PROCESS ALGEBRA WITH LAYERS: A LANGUAGE FOR MULTI-SCALE INTEGRATION MODELLING

Erin Gemma Scott

Thesis submitted for the degree of
Doctor of Philosophy

Computing Science and Mathematics
University of Stirling

April 2016

ABSTRACT

Multi-scale modelling and analysis is becoming increasingly important and relevant. Analysis of the emergent properties from the interactions between scales of multi-scale systems is important to aid in solutions. There is no universally adopted theoretical/computational framework or language for the construction of multi-scale models. Most modelling approaches are specific to the problem that they are addressing and use a hybrid combination of modelling languages to model specific scales. This thesis addresses if process algebra can offer a unique opportunity in the definition and analysis of multi-scale models.

In this thesis the generic Process Algebra with Layers (PAL) is defined: a language for multi-scale integration modelling. This work highlights the potential of process algebra to model multi-scale systems. PAL was designed based on features and challenges found from modelling a multi-scale system in an existing process algebra. The unique features of PAL are the layers: Population and Organism. The novel language modularises the spatial scales of the system into layers, therefore, modularising the detail of each scale. An Organism can represent a molecule, organelle, cell, tissue, organ or any organism. An Organism is described by internal species. An internal species, dependent on the scale of the Organism, can also represent a molecule, organelle, cell, tissue, organ or any organism. Populations hold specific types of Organism, for example, life stages, cell phases, infectious states and many more. The Population and Organism layers are integrated through mirrored actions.

This novel language allows the clear definition of scales and interactions within and between these scales in one model. PAL can be applied to define a variety of multi-scale systems. PAL has been applied to two unrelated multi-scale system case studies to highlight the advantages of the generic novel language. Firstly the effects of ocean acidification on the life stages of the Pacific oyster. Secondly the effects of DNA damage from cancer treatment on the length of a cell cycle and cell population growth.

ACKNOWLEDGEMENTS

I would now like to thank the people that made up the "layers" of my life during this project.

I wish to acknowledge and thank my brilliant supervisor Professor Carron Shankland for her constant support, enthusiasm, encouragement, patience and guidance. She has taught me that every question is an open ended opportunity. I also wish to thank my second supervisor Dr Andy Hoyle for his support and mathematical knowledge. He had the unique ability to answer any of my questions no matter how large or small with a simple drawing of a graph. Their supervision has been greatly appreciated and the work presented in this thesis was greatly improved thanks to their valuable advice.

I express my thanks to my SICSA supervisor Professor Jane Hillston for her knowledge and assistance in the formulation of syntax and semantics of a process algebra language. I am also grateful to Dr Allan Clarke for his knowledge and advice in Bio-PEPA and the Bio-PEPA plug-in.

My thanks also go to Dr Andrea Degasperi for his knowledge and assistance with his multi-scale process algebra language. Thanks also to Mr James Nicol for providing the radiation data for the cell cycle model and Dr Rene Robert for providing the data for the Larva model.

Further thanks go to the members of the administrative team in the Computing Science and Mathematics Department: Grace, Linda and Gemma.

Finally, my heartfelt thanks go to my family and friends for their continuous love and support. To Evelyn Humphrey for your praise and giving me a great many laughs throughout the years. To Sadie Flanagan for your cheer, continuous questions and genuine interest in my work. To Aunt Frances and Uncle Karl for all your words of encouragement, advice and hugs from Pippa the salty sea dog. Finally to my Mum and Dad, Mary and Colin, for all your help being there for me through the highs and lows of this process. You both are an inspiration to me and are the ones who have encouraged me to reach a point in my life I never thought possible. This thesis is dedicated to you.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION AND LITERATURE REVIEW

1.1 INTRODUCTION

Mathematical and computational multi-scale modelling of multi-scale systems is now commonplace and indeed essential to many investigations. Analysis of the emergent properties from the interactions between scales of multi-scale systems is important to aid in solutions. There is no universally adopted theoretical/computational framework or language for the construction of multi-scale models. Most multi-scale models are specific to the problem they are addressing and are defined by integrated scales that are modelled in different mathematical and computational languages [22, 75]. A common multi-scale hybrid model example is the Met Office climate prediction model [39] which couples the atmospheric and oceanic scales together in one model. The model utilises different mathematical approaches to describe the different scales and is specific to the problem of climate change forecasting. These hybrid models make the structure and the analysis of the model difficult as the scales are defined in separate models. The modeller has to be knowledgeable in integration techniques to link the models together.

There are particular problems associated with multi-scale modelling. Firstly, choosing the most appropriate scale, or level of abstraction, especially given that there may be different important characteristics at each scale. Secondly, the integration of the scales: this is important to ensure the characteristics in the model are not sacrificed in any way to accommodate the integration. Noble [53] and Allen et al [10] from the fields of systems biology and marine ecological modelling respectively, state there are specific features to consider in multi-scale modelling. These features are: abstraction, descriptive, integrative, explanatory and generic.

Abstraction: The balance in the level of detail needs to be established in order to ensure efficiency of model simulations. As indicted by Allen et al [10] ecological models should be constructed at an appropriate level of detail to address the hypothesis being tested and the data available to validate it. This approach is also supported by Noble [53] who advocates that computational models are constructed and tested at levels where the most knowledge of the system is found. This feature advocates the middle-out strategy described below.

Descriptive: The model has to reflect the system accurately and represent the available data appropriately [10]. This feature is essential for the biological insight the model will produce. This will further aid communication of the model between disciplines. It is important that enough knowledge and relevant data is available to make the model realistic.

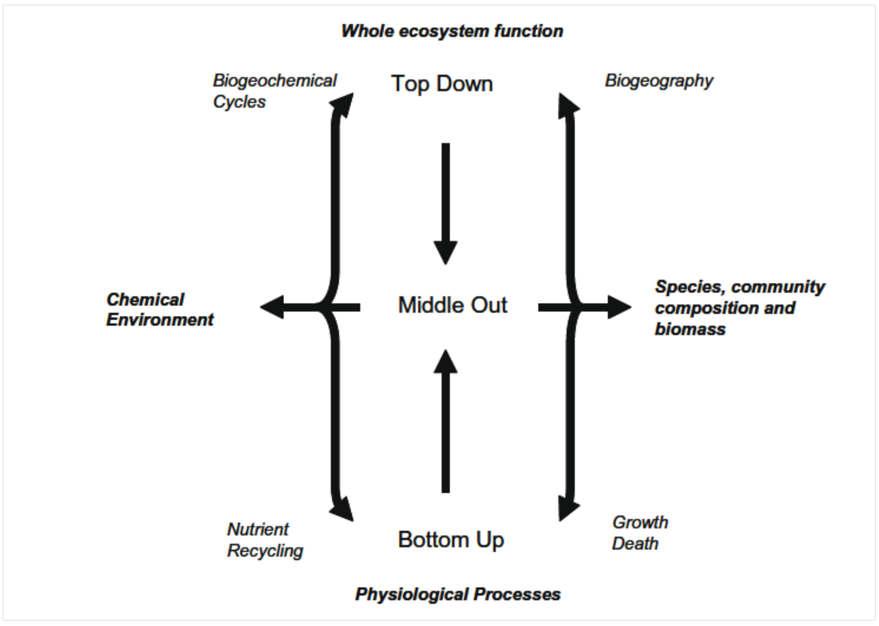


Figure 1.1: Strategies for modelling marine ecosystems [10].

Integrative: The model has to demonstrate how different elements of the system interact [10]. There will be many types of interactions. These interactions will be between local internal and external components and these affect thresholds and thereafter these altered thresholds interact with the components. A multi-scale model has different levels of scales of interactions. Knowledge of the importance of these multi-scale interactions is essential in order to create a complex and efficient model. Interactions that are not relevant to the specific problem need to be excluded in order to avoid extraneous detail that will make the model simulations less efficient.

Explanatory: The model should provide predictions and insights of the system [10]. It should allow analysis by different techniques and have the ability to be exported to other applications for further analysis. This will allow users of the model to use their preferred model and analysis tool.

Generic: The model should be generic and re-usable to allow it to be applied to different systems [53]. The model should be able to adapt to the data that is available, for example, the time period of experiments will vary from days to years [10]. The description of generic by Noble [53] and Allen et al [10] is interpreted as "generally applicable". Hence, when generic is used in this thesis it means "generally applicable".

There are three well known strategies for constructing multi-scale models which are utilised within the fields of systems biology and marine ecological modelling: Top-down, Bottom-up, and Middle-out [10, 53]. A schematic of these strategies within a marine ecosystem is shown in Figure 1.1.

The **Top-down** strategy involves looking at a whole system and its high level functions and then works downward to envisage low level parts of the system that are less well known [53]. This strategy starts by modelling of high-level physiological systems (such as the circulatory, endocrine or immune systems) and then works downwards progressively describing in more detail the lower-level mechanisms (such as the molecules such as haemoglobin located in red blood cells or the signalling molecules of immune T-helper cells). The problem in this approach is that it tries to simplify everything to basic principles, understanding the components is important but not sufficient for a systems level understanding [53]. Allen et al [10] further suggests it can suffer from the fundamental weakness of being over general and therefore not completely useful for targeted questions.

The **Bottom-up** strategy involves starting at a specific low level for example the physiology and behaviour of individual species, cells, molecules, genes or atoms. The problem with this approach is how identifiable the behaviour of a cell can be linked to a high level function. For example, a vast assembly of nerve cells can give you joy or sorrow [53]. In ecological modelling the problem of this approach is that only a few species can be characterised physiologically and mechanistically well enough in experiments to make such models. Investigating lower levels in an ecological system needs knowledge of higher levels of organisation [10].

The **Middle-out** strategy involves building the model at a level that has sufficient data and knowledge. Middle-out is advocated by Noble [53] as it is a simple and pragmatic strategy. Essentially since all levels can be starting points for a causal chain, any level can be the starting point for building and resulting in a successful simulation. Allen et al [10] indicates that a balanced combination of all three strategies and focus on descriptive, integrative and explanatory features will result in a good model.

In this thesis Process Algebra with Layers (PAL) is defined. The design incorporates the multi-scale features discussed and follows the middle-out strategy.

1.2 BACKGROUND

There are many modelling approaches and languages utilised by modellers today. Many are designed to efficiently express and analyse a model at a specific time and spatial scale [75]. To achieve multi-scale modelling across these scales certain modelling approaches and languages have been combined. These hybrid multi-scale models are usually specific to the problem they are addressing. The following languages and approaches are given as an indication of existing options and to give some historical context.

1.3 MATHEMATICAL MODELLING

1.3.1 *Ordinary Differential Equations*

Ordinary differential equations (ODEs) are the most traditional mathematical modelling technique to describe and quantitatively analyse systems in several research areas including systems biology and ecology [62, 40]. A model is presented as a system of equations. Each equation describes the rate of change of a continuous variable [44]. ODEs solve a continuous, deterministic regime in which the variables are subject to continuous change. The solution of the system of equations provides an average variation of the concentrations of the variables that are included [62]. The limitations of using ODEs is that they do not capture variability as they use a continuous deterministic approach that shows the most likely behaviour of the system. The advantages of this is that the run time of model simulations are fast and are less computational expensive with respect to other modelling approaches.

ODEs are frequently used in hybrid multi-scale models to describe a specific scale of the model and this approach is rarely used alone in multi-scale modelling (an example of this is given in Section 1.3.3). They are specifically used in hybrid multi-scale models to define intracellular, biochemical and physiological scales. They can effectively describe molecular, chemical and metabolic interactions. For example Powathil et al [61] in their hybrid multi-scale mathematical model of cancer cell growth use a system of ODEs to describe the intracellular proteins that drive the cell growth and division. The cellular scale in the model is described by the Cellular automata approach (described in Section 1.4.1.1).

Powathil et al [61] state to account for the issue of non-variability in the ODEs model and therefore not to get a synchronous population of cells they introduce a random growth distribution variable to each individual cell. Therefore the cell cycle length varies with the population. This has limitations as other variations in the intracellular scale are not considered and that these may be important in respect to the emergent properties of this multi-scale system.

1.3.2 *Dynamic Energy Budget Modelling*

ODEs modelling gives the foundations for the Dynamic Energy Budget (DEB) [40] modelling approach. The multi-scale problem being explored in this thesis includes the challenge of modelling the physiological scale of a marine organism. DEB theory describes an organism's physiology in an abstracted and descriptive approach. DEB theory is a mechanistic, mathematical, physiological modelling theory. It describes in a generic way an organism's physiology and how it adapts to its environment. From its inception in the 1970s to the present, DEB theory is popularly utilised in a large number of published case studies (over 425) of biological

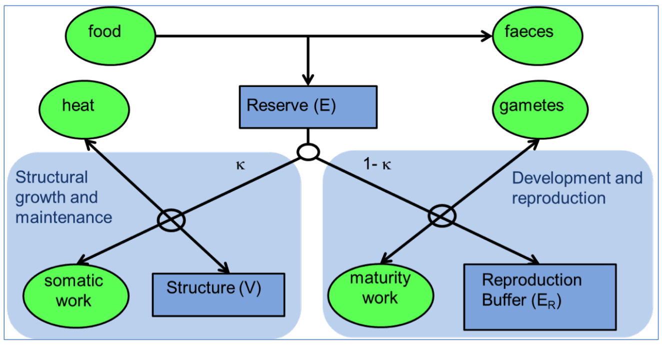

food
faeces
heat
Reserve (E)
gametes
κ
1- κ
Structural growth and maintenance
Development and reproduction
somatic work
Structure (V)
maturity work
Reproduction Buffer (E_R)

Figure 1.2: Generic DEB model schematic for a multicellular organism. Circles represent sources and sinks, squares represent state variables and arrows indicate metabolic processes.

systems in a variety of journals [41]. This generic theory of energy budgets can be used to describe the uptake and use of substrates (food) in all organisms. It is a generic theory because it assumes common physiological processes across species and life stages via a set of common DEB parameters; the only differences in species lying in the differences in these parameters. A schematic diagram of a general multicellular DEB model adapted from Kooijman [40] is given in Figure 1.2.

A basic DEB model consists of two differential equations to describe the behaviour of the two state variables: the Reserve (E) and the Structural Volume (V). A DEB model assumes that assimilated energy is first stored in a reserve and this reserve is utilised to fuel other metabolic processes. The allocation of the energy to be utilised is controlled by the parameter κ: its value represents which process has priority over the energy. These metabolic processes include maintenance, growth, development and reproduction. The complexity and sophistication of the model arises from interrelationships between the processes. These processes have many interrelationships, for example, the assimilation process has an impact on the utilisation process.

Models also have forcing variables; for example, in Figure 1.2 the forcing variables are food and heat. The model usually contains sources and sinks, for example, food is a source and gametes are a sink as shown in Figure 1.2. More realism can be included in the model by adding further state variables, for example, to describe reproduction and development.

The advantages of this approach is it can describe an organism's physiology in a generic and abstract way and does not lose any important physiological detail. This is shown by the numerous DEB models comparisons with wet laboratory data [41]. The limitations of this approach is that as explained previously they do not capture variability as they use a continuous deterministic approach that shows average system behaviour of the energy budgets. Another disadvantage is that this method does not describe the population view of an organism. This is important to understand how changes in an organism's physiology impacts the overall population.

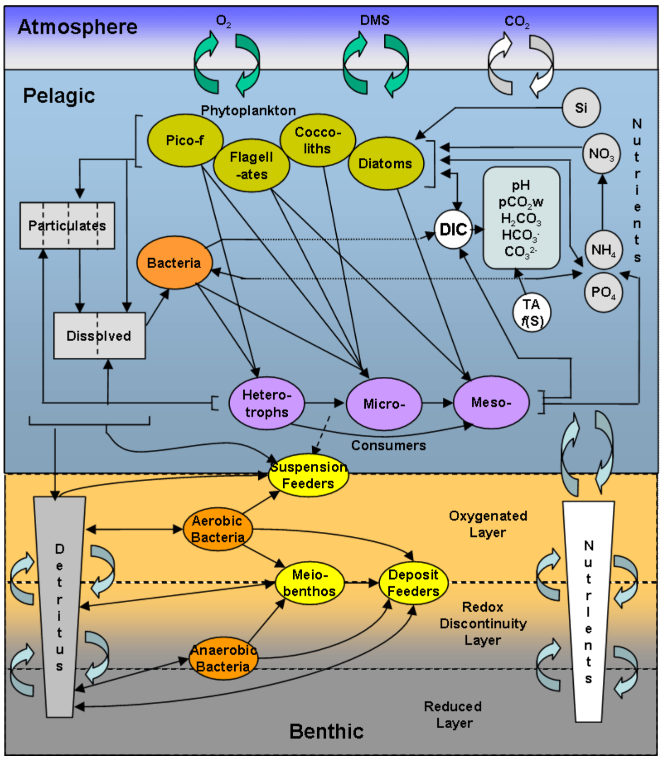


Figure 1.3: European Regional Seas Ecosystem Model Example [3].

1.3.3 *European Regional Seas Ecosystem Model*

The European Regional Seas Ecosystem Model (ERSEM) [17] was developed to give rise to a generic shelf sea ecosystem model which simulates the cycling of organic carbon, nutrients and oxygen over a seasonal cycle initially applied to the North Sea. ERSEM is represented by a group of parallel ODEs, solved as an open ended recursive system using continuous system simulation techniques. This approach is an example of a compartmental model using ODEs of varying compartment size therefore, this model contains modules describing different parts of the multi-scale system. The modules are specifically looking at a particular set of state variables, for example some of the modules describe, nutrient dynamics, mesozooplankton and top predators. Figure 1.3 shows a schematic of a ERSEM illustrating the sea water column cycles and the state variables they include.

The state variables include organisms, elements and derived carbonate system parameters from the pelagic and benthic sea water column cycles. The organisms are described by the ratios of the state variable elements (Carbon, Nitrogen, Phosphorus, Silicon and Oxygen) over a seasonal cycle. The organism is stated to be a standard organism which is defined by a group of state variables and the organism can be thought of as essentially three types; producer, consumer and decomposer, each of which may be defined as having a standard set of processes [17]. This standard organism is shown in Figure 1.4.

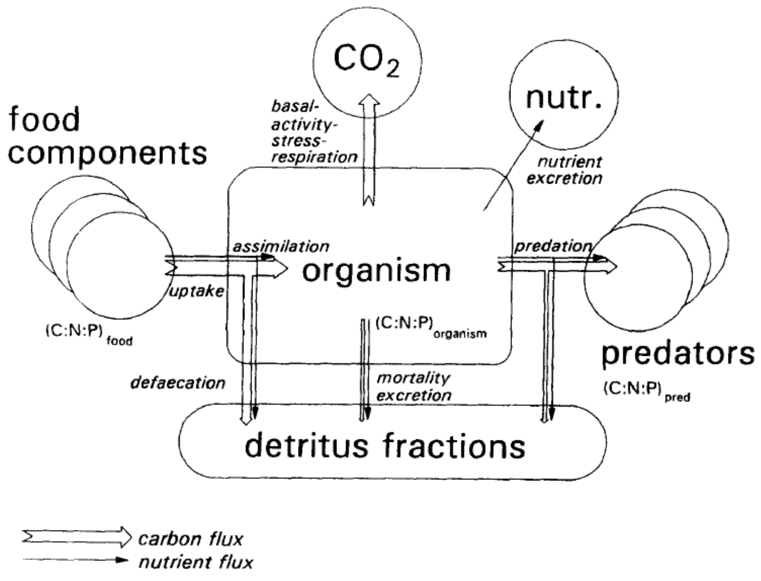


Figure 1.4: Standard organism used as a base in the ERSEM [13].

The ERSEM concentrates on the ecosystem surrounding the organism such as the suspended particles in the water column. The main focus of this approach is on the cycling of organic carbon, nutrients and oxygen over a seasonal cycle. The advantage of this approach is it can be used to explore oceans physical features and nutrient conditions impacting fish. This approach essentially treats the organisms as black boxes concentrating on the inputs and outputs of the state variable elements. This is a disadvantage, as it does not describe the organisms internal scales therefore when changes occur in the input and output of the organism the model would not describe how this impacts the organisms internal scales, for example, their physiology.

1.4 COMPUTATIONAL MODELLING

1.4.1 *Individual-Based Modelling*

1.4.1.1 *Cellular automata*

Cellular automata (CA) [54] are discrete dynamic models that consist of a grid of cells with a finite number of states. At each time step CA rules are applied to update the state of each cell in the grid according to the states of cells in its local neighbourhood. As CA is a simplistic technique it is able to be applied to many different problems, such as tumour growth and disease spread. Machado et al [44] states that due to CA spatial features CA main application are related to molecular dynamics and cellular population dynamics.

They are frequently used to build hybrid multi-scale models, specifically they are used to define the population scale as they can effectively describe interactions between individuals or cells. For example Powathil et al [61] uses CA to describe the cellular population dynamics in conjunction with the ODEs approach to model cell growth. An advantage of CA is its ability to produce discrete simulations of the model therefore allowing the analysis of the system's variability. The major drawback of CA is its high computational requirements, as it explicitly defines all cells and space, the processing time increases with the population size. ODEs cannot be derived from CA models and therefore ODEs simulation analysis cannot be performed. CA therefore restricts the user to using specific analysis tools.

1.4.1.2 *Agent-Based Modelling*

Agent-based models [64] (ABM) represent a system's individual autonomous agents and their behaviours. They are similar to CA apart from CA uses a grid, the agents within ABM move freely within the containing space. ABM describe agents that are unique; they have different characteristics from each other. These can be size, location, resource reserves and history. These agents can interact with each other and their environment locally; agents usually do not interact with all other agents but only their neighbours. This interaction can take place in a geographic space or in another sort of space like a network. Agents can be defined as any sort of entity which pursues a certain goal, for example an agent could be an organism searching for food or a business trying to generate profit. The agents are autonomous; they pursue their own objectives and act independently of each other. Adaptive behaviour is used by the agents to adapt their behaviour to the current states of other agents and of their environment.

The use of ABM similar to CA helps in the analysis of problems that concern emergence. ABM allow questions to be studied about how a system's behaviour arises from and what it is linked to. Further questions can be studied on the characteristics and behaviours of the individual agents. They are used to both look at what happens to the system because of what its individuals do and what happens to the individuals because of what the system does.

Similar to CA, ABM also are used to build hybrid multi-scale models. Specifically they are used to define the population scale as they can effectively describe interactions between individuals. For example Martin et al [47] used an Agent-based model to describe the organism population scale. They implemented a generic DEB-Agent-based model in NetLogo [76]. NetLogo is a software package which is used to implement and analyse ABM. Martin et al [47] used DEB theory (Described in Section 1.3.2) to describe the physiological scale of an organism within the ABM population. They state that basing ABM on standardised and well tested approaches such as DEB theory facilitates re-usability of ABM and their elements and further facilitates general insights from specific ABM. In general ABM are developed for specific questions, so therefore the structure and parameters defining the life history of organisms differ widely. The use of DEB theory as the basis of the model reduces this problem and facilitates broader insight from specific studies and comparison between species.

Martin et al [47] state that using ABM allows the inclusion of stochasticity and provides a framework to investigate the stochastic effects at a population level. Similar to CA, ABM has the drawback of high computational requirements, as it explicitly defines all agents and containing space, the processing time increases with the population size.

1.4.1.3 *Statistical methods: individuals to populations*

Statistical methods are used in individual based modelling to derive population models. The advantage of this approach is to create a simpler population model retaining detail on individual behaviour. This allows population analysis and the model is less computationally expensive to run. An example of this is shown in Boots et al [18] where they use stochastic analysis to derive mean field equations from individual based models. The mean field equations reduce the complex stochastic model to give an average population behaviour. They investigate disease virulence and the impact that different local and global contact of susceptible and infected individuals have on the virulence of the disease. The disadvantage of this approach is the model describes the individual's behaviour but does not include the individual's internal system. For example, this scale could be important in the case of the 'common cold'. The probability of infection is not exclusive to close contact with an infected individual but also the condition of the immune system of the susceptible individual.

McCaig et al [48] use an analytical approach to derive mean field equations from process algebra models (process algebra is discussed in Section 1.4.3). They derive a model describing HIV spread in the UK. This gives the advantage again of reducing a complex model to a simpler model but keeping the detail of the individual behaviour therefore giving a low cost for simulation analysis. The disadvantage of this approach is that it only describes individuals by their behaviour and does not include their internal system behaviour. HIV would have a more serious affect on a susceptible which is already immunologically comprised.

It is noted that the above examples are epidemiological systems. However, this approach can be applied to any system consisting of individuals in a population. The same disadvantage will still apply as for example intracellular mechanisms in a cell drives a cells behaviour and hence cell population behaviour.

1.4.2 *Petri Nets*

Petri nets were created in the 1960s by Carl Adam Petri for describing chemical processes [57]. This approach is also intensively used in computing science to model and analyse concurrent and distributed systems. A Petri net is a directed graph whose vertices can be divided into two disjointed sets (bipartite graph). It contains two sets of nodes called *places* (represented by circles) and *transitions* (represented by bars) which are connected by directed arcs. Places hold *tokens* (represented by coloured marks) that can be produced or consumed when an input or output transition fires. A transition fires whenever it is enabled by the presence of some tokens in one of the places directly connected to it. A concurrent semantics specifies the evolution in time of the token distribution. This graphical modelling approach is popular with computational systems biologists to describe biochemical reaction systems where the tokens represent molecules [44, 15].

An ordinary Petri net has only one type of token and the net is flat. Coloured Petri nets (CP-nets) can have different types of token. These tokens can have an attached data value called the token colour. The token colours can be investigated and modified by the transitions. CP-nets can be organised into hierarchies of layers giving rise to Hierarchical Petri nets [28]. This approach is similar to modular programming as the construction of the CP-net can be broken into smaller pieces by creating *substitution transitions*. Nets with these transitions therefore have multiple layers of detail. The net can be simplified to give a broad overview of the system (top-level net) and give more detail by substituting transitions of the this net with more detailed nets. Hierarchical Petri nets solve the issues of creating large and intricate CP-nets.

Hierarchical Petri nets have been used to describe multi-scale systems specifically in systems biology. An example of a multi-scale bacterial macrophage interaction Petri net is given by Carvalho et al [20]. They model the bacterium *Mycobacterium tuberculosis* one of the most efficient pathogens responsible for chronic infection. Carvalho et al [20] define biochemical compounds or receptors as places in the net. The relationships between biochemical substances are represented by transitions with arcs modelling biochemical reactions, inhibitions/degradations and signal/catalytic events. Carvalho et al [20] Hierarchical Petri net has three layers. The top level net models the overall view of the system (interactions that occur in the cellular wall and the consequences). This top level net is connected to the middle level net through substitution transitions which connects to the molecular level modelled by the bottom level net.

The advantages of this approach is it provides a graphical framework for qualitative analysis given the static structural topology of Petri nets. Qualitative analysis is carried out by Carvalho et al [20] in the animation of their Petri net in the Snoopy software [34]. This software allows animating the token flow of the Hierarchical Petri net visualising the causality of the model and its behaviour. Bartocci et al [15] also state quantitative analysis can also be carried out given by the time evolution of the token distribution. The disadvantages of this approach is that CP-nets are static structures whereas real life scenarios are dynamic. For example if there is updated information concerning the role of a molecule or dynamics of a reaction from the literature, the structure of the CP-net would need to be completely changed to add new features and dependencies as a result. Another drawback is single components of a system and their connections may not be easily identified in a CP-net model. For example proteins communicate and synchronise on certain reactions which cause a cell to modify its behaviour.

This is important when modelling multi-scale interactions and their emergent properties. There are numerous extensions to CP-nets which try to solve some of the issues mentioned above [74].

Process algebra traces its roots back to the development of Petri nets and the following section focuses on process algebra languages that are designed to model systems and multi-scale systems.

1.4.3 *Process Algebra*

Process algebra (or process calculus) is a compositional approach to formally model concurrent systems. Process algebra gives a high-level description of interactions, communications, and synchronizations between a collection of independent agents or processes [12]. It is used to model networks from computer and biological systems including biochemical networks. Its application provides many analysis techniques for the network's behaviour and properties. The first modern examples include the Calculus of Communicating Systems (CCS) [49] and Communicating Sequential Processes (CSP) [37]. Numerous process algebras have been designed since and each applied to different systems or questions studied. Process algebra shares similarities to IBM as both approaches concentrate on individual behaviour of an agent or cell.

In the context of this thesis, focus will now be directed on Biochemical-Performance Evaluation Process Algebra (Bio-PEPA) [21] which is a modification of Performance Evaluation Process Algebra (PEPA) [35]. Ciocchetta and Hillston [21] states that Bio-PEPA makes it possible to represent explicitly features of biochemical models, specifically the stoichiometry and the role of the species in a given reaction. Also kinetic laws can be implemented to represent each action's functional rate. Bio-PEPA is equipped with an operational semantics and a stochastic labelled transition system based on discrete levels of concentration. Therefore, Bio-PEPA models are implemented in the reagent-centric view. This is a low level view of the biological system, describing each reagent and their interactions. This is in contrast to the pathway-centric view which represents a high level view of the system describing pathways and the interactions which occur within them. This high level view therefore does not explicitly describe the reagents within the pathways. Scott [69] states the advantage of the reagent-centric view is that Bio-PEPA models will have an explicitly descriptive view of the biological system. This is why in the scope of this thesis Bio-PEPA is chosen to investigate the potential of process algebra to model multi-scale systems. This work shows the important challenges of multi-scale modelling and highlights the multi-scale features that need to be included in PAL.

Bio-PEPA is an extension of PEPA and therefore it carries forward some design features. These features are referred to in the following section before Bio-PEPA is described.

1.4.3.1 *PEPA*

PEPA [35] (an extension of CCS) is a process algebra originally defined for the performance analysis of computer systems. PEPA models are described as interactions of components (P) and these components engage, either singly or multiply, in activities (α). The components correspond to identifiable parts in the system, or roles in the behaviour of the system. Each component can perform a set of activities which capture the actions of the system. Every activity/action in PEPA has an associated duration which is a random variable with an exponential distribution. Each component can perform a set of actions: an action is described by a pair (α,r), where α is the type of the action and r is the parameter of a negative exponential distribution governing its duration, i.e. the rate of the action. PEPA has a set of combinators which are used to build up complex system/network behaviour. The combinators are: prefix, choice, parallel composition (cooperation) and hiding (abstraction). Each of these combinators are described by Hillston [35, 36] and reproduced below for convenience.

Prefix (.) is the basic mechanism by which the behaviours of components are constructed. The component (α,r).P carries out activity (α,r), which has action type α and a duration which is exponentially distributed with parameter r. The component subsequently behaves as component P.

Choice (+): a choice between two possible behaviours is represented as the sum of the possibilities. Thus the choice combinator represents competition between components or activities depending on their rate.

Cooperation ($\underset{L}{\bowtie}$): actions in the cooperation set require the simultaneous involvements of components. The resulting action, a shared action, will have the same type as the contributory actions and a rate reflecting the rate of the action in the slowest participating component.

Hiding (/): it is often convenient to hide some actions, making them private to the components involved [35, 36].

PEPA has a number of analysis techniques which can be used on PEPA models if appropriate. These analysis techniques include: Static, Markovian, Performance, and Discrete and Continuous Simulation. The PEPA plug-in incorporated in the Eclipse tool utilises all these techniques [27].

1.4.3.2 *Bio-PEPA*

Bio-PEPA [21] is a language for the modelling and the analysis of biochemical networks. It has also been applied to describe a variety of systems [46, 16]. Bio-PEPA is based on PEPA and extends it in order to handle some features of biochemical networks, such as stoichiometry (quantity of agents[1] involved in a reaction), the role of the agent in a given reaction, and

1 The Bio-PEPA term for these is species, in-line with the biochemical interpretation as molecules, compounds etc. The term agents is used here to avoid the obvious confusion with biological species classification.

different kinetic laws (different rates of reactions). The syntax for terms in Bio-PEPA is already presented in Ciocchetta and Hillston [21] and reproduced here for convenience:

$$\begin{aligned} S :: &= (\alpha, \kappa)\ \mathrm{op}\ S \mid S + S \mid C \\ P :: &= P \underset{L}{\bowtie} P \mid S(x) \\ \text{where op} &= \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot \end{aligned}$$

The two main components of a Bio-PEPA model are agent components S which describe the behaviour of individual entities, and the model component P, which describes the interactions between the various agents. The prefix in PEPA is replaced by a new one, (α, κ) op S, containing information about the role of the agent in the reaction associated with α. The prefix is (α, κ) where α is the action type and κ is the stoichiometry coefficient of the agent(s) in that reaction. The stoichiometric coefficient captures how many molecules of an agent are required for a reaction. The rate of the reaction α is given by a *kinetic law*: an arithmetic expression which may include numeric rate parameters, some simple geometric functions (e.g. sin, exp), and which may depend functionally on the numbers of agents in the model.

The prefix combinator op represents the role of S in the action or the impact the action has on that agent. The prefix combinators are: $\downarrow$ indicating a reactant, $\uparrow$ a product, $\oplus$ an activator, $\ominus$ an inhibitor and $\odot$ a generic modifier. A reactant will be consumed in the action and a product will be produced as a result of the action. Activators, inhibitors and generic modifiers play a role in an action without being produced or consumed and have a meaning defined in the biochemical context. A choice between two possible behaviours is represented as the sum of the possibilities. Thus the choice combinator + represents competition between agents or actions depending on their rate. Actions in the cooperation set $\underset{L}{\bowtie}$ require the simultaneous involvements of agents. The resulting action, a shared action, will have the same type as the contributory actions and a rate reflecting the rate of the action in the slowest participant. In the model component $S(x)$ the parameter x represents the initial amount of the agent.

The underlying semantics of Bio-PEPA is as a Continuous Time Markov Chain. The model can be broken down into states and transitions between these states.

An example Bio-PEPA model of a genetic network with a negative feedback through dimers is given in [21] and reproduced in Figure 1.5. The transcription of M (messengerRNA molecule) is inhibited by the protein P2 (molecule in dimeric form). This has an affect on the translation of the protein P (molecule in monomer form) as M is an activator of this translation. The model is made up of agents representing molecules of the specific genetic network shown in Figure 1.6. All the actions in the model are given specific kinetic laws based on the set of rate parameters. The *transcription* action is given a kinetic law to build in inhibition to the model as the one transcription action which produces mRNA (M) is inhibited by the dimer protein (P2). The other actions are given kinetic laws that use the function of mass action indicated by (fMA) which takes into account the agent populations that take part in the action and

Parameters of model

$$
\begin{aligned}
\mathrm{kM} &= 356;\\
\mathrm{k2} &= 0.043;\\
\mathrm{k3} &= 0.0039;\\
\mathrm{k4} &= 0.0007;\\
\mathrm{k5} &= 0.025;\\
\mathrm{k5i} &= 0.5;\\
\nu &= 2.19;
\end{aligned}
$$

Actions and their associated kinetic rates

$$
\begin{aligned}
\textit{kineticLawOf transcription} &: \tfrac{\nu}{(\mathrm{kM}+\mathrm{P2})};\\
\textit{kineticLawOf translation} &: \textit{fMA}(\mathrm{k2});\\
\textit{kineticLawOf degradation\_M} &: \textit{fMA}(\mathrm{k3});\\
\textit{kineticLawOf degradation\_P} &: \textit{fMA}(\mathrm{k4});\\
\textit{kineticLawOf dimerization} &: \textit{fMA}(\mathrm{k5});\\
\textit{kineticLawOf dimerization\_inv} &: \textit{fMA}(\mathrm{k5i});
\end{aligned}
$$

Agent definitions

$$
\begin{aligned}
\mathrm{M} &= (\textit{transcription}, 1)\uparrow + (\textit{translation}, 1)\oplus + (\textit{degradation\_M}, 1)\downarrow;\\
\mathrm{P} &= (\textit{translation}, 1)\uparrow + (\textit{degradation\_P}, 1)\downarrow + (\textit{dimerization}, 2)\downarrow + (\textit{dimirization\_inv}, 2)\uparrow;\\
\mathrm{P2} &= (\textit{transcription}, 1)\ominus + (\textit{dimerization}, 1)\uparrow + (\textit{dimerization\_inv}, 1)\downarrow;
\end{aligned}
$$

Model Component

$$
\mathrm{M}[0] \underset{\mathit{translation}}{\bowtie} \mathrm{P}[0] \underset{\mathit{dimerization},\ \mathit{dimerization\_inv}}{\bowtie} \mathrm{P2}[0]
$$

(fMA = formula of mass action)

Figure 1.5: Example Bio-PEPA model of a genetic network [21].

changes the kinetic law rate accordingly. As this function is inbuilt in the Bio-PEPA plug-in the only parameter needed is the definition of the rate of the action. There are three different agents defined in this model. The first agent M represents mRNA which has three different actions associated with it. M is defined as a product of the *transcription* action, as the activator for the *translation* reaction and as a reactant of the *degradation_M* action. All these reactions require one M agent except for the *transcription* action where one M agent is produced. The second agent P represents the protein which has four different actions associated with it. P is defined as the product of the *translation* and *dimerization_inv* actions and is a reactant of the actions *degradation_P* and *dimerization*. *Translation* and *degradation* reactions require one P agent, and *dimerization* and its inverse action require two P agents. The third agent P2 represents the dimer protein which has three different actions associated with it. P2 is defined as the inhibitor of *transcription*, as the product of *dimerization* and as the reactant of *dimerization_inv*. All of these reactions require one P2 agent. The model component contains these three agents with starting populations all at zero and the actions on which they synchronise e.g. agent M synchronises with an agent P on the action *translation* [69].

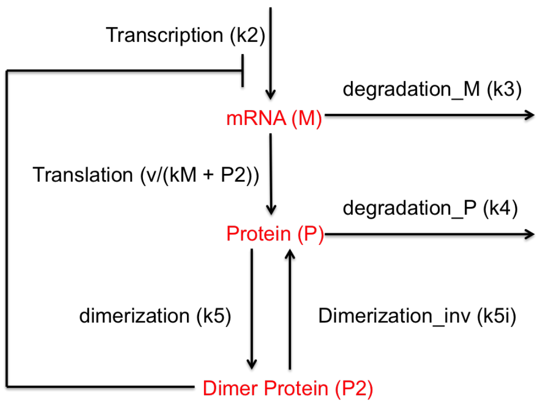


Figure 1.6: Genetic Network Model [21]. Agents are in red and actions are shown with their associated rates.

The Bio-PEPA language is supported by a suite of software tools which automatically process Bio-PEPA models and generate internal representation suitable for different types of analysis. These analysis techniques include: Static, Markovian, Invariant, Simulation traces, Simulation Distributions, Parameter Estimation and Discrete and Continuous Simulation [11]. These analysis techniques are described by Avgeri et al [11] and are reproduced here as some are utilised (highlighted in bold) throughout this thesis.

Static analysis informs the modeller of any syntactic and simple semantic errors in the model before any simulations have been run. The "outline view" in the plug-in shows the actions and agents that are present in the model and also shows which of these actions are sources or sinks. This view informs the user about the model in an abstract way. The "problems view" highlights any errors in the syntax of the model and also shows warnings suggesting that particular actions need to be assessed. This static analysis provides confidence to the modeller in their understanding of the system and the syntactic correctness and consistency of their model

Markovian analysis: The Bio-PEPA plug-in can translate the model into a Probabilistic Symbolic Model Checker (PRISM) model, therefore the model's Continuous Time Markov Chain (CTMC) can be analysed in the PRISM tool. CTMC derivation produces all the possible states and therefore the evolutions of a model and subsequently can be used for functional verification of the model.

Invariant analysis highlights the state and activity invariants in the model. Invariants are expressions whose value does not change during program execution. This is useful when modelling biochemical networks as some activities during reactions should remain constant.

Time Series Analysis: This includes two techniques, one via the mapping of a Bio-PEPA model to a set of ODEs and the other via Stochastic Simulation Algorithms (SSA). The modeller is required to select the components they want to investigate and set the main parameters (start time, stop time and number of data points). An important SSA-related parameter to set is the number of replications wanted during the simulation. The time series of each component will be shown as a different line on the 2D graph which this analysis produces. The graph shows how the number of the selected components evolves over time. The results can be exported to a comma separated value (csv) file for further analysis.

