, where there is a probability that they die. If they do so, they go to a state *DeadVector*, and stay there for the remaining of the simulation. Finally, when the culling is over, the agent in *Culling* fires the action *endofculling* and moves to the state *CullingOver*, while the agents in *InfVector* and *HealthyVector* come back respectively to the state *InfVector* and *HealthyVector*.

Figure 6.2 shows one 250 days simulation of the model presented above. In this case, the culling starts after five days, lasts for fifty days, and each living host has a probability of 25% of dying every day. The culling has in this case a major impact, as the peak of the infection is under 1000 infectious individuals, while it reaches over 6000 without the culling.

Host

$$S \stackrel{def}{=} (contact, \top).Exposed$$

$$Exposed \stackrel{def}{=} (infected, incubation\_rate).Inf$$

$$Inf \stackrel{def}{=} (infect\_vector, inf\_vector\_rate).Inf + (contact, \top).Inf + (recover, recover\_rate).R$$

$$R \stackrel{def}{=} (contact, \top).R$$

Vector

$$HealthyVector \stackrel{def}{=} (infect\_vector, \top).InfVector + (culling, \top).VectorCullingH$$

$$InfVector \stackrel{def}{=} (infect\_vector, \top).InfVector + (contact, contact\_rate).InfVector + (recovery\_vector, recovery\_rate).HealthyVector + (culling, \top).VectorCullingI$$

$$VectorCullingH \stackrel{def}{=} (killed, killed\_prob).DeadVector + (infect\_vector, \top).VectorCullingI$$

$$VectorCullingI \stackrel{def}{=} (killed, killed\_prob).DeadVector + (contact, contact\_rate).VectorCullingI + (recovery\_vector, recovery\_rate).VectorCullingH$$

$$DeadVector \stackrel{def}{=} (stay, 1.0).DeadVector$$

Culling

$$BeforeCulling \stackrel{def}{=} (before\_culling, fctr)^n.Culling$$

$$Culling \stackrel{def}{=} (culling, big).Culling + (endofculling, 0.02).CullingOver$$

$$CullingOver \stackrel{def}{=} (stay, 1.0).CullingOver$$

System equation

$$S[9900] \parallel Inf[100] \underset{contact, infect\_vector}{\bowtie} HealthyVector[100000] \underset{culling}{\bowtie} BeforeCulling[1]$$

Figure 6.1: Example presenting how culling can be implemented.

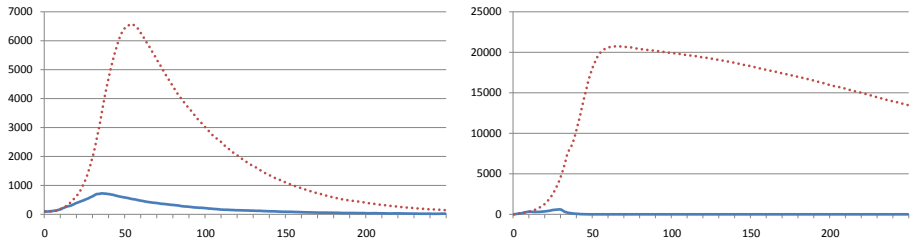

Figure 6.2: Comparison between two simulations where the vector has been culled (solid line) or not (dotted line). The graph on the right corresponds to the infectious host, the one on the left the infectious vector.

6.3.2 Vaccination

This model, presented in Figure 6.3 is a simple indirect transmission SEIRS model with yearly vaccination. In this case, a percentage of the susceptible individuals are vaccinated every year, and moved to the recovered category.

The principle behind vaccination is similar to the one used for culling. The agent is in the state *No_vaccination* until the action *vaccine_starting* is fired n times, which takes *wait_period* on average, and results in the agent moving to the state *Vaccination*. The action *vaccine* is then fired in quick succession. All the agents in S cooperate over this action and move to the state *SVaccine*. Each agent in *SVaccine* has then a probability *vaccined_prob* to be vaccinated, which results in it being immune to the disease. It can be noticed that the susceptible agents can still be contacted when in a *SVaccine* state, so the vaccination process does not interrupt the dynamics of the disease. If an agent is successfully vaccinated, it becomes recovered and immune to the disease.

Figure 6.4 shows an example of how vaccination could decrease the number of infectious individuals in the case of an outbreak. In this case, the vaccination starts at day 70 of the start of the simulation and half the susceptible individuals are vaccinated. While the number of infectious

Original subgroup

$$S \stackrel{def}{=} (contact, \top).Exp + (vaccine, big).SVaccine$$

$$SVaccine \stackrel{def}{=} (contact, \top).Exp + (vaccined, vaccined\_prob).R$$
$$+(not\_vaccined, (1 - vaccined\_prob)).S$$

$$Exp \stackrel{def}{=} (infected, incubation\_rate).Inf$$

$$Inf \stackrel{def}{=} (contact, \top).Inf + (recover, recover\_rate).R$$

$$R \stackrel{def}{=} (contact, \top).R + (lose\_immunity, lose\_immun\_rate).S$$

Mirror subgroup

$$Transmitter \stackrel{def}{=} (contact, contact\_rate).Transmitter + (recover, \top).Dormant$$

$$Dormant \stackrel{def}{=} (infected, \top).Transmitter$$

Vaccination

$$No\_vaccination \stackrel{def}{=} (vaccine\_starting, n/wait\_period)^n.Vaccination$$

$$Vaccination \stackrel{def}{=} (endVaccine, 1/vaccine\_period).No\_vaccination$$
$$+(vaccine, \top).Vaccination$$

System equation

$$(S[990] \parallel Inf[10] \underset{infected, contact, recover}{\bowtie} Transmitter[10] \parallel Dormant[990])$$
$$\underset{vaccine}{\bowtie} No\_vaccination[1]$$

Figure 6.3: Example presenting how vaccination can be implemented.

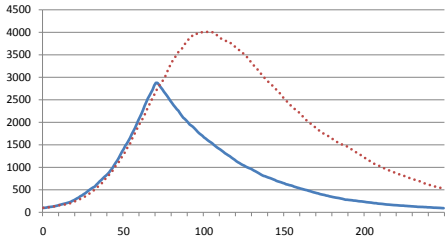

Figure 6.4: Comparison between two simulations where the susceptibles are being vaccinated (solid line) or not (dotted line).

individuals continues to increase in the case where the vaccination is not performed, it immediately drops in the case where susceptibles are vaccinated.

6.3.3 Quarantine

Finally, this model corresponds again to a simple indirect transmission SEIRS model. A number of days after the outbreak has started, identified infected individuals are asked to remain at home, decreasing their contact rate dramatically.

Original subgroup

$$S \stackrel{def}{=} (contact, \top).Exp$$

$$Exp \stackrel{def}{=} (infected, incubation\_rate).InfProv$$

$$InfProv \stackrel{def}{=} (quarantined, quar\_prob * big).Inf\_temp$$

$$+(not\_quarantined, (1 - quar\_prob) * big).Inf$$

$$Inf\_temp \stackrel{def}{=} (infQ, big).Inf$$

$$Inf \stackrel{def}{=} (contact, \top).Inf + (recover, \top).R$$

$$R \stackrel{def}{=} (contact, \top).R + (lose\_immunity, lose\_immun\_rate).S$$

Mirror subgroup

$$Transmitter \stackrel{def}{=} (contact, contact\_rate).Transmitter + (recover, recover\_rate).Dormant$$

$$TransmitterQ \stackrel{def}{=} (contact, contact\_rate\_Q).TransmitterQ + (recover, recover\_rate).Dormant$$

$$Dormant \stackrel{def}{=} (not\_quarantined, \top).Transmitter + (infQ, \top).TransmitterQ$$

Quarantine

$$No\_quarantine \stackrel{def}{=} (quarantine\_starting, n/waitperiod)^n.Quarantine$$

$$Quarantine \stackrel{def}{=} (quarantined, 50000.0).Quarantine$$

System equation

$$(S[9900] \parallel Inf[100] \underset{not\_quarantined, contact, recover, infQ}{\bowtie} Transmitter[100] \parallel Dormant[9900])$$

$$\underset{quarantine\_started}{\bowtie} No\_quarantine[1]$$

The mechanism behind the quarantine subgroup is fairly simple. First, the action *quarantine_starting* must be fired n times. The total period before the quarantine starts is *wait_period* time steps, as explained in section 6.1. Once the quarantine has started, and the agent in *No_quarantine* has moved to the state *Quarantine*, the action *quarantine_started* is passively available for cooperation. A few changes were also made to the subgroup describing the behaviour of the population. When the incubation period ends, the agent in *Exp* moves to a state

InfProv. The objective of this component type is to determine whether the infected individual will be quarantined or not. Two actions are available: *quarantined* and *not_quarantined*. The first action must cooperate with the agent in *Quarantine*. This ensures that an individual can only be quarantined if the quarantine has actually started. The rate at which the action takes place is $quar\_prob * big$ to ensure this is performed quickly and the delay is negligible compared to the other actions of the model. *quar_prob* corresponds to the probability to be quarantined, and here again, the actual rate is $quar\_prob * big$. Once the action *quarantine_started* has been performed, the action *infQ* is then fired at rate *big*, which has the role of cooperation with an agent in *Dormant* to make sure an agent is moved to the expected component type. The action *not_quarantined* plays a similar role. However, as it is not required to cooperate with an agent in *Quarantine*, the action *not_quarantined* can directly cooperate with an agent in *Dormant*. Depending on which action is fired, the agent in *Dormant* moves to either *Transmitter* or *TransmitterQ*. The two component types have exactly the same behaviour, the only difference lying in the rate at which the contact happens. It can be noticed that the action *recover* is active in the mirror subgroup. This is to ensure the correct component, *Transmitter* or *TransmitterQ*, is updated. Indeed, if it was active in the original subgroup, the information about whether a quarantined or not quarantined infectious individual is recovering is not available, and it would randomly choose from *Transmitter* or *TransmitterQ*. By having *recover* active in the mirror subgroup, the choice is avoided.

Figure 6.5 shows a simulation of the model presented in this section, when the quarantine is and is not active. In this case, the quarantine starts 6 days after the start of the outbreak and the contact rate is divided by 80 when an infectious individual is quarantined. In the case of the solid line, once the quarantine has started, 100% of newly infectious individuals are quarantined, while in the case of the dashed line, only 90% of them are quarantined. We can see there the influence of this parameter as a mere 10% difference completely changes the behaviour of the disease.

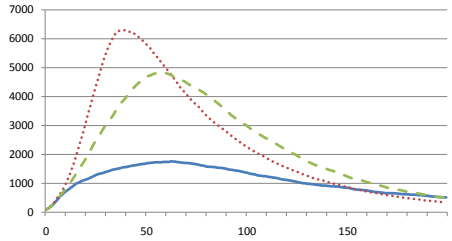

Figure 6.5: Comparison between two simulations where the infectious can be efficiently quarantined (solid line), less efficiently quarantined (dashed line) or not (dotted line).

6.4 Measles in Leeds between 1944 to 1964

The last tools needed to build a model of measles are now available. The model is a representation of the city of Leeds from 1944 to 1964. The city initially has a population of 508,010, ten of whom are assumed to be infectious. In this section, the model will be described, and simulations performed. A comparison of the simulations and the data available on the University of Cambridge website [89] will be performed.

6.4.1 Presentation of the model

The model presented in Figure 6.6 is based on a paper from Bjørnstad et al. [9], apart from the immigration mechanism and parameters, which are inspired by a paper from Finkenstädt et al. [29]. Measles has one route of transmission: it transmits from one individual to another via direct frequency dependent contact only. A few additional features are important. Births and deaths are described in a very simple manner: the birth rate and the death rate share the same value. As the disease is not usually deadly, and the number of immigrants over the twenty years is low compared to the total population, this will ensure that the population size remains constant. All the living agents S,Exp, Inf and R can give birth and die naturally.

$$
\begin{aligned}
Summer &\stackrel{def}{=} (go\_winter, srs)^p.Winter + (insummer, big).Summer \\
Winter &\stackrel{def}{=} (go\_summer, srw)^p.Summer + (inwinter, big).Winter \\
S &\stackrel{def}{=} (contact, \top).Exp + (birth, br).S + (die, dr).Ghost \\
Exp &\stackrel{def}{=} (contact, \top).Exp + (incubation, ir).Inf + (die, dr).Ghost + (birth, br).Exp \\
Inf &\stackrel{def}{=} (contact, \top).Inf + (recover, rr).R + (dieI, dr).Ghost + (birth, br).Inf \\
R &\stackrel{def}{=} (contact, \top).R + (birth, br).R + (die, dr).Ghost \\
Ghost &\stackrel{def}{=} (born, big).S + (immigration, \top).Immi \\
Immi &\stackrel{def}{=} (gotoInf, big).Inf \\
S' &\stackrel{def}{=} (incubation, \top).Inf'_s + (birth, \top).DS + (gotoInf, \top).Inf'_s \\
Inf'_s &\stackrel{def}{=} (contact, crs).Inf'_s + (recover, \top).S' \\
&\quad +(inwinter, \top).Inf'_w + (dieI, \top).S' \\
Inf'_w &\stackrel{def}{=} (contact, crw).Inf'_w + (recover, \top).S' \\
&\quad +(insummer, \top).Inf'_s + (dieI, \top).S' \\
DS &\stackrel{def}{=} (born, \top).S' + (timeout, 100.0).S' \\
Immigration &\stackrel{def}{=} (immigration, imrate).Immigration \\
((S[508000] &\parallel Inf[10] \parallel Ghost[100000] \\
&\underset{born, birth, incubation, recover, contact, dieI, gotoInf}{\bowtie} \\
S'[608000] &\parallel Inf'_w[10]) \underset{immigration}{\bowtie} Immigration) \underset{insummer, inwinter}{\bowtie} Winter
\end{aligned}
$$

Figure 6.6: The final measles model for Leeds.

Immigration and seasonality have been implemented using the method described in the previous section. Immigration is represented by a subgroup formed by a single component type, *Immigration*. The one instantiation of the component type fires the action *immigration* every $1/imrate$ time step. Each time this happens, one infectious immigrant arrives in the city, which translates in the model by one agent in *Ghost* moving to the *Inf* state. The reason why only infectious immigrants are represented here is linked to the influence of immigration. Susceptible immigrants and emigrants do not have an influence on the dynamics of the disease. As their numbers are considered equal, they do not have any impact on the evolution of the model. Infectious emigrants also do not have any influence on the dynamics of the disease in the studied city, and their number is sufficiently low to have a negligible impact on the population size. Infectious immigrants on the other hand, can start a new outbreak, if the timing is right. The model is composed of two seasons: a four month summer, and an eight month winter. The contact rate is the only difference between the two seasons. The reason for this difference lies in the difference in behaviour of the children. Indeed, because the disease is very infectious, and getting infected grants a lifelong immunity, the average age of the infected individual in the real world is low [65, Table II]. The average number of contacts a child makes change significantly depending on the period of the year, as more contacts are made when children go to school. For this purpose, the year has been divided into two seasons, summer and winter, that correspond to the children's holidays and school period respectively. The two seasons only have an impact on Inf', the replicated component of Inf. Inf' has been divided in Inf'_s for the summer, and Inf'_w for the winter. Seasonality is implemented as follows:

$$
\begin{aligned}
Summer &\stackrel{def}{=} (go\_winter, p/summer\_duration)^p.Winter + (insummer, big).Summer \\
Winter &\stackrel{def}{=} (go\_summer, p/winter\_duration)^p.Summer + (inwinter, big).Winter \\
Inf'_s &\stackrel{def}{=} (contact, crs).Summer + (inwinter, \top).Inf'_w \\
Inf'_w &\stackrel{def}{=} (contact, crw).Summer + (insummer, \top).Inf'_s
\end{aligned}
$$

In this sub-model, all the actions not related to or not modified by seasonality have been removed from the component types' expressions. In this extract of the measles model, once the season changes, from *Summer* to *Winter* for example, *Winter* cooperates with the agents in Inf_s' over the action *inwinter*, in order for them to move to Inf_w'. The rate at which the cooperation is performed has to be very big compared to the other parameters of the model, in order for the process to be considered instantaneous. In this model, this rate is at least $big/crs = 1.66 \times 10^8$ times bigger than any other rate in the model. The number of infectious individuals is always under 1000, so the whole operation takes less than $1000/big = 10^{-6}$ time step to be performed. This way, the contact rate used is indeed *crw*, which is the rate of contact over the winter season. After the action *go_summer* has been fired p times, the agent describing the season moves from *Winter* to *Summer*, and the agents in Inf_s' are updated again. The choice of the value of p entirely depends on the precision required by the modeller, as well as the processing time of the model.