Stochastic and ODEs simulations can be carried out on the model in Figure 1.5 to see the relative numbers of the populations over a period of time. An example of the automatic mapping of the Bio-PEPA model to a set of ODEs is presented in [21] and the ODEs are reproduced here for convenience. Agents M, P and P2 are mapped to Equations 1.1, 1.2 and 1.3 respectively. It is noted that it is not yet possible to obtain the automatically derived ODEs from the plug-in.

$$\frac{dM}{dt} = \frac{v}{k_M + P_2} - k_3 * M \tag{1.1}$$

$$\frac{dP}{dt} = k_2 * M - k_4 * P - 2 * k_5 * P^2 + 2 * k_5 i * P_2 \tag{1.2}$$

$$\frac{dP_2}{dt} = k_5 * P^2 - k_5 i * P_2 \tag{1.3}$$

Simulation Distributions obtain the percentage of a user-defined number of stochastic simulations for which some property is true at or before a given time t. The Bio-PEPA plug-in plots the Cumulative Distribution Function (cdf) and the Probability Distribution Function (pdf) of any agents in the model, with respect to the target value. This simulation distributions technique provides an opportunity to obtain many more statistics about the model. The results can be exported to a csv file for further analysis.

Parameter Estimation can be used on Bio-PEPA models to find unknown values for certain parameters by comparing the time series data with experimental data. Parameter estimation is easily available for Bio-PEPA models via either Systems Biology Software Interface (SBSI) [6] or the Evolving Process Algebra (EPA) framework [45] (recently adapted to accept Bio-PEPA input).

The Bio-PEPA plug-in utilises these techniques and allows the user to export appropriate file types to analyse the model in other applications [2], most notably Systems Biology Markup Language (SBML) [5]. The SBML modelling language is incorporated into over 230 modelling and analysis tools and is regarded as a generic standard modelling language for the computational biology field [4].

1.4.3.3 *Multi-scale Process Algebra*

There are few process algebra languages that are specifically designed for multi-scale systems. In this thesis focus is on the integration of spatial scales and the interactions between these scales. PAL assumes events in different spatial scales take place at the same time scale. Two languages are compared with PAL. They are Parametric Stochastic Process Algebra with Hooks (psPAH) [24] and PEPA nets [30]. These multi-scale languages were chosen for comparison as they also focus on the integration of spatial scales and the interactions between these scales taking place at the same time. Both languages are described below with example models given.

1.4.3.4 *psPAH*

Parametric Stochastic Process Algebra with Hooks (psPAH) [24, 25] is designed specifically for modelling tissue growth and pattern formation. It follows the middle-out [53] strategy where modelling can begin at any scale and then relate to higher or lower scales. Put generally, psPAH allows the modelling of interrupt like events happening across scales.

The main differences between psPAH and a traditional process algebra with multi-way synchronisation, such as PEPA and Bio-PEPA, are the substitution of the simple α actions with more complex *composed actions* $L'[L'']$ and the addition of the *vertical cooperation operator*. The composed action $L'[L'']$ is interpreted while on this scale perform the actions in set L' altogether, while broadcasting actions in set L'' to the other scales. It implies that actions in L' affect the local scale while actions in L'' affect other scales. If an action belongs to set L' it is called a *layer action* and if an action belongs to set L'' it is defined as a *hook action*. Hook actions synchronise with layer actions in other scales, an example of a composed action is shown in Figure 1.7.

The vertical cooperation operator synchronises layer actions on one side of the operator with hook actions on the other side via actions present in L''. No hook with hook or layer

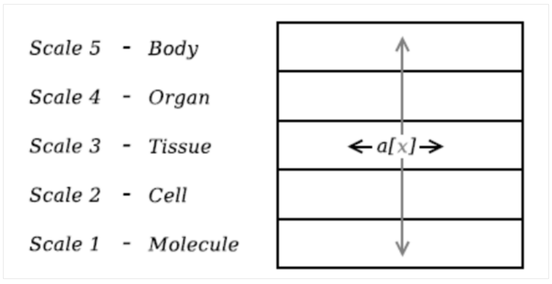


Figure 1.7: Composed action $a[x]$ presents layer action a that operates within the current scale and hook action x that operates between scales [24].

Agent Definitions

$A_0 \;=\; a.A_1 \;+\; f.A_1$ $\qquad B_0 \;=\; c.B_1 \;+\; e.B_1$

$A_1 \;=\; a[p].A_2 \;+\; b.A_0 \;+\; f[p].A_2 \;+\; e.A_0$ $\qquad B_1 \;=\; c[p].B_2 \;+\; d.B_0 \;+\; e[p].B_2 \;+\; f.B_0$

$A_2 \;=\; b[q].A_1 \;+\; e[q].A_1$ $\qquad B_2 \;=\; d[q].B_1 \;+\; f[q].B_1$

$P_0 \;=\; p.P_1$ $\qquad P_1 \;=\; q.P_0 \;+\; p[x].P_2 \;+\; \{p,q\}.P_1$ $\qquad P_2 \;=\; q[y].P_1$

$Cell_M \;=\; move.Cell_M \;+\; x.Cell_A$ $\qquad Cell_A \;=\; absorb.Cell_A \;+\; y.CellM$

Model Initial State

$((A_0 \underset{e,f}{\bowtie} B_0) \underset{p,q}{\bowtie} P_0) \underset{x,y}{\bowtie} Cell_M$

Figure 1.8: Example psPAH model. Please note the layout is used to indicate sections and is not part of the syntax.

with layer action synchronisations are allowed with this operator. This operator expresses cooperation between processes at different scales.

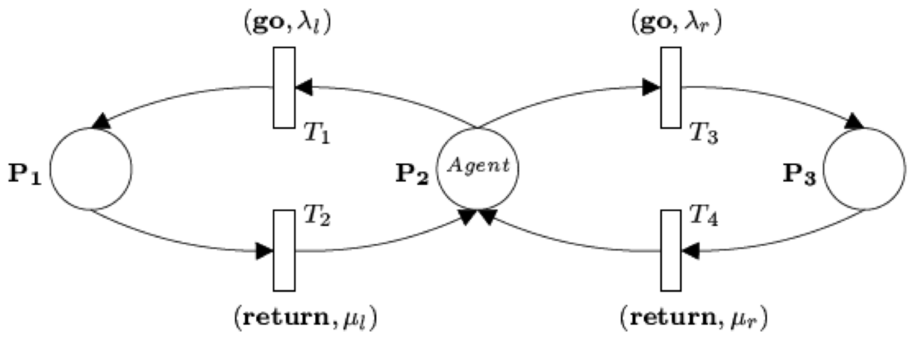
It is noted that the psPAH language cannot generate new agents and cannot remove agents therefore this may hinder modelling features such as reproduction and death. A solution to this problem could be the creation of ghost agents that would wait around to be used in the generation of specific agents. Agents could also become ghost agents when they require to be removed from the model. The initialisation of ghost agents will make the model's initial state huge. An example of the use of these ghost agents is given in Section 4.4.1 of Chapter 4.

An example of a psPAH model of a biological system is given in [24] and is reproduced in Figure 1.8. It represents the interactions between biochemical and cellular scales. There is a middle scale between these two scales that is used to count how many biochemical agents A and B have their concentration above a specific threshold. The biochemical scale interacts with the middle scale through the hook actions p and q when the agents are at a specific concentration and perform specific layer actions. The choice of $\{p, q\}.P1$ defines that P1 agent should stay in the P1 concentration if two hook actions originate from the biochemical scale. The middle scale interacts with the cellular scale through the hook actions x and y when the P agent is at a specific value and performs specific layer actions. These multi-scale interactions cause the cell agent to change from either moving or absorbing.

1.4.3.5 *PEPA Nets*

PEPA Nets [30] is a formalism which uses PEPA as the inscription language for labelled stochastic Petri nets (Petri nets are discussed in Section 1.4.2). The net is used to provide a structure for combining related PEPA systems. It is an example of a two level modelling language. It is designed for modelling peer-to-peer file stores, for example, the PEPA terms are used to model the program code which moves between network hosts (the places in the net). Petri nets [8] provide a graphical presentation of a model which has an easily accessible interpretation and they also have the advantage of being supported by an unambiguous formal interpretation. Coloured Petri nets are a high-level form of classical Petri nets. The plain tokens of a classical Petri net are replaced by arbitrary terms which are distinguishable. In PEPA nets the colours used as the tokens of the net are PEPA components. A PEPA net with only one place and no transitions is a PEPA model.

There are two types of change of state in a PEPA net: *firings* of the net and *transitions* of PEPA components. These two types of change of state can be used to model changes which take place on different scales. Transitions of PEPA components will typically be used to model small-scale (or local) changes of state as components undertake activities. Firings of the net will typically be used to model macro-step (or global) changes of state such as a mobile software agent moving from one network host to another.



$(\mathbf{go}, \lambda_l)$
$(\mathbf{go}, \lambda_r)$
T_1
T_3
$\mathbf{P_1}$
$\mathbf{P_2}$
Agent
$\mathbf{P_3}$
T_2
T_4
$(\mathbf{return}, \mu_l)$
$(\mathbf{return}, \mu_r)$

Figure 1.9: Mobile agent system [30].

A *firing* of a PEPA net causes the transfer of one token from one place to another. The token which is moved is a PEPA component, which causes a change in the remainder of the evaluation both in the source (where existing co-operations with other components now can no longer take place) and in the target (where previously disabled co-operations are now enabled by the arrival of an incoming component which can participate in these interactions). Firings have global effect because they involve components at more than one place in the net.

A *transition* in a PEPA net takes place whenever a transition of a PEPA component can occur (either individually, or in co-operation with another component). Transitions can only take place between components which are resident in the same place in the net. The PEPA net language does not allow components at different places in the net to co-operate on a shared activity. Transitions in a PEPA net have local effect because they involve only components at one place in the net.

A PEPA net is made up of PEPA *contexts*, one at each place in the net. Contexts contain a *cell*. A cell is a storage area dedicated to storing a PEPA component. The components which fill cells can circulate as the tokens of the net. Components which are not in a designated cell are static and cannot move. PEPA nets use the notation $P[-]$ to denote a context which could be filled by the PEPA component P or one with the same alphabet. If P has derivatives P' and P'' only and no other component has the same alphabet as P then there are four possible values for such a context: $P[-]$, $P[P]$, $P[P']$ and $P[P'']$.

$P[-]$ enables no transitions.

$P[P]$ enables the same transitions as P.

$P[P']$ enables the same transitions as P'.

$P[P'']$ enables the same transitions as P''.

A PEPA net model must contain an initial marking of the net, similar to the marking of a Petri net which records the number of tokens which are resident in each place in the net. Instead of tokens PEPA nets record where the dynamical PEPA components are in the net.

PEPA context definitions

$$P_1[Agent] = Agent[A] \underset{\{interrogate\}}{\bowtie} Probe$$

$$P_2[Agent] = Agent[A] \underset{\{dump\}}{\bowtie} Master1$$

$$P_{2'}[Agent] = Agent[A] \underset{\{dump\}}{\bowtie} Master2$$

$$P_3[Agent] = Agent[A] \underset{\{interrogate\}}{\bowtie} Probe$$

Initial marking of the net

$(P_1[\_], P_2[Agent], P_3[\_])$

PEPA definitions: Static components

$$Master1 = (dump, \tau).Master2$$

$$Master2 = (analyse, r_a).Master1$$

$$Probe = (monitor, r_m).Probe + (interrogate, \tau).Probe$$

PEPA definitions: Dynamic component

$$Agent = (\mathbf{go}, \lambda).Agent1$$

$$Agent1 = (interrogate, r_i).Agent2$$

$$Agent2 = (\mathbf{return}, \mu).Agent3$$

$$Agent3 = (dump, r_d).Agent$$

Arcs of the Net

$$P_2 - (\mathbf{go}, \lambda) > - P_3$$

$$P_3 - (\mathbf{return}, \mu) > - P_2$$

$$P_2 - (\mathbf{go}, \lambda) > - P_1$$

$$P_1 - (\mathbf{return}, \mu) > - P_2$$

Figure 1.10: Example PEPA nets model [30].

A simple example of a PEPA nets model of a mobile agent system is given by Gilmore et al [30] and is reproduced in Figure 1.10. The PEPA net represents a system made up of a travelling agent that visits three sites shown in Figure 1.9. The Agent interacts with the static components Master and Probe at the sites causing it to change state. On visiting a site which has a Probe component it synchronises on the activity *interrogate*. On visiting a site which has a Master1 component it synchronises on the activity *dump*. After completion of the dump activity the Master1 component changes state to Master2. In this state a Master2 component can carry out the slow activity of *analyse*. The system allows the Agent to continue travelling and performing activities while the analyse activity is being carried out by the Master2 component. Figure 1.9 shows the initial marking of the net where the mobile agent is resident at the central site P_2. The activities that cause a firing of the net (go and return) are highlighted in bold.

1.4.4 *Other Modelling Formalisms*

1.4.4.1 *Temporal logic*

Temporal logics are very concise modelling languages. They meticulously specify the occurrence of specific temporal behaviours. This approach uses rules and symbolism for representing and reasoning about, propositions qualified in terms of time. High level statements are used to describe a system to satisfy the properties of the system [15].

A popular temporal logic in computing science is Linear Temporal Logic (LTL) [58] introduced by Pnueli to reason about the order of events occurring during the execution of a program. The basic proposition (p) in LTL indicates a Boolean value that may express the relationship between a state variable of the system and a value for a particular time instant. For example, LTL syntax can specify that the concentration of the molecule A is greater than a certain threshold t ($A > t$) or that a specific event e should occur. More complex logical formulae can be obtained by combining propositions using logical operators such as *or* and *not*. The other classical logical operators such as *and* and *implication* can be derived by combining the previous two operators. LTL also includes two temporal operators: *next* (a formula should hold until the next step) and *until* (a formula *one* requires to hold until a formula *two* becomes true). The combination and the nesting of the basic propositions with the logical and temporal operators allow the specification of several different types of temporal behaviours. LTL operates on a single path of the model execution, and a temporal property can be formulated only for one possible trajectory of the system [15].

A new logic called Hybrid Linear Logic (HyLL) [23] developed by Despeyroux provides a unified language to describe biological systems in systems biology, specifically to express properties of their dynamic behaviour and to prove these properties. de Maria et al [23] states in contrast to LTL, HyLL describes both biological and temporal properties and prove these properties from the system. In HyLL syntax propositions are interpreted as resources which may be composed into a state using linear connectives and linear implications which denotes transitions between states. de Maria et al [23] constructed a simple HyLL model consisting of six rules of the biological example of the tumour suppressor protein P53 and its relationship with the protein Mdm2 in the DNA-damage repair mechanism. This system is important in the study of cancer therapies. de Maria et al [23] analysed the model through model checking techniques to formally verify the biological system.

Advantages of the temporal logic approach is that it allows the specification, reasoning and verification of systems and utilises numerous model checking tools. The drawback of this approach is it represents systems in abstract models created specifying what the system is, instead of describing it mechanistically like other modelling approaches mentioned above. It is important to describe these mechanisms to understand how emergent behaviour occurs.

1.4.4.2 *P Systems*

The P systems [56] formalism was created by Paun and is inspired by biology. P systems are based on the structure of biological cells and the way chemicals interact and cross cell membranes. Variations of the P system formalism initiated a new area within computer science called 'membrane computing'. P systems were developed for their use in computational modelling rather than biological modelling although it has been applied to some specific case studies in cellular systems [55].

Romero-Campero et al [68] presented how P systems can be used as a multi-scale framework in systems biology. Romero-Campero et al [68] states specifically how it explicitly specifies the molecular, cellular and colony levels in cellular systems in a relevant and understandable manner. Advantages of this approach is that because it is biologically inspired it can be successfully applied to cellular systems in systems biology. There is little research on its general application to other multi-scale systems outwith this area. Its specificity with cells and membranes may be a disadvantage making it less generally applicable to areas such as ecology and immunology. Bartocci et al [15] states in their review, P system research is more suitable for the theory of computation than for modelling in systems biology. Hence, this may be the case for other multi-scale systems.

A preliminary theoretical language called Membrane Calculus [63] was developed by Qi et al [63] and is based on P systems and Petri Nets (discussed in Section 1.4.2) to formalize transaction processing in web services and grid services. Qi et al [63] states combining these two approaches means that the system can be analysed graphically as well as being an algebraic model and extends Coloured Petri nets (CP-nets) by introducing the dynamic and reflective structure inspired by P systems. The semantics of Membrane Calculus is divided into two parts: *object rules* and *membrane rules*. Every object rule can be presented by a transition in the term of CP-nets and the semantics of this rule can be represented by the changing of the marking in CP-nets. The operational semantics of one object rule is delimited by a transition similar to CP-nets. Qi et al [63] show that Membrane Calculus is very suitable to describe grid transactions through one example. It would be interesting to see if this approach of a combination of modelling approaches would be generally applicable to multi-scale systems, as yet this has not been seen in the literature.

1.4.4.3 *BioSPI*

BioSPI [65] is a computer application developed by Regev et al [65] for simulating the behaviour of biochemical systems specified in pi-calculus. Pi-calculus [50] was developed by Milner as a formal language for concurrent computational processes. Pi-calculus provides a framework for the representation, simulation, analysis and verification of mobile communication systems. A typical system in the pi-calculus consists of multiple concurrent processes. Pairs of processes interact with each other by sending and receiving messages in a synchronized way. This communication is done on complementary input and output channels.

Regev et al [65] shows the suitability of pi-calculus for modelling biological systems such as signal transduction networks, metabolic pathways and transcriptional regulation. They treat molecules as computational processes. The molecules complementary structural and chemical determinants correspond to communication channels. Chemical interaction and subsequent modification coincide with communication and channel transmission.

This approach has the same advantages like other process algebras of being able to formally represent complex networks, simulate and monitor their behaviour and formally verify their properties and compare networks across organisms. Regev et al [65] states further development is required for this approach as some biochemical events require elaborate encoding in pi-calculus. For example, pi-calculus only supports pair synchronisation (between two processes that present complementary channels) and many biological reactions involve numerous molecules interacting together. Regev et al [65] states that this issue can be overcome with specifying the system in a different level of detail (abstraction). Another disadvantage of this approach is it has only been applied to systems in molecular biology. Further research needs to be completed to investigate whether this approach is applicable to a variety of multi-scale systems.

1.5 BACKGROUND CASE STUDY: THE PACIFIC OYSTER

The case study of the marine invertebrate the Pacific oyster (*Crassostrea gigas*) is used throughout this thesis. The motivation of choosing this case study comes from the problem of ocean acidification and the consequences it has on marine invertebrates. These consequences affect each life stage of a marine invertebrate in different severity. Concentrations of carbon dioxide

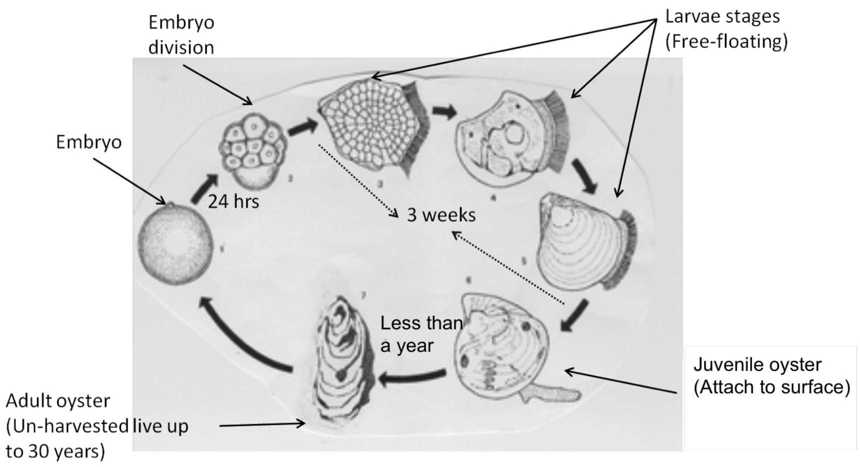

Embryo division
Larvae stages (Free-floating)
Embryo
24 hrs
3 weeks
Less than a year
Juvenile oyster (Attach to surface)
Adult oyster (Un-harvested live up to 30 years)

Figure 1.11: Life cycle model of the Pacific oyster adapted from [1].

(CO_2) in the atmosphere are rising, which is widely accepted will primarily cause an increase in mean global temperatures. A second problem linked with the rise in particle CO_2 is associated with its dissolution in sea water and the subsequent acidification of the world oceans. There are model predictions of the difference in pH of sea water by 2100 ranging between -0.4 and -0.5 units [26]. This decrease in sea water pH will have two consequences for marine organisms. Firstly it has been predicted that a reduction in sea water pH will produce a reduction in the calcification rate of shelled marine organisms [14, 72]. Secondly changes in the pH of seawater will potentially cause a disruption to the internal acids/base balance in calcifying and non-calcifying invertebrates. The maintenance of internal acids/base balance is essential for maintaining protein conformation and subsequently enzyme function and metabolism. These consequences will have an effect not only in the physiology of marine organisms and their persistence in their populations but will also affect the stability of ecosystems in which they inhabit and therefore affect the goods and services those ecosystems provide [19, 29]. Hence this is a multi-scale system that requires a multi-scale model in order to investigate these multi-scale effects.

The Pacific oyster is potentially the largest harvested and collected shellfish in European waters. In 2006, global Pacific oyster aquaculture production reached 4.6 million tonnes (t). European production was around 126 000 t [51]. The Pacific oyster is an isomorph, an organism that does not change shape during growth, which means its surface area is proportional to its volume. These bivalves are ectotherms and osmoconformers: their body temperature and internal salinity is the same as the surrounding environmental conditions [32]. Hence any change in environmental conditions can significantly affect the oyster.

A life cycle model of the Pacific oyster is given in Figure 1.11. A life cycle model is a collection of facts of periods of an organisms' life usually depicted in a graphical composition. The whole life history of an organism is usually represented through a series of developmental

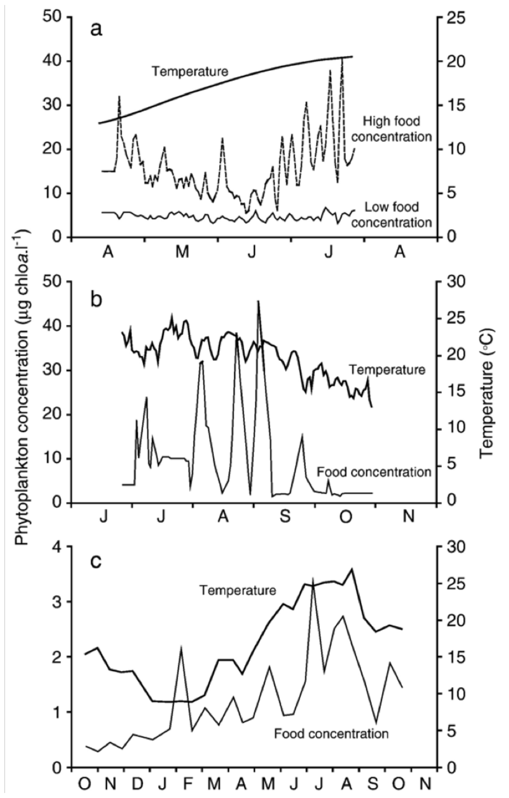


Figure 1.12: Temporal variations of phytoplankton concentration and temperature, in experiments A (top panel); B (middle panel) and C (lower panel) reproduced from [59].

life stages which an organism goes through. Figure 1.11 shows the development of the Pacific oyster from embryo through larval stages to juvenile and adult. The time an oyster spends in each life stage varies dependent on temperature and diet (food density). A Pacific oysters' diet consists mainly of phytoplankton which is made up of minute plants and other photosynthetic organisms.

1.5.1 *Juvenile and Adult life stage experiments*

Chapter 2 refers to wet laboratory experiments on juvenile and adult Pacific oysters from Pouvreau et al [59]. The purpose, methodology and results of these experiments are reproduced here for convenience. Pouvreau et al [59] carried out three (A, B and C) long-term growth experiments (>5 months) located in France on Pacific oysters reared under different conditions of food (phytoplankton densities) and environment (temperature). Pouvreau et al [59] states the resulting data sets from these experiments where used to validate their Pacific oyster DEB model results and demonstrate the DEB models' ability to capture the dynamics of the energy budget in the Pacific oyster in various environments. The DEB model is described in Chapter 2.

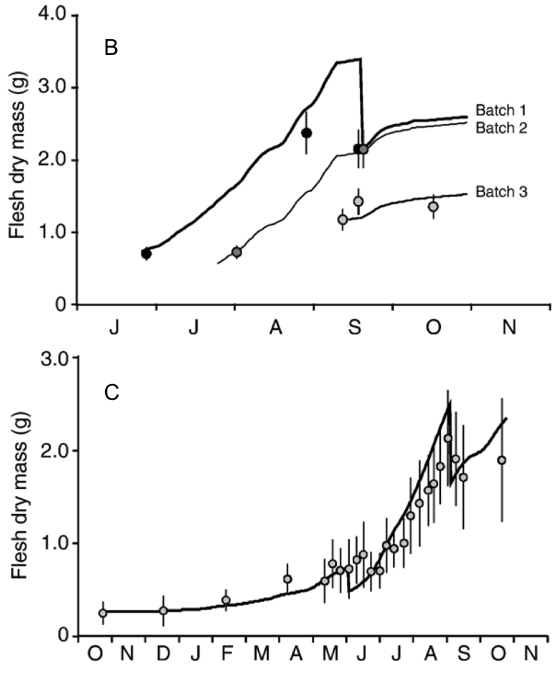


Figure 1.13: Comparison of observations ± SD (dots) and DEB model simulation (line) of Dry Flesh Weight in the Pacific oyster in Experiment B for each stock and Experiment C (Thau lagoon). Note that the sharp drops that can be observed on simulation lines indicate spawning event predicted by the DEB model [59].

The variations in phytoplankton concentration and temperature in all three experiments is reproduced in Figure 1.12. In experiment A, the oysters were placed in experimental facilities and reared at two contrasting food density levels. Experiment A of [59] is omitted in Chapter 2 and therefore only experiments B and C are described in detail here. The reason for this omission is explained in Chapter 2. Experiment B oysters were placed in an oyster pond and reared at a fluctuating food density. The oysters were reared in a natural environment (Thau lagoon) over a complete annual cycle in experiment C.

Pouvreau et al [59] states experiment B lasted from June to October 2002. Oysters were cultivated in a 600 m^3 pond located on Oleron island. They were fed with pure phytoplankton. Biometry (flesh and shell mass) was conducted at the start and end of the experiment. Oxygen, temperature and in vivo chlorophyll-a were monitored daily. Three batches of oysters from different origins were placed in the pond. Batch 1 was composed of about 30 month old oysters from Brittany transferred to the pond in June 2002. Batch 2 consisted of smaller oysters from the same origin, but transported to the pond one month later, presumably after spawning. Batch 3 was composed of oysters from the Marennes-Oleron Bay and transferred to the pond after spawning.

Pouvreau et al [59] states experiment C consisted of an annual growth survey conducted in the Thau lagoon. This study was carried out between September 2000 and October 2001.

Oysters were installed on ropes in late September 2000 at one site located in the north western part of the lagoon. 27 groups of 3 oysters were glued on 3 metre long ropes with cement, and density on the ropes was adjusted to 34 individuals per metre of rope; water depth was 4 metres at the study site. The potential food consisted of natural phytoplankton and growth was monitored monthly over a year. Temperature, salinity and chlorophyll-a values were collected once a week during the growth period and every month outside this period.

The methods used to assess the growth of oysters over all experiments were similar. Under each condition, oysters (> 12) were randomly collected twice a month in experiment C, and at the start and end of experiment B. They were cleaned and weighed after draining. Individual total mass (grams) was recorded. Then, the oysters were opened, and their flesh was removed and drained prior to weighing. The total dry mass of soft tissues was determined after freeze-drying and termed Dry Flesh Weight (DFW) in grams. Figure 1.13 reproduces comparison of the DFW observation data from experiment B and C to Pouvreau et al [59] DEB model. The error bars show the standard deviation in the observed data.

1.5.2 *Larval life stage experiments*

Chapter 3 refers to wet laboratory experiments on the larval life stage of Pacific oysters from Rico-Villa et al [67]. The purpose, methodology and results of these experiments are reproduced here for convenience. Rico-Villa et al [67] carried out two growth experiments (1 and 2) on Pacific oyster larvae reared under different levels of phytoplankton density and temperature. The data sets collected from these experiments where used to validate their Larval DEB model. Rico-Villa et al [67] states the Larval DEB model showed good growth simulations and provided an extensive description of the energetic needs of the Pacific oyster during its larval stage. The Larval DEB model is described in Chapter 3. Rico-Villa et al [67] states all larval experiments (1 and 2) were run starting with two-day-old larvae, and feeding supply was expressed in cell biovolume ($\mu m^3 \mu l^{-1}$).

Rico-Villa et al [67] states in experiment 1 larvae were reared in a flow-through system to maintain a constant flow of algal cells and stable temperature conditions, and allow continuous data recording. Larvae were reared at a density of 30 larvae ml^{-1}, at five different temperatures: 17, 22, 25, 27 and 32 $^\circ$C. Food ration was adjusted as the larvae grew, allowing 1400 $\mu m^3 \mu l^{-1}$ of phytoplankton to always be available. Each set of experimental conditions consisted of a test tank with larvae and food supply, and a control tank with only a constant flow of phytoplankton (no larvae), both at a defined temperature. Larvae, initially reared at 25 $^\circ$C, were acclimated over one day at each temperature before the experiment began.

Rico-Villa et al [67] states in experiment 2 larvae were grown under rearing conditions similar to those described above, except that temperature was maintained at 25 $^\circ$C and it was the food supply that was varied. Larvae were thus continuously fed at several phytoplankton

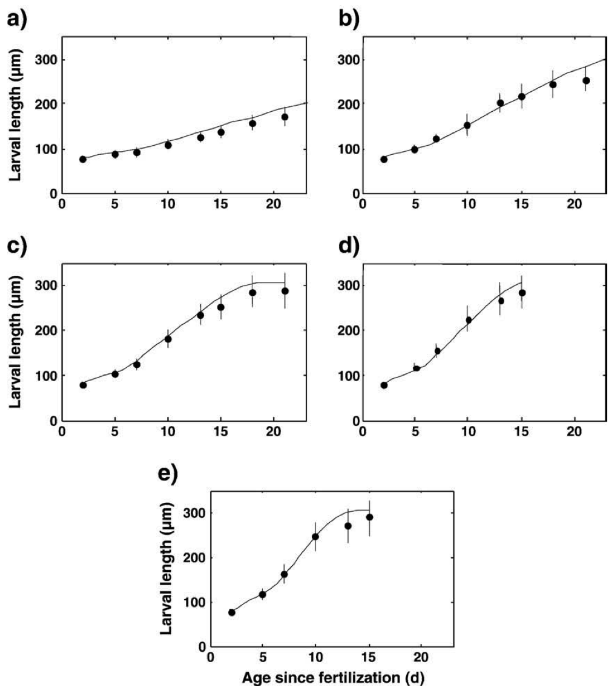


Figure 1.14: Comparison of observations ± SD (dots) and DEB model simulation (line) of growth for Pacific oyster Larvae in experiment 1 under different temperatures: 17 (a), 22 (b), 25 (c), 27 (d) and 32 °C (e). The other environmental conditions were optimal: food density of 1400 $\mu m^3 \mu l^{-1}$ [67].

densities (expressed in cell biovolume) providing food availabilities of 70, 280, 450, 960, 1000, 1900, 2100 and 3300 $\mu m^3 \mu l^{-1}$. The lowest value, 70 $\mu m^3 \mu l^{-1}$, considered as minimal amount of particles, was that found in tank seawater after 1 μm filtration when no phytoplankton was added. Food densities from 200 to 500 $\mu m^3 \mu l^{-1}$ were considered as low diets; 700 to 1000 $\mu m^3 \mu l^{-1}$ as restricted diets, and 2000 to 3300 $\mu m^3 \mu l^{-1}$ as non-restricted diets.

Shell length was measured during the larval experiments using an image analysis technique. Shell length data was assessed during the period where larvae fed exclusively on phytoplankton, exhibited a linear shell length increase and had not achieved competent size of 280 μm . Rico-Villa et al [67] states in their larvae rearing experiment the assumption that when larvae achieved competent size earlier they were removed from the population in which they were measuring mean size. They hypothesised that this is the reason for the plateau in larval growth at the end of some of the observation experiments. Figure 1.14 and Figure 1.15 reproduces the comparison of the shell length observation data from experiment 1 and 2 to Rico-Villa et al [67] DEB model. Mean observed values are presented with their standard deviation.

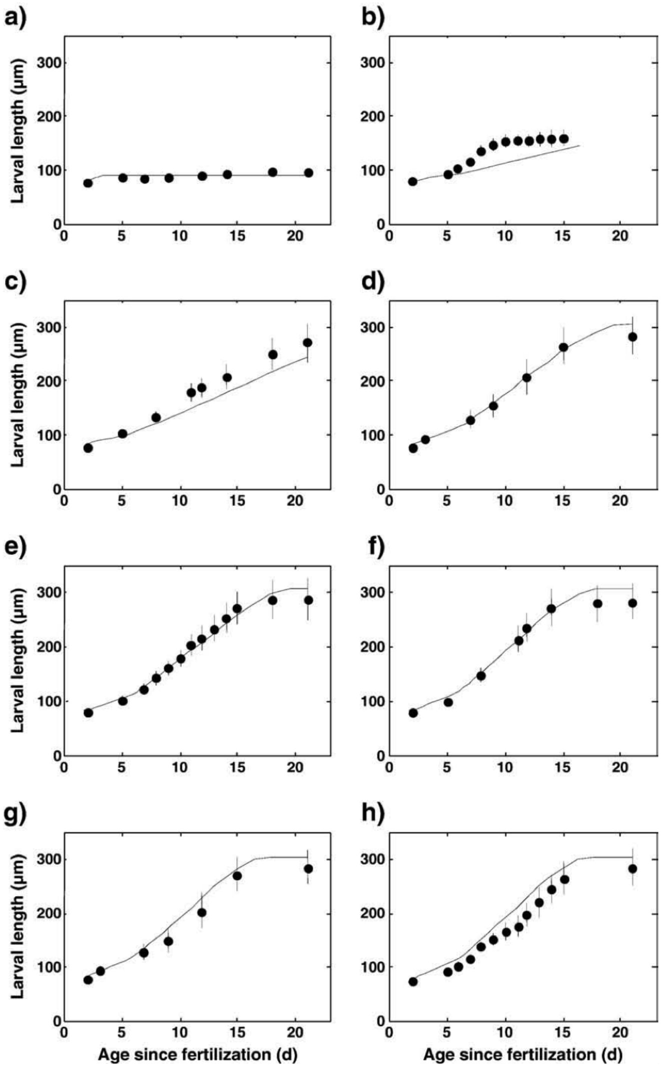


Figure 1.15: Comparison of observations ± SD (dots) and DEB model simulation (line) of growth for Pacific oyster Larvae in experiment 2 under different food density conditions: 70 (a), 280 (b), 450 (c), 960 (d), 1000 (e), 1900 (f), 2100 (g) and 3300 $\mu m^3 \mu l^{-1}$ (h). The other environmental conditions were optimal: temperature of 25 °C [67].

The aim of this thesis is to define a generic multi-scale process algebra language for multi-scale integration modelling. There are essential stages that need to be completed before defining such a language. Firstly, it is important to gain insight into whether a process algebra language such as Bio-PEPA has the potential to model an organism's physiology within a specific life stage scale. A generic translation approach to translate DEB models to Bio-PEPA models will be defined in order to describe an organism's physiology in an abstracted but descriptive view. This work will allow an appreciation on the further analysis available to the model which the Bio-PEPA plug-in can provide. This work will also show the challenges of modelling the physiological scale in Bio-PEPA and comparison analysis with original DEB model results. Secondly, investigations into the multi-scale challenges of integration of the physiological life stages of an organism in Bio-PEPA need to be carried out. Thirdly, the multi-scale challenges and features attained from these Bio-PEPA models will allow the conception of Process Algebra with Layers (PAL). The syntax and semantics of PAL will be defined. The comparison of PAL to other multi-scale process algebra languages will highlight its ability to model specific multi-scale features in an elegant fashion showing that PAL encapsulates the multi-scale features and middle-out strategy. Fourthly, PAL will be applied to the multi-scale system that is the motivation of this thesis. Fifthly, application of PAL to an unrelated multi-scale system will be carried out to test PAL's generic ability and emphasise its usefulness to aid in the analysis of multi-scale systems.

This thesis is separated into the following chapters. **Chapter 2** (published in the proceedings of the *Sixth International Workshop on the Practical Application of Stochastic Modelling (PASM)*, 2013) presents the generic translation approach to translate DEB models to Bio-PEPA models. This is achieved by the translation of a specific DEB model of a Pacific oyster Juvenile-Adult physiological life stage case study. Analysis of the Bio-PEPA model is undertaken to first compare to the original DEB model results and secondly to gain extra insight of the model by utilising analysis techniques in the Bio-PEPA plug-in.

Chapter 3 presents an integrated life stage Bio-PEPA model. This is completed by firstly using the generic translation approach to translate the Pacific oyster Larval physiological life stage DEB model to a Bio-PEPA model. Secondly, this model is integrated with the Pacific oyster Juvenile-Adult physiological life stage Bio-PEPA model. This integrated life stage Bio-PEPA model is analysed to show its predictive capabilities. Multi-scale challenges and features are gathered from this model to discuss the conceptualisation of a multi-scale process algebra language.

Chapter 4 introduces Process Algebra with Layers a language for multi-scale integration modelling. The unique features of PAL are presented, in particular the layers of the language: Population and Organism. The mirrored actions that allow the integration of these layers are shown. The formal syntax and semantics of PAL is presented. A simple PAL model example is given to show the configuration of a PAL model and transitions. Comparison of PAL with process algebra languages psPAH and PEPA nets is presented. Finally in this chapter an implemented PAL parser is described in order that the novel language can be applied to multi-scale systems and the models can be analysed.

Chapter 5 shows the application of PAL to a Pacific oyster life stage case study investigating the impacts of ocean acidification. A PAL model is created to include the oyster's physiology, life stage and population scale. Analysis focuses on the Larva life stage scale and the population scale.

Chapter 6 shows the application of PAL to a cell cycle and DNA damage case study investigating the effects of cancer treatment. A PAL model is created to include the intracellular, cellular and population scale. Analysis of this model focuses on the length of a cell cycle and population growth. The results are compared to wet laboratory data to show that PAL can aid in analysis of multi-scale systems.

In **Chapter 7** conclusions and future work related to this thesis are presented.

2 CONVERTING DYNAMIC ENERGY BUDGET MODELS TO BIO-PEPA, ILLUSTRATED BY A PACIFIC OYSTER CASE STUDY

The work presented in this chapter has been published in the proceedings of the *Sixth International Workshop on the Practical Application of Stochastic Modelling (PASM)*, credited to E. Scott, A. Hoyle and C. Shankland [71]. In this chapter, we illustrate the potential of process algebra to illuminate physiology in a component-based high-level way. An existing Dynamic Energy Budget (DEB) model of the Pacific oyster in Pouvreau et al [59] is translated to a Bio-PEPA model. Moreover, as DEB theory presents models for different organisms in a similar way, generic principles for translating DEB models to Bio-PEPA can be found. My Bio-PEPA model has been validated through testing in a number of experiments with different environmental conditions and initial physical values for the oyster. The results are equivalent to those of the original DEB model, showing the translation to be faithful in this sense. My translated Bio-PEPA DEB model can thereafter be utilised and analysed in a variety of different modelling language tools. Some novel analysis is carried out using the Bio-PEPA plug-in [11]. This new translation approach, therefore, broadens the audience for the implementation and analysis of DEB models. In addition, this demonstrates the utility of Bio-PEPA outside the realm of biochemical networks for which it was developed.

In the context of this thesis, it is important to gain insight into whether a process algebra language such as Bio-PEPA has the potential to model an organism's physiology within a specific life stage scale. DEB models offer a method to describe an organism's physiology in an abstracted way but still captures enough descriptive detail to accurately represent the system. The organism's physiological detail should not be lost in an integrated multi-scale model. It is essential that the detail at each scale is preserved and therefore will reflect the multi-scale system accurately. It is further essential to gain an appreciation on the further analysis available to the model which the Bio-PEPA plug-in can provide.

2.1 TRANSLATING THE PACIFIC OYSTER DEB MODEL TO BIO-PEPA

DEB theory has been utilised to describe a variety of marine invertebrates including the bivalve Pacific oyster [59, 66, 67] studied here. The Pacific oyster and its life cycle are described in Section 1.5 of Chapter 1. The wet laboratory experiments carried out by Pouvreau et al [59] and referred to throughout are described in Section 1.5.1 of Chapter 1.

In Pouvreau et al [59] DEB model there are three state variables: Energy Reserve (E) describes the dynamics of the energy reserve, Structural Volume (V) specifies the growth of the structural body volume and Energy Reproduction Buffer (E_R) describes the storage and use of the energy allocated to development and reproduction. Each state variable is described by an ordinary differential equation, reproduced here for convenience [59]. In DEB modelling $\dot{P}$ represents an energy process.

$$\frac{dE}{dt} = \dot{P}_A - \dot{P}_C \tag{2.1}$$

$$\frac{dV}{dt} = \frac{\dot{P}_G}{[E_G]} = \frac{\kappa.\dot{P}_C - \dot{P}_M}{[E_G]} \tag{2.2}$$

$$\frac{dE_R}{dt} = (1-\kappa).\dot{P}_C - \dot{P}_J \tag{2.3}$$

Equation (2.1) describes the increase of E by the assimilation process which produces energy and the decrease by utilisation of this energy by many processes. Equation (2.2) specifies V is increased by utilised energy which is specifically allocated by the parameter κ. V is decreased by somatic maintenance which stands for a collection of processes necessary to maintain life and also by the volume-specific cost for growth which includes all types of overheads, for example, biosynthesis. E_R (2.3) is increased by an allocated amount of utilised energy and is decreased by maturity maintenance processes.

The DEB model parameters are reproduced in Table 2.1 for convenience with the Bio-PEPA model parameters. In translating the model to Bio-PEPA the relationship between state variables and ODEs, and agents and actions, must be considered. Also of importance are units of measurement, and how the outputs of the model should be formulated.

2.1.1 *Conversion of the state variables to agents*

My Bio-PEPA model is given in Figure 2.1. The first step of the translation is to represent the state variables (E, V and E_R) by agents in the Bio-PEPA model. See **Agent definitions** of Figure 2.1. The equations of the state variables become actions of these agents. Some actions indicate increase or decrease of an agent corresponding to the positive and negative terms of the ODE. Some terms include state variables that influence the state variable. This corresponds to actions that indicate that an agent influences a kinetic rate of another agent's action even though the agent does not increase or decrease when the action occurs. Reserve (E) agent is assigned four actions: *a1*, *a3*, *a4* and *a5*. E is increased by *a3* and decreased by *a4*. The associated rate of action *a3* is defined as the assimilation rate and the rate of action *a4* is defined as the utilisation rate. See **Actions and their associated kinetic rates** of Figure 2.1. Both these rates are as defined in the DEB equations. E is a generic modifier in the actions *a1*

		DEB		Bio-PEPA	
Symbol	Definition	Value	Dimension	Value	Dimension
$[E_G]$	Volume-Specific costs for structure	1900	Jcm^{-3}	1.9	Jmm^{-3}
$[E_M]$	Maximum energy storage density	2295	Jcm^{-3}	2.295	Jmm^{-3}
κ	Fraction of utilised energy spent on growth and maintenance	0.45	-	0.45	-
κ_R	Fraction of reproduction energy fixed in eggs	0.7	-	0.7	-
V_P	Structural body volume at puberty	0.4	cm^3	400	mm^3
$\{\dot{P}_{Xm}\}$	Maximum surface area-specific ingestion rate	560	$Jcm^{-2}d^{-1}$	5.6	$Jmm^{-2}d^{-1}$
$\{\dot{P}_{Am}\}$	Maximum surface area-specific assimilation rate	420	$Jcm^{-2}d^{-1}$	4.2	$Jmm^{-2}d^{-1}$
ae	Assimilation efficiency	0.75	-	0.75	-
$[\dot{P}_M]$	Volume-specific maintenance rate	24	$Jcm^{-3}d^{-1}$	0.024	$Jmm^{-3}d^{-1}$
μ_E	Energy content of reserves	17.5	Jmg^{-1}	17500	Jg^{-1}
ρ	Volume-specific dry flesh weight	0.2	gcm^{-3}	0.2	gcm^{-3}
GSI	Gonadosomatic index triggering spawning	35	%	35	%
T_S	Temperature threshold triggering spawning	20	$^\circ C$	20	$^\circ C$
T_1	Reference temperature	293	K	293	K
T_A	Arrhenius temperature	5800	K	5800	K
T_{AH}	Rate of decrease at upper boundary	30000	K	30000	K
T_{AL}	Rate of decrease at lower boundary	75000	K	75000	K
T_H	Upper boundary of tolerance range	305	K	305	K
T_L	Lower boundary of tolerance range	281	K	281	K

Table 2.1: Model parameters used in this study. The DEB parameters are as given by Pouvreau et al [59].

Parameters of model

$$\begin{array}{rcl}
\mathtt{Actual\_temperature} & = & \text{value dependent on experiment ;} \\
\mathtt{Temperature\_correction} & = & \exp((T_A/T_1) - (T_A/(273 + \mathtt{Actual\_temperature}))) \\
 & & *((1 + \exp((T_{AL}/(273 + \mathtt{Actual\_temperature})) - (T_{AL}/T_L)) \\
 & & + \exp((T_{AH}/T_H) - (T_{AH}/(273 + \mathtt{Actual\_temperature}))))^{-1}); \\
\{\dot{P}_{Xm}\} & = & 5.6 * \mathtt{Temperature\_correction}; \\
\{\dot{P}_{Am}\} & = & ae * \{\dot{P}_{Xm}\}; \\
[\dot{P}_M] & = & 0.024 * \mathtt{Temperature\_correction}; \\
\dot{P}_M & = & [\dot{P}_M] * V; \\
[E] & = & E/V; \\
\dot{P}_C & = & ([E]/([E_G] + (\kappa * [E]))) * ((\frac{[E_G]*\{\dot{P}_{Am}\}*V^{2/3}}{[E_M]}) + ([\dot{P}_M] * V)); \\
\mathtt{Food\_density\_chloa} & = & \text{value dependent on experiment ;} \\
X_\kappa & = & \text{value dependent on experiment ;} \\
\mathtt{Functional\_response} & = & \frac{\mathtt{Food\_density\_chloa}}{(\mathtt{Food\_density\_chloa}+X_\kappa)}; \\
\dot{P}_A & = & \mathtt{Functional\_response} * \{\dot{P}_{Am}\} * V^{2/3}; \\
\mathtt{Maturity} & = & H(V - V_p); \\
\dot{P}_J & = & (((\frac{(1-\kappa)}{\kappa}) * V * [\dot{P}_M]) * (1 - \mathtt{Maturity})) \\
 & & + (((\frac{(1-\kappa)}{\kappa}) * Vp * [\dot{P}_M]) * (\mathtt{Maturity})); \\
\mathtt{Percentage\_E_R} & = & (\frac{E_R\_DFW}{Total\_DFW}) * 100; \\
E_R\mathtt{\_start\_spawn} & = & H(\mathtt{Percentage\_E_R} - GSI); \\
\mathtt{Stop\_spawn} & = & H(1 - \mathtt{Percentage\_E_R}); \\
\mathtt{T\_start\_spawn} & = & H(\mathtt{Actual\_temperature} - T_s);
\end{array}$$

Actions and their associated kinetic rates

$$\begin{array}{rcl}
\mathit{kineticLawOf}\ a1 & : & \frac{(\kappa*\dot{P}_C)}{[E_G]}; \\
\mathit{kineticLawOf}\ a2 & : & \frac{\dot{P}_M}{[E_G]}; \\
\mathit{kineticLawOf}\ a3 & : & \dot{P}_A; \\
\mathit{kineticLawOf}\ a4 & : & \dot{P}_C; \\
\mathit{kineticLawOf}\ a5 & : & ((1 - \kappa) * \dot{P}_C) * \mathtt{Maturity}; \\
\mathit{kineticLawOf}\ a6 & : & \dot{P}_J * \mathtt{Maturity} * (1 - \mathtt{stop\_spawn}); \\
\mathit{kineticLawOf}\ \mathtt{empty} & : & \mathit{fMA}(100 * \mathtt{Maturity}); \\
\mathit{kineticLawOf}\ \mathtt{switch\_on} & : & \mathit{fMA}(10 * E_R\mathtt{\_start\_spawn} * \mathtt{T\_start\_spawn}); \\
\mathit{kineticLawOf}\ \mathtt{switch\_off} & : & \mathit{fMA}(10 * \mathtt{stop\_spawn});
\end{array}$$

Agent definitions

$$\begin{array}{rcl}
V & = & a1 \uparrow + a2 \downarrow + a3 \odot + a4 \odot + a5 \odot + a6 \odot + \mathtt{empty}\odot; \\
E & = & a3 \uparrow + a4 \downarrow + a1 \odot + a5 \odot; \\
E_R & = & a5 \uparrow + a6 \downarrow + \mathtt{empty} \downarrow; \\
\mathtt{Tracker\_off} & = & (\mathtt{switch\_on}, 1) \downarrow + (\mathtt{switch\_off}, 1) \uparrow; \\
\mathtt{Tracker\_on} & = & (\mathtt{switch\_on}, 1) \uparrow + (\mathtt{switch\_off}, 1) \downarrow + (\mathtt{empty}, 1) \oplus;
\end{array}$$

Model Component

$$V[0] \underset{*}{\bowtie} E[0] \underset{*}{\bowtie} E_R[0] \underset{*}{\bowtie} \mathtt{Tracker\_off}[1] \underset{*}{\bowtie} \mathtt{Tracker\_on}[0]$$

(fMA = formula of mass action)

Figure 2.1: Pacific oyster Bio-PEPA model. See Table 2.1 for other parameters.

and *a5* as E influences the kinetic rates of the increasing actions of the Structural Volume (V) and Reproduction Buffer (E_R) although E does not increase or decrease when these actions occur.

Structural Volume (V) is assigned seven actions: *a1* which increases V, and *a2* which decreases it, and actions *a3, a4, a5, a6* and *empty* which leave V unchanged. The associated rate of action *a1* is defined as utilisation rate multiplied by κ divided by the volume-specific cost for growth. Rate of action *a2* is specified as the somatic maintenance rate divided by the volume-specific cost for growth. These rates again use the specific rates as defined in the DEB model. As V is a generic modifier in the other five actions it influences the kinetic rates and does not increase or decrease when these actions occur.

The Reproduction Buffer agent (E_R) has three actions: *a5, a6* and *empty*. Actions *a5* and *a6* are as before where *a5* increases E_R and *a6* decreases it. This agent has the extra action of *empty* to describe the spawning event of the oyster. The associated rate of action *a5* is defined as the utilisation rate multiplied by $1 - \kappa$ multiplied by the parameter *Maturity*. The rate of the action *a6* is specified by maturity maintenance rate multiplied by the maturity parameter. This maturity parameter is created in Bio-PEPA to acknowledge the additional rule of the DEB model that specifies that E_R becomes active when the individual has reached a specific structural volume. The maturity maintenance rate varies with V when the oyster is below the specific structural volume of maturity and becomes constant when V reaches or is above this specific value. The DEB model uses the min function to achieve this whereas in Bio-PEPA the Heaviside step function (H) [9] is utilised. Apart from the maturity parameter the rates used for the actions *a5* and *a6* are as defined in the DEB model.

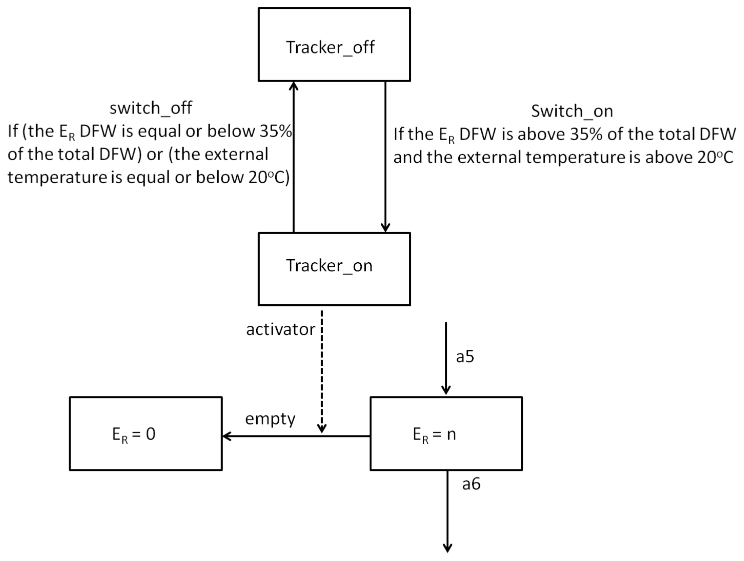

Tracker_off
switch_off
If (the E_R DFW is equal or below 35% of the total DFW) or (the external temperature is equal or below 20°C)
Switch_on
If the E_R DFW is above 35% of the total DFW and the external temperature is above 20°C
Tracker_on
activator
a5
E_R = 0
empty
E_R = n
a6

Figure 2.2: State diagram for the behaviour of the tracker component.

The spawning event of the oyster is described partly in the ODEs of the DEB model, but mainly through the accompanying textual description. The translating process therefore does not only require the translations of the ODEs but also requires the novel interpretation and implementation of timed events with specific conditions. There are two conditions that have to be fulfilled before spawning can take place. The first condition refers to the build up of gonad material and the second condition is dependent on the temperature during the seasons. The first condition is that a certain gonadosomatic index (GSI) has to be reached: this means the ratio between the gonad and total tissue mass is above the GSI. Secondly, the external temperature must be above a specific threshold (T_S). It is not sufficient to only use the Heaviside step function to implement the conditions described above, because the *empty* action would only be active as long as both conditions were true and hence spawning would be partial (E_R would never fully empty). Instead, these conditions and event are implemented by a tracker component in the Bio-PEPA model. The tracker component only switches on when both conditions are met and acts as an activator to the *empty* action of the reproduction buffer. As defined in the DEB model when a spawning event occurs E_R is completely emptied, therefore the empty action decreases E_R at a fast rate, it is triggered multiple times. The tracker component switches off when E_R becomes zero, therefore the empty action cannot take place. Thus it is never possible for E_R to become negative. A state diagram of the tracker component's behaviour is given in Figure 2.2. The kinetic rate of the tracker is given by the built-in mass action function (*fMA*).

2.1.2 *Adding the forcing variables into the model*

Temperature and food density are forcing variables. Temperature affects two physiological rates, maximum surface area-specific ingestion rate and volume-specific maintenance rate. In the DEB model this dependency on the temperature is described by an Arrhenius-type equation [31] and this is utilised in the Bio-PEPA model. The second forcing variable, food density, affects the assimilation rate and is implemented in the same way as the DEB model. Both forcing variables vary over time in the DEB model: experimental data was imputed for both values at each data point in time. Since both temperature and food density are measured variables from experiments, there exists a time series for each (as shown in Figure 1.12 in Chapter 1 and again in Figure 2.3 left). It is desirable to be able to directly input these time series to the Bio-PEPA tool as background data to use in calculations. This is not currently possible. Instead hand crafted functions must be coded to approximate the time series for experimental data (as shown in Figure 2.3 right). These make use of the inbuilt time variable and the Heaviside step function. There is a trade-off between the complexity of these functions and the closeness of the approximation.

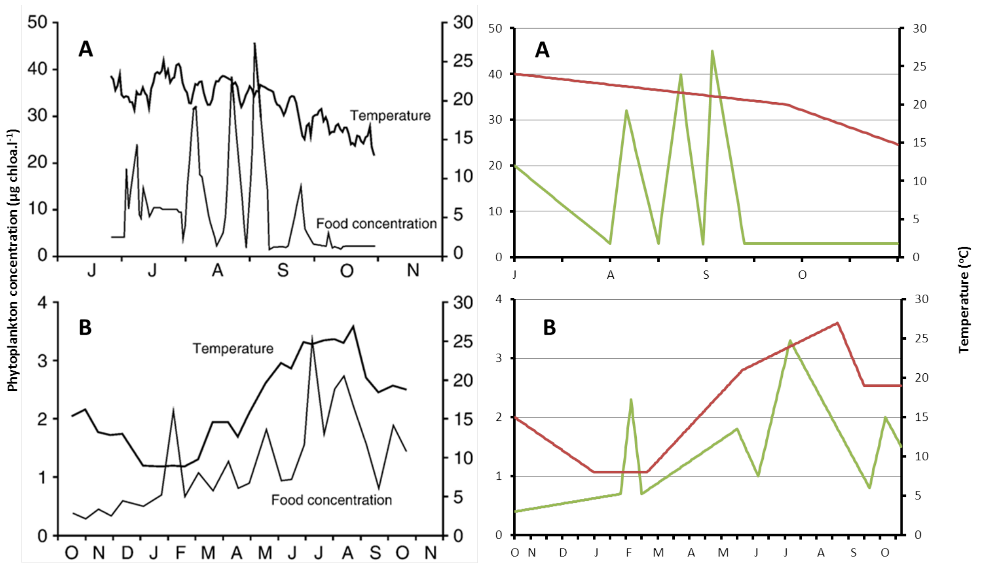


Figure 2.3: Temporal variations of the forcing variables: temperature and phytoplankton in Experiment A and B. DEB model values on left (reproduced from Pouvreau et al [59]) and Bio-PEPA values on right. The scales are phytoplankton concentration on the left, and temperature on the right of each graph.

2.1.3 *Changing the units of specific parameters*

In Bio-PEPA initial values for each agent are required to be an integer. The initial value of V in the DEB model is a decimal number therefore changing of some units in the model must be made. The unit of V is cm^3 and is changed to mm^3 to gain integer values with acceptable precision. Other model parameters that are affected by V had their units appropriately changed. See Table 2.1.

2.1.4 *Addition of dry flesh weight equation for comparison*

The DEB model results are compared with wet laboratory results using a calculated total dry flesh weight value (DFW). The Bio-PEPA simulation results must therefore be calculated into DFW values and also their units to be scaled and changed appropriately for the comparison. Equation (2.4) from Pouvreau et al [59] gives the total DFW. The other values (such as the assimilation and respiration rates) from the Bio-PEPA model can be compared to the DEB model and the wet laboratory results. Analysis of this model is shown in Section 2.3.

$$DFW = \frac{E}{\mu_E} + (\frac{V}{1000}) * \rho + \frac{\kappa_R * E_R}{\mu_E} \tag{2.4}$$

Having learned from the Pacific oyster Bio-PEPA model, it is possible to describe a generic approach that can be used to transform an organism's DEB model that includes the state variables of Structural Volume (V), Reserve (E) and Reproduction Buffer (E_R). A more complex DEB model may require further investigation.

Conversion of the state variables to agents: DEB model state variables usually are V, E and E_R. There may be more than one V and E. These will become agents in the Bio-PEPA model. As noted by Gurriero and Heath [33], the translation from simple ODEs to Bio-PEPA is straightforward. This is partly true for the DEB model here. It is worth noting that DEB models include timed events such as spawning which are not described in the ODEs of the models, therefore interpretation and implementation of the textual model assumptions must be made e.g using a tracker component.

Implementation of the actions of agents from the state variable equation definitions: The equations of the state variables form the kinetic rates of the agent's actions. The part within the equation which increases the state variable becomes the kinetic rate of an action for which that state variable's agent is a product. The part of the equation that decreases the state variable will become the kinetic rate of the action for which that agent is a reactant.

The state variable E_R is involved in a reproduction event, therefore the use of the Heaviside step function and a tracker component may be required to set the specific conditions of the event, e.g. the use of a tracker component for a specific reproduction event of Section 2.1.1.

Adding the forcing variables into the model: Values of the forcing variables are usually wet laboratory values that are entered at each data point in time. As it is not possible to add each data point to each time point in a Bio-PEPA simulation, hand crafted functions should be implemented to create similar behaviour of the forcing variables over time. Statistical techniques such as regression can assist here.

Changing the units of specific parameters: In Bio-PEPA initial values for each agent are required to be integer. Changing of some units and suitable scaling in the values of the state variables must be made. An example is shown in Table 2.1. Other parameters affected by the state variables that have been changed in this way must be changed accordingly.

Addition of equations for comparison and analysis of results: DEB model results are compared to wet laboratory results by an equation to convert the state variable values to an appropriate unit value. This equation can be used on the results of the Bio-PEPA model. In the Pacific oyster model in Section 2.1 the comparison equation is Equation (2.4).

2.3 MODEL ANALYSIS

2.3.1 *Comparison analysis results*

Two time-series analysis techniques, continuous ODEs and discrete stochastic simulation, were used on the Bio-PEPA model. The Bio-PEPA model is validated by comparing its output with the results of the original DEB model. It is further validated by statistical comparison: the original DEB model [59] compared the simulated predicted results with observed wet laboratory data using R^2 statistics; the Bio-PEPA model simulation results were also compared to the observed wet laboratory data using the same technique. The goodness-of-fit between prediction (Y) and Observation (X) was tested according to the R^2 value of the regression Y=X.

Pouvreau et al [59] carried out three wet laboratory experiments described in Section 1.5.1 of Chapter 1. Two of these are shown here: experiment A and experiment B corresponding to B and C respectively of [59]. Experiment A of [59] was omitted because food concentration data had a high number of fluctuations and a complex hand crafted function needed to be created. Also this experiment took place with oysters in tanks whereas the other two experiments were in natural environments. The results from the DEB model are from ODEs simulations using the Systems Thinking for Education and Research (STELLA) tool [38]. The Bio-PEPA model results are stochastic simulations of multiple replications (1000); therefore, simulating the growth and reproduction of a population of oysters. 1000 replications are chosen consistently here across analyses to give a representative average system behaviour. The Bio-PEPA model is also used to generate ODEs simulation results, to give comparability with the original DEB model.

Experiment	DEB V (cm^3)	Bio-PEPA V (mm^3)	E (J)	E_R (J)	X_K ($\mu g\ chl - al^{-1}$)
Experiment A					
Batch 1	2.3	2300	2000	4000	8
Batch 2	2.6	2600	500	0	8
Batch 3	3.1	3100	3500	8500	8
Experiment B	1	1000	500	500	3.5

Table 2.2: Initial values of X_K and the state variables: V, E and E_R.

The two experiments from the original DEB model had different initial values for the state variables and a different value for X_K (half-saturation coefficient). The half-saturation coefficient is changed due to the different diet composition between experiments [59]. Both experiments are carried out over a different time period and under different environmental conditions. This demonstrates the Bio-PEPA model's generic ability to capture the dynamics of the energy budget in the Pacific oyster in various environments. Table 2.2 reproduces [59] the initial values of the state variables and X_κ for each experiment and includes the Bio-PEPA model scaled state variable V. The graphs of the DEB model [59] (Shown in Figure 1.5.1 of Chapter 1) are reproduced here for convenience for comparison to the outputs of the Bio-PEPA model.

2.3.1.1 *Experiment A*

The wet laboratory methodology of this experiment is described in Section 1.5.1 of Chapter 1. This experiment had a time period of 120 days (July to October). The model is tested here as the experiment has a fluctuating environment (see Figure 2.3) because the food concentration varies erratically and oysters from various origins are analysed. The experiment encapsulates three sub-experiments (batch 1, 2 and 3) and each batch has different initial state variable values indicating oysters from different origins. The batch 1 experiment lasted the whole time period, batch 2 had a duration of 90 days commencing from August and batch 3 started ten days from the start of September lasting 50 days. Batch 1 were allowed to continue and complete a spawning event whereas the other two batches were introduced too late for spawning to take place. The forcing variables' values, temperature and phytoplankton concentration, of both the original DEB model and the Bio-PEPA model are given in Figure 2.3. The differences in the values occurs as the values in the Bio-PEPA model are produced from functions which approximate the actual measurements whereas the DEB model uses wet laboratory values. The total dry flesh weight values for all batches in the DEB model and Bio-PEPA model is given in Figure 2.4.

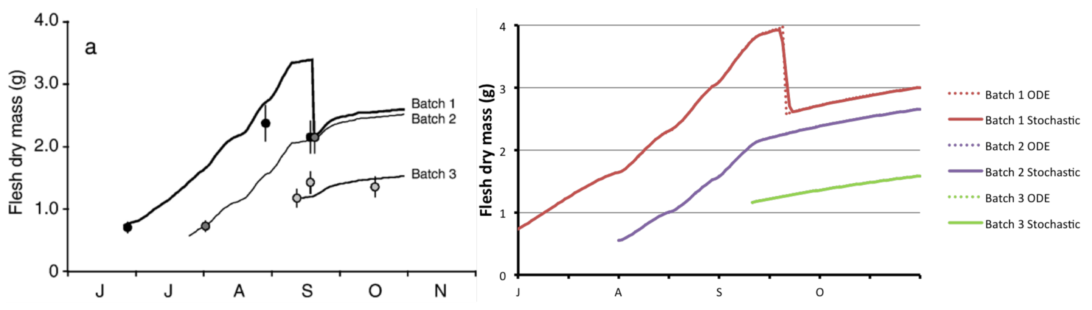


Figure 2.4: Experiment A comparison of total dry flesh weight results of DEB model left [59] and Bio-PEPA model (ODEs and stochastic results) right. DEB model includes comparison of observations ± SD (dots). The error bars show the standard deviation in the observed data. Note that the sharp drops that can be observed on simulation lines indicate spawning events predicted by the models.

The Bio-PEPA model produced comparable results to the original DEB model. It confirms a very good simulation of somatic growth and the replication of a spawning event. The slight differences in batch 1 are derived from the difference in the forcing variable values. Although it cannot be seen clearly on the graphs, the values of reproduction weight released at the spawning event in batch 1 are similar and the time of the spawning is the same.

Pouvreau et al [59] state the observed wet laboratory data of the three sub-experiments of this experiment were all pooled together for the statistical comparison with the simulation results of the original DEB model. This was carried out as the aim of this experiment was to test the model not only on a more fluctuating environment but on several populations of oysters from various origins [59]. There is also a limited amount of data available in the observation results, for example batch 2 only has two observation data points. I also grouped the observed wet laboratory data of this experiment therefore the statistical comparison is faithful to the original DEB model. The original DEB model gave $R^2 = 0.81$ (n=8, $p<0.002$) between observation and simulation. The Bio-PEPA model gave $R^2 = 0.813$ (n=8, p=0.002) between observation and stochastic simulation and gave $R^2 = 0.812$ (n=8, p=0.002) between observation and ODEs simulation. Here, n represents the number of data points and p represents the p-value. This statistical comparison confirms the Bio-PEPA model in this experiment gives comparable results to the original DEB model.

2.3.1.2 *Experiment B*

The wet laboratory methodology of this experiment is described in Section 1.5.1 of Chapter 1. This experiment had a duration of 365 days, that is a complete annual cycle. The experiment has typical natural environmental field conditions. These conditions are presented in Figure 2.3; again differences in the forcing variables values occur as the values in the Bio-PEPA model are produced from functions whereas the DEB model uses wet laboratory values. The total dry flesh weight value for this experiment in the DEB model and Bio-PEPA model is given in Figure 2.5.

The Bio-PEPA model simulated the growth of an oyster over a complete annual cycle and also the two spawning periods. The first spawning event is at the beginning of June and the weight lost is approximately 0.15g in the stochastic simulations and 0.28g in the ODEs simulation. The second spawning event takes place around the end of August and the weight lost is approximately 0.61g in the stochastic simulations and 0.82g in the ODEs simulation. These results are comparable to the original DEB model with 0.2g for the first event and 0.5g for the second.

The goodness-of-fit for the original DEB model was $R^2 = 0.92$ (n=24, $p<0.0001$) against observation and predicted. The Bio-PEPA model gave $R^2 = 0.86$ (n=24, $p<0.0001$) between observation and stochastic simulation. Analysis of the ODEs simulation against observation gave $R^2 = 0.824$ (n=24, $p<0.0001$). The Bio-PEPA result are less comparable to the original

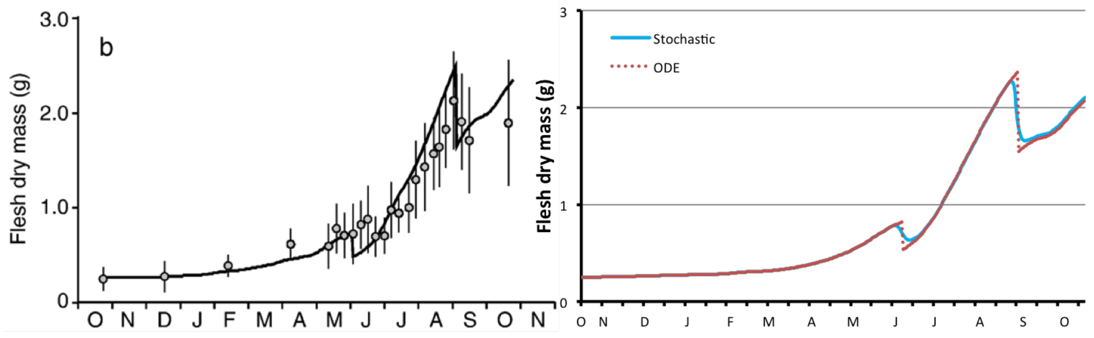


Figure 2.5: Experiment B comparison of total dry flesh weight results of DEB model left [59] and Bio-PEPA model (ODEs and stochastic results) right. DEB model includes comparison of observations ± SD (dots). The error bars show the standard deviation in the observed data. Bio-PEPA graph horizontal axis tick marks indicate 15 days and month letters are at the start of each month. Note that the sharp drops that can be observed on simulation lines indicate spawning events predicted by the models.

DEB model result because of the functions that describe the behaviour of the temperature and food forcing variables. The functions are more simplistic in their behaviour than the original collected data. For example, the temperature in the Bio-PEPA model may be decreasing below 20°C too early in September, artificially preventing some simulations spawning for a second time.

The Bio-PEPA model, similar to the DEB model, outputs not only the total dry flesh weight values but also values of internal parameters of the model such as the assimilation rate and functional response. These results can be used to analyse the models internal functioning. Figure 2.6 shows the assimilation rate plotted against the maintenance costs. This demonstrates the assimilation is just sufficient to meet the maintenance costs when food is limited during the winter period [59]. It is noted that the parameters of scaled energy density and functional response also displayed the same internal relationship values as the original DEB model.

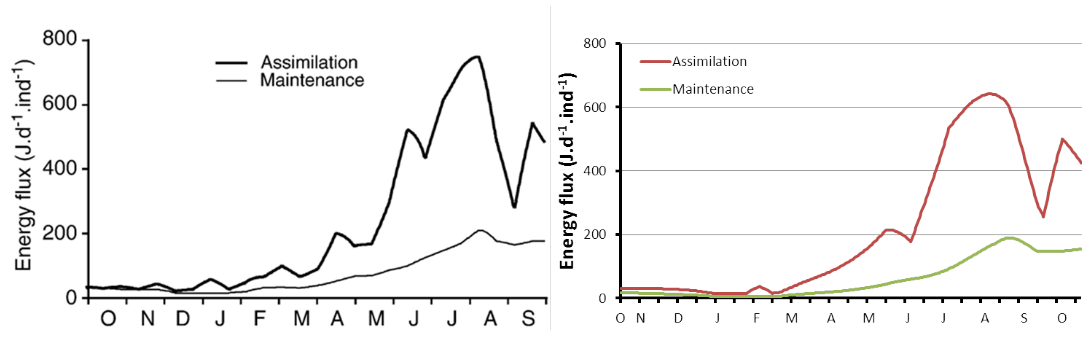


Figure 2.6: Experiment B assimilation of energy against maintenance costs. DEB model results left reproduced [59].

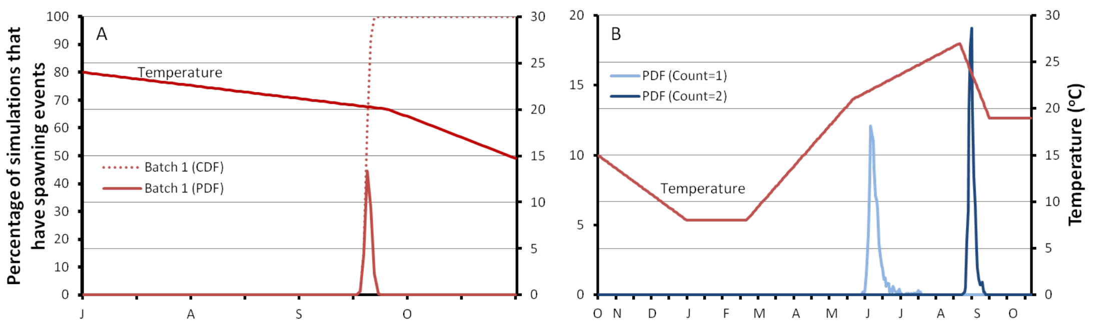


Figure 2.7: Simulation distributions for experiment A (left) and B (right). Temporal variations of the forcing variable temperature for each experiment is also shown. The scales are CDF and PDF percentage values on the left, and temperature on the right hand side of each graph.

2.3.2 *Simulation distributions analysis of the Bio-PEPA model*

Further analysis of the Bio-PEPA model is performed using the analysis technique simulation distributions in the Bio-PEPA plug-in. Results from this analysis are presented for both experiments and are given in Figure 2.7. This technique allows the analysis of the spawning events and when they are most likely to occur. The chosen component in this analysis is an agent which counts the number of times the Tracker_on component becomes equal to 1, i.e. when spawning occurs. The target value is set to 1 for experiment A and set to 1 and then 2 in experiment B. The number of stochastic simulation replications is set to 1000.

2.3.2.1 *Experiment A*

For batch 1 spawning starts to occur at day 77 with 1.3% of the simulations reaching the target value of 1 around mid September. 90.7% of the simulations reach this target between day 78 and 80. By day 82 all simulations reach the target. This certifies the spawning event occurring at a narrow time frame. Batch 2 and 3 simulations never reach the target value of 1 indicating that a spawning event will never occur in either experiment.

2.3.2.2 *Experiment B*

Simulations start to reach the target value of 1 at day 223. All the simulations had reached this target value by day 272. This indicates a large window of time for the first spawning to occur (beginning of June to mid July). The distribution is skewed: 72% of simulations spawn within the first 9 days of June then a long tail.

Simulations start to reach the target value of 2, i.e. a second spawning event, at day 308 (around the end of August). 97.6% of all simulations had reached this value by day 326 at the start of September. 24 simulations did not produce a second spawning event. This may be due to these simulations having late first spawning events and therefore do not have time to build up to the GSI condition before the temperature drops below 20°C. These types of results are not available in the original DEB model ODEs results. It is possible in principle to do stochastic simulations in ODEs models but the model would first need to be transformed into a different equation type to allow such analysis. This may be the reason that such analysis was not carried out in the original DEB model.

2.3.3 *Parameter Estimation*

Pouvreau et al [59] estimate some model parameters, as is common in modelling; for example, finding the values of volume specific cost for structure $[E_G]$ and the maximum energy storage density $[E_M]$ in a starvation experiment [66]. Parameter optimisation can be used on Bio-PEPA models to find unknown values for certain parameters by comparing the time series data from the relevant experiment, removing the need to carry out additional wet laboratory experiments for these certain parameters. Given experimental data, this is easily available for Bio-PEPA models. This analysis has not been carried out for this model due to lack of access to experimental data.

2.4 SUMMARY

In this chapter I described a generic translation approach to easily convert mathematical DEB models to Bio-PEPA models. A concrete example model of the translation process has been constructed and its results have been compared to the original DEB model. I carried out new analysis (Section 2.3.2) on a specific DEB model in the Bio-PEPA plug-in by using simulation distributions and new results have been generated about the system demonstrating the utility of the translation process.

The Pacific oyster Bio-PEPA model also shows that it is generic, producing results for different environmental conditions and for different state variables. The model can therefore be used again for other related bivalve experiments, potentially feeding back to further, more targeted, wet laboratory experiments.

The Bio-PEPA plug-in tool [11] has a range of analysis techniques which can further aid in examination of results. The Bio-PEPA model can be exported and converted into other computational modelling and analysis tools. This allows a wider audience to access the model. This range of further analysis techniques is not available for the DEB model.

A problem identified with the Bio-PEPA plug-in is that functions approximating the environmental data were required: it would be desirable to add these directly from the collected data. This may account for the differences between our results and those of the DEB model as the forcing variables have a significant effect.

My generic translation approach can be used in future work to investigate not only marine invertebrates DEB models but also other organism DEB models [41], therefore, broadening the audience for modelling and analysis. In the next chapter this approach is used to implement a DEB Bio-PEPA model of the Larval life stage of the Pacific oyster.

This work shows that a process algebra language such as Bio-PEPA can successfully model an organism's physiology within a specific life stage scale. The comparison analysis of the model shows that DEB models accurately capture enough detail of the system. In Section 2.3.2 further analysis of this model was undertaken using the simulation distributions analysis available within the Bio-PEPA plug-in. This shows the Bio-PEPA plug-in can aid in the further analysis of this type of system.

3 LARVA AND INTEGRATED LIFE STAGE BIO-PEPA MODELS

In this chapter we illustrate how Bio-PEPA can be used to model the Larval life stage of the Pacific oyster by translating a Dynamic Energy Budget (DEB) model using my novel translation approach described in Chapter 2. Ocean acidification has its greatest effect on the Larval life stage of the Pacific oyster [14], therefore it is important to translate and include this stage in the model. The Larva Bio-PEPA model is validated by comparing its simulation results of two experiments with the original DEB model results.

A novel integrated life stage Bio-PEPA model is implemented by linking the Larva model and Juvenile-Adult model from Chapter 2. Analysis of the integrated model shows its predictive capacity and usefulness. Important multi-scale challenges and features are identified from this model. The highlighted features and issues observed need to be resolved in the PAL language to model multi-scale systems appropriately.

3.1 LARVA MODEL

An existing mathematical DEB model of the life stage of the Pacific oyster Larva in Rico-Villa et al [67] is translated to a computational Bio-PEPA model. The Pacific oyster and its life cycle are described in Section 1.5 of Chapter 1. The wet laboratory experiments carried out by Rico-Villa et al [67] and referred to throughout are described in Section 1.5.2 of Chapter 1. Chapter 3 shows that DEB models which have different parameter values for physiology and driving forces can be translated to Bio-PEPA models using the generic translation approach detailed in Chapter 2.

3.1.1 *Translating the Larva Pacific oyster DEB model to Bio-PEPA*

My Bio-PEPA model is given in Figure A.1 of Appendix A. In Rico-Villa et al [67] DEB model there are three state variables: Reserve (E) describes the dynamics of the energy reserve, Biovolume (E_V) specifies the flow of energy from the Reserve for the process of growth of the structural body volume and Development (E_R) describes the storage and use of the energy allocated to development and the maintenance of this development. The state variables are described by a set of ODEs, reproduced here for convenience [67]. In DEB modelling $\dot{P}$ represents an energy process.

$$\frac{dE}{dt} = \dot{P}_A - \dot{P}_C \tag{3.1}$$

$$\frac{dE_V}{dt} = \dot{P}_G = \kappa.\dot{P}_C - \dot{P}_M \tag{3.2}$$

$$\frac{dE_R}{dt} = (1-\kappa).\dot{P}_C - \dot{P}_J \tag{3.3}$$

Equation (3.1) describes the increase of E by the assimilation process which produces energy and the decrease by utilisation of this energy by many processes. Equation (3.2) specifies E_V is increased by utilised energy which is specifically allocated by the parameter κ and is decreased by somatic maintenance which stands for a collection of processes necessary to maintain life. E_R (3.3) is increased by an allocated amount of utilised energy and is decreased by maturity maintenance processes. The Bio-PEPA model parameters are shown in Table A.1 of Appendix A.

The initial agents' (Reserve, Biovolume and Development) values are scaled up by 10^6 as agent initial values need to be integers. The agents' values are scaled back by 10^{-6} in the parameters: E, E_V and E_R. These parameters are used throughout the model for parameter and rate equations. The agents' units values are therefore changed in one place in the model. This makes writing the model more efficient. For example if the units need to be modified later, only one alteration is needed. This will reduce manual writing errors. See the parameters of the model in Figure A.1. The rates of the actions were multiplied by 10^6 to be at the same scale as the agents' values.

To compare the model output to available data, Larval length L is used, which was obtained by converting the state variable E_V into μm using the equation (3.4) from Rico-Villa et al [67].

$$L = (E_V/[E_G])^{1/3}/\delta_M \tag{3.4}$$

3.1.2 *Analysis*

Both continuous ODEs and discrete stochastic simulation time-series analysis techniques of the Bio-PEPA plug-in were used in the analysis of the Larva model. The Bio-PEPA model is validated by comparing its output with the results of the original DEB model. It is further validated by statistical comparison: the original DEB model [59] compared the simulated predicted results with observed wet laboratory data using R^2 statistics; the Bio-PEPA model simulation results were also compared to the observed wet laboratory data using the same technique. The goodness-of-fit between prediction (Y) and Observation (X) was tested according to the R^2 value of the regression Y=X.