6.4.2 The parameters

According to Bjørnstad et al. [9, p. 171], the critical community size, in order for the virus not to go extinct, appears to be between $300,000$ and $500,000$ in England and Wales. For this reason, the city of Leeds has been chosen to test this model. Indeed, its population in 1944 was about $508,000$ inhabitants. The proportion of susceptibles seems to vary from one study to another. Bjørnstad [9] mentions studies where the proportion of susceptibles varies from 3.5% [9, p. 180] to 9% [28]. This number, however, is not crucial. According to our experiments, the model evolves to a steady state proportion of susceptibles between 3.5 and 9%, regardless of initial proportion of susceptibles. For this reason, the model starts with 100% susceptible individuals, and evolves naturally towards its steady state susceptible proportion. In the graphs shown in this section, the transient section, which has been empirically determined to correspond to the first thousand steps,

Parameter	Rate (per day)	Description
n	508010	Total population size
big	999999999	The big rate used of immigration and seasonality
ir	1/7.5 [9]	Incubation rate
rr	1/6.5 [9]	Recover rate
crw	39.1/6.5 [9]	Winter contact rate
crs	19.8/6.5 [9]	Summer contact rate
br	0.017/365 [9]	Birth rate
dr	0.017/365 [9]	Death rate
$imrate$	$0.02 * \sqrt{n}/365$ [29]	Immigration rate of infectious individual
p	96	Number of iterations of the change of season action
srw	$1/(8 \times 30) \times p$	Rate of one iteration of the change of season action (winter)
srs	$1/(4 \times 30) \times p$	Rate of one iteration of the change of season action (summer)

Figure 6.7: The parameters used in the model in Figure 6.6.

has been removed from the results shown in the next section. The results obtained in this portion of the graph does not indeed have any basis on the real behaviour of the disease, and does not provide any insight on the disease behaviour.

The incubation and infectiousness period are respectively 6 to 9 days, and 6 to 7 days [9]. The average of those values has been chosen in order to derive the rate used in the model. The annual per capita birth rate ranges from 0.01 to 0.02 for the period between 1944 and 1964 [9]. The data on the city of Leeds provided by Grenfell [89] provides an average birth rate over this time period of 0.017 birth per year and inhabitant. In order to keep the population constant, the death rate has been chosen to have the same value. The only increase in population size results from the immigration of infectious individuals, which is proved to be negligible later in this section. The fact that the population remains constant throughout the study period does not have an important impact on the behaviour of the disease. Indeed, the simulation time is short compared to the lifespan of an individual, and every individual already infected gets lifelong immunity. This means that the adult population does not participate in disease transmission, as almost all of them are immune to the disease. Also, the number of births is the important number here: young children are the ones susceptible to the disease. The data shows that the number of births does not vary significantly during the studied period. Having a constant population and a constant birth rate ensures that the number of births does indeed remain constant during the time of the simulation. The value for immigration has been given in a paper from Finkenstädt et al [29, p. 755]. According to this paper, the number of infectious imports per year can be approached by the formula:

$$\text{average number of imports per year} = 0.02\sqrt{\text{population size}}$$

In the case of our model, it results in $0.02\sqrt{508000} = 14.25$ imports per year. This results in a total of 285 immigrants across the twenty years studied, who increase the overall population by 0.056%.

The contact rate has been chosen based on the value of R_0 given in the paper by Bjørnstad et

al. [9, p. 180]. R_0 is the number of successful contacts an infectious individual would make in an entirely susceptible population. According to this paper, its maximum value is 39.1 in December, and its minimum value is 19.8 in August. In the absence of an average value over the course of a season, or a monthly value, these values are used for the winter and summer season respectively. The daily number of contacts can then be derived from R_0 by dividing it by the number of days a sick person is infectious. Finally, the parameters related to the seasons are *srw*, *srs* and *p*. As explained in the previous section, in order to change the season from winter to summer, a series of actions $(go\_summer, p/winter\_duration)^p$ is needed. In this model, the winter season lasts eight months. If we consider that a month lasts 30 days [1] on average, the overall winter duration is 8×30 days, so the rate of change is $1/(8 \times 30)$. However, as the action *go_summer* needs to be fired p times, the resulting rate for each action is $p/(8 \times 30)$. The method is similar for the winter to summer action. Because the summer, in this model, lasts four months, the resulting rate is $p/(4 \times 30)$. Finally, the choice of the value of p lies in the hands of the modeller: it must be chosen in order to give an acceptable variability in the season length, while not increasing the length of the simulations too much. With p=96, and using equation (6.3), the probability that the winter lasts between 7 and 9 months, and that the summer lasts between 3.5 and 4.5 months is 78%, which is deemed sufficient for the scope of this study.

6.4.3 Results

The analysis of this model is performed through a series of single stochastic simulations. Indeed, due to the variability of the time of change of season, stochastic simulations cannot be averaged meaningfully. Also, as mentioned earlier, ODEs cannot be derived from the PEPA model, as the

[1] To simplify the way seasonality is modelled, a month has been decided to always last 30 days. This results in a year of 360 days in the simulations.

hypothesis behind the derivation, that the number of agents is large, is not met. However, given the size of the population studied here, the difference between different simulations is reduced. Moreover, the analysis will only be performed on quantitative factors, such as the length of the cycle between two consecutive outbreaks and the average size of the peak of each epidemic.

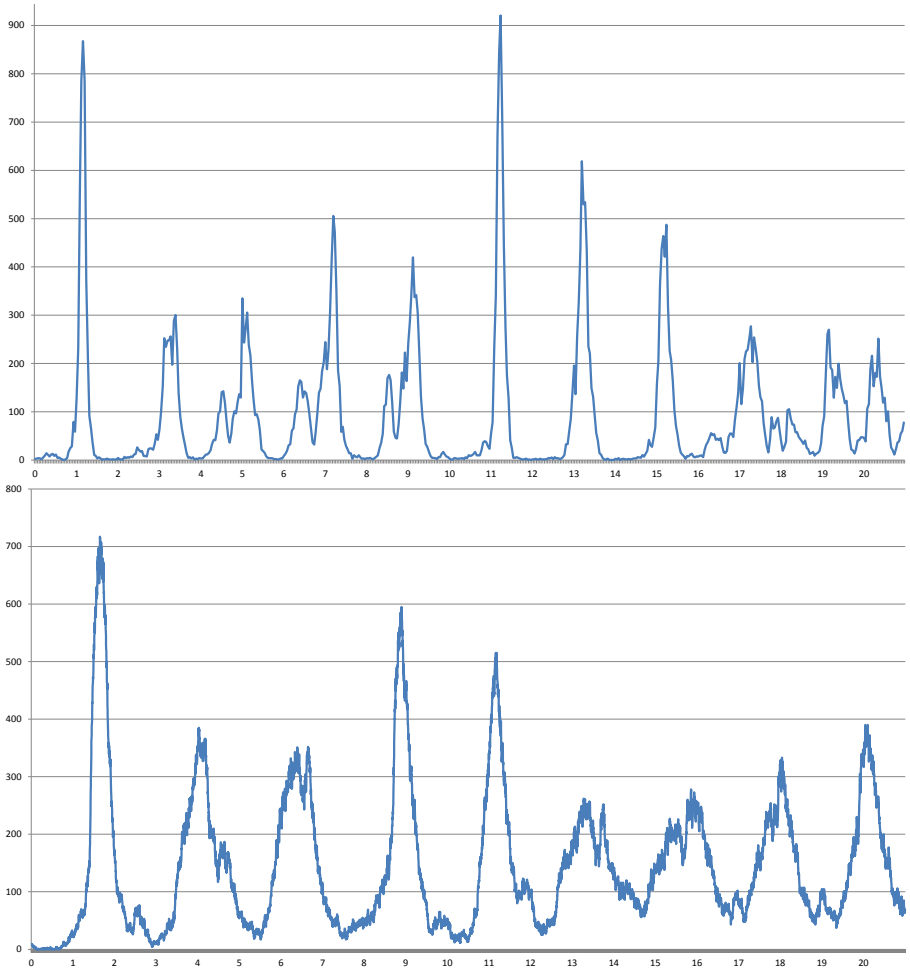

Figure 6.8: Comparison between the reconstructed and the number of infectious individuals in a single simulation in Leeds between 1944 and 1964 .

The data the model is compared to is accessible from the University of Cambridge's website [89]. In this section, only the data related to the city of Leeds will be used. Changes to the data are

first required to allow the analysis to be performed. First of all, the data of the number of reported cases has a time step of two weeks. In other words, each year is divided into 26 data points, each of the data points recording the number of cases in the previous 2 weeks. Two modifications have to be made. The first change is necessary to make sure the format of the data is compatible with the data obtained with the stochastic simulation of the PEPA model, in order for the two sets of data to be easily compared. It seemed more natural to translate the original data to the format of the result of the simulation, as it seemed more intuitive to record the number of infectious individuals at any given time. For that purpose, each data point in the original data has been divided by 14, in order to get the number of reported infectious individual per day during that period of time. It has then been multiplied by 6.5, to reflect the fact that each infected individual remains infectious for an average of 6.5 days. The second change concerns the reporting rate. Bjørnstad et al. [9, p. 172] mention a reporting rate across England and Wales of just over 50%. In this study, it is assumed that every individual is infected at some point in her life. Given the high contact rate, this is a fair assumption. This means that the number of births should correspond to the number of cases. In the case of Leeds, between 1944 and 1964 (inclusive), $181,539$ births were recorded, while $109,730$ cases of measles were reported. Assuming the birth rate was constant in the years just before the record, this gives an average reporting of 60.4%. A further assumption that the birth rate is constant throughout the twenty recorded years is needed. While this was not the case in reality (the maximum number of births recorded was 10821 in 1947, and the minimum was 7584 in 1954), the difference can be considered negligible for the scope of this analysis. Finally, a constant reporting rate will be assumed. This allows the number of infectious individuals at each data point to be corrected by multiplying the reported number of cases by the ratio between the total number of birth over the twenty years and the total number of reported cases. The graph in Figure 6.8 shows the corrected data.

The result of the simulation of the model described in the previous section is shown in Figure 6.8.

It corresponds to 8200 days of simulations, where the first 1000 days have been removed, as described previously. The remaining 8200 days correspond to the twenty years of simulations. The two graphs presented exhibit a comparable behaviour. In both cases, measles outbreaks occur on a regular cycle. The length of the cycle varies between the gathered data and the stochastic simulation. In the case of the data, the cycle lasts exactly two years, with very little variation over the length of the studied period, apart from a single one year cycle between the 1963 and the 1964 epidemics. The simulation on the other hand, has biennial cycles most of the time, but sometimes exhibits 2.5 years cycles. Across the studied period, the gathered data experiences 11 outbreaks, for 9 in the simulation. As detailed by Bjørnstad et al. [9], the cycles are a consequence of the addition of seasonality to the model. The presence of a summer season where the contact rate is lower allows more time for the pool of susceptibles to increase in size, until the contact rate increases again in the winter. The number of infectious individuals at the peak of infection in both cases can vary a lot, between 260 and 920 in the case of the gathered data, and 260 and 710 in the simulation. The average of the number of individuals at the peak of each outbreak confirms this difference: 479 infecteds in the case of the gathered data, for 424 in the case of the simulations. Finally, the graph of the gathered data seems to experience less local variability than the one representing the simulation data. This is however only an artefact of the reconstruction technique applied to the data.

6.4.4 Conclusion

For the first time in this study, a PEPA model has been compared to actual gathered data. The results are promising: many similarities can be noticed between the simulations and the actual data. The difference can have several origins. Some are tied to some of PEPA's limitations. As the rates cannot change during the simulation, the birth and death rates have the same value

throughout the simulation. Even if the value does not seem to vary vastly according to the data, it might have an influence on the results. A second limitation of PEPA might have a bigger impact on the simulation result. Seasonality has been approximated by splitting the year into two seasons each with a single contact rate which does not change throughout the season. The value used by Bjørnstad et al. [9, Fig. 7, p. 178] varies noticeably within each season. Since the results are very sensitive to the contact rate's value, this might affect the conclusions of the PEPA model. Some additional explanations might have a link with insufficient biological knowledge of the disease. First, some of the parameter values given by Bjørnstad et al. [9] and used in this model seem to have a degree of uncertainty. Indeed, different papers give different values to the parameters. The paper from Bolker et al. [11] report an incubation period of ten days, and infectious period of 3.7 days, while for Bjørnstad et al. [9], the values are actually 7.5 and 6.5 days respectively. The approximation made regarding the reporting rate might also be sometimes excessive. However, while this might impact on the actual number of infectious individuals at any time step, it would not have an influence on the duration of the cycles. Finally, the influence of the nearby cities have been completely ignored in this study. This will be analysed in the next chapter.

6.5 Summary

This chapter has first presented techniques to approximate different timed events features, such as interventions, seasonality and the circadian clock. While the methods do not give an exact solution to the problem, the modeller has control over the precision of this approximation. Three different control policies, each offering a different challenge, have then been presented. Finally, the model describing the behaviour of measles in Leeds has been presented and analysed. While the results do not fit the data exactly, the model showed promising similarities with the expected behaviour.

In the next chapter, the possibility of incorporating compartments in a PEPA model will be explored and added to the models of the bubonic plague (chapter 5) and measles (this chapter).

Chapter 7

Modelling structured populations

The capacity to sort the population in different categories can be essential in describing certain diseases. For AIDS, age plays an essential role in the sexual behaviour of individuals. Indeed, young adults usually have more sexual contacts than elderly people, while children do not have any. In the case of measles, the interaction between cities has an important impact on the overall behaviour of the disease. Smaller cities usually experience fade outs of the disease, but still experience regular outbreaks, often within weeks of neighbouring cities. The actual structuring category can be diverse: age, social, hobbies, space, etc... depending on which feature has an important influence in the behaviour of the disease.

This chapter will present how this can be done in PEPA. In particular, syntactic sugar on top of PEPA has been developed in order to be able to concisely describe situations where the modelled population can be divided into different categories. Software that allows the translation of the model based on this syntactic sugar into a valid PEPA model will then be presented. Finally, the models of the bubonic plague in prairie dogs and measles in England and Wales will be expanded

to describe the behaviour and interactions of a few cities, along with an analysis of the problems encountered while modelling these cases.

7.1 PEPA and structured populations

Two approaches have already been made to incorporate structured populations in PEPA. PEPA nets [35] are a modification of PEPA's grammar to incorporate compartments. Inside each compartment, the components have the same behaviour as a PEPA component. However, a second type of change of state is allowed. In addition to the standard transitions, *firings* of the net correspond to moving an agent from one compartment to another. PEPA nets are based on Petri nets [74], which have shown their efficacy in different fields. However, PEPA nets suffer from a handicapping drawback: the formalism imposes severe limitations on communication [35]. Indeed, shared activities between components in different compartments are not allowed and two components in one place cannot cooperate and transfer to another place in a single atomic action. While the second limitation can be easily bypassed by including a temporary component type in the initial compartment, the first one can prevent some systems from being fully modelled. A situation where two actions need to compete, one of them being shared with a component in a different compartment, cannot be described using PEPA nets. However, this may arise in epidemiology: for example, when seasonality is included in the model. Indeed, as shown in Chapter 6, each season has two actions competing: one regulating the change of season (*go_winter* in the case of the *Summer*), the other one communicating with the component types in different subgroups in order to make sure they behave in the correct way (*insummer*). Because the season must be the same in all the subgroups, it is essential that only one agent of the *Winter* and *Summer* component types are be introduced in the model. This agent needs to communicate with agents in different

subgroups, in order to make sure the behaviours that depend on the season perform the right set of actions and use the correct rate. This type of behaviour is not permitted by PEPA nets.