Day	Observation (μm)	Stochastic (μm)
2	79.06	77.298
5	86.01	95.237
7	91.95	107.259
10	109.93	125.277
13	123.93	143.177
15	136.16	155.098
18	157.49	172.946
21	171.40	190.770

Table 3.1: Experiment 1(a) comparison data used in R^2 regression for Bio-PEPA model stochastic result. Observation data reproduced from Rico-Villa et al [67].

Rico-Villa et al [67] carried out two experiments (Experiment 1 and 2 described in Section 1.5.2 of Chapter 1) focusing on changes of the forcing variables (food density and temperature) of the model and how these will affect Larval growth. Similarly to the Juvenile-Adult Bio-PEPA model the Larva Bio-PEPA model results are stochastic simulations of multiple replications (1000) as before. The Bio-PEPA model is also used to generate ODEs simulation results, to give comparability with the original DEB model, which are faster to generate than stochastic results. To match the data both experiments had the same initial value (2.5×10^{-4} Joules) for the state variables Biovolume and Reserve. E_R has the initial value of zero. X_K (half-saturation coefficient) is the same in both experiments with the value of $600\mu m^3\ \mu l^{-1}$. The simulations start at day 2 as this is when the first collected wet lab Larval length data was taken. The simulations time period is around 25 days when the Larvae reach the size of 300 μm. Rico-Villa et al [67] state that at this size the process of metamorphosis occurs and this is when a Larva becomes a Juvenile.

Rico-Villa et al [67] stated in their larvae rearing experiment the assumption that when larvae achieved competent size earlier they were removed from the population in which they were measuring mean size. They hypothesised that this is the reason for the plateau in larval growth at the end of some of the observation experiments. To reflect this the original DEB model simulations ended when the Larva reached the size of 300 μm. In my Larva Bio-PEPA model the feeding mechanism was allowed to continue even when the Larva reached 300 μm. It is unrealistic to try and match the Bio-PEPA simulation results to some of the last data points of the observed Experiments that have this plateau because the observed data points are not showing the realistic growth of Larvae. To take this into account the last data point of the observed data was removed when comparing the simulation results of the Bio-PEPA model in specific experiments that were effected by this plateau. These include Experiment 1 (c to e) and Experiment 2 (d to h). For Experiment 2 (f) the last two data points were removed as they both occurred when the plateau was in effect. Table 3.1 shows the comparison data for

Experiment 1(a) shown in Figure 3.1 as an example of the data used to find the R^2 value in the statistical analysis.

3.1.2.1 *Experiment 1*

The wet laboratory methodology of this experiment is described in Section 1.5.2 of Chapter 1. Experiment 1 focused on Larvae growth under different temperatures with food density at a constant optimal value of $1400 \mu m^3 \mu l^{-1}$. The experiment shows the strong influence that temperature has on growth and that increasing the temperature results in a shorter time for a Larva to reach the size of 300 μm (a shorter larval rearing period). The Larval length values for this experiment in the original DEB model and my Bio-PEPA model is given in Figure 3.1. Both the stochastic and ODEs simulation results of the Bio-PEPA model are similar to each other indicating consistency in the model.

The Bio-PEPA model produced comparable results to the original DEB model. The statistical comparison R^2 results of the Bio-PEPA model comparison with the DEB model is given in Table 3.2. This statistical comparison confirms the Bio-PEPA model in this experiment gives comparable results to the original DEB model.

Experiments	Temperature (°C)	Food Density ($\mu m^3 \mu l^{-1}$)	R^2: Original DEB	R^2: Bio-PEPA Stochastic	R^2: Bio-PEPA ODEs
Experiment 1					
(a)	17	1400	0.987	0.983	0.983
(b)	22	1400	0.972	0.983	0.983
(c)	25	1400	0.996	0.985	0.985
(d)	27	1400	0.984	0.989	0.989
(e)	32	1400	0.977	0.970	0.970
Experiment 2					
(a)	25	70	0.954	0.547	0.552
(b)	25	280	0.611	0.881	0.881
(c)	25	450	0.987	0.991	0.991
(d)	25	960	0.983	0.968	0.968
(e)	25	1000	0.974	0.972	0.972
(f)	25	1900	0.961	0.973	0.973
(g)	25	2100	0.987	0.955	0.955
(h)	25	3300	0.990	0.962	0.962

Table 3.2: Comparison of Bio-PEPA model and the original DEB model [67] goodness-of-fit to the observed data.

3.1.2.2 *Experiment 2*

The wet laboratory methodology of this experiment is described in Section 1.5.2 of Chapter 1. Experiment 2 focused on Larvae growth under different food densities with temperature at a constant optimal value of 25°C. The experiment shows that increasing food density enhances

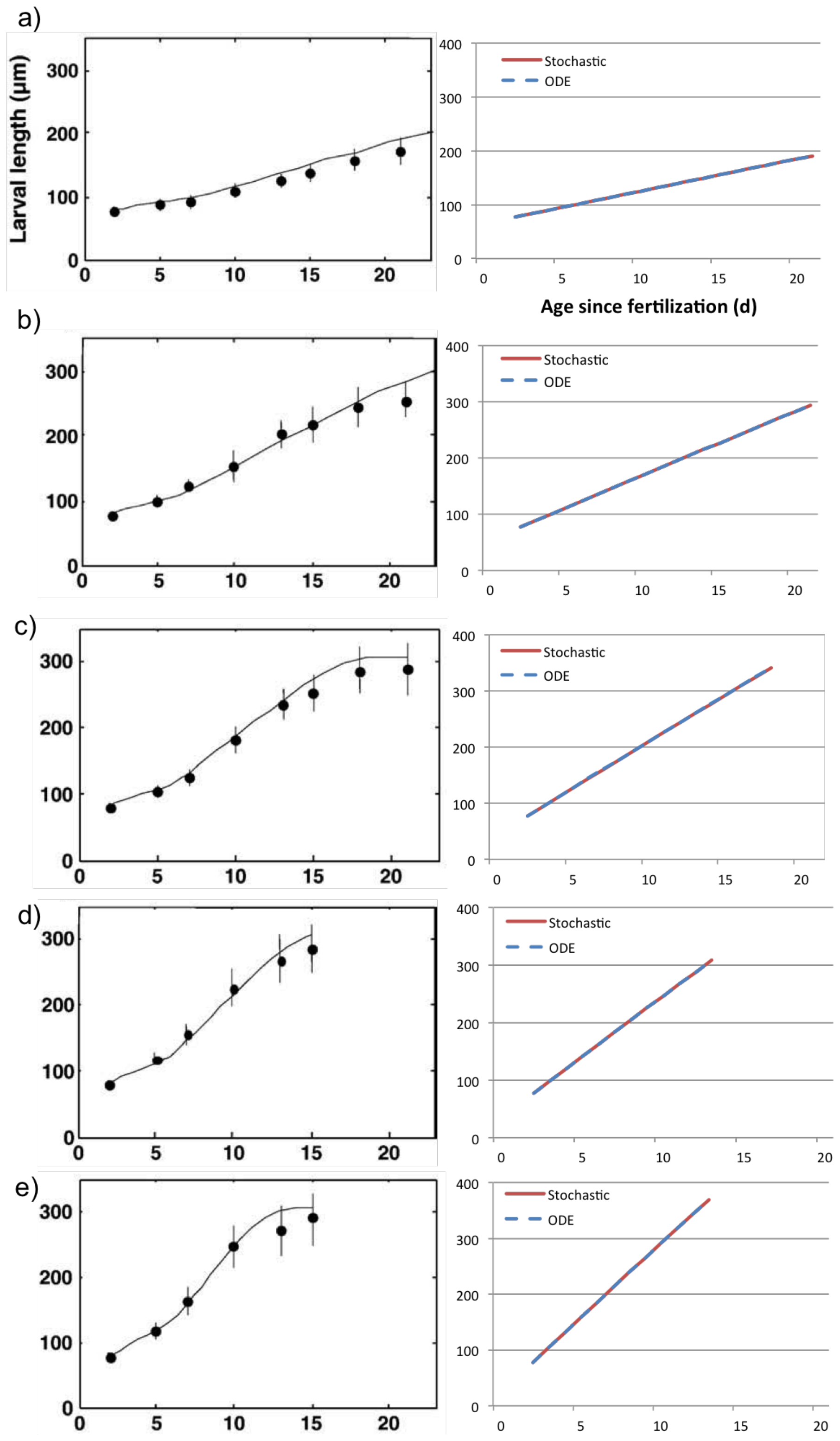


Figure 3.1: Experiment 1 comparison of Larval Length results of DEB model left [67] and Bio-PEPA model (ODEs and stochastic results) right under different temperatures: (a) 17, (b) 22, (c) 25, (d) 27 and (e) 32 °C. The DEB model includes comparison of observations ± SD (dots).

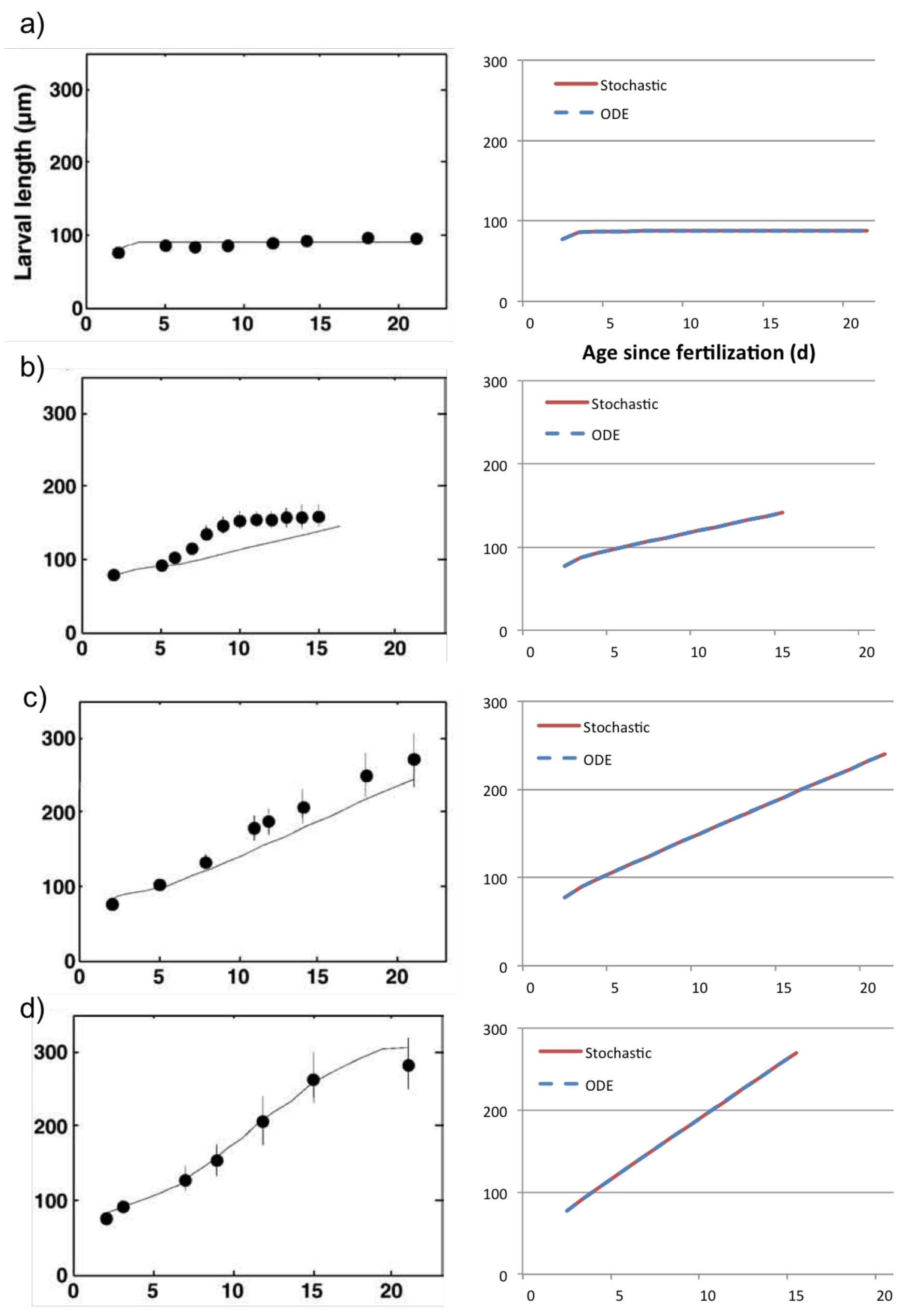


Figure 3.2: Experiment 2 (part 1 of 2) comparison of Larval Length results of DEB model left [67] and Bio-PEPA model (ODEs and stochastic results) right under different food density conditions: (a) 70, (b) 280, (c) 450 and (d) 960 $\mu m^3 \mu l^{-1}$. The DEB model includes comparison of observations $\pm$ SD (dots).

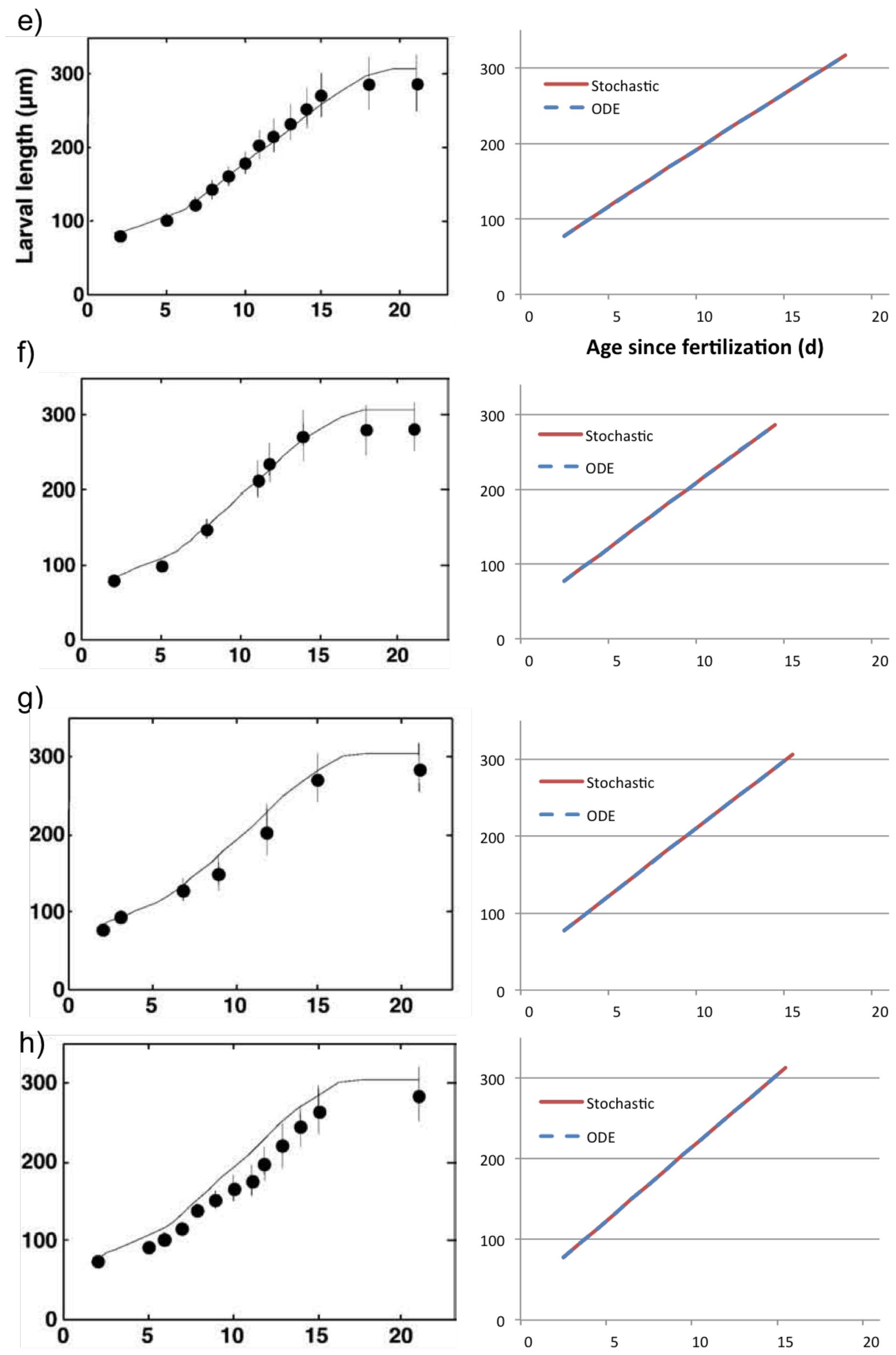


Figure 3.3: Experiment 2 (part 2 of 2) comparison of Larval Length results of DEB model left [67] and Bio-PEPA model (ODEs and stochastic results) right under different food density conditions: (e) 1000, (f) 1900, (g) 2100 and (h) 3300 $\mu m^3 \mu l^{-1}$. The DEB model includes comparison of observations $\pm$ SD (dots).

Larval growth and Rico-Villa et al [67] stated that an optimal food supply of a minimum level of 1000 $\mu m^3 \mu l^{-1}$ would be necessary to enhance larval growth and success to metamorphosis. The Larval length values for this experiment in the original DEB model and my Bio-PEPA model is given in Figure 3.3. The stochastic and ODEs simulation results of the Bio-PEPA model are similar indicating consistency in the model.

The Bio-PEPA model produced comparable results to the original DEB model. The statistical comparison R^2 results of the Bio-PEPA model comparison with the DEB model is given in Table 3.2. The Bio-PEPA result for 70 $\mu m^3 \mu l^{-1}$ Figure 3.3 (a) is less comparable to the original DEB model, this may be a result of the low value of the food density. The other Bio-PEPA results in this statistical comparison are more comparable to the original DEB model validating the Bio-PEPA model as a accurate translation.

3.2 INTEGRATED LIFE STAGE MODEL

An integrated life stage model was implemented linking my two separate validated Larva and Juvenile-Adult Bio-PEPA models. My integrated life stage Bio-PEPA model is given in Appendix A. This novel model includes three distinct life stages of the Pacific oyster (Larva, Juvenile and Adult). There is a need to have all three life stages included in one model as each stage is affected differently with environmental changes [7, 14]. The model was built to firstly ascertain mechanisms within Bio-PEPA to link the stages and secondly to highlight problems which arise from this integration. These challenges identify features that need to be included in a multi-scale process algebra language.

3.2.1 *Linking the life stages*

To integrate and link the two models a switching life stage mechanism (L_Switch_J) was implemented using a simple Heaviside step function. The switching mechanism is a conditional for a range of Boolean expressions. This function tracks when the length of the oyster reaches 300μm; Rico-Villa et al [67] stated that at this length an oyster begins the process of metamorphosis and changes from a Larva to a Juvenile instantaneously. The timing of this switch mechanism became important to realistically model when the Larva changes to a Juvenile. The value of L_Switch_J equals 1 when the model is in the Larval life stage and 0 when it is in the Juvenile-Adult life stage. An advantage of this mechanism is that if the length value for metamorphosis changes the model only needs updated in one place (the parameter Metamorphosis). L_Switch_J is used to carry out three changes throughout the model. These changes include the values of parameters, some equations and units. Figure 3.4 shows the switching mechanism and the three ways it is used throughout the model through three examples.

Switching mechanism from Larva to Juvenile

$$\begin{aligned}
\mathit{Metamorphosis} &= 300; \\
\mathit{L\_Switch\_J} &= H(\mathit{metamorphosis} - L);
\end{aligned}$$

Example of switch used for changing the value of parameters

$$T_L = (285 * \mathit{L\_Switch\_J}) + (281 * (1 - \mathit{L\_Switch\_J}))$$

Example of switch used for changing an equation

$$\mathit{Functional\_response} = \left(\frac{\mathit{Food\_density}^2}{(\mathit{Food\_density}^2 + X_\kappa^2)} * \mathit{L\_Switch\_J}\right) + \left(\frac{\mathit{Food\_density}}{(\mathit{Food\_density} + X_\kappa)} * (1 - \mathit{L\_Switch\_J})\right);$$

Example of switch used for changing of units

$$[\dot{P}_M] = (((24 * 10^{-12}) * \mathit{L\_Switch\_J}) + (24 * (1 - \mathit{L\_Switch\_J}))) * \textit{Temperature\_correction};$$

Figure 3.4: Pacific oyster integrated life stage Bio-PEPA model switching mechanism for Larva to Juvenile-Adult life stage.

Parameters: Several parameters needed to be switched, for example, the lower boundary temperature tolerance of a Larva is lower from that of its next life stage.

Equations: Some equations had to be switched, for example, the functional response for food consumption of the Larva model was of type 3 compared to the Adult which was of type 2 [59, 67]. Many of the equations did not change as both of my models use the DEB approach and have generic parameters. This consistency was an advantage as it simplifies the integration process.

Units: The two separate models focus on a specific time and physical scale for their specific biological system. Some units of parameters had to be switched appropriately. The Larva physical scale is in μm and its time scale is in days compared to the Adult's physical scale in cm and time scale of months. Most models focus on one unit of scale [59, 67]: the challenge was to allow the integrated life stage model to focus on the units that are appropriate for the life stage it is simulating. This was somewhat achieved by utilising L_Switch_J so that the model must switch its parameters' spatial units at an appropriate time to focus on a specific life stage, however, the model's agents units are scaled to the Larva model agent scale.

As the model contains two sets of parameters for the Larva and Juvenile-Adult models and further includes the equations to convert the state variables to dry flesh weight and the length of an oyster needed for comparison to observational data, this resulted in the integrated model being large with many lines of code. The model is scaled to the Larva agent initial values 10^6, therefore agent values for the Juvenile and Adult life stages become very large. This makes the stochastic simulations infeasible to run. This is partly overcome as the Bio-PEPA tool can perform ODEs simulations. It would be useful to switch scaling of values appropriately depending on the model's life stage. This cannot be implemented in Bio-PEPA. This issue has to be overcome in a multi-scale process algebra language.

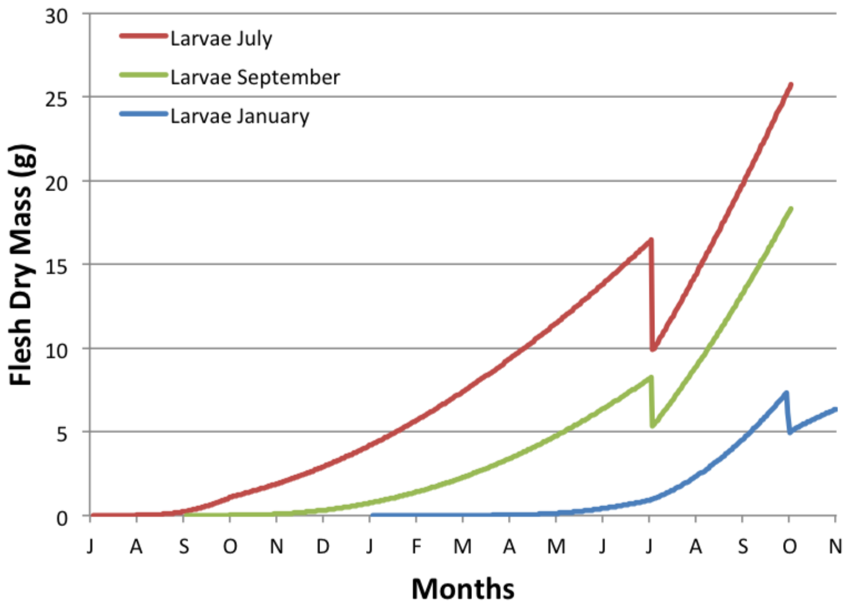


Figure 3.5: Total dry flesh weight for oysters produced in July, September and January. The sharp drops show the first spawning events of these oysters.

3.2.2 *Analysis*

Experiments were carried out using the knowledge of the system to demonstrate the value of the integrated life stage model. This allows the validation of my model. There were 3 experiments and the difference between them are when the larvae are produced from the first spawning event. Gosling [32] states that Pacific oyster spawning occurs between July and September. In the first experiment the Larva life stage commenced in July showing an early spawning, in the second experiment the Larva life stage began in September, indicating a late spawning. In the third experiment the Larvae begins their Life stage in January, this is an unrealistic spawning time to show the models predictive capacity to deal with this situation.

The time scale of each experiment was 450, 390 and 360 days respectively. For simplicity a month in all experiments had 30 days making a year 360 days. The difference in time scales reflects the inclusion of the progression of all the life stages from Larva to Juvenile to Adult until the first spawning event. The temperature for the months of July to September was set to the optimal value of 25°C [67], the remaining months had the temperature of 15°C. This temperature is too low for spawning to occur but high enough for the lower temperature tolerance range for all life stages [59, 67]. The food density was set at a constant optimal value of $1400\mu m^3 \mu l^{-1}$ [67], this ensures the results from the model are affected by the temperature forcing variable. The Bio-PEPA model generated ODEs simulation results for these experiments and the results are shown in Figure 3.5.

For Larvae produced in July and September their first spawning events as Adults occur in early July. The Larvae produced in January spawn as Adults in late September with the weight before spawning at 7g. The weight of the Adult oyster that was produced in July was 16g at the time before their first spawning. The weight of the Adult oyster produced in September was 8g at the time before their first spawning. The model predicts that Larvae that are produced early in the spawning season i.e. July, at the Adult stage of first spawning will weigh double than those produced in September. Both Larvae that were produced in July and September switched to the Juvenile life stage at 14 days, this is consistent with Rico-Villa et al [67]. It takes 52 days for the Larvae produced in January to switch to the Juvenile life stage, this shows spawning in January is impossible. January is indeed an unrealistic spawning time as the temperature is below the optimal temperature for Larvae growth which is 25°C [67].

This analysis shows the model can be used effectively to predict the best time of year that Larvae should be produced to ensure optimal weight and earlier spawning in their Adult life stage. The experiments were simplistic in the forcing variable values of temperature and food density, more detailed values may give more informative results. To validate the model, further observation data is needed from wet laboratory experiments that include all the three life stages.

3.3 SUMMARY

This Chapter illustrated that my translation steps created for the Juvenile-Adult model in Chapter 2 worked successfully for the Larva life stage Bio-PEPA model. This shows the steps are generic. It further illustrated that Bio-PEPA can model a system where the values are small and the units are different (μm) therefore the model allows the user to focus on a specific life stage. The Bio-PEPA model is validated against the original DEB model results.

The two validated life stage models can be seamlessly integrated using a switching mechanism. This switching mechanism allows the user to change easily the value for the switch from Larva to Juvenile in one place. Analysis of this integrated model shows the use and predictive capacity to allow questions to be asked about the whole progression of three life stages. Rico-Villa et al [67] stated that it would be an attractive aspect to have a unique DEB model to simulate growth from larva to adult and thus to encompass the whole life cycle of Pacific oyster. This has been accomplished in my integrated Bio-PEPA model and it is the first DEB Bio-PEPA model to detail the whole life-span of the Pacific oyster.

3.4 DISCUSSION

On critical analysis of the integrated Bio-PEPA model the main multi-scale challenges and features identified are discussed as follows:

The integration of the two Bio-PEPA models creates a large model which is difficult to read. This is due to the Bio-PEPA language not having integrative features that modularise the multi-scale details.

The unit of scale of the organism's internal system has to be set to the life stage with the smallest unit of scale e.g. the Larva life stage μm. This results in large values being processed for the other life stages which makes some analysis techniques in the Bio-PEPA plug-in computationally expensive.

Population actions such as the addition or removal of organisms are not included in this model. The integrated Bio-PEPA model does not model the population scale of an organism explicitly, therefore, there is no population view of the system and no analysis can be completed in this scale. Analysis of the population is essential to gain insight on how the changes at other scales impacts on the population as a whole e.g. ocean acidification affecting specific life stages of marine organisms has an impact on their persistence as a population.

These challenges of effectively integrating and modularising multi-scales, scaling of units appropriate to an organism's life stage and including a population view of the system are addressed in the next chapter with the conceptualisation and definition of PAL.

4

PAL: PROCESS ALGEBRA WITH LAYERS

This Chapter introduces Process Algebra with Layers (PAL): a language for multi-scale integration modelling. The novel features of PAL are the layers Population and Organism. The multi-scale integration modelling challenges and features from the previous Bio-PEPA models allowed the conceptualisation of these layers. PAL modularises the spatial scales of the system with these layers. The layers seamlessly integrate the scales through mirrored actions allowing multi-scale interactions. The PAL layers, for example, provide the modelling of an individual's physiology at different life stages within a population view within one PAL model. The modularisation allows the appropriate unit of scale to be used depending on the individual's life stage.

The syntax and semantics of PAL are defined. A simple example PAL model is given and example transitions are shown. A comparison of PAL with process algebra languages: psPAH and PEPA nets is presented. The implementation of a PAL parser for the automatic translation of a PAL model to a Bio-PEPA model is shown.

4.1 LAYERS OF THE LANGUAGE: POPULATION AND ORGANISM

As explained above, PAL has two layers which are named Population and Organism. These two layers were defined to encapsulate the features of multi-scale systems. An Organism is defined as an individual internal system model of internal species components. An Organism can represent a molecule, organelle, cell, tissue, organ or any organism. An internal species component, dependent on the scale of the Organism layer, can also represent a molecule, organelle, cell, tissue, organ or any organism. The internal system model of an Organism can be, for example, a DEB model which describes the physiology with the internal species components representing energy budgets. The internal system model can indeed represent any abstract system. Populations hold specific types of Organism, for example, life stages, cell phases, infectious states etc. Figure 4.1 shows a conceptual schematic of these layers, which are described in more detail in Sections 4.1.1 and 4.1.2.

The two layers interact through mirrored and hidden actions discussed in more detail in Section 4.2.2.1. Internal actions of the Organism's internal system are mirrored by external Population actions allowing multi-scale transitions. Figure 4.2 shows example transitions of a PAL model. Specific multi-scale transitions that can be easily modelled in PAL include: life stage switch, reproduction and death transitions. These transitions are achieved through

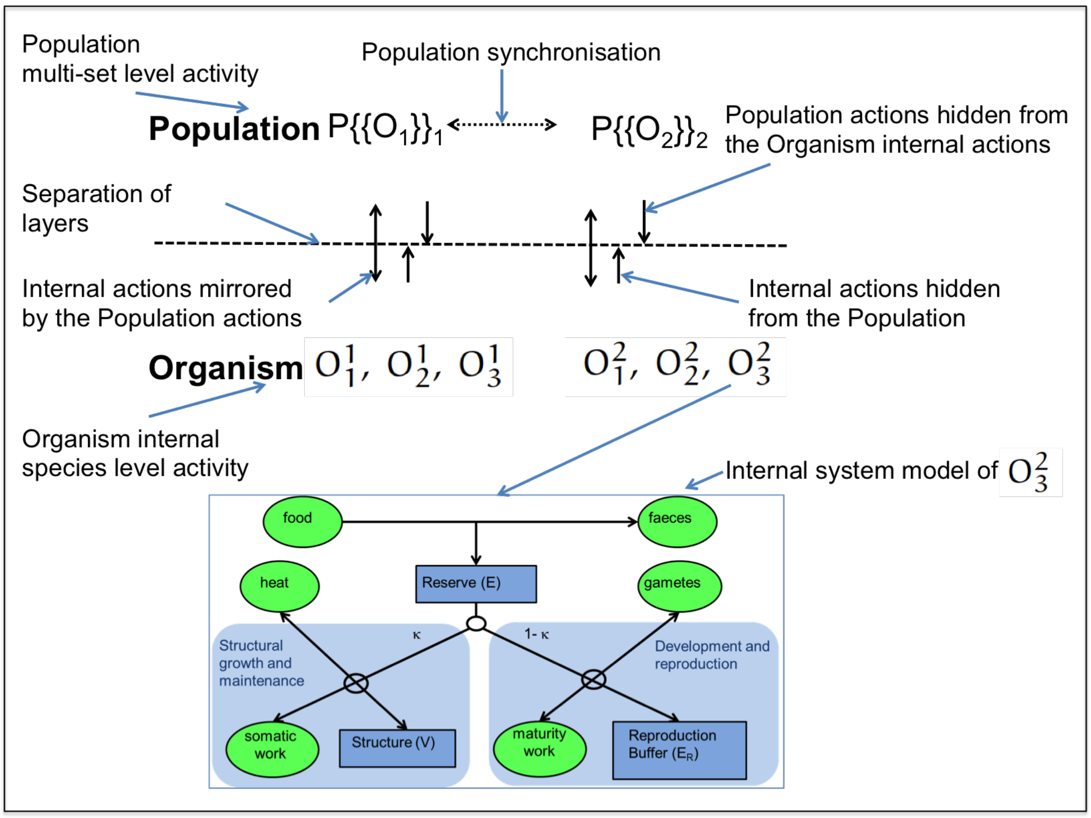

Population multi-set level activity
Population synchronisation
Population P{{O1}}1
P{{O2}}2
Population actions hidden from the Organism internal actions
Separation of layers
Internal actions mirrored by the Population actions
Internal actions hidden from the Population
Organism O11, O12, O13
O21, O22, O23
Organism internal species level activity
Internal system model of O23
food
faeces
heat
Reserve (E)
gametes
κ
1- κ
Structural growth and maintenance
Development and reproduction
somatic work
Structure (V)
maturity work
Reproduction Buffer (ER)

Figure 4.1: Schematic of Population and Organism Layers. Organism numbering should be read as follows: O^1_2 is Organism 2 of Population 1.

mirrored actions and Population synchronisation. Figure 4.2 shows a life stage transition where Organism $\mathbf{O}_2^1$ performs a mirrored action and is removed from the Population P{{O$_1$}}$_1$ and new Organism $\mathbf{O}_4^2$ is added to Population P{{O$_2$}}$_2$. The Figure also shows a reproduction transition where Organism $\mathbf{O}_2^2$ carries out a mirrored action that causes the two Populations to synchronise and a new Organism $\mathbf{O}_4^1$ is added to the Population P{{O$_1$}}$_1$. Death transition is shown where the Organism $\mathbf{O}_1^1$ carries out a mirrored action causing this Organism to be removed from the Population P{{O$_1$}}$_1$.

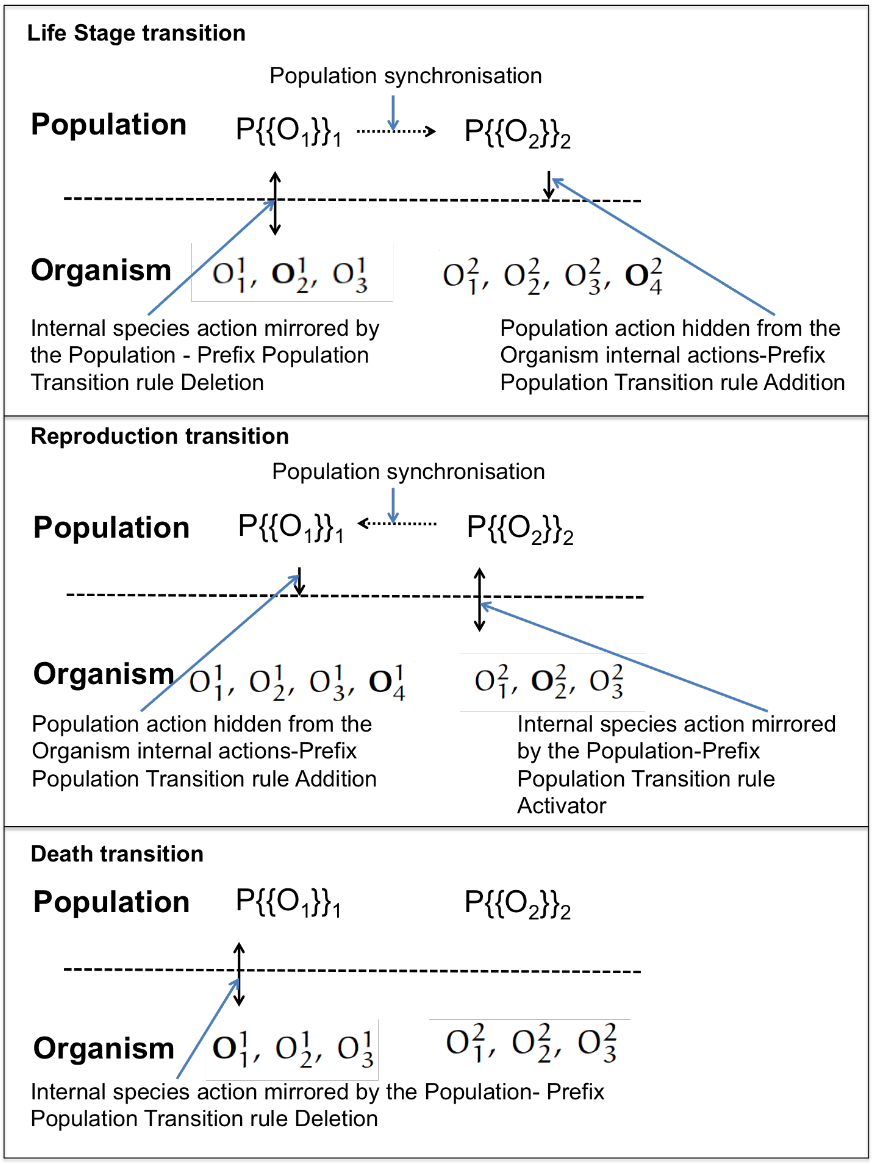

Life Stage transition
Population synchronisation
Population
P{{O1}}1
P{{O2}}2
Organism
O11, O21, O31
O12, O22, O32, O42
Internal species action mirrored by the Population - Prefix Population Transition rule Deletion
Population action hidden from the Organism internal actions-Prefix Population Transition rule Addition
Reproduction transition
Population synchronisation
Population
P{{O1}}1
P{{O2}}2
Organism
O11, O21, O31, O41
O12, O22, O32
Population action hidden from the Organism internal actions-Prefix Population Transition rule Addition
Internal species action mirrored by the Population-Prefix Population Transition rule Activator
Death transition
Population
P{{O1}}1
P{{O2}}2
Organism
O11, O21, O31
O12, O22, O32
Internal species action mirrored by the Population- Prefix Population Transition rule Deletion

Figure 4.2: Example transitions of a PAL model. Organisms which are invovled in transitions are highlighted in bold.

4.1.1 *Population*

A Population describes a specific population (life stage) of an Organism. For example, in Figure 4.1 $P\{\{O_1\}\}_1$ and $P\{\{O_2\}\}_2$ could describe the larva and juvenile life stage from the Pacific oyster example. A Population is a multi-set that holds a population of Organisms as defined in Figure 4.3. A multi-set is a set that can contain duplicate elements, therefore, a Population can contain duplicate Organisms. In some cases duplicate Organisms can occur, for example, at initial set up if there is more than one Organism in a Population, these Organisms will have the same initial internal parameter values set up. The Population layer allows the physiological view of an Organism to be linked to a population view.

4.1.2 *Organism*

An Organism is an individual internal system model of internal species components as shown schematically in Figure 4.1. For example, O_3^2 has an individual DEB model description. Each Organism holds specific initial set up information about the internal system model of internal species relevant to that specific population (life stage). For example, the initial set up of a larva life stage would be different from the initial set up of a juvenile life stage. Allocation of energy and how tolerant the organism is to forcing variables will change dependent on life stage. As an Organism has a internal system model description it has internal species components which dynamically change over a time period to evolve the Organism's physiology, and which generate internal actions. Some of these internal actions have an impact on the population view of the system.

4.2 PROCESS ALGEBRA WITH LAYERS

4.2.1 *The Syntax of PAL*

The syntax of PAL is given in Figure 4.3. The component $P\{\{O\}\}_A$ is called a Population component and represents a multi-set of Organism components $P\{\{O\}\}_A = P\{[\![O_1, ..., O_n]\!]\}_A$. A multi-set is an unordered collection of objects with repetitions allowed. The component O called an Organism component describes an internal system and the interactions among internal species components S. PAL uses the same syntax as Bio-PEPA [21] to define the internal species components. The component M, called a model component, describes the system and the interactions among Population components. The element C is a constant which is a component whose meaning gives a defining equation $C = S$. The element x is a positive real-valued parameter. The element D is a constant, which is a component whose

$$
\begin{aligned}
P\{\{O\}\}_A :: &= (\alpha, 1)\ PAL_{op}\ P\{\{O\}\}_A \mid P\{\{O\}\}_A\ +\ P\{\{O\}\}_A \mid D \\
\text{Where } PAL_{op} &= \downarrow \mid \uparrow \mid ((+)) \\
O :: &= O \underset{L}{\bowtie} O \mid S(x) \\
S :: &= (\alpha, \kappa)\ op\ S \mid S + S \mid C \\
\text{Where op} &= << \mid >> \mid (+) \mid (-) \mid (\ .\ ) \\
M :: &= M \underset{L_s}{\diamondsuit} M \mid P\{\{O\}\}_A
\end{aligned}
$$

Figure 4.3: Syntax of PAL.

meaning is given by a defining equation $D = P\{\{O\}\}_A$. Constants allow us to assign names to patterns of behaviour associated with components.