The second approach to incorporate structured populations in PEPA is proposed by Galpin [32]. This solution is meant to be a more general approach to introducing compartments to process algebras in general, but PEPA is used as the case study. The objective of the formalism is to add named compartments and labelled transitions to the grammar of the process algebra. The approach is promising, and could lead to some interesting form of analysis for epidemiology. For the moment though, it is still at a very early stage of development. It is not supported by any tool, making any analysis on a model very tedious, especially as the number of compartments grows.

In order to overcome the absence of formalisms adapted to spatial epidemiological models, two approaches were considered. A new formalism could have been developed which would include the features required. This is the approach chosen by Galpin [32]. In this work, we have chosen a more pragmatic approach. PEPA already benefits from existing software which allows the analysis of a given model to be performed. Instead of developing a whole new formalism, we have decided to describe a new grammar, which would allow compartments, but the model would then have to be converted into a valid PEPA model for any analysis to be performed. This grammar, referred to as CPEPA, consists of a syntactic sugar on top of PEPA rather than a new formalism, as it is only a way of expressing some PEPA concept in a more compact way. In order to avoid the tedious conversion operation being performed by the modeller, the PEPAdum software initially presented in section 3.4 has been amended to allow this task to be automated, as described in section 7.2.5.

7.2 Grammar

In order to describe structured populations in PEPA models, a grammar that would reduce the size of the model written has been conceived. Often, groups of individuals in different locations will

have a very similar behaviour: minor differences might result from difference in actions available and parameters. In the case of disease modelling for example, the population might be divided into susceptible, infectious, recovered in every location, but the contact rate might be different in some location, or people might be quarantined in some places only. For this purpose, the grammar described here allows the modeller to describe all the similar subgroups in different compartments as a single subgroup. She can then specify which actions/rates are identical or different from a subgroup to another.

7.2.1 General overview

The syntactic sugar defines a general grammar which has a very similar format compared to a PEPA model. At the top of the file, it needs to define the names of the different compartments. The grammar of the new format is defined as detailed in Figure 7.1. Similarly to section 1.1.2, S denotes a *sequential component* and P denotes a *model component* which executes in parallel. C stands for a constant which denotes either a sequential component or a model component. C_S stands for constants which denote sequential components. T_a and T_s correspond to the valid expressions of the list of locations. $C_S@T_s$ is the expression of a sequential component, but with information about the location attached to it. β is a strictly positive integer representing the number of agents in a particular state initially, and r is a strictly positive real representing the rate. $f(\#i)$ corresponds to a function of $\#i$, which determines the location to be used (more detailed are given in section 7.2.3). Its expression is detailed in section 7.2.3. Finally, there is one limitation on the expression of $G \underset{L}{\bowtie} G$, G a model component. When two model components G_1 and G_2 cooperate over a non-empty cooperation set, the intersection of the union of the subgroups of the sequential components of G_1 and G_2 must be empty. More details are provided in section 7.2.4.

$$
\begin{aligned}
S &::= S_c \mid C_S@T_s \\
S_c &::= (\alpha, r).S \mid S + S \\
C_S@T_s &::= (\alpha@T_a, r@T_a).C_S@T_a \mid [i](a@T_a, r@T_a).C_S@T_a \mid (\alpha, r).S \mid C_S@T_s + C_S@T_s \\
\\
T_a &::= list \mid [a, b] \mid f(\#i) \\
T_s &::= list \mid [a, b] \mid ALL \\
\\
P &::= G \underset{L}{\bowtie} G \mid G \\
G &::= G_l@T_g < L > \mid G_c \\
\\
G_c &::= G_c \bowtie G_c \\
G_l &::= G_t \bowtie G_t \mid G_t \\
G_t &::= C[\{\beta_a\}] \mid C[\beta] \\
\beta_a &::= \beta_a, \beta \mid \beta \\
\\
L &::= L, action \mid action \\
action &::= name \mid name@[a, b] \mid name@list \\
T_g &::= list \mid [a, b] \\
list &::= list, location \mid location
\end{aligned}
$$

Figure 7.1: Grammar of the syntactic sugar allowing space for a PEPA model.

Each component type can exist in several locations. The actions and rates can also have locations attached to them. In the case of the actions, it is important to have the option to define them with or without a location. Locations are required in the case where an agent from a single location cooperates with an agent in a different location or in a different subgroup altogether. In this case, the location of the action is only used to differentiate different actions: only one location is defined even if agents from two different compartments are involved. Moreover, the choice of the location is in the hands of the modeller. This means that a cooperation between an agent in location A and an agent location B may happen over an action which is defined to take place in location A, B or even a different location C. Actions without specific locations are also essential, for example in the case where an agent in each location needs to be involved in the cooperation, or in the case where agents in every location are competing over the same action.

7.2.2 Definition of the compartments

A PEPA model can be divided into 3 parts: the definition of the variables, definition of the sequential components (or component types) and definition of the model component (or system equation). A fourth one is added here. At the start of the model, the names of the different compartments are defined in the form $\#compartments{=}L$, with L expressed in one of the two following ways:

- $\{name\_0, ..., name\_(n-1)\}$; : The compartments are described as a list of names. This will be preferred in the case where locations names are meaningful, such as cities.
- $[start, stop]$; with *start* and *stop* positive natural numbers: This time, the compartments are only described with numbers, going from *start* to *stop* included, incremented by 1. This case will be chosen in the case of a large number of locations (in order to avoid having to name all of them), or if the locations do not have any meaningful name.

In the next sections, the locations are in some cases referred to as numbers. In the case of a list of names of locations, the indices of the locations go from 0 to $n-1$, while if the locations are described using the second option, the indices are the numbers used for each location.

7.2.3 Definition of the component types

While the definition of variables remains the same as in the original definition of PEPA, the sequential components grammar has been heavily modified to be able to describe different population groups. The original PEPA plugin grammar for a sequential component is:

$$\begin{aligned} S &::= Action + Action \mid Action \\ Action &::= (\alpha, r).S \end{aligned}$$

The grammar of all the terms in this definition is changed. For the component types, compartment can be defined in three ways: a list of names of compartments, a pair of numbers designating a range of compartments, or using the reserved word ALL as follows:

- ComponentName@{l1, l2,..., ln}
- ComponentName@[a,b]
- ComponentName@ALL

The list of compartments must contains names already specified in the list L.

In the definition of a component type that has a different expression depending on the compartment, each action that it can perform can also be different depending on the compartment. In the case where the action is only allowed in one of the compartments, adding $[a]$ before its definition

expresses the fact that the action only appears in the expression of the component type in the compartment at the index a in the list of compartments attached to the component type. The action name then has different options to express the compartment. Every other action is assumed to appear in the expression of the component type in every compartment, with possible differences in each of the components of the action (action name, rate, target component) depending on the compartment.

Before detailing the different methods to express the compartment in the action name, an explanation on the meaning of the compartment of the action name is necessary. Indeed, an action named *action_at_l1* in the derived PEPA model does not mean that the action only happens between individuals in *l1*. This name is only a convention that implies that one of the individuals firing the action is in *l1*. The grammar does not prevent two individuals respectively in *l2* and *l3* from communicating over the action *action_at_l1*. The modeller is the one responsible for making sure the naming remains coherent. Also, a series of compartments is attached to each action of the definition of the component type. The order of the compartments in this list indicates which action (and compartment) is linked to which component type (and compartment). For example, if the component type C has the compartments $\{l1, l2, l3\}$ and the action a has $\{l2, l1, l3\}$, the resulting model will have a_at_l2, a_at_l1 and a_at_l3 as a respective possible action for C_at_l1, C_at_l2 and C_at_l3.

An activity of a component type can either apply to every location, or for a single location only:

- $(action, rate).targetComponent$: this syntax indicates that the expression of the activity applies to all the compartments defined in the component type definition
- $[j](action, rate).targetComponent$: this syntax indicates that the expression of the activity only applies to the j^{th} compartment of the component type list of locations. In this case,

a single action (with or without compartment defined), a single rate and a single target component are required.

The first method to indicate the compartments attached to the action consists of a list of names of compartments. It is also possible to define the compartments with an expression $[a, b]$, which is equivalent to a list of compartments between the one at index a and the one at index b in the list L. The reason the definition of the compartment given in the first section of the file has been used rather than the component type's one is that it is possible to have a compartment for the action name not included in the list of compartments defined in the component type. The third method is to write a function $f(\#i)$ that determines which compartment to use, where $\#i$ is the variable of the function f. Only the four arithmetic operations $+$, $-$, $\times$ and $/$ can be used, as well as parentheses. In other words, in order to express the fact that we want the next location (as detailed next), the modeller would ask for the action $\#i + 1$. The value of $\#i$ depends on how the compartments on the component type are defined. If the compartments of the component type are defined as a list, or with the reserved word ALL, then the j^{th} value of $\#i$ (with $j \in [0, n-1]$ with n the number of possible compartments for the component type) will take the value of the index in the original subgroup of the j^{th} compartment in the list of possible compartments for the component type. If the component type is described as $ComponentName@[a, b]$, then the value of $\#i$ will vary from a to b and for each $j \in [a, b]$, the compartment of the action allowed by the component at the j^{th} compartment in L will be the $f(j)^{th}$ in L when translated to PEPA. If the number obtained is bigger than the number of compartments in L or negative, the positive value of the modulo of $f(j)$ and the size of L is used.

Even though it does not necessarily indicate that an action happens in a specific compartment or group, specifying a compartment appears to be useful in practice. This is particularly useful when two component types communicate only when in the same compartment. For example, in

the case of vector-borne transmission, the vector can only contact the host when they are in the same compartment. This can be easily modelled by adding the notion of space to the action they communicate over. This is also useful in the case of a communication between two compartments, in order to have a unique action name between compartment a and compartment b. This might happen in the case of age groups, where individuals in different groups have to communicate over an action (in other words, they are in different subgroups). In that case, for an individual to move from one group to another, it needs to communicate with a *Ghost* in the next group so a new individual in that group is added, while the one on the lower age group is removed (and goes to the *Ghost* state). In order to simply and concisely allow this, a function can be used to define the compartment of the action depending on the component rate. For example, in the case of n age groups a_1, a_2 ... a_n, a component $C\_at\_a_j$ in a_j needs to communicate with $C\_at\_a_{j+1}$ over the action $grow\_at\_a_j$, for j between 1 and $n-1$. In that case, for every component $C\_at\_a_j$ at the compartment a_j, $j \in [2, n-1]$[1], two actions are required: $grow\_at\_a_j$, which corresponds to individuals growing from a_{j-1} to a_j and $grow\_at\_a_{j+1}$ which corresponds to individuals growing from a_j to a_{j+1}. The first action can be generally described as $grow@(\#i)$ and the second one as $grow@(\#i+1)$. In this particular case though, it can also be expressed as $grow@[2, n-1]$ and $grow@[3, n]$ respectively.

The rate can also be different depending on the compartment with which an action is associated. It can be expressed in two ways. First, a list of strictly real numbers or operations over real numbers (the result must be a real strictly positive number) indicate that for each number on the list, the i^{th} number of this list needs to be associated with the i^{th} action. The second method is a function $f(\#i)$. The $\#i$ corresponds to the index in L of the i^{th} compartment of the component type. In both cases, only the four arithmetic operations $+$, $-$, $\times$ and $/$ can be used.

[1]The cases of $j = 1$ and $j = n$ would need to be treated separately, as only one of the two actions is required.

Finally, the target component also contains information about the compartment. Here, it can be expressed in three ways.

- List of compartments: the order of the compartments indicates to which compartment of the target component type defined here each of them corresponds.
- Function of $\#i$: where the j^{th} value of $\#i$ has the value of the index in L of the j^{th} compartment in the list of compartments in the component type defined here.
- A subset of L: in the form $start, stop$. This indicates that the list of compartments is the subset from the compartment index *start* until the compartment at index *stop* of L.

7.2.4 Definition of the model component

The PEPA grammar of the model component has also been altered to incorporate information about the compartment. Some restrictions have been applied, as some available expressions of the model component are not used in epidemiology. Indeed, the way communication between two component types in the same subgroup works renders it impractical in most cases, as it requires all the agents in that subgroup to communicate at the same time for the action to happen. In the case of epidemiological studies, which often involve large groups of individuals, this is not a common behaviour. Hiding has also been prohibited from the model components grammar considered here. This results in the following grammar for a model component:

$$P ::= P_1 \underset{S}{\bowtie} P_2 | K \tag{7.1}$$

where K denotes a constant which can be either a sequential component, or a model component, and P_1 and P_2 two model components that verify the following condition: with $C(P)$ the set of reachable sequential components in the model component P, $\forall C_1 \in C(P_1)$ and $\forall C_2 \in C(P_2)$, $C_1 \neq C_2$. This prohibits any communication between component types belonging to the same subgroup. This is enforced in order to make the underlying PEPA model valid for PEPAdum. This type of behaviour is also very rarely used in epidemiology and is more likely to be a mistake from the modeller.

Based on these restrictions on the grammar of the model component in PEPA, a new grammar allowing compartments has been defined. This new grammar only allows the model component to be written in such a way that it can be then translated into a valid PEPA model component. It is based on the restricted PEPA grammar, with two modifications adding the information about compartments. The first modification concerns cooperation set, called S in the grammar presented here. In order to indicate the compartments in which each action can be located, two options are available:

- *actionName*@[*start*,*stop*]: this translates to a list of actions *actionName_at_compartment* where the compartments are the ones between indices *start* and *stop* in the original list of compartments L.
- *actionName*@{list of compartment names}: this simply attaches each of the compartments to the action name.

The second modification concerns the model components composed of component types all belonging to the same subgroup. Because of the simplification made here, these component types cannot communicate with one another. However, component types in different compartments can still communicate, as long as they do not belong to the same subgroup. For example, let's consider

a model consisting of two components $C1$ and $C2$, each of which can be in a compartment a and b. $C1\_at\_a$ and $C2\_at\_a$ belong to the same subgroup, and so do $C1\_at\_b$ and $C2\_at\_b$. However, the components in a and b belong to different subgroups. A formulation is needed to express this situation. The allowed expressions to describe this are:

1. $[C_1[c_1]||C_2[c_2]||..||C_n[c_n]]$@{list of compartment names} <list of action names>
2. or $[C_1[c_1]||C_2[c_2]||..||C_n[c_n]]$@$[start, stop]$ <list of action names>

The list of action names corresponds to the cooperation set between the subgroups located in different compartments.

The first block of this expressions is $[C_1[c_1]||C_2[c_2]||..||C_n[c_n]]$. This corresponds to the initial number of agents in each component type. However, for each component C_i, c_i needs to contain the information about the initial number of agents in C_i at each possible compartments. This can be done in two ways. The first way is a simple arithmetic expression. This implies that the initial value for this component type is the same regardless of the compartment. The second option is to give a list of values $\{a, b, ...\}$, the number of value having to correspond to the number of compartments allowed for this component type.

The second block can be written in two ways, but both options correspond to a list of compartments. In the first case, where the list is defined: {list of compartment names}, the list of compartments is given explicitly. In the second case $[start, stop]$, two numbers are given, *start* and *stop*, which correspond to the sub-list of L starting at index *start* and stopping at index *stop*. This list of compartments will then be attached to each of the component types specified in the previous block.

The final block follows the expression <list of action names>. The grammar of this expression is exactly the same as the one for the set of actions previously called S over which the model components communicate. This expression corresponds to the actions over which the component types in different compartments communicate.

To illustrate how the grammar works, consider the following example containing the initial list of compartments and the different options for the model component:

$$\#compartments = \{l1, l2, l3\};$$
$$([C1[\{98, 0, 2\}] \parallel C2[\{0, 50, 0\}]]@[0, 2] < action1@[0, 1] >)$$
$$< action2@\{l1, l2, l3\} >$$
$$(C3[10] <> C4[2])$$

How is this translated to a valid PEPA model component? The line $([C1[\{98, 0, 2\}] \parallel C2[\{0, 50, 0\}]]@[0, 2] < action1@[0, 1] >)$ will be first considered. The number of agents initially in $C1$ and $C2$ are defined for the compartments at index 0, 1 and 2 in the initial list of compartments, which respectively correspond to $l1$, $l2$ and $l3$. In order to derive the PEPA model component, the cooperation set between components at different compartments still needs to be determined. $action1@[0, 1]$ corresponds to the list of action $\{action1@l1, action1@l2\}$. This results in this valid PEPA model component: $(C1\_at\_l1[98] < action1\_at\_l1, action1\_at\_l2 > C2\_at\_l2[50] < action1\_at\_l1, action1\_at\_l2 > C1\_at\_l3[2]$.

The line $\{< action2@\{l1, l2, l3\} >\}$ requires a version of $action2$ for each of the compartments specified in the list. This translates as: $< action2\_at\_l1, action2\_at\_l2, action2\_at\_l3 >$. Finally, the last line $(C3[10] <> C4[2])$ does not need any changes, as it does not have any information about compartments. However, for harmonisation purposes, $<>$ will be rewritten $||$. The initial

model component can finally be rewritten:

$$(C1\_at\_l1[98] \underset{action1\_at\_l1,action1\_at\_l2}{\bowtie} C2\_at\_l2[50] \underset{action1\_at\_l1,action1\_at\_l2}{\bowtie} C1\_at\_l3[2])$$
$$\underset{action2\_at\_l1,action2\_at\_l2,action2\_at\_l3}{\bowtie} (C3[10] \parallel C4[2])$$

7.2.5 Automatically deriving a PEPA model from a PEPA model with compartments

The software presented in Section 3.4 has been expanded in order to derive PEPA models from models following the new grammar presented here. The change is straightforward: the initial window, previously used to select the PEPA file to analyse, has now a second option. It is now possible to select a file written following the rules of the new grammar allowing compartments, in order to translate it into a valid PEPA model.

The underlying algorithm used to translate the model is based on JavaCC [49]. The model is parsed, and translated to a valid PEPA model using the rules presented in the previous section 7.2.

7.3 Modelling the interaction between prairie dog towns affected by the bubonic plague

The model of a prairie dog town during a bubonic plague outbreak presented in section 5 has provided useful insight on the behaviour of the disease in one town, but ignored the interactions with nearby towns. While their influence is negligible for the study of one, already infected, town, given the duration and impact of the outbreak, it is essential to include them to understand the dynamic of the disease on the bigger scale.

In this section, a system consisting of four nearly identical prairie dog towns will be analysed. Each town consists of a hundred prairie dogs and a thousand fleas. The only difference lies in the infectivity of each of the towns. Indeed, only one town is initially infected, while the other three are healthy. The initial population break down of the infected town is the same as in the model described in section 5.5.1: 2 prairie dogs and 10 fleas are infectious, while the remaining individuals of their respective species are healthy. In the other towns, every prairie dog and flea is healthy.

A model written in the grammar presented in Section 7.2 will be presented and translated into a valid PEPA model, which will in turn be analysed using both the ODEs, using the Stirling Amendment, and stochastic simulations. The issues arising from this first study will then lead to a further search for ways to overcome them.

7.3.1 The model

The model consists of four towns: $t1$, $t2$, $t3$ and $t4$. The behaviour of the prairie dogs within each of the towns is exactly identical to the behaviour described in the original model presented in section 5.5.1. In addition to that, each prairie dog and flea can travel from a town to another and infectious prairie dogs can directly infect prairie dogs from different towns, expressing the case where two prairie dogs from neighbouring towns randomly meet.

In order to reproduce the behaviour previously described in section 5.5.1 in each of the towns, each action already allowed in the original model has some added information complying with the new grammar attached to it. For example, an action $(\alpha, rate).TargetComponent$ permitted by *Component* in the original model is translated into $(\alpha@[0, 3], rate).TargetComponent@[0, 3]$, performed by $Component@ALL$. $Component@ALL$ describes the fact that the component is present in all four locations. In the final PEPA model, it will translate into *Component_at_ti* with

i between 1 and 4. $\alpha@[0,3]$ means that the action α is permitted for the component *Component* in all four locations, but the name is different depending on the location. In location ti, the action will be called α_at_ti, resulting in a different name per location, to avoid an unwanted communication between individuals from different towns. The rate however, remains the same wherever the action happens, so its expression does not change from the original PEPA model to the model written using the new grammar. Finally, *TargetComponent* is required to be at the same location as *Component*. For this reason, it is written *TargetComponent*$@[0,3]$ which ties the component *TargetComponent_at_ti* to the component *Component_at_ti*, ensuring that both belong to the same town.

In addition to these actions already present in the original PEPA model, additional actions describing the relationship between towns are required. First, the movement of both the prairie dogs and the fleas is to be added to the model. In the case of the prairie dog, each living individual from a town is capable of travelling to any of the other three towns. The fleas however, need to be on a host in order to move over long distances. In any case, whichever species is concerned, the principle behind the set of actions describing movement is the same. For a given component located in town t, it is necessary to allow a different action for each location to which it can travel. This results, in this case, in three actions per component. Ideally, the action describing the movement to a town t would be the same for each original location. This would result, in the case of the susceptible flea on a host for example, with each component *Sh_at_ti* describing fleas on a host to be able to fire three actions $(movementFlea\_at\_tj, space\_movement\_flea).Sh\_at\_tj$, with $j \in [1,4]$ and $j \neq i$. This means that the set of actions is different for each location, and it is required to specify a different set for each of them. It results in the component $Sh@ALL$ requiring

the following set of actions:

$$[0](\mathit{movementFlea1}@\{t2,t3,t4\},\{\mathit{list\_of\_rates}\}).\mathit{Sh}@\{t2,t3,t4\} +$$

$$[1](\mathit{movementFlea2}@\{t1,t3,t4\},\{\mathit{list\_of\_rates}\}).\mathit{Sh}@\{t1,t3,t4\} +$$

$$[2](\mathit{movementFlea3}@\{t1,t2,t4\},\{\mathit{list\_of\_rates}\}).\mathit{Sh}@\{t1,t2,t4\} +$$

$$[3](\mathit{movementFlea4}@\{t1,t2,t3\},\{\mathit{list\_of\_rates}\}).\mathit{Sh}@\{t1,t2,t3\} \tag{7.2}$$

For example, for the component at the location at index 0, which corresponds to *Sh_at_t1*, this set of actions indicates that the actions *movementFlea1_at_t2*, *movementFlea1_at_t3* and *movementFlea1_at_t4* are allowed. These actions specify that an agent can move from *Sh_at_t1* to *Ch_at_t2*, *Sh_at_t3* and *Sh_at_t4* respectively. A similar set is defined for the component *Sh_at_t2*, *Sh_at_t3* and *Sh_at_t4*. Similar actions have been added to *Eh* and *Ih*.

The reason why four different actions for movement are required lies in the necessity to update the mirror subgroup. Indeed, to preserve the relationship between the number of agents in the components in the original subgroup and their respective counterparts in the mirror subgroup, each change of location in the original subgroup must result in the same change in the mirror subgroup. In order to do so, the component *AliveFlea* needs to communicate with *Sh*, *Eh* or *Ih* in order for an agent to move to the correct location. This results in the addition of the same set of actions to *AliveFlea*, with the exception that the rate at which it happens needs to be $\top$, as this component is passive here. However, it is essential to distinguish a flea going from town *t1* to *t3* from a flea going from *t2* to *t3*. If the actions were only *movementFlea_at_t3*, then it would be impossible for *AliveFlea* to distinguish between the two, and it would randomly move an agent from *t1* or *t2* to *t3*. By defining precisely the town of origin of the travelling flea, this confusion is avoided. Concerning the rate, it is assumed here that the rate of movement is different for each pair of towns. This means that we need to specify a different rate for each possible cooperation. Here, we will assume that the rate of travel between town *ti* and *tj* is

the same as the rate of travel from town *tj* and *ti* [2]. If the rate of travelling from a town *ti* to a town *tj* is called *space_movement_fleaij*, with $i < j$, then *list_of_rates* for i=2 for example, will be {*space_movement_flea12*, *space_movement_flea23*, *space_movement_flea24*}. So in this case, {*space_movement_flea12* represents both the movement of the fleas from the first burrow to the second, and from the second burrow to the first. We have chosen to have $i < j$ in order to ensure we only have one action name for movements in both directions.

In the case of the prairie dogs, a single set of actions for the whole population was not sufficient to describe the movement between towns. Indeed, because the infectious and healthy individuals are represented by different components in the mirror subgroups, two different actions are required to describe the movement of the prairie dogs. This means that both will rely on a set of actions similar to the set presented earlier (7.2), with a different action name (*movementInf* and *movementHealthy*), a different rate (infectious individuals travel more or less than healthy ones, which is not the case here) and a different target component depending on the component type considered.

Another set of actions has been added to the prairie dog components, describing direct contact between two prairie dogs of different towns. In this model, infectious prairie dogs have a chance to meet animals from different towns, and may infect them. In the case of simple direct transmission within a town, the mirror component of the infectious individual *InfectiousPD* was infecting every prairie dog alive in the main subgroup, namely *S*, *Exposed* and *Infectious* through the action *contact_direct* at the rate *contact_direct_rate*. In this case, the principle is identical, but instead of infecting prairie dogs from the same town, *InfectiousPD* only infects prairie dogs in a different location via the action *direct_contact_space*. This can be modelled in different ways, but the

[2]This has only been assumed in order to reduce the number of parameters. In reality however, geographical specificities, such as the difference in altitude between two towns, might render this hypothesis unrealistic. It is possible, using the same setup, to specify a different rate for each way.

simplest is to use a function to describe the action name. Indeed, while this form of contact is very similar to direct contact between prairie dogs, the main difference is that the contact is now only allowed with a prairie dog in a different location. This could have been done using the same method described for movement, using the set of actions seen in (7.2). However, because the infected prairie dog does not need to know the town the infectious prairie dog came from, the name of the action can be the same for each target location. This means that all the actions describing the infection of a prairie dog in a town ti from a prairie dog in a different town can share the same name. It could even be the name used for direct transmission within the town. However, the names will be chosen to be different to avoid confusion. Following the grammar described in section 7.2, this set of actions for all the components in the original subgroup representing prairie dogs that are alive (S, *Exposed* and *Infectious*) can be written as follows:

$$\begin{aligned}
&(\mathit{direct_contact_space}@(\#i+1), \top).C@[0,3] + \\
&(\mathit{direct_contact_space}@(\#i+2), \top).C@[0,3] + \\
&(\mathit{direct_contact_space}@(\#i+3), \top).C@[0,3]
\end{aligned} \tag{7.3}$$

As an example, let's consider the case of the third term of the list $[0, 3]$. In other words, this corresponds to $\#i$=2 for *direct_contact_space* and C@2. This means that the component allowed to fire these actions is at the location $t3$, as it is the location with index 2 in the original list of compartments L. In this case, $\mathit{direct\_contact\_space}@(\#i + 1)$ can be rewritten $\mathit{direct\_contact\_space}@(3)$, and the location at index 3 in L being $t4$, it translates as *direct_contact_space_at_t4* in PEPA. However, for $\mathit{direct\_contact\_space}@(\#i+2)$, corresponding here to $\mathit{direct\_contact\_space}@(4)$, there is no location with index 4 in L, as the size of this list is 4. Using the modulo of 4 by 4, we obtain an index 0 for the location, which is $t1$, resulting in the action *direct_contact_space_at_t1* in PEPA. Using the same process, $\mathit{direct\_contact\_space}@(\#i + 3)$ translates as *direct_contact_space_at_t2*. It can be noticed here that the action located in $t3$, is not used for these actions. This was the

objective here, as this results in alive prairie dogs in $t3$ not being able to be infected with this action by prairie dogs in the same town. The rate is $\top$ for all three actions. This is due to the component passively waiting to be infected. Finally, the target component will be C_at_t3 in all cases in our example.

InfectiousPD, on the other hand, only requires one term in the new grammar to describe the action of infecting prairie dogs in different towns. Indeed,

$$\begin{aligned}&(\mathit{direct\_contact\_space}@[0,3], \{\mathit{space\_direct\_contact\_rate1},\\ &\mathit{space\_direct\_contact\_rate2}, \mathit{space\_direct\_contact\_rate3},\\ &\mathit{space\_direct\_contact\_rate4}\}).\mathit{InfectiousPD}@[0,3]\end{aligned} \tag{7.4}$$

contains all the information needed to express it. For example, *InfectiousPD_at_t3* would translate as: $(\mathit{direct\_contact\_space\_at\_t3}, \mathit{space\_direct\_contact\_rate3}).\mathit{InfectiousPD\_at\_t3}$, which will only communicate with prairie dogs not in $t3$. Concerning the rate, it has been assumed that the rate is for infectious individuals of a given town, regardless of the origin of the contacted prairie dog. The value of this rate is to be determined by the modeller. This is obviously not biologically plausible, as the distance between the towns will have an impact on the rate of contact. However, modelling this would have required actions similar to the set of actions (7.2) to be incorporated, as this time, the location of the contacted prairie dog is necessary to determine the rate. In order to simplify the model, and exhibit the use of functions in the definition of the actions, this assumption has been made.

Finally, the system equation also needs to be modified to account for the introduction of space. Each of the subgroups is modified to incorporate the information about space. The original prairie subgroup for example, is now written: $([S[\{98,100,100,100\}]||\mathit{Infectious}[\{2,0,0,0\}]||\mathit{Ghost}[150]]@[0,3]<>)$. This expression means that

the first town $t1$ initially has 98 susceptible prairie dogs, 2 infectious and 150 ghosts while the other towns have 100 susceptible, no infectious and 150 ghosts. The final <> expresses the fact that these component types do not communicate over any action. The other subgroups have also been modified in a similar fashion. The expression of the actions in each communication set has also been updated. Each of the actions already present in the original model is now written $action\_name@[0, 3]$ to express the fact that the four actions *action_name_at_t1*, *action_name_at_t2*, *action_name_at_t3* and *action_name_at_t4* are now in the cooperation set. This has also been performed on *space_direct_contact*. The other new actions, *movementSi*, *movementInfi* and *movementFleai* with $i \in [1, 4]$, have not been dealt in the same way, as *movementSi_at_ti* is not an action used in the model. In order to avoid having this extra action in the cooperation set[3], these actions have been expressed using a list of locations. For example, in the case of the action *movementS2*, it has been described as $movementS2@\{t1, t3, t4\}$ in the new grammar. This translates as *movementS2_at_t1*, *movementS2_at_t3*, *movementS2_at_t4* in the derived PEPA model.

The model of a bubonic plague outbreak in a prairie dog community of four towns has now been written. The next step is now to study the results that the model can provide, given the tools that are available.

7.3.2 Results

The objective of this section is to test the model described in the previous section, to see whether the results obtained from it can be used to increase our knowledge of the disease. As explained in section 1.4, it does not however aim to give any actual additional biological knowledge, as too

[3]Writing an expression which would include this extra action would result in a perfectly valid PEPA model. However, we chose here to prevent this as it would only add actions not used in the model.

many assumptions, and in particular the new parameters used in this section, have no biological justification.

First, the remaining parameters have to be chosen. In order to test different scenarios, the four towns are placed such that three of them are close to one another, and the last one is further away. If placed in a grid, $t1$, $t2$, $t3$ and $t4$ would have the coordinate $(0,0)$, $(0,1)$, $(1,0)$ and $(12,5)$ respectively. The hypothesis is made that the ground is uniformly flat with no obstacles. Also, a prairie dog or a flea travels a distance of 1 in the grid once every 1000 days, which results in a rate of 0.001 and the travelling rate is inversely proportional to distance travelled. In other words, for a travel of 5 on the grid, the rate is 0.0002. Finally, the effective contact rate of infectious prairie dogs with prairie dogs from a different location of different towns have been set to 0.00073, which corresponds to a hundredth of the value for the direct contact rate within a town.