(α,1) is the prefix, where $\alpha \in$ Actions is the action type and 1 is the stoichiometry coefficient of the Organisms in that action. The design choice of a stoichiometry of 1 was chosen to simplify the resulting states the action produces. There are three prefix combinators called $PALop$ which represent the role of the Organisms in the action. These are: $\downarrow$ indicates a deletion of an Organism, $\uparrow$ an addition of an Organism (an `initialO` element will be added to a specific Population with a specific initial set up relevant to its Population) and $((+))$ an Organism which is an activator. (α, κ) is the internal species prefix, where $\alpha \in$ SpeciesActions is the action type and k is the stoichiometry coefficient of the species in that reaction. The prefix combinators op are: $<<$ indicating a reactant, $>>$ a product, $(+)$ an activator, $(-)$ an inhibitor and $(\ .\ )$ a generic modifier. The choice operator for $P\{\{O\}\}_A + P\{\{O\}\}_A$ and $S + S$ is the same because it represents the same non-deterministic choice between actions whether these be Population actions or internal species actions. Once one is chosen the others are discarded. Thus the choice combinator represents competition between actions depending on their rate.

$O \underset{L}{\bowtie} O$ denotes the cooperation between internal species over the cooperation set L. Set L determines those activities the cooperands are forced to synchronise on. The cooperation between Populations over the multi-scale action cooperation set L_s is expressed by $M \underset{L_s}{\diamondsuit} M$. Set L_s determines those actions the cooperands are forced to synchronise on. There can be more than two Population components in a PAL model and each Population component must have a hidden action set A. The set A identifies those internal species actions which are hidden to the Population component. Hidden actions should not be in the set L_s in a well defined PAL system.

A PAL system $\mathcal{P}$ is a septuple $\langle Pcomp, Ocomp, Scomp, F_R, \mathcal{K}, \mathcal{N}, \mathcal{M}\rangle$, where:

- $Pcomp$ is the set of definitions of Population components;
- $Ocomp$ is the set of definitions of Organism components;
- $Scomp$ is the set of definitions of internal species components;
- F_R is the set of functional rate definitions;
- $\mathcal{K}$ is the set of parameter definitions;
- $\mathcal{N}$ is the set of quantities describing each internal species;
- $\mathcal{M}$ is the model component describing the system.

In a well-defined PAL system each element has to satisfy the following conditions. Set $\mathcal{N}$ has to contain all the internal species components. The functional rates are well defined if each variable in their definition refers to the name of a species component in the set $\mathcal{N}$ or a constant parameter in the set $\mathcal{K}$. The definition of the internal species components in $Scomp$ must have sub-terms of the form $(\alpha, k)op\ S$ and the action types in each single component must be all distinct. The definition of the Organism components in $Ocomp$ must be defined in terms of the internal species components defined in $Scomp$ and for each cooperation set L_i in O, $L_i \subseteq$ SpeciesActions (O). The definition of the Population components $Pcomp$ must be defined in sub-terms of the form $(\alpha,1)\ PALop\ P\{\{O\}\}_A$ and the action types in each single component must be all distinct. The model component $\mathcal{M}$ must be defined in terms of the Population components defined in $Pcomp$ and for each cooperation set L_{si} in $\mathcal{M}$, $L_{si} \subseteq$ Actions (M).

4.2.2 *The Semantics of PAL*

The semantics of PAL are defined in Figure 4.4, 4.5, 4.6 and 4.7. The rules of the language specify Population behaviour. The Prefix Population Transition Rules describe specific actions that a Population can perform, specifically the addition and deletion of an Organism from a Population. The Action Mirror/Hidden Rules describe how internal species actions in Organisms can be mirrored by external Population actions. Internal species actions in Organisms can also be hidden from the Population. These rules are discussed in Section 4.2.2.1. The Population Transition Rules define that a Population can perform actions independently and can also communicate and synchronise with other Populations. As stated previously PAL has similar syntax and semantics to Bio-PEPA to allow easy expansion of Bio-PEPA models within this language. For example, a Bio-PEPA model describing one life stage of an organism can be integrated to include more life stages in populations in PAL.

Action Mirror/Hidden Rules

Internal Actions that are seen and mirrored by Populations

$$\frac{O_i \xrightarrow{(\alpha,\ [S{:}op(l,k)])} O_i'}{P\{\{O\}\}_A \xrightarrow{\alpha} P\{\{O'\}\}_A} \ with\ \exists\ O_i \in O\ \wedge\ \alpha \notin A$$

Where $O' = O\ \oplus\ O_i' \wedge P\{\{O\}\} \xrightarrow{\alpha} P\{\{O'\}\}$

Where $\oplus$ overwrites O_i in O with new O_i' state, leaving the rest of O unchanged.

Where S is a sequential internal species component, op is the internal action type, l the level and k the stoichiometry coefficient.

Internal Actions that are hidden from Populations

$$\frac{O_i \xrightarrow{(\alpha,\ [S{:}op(l,k)])} O_i'}{P\{\{O\}\}_A \xrightarrow{\tau} P\{\{O'\}\}_A} \ with\ \exists\ O_i \in O\ \wedge\ \alpha \in A$$

Where $O' = O\ \oplus\ O_i'$

Where $\oplus$ overwrites O_i in O with new O_i' state, leaving the rest of O unchanged.

Where S is a sequential internal species component, op is the internal action type, l the level and k the stoichiometry coefficient.

Figure 4.4: Semantics of PAL.

Prefix Population Transition Rules

Internal species actions mirrored by Population actions

Adding an Organism to a Population

$((\alpha, 1) \uparrow P\{\{O\}\}_A) \xrightarrow{(\alpha, r)} P\{\{O'\}\}_A$ *with* $\alpha \notin A$

$O' = O \cup [\![\mathrm{initialO}]\!]$

Deleting a specific Organism from a Population

$((\alpha, 1) \downarrow P\{\{O\}\}_A) \xrightarrow{(\alpha, r)} P\{\{O'\}\}_A$ *with* $\alpha \notin A$

$\exists i.\ O_i \in O \wedge O_i \xrightarrow{(\alpha, r)} O_i' \wedge O = [\![O_1, ..., O_n]\!] \wedge |O| \geqslant 1 \wedge O' = O \backslash [\![O_i]\!]$

Activator does not increase or decrease a Population

$((\alpha, 1)((+))P\{\{O\}\}_A) \xrightarrow{(\alpha, r)} P\{\{O\}\}_A$ *with* $\alpha \notin A$

$O = [\![O_1, ..., O_n]\!] \wedge |O| \geqslant 1$

Where the rate r is the rate of internal species action (α, w).

Population actions hidden from Organism internal actions

Adding an Organism to a Population

$((\alpha, 1) \uparrow P\{\{O\}\}_A) \xrightarrow{(\alpha, r)} P\{\{O'\}\}_A$ *with* $\alpha \notin A$

$O' = O \cup [\![\mathrm{initialO}]\!]$

Deleting a random Organism from a Population

$((\alpha, 1) \downarrow P\{\{O\}\}_A) \xrightarrow{(\alpha, r)} P\{\{O'\}\}_A$ *with* $\alpha \notin A$

$\exists i.\ O_i \in O \wedge O = [\![O_1, ..., O_n]\!] \wedge |O| \geqslant 1 \wedge O' = O \backslash [\![O_i]\!]$

Activator does not increase or decrease a Population

$((\alpha, 1)((+))P\{\{O\}\}_A) \xrightarrow{(\alpha, r)} P\{\{O\}\}_A$ *with* $\alpha \notin A$

$O = [\![O_1, ..., O_n]\!] \wedge |O| \geqslant 1$

Where the rate r is the population action rate.

Figure 4.5: Semantics of PAL.

Population Transition Rules

Constant

$$\frac{P\{\{O\}\}_A \xrightarrow{(\alpha, r)} P\{\{O'\}\}_A}{D \xrightarrow{(\alpha, r)} P\{\{O'\}\}_A} \; \textit{with } D = P\{\{O\}\}_A$$

Asynchronous Left

$$\frac{P\{\{O\}\}_A \xrightarrow{(\alpha, r)} P\{\{O'\}\}_A}{P\{\{O\}\}_A \underset{L_s}{\Diamond} M \xrightarrow{(\alpha, r)} P\{\{O'\}\}_A \underset{L_s}{\Diamond} M} \; \textit{with } \alpha \notin L_s$$

Asynchronous Right

$$\frac{P\{\{O\}\}_A \xrightarrow{(\alpha, r)} P\{\{O'\}\}_A}{M \underset{L_s}{\Diamond} P\{\{O\}\}_A \xrightarrow{(\alpha, r)} M \underset{L_s}{\Diamond} P\{\{O'\}\}_A} \; \textit{with } \alpha \notin L_s$$

Population Synchronisation

$$\frac{P_1\{\{O_1\}\}_A \xrightarrow{(\alpha, r)} P_1\{\{O_1'\}\}_A \quad P_2\{\{O_2\}\}_A \xrightarrow{(\alpha, r)} P_2\{\{O_2'\}\}_A}{P_1\{\{O_1\}\}_A \underset{L_s}{\Diamond} P_2\{\{O_2\}\}_A \xrightarrow{(\alpha, r)} P_1\{\{O_1'\}\}_A \underset{L_s}{\Diamond} P_2\{\{O_2'\}\}_A} \; \textit{with } \alpha \in L_s$$

Where L_s is the multi-scale synchronisation action set

Figure 4.6: Semantics of PAL.

prefixReac $(\alpha, k) << S)(l) \xrightarrow{(\alpha,\,[S :<< (l,\kappa)])} S(l-k) \quad k \leqslant l \leqslant N$

prefixProd $(\alpha, k) >> S)(l) \xrightarrow{(\alpha,\,[S :>> (l,\kappa)])} S(l+k) \quad 0 \leqslant l \leqslant (N-k)$

prefixMod $(\alpha, k)\text{op } S)(l) \xrightarrow{(\alpha,\,[S : op(l,\kappa)])} S(l)$ with op = (.), (+), (-) and

$0 \ < \ l \leqslant$ N if op $= (+), \ 0 \ \leqslant \ l \leqslant$ N $otherwise$

Where S is the name of the species component, op is the action type, l the level and κ the stoichiometry coefficient.

choice1 $\dfrac{S_1(l) \xrightarrow{(\alpha,w)} S_1'(l')}{(S_1+S_2)(l) \xrightarrow{(\alpha,w)} S_1'(l')}$ choice2 $\dfrac{S_2(l) \xrightarrow{(\alpha,w)} S_2'(l')}{(S_1+S_2)(l) \xrightarrow{(\alpha,w)} S_2'(l')}$

constant $\dfrac{S(l) \xrightarrow{(\alpha,S:\,[op(l,k)])} S'(l')}{C(L) \xrightarrow{(\alpha,C:\,[op(l,k)])} S'(l')}$ $with\ C = S$

coop1 $\dfrac{O_1 \xrightarrow{(\alpha,w)} O_1'}{O_1 \underset{L}{\bowtie} O_2 \xrightarrow{(\alpha,w)} O_1' \underset{L}{\bowtie} O_2}$ $with\ \alpha \notin L$

coop2 $\dfrac{O_2 \xrightarrow{(\alpha,w)} O_2'}{O_1 \underset{L}{\bowtie} O_2 \xrightarrow{(\alpha,w)} O_1 \underset{L}{\bowtie} O_2'}$ $with\ \alpha \notin L$

coop3 $\dfrac{O_1 \xrightarrow{(\alpha,w)} O_1' \quad O_2 \xrightarrow{(\alpha,w)} O_2'}{O_1 \underset{L}{\bowtie} O_2 \xrightarrow{(\alpha,w)} O_1' \underset{L}{\bowtie} O_2'}$ $with\ \alpha \in L$

Where w is a list recording the species that participate in the reaction and L is the cooperation action set

Figure 4.7: Rules for Bio-PEPA included in the semantics of PAL. These rules are presented in [21] and are repeated here for convenience and completeness.

4.2.2.1 *External and Internal actions*

In PAL some of the Organism's internal actions are seen and some are hidden from the Population. The internal actions that are seen are mirrored by the Population as defined by the Action Mirror Rule in Figure 4.4. These actions could include internal actions such as life stage transition, reproduction and death. Each of these internal actions have an impact on the external population view of the system. These internal actions are mirrored by external Population actions which are defined by the Prefix Population Transition Rules shown in Figure 4.5.

There are essentially three Prefix Population Transition Rules: adding, deleting and activator. These rules are repeated twice in Figure 4.5 to show that Populations can perform actions independently from Organisms. These rules are asymmetric because when deleting an organism from a population the rule needs to identify the specific organism that is to be deleted. For example, the deletion rule can be used for life stage transitions and deaths of specific organisms, therefore the organism must be known to the rule so that the correct organism is deleted. In the case of addition and the activator rule a specific organism does not need to be known. For example, in the addition rule a new initialisation of an organism is added to a population. The organism in the activator rule does not need to be known by the rule as the rule does not change the organism.

Some internal actions are hidden from the Population level as defined in the Action Hidden Rule in Figure 4.4. The modeller defines a set of hidden actions when describing a model in PAL. These could include actions such as energy allocation and utilisation within an Organism. Actions such as these do not change the external population of the system, therefore do not need to be seen by the Population.

In comparison with Process Algebra with Hooks [24] PAL is simple and restricted because external Population actions are not allowed to be mirrored by internal Organism actions. Therefore, communication between layers is only from internal actions to external actions. This approach was adopted in consideration of the marine organism systems researched, to construct an elegant language, restrict the complexity of the rules and reduce state space explosion.

Populations can perform actions independently and this allows actions such as death to be defined in a model. Populations synchronise/communicate on specific external actions as defined in the Population Transition Rules shown in Figure 4.6. This allows the definition of life stage transitions and reproduction as these actions involve two Populations to change in number.

4.2.2.2 *PAL Parser*

A PAL model parser has been implemented in the Python programming language utilising the PLY tool. The parser source code can be found in the following repository [70]. PLY is an implementation of lex and yacc parsing tools for Python [42]. The lex.py module is used to break input text into a collection of tokens specified by a collection of regular expression rules. For example, the tokens in the PAL model parser include the PAL prefix combinators deletion, addition and activator etc. The yacc.py module is used to recognise language syntax that has been specified in the form of a context free grammar. The grammar rules of the PAL model parser define how to manipulate and translate different parts of the PAL model into a Bio-PEPA model. Rules include how to specify parameters, action rates, internal species behaviours, hidden actions set, Organisms, Populations and the model component etc. Hence, the PAL model parser formally translates a PAL model into a Bio-PEPA model. Essentially a PAL model with one Population with no Population actions and that has a Population of one Organism is a Bio-PEPA model. A parser was chosen to be implemented instead of a PAL model solver because the parsed models can utilise the various analysis techniques available within the Bio-PEPA plug-in. Users already familiar with the Bio-PEPA language and the plug-in can easily create PAL models.

An example of the automatic translation the PAL parser performs is shown in Appendix B. A toy PAL model in the PAL parser file format is shown in Figure B.1 and the Bio-PEPA model the Parser produces is in Figures B.2 and B.3. The toy model is made up of two populations: One and Two. Population One contains Organisms of type A and Population Two contains Organisms of type B. An A organism has an internal system made up of three species: X, $Tracker\_off$ and $Tracker\_on$. A B organism has an internal system of two species Y and Z. When internal species X reaches the threshold of 10 it triggers the mirrored population action **switch** which deletes the A organism and also synchronises with Population Two which adds a new B Organism. Population Two can perform the asynchronous Population action **remove** at a constant rate.

4.2.2.3 *Underlying CTMC of PAL models*

The PAL parser formally translates PAL models to Bio-PEPA therefore a Continuous-time Markov chain (CTMC) can be derived from a PAL model. This can be achieved as once translated into a Bio-PEPA model, the model can be applied to model checking tools available to Bio-PEPA models. These tools allow the derivation of Bio-PEPA models' CTMC. An example of a PAL model's CTMC is described below using the toy PAL model in Figure B.1.

The toy PAL model in Figure B.1 has nine reachable states from the initial state and these are shown in Figure 4.8. The states and transitions of the model are shown in Figure 4.9.

$$
\begin{aligned}
&1: \mathit{One}\llbracket A_1, A_2 \rrbracket_H \quad \lozenge \quad \mathit{Two}\llbracket\ \rrbracket_H\\
&2: \mathit{One}\llbracket A_1 \rrbracket_H \quad \lozenge \quad \mathit{Two}\llbracket B_1 \rrbracket_H\\
&3: \mathit{One}\llbracket A_2 \rrbracket_H \quad \lozenge \quad \mathit{Two}\llbracket B_1 \rrbracket_H\\
&4: \mathit{One}\llbracket\ \rrbracket_H \quad \lozenge \quad \mathit{Two}\llbracket B_1, B_2 \rrbracket_H\\
&5: \mathit{One}\llbracket A_2 \rrbracket_H \quad \lozenge \quad \mathit{Two}\llbracket\ \rrbracket_H\\
&6: \mathit{One}\llbracket\ \rrbracket_H \quad \lozenge \quad \mathit{Two}\llbracket B_2 \rrbracket_H\\
&7: \mathit{One}\llbracket A_1 \rrbracket_H \quad \lozenge \quad \mathit{Two}\llbracket\ \rrbracket_H\\
&8: \mathit{One}\llbracket\ \rrbracket_H \quad \lozenge \quad \mathit{Two}\llbracket B_1 \rrbracket_H\\
&9: \mathit{One}\llbracket\ \rrbracket_H \quad \lozenge \quad \mathit{Two}\llbracket\ \rrbracket_H
\end{aligned}
$$

Figure 4.8: The nine reachable states of the toy PAL model in Figure B.1.

$$\begin{tikzcd}
1 \arrow[r, "s"] \arrow[d, "s"'] & 3 \arrow[r, "r"] \arrow[d, "s"'] & 5 \arrow[dll, "s"'] \\
2 \arrow[r, "s"] \arrow[d, "r"'] & 4 \arrow[r, "r"] \arrow[d, "r"'] & 6 \arrow[d, "r"'] \\
7 \arrow[r, "s"] & 8 \arrow[r, "r"] & 9
\end{tikzcd}$$

Figure 4.9: Reachable states and transitions of the toy PAL model in Figure B.1. s and r represent Population actions switch and remove respectively.

4.3 SIMPLE MODEL EXAMPLE USING PAL

To show how specific multi-scale features can be defined by PAL the following simple model is used. This model describes a generic simplistic marine organism which has two life stages: Larva and Juvenile. A Larva will switch to become a Juvenile when a structural volume threshold has been reached. In this example Juvenile organisms can reproduce Larvae dependent on the energy they have reserved for reproduction reaching a specific threshold. For this simple model the term Juvenile encompasses the Juvenile/Adult life stage. Organisms in both life stages can die based on a simple death rate. The initial population is two Larvae and three Juveniles.

4.3.1 *Model configuration*

To describe this system two Populations are defined: Larva and Juvenile. These Populations hold specific life stage Organisms. The Organisms describe the DEB physiology of an Organism under a specific life stage by internal species components. The Populations also have external actions which mirror the actions of the internal species allowing multi-scale transitions such as life stage transitions, reproduction and death. The Populations synchronise on specific actions in order to communicate on certain transitions such as life stage transitions where both Populations change or one Population has a direct effect on another, such as reproduction. Internal actions which are hidden from the Population layer are defined in the HiddenActions set A. The Populations have external actions which are hidden from the Organism internal actions. These include actions that add new J Organisms and delete random Organisms. The initial model configuration initialises two L Organisms in the Larva Population and three J Organisms in the Juvenile Population. Figures 4.10 and 4.11 shows the formal definition of the model.

4.3.2 *Transitions*

The following transitions occur when the internal actions are mirrored by external actions causing a multi-scale transition.

4.3.2.1 *Life stage switch*

A life stage switch occurs in the model when an L Organism based on internal actions is ready to switch to become a J Organism. An internal action occurs in the Organism layer and is mirrored by an external Larva Population action. The transition takes place at the Population layer, the external Larva Population action deletes the specific Organism L from its multi-set

General parameters

K = 0.5;

Deathrate = 0.05;

L life stage parameters

LPc = 0.5*LE;

LPa = 0.5;

LPm = 0.25*LV;

LPj = 0.3*LER;

LVT = 260;

LtoJ = H(LV − LVT);

J life stage parameters

JPc = 0.55*JE;

JPA = 0.6;

JPm = 0.3*JV;

JPj = 0.35*JER;

JERT = 200;

JReproduce = H(JER − JERT);

ReproduceStop = H(1 − JER);

L internal species actions

LG : K*LPc;

LS : LPm;

LA : LPa;

LC : LPc;

LR : (1 − k)*LPc;

LJ : ((1 − k)\k)*LPj;

LSwitch : fMA(100 ∗ (LtoJ));

LDie : Deathrate;

J internal species actions

JG : K*JPc;

JS : JPm;

JA : JPa;

JC : JPc;

JR : (1 − k)*JPc;

JJ : ((1 − k)\k)*JPj;

ReproduceL : fMA(10*(JReproduce));

StopReproduceL : fMA(10*(ReproduceStop));

Empty : fMA(100);

JDie : Deathrate;

Figure 4.10: Formal Definition of example model in PAL Part 1 of 2. Please note the layout is used to indicate sections and is not part of the syntax. Also note population actions that mirror internal organism actions get their rate from the internal organism action.

L internal species

$LE = LA >> +LC << +LG(.) + LR(.);$

$LV = LG >> +LS << +LA(.) + LC(.) + LJ(.);$

$LER = LR >> +Lj <<;$

$Switch\_on = (LSwitch, 1) >>;$

$Switch\_off = (LSwitch, 1) <<;$

$LDTracker\_on = (LDie, 1) >>;$

$LDTracker\_off = (LDie, 1) <<;$

J internal species

$JE = JA >> +JC << +JG(.) + JR(.);$

$JV = JG >> +JA(.) + JC(.) + JR(.) + JJ(.);$

$JER = JR >> +Empty <<;$

$Tracker\_on = (ReproduceL, 1) >> +$

$\quad (StopReproduceL, 1) << +Empty(+);$

$Tracker\_off = (ReproduceL, 1) >> +$

$\quad (StopReproduceL, 1) >>;$

$JDTracker\_on = (JDie, 1) >>;$

$JDTracker\_off = (JDie, 1) <<;$

Set of hidden internal actions

$HiddenActions\ A : \{LG, LS, LA, LC, LR, JG, JS, JA, JC, JR, JJ, StopReproduceL\};$

Organisms (Internal species model)

$L = LE[100] \bowtie LV[250] \bowtie LER[0] \bowtie Switch\_on[0]$

$\quad \bowtie Switch\_off[1] \bowtie LDTracker\_on[0] \bowtie LDTracker\_off[1]$

$J = JE[200] \bowtie JV[260] \bowtie JER[100] \bowtie Tracker\_on[0]$

$\quad \bowtie Tracker\_off[1] \bowtie JDTracker\_on[0] \bowtie JDTracker\_off[1]$

Population action rates

$RandomLDie : 0.02;$ $RandomJDie : 0.06;$ $RandomJAdd : 0.02;$

Populations (External actions)

$Larva\{\{L\}\} = LSwitch \downarrow +ReproduceL \uparrow +LDie \downarrow +RandomLDie \downarrow;$

$Juvenile\{\{J\}\} = LSwitch \uparrow +ReproduceL((+)) + JDie \downarrow +RandomJDie \downarrow +RandomJAdd \uparrow;$

Model Component

$$Larva[\![L_1, L_2]\!]_A \underset{\{LSwitch,\ ReproduceL\}}{\diamond} Juvenille[\![J_1, J_2, J_3]\!]_A$$

Figure 4.11: Formal Definition of example model in PAL Part 2 of 2. Please note the layout is used to indicate sections and is not part of the syntax. Also note population actions that mirror internal organism actions get their rate from the internal organism action.

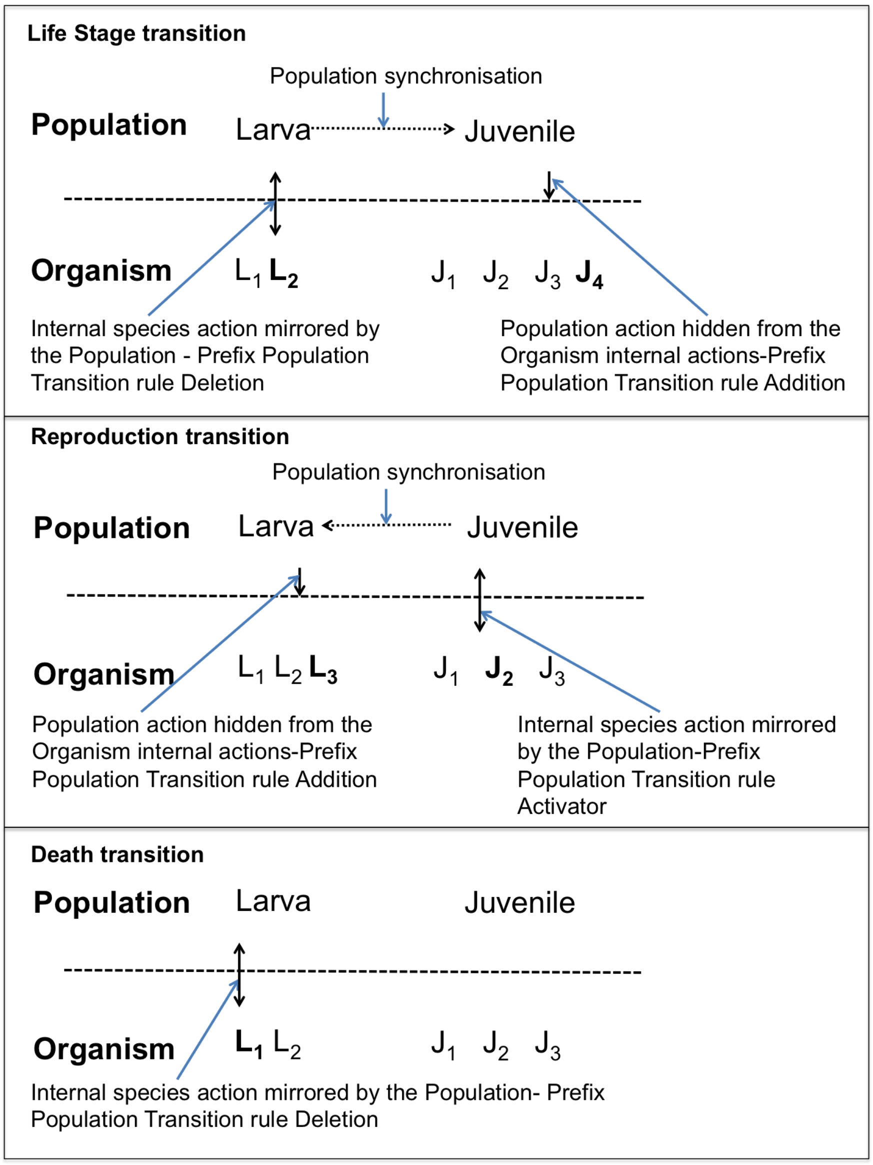

Life Stage transition
Population synchronisation
Population
Larva
Juvenile
Organism
L_1 L_2
J_1 J_2 J_3 J_4
Internal species action mirrored by the Population - Prefix Population Transition rule Deletion
Population action hidden from the Organism internal actions-Prefix Population Transition rule Addition
Reproduction transition
Population synchronisation
Population
Larva
Juvenile
Organism
L_1 L_2 L_3
J_1 J_2 J_3
Population action hidden from the Organism internal actions-Prefix Population Transition rule Addition
Internal species action mirrored by the Population-Prefix Population Transition rule Activator
Death transition
Population
Larva
Juvenile
Organism
L_1 L_2
J_1 J_2 J_3
Internal species action mirrored by the Population- Prefix Population Transition rule Deletion

Figure 4.12: Diagrams of all transition examples in the simple PAL model.

Action Mirror rule

$$\frac{L_2 \xrightarrow{(LSwitch,[Switch\_on{:}{>}{>}\ (0,1)])} L_2'}{Larva\{\{L\}\}_A \xrightarrow{LSwitch} Larva\{\{L'\}\}_A} \text{ with } LSwitch \notin A$$

Population Synchronisation

$$\frac{Larva\{\{L\}\} \xrightarrow{LSwitch} Larva\{\{L'\}\} \quad Juvenile\{\{J\}\} \xrightarrow{LSwitch} Juvenile\{\{J'\}\}}{Larva\{\{L\}\} \underset{L_s}{\diamondsuit} Juvenile\{\{J\}\} \xrightarrow{LSwitch} Larva\{\{L'\}\} \underset{L_s}{\diamondsuit} Juvenile\{\{J'\}\}}$$
with $LSwitch \in L_s$

Deletion

$((LSwitch, 1) \downarrow Larva\{\{L\}\}) \xrightarrow{LSwitch} Larva\{\{L'\}\}$

$L_2 \in L \wedge L_2 \xrightarrow{LSwitch} L_2' \wedge L = [\![L_1, L_2]\!] \wedge |L| \geqslant 1 \wedge L' = L \backslash [\![L_2]\!]$

Addition

$((LSwitch, 1) \uparrow Juvenile\{\{J\}\}) \xrightarrow{LSwitch} Juvenile\{\{J'\}\}$

$J' = J \cup [\![initialJ]\!]$

Figure 4.13: Application of rules for life stage transition example.

and synchronises with an external Juvenile Population action which adds a new initialisation of a J Organism. This is shown schematically at the top of Figure 4.12 and Figure 4.13 shows the example applied to the rules.

4.3.2.2 *Reproduction*

Reproduction occurs in the model when a J Organism based on internal actions is ready to reproduce. An internal action occurs in the Organism layer and is mirrored by an external Juvenile Population action. The transition happens at the Population layer where the external Juvenile Population action synchronises with an external Larva Population action which adds a new initialisation of an L Organism to its multi-set. This is shown schematically in the middle of Figure 4.12 and Figure 4.14 shows the example applied to the rules.

Action Mirror rule

$$\frac{J_2 \xrightarrow{(\text{ReproduceL}, [\text{Tracker\_on:>> } (0,1)])} J_2'}{\text{Juvenile}\{\{J\}\}_A \xrightarrow{\text{ReproduceL}} \text{Juvenile}\{\{J'\}\}_A} \; \textit{with } \text{ReproduceL} \notin A$$

Population Synchronisation

$$\frac{\text{Larva}\{\{L\}\} \xrightarrow{\text{ReproduceL}} \text{Larva}\{\{L'\}\} \quad \text{Juvenile}\{\{J\}\} \xrightarrow{\text{ReproduceL}} \text{Juvenile}\{\{J'\}\}}{\text{Larva}\{\{L\}\} \underset{L_s}{\diamondsuit} \text{Juvenile}\{\{J\}\} \xrightarrow{\text{ReproduceL}} \text{Larva}\{\{L'\}\} \underset{L_s}{\diamondsuit} \text{Juvenile}\{\{J'\}\}}$$

with $\text{ReproduceL} \in L_s$

Activator

$$((\text{ReproduceL}, 1)((+))\text{Juvenile}\{\{J\}\}) \xrightarrow{\text{ReproduceL}} \text{Juvenile}\{\{J\}\}$$

$$J = [\![J_1, J_2, J_3]\!] \wedge |J| \geqslant 1$$

Addition

$$((\text{ReproduceL}, 1) \uparrow \text{Larva}\{\{L\}\}) \xrightarrow{\text{ReproduceL}} \text{Larva}\{\{L'\}\}$$

$$L' = L \cup [\![\text{initialL}]\!]$$

Figure 4.14: Application of rules for reproduction transition example.

Action Mirror rule

$$\frac{L_1 \xrightarrow{(LDie,[LDTracker\_on{:}{>}{>}\ (0,1)])} L_1'}{Larva\{\{L\}\}_A \xrightarrow{LDie} Larva\{\{L'\}\}_A}\ with\ LDie \notin A$$

Asynchronous Left

$$\frac{Larva\{\{L\}\} \xrightarrow{LDie} Larva\{\{L'\}\}}{Larva\{\{L\}\} \underset{L_s}{\Diamond} Juvenile\{\{J\}\} \xrightarrow{LDie} Larva\{\{L'\}\} \underset{L_s}{\Diamond} Juvenile\{\{J\}\}}\ \textit{with}\ LDie \notin L_s$$

Deletion

$$((LDie, 1) \downarrow Larva\{\{L\}\}) \xrightarrow{LDie} Larva\{\{L'\}\}$$

$$L_1 \in L \wedge L_1 \xrightarrow{LDie} L_1' \wedge L = [\![L_1, L_2]\!] \wedge |L| \geqslant 1 \wedge L' = L \backslash [\![L_1]\!]$$

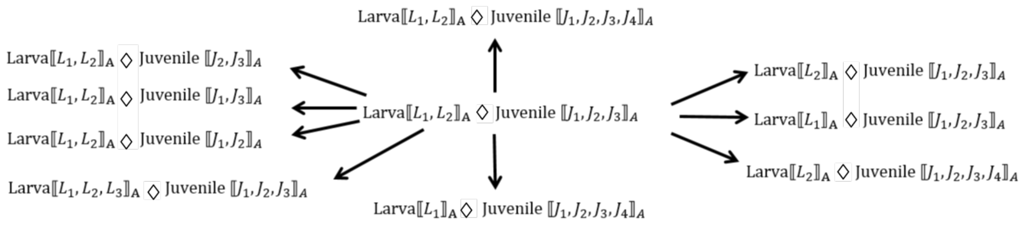
Figure 4.15: Application of rules for death transition example.

4.3.2.3 *Death*

Death occurs in the model based on a simple death rate. This rate could be changed to be based on when an Organism's internal tolerance to their external environment is outwith their thresholds. An internal action occurs in the Organism layer and is mirrored by an external Population action. The transition happens at the Population layer where the Population action deletes the specific organism from its multi-set. This is shown schematically at the bottom of Figure 4.12 and Figure 4.15 shows the example applied to the rules.

4.3.3 *Underlying CTMC of simple model*

A full example of the first transition is shown of the simple model in Figure 4.16, not all transitions of the model are shown. This is due to the fact that there are numerous transitions making a large CTMC. The model has seven labelled transitions from the initial state corresponding with the Population external actions: ReproduceL, LSwitch, LDie, JDie, RandomLDie, RandomJDie and RandomJAdd. The actions LSwitch will delete an L Organism from the Larva Population and add a J Organism to the Juvenile Population. This will leave one L Organism in the Larva Population and four J Organisms in the Juvenile Population. As this action deletes an Organism from a population it has to select a specific Organism that is ready to perform this action either L_1 or L_2. This action therefore can produce two states, one where L_1 remains in the Larva population and one where L_2 remains in the Larva population. This means that the model has a total of nine possible labelled transitions from the initial state. These nine labelled transitions are shown in Figure 4.16.



Larva$[\![L_1, L_2]\!]_A$ ◇ Juvenile $[\![J_1, J_2, J_3, J_4]\!]_A$
Larva$[\![L_1, L_2]\!]_A$ ◇ Juvenile $[\![J_2, J_3]\!]_A$
Larva$[\![L_1, L_2]\!]_A$ ◇ Juvenile $[\![J_1, J_3]\!]_A$
Larva$[\![L_1, L_2]\!]_A$ ◇ Juvenile $[\![J_1, J_2]\!]_A$
Larva$[\![L_1, L_2, L_3]\!]_A$ ◇ Juvenile $[\![J_1, J_2, J_3]\!]_A$
Larva$[\![L_1, L_2]\!]_A$ ◇ Juvenile $[\![J_1, J_2, J_3]\!]_A$
Larva$[\![L_1]\!]_A$ ◇ Juvenile $[\![J_1, J_2, J_3, J_4]\!]_A$
Larva$[\![L_2]\!]_A$ ◇ Juvenile $[\![J_1, J_2, J_3]\!]_A$
Larva$[\![L_1]\!]_A$ ◇ Juvenile $[\![J_1, J_2, J_3]\!]_A$
Larva$[\![L_2]\!]_A$ ◇ Juvenile $[\![J_1, J_2, J_3, J_4]\!]_A$

Figure 4.16: Initial model state and reachable states in first transition

4.4 COMPARISON OF PAL WITH OTHER PROCESS ALGEBRAS

A comparison of PAL is presented by constructing models in psPAH and PEPA Nets of the simple example in section 4.3.

4.4.1 *psPAH comparison with PAL*

Parametric Stochastic Process Algebra with Hooks (psPAH) [24] was chosen as a language to compare with PAL as it was designed to model biological systems at multiple scales. It uses composed hook actions and a vertical cooperation operator to allow events at higher scales to influence the behaviour of lower scales and vice versa. psPAH was previously explained in Chapter 1.

The simple model in Section 4.3 is defined in psPAH and is shown in Appendix B in five Figures. Figure B.4 shows the constants, functional rates and organism scale agent definitions. Figure B.5 shows the agent definitions for the organism physiological scale and Figures B.6, B.7 and B.8 show the initial state of the model. The agent definitions include the definitions of agents at the higher (organism scale) and the lower scale (organism physiological scale). In the lower scale the energy budget processes have three states; inactive, Larva and Juvenile. For example, for the energy reserve process, the states are NE(i), LE(i,w) and JE(i,w). The parameter i gives the unique identification of an organism and w gives the concentration level of the energy budget. The higher scale processes have three states which are NotInUse(i), Larva(i) and Juvenile(i).

The hook actions in this model are EnergyOn, EnergyOff, LDie, JDie, SwitchJEnergy, JEnergyOn, Switch and Reproduce. The lower scale triggers the Switch action of the higher scale when the Structural Volume (LV) of the Larva reaches a threshold concentration level. The Switch action causes the organism to change from a Larva to a Juvenile. The lower scale also triggers the Reproduce action of the higher scale when the Reproduction Buffer (JER) of the Juvenile reaches a threshold concentration level. The Reproduce action causes a Juvenile to synchronise locally with a NotInUse agent, this agent changes to a Larva and the Juvenile agent remains in the same state. The action that causes an organism to die from its internal physiology is triggered from the lower scale both for Larva (LDie) and Juvenile (JDie) agents.

The higher scale triggers the JEnergyOn action of the lower scale when a NotInUse agent's RandomJAdd action is fired by a specific rate. This action causes a Juvenile agent to be added to the population. The other three hook actions are triggered from the higher scale and cause changes in the lower scale. These changes ensure that the physiology of the organism is correct and aligned with the organism's state, for example, a Juvenile organism should have JE, JV and JER internal energy budgets. The lower scale processes synchronise on actions in their local scale, for example, a LV process has to synchronise with a LE process on the action Structural Volume growth (LG) which increases LV. The higher scale processes also synchronise locally with each other specifically on the reproduce action.

The psPAH model is considerably larger than the equivalent PAL model. This occurs because psPAH cannot handle specific multi-scale features that the PAL language possesses that allow features of the multi-scale system to be written easily into the model.

Issues constructing the example in psPAH are as follows: Firstly, psPAH cannot delete organisms from the model. NotInUse agents (ghost agents) were created to allow reproduction and death to occur in the model. When a Larva or Juvenile dies it becomes a NotInUse organism. Once in this state they will be used again when actions Reproduce and RandomJAdd are fired. The initialisation of NotInUse organisms make the model's initial state huge. In contrast PAL can delete specific organisms from the Population layer (either from a Population action or from internal mirrored actions for a specific organism) without the use of ghost agents.

Secondly, psPAH can produce new initialisations of organism processes. The NotInUse agents were created to be used to produce new initialisations of Larva. This is achieved when a NotInUse agent synchronises locally on the Reproduce action with a Juvenile agent. The NotInUse agent becomes a Larva and the Juvenile agent remains in the same state. The drawback of creating reproduction in the model is the creation of listing all possible options so the agents are able to synchronise correctly. To allow random Juveniles to be added to the population of Juveniles, NotInUse agents are used to add these agents. The novel language PAL can produce new initialisations of organisms from the Population layer based on individual organism thresholds and random additions based on specific rates without the need to create ghost agents such as the NotInUse processes.