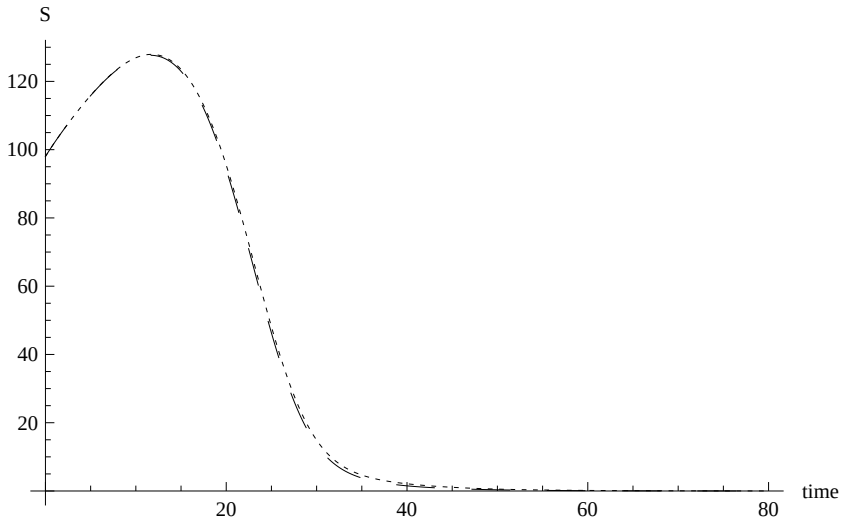

Figure 7.2: Comparison of the total number of susceptible prairie dogs during a bubonic plague outbreak in the initially infected town between the ODEs of the model featuring only one town (dashed line), and the ODEs of the model with four towns (dotted line).

The simulations of the resulting model cannot be performed. Indeed, the size and complexity of

the model was too much to handle for the PEPA Eclipse plugin. Even reducing the model to only represent two locations does not solve the issue. However, it is still possible to derive and solve the ODEs from the model. First, a validation of the behaviour of the location already infected ($t1$) was performed, and the resulting graph from the ODEs was compared to the graph obtained using the original model presented in section 6.4.1. The result of the comparison is presented in Figure 7.2. This graph confirms that the model representing four towns still exhibits the same behaviour within a single town.

The next step is to analyse the impact of the infectivity of the first town $t1$ on its three neighbouring towns. Figure 7.3 shows the evolution of the number of susceptible prairie dogs in the towns $t1$, $t2$ and $t4$. The town $t3$ is not represented as it is very similar to $t2$. This graph shows the expected delay between the outbreak in $t1$ and the one in $t2$, and a bigger delay for the one in $t4$, which is consistent with the parameter values. However, the difference in delay between the outbreaks in $t2$ and $t4$ is very small (about five days) compared to the delay between the ones in $t1$ and $t2$ (about 25 days), considering that $t4$ is 13 times further from the original source of transmission than $t2$. This is related to how ODEs function, as detailed in section 3.1.2. Indeed, the ODEs only record occurring epidemics, and discard situations where the disease dies out. This means that what Figure 7.3 actually shows is the average number of susceptible prairie dogs when an outbreak occurred in their town of origin. This explains why the delay does not increase significantly with the distance between towns. Indeed, all the prairie dogs in $t1$ are dead by the 45^{th} day of the epidemic. This means that regardless of the distance between $t1$ and $t4$, either $t4$ is contacted by an infectious prairie dog before day 45, or the outbreak in $t1$ does not have a direct influence on the presence of the bubonic plague in $t4$. One could argue that $t2$ and $t3$ also can have an influence on the presence or not of the bubonic plague in town $t4$. However, in practice, their influence is actually minimal compared to the influence of $t1$. Indeed, $t2$ or $t3$ infecting $t4$ would require both $t1$ to infect $t2$ (or $t3$) and $t2$ (or $t3$) to infect $t4$. As we will see

later in this section, the probability of each of these events happening is low, which results in an influence of $t2$ and $t3$ negligible compared to the influence of $t1$, especially as the distance between $t1$ and $t4$ is almost the same as the distance between $t2$ and $t4$ and between $t3$ and $t4$.

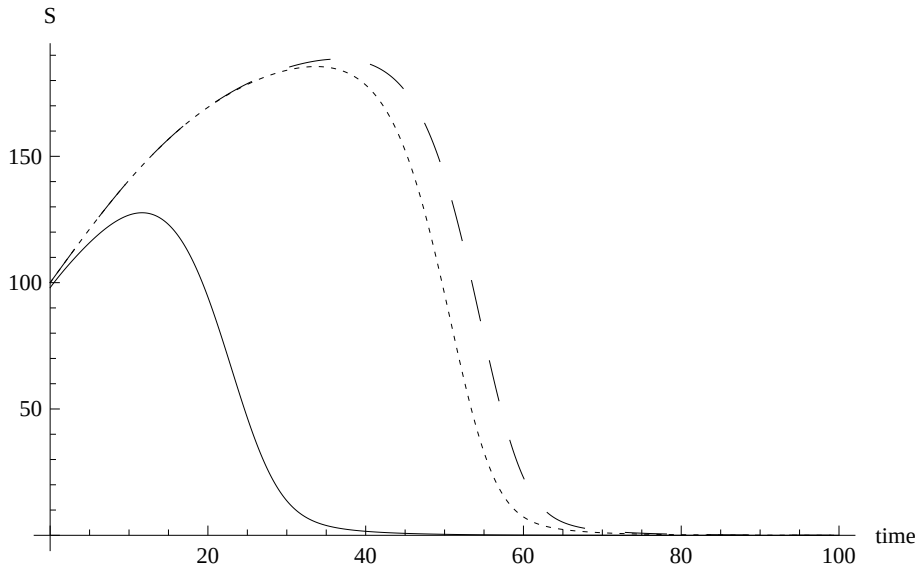

Figure 7.3: Comparison of the total number of susceptible prairie dogs during a bubonic plague outbreak in the initially infected town $t1$ (solid line), the town $t2$ (dotted line) and the town $t4$ (dashed line).

While the model and the ODEs derived from it do not give us the average behaviour of the biological system, but rather the average behaviour of the successful transmission from one burrow to another, they give us information about what happens when the disease spreads to nearby towns. Indeed, assuming that the parameters used in this study are correct, it shows how long it takes on average for the bubonic plague to spread to nearby towns, and how severe the outbreak is. But in this section so far, PEPA has not given any information about the model that directly writing the ODEs would not have provided.

The key information missing here is, given a set of parameters, how often does an infected town

A infect a nearby town B?

In order to answer this question, the original prairie dog model described in section 5.5.1 will be altered in order to add some information about space in it, while keeping it simple. A few actions representing the movement of the prairie dogs and fleas, and the direct contact of infectious prairie dogs with individuals from a different town will be incorporated to the model. The objective here is to record how a nearby town gets infected by its neighbours, how many infectious prairie dogs or fleas travel to this new town, and how many infectious direct contacts are made on average over the course of the outbreak with the originally infected town. With this information obtained, it is then possible to model different likely initial conditions for the newly infected town. The behaviour of the third town can then be approximated by taking into account the influence of the first two towns. The behaviour of the whole system can be approximated by increased the system's size by one burrow at a time. The resulting information must however be considered caution: the interactions between towns have been greatly simplified (what about the influence of the third town on the second?), and this might alter significantly the conclusions depending on the system.

The original model has been modified as follows:

- the action *movementPD* has been added to the *Infectious* component, which fires at the rate *movement_rate*, and brings the agent firing it to the component type *InfectiousMoved*. At the same time, the component *Infectious* communicates with *InfectiousPD* in the mirror subgroup, to ensure the number of agents remains the same in both component types.
- A similar action is added to the component *Ih*. A new component type *InfFleaMoved* has been added to the model in order to record the number of agents moving to the new town.
- A new subgroup is added to the model, representing the prairie dogs in the target town infected via direct contact with an infectious prairie dog from the originally infected town.

An agent in this subgroup can be in two component types: *Idle* or *DirectInfections*. This follows the assumption that all the prairie dogs in the target town are susceptible. However, this makes sense in this context, as if the target town is already infected, the influence of the infectious town would be insignificant, as the plague would kill the prairie dogs in this town in every case. Initially, all agents are in the *Idle* state. Each agent in *Idle* can passively communicate with an agent in *Infectious* over the action *direct_contact_space* to move to the *DirectInfections* state. The rate at which the action can be fired in *Infectious* is *direct_contact_space_rate*.

The parameters describing the movement between the infected and target towns are the ones previously used to describe the movement and communication between towns $t1$ and $t2$. In other words, we used *movement_flea_rate*$=0.0001$, *movement_rate*$=0.0001$ and *direct_contact_space_rate*$=0.00073$. Under these conditions, a thousand simulations have been performed, and their average, as well as the ODEs derived from the model, are shown in Figure 7.4. According to the stochastic simulations, on average, 0.041 infectious prairie dogs and 0.004 fleas travel from the original town to the new one, and 0.277 direct contacts are made. The reason more infectious prairie dogs are travelling is the result of two factors. First, over the course of the disease outbreak, more prairie dogs are infected than fleas. Furthermore, only fleas on the host are capable of travelling, as the insect cannot cover long distances, which reduce even further the number of infected fleas which can potentially infect the susceptible town. The number of fleas having travelled must however be taken with a pinch of salt: only four fleas overall have travelled to the susceptible town during the thousand runs. Given this information, and considering the behaviour of a town with one infected prairie dog or flea[4], the behaviour of the system including

[4]By only considering situations where one prairie dog or one flea is infected at the start of the experiment, the hypothesis that the town is infected at most once is made. Given the scale of the number of infections, the case where the town is infected more than once is negligible. However, if it ceases to be the case, it is possible to write

both towns can be derived. Towns can then be added to the system one by one, by determining the influence of each previously added town.

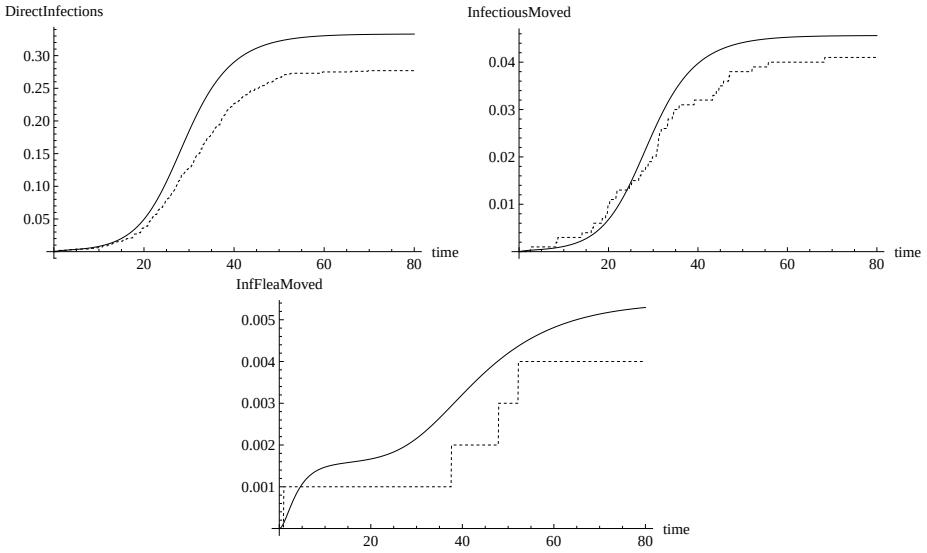

Figure 7.4: Average number of infection of a nearby prairie dog town by vector (direct contact, movement of an infectious prairie dog, and movement of a infectious flea). Both the average of 1000 stochastic simulations (dotted line) and the ODEs (solid line) are shown.

While this solution is not optimal and the analysis can be tedious, it allows a good understanding of the behaviour of the system, and gives detailed information about how the disease spreads in a spatial context.

7.4 Adding space to measles

The incorporation of different towns in the prairie dog model showed some promising results. Indeed, the model itself was simpler to write than it would have been to write it all by hand, and

a supplementary model that only records if the town has been infected, and which vector was responsible of the infection. With the information given by these two graphs, a more accurate picture of the system can be drawn.

would be fairly simple to extend to more towns. Also, some results were obtained through the automatic derivation of the ODEs, while others were obtained by using a slightly modified version of the initial model. However, due to computer/software limitations, simulations of the model including the four towns could not be performed, and obtaining results about the behaviour of the whole system using the presented methods is tedious.

In this section, the measles model presented in section 6.4.1 will be extended to include four cities. The four cities chosen are Cardiff, Newport, Bristol and Bath. The reason behind this choice lies in the proximity of the cities (typically about 15 to 30 miles from one city to the next one), and their size. The cities cover a wide range of scenarios: the biggest, Bristol, had $436,300$ inhabitants in 1944, the smallest, Bath, had $79,550$ inhabitants, while Cardiff and Newport population sizes lay somewhere in the middle, with $248,000$ and $104,795$ inhabitants respectively [89]. The system underlying this model is less complex than the one underlying the bubonic plague: only one species is represented, the disease spreads through one transmission route only and the way birth has been modelled is simpler. For these reasons, the model is more suitable to an incorporation of different locations, and it is possible to perform stochastic simulations directly on the model.

7.4.1 The model

The model is based on the original model describing measles in Leeds presented in section 6.4.1. All the actions already present in that model are still allowed in the one including several locations, resulting in preserved behaviour within a city. While people moving permanently from one of those cities to another have been ignored due to the limited impact it has, commuting has been added to the model.

First, the list of cities has been added to the model. This has been done with the instruction $\#compartments{=}\{cardiff, newport, bristol, bath\};$. Capital letters for the name of the different

cities is not allowed by the grammar. Contact rates for individuals, and birth and death rates depend on the city. The contact rates are designated by $crCN$, $crCBr$, $crCBa$, $crNBr$, $crNBa$ and $crBrBa$, with the assumption that the contact rate between an infectious individual from a city A and a susceptible individual from a city B is the same as the contact rate between an infectious individual from a city B and a susceptible individual from a city A. In the case of the birth and the death rate, the rates have been respectively renamed brC and drC for Cardiff, brN and drN for Newport, $brBr$ and $drBr$ for Bristol and $brBa$ and $drBa$ for Bath. The values chosen will be discussed in the parameters section 7.4.2. Each component type expression has also been updated. The component type itself had the information about the location added, by attaching $@ALL$ to its name. Each action and target component type have also been updated by adding $@[0,3]$ to their respective expression. When the rate is the same regardless of the city, the rate's expression remains unchanged.

One subgroup remains unchanged. The *Winter* and *Summer* component types are supposed to provide identical information regardless of the location, and thus do not have a location. Also, a limitation in the software used to translate the model from the new grammar to a valid PEPA model prevents the use sequences of actions. This results in a modification in the description of the change of season. It was modelled in the original model as $(go\_summer, srw)^p.Winter$ for the summer to winter transition. Here, it was simplified as $(go\_summer, srw).Winter$, reducing the sequence to a single action. Given that no location is involved in this subgroup, the expression can be manually reverted back to its original description in the resulting PEPA model. The same operation would have to be performed for the winter to summer transition.

The only new possible behaviour allowed in this model corresponds to commuting. Two types of individuals can commute: healthy individuals[5] and infected individuals. One solution to avoid

[5]Healthy individual commuting is restricted to S only. Including Exp or R would not change the behaviour of

this distinction would have been to consider that if a susceptible individual contacts an infectious one when she commutes, it is the same as if the infectious one was commuting. However, while this might be a sensible approximation when the proportion of healthy and infectious individuals is similar in the two towns, the behaviour can change dramatically if it is not. If a situation where two towns $t1$ and $t2$, with 2 and 5 *Infectious*, 0 and 1 S and 10 and 50 R is considered, the overall rate of transmission would be dramatically different if this approximation is made. Indeed, if the susceptible individual from $t2$ travels to $t1$ and contacts 3 individuals (it can be the same individual twice), assuming a contact with an *Infectious* is always successful, then the probability of her being infected after her day is $1-(9/10)^3$=0.271. If the approximation was made, and the infectious individual travels to $t2$, and contacts 3 individuals, the probability for the S to be infected is $1-(55/56)^3 \simeq 0.053$, about 5 times lower. In the case of measles, the proportion of susceptible individuals between a city where an outbreak started a few weeks earlier and one where the disease has faded out a few years before might vary significantly, especially in smaller communities, where the disease might fade out for extended periods of time. This is the choice of the modeller.

First, the case where *Infectious* individuals commute has been introduced to the model. The opposite approach compared to the prairie dog *direct_contact_space* is chosen: in this case, the location specified in the action name will indicate the location to which the infectious individual commutes. This simplifies greatly the expression added to S, *Exp*, *Infectious* and R. Indeed, only the term $(contact\_space@[0,3], \top).TargetComponent@[0,3]$ is required. *Infprimes* and *Infprimew* contain the remaining information. Indeed, it is essential to ensure that each infectious individual only communicates with individuals in different towns. This has been realised with the following

the disease: S drives the cooperation (as detailed later in the section), which means there is no competition between the susceptible individuals and the exposed and recovered ones.

expression for *Infprimew*:

$$
\begin{aligned}
&[0](contact\_space@\{newport, bristol, bath\}, \{crCN, crCBr, crCBa\}) \\
&\qquad\qquad .Infprimew@\{cardiff, cardiff, cardiff\} + \\
&[1](contact\_space@\{cardiff, bristol, bath\}, \{crCN, crNBr, crNBa\}) \\
&\qquad\qquad .Infprimew@\{newport, newport, newport\} + \\
&[2](contact\_space@\{cardiff, newport, bath\}, \{crCBr, crNBr, crBrBa\}) \\
&\qquad\qquad .Infprimew@\{bristol, bristol, bristol\} + \\
&[3](contact\_space@\{cardiff, newport, bristol\}, \{crCBa, crNBa, crBrBa\}) \\
&\qquad\qquad .Infprimew@\{bath, bath, bath\} \qquad (7.5)
\end{aligned}
$$

The four expressions follow a similar principle. For example, in the case of the first expression, it corresponds to the infectious individual being in Cardiff, which means she can only communicate with individuals in Newport, Bristol and Bath, at different rates, and remain in Cardiff in all cases. The corresponding expression in *Infprimes* only requires every *Infprimew* to be replaced by *Infprimes*. The rate remains the same, as will be detailed in the parameters section 7.4.2.

The second term added to the original model corresponds to the contact of susceptible individuals commuting with infectious individual in a different city. In this term, the information about the location corresponds to the city of origin of the commuting S. First, the S component type is required to allow the action *contact_spaceS* to be fired. Because the rates are different depending on the pair of cities involved in the communication, a different expression is required depending

on the city from which the susceptible individual comes:

$$
\begin{aligned}
&[0](contact\_spaceS@\{cardiff, cardiff, cardiff\}, \{crCN, crCBr, crCBa\}) \\
&\qquad\qquad .S@\{cardiff, cardiff, cardiff\}+ \\
&[1](contact\_spaceS@\{newport, newport, newport\}, \{crCN, crNBr, crNBa\}) \\
&\qquad\qquad .S@\{newport, newport, newport\}+ \\
&[2](contact\_spaceS@\{bristol, bristol, bristol\}, \{crCBr, crNBr, crBrBa\}) \\
&\qquad\qquad .S@\{bristol, bristol, bristol\}+ \\
&[3](contact\_spaceS@\{bath, bath, bath\}, \{crCBa, crNBa, crBrBa\}) \\
&\qquad\qquad .S@\{bath, bath, bath\}; \qquad (7.6)
\end{aligned}
$$

It can also be noticed that S needs to be the component driving the communication. Indeed, unlike previous direct transmission examples, the infectious individuals are the ones competing with healthy individuals sharing the same location to contact the susceptible commuters. Also, having Inf ($Infprimes/Infprimew$) driving the communication would mean having the commuting susceptible competing against individuals in a different location.

The action $contact\_space\_S$ is also required in the expression of $Sprime$, $Infprimes$ and $Infprimew$. Here again the expression is different depending on the location. For $Sprime$, the

expression would be:

$$
\begin{aligned}
&[0](contact\_spaceS@\{newport, bristol, bath\}, \top) \\
&\qquad\qquad .Sprime@\{newport, bristol, bath\}+ \\
&[1](contact\_spaceS@\{cardiff, bristol, bath\}, \top) \\
&\qquad\qquad .Sprime@\{cardiff, bristol, bath\}+ \\
&[2](contact\_spaceS@\{cardiff, newport, bath\}, \top) \\
&\qquad\qquad .Sprime@\{cardiff, newport, bath\}+ \\
&[3](contact\_spaceS@\{cardiff, newport, bristol\}, \top) \\
&\qquad\qquad .Sprime@\{cardiff, newport, bristol\}
\end{aligned} \tag{7.7}
$$

An agent in one of the component types in a given location can only passively communicate over an action with an agent in a different location. For example, the first term corresponds to Cardiff, and can only communicate over the actions $contact\_spaceS\_at\_newport$, $contact\_spaceS\_at\_bristol$ or $contact\_spaceS\_at\_bath$. In the case of *Infprimes* or *Infprimew*, the expression used is similar. However, if left as it is, when the communication happens the agent in S cannot differentiate between a communication with an *Sprime* agent or a communication with an *Infprimes* or *Infprimew* agent. In order to make sure a communication with an infectious agent results in a new infection in the original location of the agent in S, the target component of the actions in *Infprimes* and *Infprimew* is replaced by *InfprimeTempi*, with $i \in [1, 4]$. These four new component types have

the following expression:

$$
\begin{aligned}
&\mathit{InfprimeTemp1}@[1,3] = (\mathit{infected}@[1,3], \mathit{big}).\mathit{Infprimes}@\{\mathit{cardiff}, \mathit{cardiff}, \mathit{cardiff}\};\\
&\mathit{InfprimeTemp2}@\{\mathit{cardiff}, \mathit{bristol}, \mathit{bath}\} = (\mathit{infected}@\{\mathit{cardiff}, \mathit{bristol}, \mathit{bath}\}, \mathit{big})\\
&\qquad\qquad .\mathit{Infprimes}@\{\mathit{newport}, \mathit{newport}, \mathit{newport}\};\\
&\mathit{InfprimeTemp3}@\{\mathit{cardiff}, \mathit{newport}, \mathit{bath}\} = (\mathit{infected}@\{\mathit{cardiff}, \mathit{newport}, \mathit{bath}\}, \mathit{big})\\
&\qquad\qquad .\mathit{Infprimes}@\{\mathit{bristol}, \mathit{bristol}, \mathit{bristol}\};\\
&\mathit{InfprimeTemp4}@[0,2] = (\mathit{infected}@[0,2], \mathit{big}).\mathit{Infprimes}@\{\mathit{bath}, \mathit{bath}, \mathit{bath}\};
\end{aligned}
\tag{7.8}
$$

Each component type corresponds to the location where the infection happened. For example,*InfprimeTemp1* indicates that the infection happened in Cardiff. The location of these component type specifies the location from which the infected susceptible commuter came. With these two pieces of information, the component first fires the action *infected_at_location*, which communicate with an S in *location* that has been infected, and returns at a high rate to *Infprimes*[6] at the location indicated by the component type's name. An action *infected* is also added in the expression of S, with both the action and the component belonging to the same location. This action passively brings an agent to the state *Exp*, as it indicates that it has been infected while commuting. This agent might not be the one that has been infected in the first place, but as they both belong to the same component type, they are indistinguishable anyway.

The final modifications concern the system equation. Most of the changes have already presented in the prairie dog model in section 7.3.1: each component type needs to indicate its initial conditions in each location, each subgroup is modified to specify that it exists in all four locations, and the cooperation set needs to be updated in the same manner. The actions *contact_space*@$[0,3]$,

[6]It returns to *Infprimes* even when the season is *Winter*. It does not change the overall behaviour of the model as the agent will immediately go to *Infprimew* in the case where the season is *Winter*, given the way this has been modelled.

contact_spaceS@$[0,3]$ and *infected*@$[0,3]$ are to be added to the cooperation set between the subgroup describing the behaviour of the individuals and its mirror subgroup. However, the subgroup representing seasonality, along with its cooperation set with the rest of the model, remains unchanged, as the season is the same regardless of the location.

7.4.2 Parameters

The parameters describing the behaviour of the system within a city have been detailed in section 6.4.2. The birth rate and the death rate have however been adapted to fit to the new cities presented here. The method used to derive them is identical to the one used to derive them in the case of Leeds. This results in the birth and death rates presented in Figure 7.5.

	Birth rate (10^{-3})	Death rate (10^{-3})
Cardiff	5.23	5.23
Newport	5.07	5.07
Bristol	4.06	4.06
Bath	4.20	4.20

Figure 7.5: The birth and death rate used for the different cities.

The only new parameters in this model correspond to the contact rate of individuals belonging to different cities. Miller [65, Table II] confirmed that measles infects essentially children before they reach the age of 10. This means that work commuting does not play a role in disease behaviour. Only commuting resulting from weekend trips to neighbouring cities, for shopping or family/friends' visits, is to be considered. The consequence of this is that the season has no influence on the contact rate for individuals in different cities: it remains the same throughout the

year. Three assumptions have been made to determine the actual contact rate between individuals from two given cities. The first one is that the commuting rate is independent of the city size. The second is that the rate is inversely proportional to the distance between the two cities. The third assumption is that one in thirty individuals (adults and children alike) travels to a city 30 minutes drive from his home town once a week. The approximate driving times between Cardiff and Newport, Newport and Bristol, and Bristol and Bath are 30, 45 and 30 minutes respectively. It is also assumed that the time to go from two cities is the sum of the times it takes to travel through all the cities along the way. This results in the trip between Cardiff and Bristol assumed to take 30 + 45=75 minutes, 30 + 45 + 30=105 minutes for Cardiff-Bath, and 45 + 30=75 for Newport-Bath. The values used in this study are presented in Figure 7.6.

	Cardiff	Newport	Bristol	Bath
Cardiff	NA	0.0145	0.0058	0.00414
Newport	0.0145	NA	0.00967	0.0058
Bristol	0.0058	0.00967	NA	0.0145
Bath	0.00414	0.0058	0.0145	NA

Figure 7.6: The contact rate between individuals from different cities.

Different sets of initial conditions have been tested. The initial proportion of susceptible varied between 3% and 100%, while the number of cities initially infected varied from one to four. The size of the pool of *Ghost*, on the other hand, is constant throughout the simulations, and has been chosen large enough to avoid it being depleted, but small enough to reduce its impact on the performance of the simulations. In the case of the cities studied here, the number of *Ghost*, determined experimentally, correspond to 13.7% of the population of Bristol, up to 31% of the population of Bath, with 20% and 28% for Cardiff and Newport. The reason behind this difference

lies in the reduced influence of stochastic events in larger populations.

7.4.3 Results

The models and parameters exposed in the previous sections have been used to perform a series of single stochastic simulations, similarly to what was done in section 6.4.3. Unfortunately, probably due to the PEPA Eclipse plugin not having been conceived to deal with the type of models presented here, all the simulations were giving incorrect results after a varying number of time steps, with negative numbers of agents, or unnaturally constant number of agents in a certain state.

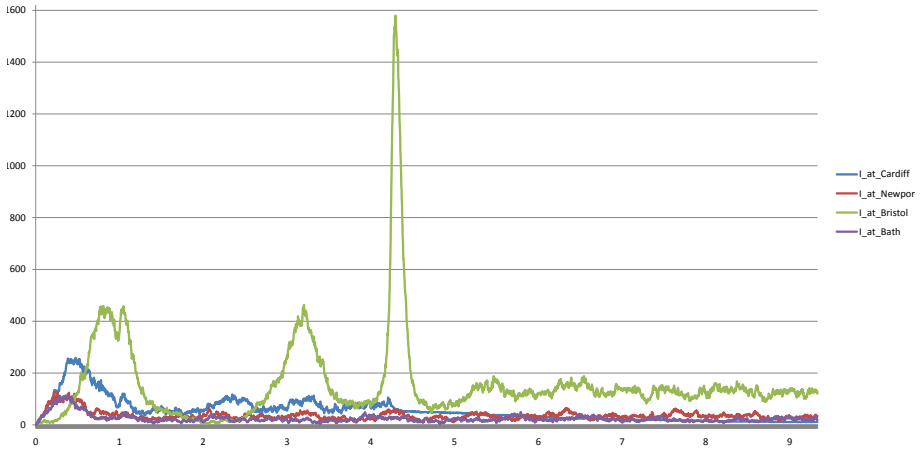

Figure 7.7: Number of infectious individuals in Cardiff, Newport, Bristol and Bath over one nine-year simulation.

The graph presented in Figure 7.7 shows the best simulation obtained. In this simulation, 5% of the population of each city is initially susceptible, while the rest is immune to the disease. The reason the number is lower than before lies in the fact that the simulation were only running for a few years, and we wanted to discard as few days as possible, due to the proportion of susceptible being unrealistic. In this simulation, the whole range of the simulation was kept, as the initial

conditions were not very far from the range of proportions of susceptibles witnessed throughout the experiment. Initially, only Bristol has ten infectious individuals, with the rest of the cities free of infection. As a result, it is the only city that experiences disease outbreaks after the first year, and they seem to happen every one to two years, which is consistent with the data available. Whether it has a link remains to be determined. The disease then settles in a steady state, and no subsequent outbreaks are witnessed. The three remaining cities do not experience any outbreak and quickly reach steady state.

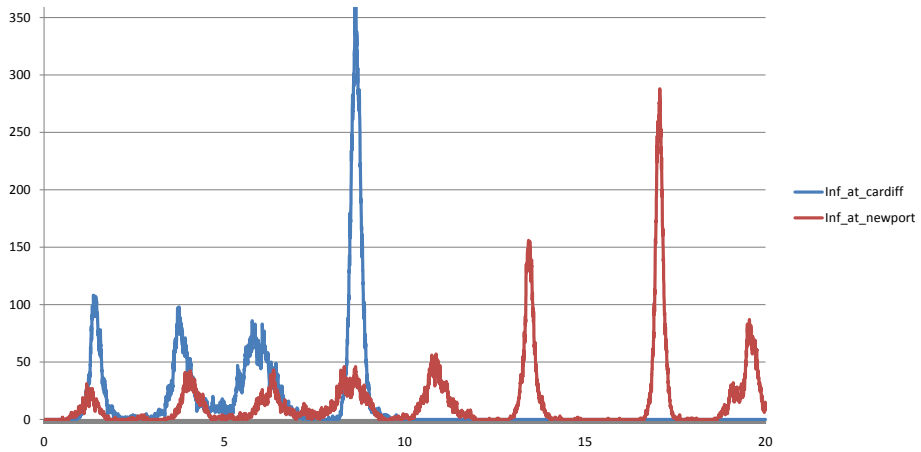

Figure 7.8: Number of infectious individuals in Cardiff and Newport over a twenty-year simulation.

In an attempt to get more significant results, the model was simplified to consist of only two cities (Cardiff and Newport), with the remaining parameters the same as before, with the whole population but 10 infectious individuals in each city being susceptible to the disease.Once again, some of the simulations exhibited unrealistic results, and were discarded. This time however, the experiments were running for longer on average before their behaviour was incorrect. The graph on Figure 7.8 shows the most successful simulation obtained. In this particular simulation, the first 1000 days have been removed from the graph. In this case, sensible results are obtained during the

first 9 years of simulations. After that, the number of infectious individuals in Cardiff stabilises around 11, which is not biologically realistic. In the case of Newport on the other hand, the results remain more sensible. In the initial years, both towns seem to experience a disease outbreak every other year, which is consistent with the data. Afterwards, only 4 outbreaks are recorded in 11 years in the case of Newport and none in Cardiff. The reduced number of outbreaks in Newport could be the consequence of increase stochasticity resulting from the absence of nearby cities. The higher peak witnessed at the outbreak following the four year period without any is sensible, as the number of susceptible individuals will have increased in those years.