In summary the simple model can be approximately defined in psPAH. The language has limitations of not allowing the deletion or addition of processes. These limitations causes the user to create states of processes that are not realistic states (NotInUse) but states that permit the inclusion of actions such as death and reproduction. Population actions such as random death of an organism and the random addition of a Juvenile to the Population had to be modelled using the non-realistic processes. PAL can handle all these features of multi-scale modelling without the inclusion of extra non-realistic states. To model a more complex multi-scale system psPAH would create an even larger model. Abstraction of details of the system would have to be carried out which means the model would not reflect the system accurately. This would further limit the descriptive nature of the model. psPAH does have an integrative modelling feature but having to explicitly define composed actions for each organism makes the model large and less readable. psPAH was designed specifically for modelling pattern formation and tissue growth. The language does not seem to have generic capabilities as shown by the creation of this simple model.

4.4.2 *PEPA Nets comparison with PAL*

PEPA Nets [30] was chosen as a language to compare with PAL as it is an example of a two level modelling language: the Petri Net is used to provide a structure for combining related PEPA systems. PEPA nets is previously explained in Chapter 1.

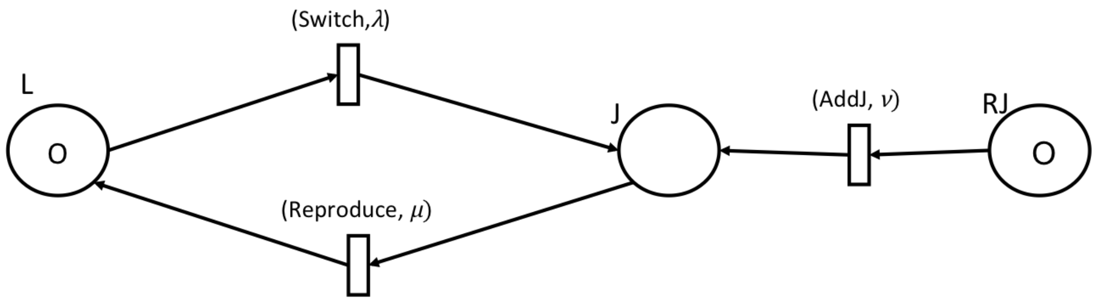


Figure 4.17: Simple schematic version of the PEPA Net.

The simple model in Section 4.3 is defined in a PEPA Net model and is shown in Appendix B and is split into six Figures. Figure B.9 shows the PEPA context definitions and the initial marking of the net, Figures B.10 and B.11 show the rates of actions and the PEPA definitions of the static and dynamic components. Figures B.12, B.13 and B.14 show the arcs of the net.

The places in the net are the life stages Larva (L) and Juvenile (J). The other place in the net named Random Juvenile (RJ) is a mechanism to add new Juvenile organisms at random to the population. These places are replicated to accommodate a maximum number of organisms (in this example, ten of each and five RJ places). The PEPA Net token is a generic organism which performs the activities Switch, Reproduce and AddJ which cause firing of the net. The names of the activities that cause a firing of the net are printed in bold in Figure B.11. In Figure 4.17 a simple schematic version of the PEPA Net of the system is shown with only one Larva, one Juvenile, one Random Juvenile place and two organisms present. The net described in Figures B.12, B.13 and B.14 have 250 arcs with each L place having 20 arcs (10 arcs arriving and 10 leaving), each J place having 25 arcs (15 arcs arriving and 10 leaving) and each RJ place having 5 leaving arcs. Each L and J place has many local states because of the activities of the static components. The static components are the internal components of the organism (energy budgets) specifically structural volume (V), energy reserve (E) and reproduction buffer (ER). The static components change dependent on each other's activities.

Issues constructing the example in PEPA-nets are as follows: Firstly, the PEPA nets language is non parametric, therefore, internal components of the organism do not have numerical information associated with them. Numerical information is important in the setting of organism thresholds, for example, when an organism is ready to reproduce. To solve this issue abstraction of some detail was necessary. Internal components of the organism were given specific states of concentration levels, for example, zero, low, medium and high. This abstraction limits the system being described accurately. The novel language PAL allows the user to have numerical information of internal Organism components and their rates can be influenced by each other.

Secondly, activities had to be included to turn off the static components when an organism is not present in a place. When an organism arrives in a place the static components are turned on and reset to their initial state. If the static components were left on they would change their state independently without a present organism and when an organism arrives in the place the static components would not be set to their initial states. In contrast, in PAL internal agents are always associated with an Organism. Organism's internal agents in PAL are initialised when an Organism is added to a Population and are deleted when an Organism is deleted from a Population. PEPA nets, therefore, does not allow the interactions between scales to be modelled easily.

Thirdly, PEPA nets cannot produce new initialisations of organism components. To solve this issue, organisms in a Juvenile place in the net can move back into a Larva place by the reproduce activity. This does not increase the population, therefore, the model will always have the same number of organisms. There is another solution to this issue by constructing another place in the model that holds organisms and fires them at a certain rate, but this has the problem of not associating the production of larvae with the behaviour of a juvenile. This solution is used to model the random addition of Juvenile organisms to the population. This is a population action and therefore does not need to be associated with the specific behaviour of an organism. The novel language PAL can produce new initialisations of Organisms based on individual thresholds (through an Organism's internal action) or Population action rates (through a Population action). This shows once more that PEPA nets cannot model interactions between scales efficiently.

Fourthly, PEPA nets cannot delete organisms from the model. A solution was found by creating a death state for organisms and have them carry out a death activity at a certain rate. There are two death activities, one associated with an organism's internal components' death rate and one associated with population death rate. In the death state the organism cannot change state or interact with static components. This is not ideal as the place where the organism dies therefore is occupied by the dead organism. In contrast PAL can delete specific Organisms from the Population layer.

In summary the simple model cannot easily be defined in a PEPA net. The language restricts the user in modelling some multi-scale features such as including numeric details of internal components of an organism and the reproduction and death of an organism. PAL can handle all these features of multi-scale modelling.

4.5 SUMMARY

This chapter defined PAL, a multi-scale process algebra designed to model multi-scale systems. The novel features of PAL are the layers of the language: Population and Organism. These layers allow the user to elegantly describe the integration and interactions between scales in one PAL model. PAL is shown to adhere to the specific multi-scale modelling features: abstraction, descriptive, integrative, explanatory and generic. It follows the middle-out strategy. PAL addresses all the features and challenges from Chapter 3. The layers of the language allow the integration and interaction of multi-scales in a system to be modelled easily. The modularity of the language allows the the appropriate unit scales to be used for specific life stages. A population view of a system can be modelled and analysed.

The languages of psPAH and PEPA nets restricts the modelling of some multi-scale system features. PAL allows the user to easily define the addition and deletion of organisms within its language, whereas psPAH and PEPA nets do not. PAL also modularises the specific organism populations from the organisms internal model giving a more elegant model whereas the others create large models which make them more difficult to read.

PAL is applicable to models of a variety of marine organisms, including oysters; however, it is designed to be generic, so it can be applied to other systems in which modelling at multi-scales is important. The explanatory and generic features of PAL are explored in the next chapters.

5 APPLICATION OF PAL TO THE PACIFIC OYSTER LIFE STAGE CASE STUDY

In this chapter PAL firstly is applied to the Pacific oyster life stage case study which was investigated in the integrated life stage Bio-PEPA model in Section 3.2 of Chapter 3. PAL allows the integration of the three main life stages of the Pacific oyster. PAL enables the analysis of a specific oyster within its life stage. To validate the PAL model the results are compared to the Bio-PEPA integrated life stage model results. Secondly the motivation of this PAL model case study is to incorporate the effects of ocean acidification on the Larva life stage of an oyster and its effect to the overall population. PAL enables the analysis of the population view. The PAL model allows multi-scale analysis of larvae length and mortality and oyster population growth. Results are compared with wet laboratory ocean acidification experiments by Barros et al [14] and Timmins-Schiffman et al [72]. The application of PAL to this case study shows PAL addresses the multi-scale challenges and features discussed in Chapter 3.

5.1 INTRODUCTION TO CASE STUDY

The specific motivation in this case study is ocean acidification's affect on the Larva life stage of the Pacific oyster and how this changes population growth. This is a multi-scale system as ocean acidification affects the energy budgets within each Larva and this causes the population of oysters to change. Barros et al [14] and Timmins-Schiffman et al [72] states that ocean acidification causes a reduction in Larva growth rate and a higher mortality rate. They indicate that there is a small reduction in the growth rate and a large increase in mortality when larvae are exposed to high and low levels of CO_2. As previously discussed in Chapter 2 the Pacific oyster is potentially the largest harvested and collected shellfish in European waters. Gazeau et al [29] argues that the predicted effects of ocean acidification will most certainly cause a significant economic loss.

There are existing DEB models of the Pacific oyster which were discussed in Chapters 2 and 3 that model up to two life stages of an oyster's physiology. These models are in one scale and do not include the effects of ocean acidification. The novel PAL model is multi-scale as it models three different life stage populations of Pacific oysters and includes a basic feature of ocean acidification that affects only the larvae population. The ocean acidification feature is based on Barros et al [14] and Timmins-Schiffman et al [72] findings. Bio-PEPA DEB models from Chapters 2 and 3 are utilised to create the physiological scale of the specific life stage of the oysters in the PAL model.

5.2 PAL MODEL

The PAL model is given in Figures 5.1, 5.2, 5.3 and 5.4 and the parameter values are shown in Table 5.1. In the model the Population actions are highlighted in bold. The PAL model is made up of two layers: an oyster population and an oyster physiology. The model has three types of population: Larvae, Juveniles and Adults. Each population contain specific oysters (Larva, Juvenile or Adult) with specific internal species making up a dynamic energy budget model for the oyster's physiology. A Larva switches its life stage to become a Juvenile when it reaches the threshold length of $300\mu m$, which triggers the **LSwitch** Population action. A Juvenile switches its life stage to become an Adult when its structural volume is greater than the threshold of $0.4cm^3$ triggering the Population action **JSwitch**. An Adult can reproduce a Larva when two thresholds are reached (when temperature is greater than $20^\circ C$ and the energy allocated to reproduction is 35% of the total dry flesh weight) triggering **reproduceL** Population action. To ensure simplicity the Adult only produces one Larva. The internal species values of a PAL model have to be an integer, therefore, the initial values for a Larva are scaled up by 10^6, a Juvenile up by 10^3 and no scaling for an Adult.

The ocean acidification feature is created using a simple ocean acidification parameter that ranges from zero to six. The control simulation experiment is at zero. Low CO_2 is in the range of 1 to 4 and high CO_2 is in the range of 3 to 6. The energy allocated to Larva growth is slightly reduced to 0.39 if the ocean acidification parameter is in the range 1 to 3. Death occurs to a Larva when the ocean acidification parameter is greater than 3. This triggers the **LDeath** Population action of a Larva.

The food and temperature values in the PAL model were set to be the same as the first experiment (Larvae July) of the integrated Bio-PEPA model in Chapter 3 where the Larvae life stage commenced in July showing an early spawning. The temperature for the months of July to September was set to the optimal value of $25^\circ C$ [67], the remaining months had the temperature of $15^\circ C$. This temperature is too low for spawning to occur but high enough for the lower temperature tolerance range for all life stages [59, 67]. The food density was set at a constant optimal value of $1400\mu m^3 \mu l^{-1}$ [67]. A time series analysis is carried out on the PAL model which had the initial population of 1 Larva, zero Juveniles and zero Adults. The dry flesh weight (DFW) results from the simulation are compared to the results of the integrated life stage Bio-PEPA model in Section 3.2 of Chapter 3 in Figure 5.5. This validates that the PAL model is parametrised correctly for each life stage according to the Bio-PEPA DEB models from Chapters 2 and 3.

Parameter	Value	Parameter	Value
DeathT	3	Ts	20
K	0.45	LEG	$1900 * 10^{-12}$
UE	17500	JAEG	1900
KR	0.7	Ldm	0.658
p	0.2	Jdm	0.175
TAL	75000	GSI	35
LT1	298	Vp	0.4
LTA	11000	Lae	0.4
LTAH	75000	JAae	0.75
LTH	306	Lux	4.5
LTL	285	LEM	$2295 * 10^{-12}$
JAT1	293	JAEM	2295
JATA	5800	LFood_ density	1400
JATAH	30000	Lsaturation_ coefficient	600
JATH	305	JAFood_ density	$1400 * 10^{-12}$
JATL	281	JAsaturation_ coefficient	$600 * 10^{-12}$
OceanAcidification	0 to 6	metamorphosis	300

Table 5.1: Parameter values of Pacific oyster PAL model.

General parameters

$Lev = LV/1000000; \quad Le = LE/1000000; \quad Ler = LER/1000000;$

$Jev = JV/1000; \quad Je = JE/1000; \quad Jer = JER/1000;$

$OADeath = H(OceanAcidification - DeathT);$

$RGrowth = H(OceanAcidification);$

$Actual\_temperature = dependent\_on\_experiment;$

$T = 273 + Actual\_temperature;$

$$LTemperature\_adjustment = exp((LTA/LT1) - (LTA/T)) * ((1 + exp((TAL/T) - (TAL/LTL)) \\ + exp((LTAH/LTH) - (LTAH/T)))^{-1});$$

$$JATemperature\_adjustment = exp((JATA/JAT1) - (JATA/T)) * ((1 + exp((TAL/T) - (TAL/JATL)) \\ + exp((JATAH/JATH) - (JATAH/T)))^{-1});$$

$LLength = (((Lev/LEG)^{1/3})/Ldm);$

$LVum = ((Ldm * LLength)^{3});$

$JLength = ((Jev/JAEG)^{1/3})/Jdm;$

$JVcm = ((Jdm * JLength)/Jdm))^{3});$

$ALength = (((AV/JAEG)^{1/3})/Jdm);$

$AVcm = ((Jdm * ALength)/Jdm))^{3});$

$LE\_DFW = (Le/UE);$

$LER\_DFW = ((KR * Ler)/UE);$

$LV\_DFW = ((LVum * 10^{-12}) * p);$

$LTotal\_DFW = LE\_DFW + LV\_DFW + LER\_DFW;$

$JE\_DFW = (Je/UE);$

$JER\_DFW = ((KR * Jer)/UE);$

$JV\_DFW = (JVcm * p);$

$JTotal\_DFW = JE\_DFW + JV\_DFW + JER\_DFW;$

$AE\_DFW = (AE/UE);$

$AER\_DFW = ((KR * AER)/UE);$

$AV\_DFW = (AVcm * p);$

$JTotal\_DFW = AE\_DFW + AV\_DFW + AER\_DFW;$

$Percentage\_ER = ((AER\_DFW/ATOTAL\_DFW) * 100);$

Figure 5.1: Pacific Oyster PAL Model Part 1 of 4.

General parameters

$LPXm = (137 * LTemperature\_adjustment);$

$JAPXm = (560 * JATemperature\_adjustment);$

$LPAm = ((Lae * Lux * LPXm)/10) * 10^{-8});$

$JAPAm = JAea * JAPXm;$

$LVol\_costs\_maintenance\_Pm = ((24 * 10^{-12}) * LTemperature\_adjustment);$

$JAVol\_costs\_maintenance\_Pm = (24 * JATemperature\_adjustment);$

$LPm = LVol\_costs\_maintenance\_Pm * LVum;$

$JPm = JAVol\_costs\_maintenance\_Pm * JVcm;$

$APm = JAVol\_costs\_maintenance\_Pm * AVcm;$

$LEnergy\_Density\_in\_Organism = Le/LVum;$

$JEnergy\_Density\_in\_Organism = Je/JVcm;$

$AEnergy\_Density\_in\_Organism = AE/AVcm;$

$LPc = (LEnergy\_in\_Organism/(LEG + (K * LEnergy\_Density\_in\_Organism)))$
$\quad *(((LEG * LPAm * (LVum)^{2/3})/LEM) + (LVol\_costs\_maintenance\_Pm * LVum));$

$JPc = (JEnergy\_in\_Organism/(JAEG + (K * JEnergy\_Density\_in\_Organism)))$
$\quad *(((JAEG * JAPAm * (JVcm)^{2/3})/JAEM) + (JAVol\_costs\_maintenance\_Pm * JVcm));$

$APc = (AEnergy\_in\_Organism/(JAEG + (K * AEnergy\_Density\_in\_Organism)))$
$\quad *(((JAEG * JAPAm * (AVcm)^{2/3})/JAEM) + (JAVol\_costs\_maintenance\_Pm * AVcm));$

$Lfunctional\_response = ((LFood\_density)^2)/(((LFood\_densisty)^2) + ((Lsaturation\_coefficient)^2));$

$JAfunctional\_response = JAFood\_density/(JAFood\_densisty + JAsaturation\_coefficient);$

$LPa = Lfunctional\_response * LPAm * ((LVum)^{2/3});$

$JPa = JAfunctional\_response * JAPAm * ((JVcm)^{2/3});$

$APa = JAfunctional\_response * JAPAm * ((AVcm)^{2/3});$

$JtoA = H(JVcm - Vp);$

$ER\_start\_spawn = H(Percentage\_ER - GSI);$

$stop\_spawn = H(1 - Percentage\_ER);$

$T\_start\_spawn = H(Actual\_temperature - Ts);$

$LPj = (((1 - K)/K) * LVum * LVol\_costs\_maintenance\_Pm);$

$JPj = (((1 - K)/K) * JVcm * JAVol\_costs\_maintenance\_Pm);$

$APj = (((1 - K)/K) * Vp * JAVol\_costs\_maintenance\_Pm);$

$Lstop\_action = H(1 - LER);$

$Jstop\_action = H(1 - JER);$

$Astop\_action = H(1 - AER);$

$LtoJ = H(LLength - metamorphosis);$

Figure 5.2: Pacific Oyster PAL Model Part 2 of 4.

internal species actions

$LG : ((K - (0.06 * RGrowth)) * LPc) * (1000000);$

$JG : (K * JPc) * (1000);$

$AG : (K * APc);$

$LS : LPm * (1000000);$

$JS : JPm * (1000);$

$AS : APm;$

$LA : LPa * (1000000);$

$JA : JPa * (1000);$

$AA : APa;$

$LC : LPc * (1000000);$

$JC : JPc * (1000);$

$AC : APc;$

$LR : ((1 - K) * LPc) * (1000000);$

$JR : ((1 - K) * JPc) * (1000) * JtoA;$

$AR : ((1 - K) * APc);$

$LJ : (LPj * (1000000)) * (1 - Lstop\_action);$

$JJ : (JPj * (1000)) * (1 - Jstop\_action) * JtoA;$

$AJ : (APj) * (1 - Astop\_action);$

LSwitch $: fMA(100 * (LtoJ));$

JSwitch $: fMA(100 * (JtoA));$

$empty : fMA(100);$

reproduceL $: fMA(10 * ER\_start\_spawn * T\_start\_spawn);$

$stopreproduceL : fMA(10 * stop\_spawn);$

LDeath $: fMA(100 * OADeath);$

Figure 5.3: Pacific Oyster PAL Model Part 3 of 4.

Internal species

LE = LA >> +LC << +LG(.) + LR(.);

JE = JA >> +JC << +JG(.) + JR(.);

AE = AA >> +AC << +AG(.) + AR(.);

LV = LG >> +LS << +LA(.) + LC(.) + LR(.) + LJ(.);

JV = JG >> +JS << +JA(.) + JC(.) + JR(.) + JJ(.);

AV = AG >> +AS << +AA(.) + AC(.) + AR(.) + AJ(.) + empty(.);

LER = LR >> +LJ <<;

JER = JR >> +JJ <<;

AER = AR >> +AJ << +(empty, 1) <<;

LSwitch_on = (**LSwitch**, 1) >>;

LSwitch_off = (**LSwitch**, 1) <<;

JSwitch_on = (**JSwitch**, 1) >>;

JSwitch_off = (**JSwitch**, 1) <<;

ATracker_off = (**reproduceL**, 1) << +(stopreproduceL, 1) >>;

ATracker_on = (**reproduceL**, 1) >> +(stopreproduceL, 1) << +(empty, 1)(+);

Set of hidden internal actions

HiddenActionsA : {LG, LS, LA, LC, LR, LJ, JG, JS, JA, JC, JR, JJ, AG, AS, AA, AC, AR, AJ, empty, stopreproduceL};

Oysters (Internal species model)

Larva = LE[250] $\bowtie_{*}$ LV[250] $\bowtie_{*}$ LER[0] $\bowtie_{*}$ LSwitch_on[0] $\bowtie_{*}$ LSwitch_off[1]

Juvenile = JE[14] $\bowtie_{*}$ JV[14] $\bowtie_{*}$ JER[16] $\bowtie_{*}$ JSwitch_on[0] $\bowtie_{*}$ JSwitch_off[1]

Adult = AE[614] $\bowtie_{*}$ AV[726] $\bowtie_{*}$ AER[0] $\bowtie_{*}$ ATracker_on[0] $\bowtie_{*}$ ATracker_off[1]

Oyster Populations (External actions)

Larvae{{Larva}} = **LSwitch** $\downarrow$ +**reproduceL** $\uparrow$ +**LDeath** $\downarrow$;

Juveniles{{Juvenile}} = **LSwitch** $\uparrow$ +**JSwitch** $\downarrow$;

Adults{{Adult}} = **JSwitch** $\uparrow$ +**reproduceL**((+));

Model Component

$\mathit{Larvae}[\![\mathit{Larva\_1}, ..., \mathit{Larva\_5}]\!]_A \underset{\{LSwitch\}}{\Diamond} \mathit{Juveniles}[\![\ ]\!]_A \underset{\{Jswitch, reproduceL\}}{\Diamond} \mathit{Adults}[\![\ ]\!]_A$

Figure 5.4: Pacific Oyster PAL Model Part 4 of 4.

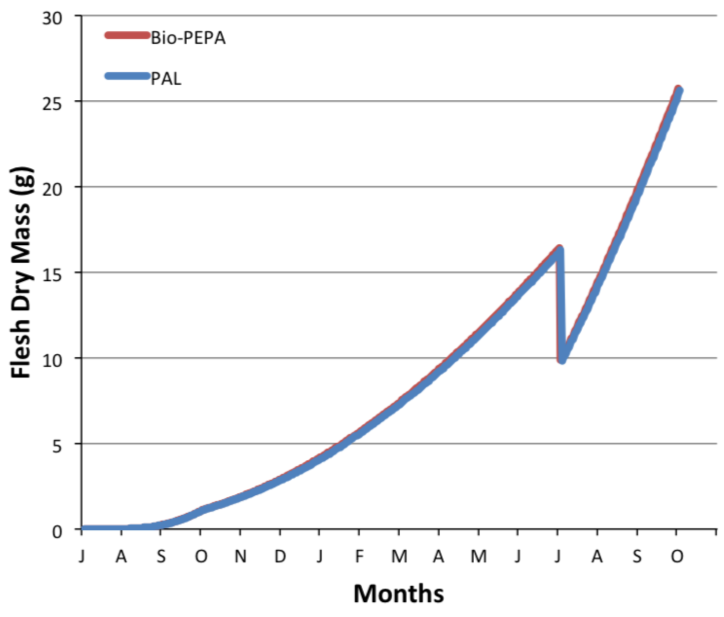


Figure 5.5: Comparison of PAL model results to integrated Bio-PEPA model from Chapter 3.

5.3 MODEL ANALYSIS

The PAL parser was used to translate the PAL model to a Bio-PEPA model to allow analysis of the model in the Bio-PEPA plug-in. Discrete stochastic time series analysis simulations were carried out. Continuous ODEs simulation time-series analysis could not be carried out as the ODEs solver in the Bio-PEPA plug-in did not handle the translated PAL model. In particular the solver cannot handle the 'on' and 'off' agents used to describe Organisms in the Bio-PEPA model. The solver in this case allowed the organisms to remain 'on' even after a removal action such as death. They remained in the population whereas they should change to an 'off' agent. This issue does not occur in the discrete stochastic time series analysis simulations.

Three experiments were carried out: control, low CO_2 and high CO_2. The initial population values were 10 Larvae, zero Juveniles and zero Adults. The initial population is relatively low due to the capabilities of the Bio-PEPA plug-in. The ocean acidification parameter of each oyster was assigned a value within a certain range dependent on the experiment. The ocean acidification parameter is set to zero in the control. Low CO_2 is in the range of 1 to 4 and high CO_2 is in the range of 3 to 6. The total time for each experiment is 450 days. For simplicity a month in all experiments had 30 days making a year 360 days. The start month is July. Each experiment is one stochastic simulation.

5.3.1 *Larval Length*

The analysis of the results of the effect of ocean acidification on Larval growth shows that there is a small decrease in length. The average length of Larvae from day 2 to 6 from each experiment is given in Figure 5.6. The results show that at day 6 the control Larvae are greater in length by 7μm than the low CO_2 larvae and 8μm than the high CO_2. This is comparable with the Larval length results from Barros et al[14] and Timmins-Schiffman et al [72] where the control length is greater by 10μm. The results indicate that, as a result of ocean acidification, Larvae take slightly longer to reach metamorphosis (Larva to Juvenile). The control reached metamorphosis at day 15 whereas the other experiments it occurred in day 17. The decreased growth rate also showed similar effects to the Larvae DFW results.

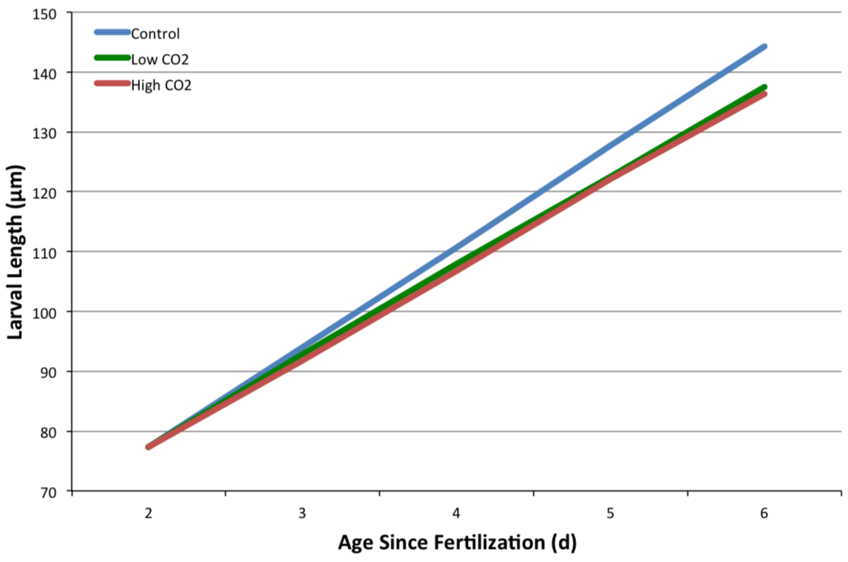


Figure 5.6: Time Series Analysis of Larval length from day 2 to 6.

5.3.2 *Population growth*

The analysis of the results of the effect of ocean acidification on oysters population growth shows that Larvae death decreases total population numbers. It also shows that the decrease in Larval growth does not impact on the later life stage events e.g. spawning. The total population of oyster numbers for all experiments throughout the time frame are given in Figure 5.7. There is a 40% mortality from the control to the low CO_2 and a 60% mortality for the high CO_2 in the Larva life stage at the beginning of the experiments. There is a 40% mortality from the control to the low CO_2 and a 65% mortality for the high CO_2 in the overall population. These figures are similar to the results from Barros et al[14] and Timmins-Schiffman et al [72] although their initial Larvae population numbers are higher.

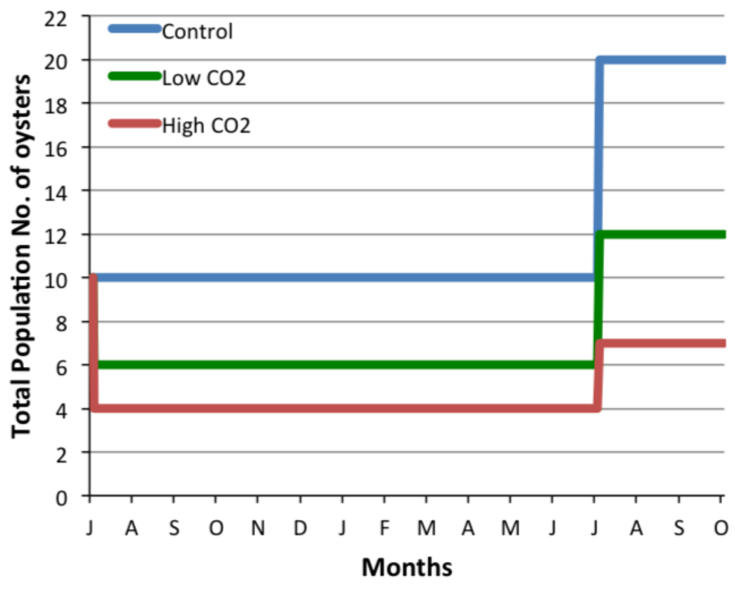


Figure 5.7: Time Series Analysis of oyster Population Growth.

5.4 SUMMARY

In this chapter PAL has been successfully applied to a Pacific oyster life stage case study. The PAL model was firstly validated, its results are comparable with the results of the integrated life stage Bio-PEPA model in Section 3.2 of Chapter 3. Secondly, analysis of the model focused on different scales of the case study from the Larva life stage to the overall population specifically on the effect of ocean acidification. It is noted that the ocean acidification feature in the PAL model is simplistic. It only affects the Larva life stage whereas in reality it may have an effect on the other life stages. The ocean acidification literature on Pacific oysters is mainly focused on the early Larval life stage development. If research becomes available for later life stages this could be easily added to the PAL model. A combination of different food densities, temperature and ocean acidification values could be investigated in the PAL model to measure the combined impact these have on all life stages. The PAL model gives the potential for users to focus not just on one life stage but a whole life cycle and also the population view of the system.

It is worth noting all the features and challenges from Chapter 3 are overcome in this PAL model. The layers of the language allow the integration and interaction of multi-scales of the system in one PAL model. The modularity of the language allows the the appropriate unit scales to be used in specific life stages. The population view of the system is modelled and analysis is shown.

6 APPLICATION OF PAL TO CELL CYCLE AND DNA DAMAGE CASE STUDY

PAL is a generic language therefore it can be applied to a variety of multi-scale systems. To demonstrate this, in this chapter PAL is applied to a mammalian cell cycle and DNA damage case study. The motivation of this case study is to analyse the affects of damage from cancer treatments to the length of a cell cycle and cell survival. This is a multi-scale system as the damage affects the levels of intracellular proteins within a cell and therefore affects the cell population levels. A PAL model has been created by linking together a cell cycle model with a repair model with an external force applying damage. The PAL model allowed multi-scale analysis of an average length of a single cell cycle and cell population growth. Results from the PAL model are compared with wet laboratory data that show the survival fraction of cells receiving different radiation doses [52].

6.1 INTRODUCTION TO CASE STUDY

The motivation of this case study is to investigate the effects of cancer treatment doses on the length of a cell cycle and cell population growth. The availability of wet laboratory data allows comparison with results from the PAL model [52]. PAL allows the multi-scale investigation of the system, whereas previous existing models focus on one scale only [73, 77]. These models are utilised in the PAL model to represent a specific scale. There are multi-scale cellular automaton models that simulate spatial temporal growth [61, 60]. These models use different modelling approaches for different scales of the system e.g. the intracellular proteins are defined in ODEs and the cell populations are modelled in cellular automaton. PAL allows all scales to be modelled in one language. The PAL model shows the potential of multi-scale modelling in process algebra albeit it does not represent explicit spatial awareness. Organisms in PAL currently do not have the ability to interact explicitly with one another.

To model the cell dynamics within each cell, the intracellular layer, a basic model originally developed by Tyson et al [73] consisting of a system of six ODEs of cell cycle proteins is translated to internal species of the PAL model. This basic model includes only the interactions which are considered to be essential for cell cycle regulation and control. The model is reproduced for convenience in Equations (6.1) to (6.6) [73]. The model describes the transitions between two main steady states, growing (G1) and dividing (S-G2-M), of the cell cycle which is controlled by the relationships between the proteins. These proteins include the Cdk-cyclin

B complex (CycB), the APC-Cdh1 complex (Cdh1), the active form of Cdc20 ($Cdc20_A$), the total Cdc20 ($Cdc20_T$), the intermediary enzyme (IEP) and the mass of the cell (m). The states of the cell are controlled by changes in cell mass and threshold values of the CycB. To make the cell cycle relevant to mammalian cells the parameter values of this model are taken from Powathil et al [61], therefore, time in the model is in hours.

$$\frac{d[CycB]}{dt} = k_1 - (k'_2 + k"_2[Cdh1])[CycB] \tag{6.1}$$

$$\frac{d[Cdh1]}{dt} = \frac{(k'_3 + k"_3[Cdc20_A])(1-[Cdh1])}{J_3 + 1 - [Cdh1]} - \frac{k_4 m[CycB][Cdh1]}{J_4 + [Cdh1]} \tag{6.2}$$

$$\frac{d[Cdc20_T]}{dt} = k'_5 + k"_5 \frac{(\frac{[CycB]m}{J_5})^n}{1 + (\frac{[CycB]m}{J_5})^n} - k_6[Cdc20_T] \tag{6.3}$$

$$\begin{aligned}\frac{d[Cdc20_A]}{dt} = &\frac{k_7[IEP](Cdc20_T] - [Cdc20_A])}{J_7 + [Cdc20_T] - [Cdc20_A]} \\ &- \frac{k_8[Mad][Cdc20_A]}{J_8 + [Cdc20_A]} - k_6[Cdc20_A]\end{aligned} \tag{6.4}$$

$$\frac{d[IEP]}{dt} = k_9 m[CycB](1 - [IEP]) - k_{10}[IEP] \tag{6.5}$$

$$\frac{dm}{dt} = \mu m(1 - \frac{m}{m_*}) \tag{6.6}$$

To model how the cell cycle intracellular proteins are effected by DNA damage a model originally developed by Zhang et al [77] consisting of four ODEs is translated to internal species and parameters of the PAL model. The model consists of DNA damage, the transcription factor p53 and Mdm2 (nucleus and cytoplasmic) that promote the degradation of p53. The model is reproduced for convenience in Equations (6.7) to (6.10) [77]. The DNA damage ODE has an effect on these proteins to cause the levels of p53 to pulse. A simple assumption is made that damage is repaired at a constant rate independent of the proteins. p53 inhibits the activity of CycB preventing the progression of the cell cycle. In the PAL model the Tyson et al [73] and Zang et al [77] models are linked together by changing the CycB degradation rate to be influenced by changes in the p53 levels.

$$\frac{d[p53]}{dt} = k'_{s53} + k"_{s53}\frac{[Mdm2_{cyt}]^4}{J_{s53}^4 + [Mdm2_{cyt}]^4} - k_{d53}[p53] \tag{6.7}$$

$$\begin{aligned}\frac{d[Mdm2_{cyt}]}{dt} = &k'_{s2} + k"_{s2}\frac{[p53]^4}{J_{s2}^4 + [p53]^4} \\ &- k_i[Mdm2_{cyt}] + k_o[Mdm2_{nuc}] - k'_{d2}[Mdm2_{cyt}]\end{aligned} \tag{6.8}$$

$$\begin{aligned} d[\mathrm{Mdm2}_{nuc}] = k_i[\mathrm{Mdm2}_{cyt}] - k_o[\mathrm{Mdm2}_{nuc}] \\ - k'_{d2}(1 + \mathrm{DNAdamage})[\mathrm{Mdm2}_{nuc}] \end{aligned} \tag{6.9}$$

$$\frac{d\mathrm{DNAdamage}}{dt} = -k_{repair}H(\mathrm{DNAdamage}) \tag{6.10}$$

6.2 PAL MODEL

The PAL model is given in Figures 6.1, 6.2 and 6.3 and the parameter values are shown in Table 6.1. In the model the Population actions are highlighted in bold. The case study has two distinct layers; cell population and intracellular. The cell population layer is described in the PAL model by defining two PAL Populations based on the two steady states of a cell; Growing and Dividing. These Populations contain G and D Organism components. A G Organism is a cell in a growing state (G cell) and a D Organism is a cell in a dividing state (D cell). G and D cells contain internal species which create the intracellular layer which are the system of proteins translated from Tyson et al [73] and Zhang et al [77]. G and D cells have different initial values of intracellular proteins. The internal species value of a PAL model have to be integer, therefore, the initial internal species values are scaled up by one hundred. A G cell has a low CycB and high Cdh1 initial value. When the levels of CycB are above ten the Population action **start** triggers a G cell to become a D cell. A D cell has a standard CycB initial value just above ten and a low standard Cdh1 initial value. When the levels of CycB are lower than ten the Population action **finish** triggers the D cell to reproduce a G cell and become a G cell, therefore, producing two new G cells. The switching from G cell to D cell to G cell completes one cell cycle.

Both G and D cells are indirectly affected by damage through p53. Levels of p53 are at equilibrium (steady state) when there is no damage in the system. When there is damage it causes p53 levels to pulse. The internal species CycB degradation rate is multiplied by the level of p53 only when p53 is above its equilibrium threshold of 0.19. Damage is a parameter of the model and a simple assumption is made that damage is repaired at a constant rate.

Damage levels tested in the model range from integer values of zero to ten. The assumption has been made that a damage greater than four causes cell death through the Population actions **GDeath** and **DDeath**. Cell cycle length will still be impacted with a damage less than five through the increase of CycB degradation rate.

To analyse the effects of damage on population growth different damage levels were assigned to each cell randomly depending on the highest level of damage at the start of the simulation. For example, if the highest damage in a simulation is five the cells are assigned damages in the range of zero to five. This randomness of damage assignment is to replicate

General parameters

$\mathrm{Gmscaled} = \mathrm{Gm1}/100; \quad \mathrm{GCdh1scaled} = \mathrm{GCdh1}/100; \quad \mathrm{GCycBscaled} = \mathrm{GCycB}/100;$

$\mathrm{GCdc20Tscaled} = \mathrm{GCdc20T}/100; \quad \mathrm{GCdc20Ascaled} = \mathrm{GCdc20A}/100; \quad \mathrm{GIEPscaled} = \mathrm{GIEP}/100;$

$\mathrm{Dmscaled} = \mathrm{Dm1}/100; \quad \mathrm{DCdh1scaled} = \mathrm{DCdh1}/100; \quad \mathrm{DCycBscaled} = \mathrm{DCycB}/100;$

$\mathrm{DCdc20Tscaled} = \mathrm{DCdc20T}/100; \quad \mathrm{DCdc20Ascaled} = \mathrm{DCdc20A}/100; \quad \mathrm{DIEPscaled} = \mathrm{DIEP}/100;$

$\mathrm{p53scaled} = \mathrm{p53}/100; \quad \mathrm{Mdm2cscaled} = \mathrm{Mdm2c}/100; \quad \mathrm{Mdm2nscaled} = \mathrm{Mdm2n}/100;$

$\mathrm{GtoD} = H(\mathrm{GCycB} - \mathrm{CycBT});$

$\mathrm{divide1} = H(\mathrm{CycBTdivide1} - \mathrm{DCycB});$

$\mathrm{divide2} = H(\mathrm{CycBTdivide2} - \mathrm{DCycB});$

$G = ((2 * \mathrm{Mdm2nscaled} * (\mathrm{j2}/\mathrm{p53scaled}))/$

$\quad(\mathrm{theta} - \mathrm{Mdm2nscaled} + (\mathrm{theta} * (\mathrm{j1}/\mathrm{p53scaled})) + (\mathrm{Mdm2nscaled} * (\mathrm{j2}/\mathrm{p53scaled}))$

$\quad+((\mathrm{theta} - \mathrm{Mdm2nscaled} + (\mathrm{theta} * (\mathrm{j1}/\mathrm{p53scaled})) + (\mathrm{Mdm2nscaled} * (\mathrm{j2}/\mathrm{p53scaled})))^{(2)}$

$\quad-4 * (\mathrm{Mdm2nscaled} * (\mathrm{j2}/\mathrm{p53scaled})) * (\mathrm{theta} - \mathrm{Mdm2nscaled}))^{(0.5)}));$

$\mathrm{kd53} = 0.27 + 8.25 * G;$

$D = ((\mathrm{DAM} - ((\mathrm{time}) * \mathrm{rep})) * (H(\mathrm{time}) - H(\mathrm{time} - (\mathrm{DAM}/\mathrm{rep}))));$

$\mathrm{Death} = H(D - \mathrm{deathT});$

$\mathrm{kd} = 0.05 * (1 - D);$

Internal species actions

$\mathrm{Gsythesis} : \mathrm{k1} * 100;$

$\mathrm{Gdegradation} : ((\mathrm{k2} * (\mathrm{p53scaled}/0.19) + \mathrm{k21} * \mathrm{GCdh1scaled}) * \mathrm{GCycBscaled}) * 100;$

$\mathrm{Gactivation} : (((\mathrm{k3} + \mathrm{k31} * \mathrm{GCdc20Ascaled}) * (1 - \mathrm{GCdh1scaled}))/(\mathrm{j3} + 1 - \mathrm{GCdh1scaled})) * 100;$

$\mathrm{Ginactivation} : ((\mathrm{k4} * \mathrm{Gmscaled} * \mathrm{GCycBscaled} * \mathrm{GCdh1scaled})/(\mathrm{j4} + \mathrm{GCdh1scaled})) * 100;$

$\mathrm{Gcellgrowth} : ((u * \mathrm{Gmscaled}) * (1 - (\mathrm{Gmscaled}/\mathrm{m2}))) * 100;$

$\mathrm{GsynthCdc20} : (\mathrm{k5} + (\mathrm{k51} * ((\mathrm{GCycBscaled} * \mathrm{Gmscaled}/\mathrm{j5})^{n}/(1 + (\mathrm{GCycBscaled} * \mathrm{Gmscaled}/\mathrm{j5})^{n})))) * 100;$

$\mathrm{GdegraCdc20} : (\mathrm{k6} * \mathrm{GCdc20Tscaled}) * 100;$

$\mathrm{GactCdc20} : ((\mathrm{k7} * \mathrm{GIEPscaled} * (\mathrm{GCdc20Tscaled} - \mathrm{GCdc20Ascaled})$

$\qquad\qquad/(\mathrm{j7} + \mathrm{GCdc20Tscaled} - \mathrm{GCdc20Ascaled}))) * 100;$

$\mathrm{GinactCdc20} : (((\mathrm{k8} * \mathrm{Mad} * \mathrm{GCdc20Ascaled})/(\mathrm{j8} + \mathrm{GCdc20Ascaled})) + (\mathrm{k6} * \mathrm{GCdc20Ascaled})) * 100;$

$\mathrm{GaddIEP} : (\mathrm{k9} * \mathrm{Gmscaled} * \mathrm{GCycBscaled} * (1 - \mathrm{GIEPscaled})) * 100;$

$\mathrm{GremoveIEP} : (\mathrm{k10} * \mathrm{GIEPscaled}) * 100;$

Figure 6.1: Cell Cycle PAL Model Part 1 of 3.