7.5 Summary

This chapter aimed at providing ways to add spatiality to PEPA models. The choice has been made to conceive syntactic sugar on top of PEPA which would simplify the conception of PEPA models with compartments. The resulting model can then automatically be translated to a valid PEPA model. This solution comes with strengths and weaknesses. The fact that the analysis is performed on a PEPA model allows the powerful tools already available to be used. Also, it is easier to learn how to use this syntactic sugar when PEPA is already known, allowing the modeller to avoid having to learn two completely different modelling tools. On the other hand, this also means that the tools are not tailored to the particular form of those PEPA models, and may not be very efficient in terms of performance. Additionally, the expressivity of those models is limited by PEPA's grammar, which was not initially developed with compartments in mind.

The method developed was then illustrated through a generalisation of the models initially presented in section 5.5 and section 6.6, of the bubonic plague in prairie dogs to include more than one town, and the model of measles in Leeds, in order to account for four cities. In the case of the

prairie dogs model, the resulting PEPA model, even restricted to two towns, could not be used for analysis, as the available tools were not able to handle its complexity. A solution has been provided to nonetheless allow for some limited analysis to be performed. The measles model described the behaviour of four cities (Newport, Cardiff, Bristol and Bath). The simulations including all four cities were usually limited to a few years, and the results were often questionable. When reduced to Newport and Cardiff only, the model provided more meaningful results. It showed however that tools adapted to these kind of models might provide a significant improvement in their capacity to deal with PEPA models with compartments.

Chapter 8

Conclusion

8.1 Thesis summary

The aim of this thesis is to investigate the potential of PEPA as a tool to model and analyse epidemiological systems, and extend the formalism when necessary. The approach used in this work was to study two systems as case studies, bubonic plague outbreaks in a prairie dog population and measles in England and Wales, to determine the important features in epidemiology, and examine how these features can be modelled using PEPA.

First, the two different types of transmission, direct and indirect have been modelled in chapter 2. The result of the analysis led to the conclusion that a new algorithm to derive ODEs was required. Indeed, the Hillston algorithm [43] was not initially tailored to deal with epidemiological models. As those often have terms which are not common in computer system models, the resulting ODEs were often not describing the average behaviour of the model accurately. Chapter 3

presented an amendment of this algorithm, extending its scope to allow cooperation between several component types from two subgroups. As a result, most epidemiological models can now be translated from PEPA to ODEs. Four aspects of epidemiological systems have then been studied in separate chapters. First, methods to obtain frequency and density dependent transmission in a model have been described in chapter 4. The birth and death feature has then been studied in depth in chapter 5. Different methods have been designed to adapt to the modelled system and to the amount of complexity the modeller wants to reach. The chapter also showed the model of a bubonic plague outbreak in a single prairie dog town, which included the features presented in the first chapters. In chapter 6, timed events were presented, and the principle has been used to derive a method to model seasonality and the circadian clock. It has then been applied to simple models with control measures and measles in Leeds. Finally in chapter 7, a new grammar based on PEPA was conceived to allow the inclusion of compartments in the model. The grammar consists of a syntactic sugar build on top of PEPA, and the rules used to translate the model in a valid PEPA model have been provided. The translation is automated by a software presented in that chapter. The two cases studies seen earlier, the bubonic plague and measles, have then been expanded to represent four prairie dog towns and four cities respectively.

8.2 Strengths and weaknesses of PEPA when modelling epidemiological systems

PEPA inherently has some characteristics that makes it a good potential candidate to be used as a tool to model epidemiological systems. The most important of those is the ability to move from the individual to the population scale. Indeed, being able to express the behaviour of single individuals allows the modeller to express more biologically realistic information in the model. The

analysis is obtained at the population level, which results in conclusions applicable to the whole system.

The ability to express stochastic events also helps to describe epidemiological systems more accurately. Indeed, the variability resulting from stochastic events shows the spectrum of possible outputs of a given situation which a deterministic tool, such as ODEs, would not exhibit. However, this comes with a drawback: not all events are stochastic. As seen in the case of seasonality, or for external interventions, being able to impose a specific time for an event can be essential for some systems. The method presented in chapter 6 gives an approximation which is satisfactory in most cases, but it may still lack precision in cases where the timing is essential.

PEPA also gives access to ODEs describing the average behaviour of the model in most cases. Using PEPA to obtain the ODEs has several advantages over writing them directly. As seen previously, the assumptions used in the model are at the individual level, and some model checking is available to prevent some common mistakes. Moreover, the same model has now access to both the type of analyses available for PEPA models and the extensive research performed in the use of ODEs in modelling.

PEPA also has access to a very complete tool, the PEPA Eclipse plugin, to facilitate the analysis of a model. One of its underestimated potential uses is the capacity to perform a basic model checking. A series of properties are tested, which often leads to the early discovery of errors. These errors are usually PEPA related, but some of them may indicate modelling or conceptual mistakes made in the model. A previously unknown error in a published biological model has been spotted by Clark et al. [19]. In that case, the original model was translated in BioPEPA, which immediately detected a dubious behaviour, leading to a straightforward correction. Similar verifications are performed on PEPA models, and errors such as using one parameter instead of

another, or forgetting to add an action, could more easily be spotted. These potential errors are usually the most difficult to uncover as they can give perfectly sensible results.

From the work presented in this thesis, several features have been modelled using PEPA, with varying degrees of success. Indirect transmission can be modelled very naturally in PEPA, whether through an infected environment or via a host. Direct transmission requires more subtle changes to the model. A solution has been provided in chapter 4, whether it concerns frequency (section 4.2) or density (section 4.3) dependent transmission. However, density dependent transmission required a non-biologically inspired solution, which required population to individual level reasoning, which defeats the purpose of using process algebra in this case. Births and deaths however, were provided with very satisfactory solutions. Various solutions were conceived, each of them adapted to a different situation.

Compartments cannot directly be modelled in PEPA. However, a solution approaching it has been provided in chapter 7. The grammar conceived allowed them to be easily expressed, in a concise and intuitive way. The translation of the model written following the rules of this grammar to a valid PEPA model has been made simple by the addition of software that does it automatically. However, the resulting model grows in size and complexity linearly with the number of compartments, quickly resulting in a situation where the current tools are not able to deal with it.

PEPA does not have the option to directly allow some features to be included in the model. Functional rates, timed events, or direct cooperation between 2 agents in the same subgroup, among others, are not permit by PEPA's grammar. It has been shown however that for each of the limitations encountered throughout this thesis, there was a way to approach the desired behaviour.

8.3 Current developments in PEPA

While the scope of PEPA may seem currently constrained in the complexity of the model and the size of the population used, some largely unpublished ongoing research focuses on increasing the set of PEPA models and initial parameters amenable to automated analysis.

A new tool is being developed by Anton Stefanek [81, 80] based on the slightly modified version of PEPA, GPEPA, proposed by Hayden and Bradley [40]. Several types of analyses are available: average of stochastic simulations, ODEs, variance of one component, covariance of two components, central moment and standardised central moment of one component[1].

Another novel type of analysis, also based on GPEPA, is being developed [82] and already exhibits some potential in allowing simulations to be performed for models considered too large or complex by previous tools. The principle underlying this analysis is that ODEs give a good approximation of the average of stochastic simulations on some intervals of the timeline, but cannot describe the behaviour properly on others. Within the intervals where the ODEs give an unsatisfactory approximation of the average of the stochastic simulation, the ODEs are replaced by the average of a number of stochastic simulations. In practice, if the interval where the ODEs are unreliable is $[t_0, t_1]$, and the analysis is run between $[a, b]$, with $a < t_0 < t_1 < b$, then the ODEs are used to calculate the average number of each component between a and t_0. The values for each component at $t = t_0$ are then fed to stochastic simulation software[2]. A series of simulations are

[1]The variance, covariance, central and standardised centre moments calculations are based on the approximation of the component counts given by the ODEs analysis.

[2]Two methods exist at the moment to perform this task. The first one simply uses the real values for the components, and the stochastic simulation software has been adapted to be able to perform this task. The second one uses ODEs to calculate the mean and covariance of each component to determine the proportion of simulations starting with different natural numbers of agents in each component type.

performed between t_0 and t_1. The average number of agents in each component is then fed to the ODEs at $t{=}t_1$. The component counts are then computed for the interval $[t_1, b]$.

8.4 Future work

PEPA could be used as it is to model epidemiological systems. However, some of the issues which prevent the model being fully expressed or analysed can be handled better.

Getting the mean of simulations for a model with timed events

The analysis of the measles model in section 6.4 was limited by the fact that the mean of different stochastic simulations would not have provided meaningful results. Indeed, because timed events, and in this case season changes, do not happen exactly at the same time from one simulation to the next, a simple average of the different runs would result in intervals where different simulations are in different seasons. In the case of measles, increasing the population sizes decreases the stochasticity, and allows results that would be reasonably close to the average behaviour using only one simulation, but this is not the case in general. Indeed, the smaller the population size, the more stochasticity has an influence on the behaviour of the system, and as a result, individual simulations will show greater differences with one another.

A solution would be to align all the points where the behaviour changes, and extend or reduce the length of the intervals so they have the same duration. Consider an event that must happen exactly t_{diff} time steps after $t = t_0$. In some simulations, the event might happen at time $t = t_s$, which might occur after or before t_{diff}. In order to normalise the duration of the event in every simulation, the time at which each data point is recorded is updated in order to be equally

distributed between t_0 and t_{diff}. For each data point di_t recorded at time t, the new time associated with it is:

$$t' = \frac{(t_{diff} - t_0)}{(t_s - t_0)} \times (t - t_0) + t_0$$

$\frac{(t_{diff} - t_0)}{(t_s - t_0)}$ represents the new interval between data points for the event. $t - t_0$ represent the number of time steps since the start of the event. The final $+t_0$ term simply adds the time steps that happened before the event started.

For example, consider the case where a winter season starts at t_0=6, and finishes at t_1=11 for simulation 1, and t_2=13 for simulation 2, and the season lasts until t_{diff}=12 on average. We want to force the season to finish at t=12 in both cases. This implies that the winter has to be extended in the first simulation and shortened in the second. Assuming the data on the population size is gathered at each time step, the number of data points available is 5 and 7 respectively, denoted $d1_7$ to $d1_{11}$ for simulation 1 and $d2_7$ to $d2_{13}$ for simulation 2.

For simulation 1, the new interval between data points is $\frac{6}{(11-6)} = \frac{6}{5}$, expressing the fact that fewer data points are available than there are time steps between 6 and 12. This results in the points $(7.2, d1_7)$, $(8.4, d1_8)$, $(9.6, d1_9)$, $(10.8, d1_{10})$, $(12, d1_{11})$. The same principle can be applied to simulation 2, with in this case, intervals between data points equal to 6/7, representing the fact that more data points are available than there are time steps between 6 and 12.

This algorithm could be automated to the simulations, detect the switching points, and average the runs after having aligned the switching points.

Functional rates

Hillston and Kloul [45, 44] introduced a modified version of PEPA where rates can be a function of the state of a particular agent, while all the remaining grammar remains the same. The new

syntax for the rate is the following:

$$e ::= r|f|r \times f$$

where $f : 2^C \longrightarrow \mathbb{R}^*$, with 2^C the set of functions from C, the set of component types, to $\{0, 1\}$, and $\mathbb{R}^*$=$\{r|r \geq 0; r \in \mathbb{R}\} \cup \{\top\}$.

In epidemiology, rates can be a function of the number of individuals in a particular state. The new definition of the rate presented above can be used for that purpose. Indeed, all the agents in a particular subgroup can be tracked, and the rate can be adjusted depending on the number of agents in a particular state. For example, the definition of the birth rate as seen in Section 5.2 can be simplified dramatically by using the rate:

$$f(x_1, x_2, ..., x_n) = birth\_rate \times (1 - \sum_{i=1}^{n} x_i/K)$$

with the value x_i being 1 if the i^{th} agent of the subgroup is considered alive, 0 otherwise. The functional rate would replace an expression that already indirectly uses population level assumptions, while potentially being more intuitive that its PEPA model counterpart.

However, this novel definition of the rate cannot be used for practical purposes yet. Indeed, there is no tool that supports this new formalism, preventing any stochastic simulation analysis. Scalability is an issue as well. The new definition of the rate requires every agent to be tracked individually. Therefore, the performance of the algorithm might suffer from the size of biological systems, even more than it does with the currently-used definition of PEPA. Finally, using functional rates might defeat the purpose of using PEPA in the first place. Indeed, the objective in this thesis is to make assumptions about individual behaviour explicit in order to draw conclusions about the population level behaviour via analysis. Rates that depend on the size of the population already include some assumptions on the overall behaviour of the population. However, in the case of well-known features, the knowledge already available might this approximation and keep the model amenable to automated analysis.

Improving the efficiency of models with compartments

While it is now possible to split the population into different compartments, the resulting PEPA model is largely inefficient in the sense that the tools used to analyse it do not take into account the fact that some of the components belonging to different compartments might have a similar behaviour.

This is due to the fact that PEPA was not inherently conceived to allow compartments, and neither were the tools build to analyse models in PEPA. Two evolutions are possible to improve the efficiency of these models. The first one would be to develop a tool which would take into account the similarities between components in different compartments. Whether it is possible or not is still to be determined. The second option is to design a new process algebra based on PEPA, which would allow the possibility of having different compartments, and whose tools are conceived with this new feature in mind. The approach proposed by Galpin [32] described in section 7.1 might provide the required theory.

Helping the epidemiologists

While PEPA is based on a fairly simple principle, its syntax might look daunting to epidemiologists more used to handling ODEs. Furthermore, some of the methods presented in this thesis are complex and may easily lead to mistakes in the model, despite the use of tools to help detect and correct errors.

In order to overcome these issues, software that would hide the PEPA file, could be conceived, in which a basic understanding of the principle of components undertaking actions would be sufficient to generate PEPA models. The complex terms, such as density dependent transmission, birth and

death, or seasonality could also be easily added without having to understand the concept and expression behind the feature. For example, Harel statecharts [39] could be used as a graphical representation of the model, which would then be translated to a valid PEPA model.

8.5 General conclusion

Overall, PEPA has proven to be a useful tool that could be used to model biological systems. In the systems studied in this thesis, it has successfully described the population and disease behaviour, and when analysis could be performed, it has reproduced the results obtained in the literature. It has also been shown how the tools could be used to further the analysis of the model, with potential additional biological knowledge to be gained.

Over the course of this thesis, it has also been noted that some features could not be naturally described in PEPA, and artificial solutions had to be provided to represented them in the model. The size of the studied population and the complexity of the models have also proven to be a challenge as some of the existing tools could not handle some of the developed models.

Bibliography

[1] G. Anderson, S. Leary, D. Williamson, R. Titball, S. Welkos, P. Worsham, and A. Friedlander. Recombinant V antigen protects mice against pneumonic and bubonic plague caused by F1-capsule-positive and-negative strains of Yersinia Pestis. *Infection and Immunity*, 64(11):4580–4585, 1996.

[2] A. Argent-Katwala and J. T. Bradley. PEPA queues: Capturing customer behaviour in queueing networks. *Electr. Notes Theor. Comput. Sci.*, 190(3):3–25, 2007.

[3] R. K. Bangert and C. N. Slobodchikoff. Conservation of prairie dog ecosystem engineering may support arthropod beta and gamma diversity. *Journal of Arid Environments*, 67(1):100 – 115, 2006.

[4] C. Beauchemin, J. Samuel, and J. Tuszynski. A simple cellular automaton model for influenza A viral infections. *Journal of Theoretical Biology*, 232(2):223 – 234, 2005.

[5] M. Begon, M. Bennet, R. Bowers, N. French, S. Hazel, and J. Turner. A clarification of transmission terms in host-microparasite models: numbers, densities and areas. *Epidemiology and Infection*, 129(01):147–153, 2002.

[6] M. Begon, C. R. Townsend, and J. L. Harper. *Ecology : individuals, populations and communities*. Oxford : Blackwell Scientific.

[7] S. Benkirane, J. Hillston, C. McCaig, R. Norman, and C. Shankland. Improved continuous approximation of PEPA models through epidemiological examples. *Electron. Notes Theor. Comput. Sci.*, 229(1):59–74, 2009.

[8] A. A. Berryman. The origins and evolution of predator-prey theory. *the Ecological Society of America*, 73(5):1530–1535, 1992.

[9] O. N. Bjørnstad, B. F. Finkenstädt, and B. T. Grenfell. Dynamics of measles epidemics: estimating scaling of transmission rates using a time series SIR model. *Ecological Monographs*, 72(2):169–184, 2002.

[10] C. Bodei. A control flow analysis for beta-binders with and without static compartments. *Theoretical Computer Science*, 410(33-34):3110 – 3127, 2009.

[11] B. Bolker and B. Grenfell. Space, persistence and dynamics of measles epidemics. *Philosophical Transactions of the Royal Society of London - Series B: Biological Sciences*, 348(1325):309–320, 1995.

[12] G. E. P. Box and N. R. Draper. *Empirical Model-Building and Response Surfaces*. Wiley, 1987.

[13] J. T. Bradley, S. T. Gilmore, and J. Hillston. Analysing distributed internet worm attacks using continuous state-space approximation of process algebra models. *Journal of Computer and System Sciences*, 74(6):1013 – 1032, 2008.

[14] L. Cardelli. On process rate semantics. *Theoretical Computer Science*, 391:190–215, 2008.

[15] J. Cathelyn, S. Crosbyt, W. Lathem, W. Goldman, and V. Miller. RovA, a global regulator of Yersinia Pestis, specifically required for bubonic plague. *National Academy of Sciences*, 103(106):13514–13519, 2006.

[16] Centers for Disease Control and Prevention. Measles outbreaks. `http://www.cdc.gov/measles/outbreaks.html`, 2009.

[17] F. Ciocchetta and J. Hillston. Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theor. Comput. Sci.*, 410(33-34):3065–3084, 2009.

[18] F. Ciocchetta and J. Hillston. Bio-PEPA for epidemiological models. *Electronic Notes in Theoretical Computer Science*, 261:43 – 69, 2010. Proceedings of the Fourth International Workshop on the Practical Application of Stochastic Modelling (PASM 2009).

[19] A. Clark, S. Gillmore, H. Jane, and P. Kemper. Verification and testing of biological models. In *Proceedings Winter Simulation Conference (WSC 2010), Dec 5-8 2010, Baltimore, Maryland, USA (to appear).*

[20] P. Degano, D. Prandi, C. Priami, and P. Quaglia. Beta-binders for Biological Quantitative Experiments. *Electronic Notes in Theoretical Computer Science*, 164(3):101–117, Oct. 2006.

[21] R. J. Delahay, S. Langton, G. C. Smith, R. S. Clifton-Hadley, and C. L. Cheeseman. The spatio-temporal distribution of Mycobacterium Bovis (bovine tuberculosis) infection in a high-density badger population. *Journal of Animal Ecology*, 69(3):428–441, 2000.

[22] T. Duke and C. S. Mgone. Measles: not just another viral exanthem. *The Lancet*, 361:763–773, 2003.

[23] R. Durrett. *Probability: Theory and Examples.* Cambridge series in statistical and probabilistic mathematics, 2010.

[24] R. Eisen, A. Wilder, S. Bearden, J. Montenieri, and K. Gage. Early-phase transmission of Yersinia Pestis by unblocked Xenopsylla Cheopis (Siphonaptera: Pulicidae) is as efficient as transmission by blocked fleas. *Journal of tropical medicine and hygiene*, 44(4):678–82, 2007.

[25] J. F. Enders and S. L. Katz. Immunization of children with a live attenuated measles vaccine. *American journal of diseases of children*, 98(605), 1959.

[26] EnvironmentalChemistry.com. Prairie dogs: A threatened species or public health nuisance? `http://environmentalchemistry.com/yogi/environmental/200703prairiedogs.html`, March 2007.

[27] P. E. Fine and J. A. Clarkson. Measles in England and Wales–I: An analysis of factors underlying seasonal patterns. *International Journal of Epidemiology*, 11(1):5–14, 1982.

[28] P. E. M. Fine and J. A. Clarkson. Measles in England and Wales–II: The impact of the measles vaccination programme on the distribution of immunity in the population. *International Journal of Epidemiology*, 11:15–25, 1982.

[29] B. F. Finkenstädt, M. Keeling, and B. T. Grenfell. Patterns of density dependence in measles dynamics. *Proceedings of the Royal Society B.*, 265:753–762, 1998.

[30] D. A. Focks, E. Daniels, D. G. Haile, and J. E. Keesling. A simulation model of the epidemiology of urban dengue fever: Literature analysis, model development, preliminary validation, and samples of simulation results. *The American Journal of Tropical Medicine and Hygiene*, 53(5):489–506, 1995.

[31] J. M. Fourneau, L. Kloul, and F. Valois. Performance modelling of hierarchical cellular networks using PEPA. *Performance Evaluation*, 50(2-3):83 – 99, 2002.

[32] V. Galpin. Modelling network performance with a spatial stochastic process algebra. In *AINA '09: Proceedings of the 2009 International Conference on Advanced Information Networking and Applications*, pages 41–49, Washington, DC, USA, 2009. IEEE Computer Society.

[33] S. Gee, S. Cotter, and D. O'Flanagan. Spotlight on measles 2010: Measles outbreak in Ireland 2009-2010. `http://www.eurosurveillance.org/images/dynamic/EE/V15N09/art19500.pdf`, 2010.

[34] S. Gilmore, J. Hillston, R. Holton, and M. Rettelbach. Specifications in stochastic process algebra for a robot control problem. *International Journal of Production Research*, 34:1065–1080, 1995.

[35] S. Gilmore, J. Hillston, L. Kloul, and M. Ribaudo. PEPA nets: a structured performance modelling formalism. *Performance Evaluation*, 54(2):79 – 104, 2003.

[36] B. M. Greenwood, B. Khalifa, C. Whitty, and G. Targett. Malaria. *Lancet*, 365:1487–1498, April 2005.

[37] C. M. Grinstead and J. L. Snell. *Introduction to Probability, second revised edition.* American Mathematical Society, 1997.

[38] M. L. Guerriero, C. Priami, and A. Romanel. Modelling static biological compartments with beta-binders. In *Proceedings of the 2nd international conference on Algebraic biology*, AB'07, pages 247–261, Berlin, Heidelberg, 2007. Springer-Verlag.

[39] D. Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8(3):231 – 274, 1987.

[40] R. Hayden and J. T. Bradley. A fluid analysis framework for a markovian process algebra. *Theoretical Computer Science*, 411:2260–2297, May 2010.

[41] R. A. Hayden and J. T. Bradley. A fluid analysis framework for a markovian process algebra. *Theoretical Computer Science*, 411(22-24):2260 – 2297, 2010.

[42] J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.

[43] J. Hillston. Fluid Flow Approximation of PEPA models. In *QEST'05, Proceedings of the 2nd International Conference on Quantitative Evaluation of Systems*, pages 33–42. IEEE Computer Society Press, Torino, September 2005.

[44] J. Hillston and L. Kloul. A function-equivalent components based simplification technique for PEPA models. *Lecture notes in computer science*, 4054:16–30, 2006.

[45] J. Hillston and L. Kloul. Formal techniques for performance analysis: blending SAN and PEPA. *Formal Aspects of Computing*, 19(1):3–33, 2007.

[46] C. Hoare. Communicating sequential processes. *Communications of the ACM*, 21:666–677, 1978.

[47] R. Holton. A PEPA specification of an industrial production cell. In *The Computer Journal*, pages 542–551, 1995.

[48] J. Hyman, J. Li, and A. Stanley. The initialization and sensitivity of multigroup models for the transmission of HIV. *Journal of theoretical Biology*, 208:227–249, 2001.

[49] java.net. Java compiler compiler [tm] (javacc [tm]) - the Java parser generator. https://javacc.dev.java.net/, 2010.

[50] M. J. Keeling and C. A. Gilligan. Bubonic plague: a metapopulation model of a zoonosis. *Proceedings of the Royal Society of London Series B-Biological Sciences*, 267:2219–2230, 2000.

[51] W. Kermack and A. McKendrick. Contributions to the mathematical theory of epidemics. *Proceedings of the Royal Society of London A*, 115:700–721, 1927.

[52] A. Korobeinikov and G. Wake. Lyapunov functions and global stability for SIR, SIRS and SIS epidemiologicals models. *Applied Mathematics Letters*, 15(8):955–960, 2002.

[53] M. Li, M. James, and P. Van Den Driessche. Global stability of SEIRS models in epidemiology. *Canadian applied mathematics quarterly*, 7(4):409–425, 1999.

[54] M. Y. Li, J. R. Graef, L. Wang, and J. Karsai. Global dynamics of a SEIR model with varying total population size. *Mathematical Biosciences*, 160(2):191 – 213, 1999.

[55] M. Y. Li and J. S. Muldowney. Global stability for the SEIR model in epidemiology. *Mathematical Biosciences*, 125(2):155 – 164, 1995.

[56] J. Lloyd-Smith, S. Schreiber, P. Kopp, and W. Getz. Superspreading and the effect of individual variation on disease emergence. *Nature*, 438(17):355–359, 2005.

[57] C. McCaig. *From individuals to populations: changing scale in process algebra models of biological systems*. PhD thesis, University of Stirling, 2008. Available from `www.cs.stir.ac.uk/~cmc/thesis.ps`.

[58] C. McCaig, R. Norman, and C. Shankland. From individuals to populations: A symbolic process algebra approach to epidemiology. *Mathematics in Computer Science*.

[59] C. McCaig, R. Norman, and C. Shankland. Density dependent growth in individual based models. 2006. Unpublished Draft.

[60] C. McCaig, R. Norman, and C. Shankland. Investigating population level transmission terms in individual based models of infectious disease spread. 2006. In prep.

[61] C. McCaig, R. Norman, and C. Shankland. An algorithm for deriving mean field equations from large process algebra models. Technical Report 175, University of Stirling, 2008.

[62] Measles Initiative. Mortality reduction. `http://tinyurl.com/6dqwwhq`, 2009.

[63] A. Merlin and G. Hains. A generic cost model for concurrent and data-parallel meta-computing. *Electronic Notes in Theoretical Computer Science*, 128(6):3 – 19, 2005.

[64] L. A. Mermel. Preventing the spread of influenza A H1N1 2009 to health-care workers. *The Lancet Infectious Diseases*, 9:723–724, 2009.

[65] D. Miller. Frequency of complications of measles, 1963. *British Medical Journal*, 2:75–78, 1964.

[66] R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[67] R. Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press, 1st edition, June 1999.

[68] Mount Sinai Hospital Department of Microbiology. Faq: Methods of diseases transmission. `http://microbiology.mtsinai.on.ca/faq/transmission.shtml`, March 2007.

[69] National Health Care. Measles outbreaks in the north-west and London. `http://www.immunisation.nhs.uk/Library/News/Measles_outbreaks_in_the_north-west_and_London`, 2010.

[70] R. Norman and C. Shankland. Developing the use of process algebra in the derivation and analysis of mathematical models of infectious disease. In *Computer Aided Systems Theory - EUROCAST 2003*, volume 2809 of *Lecture Notes in Computer Science*, pages 404–414. Springer-Verlag, 2003.

[71] R. Perry and N. Halsey. The clinical significance of measles: A review. *Journal of Infectious Diseases*, 189(1):S4–S16, 2004.

[72] J. D. Poland, A. M. Barnes, and J. J. Herman. Human bubonic plague from exposure to a naturally infected wild carnivore. *American Journal of Epidemiology*, 97(5):332–337, 1973.

[73] A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. *Pacific Symposium on Biocomputing*, 6:459 – 470, 2001.

[74] W. Reisig. *Petri nets: an introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[75] J. Robertson. *Modelling the Bubonic Plague amongst Prairie Dog Burrows*. University of Stirling, School of natural sciences, 2010.

[76] R. Scott Braithwaite, C. Omokaro, A. C. Justice, K. Nucifora, and M. S. Roberts. Can broader diffusion of value-based insurance design increase benefits from US health care without increasing costs? Evidence from a computer simulation model. *PLoS Med*, 7(2), 2010.

[77] R. Scott Braithwaite, M. S. Robert, C. C. H. Chan, M. B. Goetz, C. L. Gibert, M. C. Rodriguez-Barradas, S. Shechter, K. Nucifora, R. Koppenhaver, and A. C. Justice. Influence of alternative thresholds for initiating HIV treatment on quality-adjusted life expectancy: A decision model. *Annals of Internal Medicine*, 148(3), 02 2008.

[78] R. Semba and M. Bloem. Measles blindness. *Survey of ophthalmology*, 49(2):243–255, 2004.

[79] Smithsonian National Zoological Park. Facts: Black-tailed prairie dog. `http://nationalzoo.si.edu/Animals/NorthAmerica/Facts/fact-pdog.cfm`, 2002.

[80] A. Stefanek. Grouped PEPA analyzer. `http://www.doc.ic.ac.uk/~as1005/index.cgi?gpa`, March 2010.

[81] A. Stefanek, R. Hayden, and J. T. Bradley. A new tool for the performance analysis of massively parallel computer systems. *Electronic Proceedings in Theoretical Computer Science*, 28:159–181, 2010.

[82] A. Stefanek, R. A. Hayden, and J. T. Bradley. Hybrid analysis of large scale PEPA models. In *9th Workshop on Process Algebra and Stochastically Timed Activities*, pages 29–36, 2010.

[83] The Medical News. Measles history. `http://www.news-medical.net/health/Measles-History.aspx`.

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

[85] M. Tribastone, A. Duguid, and S. Gilmore. The PEPA eclipse plugin. *SIGMETRICS Perform. Eval. Rev.*, 36(4):28–33, 2009.

[86] A. Tsoularis and J. Wallace. Analysis of logistic growth models. *Mathematical Biosciences*, 179(1):21 – 55, 2002.

[87] S. R. Ubico, G. O. Maupin, K. A. Gaferstone, and R. G. McLean. A plague epizootic in the while-tailed prairie dogs (*Cynomys leucurus*) of Meeteetse, Wyoming. *Journal of Wildlife Diseases*, 24(3):399–406, 1988.

[88] UNICEF. *Measles: Mortality Reduction and Regional Elimination*, 2001.

[89] University of Cambridge. Pathogen population dynamics. `http://www.zoo.cam.ac.uk/zoostaff/grenfell/measles.htm`, 2002.

[90] U.S. Fish & Wildlife Service. Black-tailed prairie dog. `http://www.fws.gov/mountain-prairie/species/mammals/btprairiedog/`, December 2009.

[91] J. N. Wasserheit and S. O. Aral. The dynamic topology of sexually transmitted disease epidemics: Implications for prevention strategies. *Journal of Infectious Diseases*, 174(Supplement 2):S201–S213, 1996.

[92] C. Webb, C. Brooks, K. Gage, and M. Antolin. Classic flea-borne transmission does not drive plague epizootics in prairie dogs. *National Academy of Sciences*, 103(16):6236–6241, 2006.

[93] R. Willox, B. Grammaticos, A. S. Carstea, and A. Ramani. Epidemic dynamics: discrete-time and cellular automaton models. *Physica A: Statistical Mechanics and its Applications*, 328(1-2):13 – 22, 2003.

[94] Wolfram Research. Wolfram Mathematica 7. http://www.wolfram.com/products/mathematica/index.html, 2010.

[95] World Health Organization. The world health report 2004. `http://www.who.int/whr/2004/annex/topic/en/annex_2_en.pdf`, 2004.

[96] World Health Organization. Plague. `http://www.who.int/mediacentre/factsheets/fs267/en/`, February 2005.