Internal species actions

$Dsynthesis : k1 * 100;$

$Ddegradation : ((k2 * (p53scaled/0.19) + k21 * DCdh1scaled) * DCycBscaled) * 100;$

$Dactivation : (((k3 + k31 * DCdc20Ascaled) * (1 - DCdh1scaled))/(j3 + 1 - DCdh1scaled)) * 100;$

$Dinactivation : ((k4 * Dmscaled * DCycBscaled * DCdh1scaled)/(j4 + DCdh1scaled)) * 100;$

$Dcellgrowth : ((u * Dmscaled) * (1 - (Dmscaled/m2))) * 100;$

$DsynthCdc20 : (k5 + (k51 * ((DCycBscaled * Dmscaled/j5)^n/(1 + (DCycBscaled * Dmscaled/j5)^n)))) * 100;$

$DdegraCdc20 : (k6 * DCdc20Tscaled) * 100;$

$DactCdc20 : ((k7 * DIEPscaled * (DCdc20Tscaled - DCdc20Ascaled))$

$/(j7 + DCdc20Tscaled - DCdc20Ascaled))) * 100;$

$DinactCdc20 : (((k8 * Mad * DCdc20Ascaled)/(j8 + DCdc20Ascaled)) + (k6 * DCdc20Ascaled)) * 100;$

$DaddIEP : (k9 * Dmscaled * DCycBscaled * (1 - DIEPscaled)) * 100;$

$DremoveIEP : (k10 * DIEPscaled) * 100;$

start : $fMA(100 * GtoD);$

finish1 : $fMA(100 * divide1);$

finish2 : $fMA(100 * divide2);$

$addp53 : ((s53) + (s53i * ((Mdm2cscaled^4)/(js53^4 + Mdm2cscaled^4)))) * 100;$

$removep53 : (kd53 * p53scaled) * 100;$

$addMdm2c : ((s2) + (s2i * ((p53scaled^4)/(js2^4 + p53scaled^4))) + (ko * Mdm2nscaled)) * 100;$

$removeMdm2c : ((ki * Mdm2cscaled) + (d2i * Mdm2cscaled)) * 100;$

$addMdm2n : (ki * Mdm2cscaled) * 100;$

$removeMdm2n : ((ko * Mdm2nscaled) + (kd * Mdm2nscaled)) * 100;$

Gdeath : $fMA(Death);$

Ddeath : $fMA(Death);$

Internal species

$GCycB = (Gsythesis, 1) >> +(Gdegradation, 1) <<;$

$GCdh1 = (Gactivation, 1) >> +(Ginactivation, 1) <<;$

$Gm1 = (Gcellgrowth, 1) >>;$

$GCdc20T = (GsynthCdc20, 1) >> +(GdegraCdc20, 1) <<;$

$GCdc20A = (GactCdc20, 1) >> +(GinactCdc20, 1) <<;$

$GIEP = (GaddIEP, 1) >> +(GremoveIEP, 1) <<;$

$Gtracker\_off = (\mathbf{start}, 1) <<;$

$Gtracker\_on = (\mathbf{start}, 1) >>;$

Figure 6.2: Cell Cycle PAL Model Part 2 of 3.

Internal species

$DCycB = (Dsythesis, 1) >> +(Ddegradation, 1) <<;$

$DCdh1 = (Dactivation, 1) >> +(Dinactivation, 1) <<;$

$Dm1 = (Dcellgrowth, 1) >>;$

$DCdc20T = (DsynthCdc20, 1) >> +(DdegraCdc20, 1) <<;$

$DCdc20A = (DactCdc20, 1) >> +(DinactCdc20, 1) <<;$

$DIEP = (DaddIEP, 1) >> +(DremoveIEP, 1) <<;$

$Dtracker\_off1 = (\textbf{finish1}, 1) <<;$

$Dtracker\_on1 = (\textbf{finish1}, 1) >>;$

$Dtracker\_off2 = (\textbf{finish2}, 1) <<;$

$Dtracker\_on2 = (\textbf{finish2}, 1) >>;$

$p53 = (addp53, 1) >> +(removep53, 1) <<;$

$Mdm2c = (addMdm2c, 1) >> +(removeMdm2c, 1) <<;$

$Mdm2n = (addMdm2n, 1) >> +(removeMdm2n, 1) <<;$

Set of hidden internal actions

$HiddenActions\ A : \{Gsythesis, Gdegradation, Gactivation, Ginactivation, Gcellgrowth, GsynthCdc20,$
$GdegraCdc20, GactCdc20, GinactCdc20, GaddIEP, GremoveIEP, Dsythesis,$
$Ddegradation, Dactivation, Dinactivation, Dcellgrowth, DsynthCdc20, DdegraCdc20,$
$DactCdc20, DinactCdc20, DaddIEP, DremoveIEP, addp53, removep53,$
$addMdm2c, removeMdm2c, addMdm2n, removeMdm2n\};$

Cells (Internal species model)

$G = GCycB[5] \underset{*}{\bowtie} GCdh1[100] \underset{*}{\bowtie} Gm1[60] \underset{*}{\bowtie} GCdc20T[180] \underset{*}{\bowtie} GCdc20A[135]$
$\underset{*}{\bowtie} GIEP[70] \underset{*}{\bowtie} GTracker\_off[1] \underset{*}{\bowtie} GTracker\_on[0] \underset{*}{\bowtie} p53[19] \underset{*}{\bowtie} Mdm2c[16] \underset{*}{\bowtie} Mdm2n[65]$

$D = DCycB[12] \underset{*}{\bowtie} DCdh1[1] \underset{*}{\bowtie} Dm1[89] \underset{*}{\bowtie} DCdc20T[7] \underset{*}{\bowtie} DCdc20A[0]$
$\underset{*}{\bowtie} DIEP[34] \underset{*}{\bowtie} DTracker\_off1[1] \underset{*}{\bowtie} Dtracker\_on1[0] \underset{*}{\bowtie} Dtracker\_off2[1] \underset{*}{\bowtie} Dtracker\_on2[0]$
$\underset{*}{\bowtie} p53[19] \underset{*}{\bowtie} Mdm2c[16] \underset{*}{\bowtie} Mdm2n[65]$

Cell Populations (External actions)

$Growing\{\{G\}\} = \textbf{start} \downarrow +\textbf{finish1} \uparrow +\textbf{finish2} \uparrow +\textbf{Gdeath} \downarrow;$

$Dividing\{\{D\}\} = \textbf{start} \uparrow +\textbf{finish1}((+)) + \textbf{finish2} \downarrow +\textbf{Ddeath} \downarrow;$

Model Component

$Growing[\![G\_1, ..., G\_8]\!]_A \underset{\{start, finish1, finish2\}}{\Diamond} Dividing[\![\ ]\!]_A$

Figure 6.3: Cell Cycle PAL Model Part 3 of 3.

Parameter	Value	Parameter	Value	Parameter	Value
k1	0.12	j1	0.1	CycBT	10
k2	0.12	j2	0.1	CycBTdivide1	10
k21	4.5	theta	0.83	CycBTdivide2	9
k3	3	s53	0.6	deathT	4
k31	30	s53i	2.56		
j3	0.04	js53	0.45		
k4	105	s2	0.15		
j4	0.04	s2i	4.23		
u	0.03	js2	0.92		
m2	10	ki	0.41		
k5	0.015	ko	0.05		
k51	0.6	d2i	0.79		
k6	0.3	rep	0.08		
k7	3	DAM	0 to 10		
k8	1.5				
k9	0.3				
k10	0.06				
j5	0.3				
n	4				
j7	0.001				
j8	0.001				
Mad	1				

Table 6.1: Parameter values of Cell Cycle PAL model. First column parameters taken from Powathil et al [61] and second column parameters taken from Zhang et al [77]. Third column threshold parameters for G and D cells.

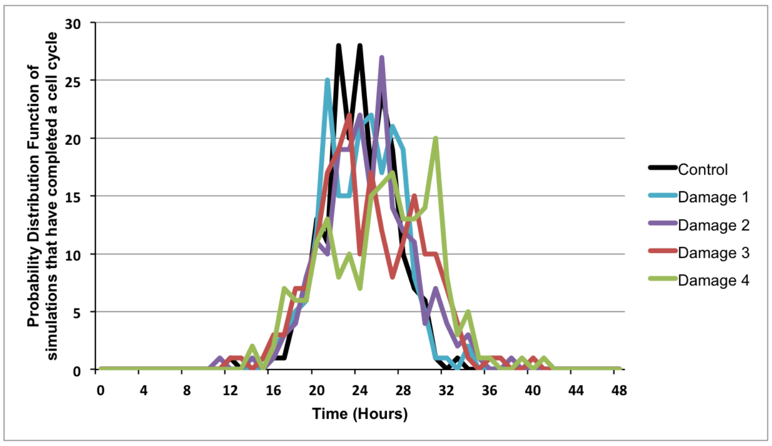


Figure 6.4: Simulation Distributions PDF values of the Length of a cell cycle.

reality that although all cells have the same dose of treatment the damage is varied over the cell population. Damage occurs immediately (at time zero) in all simulation experiments.

6.3 MODEL ANALYSIS

The PAL parser was used to translate the PAL model to a Bio-PEPA model to allow analysis of the model in the Bio-PEPA plug-in. This analysis includes simulation distributions and time series analysis.

6.3.1 *Simulation Distributions Analysis of Average Length of a cell cycle*

Simulation distributions analysis was undertaken to analyse the average length of the cell cycle and how increasing the amount of damage affects this. Results from this analysis are presented for five experiments and are given in Figure 6.4 and Table 6.2. Five experiments were carried out from a control with no damage to damage of four. As this analysis is observing one cell cycle the PAL model in these experiments starts with one G cell. The G cell is assigned a specific damage dependent on the experiment. The chosen component in this analysis is an agent which tracks when a D cell becomes a G cell, i.e. completion of one cell cycle. The number of stochastic simulation replications is set to 200 and the stop time set to 48 hours. A higher number of replications (1000) would be recommended although this would be computationally expensive (in excess of 24 hours run time).

All 100% of the simulations from the five experiments completed a cell cycle before the stop time of 48 hours. The results show that a damage from one to three does not affect the cell cycle average length. The intracellular proteins can cope with these damage levels. The average cell cycle length slightly increases when a damage of four is applied.

Experiment	Average cycle	± Confidence Interval (95%)
Control	23.96	0.4472
Damage 1	24.18	0.4780
Damage 2	24.73	0.5659
Damage 3	24.74	0.6523
Damage 4	25.88	0.6913

Table 6.2: Average length of cell cycle and 95% confidence interval in hours of each simulation distributions experiment.

6.3.2 *Time Series Analysis of cell Population Growth*

Discrete stochastic simulation time-series analysis was carried out to analyse cell growth and how increasing the amount of damage affects this. Continuous ODEs simulation time-series analysis could not be carried out as the ODEs solver in the Bio-PEPA plug-in did not handle the translated PAL model.

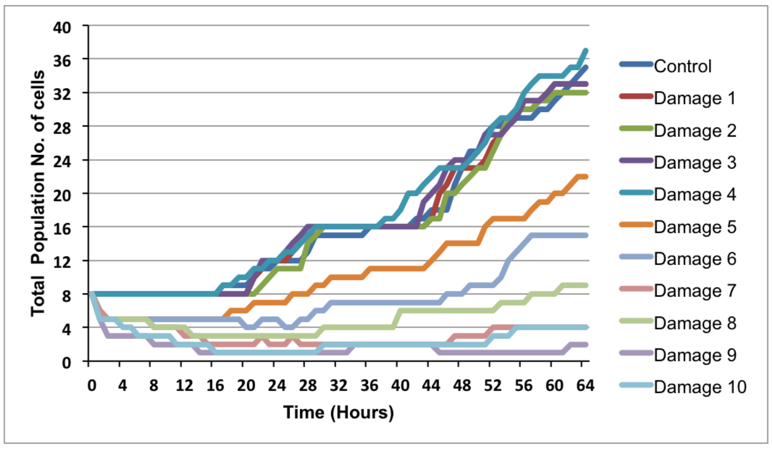


Figure 6.5: Time Series Analysis of total cell Population Growth.

The initial population is eight G cells different damage levels are assigned to each cell randomly. Throughout the time period of the simulation new cells will be assigned different damage levels reliant on the constant repair. Eleven experiments were carried out from a control with no damage to damage of ten. The total time for each experiment is 64 hours (2 days 16 hours). This time was chosen based on the control experiment and the number of cells remaining at the end (35 cells). Populations are kept to a relatively low value due to the capabilities of the Bio-PEPA plug-in. Each experiment is one stochastic simulation. Results from this analysis are presented for eleven experiments and the total population growth of G and D cells results are given in Figure 6.5 and separate population growth results of G and D cells are given in Figure 6.6.

The results show that population growth is not affected by a damage from one to four. These results are similar to the simulation distribution results which show the cell cycle is not affected by this lower damage range. The damage in each experiment is repaired at a constant rate, therefore the population of cells start to recover dependent on the damage assigned. Death occurred in the experiments where damage is above four. This feature had the effect of reducing the population. The damage levels of nine and ten had a greater reduction in the population.

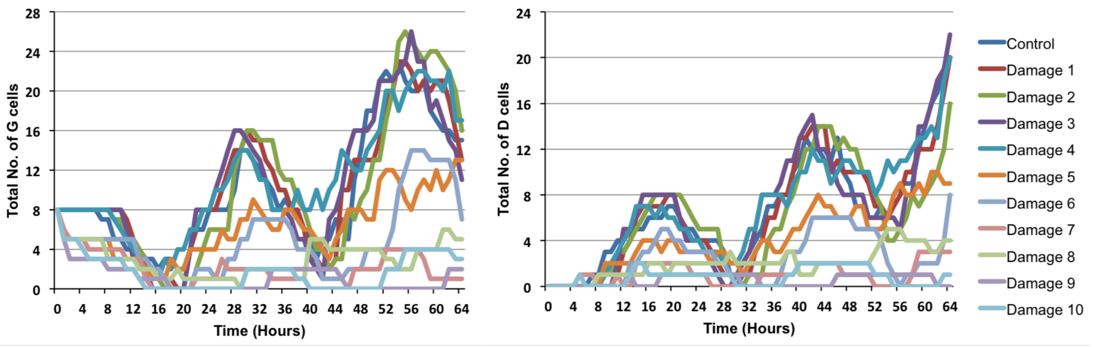


Figure 6.6: Time Series Analysis of cell Population Growth G and D cells.

6.3.3 *Comparison with wet laboratory data survival fraction results*

The population results of the PAL model are compared to wet laboratory survival fraction results of cells given different Gray (Gy) doses of radiation treatment [52]. The results from the wet laboratory data had the experiment duration of twelve days and the initial populations of around 200 to 600 cells. The capabilities of the Bio-PEPA plug-in does not allow high numbers of cells but nevertheless comparisons can be made. The end population value data point is taken from the eleven population experiment results shown in Figure 6.5 and a survival fraction is calculated based on the control experiment. The survival fraction results for damage levels of 5 and 6 are compared with the survival fraction results for Gy doses of 2.25 and 3 as suggested by Nicol et al [52]. The other survival fraction results are compared with the damage levels based on this correlation. Damage levels 0, 2, 4, 8 and 10 are compared with radiation doses of 0, 0.75, 1.5 , 4.5 and 6 respectively. The survival fraction comparison results are given in Figure 6.7. The results overall show a good comparison with the trend of the wet laboratory data. The survival fraction result for Damage 4 of the PAL model is less comparable, giving a slightly higher survival fraction than the control. This may be due to the fact that this result is taken from one stochastic simulation if an average population value was taken from repeated simulations the comparison may improve.

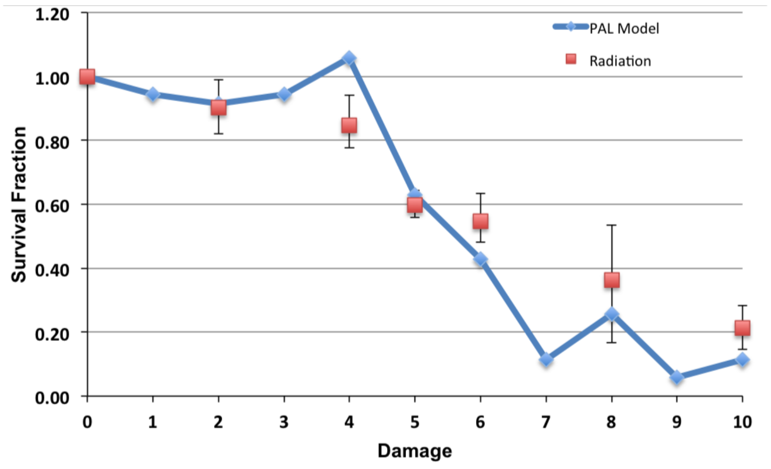


Figure 6.7: Comparison results between PAL model population and wet laboratory radiation treatment results.

Further work could be undertaken to compare the PAL model results to other cancer treatments such as Temozolomide (TMZ) and combination of these treatments (radiation + TMZ). A function of the relationship of the linear damage levels to the doses of these treatments would have to be found to allow the comparison of survival fraction results. More stochastic simulations should be carried out to find an average population value from all the experiments and may give a better comparison with the wet laboratory data.

6.4 SUMMARY

In this chapter PAL has been shown to be a generic multi-scale language. It has been applied successfully to a cell cycle and DNA damage multi-scale system. Analysis of the model focused on different scales of the case study from the length of a single cell cycle to population cell growth.

It has been shown in the literature that process algebra can be utilised to investigate this specific problem. This is presented in a single scale process algebra model that describes the activities of cells in a epidemic-type structure based on radiation treatment [43]. This model does not include the growing and dividing phases of a cell and the dynamics of their intracellular proteins, therefore, missing important multi-scale system interactions. The PAL model is the first example of a multi-scale process algebra model specifically investigating how cell damage affects intracellular protein interactions and how these interactions impact on cell population growth.

Future work on this PAL model could include testing a variety of different degradation rates for CycB affected by p53. The degradation of CycB in G and D cells could be changed as some cancer treatments specifically affect the different states of a cell [60]. The damage repair could be more specific to the levels of p53 and Mdm2. The assumption of death due to damage greater than a certain threshold could be made more specific to certain treatments and the survival fraction of cells. The comparison of results to reality shows how PAL can be used to aid in investigations of cancer treatment in multi-scale systems.

7 CONCLUSION

7.1 THESIS SUMMARY

Multi-scale modelling and analysis is becoming increasingly important and relevant. Analysis of the emergent properties from the interactions between scales of multi-scale systems is important to aid in solutions. Most modelling approaches are specific to the problem that they are addressing and use a hybrid combination of modelling languages to model specific scales.

In this thesis the potential of process algebra is highlighted to model multi-scale systems. Process Algebra with Layers has been demonstrated to provide an elegant language to write and evaluate integrated multi-scale models.

In Chapter 2 I investigated whether a process algebra language such as Bio-PEPA could model and analyse a single scale specifically an organism's physiology within a specific life stage [59]. The DEB theory [40] approach was used to gain abstraction and still capture enough detail to accurately represent the scale. As a result of this investigation I proposed a generic translation approach to translate DEB models to Bio-PEPA models. This work confirmed that Bio-PEPA can successfully model and analyse this specific scale. As the Bio-PEPA plug-in offers a variety of analysis techniques further analysis of the system was achieved [71].

In Chapter 3 I validated the generic translation approach by successfully translating a DEB model [67] of an organism's physiology within a different life stage. My two Bio-PEPA life stage models were integrated. As a result of this the limitations of Bio-PEPA in the modelling of integrated multi-scales were found. These limitations provided the challenges and features that had to be addressed in a process algebra for multi-scale integration modelling. These included effectively integrating and modularising multi-scales, scaling of units appropriate to an organism's life stage and including a population view of the system.

In Chapter 4 I introduced Process Algebra with Layers (PAL): a language for multi-scale integration modelling. PAL addresses all the features and challenges reported in Chapter 3. It also encompasses the important multi-scale modelling features and strategy stated by Noble [53] and Allen et al [10]. PAL is also compared to two multi-scale process algebra languages. Results from this comparison revealed PAL's ability to model a multi-scale system in a more concise and elegant fashion. This is because PAL specifically focuses on the easy definition of integration and interaction between scales. PAL also allows the interaction of the addition and removal of a component (with an internal system description) from a population whereas the compared languages did not. A PAL parser was implemented to automatically translate PAL models to Bio-PEPA models. This results in the various analysis techniques of the Bio-PEPA plug-in becoming available in the analysis of PAL models.

In Chapter 5 I illustrated the use of PAL by applying it to the Pacific oyster life stage case study (which I investigated in the integrated life stage Bio-PEPA model in Section 3.2 of Chapter 3) and investigated the impacts of ocean acidification. The PAL model included the oyster's physiology, life stage and population scale. Results of the analysis of different scales of the system were analysed. Results from this chapter show that PAL models can be used to aid in the analysis of topical issues such as climate change.

In Chapter 6 I demonstrated PAL's generic ability by applying it to a cell cycle and DNA damage case study investigating the effects of cancer treatment. The PAL model included the intracellular, cellular and population scale. Analysis techniques were utilised to gain results from different scales of the system including the length of a cell cycle and population growth. Results from this chapter show PAL results can be compared to wet laboratory data to aid in the analysis of this specific multi-scale case study.

With the above results the characteristics of PAL are highlighted in the following section.

7.1.1 *Characteristics of PAL*

Characteristics of the language are the use of novel multi-scale layers and the interactions between these layers using mirrored actions. These characteristics enables PAL to overcome the features and challenges found in Chapter 3. PAL effectively integrates and modularises multi-scales of a system using these layers. As the layers modularise the detail within a scale the appropriate unit of scale can be applied. A population view of a multi-scale system can be modelled in PAL. The population scale can include population actions such as addition and removal from a population.

PAL encompasses all the multi-scale modelling features stated by Noble [53] and Allen et al [10].

Abstraction, PAL allows the modeller to begin modelling at a scale and detail that is appropriate to the knowledge they have about the system and the problem they are addressing. The modularity of the layers allows the modeller to elegantly construct a model and aids them to incorporate the detail needed to be included in each scale.

Descriptive, PAL models have an explicitly descriptive view of a multi-scale system. The internal system view of the Organism layer incorporates the Bio-PEPA modelling language's descriptive view of a biological system. The language can incorporate approaches such as DEB theory or ODEs modelling to allow the scale to accurately reflect a specific scale of a multi-scale system. The layers of the language modularises the details within a PAL model making its description accessible and easy to read.

Integrative, PAL allows the modeller the ability to integrate scales of a multi-scale system and model the interactions between these scales. Changes to an internal system within a component causing a change in the components state, e.g physiological changes within an organism changing what life stage the organism is in, can be modelled easily. The interaction of the addition and removal of a component from a population, e.g birth and death, can be easily modelled.

Explanatory, PAL is shown to provide predictions and insights to multi-scale systems by utilising the Bio-PEPA plug-in's various analysis techniques. This allows users the ability to export and use the model in their preferred model and analysis tool.

Generic, PAL is shown to be a generic language as it has been applied to case studies of multi-scale systems in the fields of marine ecology and systems biology. Models can be adapted so the appropriate unit of scale within a scale can be applied.

Middle-out strategy, PAL follows this strategy as the modeller can decide the starting scale of a multi-scale model. PAL achieves this as the Organism layer can represent a molecule, organelle, cell, tissue, organ or any organism. Thereafter the internal species can also represent a molecule, organelle, cell, tissue, organ or any organism.

7.2 FUTURE WORK

In this thesis, the foundations of Process Algebra with Layers for multi-scale integration modelling has been established, additional research is needed before it can be considered as a competitor against the other well-known and traditional approaches.

The PAL parser automatically translates PAL models to Bio-PEPA models. This allows the use of a variety of analysis techniques available within the Bio-PEPA plug-in to be applied to a PAL model. The Bio-PEPA plug-in can also translate Bio-PEPA models into other modelling languages. This allows PAL modellers to use their preferred model and analysis tool. This gives PAL models the ability of the explanatory multi-scale feature. The use of the Bio-PEPA plug-in with PAL models revealed some issues. For example, in Chapters 5 and 6 the Bio-PEPA plug-in limited the population size in the model. To eliminate the issues that have arisen during the analysis of PAL models because of the restrictions of the Bio-PEPA plug-in, a possible solution could be the implementation of a PAL model solver. This proposed solver would allow larger population sizes to be set and simulated. Efficient simulation algorithms would need to be defined to accomplish this. The solver would have to encapsulate the explanatory multi-scale feature.

Another issue identified with the Bio-PEPA plug-in is that hand written functions approximating the environmental data were required: it would be desirable to incorporate collected data as the forcing variables have a significant effect on the model. This may lead to more comparable results.

Additional syntax and semantics could be defined to describe the interactions between Organisms. The Organisms in PAL currently do not have the ability to interact explicitly with one another. This would involve explicitly modelling space. Organisms would, for example, need location attributes to therefore react to their surrounding Organisms. It would be necessary to ensure this addition would not compromise the integrative nature of PAL. This spatial definition may overcomplicate the definition of a PAL model which may lead to the loss of some multi-scale features PAL already encapsulates.

Parameter passing between life stages of an Organism is another possible addition to the language. In PAL currently when an Organism transitions to another life stage it takes on the initial parameter values set up for that life stage. It does not take its internal species values forward. In reality organisms mature at different rates and values. Parameter passing would increase its accuracy to represent a system.

PAL is shown to be generic in the fields of marine ecology and systems biology. This shows that the two layers of PAL can capture different multi-scale systems within one model. It would be interesting to apply PAL to a problem outwith these fields, where modelling the interactions between scales is important. For example, consider the interaction of micro and macro parasites and the immune system, with host states (susceptible, infected and recovered) within a population. The parasites and immune system of the host could be described by the internal species of a PAL Organism, the Organism being the host. Different PAL Populations of these hosts could be defined to describe the different host states. The Organisms within these Populations would be different based on these states. These differences could be based on actions, parameter values or rates of actions etc.

7.3 SUMMARY

In this chapter the final conclusions of the thesis are discussed. Highlighting the key investigations and results. The definition and application of the novel generic Process Algebra with Layers language for multi-scale integration modelling has been realised. It allows the clear definition of scales and interactions within and between these scales in one model. Future research is needed to show its full capabilities.

A

APPENDIX A

I present below the Larva Bio-PEPA model in Figure A.1 and parameters in Table A.1 discussed in Chapter 3.

Also presented below the integrated life stage Bio-PEPA model in Figures A.2 and A.3 discussed in Chapter 3.

Symbol	Definition	Value	Dimension
$[E_G]$	Volume-Specific costs for structure	$1900 * 10^{-12}$	$J\mu m^{-3}$
$[E_M]$	Maximum energy storage density	$2295 * 10^{-12}$	$J\mu m^{-3}$
κ	Fraction of utilised energy spent on growth and maintenance	0.45	-
δ_M	Shape coefficient	0.658	-
X_K	Half-saturation coefficient	600	$\mu m^3 \mu l^{-1}$
$\{\dot{P}_{Xm}\}$	Maximum surface area-specific ingestion rate	137	$\mu m^3 d^{-1} \mu m^{-2}$
ae	Assimilation efficiency	0.4	-
$[\dot{P}_M]$	Volume-specific maintenance rate	$24 * 10^{-12}$	$Jd^{-1}\mu m^{-3}$
μ_X	Energy content of food	$4.5 * 10^{-9}$	$J\mu m^{-3}$
T_1	Reference temperature	298	K
T_A	Arrhenius temperature	11000	K
T_{AH}	Rate of decrease at upper boundary	75000	K
T_{AL}	Rate of decrease at lower boundary	75000	K
T_H	Upper boundary of tolerance range	306	K
T_L	Lower boundary of tolerance range	285	K

Table A.1: Model parameters used in Figure A.1.

Parameters of model

$$\begin{aligned}
E_V &= \mathrm{Biovolume} * 10^{-6};\\
E &= \mathrm{Reserve} * 10^{-6};\\
E_R &= \mathrm{Development} * 10^{-6};\\
L &= (E_V/[E_G])^{1/3}/\delta_M;\\
V &= (\delta_M * L)^3;\\
\mathrm{Actual\_temperature} &= \text{value dependent on experiment ;}\\
\mathrm{Temperature\_correction} &= \exp((T_A/T_1) - (T_A/(273 + \mathrm{Actual\_temperature})))\\
&\quad * ((1 + \exp((T_{AL}/(273 + \mathrm{Actual\_temperature})) - (T_{AL}/T_L))\\
&\quad + \exp((T_{AH}/T_H) - (T_{AH}/(273 + \mathrm{Actual\_temperature}))))^{-1});\\
\{\dot{P}_{Xm}\} &= 137 * \mathrm{Temperature\_correction};\\
\{\dot{P}_{Am}\} &= ae * \mu_x * \{\dot{P}_{Xm}\};\\
[\dot{P}_M] &= 24 * 10^{-12} * \mathrm{Temperature\_correction};\\
\dot{P}_M &= [\dot{P}_M] * V;\\
[E] &= E/V;\\
\dot{P}_C &= ([E]/([E_G] + (\kappa * [E]))) * ((\tfrac{[E_G]*\{\dot{P}_{Am}\}*V^{2/3}}{[E_M]}) + ([\dot{P}_M] * V));\\
\mathrm{Food\_density} &= \text{value dependent on experiment ;}\\
\mathrm{Functional\_response} &= \tfrac{\mathrm{Food\_density}^2}{(\mathrm{Food\_density}^2 + X_\kappa^2)};\\
\dot{P}_A &= \mathrm{Functional\_response} * \{\dot{P}_{Am}\} * V^{2/3};\\
\dot{P}_J &= ((\tfrac{(1-\kappa)}{\kappa}) * V * [\dot{P}_M]);\\
\mathrm{Stop\_action} &= H(1 - E_R);
\end{aligned}$$

Actions and their associated kinetic rates

$$\begin{aligned}
\textit{kineticLawOf}\ a1 &: (\kappa * \dot{P}_C) * 10^6;\\
\textit{kineticLawOf}\ a2 &: \dot{P}_M * 10^6;\\
\textit{kineticLawOf}\ a3 &: \dot{P}_A * 10^6;\\
\textit{kineticLawOf}\ a4 &: \dot{P}_C * 10^6;\\
\textit{kineticLawOf}\ a5 &: ((1 - \kappa) * \dot{P}_C) * 10^6;\\
\textit{kineticLawOf}\ a6 &: \dot{P}_J * 10^6 * (1 - \mathrm{Stop\_action});
\end{aligned}$$

Agent definitions

$$\begin{aligned}
\mathrm{Biovolume} &= a1 \uparrow + a2 \downarrow + a3 \odot + a4 \odot + a5 \odot + a6 \odot;\\
\mathrm{Reserve} &= a3 \uparrow + a4 \downarrow + a1 \odot + a5 \odot;\\
\mathrm{Development} &= a5 \uparrow + a6 \downarrow;
\end{aligned}$$

Model Component

$$\mathrm{Biovolume}[250] \underset{*}{\bowtie} \mathrm{Reserve}[250] \underset{*}{\bowtie} \mathrm{Development}[0]$$

Figure A.1: Pacific oyster Larva Bio-PEPA model. See Table A.1 for other parameters.

Parameters of model

$$
\begin{aligned}
E_V &= Biovolume * 10^{-6};\\
E &= Reserve * 10^{-6};\\
E_R &= Development * 10^{-6};\\
L &= (((E_V/[L\_E_G])^{1/3}/\delta_M) * L\_Switch\_J)\\
&\quad +(((E_V/[JA\_E_G])^{1/3}/0.175) * (1 - L\_Switch\_J));\\
V &= (((\delta_M * L)^3) * L\_Switch\_J) + (((0.175 * L)^3) * (1 - L\_Switch\_J));\\
Actual\_temperature &= \text{value dependent on experiment };\\
Temperature\_correction &= ((exp((L\_T_A/L\_T_1) - (L\_T_A/(273 + Actual\_temperature)))\\
&\quad * ((1 + exp((L\_T_{AL}/(273 + Actual\_temperature)) - (L\_T_{AL}/L\_T_L))\\
&\quad + exp((L\_T_{AH}/L\_T_H) - (L\_T_{AH}/(273 + Actual\_temperature))))^{-1}))\\
&\quad *L\_Switch\_J)+\\
&\quad ((exp((JA\_T_A/JA\_T_1) - (JA\_T_A/(273 + Actual\_temperature)))\\
&\quad * ((1 + exp((JA\_T_{AL}/(273 + Actual\_temperature)) - (JA\_T_{AL}/JA\_T_L))\\
&\quad + exp((JA\_T_{AH}/JA\_T_H) - (JA\_T_{AH}/(273 + Actual\_temperature))))^{-1}))\\
&\quad *(1 - L\_Switch\_J));\\
\{\dot{P}_{Xm}\} &= ((137 * Temperature\_correction) * L\_Switch\_J)\\
&\quad +((560 * Temperature\_correction) * (1 - L\_Switch\_J));\\
\{\dot{P}_{Am}\} &= (((ae * \mu_x * \{\dot{P}_{Xm}\}) * 10^{-9}) * L\_Switch\_J)\\
&\quad +((ae * \{\dot{P}_{Xm}\}) * (1 - L\_Switch\_J));\\
[\dot{P}_M] &= (((24 * 10^{-12}) * L\_Switch\_J)\\
&\quad +(24 * (1 - L\_Switch\_J))) * Temperature\_correction;\\
\dot{P}_M &= [\dot{P}_M] * V;\\
[E] &= E/V;\\
\dot{P}_C &= ((([E]/([L\_E_G] + (\kappa * [E]))) * ((\tfrac{[L\_E_G]*\{\dot{P}_{Am}\}*V^{2/3}}{[E_M]}) + ([\dot{P}_M] * V))) * L\_Switch\_J)\\
&\quad +(((([E]/([JA\_E_G] + (\kappa * [E]))) * ((\tfrac{[JA\_E_G]*\{\dot{P}_{Am}\}*V^{2/3}}{[E_M]}) + ([\dot{P}_M] * V)))\\
&\quad *(1 - L\_Switch\_J));
\end{aligned}
$$

Figure A.2: Integrated life stage Bio-PEPA model Part 1 of 2. The prefix L and JA on the parameter names refer to parameter values in Table A.1 and Table 2.1 respectively.

Parameters of model continued

$$
\begin{aligned}
\mathrm{Food\_density} &= (1400 * \mathrm{L\_Switch\_J}) + ((1400 * 10^{-12}) * (1 - \mathrm{L\_Switch\_J}));\\
X_{\kappa} &= (600 * \mathrm{L\_Switch\_J}) + ((600 * 10^{-12}) * (1 - \mathrm{L\_Switch\_J}));\\
\mathrm{Functional\_response} &= \left(\frac{\mathrm{Food\_density}^2}{(\mathrm{Food\_density}^2 + X_{\kappa}^2)} * \mathrm{L\_Switch\_J}\right)\\
&\quad + \left(\frac{\mathrm{Food\_density}}{(\mathrm{Food\_density} + X_{\kappa})} * (1 - \mathrm{L\_Switch\_J})\right);\\
\dot{P}_A &= \mathrm{Functional\_response} * \{\dot{P}_{Am}\} * V^{2/3};\\
\mathrm{Maturity} &= H(V - V_p);\\
\dot{P}_J &= \left(\left(\left(\frac{(1-\kappa)}{\kappa}\right) * V * [\dot{P}_M]\right) \ * \ (1 - \mathrm{Maturity})\right)\\
&\quad + \left(\left(\left(\frac{(1-\kappa)}{\kappa}\right) * Vp * [\dot{P}_M]\right) \ * \ (\mathrm{Maturity})\right);\\
\mathrm{Percentage\_E_R} &= \left(\frac{E_R\_DFW}{\mathrm{Total\_DFW}}\right) * 100;\\
E_R\mathrm{\_start\_spawn} &= H(\mathrm{Percentage\_E_R} - GSI);\\
\mathrm{Stop\_spawn} &= H(1 - \mathrm{Percentage\_E_R});\\
\mathrm{T\_start\_spawn} &= H(\mathrm{Actual\_temperature} - T_s);\\
\mathrm{Stop\_action} &= H(1 - E_R);
\end{aligned}
$$

Actions and their associated kinetic rates

$$
\begin{aligned}
\textit{kineticLawOf}\ a1 &: (\kappa * \dot{P}_C) * 10^6;\\
\textit{kineticLawOf}\ a2 &: \dot{P}_M * 10^6;\\
\textit{kineticLawOf}\ a3 &: \dot{P}_A * 10^6;\\
\textit{kineticLawOf}\ a4 &: \dot{P}_C * 10^6;\\
\textit{kineticLawOf}\ a5 &: ((((1-\kappa) * \dot{P}_C) * 10^6) * \mathrm{L\_Switch\_J})\\
&+ (((((1-\kappa) * \dot{P}_C) * 10^6) * \mathrm{Maturity}) * (1 - \mathrm{L\_Switch\_J}));\\
\textit{kineticLawOf}\ a6 &: ((\dot{P}_J * 10^6 * (1 - \mathrm{Stop\_action})) * \mathrm{L\_Switch\_J})\\
&+ (((\dot{P}_J * 10^6) * \mathrm{Maturity} * (1 - \mathrm{Stop\_action})) * (1 - \mathrm{L\_Switch\_J}));\\
\textit{kineticLawOf}\ \mathrm{empty} &: \textit{fMA}(100 * \mathrm{Maturity});\\
\textit{kineticLawOf}\ \mathrm{switch\_on} &: \textit{fMA}(10 * E_R\mathrm{\_start\_spawn} * \mathrm{T\_start\_spawn});\\
\textit{kineticLawOf}\ \mathrm{switch\_off} &: \textit{fMA}(10 * \mathrm{stop\_spawn});
\end{aligned}
$$

Agent definitions

$$
\begin{aligned}
\mathrm{Biovolume} &= a1 \uparrow + a2 \downarrow + a3 \odot + a4 \odot + a5 \odot + a6 \odot + \mathrm{empty} \odot;\\
\mathrm{Reserve} &= a3 \uparrow + a4 \downarrow + a1 \odot + a5 \odot;\\
\mathrm{Development} &= a5 \uparrow + a6 \downarrow + (\mathrm{empty}, 1) \downarrow;\\
\mathrm{Tracker\_off} &= (\mathrm{switch\_on}, 1) \downarrow + (\mathrm{switch\_off}, 1) \uparrow;\\
\mathrm{Tracker\_on} &= (\mathrm{switch\_on}, 1) \uparrow + (\mathrm{switch\_off}, 1) \downarrow + (\mathrm{empty}, 1) \oplus;
\end{aligned}
$$

Model Component

$$
\mathrm{Biovolume}[250] \underset{*}{\bowtie} \mathrm{Reserve}[250] \underset{*}{\bowtie} \mathrm{Development}[0] \underset{*}{\bowtie} \mathrm{Tracker\_off}[1] \underset{*}{\bowtie} \mathrm{Tracker\_on}[0]
$$

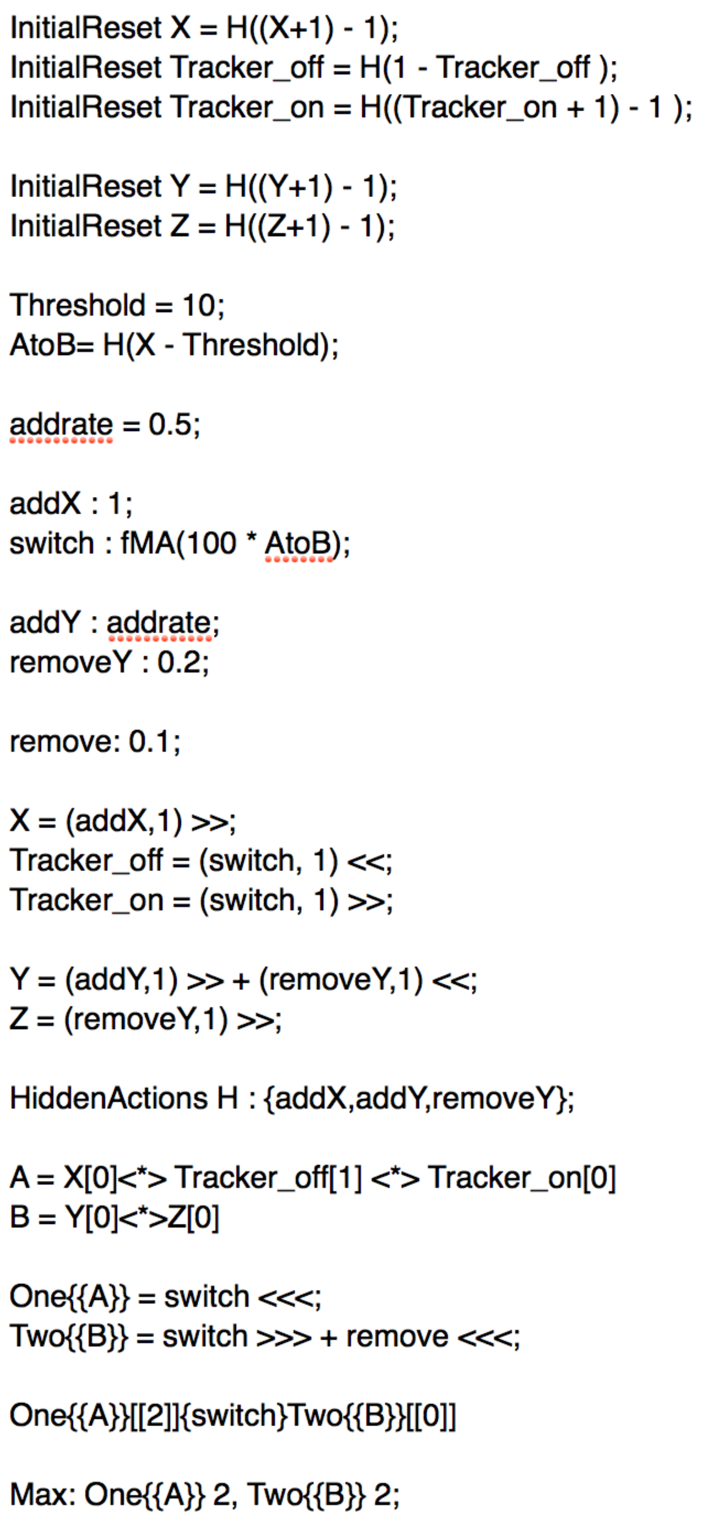
Figure A.3: Integrated life stage Bio-PEPA model Part 2 of 2.

B

APPENDIX B

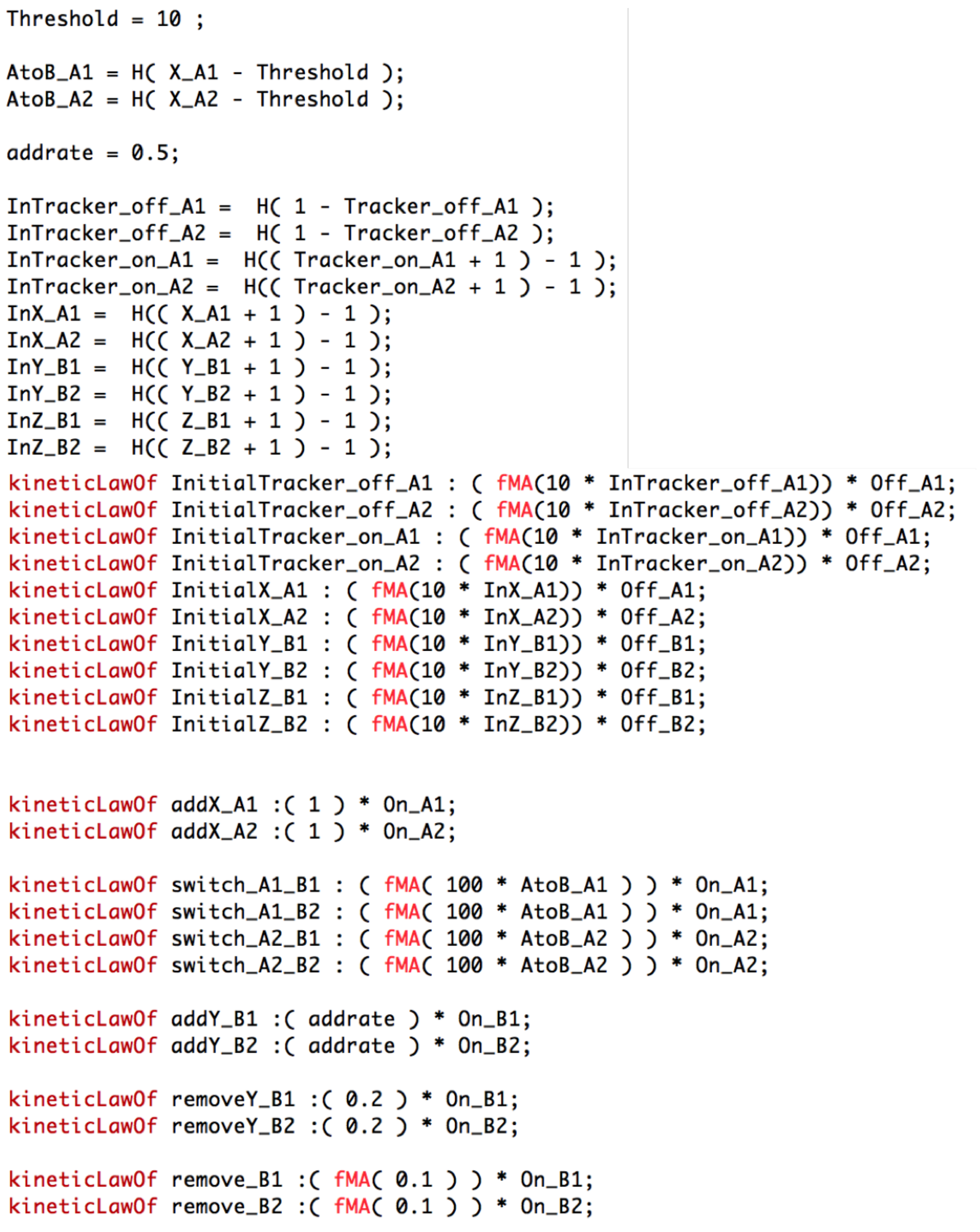
I present below the toy PAL model in PAL Parser file format in Figure B.1 and the output Bio-PEPA model in Figures B.2 and B.3 discussed in Section 4.2.2.2 of Chapter 4.

Also presented below are the psPAH and PEPA nets models discussed in Section 4.4 of Chapter 4. The psPAH model is given in Figures B.4, B.5, B.6, B.7 and B.8. The PEPA nets model is given in Figures B.9, B.10, B.11, B.12, B.13 and B.14.



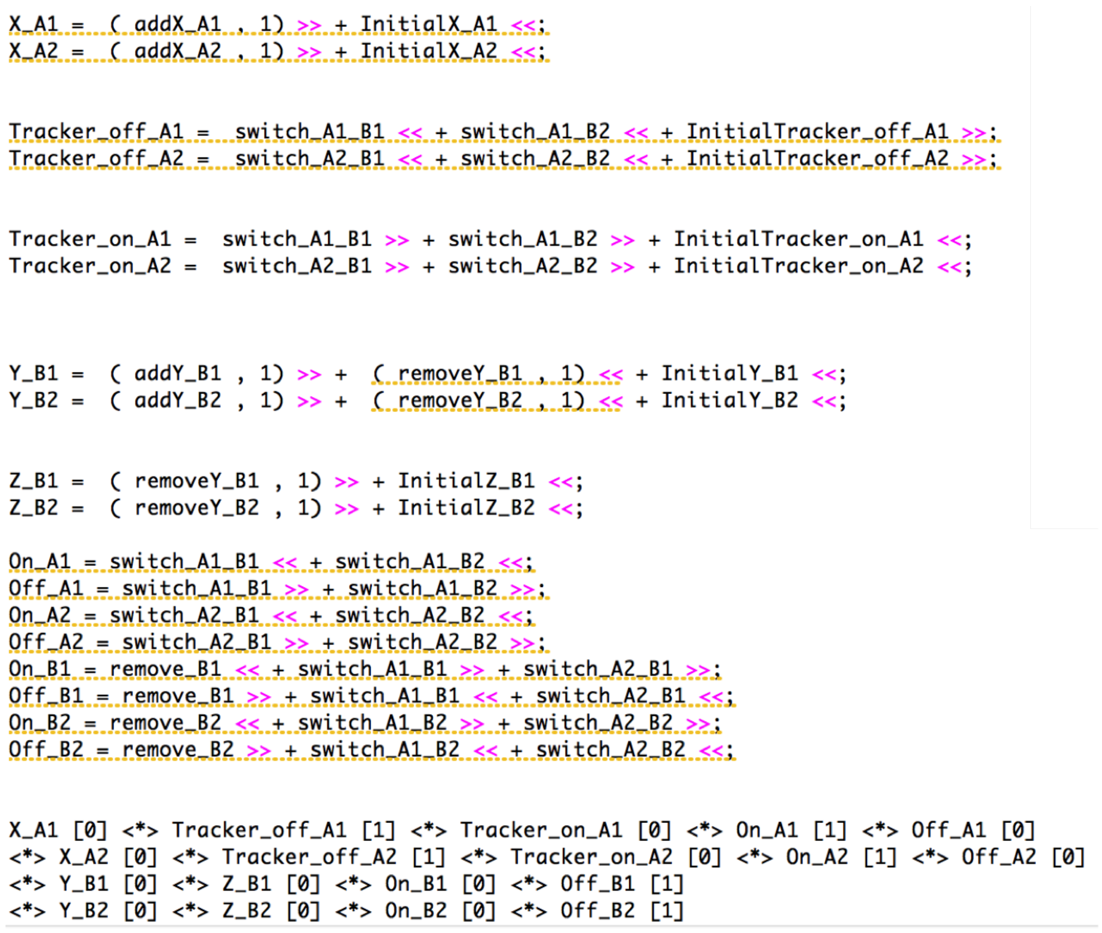
```
InitialReset X = H((X+1) - 1);
InitialReset Tracker_off = H(1 - Tracker_off );
InitialReset Tracker_on = H((Tracker_on + 1) - 1 );

InitialReset Y = H((Y+1) - 1);
InitialReset Z = H((Z+1) - 1);

Threshold = 10;
AtoB= H(X - Threshold);

addrate = 0.5;

addX : 1;
switch : fMA(100 * AtoB);

addY : addrate;
removeY : 0.2;

remove: 0.1;

X = (addX,1) >>;
Tracker_off = (switch, 1) <<;
Tracker_on = (switch, 1) >>;

Y = (addY,1) >> + (removeY,1) <<;
Z = (removeY,1) >>;

HiddenActions H : {addX,addY,removeY};

A = X[0]<*> Tracker_off[1] <*> Tracker_on[0]
B = Y[0]<*>Z[0]

One{{A}} = switch <<<;
Two{{B}} = switch >>> + remove <<<;

One{{A}}[[2]]{switch}Two{{B}}[[0]]

Max: One{{A}} 2, Two{{B}} 2;
```

Figure B.1: Toy PAL model in the PAL Parser file format.



```
Threshold = 10 ;

AtoB_A1 = H( X_A1 - Threshold );
AtoB_A2 = H( X_A2 - Threshold );

addrate = 0.5;

InTracker_off_A1 =  H( 1 - Tracker_off_A1 );
InTracker_off_A2 =  H( 1 - Tracker_off_A2 );
InTracker_on_A1 =  H(( Tracker_on_A1 + 1 ) - 1 );
InTracker_on_A2 =  H(( Tracker_on_A2 + 1 ) - 1 );
InX_A1 =  H(( X_A1 + 1 ) - 1 );
InX_A2 =  H(( X_A2 + 1 ) - 1 );
InY_B1 =  H(( Y_B1 + 1 ) - 1 );
InY_B2 =  H(( Y_B2 + 1 ) - 1 );
InZ_B1 =  H(( Z_B1 + 1 ) - 1 );
InZ_B2 =  H(( Z_B2 + 1 ) - 1 );
kineticLawOf InitialTracker_off_A1 : ( fMA(10 * InTracker_off_A1)) * Off_A1;
kineticLawOf InitialTracker_off_A2 : ( fMA(10 * InTracker_off_A2)) * Off_A2;
kineticLawOf InitialTracker_on_A1 : ( fMA(10 * InTracker_on_A1)) * Off_A1;
kineticLawOf InitialTracker_on_A2 : ( fMA(10 * InTracker_on_A2)) * Off_A2;
kineticLawOf InitialX_A1 : ( fMA(10 * InX_A1)) * Off_A1;
kineticLawOf InitialX_A2 : ( fMA(10 * InX_A2)) * Off_A2;
kineticLawOf InitialY_B1 : ( fMA(10 * InY_B1)) * Off_B1;
kineticLawOf InitialY_B2 : ( fMA(10 * InY_B2)) * Off_B2;
kineticLawOf InitialZ_B1 : ( fMA(10 * InZ_B1)) * Off_B1;
kineticLawOf InitialZ_B2 : ( fMA(10 * InZ_B2)) * Off_B2;


kineticLawOf addX_A1 :( 1 ) * On_A1;
kineticLawOf addX_A2 :( 1 ) * On_A2;

kineticLawOf switch_A1_B1 : ( fMA( 100 * AtoB_A1 ) ) * On_A1;
kineticLawOf switch_A1_B2 : ( fMA( 100 * AtoB_A1 ) ) * On_A1;
kineticLawOf switch_A2_B1 : ( fMA( 100 * AtoB_A2 ) ) * On_A2;
kineticLawOf switch_A2_B2 : ( fMA( 100 * AtoB_A2 ) ) * On_A2;

kineticLawOf addY_B1 :( addrate ) * On_B1;
kineticLawOf addY_B2 :( addrate ) * On_B2;

kineticLawOf removeY_B1 :( 0.2 ) * On_B1;
kineticLawOf removeY_B2 :( 0.2 ) * On_B2;

kineticLawOf remove_B1 :( fMA( 0.1 ) ) * On_B1;
kineticLawOf remove_B2 :( fMA( 0.1 ) ) * On_B2;
```

Figure B.2: Bio-PEPA output model from PAL Parser Part 1 of 2.



```
X_A1 =  ( addX_A1 , 1) >> + InitialX_A1 <<;
X_A2 =  ( addX_A2 , 1) >> + InitialX_A2 <<;


Tracker_off_A1 =  switch_A1_B1 << + switch_A1_B2 << + InitialTracker_off_A1 >>;
Tracker_off_A2 =  switch_A2_B1 << + switch_A2_B2 << + InitialTracker_off_A2 >>;


Tracker_on_A1 =  switch_A1_B1 >> + switch_A1_B2 >> + InitialTracker_on_A1 <<;
Tracker_on_A2 =  switch_A2_B1 >> + switch_A2_B2 >> + InitialTracker_on_A2 <<;



Y_B1 =  ( addY_B1 , 1) >> +  ( removeY_B1 , 1) << + InitialY_B1 <<;
Y_B2 =  ( addY_B2 , 1) >> +  ( removeY_B2 , 1) << + InitialY_B2 <<;


Z_B1 =  ( removeY_B1 , 1) >> + InitialZ_B1 <<;
Z_B2 =  ( removeY_B2 , 1) >> + InitialZ_B2 <<;

On_A1 = switch_A1_B1 << + switch_A1_B2 <<;
Off_A1 = switch_A1_B1 >> + switch_A1_B2 >>;
On_A2 = switch_A2_B1 << + switch_A2_B2 <<;
Off_A2 = switch_A2_B1 >> + switch_A2_B2 >>;
On_B1 = remove_B1 << + switch_A1_B1 >> + switch_A2_B1 >>;
Off_B1 = remove_B1 >> + switch_A1_B1 << + switch_A2_B1 <<;
On_B2 = remove_B2 << + switch_A1_B2 >> + switch_A2_B2 >>;
Off_B2 = remove_B2 >> + switch_A1_B2 << + switch_A2_B2 <<;


X_A1 [0] <*> Tracker_off_A1 [1] <*> Tracker_on_A1 [0] <*> On_A1 [1] <*> Off_A1 [0]
<*> X_A2 [0] <*> Tracker_off_A2 [1] <*> Tracker_on_A2 [0] <*> On_A2 [1] <*> Off_A2 [0]
<*> Y_B1 [0] <*> Z_B1 [0] <*> On_B1 [0] <*> Off_B1 [1]
<*> Y_B2 [0] <*> Z_B2 [0] <*> On_B2 [0] <*> Off_B2 [1]
```

Figure B.3: Bio-PEPA output model from PAL Parser Part 2 of 2.

Constants

InitialLE = 100	InitialJE = 200	InitialLV = 250
InitialJV = 260	InitialLER = 0	IntialJER = 100
SwitchThr = 260	ReproduceThr = 200	

Functional Rates

EnergyOn(i) = 1	EnergyOff(i) = 1	SwitchJEnergy(i) = 1
LA(i) = 0.5	LG(i) = 0.25	LR(i) = 0.25
JA(i) = 0.6	JG(i) = 0.27	JR(i) = 0.27
LSwitch(i) = 1	Switch(i) = 1	LS(i) = 0.25
JS(i) = 0.3	LJ(i) = 0.3	ReproduceL(i,j) = 1
Reproduce(i) = 1	JJ(i) = 0.35	InternalLDie(i) = 0.05
InternalJDie(i) = 0.05	LDie(i) = 1	JDie(i) = 1
RandomLDie(i) = 0.02	RandomJDie(i) = 0.06	RandomJAdd(i) = 0.02
JEnergyOn(i) = 1		

Agent Definitions (Organism Scale)

$$\begin{aligned}
NotInUse(i) = \;& Reproduce(1,i)[EnergyOn(i)].Larva(i)\\
&+Reproduce(2,i)[EnergyOn(i)].Larva(i)\\
&\dots\\
&+Reproduce(20,i)[EnergyOn(i)].Larva(i)\\
&+RandomJAdd(i)[JEnergyOn(i)].Juvenile(i)
\end{aligned}$$

$$\begin{aligned}
Larva(i) = \;& Switch(i)[SwitchJEnergy(i)].Juvenile(i)\\
&+RandomLDie(i)[EnergyOff(i)].NotInUse(i)\\
&+LDie(i).NotInUse(i)
\end{aligned}$$

$$\begin{aligned}
Juvenile(i) = \;& Reproduce(i,1).Juvenile(i)\\
&+Reproduce(i,2).Juvenile(i)\\
&\dots\\
&+Reproduce(i,20).Juvenile(i)\\
&+RandomJDie(i)[EnergyOff(i)].NotInUse(i)\\
&+JDie(i).NotInUse(i)
\end{aligned}$$

Figure B.4: psPAH model Part 1 of 5

Agent Definitions (Organism Physiological Scale)

$$\begin{aligned}
NE(i) =\ & EnergyOn(i).LE(i, initialLE) + JEnergyOn(i).JE(i, initialJE)\\
NV(i) =\ & EnergyOn(i).LV(i, \text{InitialLV}) + JEnergyOn(i).JV(i, initialJV)\\
NER(i) =\ & EnergyOn(i).LER(i, initialLER) + JEnergyOn(i).JER(i, initialJER)
\end{aligned}$$

$$\begin{aligned}
LE(i, w) =\ & EnergyOff(i).NE(i) + IntrenalLDie(i)[LDie].NE(i)\\
& +SwitchJEnergy(i).JE(i, \text{InitialJE}) + LA(i).LE(i, w + 1)\\
& +(\text{if w} > \text{0 then } LG(i).LE(i, w - 1) \text{ else nil})\\
& +(\text{if w} > \text{0 then } LR(i).LE(i, w - 1) \text{ else nil})
\end{aligned}$$

$$\begin{aligned}
JE(i, w) =\ & EnergyOff(i).NE(i) + IntrenalJDie(i)[JDie].NE(i) + JA(i).JE(i, w + 1)\\
& +(\text{if w} > \text{0 then } JG(i).JE(i, w - 1) \text{ else nil})\\
& +(\text{if w} > \text{0 then } JR(i).JE(i, w - 1) \text{ else nil})
\end{aligned}$$

$$\begin{aligned}
LV(i, w) =\ & EnergyOff(i).NV(i) + IntrenalLDie(i)[LDie].NV(i)\\
& +(\text{if w == SwitchThr then}\\
& \quad LSwitch(i)[Switch(i)].JV(i, \text{InitialJV})\\
& \quad \text{else } LG(i).LV(i, w - 1))\\
& +(\text{if w} > \text{0 then } LS(i).LV(i, w - 1) \text{ else nil})
\end{aligned}$$

$$\begin{aligned}
JV(i, w) =\ & EnergyOff(i).NV(i) + IntrenalJDie(i)[JDie].NV(i) + JG(i).JV(i, w + 1)\\
& +(\text{if w} > \text{0 then } JS(i).JV(i, w - 1) \text{ else nil})
\end{aligned}$$

$$\begin{aligned}
LER(i, w) =\ & EnergyOff(i).NER(i) + IntrenalLDie(i)[LDie].NER(i)\\
& +SwitchJEnergy(i).JER(i, intitialJER))\\
& +LR(i).LER(i, w + 1) + (\text{if w} > \text{0 then } LJ.LER(i, w - 1) \text{ else nil})
\end{aligned}$$

$$\begin{aligned}
JER(i, w) =\ & EnergyOff(i).NER(i) + IntrenalJDie(i)[JDie].NER(i)\\
& +(\text{if w == ReproduceThr then}\\
& \quad (ReproduceL(i)[Reproduce(i, 1)].JER(i, 0)\\
& \quad +ReproduceL(i)[Reproduce(i, 2)].JER(i, 0)\\
& \quad \ldots\\
& \quad +ReproduceL(i)[Reproduce(i, 20)].JER(i, 0))\\
& \quad \text{else } JR(i).JER(i, w + 1)\\
& +(\text{if w} > \text{0 then } JJ.JER(i, w - 1) \text{ else nil})
\end{aligned}$$

Figure B.5: psPAH model Part 2 of 5

Model Initial State

$$
\begin{aligned}
&(LV(1, IntialLV) \underset{L_1}{\bowtie} LE(1, InitialLE) \underset{L_1}{\bowtie} LER(1, InitialLER) \underset{\emptyset}{\bowtie} \\
&LV(2, IntialLV) \underset{L_2}{\bowtie} LE(2, InitialLE) \underset{L_2}{\bowtie} LER(2, InitialLER) \underset{\emptyset}{\bowtie} \\
&JV(3, IntialJV) \underset{L_3}{\bowtie} JE(3, InitialJE) \underset{L_3}{\bowtie} JER(3, InitialJER) \underset{\emptyset}{\bowtie} \\
&JV(4, IntialJV) \underset{L_4}{\bowtie} JE(4, InitialJE) \underset{L_4}{\bowtie} JER(4, InitialJER) \underset{\emptyset}{\bowtie} \\
&JV(5, IntialJV) \underset{L_5}{\bowtie} JE(5, InitialJE) \underset{L_5}{\bowtie} JER(5, InitialJER) \underset{\emptyset}{\bowtie} \\
&NV(6) \underset{L_6}{\bowtie} NE(6) \underset{L_6}{\bowtie} NER(6) \underset{\emptyset}{\bowtie} \\
&NV(7) \underset{L_7}{\bowtie} NE(7) \underset{L_7}{\bowtie} NER(7) \underset{\emptyset}{\bowtie} \cdots \underset{\emptyset}{\bowtie} \\
&NV(20) \underset{L_{20}}{\bowtie} NE(20) \underset{L_{20}}{\bowtie} NER(20) \underset{H}{\bowtie} \\
&Larva(1) \underset{N_1}{\bowtie} Larva(2) \underset{N_2}{\bowtie} Juvenile(3) \underset{N_3}{\bowtie} \\
&Juvenile(4) \underset{N_4}{\bowtie} Juvenile(5) \underset{N_5}{\bowtie} NotInUse(6) \underset{N_6}{\bowtie} \\
&NotInUse(7) \underset{N_7}{\bowtie} \cdots \underset{N_{19}}{\bowtie} NotInUse(20))
\end{aligned}
$$

$$
\begin{aligned}
L_1 = \{&LG(1), JG(1), LR(1), JR(1), EnergyOn(1), EnergyOff(1), SwitchJEnergy(1), \\
&InternalLDie(1), InternalLDie(1), JEnergyOn(1)\} \\
L_2 = \{&LG(2), JG(2), LR(2), JR(2), EnergyOn(2), EnergyOff(2), SwitchJEnergy(2), \\
&InternalLDie(2), InternalLDie(2), JEnergyOn(2)\} \\
L_3 = \{&LG(3), JG(3), LR(3), JR(3), EnergyOn(3), EnergyOff(3), SwitchJEnergy(3), \\
&InternalLDie(3), InternalLDie(3), JEnergyOn(3)\} \\
\ldots \\
L_{20} = \{&LG(20), JG(20), LR(20), JR(20), EnergyOn(20), EnergyOff(20), SwitchJEnergy(20), \\
&InternalLDie(20), InternalLDie(20), JEnergyOn(20)\}
\end{aligned}
$$

Figure B.6: psPAH model Part 3 of 5.

Model Initial State continued

$$
\begin{aligned}
H = \{ & EnergyOn(1), EnergyOff(1), LDie(1), JDie(1), SwitchJEnergy(1), Switch(1), JEnergyOn(1),\\
& Reproduce(1,2), Reproduce(1,3), \ldots, Reproduce(1,20), EnergyOn(2), EnergyOff(2), LDie(2), JDie(2),\\
& SwitchJEnergy(2), Switch(2), JEnergyOn(2), Reproduce(2,1), Reproduce(2,3), \ldots, Reproduce(2,20),\\
& EnergyOn(3), EnergyOff(3), LDie(3), JDie(3), SwitchJEnergy(3), Switch(3), JEnergyOn(3),\\
& Reproduce(3,1), Reproduce(3,2), \ldots, Reproduce(3,20), EnergyOn(4), EnergyOff(4), LDie(4), JDie(4),\\
& SwitchJEnergy(4), Switch(4), JEnergyOn(4), Reproduce(4,1), Reproduce(4,2), \ldots, Reproduce(4,20),\\
& EnergyOn(5), EnergyOff(5), LDie(5), JDie(5), SwitchJEnergy(5), Switch(5), JEnergyOn(5),\\
& Reproduce(5,1), Reproduce(5,2), \ldots, Reproduce(5,20), EnergyOn(6), EnergyOff(6), LDie(6), JDie(6),\\
& SwitchJEnergy(6), Switch(6), JEnergyOn(6), Reproduce(6,1), Reproduce(6,2), \ldots, Reproduce(6,20),\\
& EnergyOn(7), EnergyOff(7), LDie(7), JDie(7), SwitchJEnergy(7), Switch(7), JEnergyOn(7),\\
& Reproduce(7,1), Reproduce(7,3), \ldots, Reproduce(7,20), EnergyOn(8), EnergyOff(8), LDie(8), JDie(8),\\
& SwitchJEnergy(8), Switch(8), JEnergyOn(8), Reproduce(8,1), Reproduce(8,2), \ldots, Reproduce(8,20),\\
& EnergyOn(9), EnergyOff(9), LDie(9), JDie(9), SwitchJEnergy(9), Switch(9), JEnergyOn(9),\\
& Reproduce(9,1), Reproduce(9,2), \ldots, Reproduce(9,20), EnergyOn(10), EnergyOff(10), LDie(10), JDie(10),\\
& SwitchJEnergy(10), Switch(10), JEnergyOn(10), Reproduce(10,1), Reproduce(10,2), \ldots, Reproduce(10,20),\\
& EnergyOn(11), EnergyOff(11), LDie(11), JDie(11), SwitchJEnergy(11), Switch(11), JEnergyOn(11),\\
& Reproduce(11,1), Reproduce(11,2), \ldots, Reproduce(11,20), EnergyOn(12), EnergyOff(12), LDie(12), JDie(12),\\
& SwitchJEnergy(12), Switch(12), JEnergyOn(12), Reproduce(12,1), Reproduce(12,3), \ldots, Reproduce(12,20),\\
& EnergyOn(13), EnergyOff(13), LDie(13), JDie(13), SwitchJEnergy(13), Switch(13), JEnergyOn(13),\\
& Reproduce(13,1), Reproduce(13,2), \ldots, Reproduce(13,20), EnergyOn(14), EnergyOff(14), LDie(14), JDie(14),\\
& SwitchJEnergy(14), Switch(14), JEnergyOn(14), Reproduce(14,1), Reproduce(14,2), \ldots, Reproduce(14,20),\\
& EnergyOn(15), EnergyOff(15), LDie(15), JDie(15), SwitchJEnergy(15), Switch(15), JEnergyOn(15),\\
& Reproduce(15,1), Reproduce(15,2), \ldots, Reproduce(15,20), EnergyOn(16), EnergyOff(16), LDie(17), JDie(17),\\
& SwitchJEnergy(16), Switch(16), JEnergyOn(16), Reproduce(16,1), Reproduce(16,2), \ldots, Reproduce(16,20),\\
& EnergyOn(17), EnergyOff(17), LDie(18), JDie(18), SwitchJEnergy(17), Switch(17), JEnergyOn(17),\\
& Reproduce(17,1), Reproduce(17,3), \ldots, Reproduce(17,20), EnergyOn(18), EnergyOff(18), LDie(18), JDie(18),\\
& SwitchJEnergy(18), Switch(18), JEnergyOn(18), Reproduce(18,1), Reproduce(18,2), \ldots, Reproduce(18,20),\\
& EnergyOn(19), EnergyOff(19), LDie(19), JDie(19), SwitchJEnergy(19), Switch(19), JEnergyOn(19),\\
& Reproduce(19,1), Reproduce(19,2), \ldots, Reproduce(19,20), EnergyOn(20), EnergyOff(20), LDie(20), JDie(20),\\
& SwitchJEnergy(20), Switch(20), JEnergyOn(20), Reproduce(20,1), Reproduce(20,2), \ldots, Reproduce(20,19)\}
\end{aligned}
$$

Figure B.7: psPAH model Part 4 of 5. Note there are no vertical synchronisations on Reproduce(i,j) action where i == j.

Model Initial State continued

$$\begin{aligned}
N_1, ..., N_{19} = \{&\mathrm{Reproduce}(1,2), \mathrm{Reproduce}(1,3), ..., \mathrm{Reproduce}(1,20),\\
&\mathrm{Reproduce}(2,1), \mathrm{Reproduce}(2,3), ..., \mathrm{Reproduce}(2,20),\\
&\mathrm{Reproduce}(3,1), \mathrm{Reproduce}(3,2), ..., \mathrm{Reproduce}(3,20),\\
&\mathrm{Reproduce}(4,1), \mathrm{Reproduce}(4,2), ..., \mathrm{Reproduce}(4,20),\\
&\mathrm{Reproduce}(5,1), \mathrm{Reproduce}(5,2), ..., \mathrm{Reproduce}(5,20),\\
&\mathrm{Reproduce}(6,1), \mathrm{Reproduce}(6,2), ..., \mathrm{Reproduce}(6,20),\\
&\mathrm{Reproduce}(7,1), \mathrm{Reproduce}(7,3), ..., \mathrm{Reproduce}(7,20),\\
&\mathrm{Reproduce}(8,1), \mathrm{Reproduce}(8,2), ..., \mathrm{Reproduce}(8,20),\\
&\mathrm{Reproduce}(9,1), \mathrm{Reproduce}(9,2), ..., \mathrm{Reproduce}(9,20),\\
&\mathrm{Reproduce}(10,1), \mathrm{Reproduce}(10,2), ..., \mathrm{Reproduce}(10,20),\\
&\mathrm{Reproduce}(11,1), \mathrm{Reproduce}(11,2), ..., \mathrm{Reproduce}(11,20),\\
&\mathrm{Reproduce}(12,1), \mathrm{Reproduce}(12,3), ..., \mathrm{Reproduce}(12,20),\\
&\mathrm{Reproduce}(13,1), \mathrm{Reproduce}(13,2), ..., \mathrm{Reproduce}(13,20),\\
&\mathrm{Reproduce}(14,1), \mathrm{Reproduce}(14,2), ..., \mathrm{Reproduce}(14,20),\\
&\mathrm{Reproduce}(15,1), \mathrm{Reproduce}(15,2), ..., \mathrm{Reproduce}(15,20),\\
&\mathrm{Reproduce}(16,1), \mathrm{Reproduce}(16,2), ..., \mathrm{Reproduce}(16,20),\\
&\mathrm{Reproduce}(17,1), \mathrm{Reproduce}(17,3), ..., \mathrm{Reproduce}(17,20),\\
&\mathrm{Reproduce}(18,1), \mathrm{Reproduce}(18,2), ..., \mathrm{Reproduce}(18,20),\\
&\mathrm{Reproduce}(19,1), \mathrm{Reproduce}(19,2), ..., \mathrm{Reproduce}(19,20),\\
&\mathrm{Reproduce}(20,1), \mathrm{Reproduce}(20,2), ..., \mathrm{Reproduce}(20,19)\}
\end{aligned}$$

Figure B.8: psPAH model Part 5 of 5. Note there are no horizontal synchronisations on Reproduce(i,j) action where i == j.

PEPA context definitions

$$L_1[O] = Organism[O] \underset{\{LVOn,LSwitch,InternalDie\}}{\bowtie} (LVOff_2 \underset{\{LEOff,LEOn,LG\}}{\bowtie} LEOff_2 \underset{\{LEROff,LEROn,LR\}}{\bowtie} LEROff_1)$$

$$L_2[O] = Organism[O] \underset{\{LVOn,LSwitch,InternalDie\}}{\bowtie} (LVOff_2 \underset{\{LEOff,LEOn,LG\}}{\bowtie} LEOff_2 \underset{\{LEROff,LEROn,LR\}}{\bowtie} LEROff_1)$$

$$L_3[O] = Organism[O] \underset{\{LVOn,LSwitch,InternalDie\}}{\bowtie} (LVOff_2 \underset{\{LEOff,LEOn,LG\}}{\bowtie} LEOff_2 \underset{\{LEROff,LEROn,LR\}}{\bowtie} LEROff_1)$$

$$L_4[O] = Organism[O] \underset{\{LVOn,LSwitch,InternalDie\}}{\bowtie} (LVOff_2 \underset{\{LEOff,LEOn,LG\}}{\bowtie} LEOff_2 \underset{\{LEROff,LEROn,LR\}}{\bowtie} LEROff_1)$$

$$L_5[O] = Organism[O] \underset{\{LVOn,LSwitch,InternalDie\}}{\bowtie} (LVOff_2 \underset{\{LEOff,LEOn,LG\}}{\bowtie} LEOff_2 \underset{\{LEROff,LEROn,LR\}}{\bowtie} LEROff_1)$$

$$L_6[O] = Organism[O] \underset{\{LVOn,LSwitch,InternalDie\}}{\bowtie} (LVOff_2 \underset{\{LEOff,LEOn,LG\}}{\bowtie} LEOff_2 \underset{\{LEROff,LEROn,LR\}}{\bowtie} LEROff_1)$$

$$L_7[O] = Organism[O] \underset{\{LVOn,LSwitch,InternalDie\}}{\bowtie} (LVOff_2 \underset{\{LEOff,LEOn,LG\}}{\bowtie} LEOff_2 \underset{\{LEROff,LEROn,LR\}}{\bowtie} LEROff_1)$$

$$L_8[O] = Organism[O] \underset{\{LVOn,LSwitch,InternalDie\}}{\bowtie} (LVOff_2 \underset{\{LEOff,LEOn,LG\}}{\bowtie} LEOff_2 \underset{\{LEROff,LEROn,LR\}}{\bowtie} LEROff_1)$$

$$L_9[O] = Organism[O] \underset{\{LVOn,LSwitch,InternalDie\}}{\bowtie} (LVOff_2 \underset{\{LEOff,LEOn,LG\}}{\bowtie} LEOff_2 \underset{\{LEROff,LEROn,LR\}}{\bowtie} LEROff_1)$$

$$L_{10}[O] = Organism[O] \underset{\{LVOn,LSwitch,InternalDie\}}{\bowtie} (LVOff_2 \underset{\{LEOff,LEOn,LG\}}{\bowtie} LEOff_2 \underset{\{LEROff,LEROn,LR\}}{\bowtie} LEROff_1)$$

$$J_1[O] = Organism[O] \underset{\{JEROn,ReproduceL,InternalDie\}}{\bowtie} (JEROff_2 \underset{\{JEOff,JEOn,JR\}}{\bowtie} JEOff_1 \underset{\{JVOff,JVOn,JG\}}{\bowtie} JVOff_1)$$

$$J_2[O] = Organism[O] \underset{\{JEROn,ReproduceL,InternalDie\}}{\bowtie} (JEROff_2 \underset{\{JEOff,JEOn,JR\}}{\bowtie} JEOff_1 \underset{\{JVOff,JVOn,JG\}}{\bowtie} JVOff_1)$$

$$J_3[O] = Organism[O] \underset{\{JEROn,ReproduceL,InternalDie\}}{\bowtie} (JEROff_2 \underset{\{JEOff,JEOn,JR\}}{\bowtie} JEOff_1 \underset{\{JVOff,JVOn,JG\}}{\bowtie} JVOff_1)$$

$$J_4[O] = Organism[O] \underset{\{JEROn,ReproduceL,InternalDie\}}{\bowtie} (JEROff_2 \underset{\{JEOff,JEOn,JR\}}{\bowtie} JEOff_1 \underset{\{JVOff,JVOn,JG\}}{\bowtie} JVOff_1)$$

$$J_5[O] = Organism[O] \underset{\{JEROn,ReproduceL,InternalDie\}}{\bowtie} (JEROff_2 \underset{\{JEOff,JEOn,JR\}}{\bowtie} JEOff_1 \underset{\{JVOff,JVOn,JG\}}{\bowtie} JVOff_1)$$

$$J_6[O] = Organism[O] \underset{\{JEROn,ReproduceL,InternalDie\}}{\bowtie} (JEROff_2 \underset{\{JEOff,JEOn,JR\}}{\bowtie} JEOff_1 \underset{\{JVOff,JVOn,JG\}}{\bowtie} JVOff_1)$$

$$J_7[O] = Organism[O] \underset{\{JEROn,ReproduceL,InternalDie\}}{\bowtie} (JEROff_2 \underset{\{JEOff,JEOn,JR\}}{\bowtie} JEOff_1 \underset{\{JVOff,JVOn,JG\}}{\bowtie} JVOff_1)$$

$$J_8[O] = Organism[O] \underset{\{JEROn,ReproduceL,InternalDie\}}{\bowtie} (JEROff_2 \underset{\{JEOff,JEOn,JR\}}{\bowtie} JEOff_1 \underset{\{JVOff,JVOn,JG\}}{\bowtie} JVOff_1)$$

$$J_9[O] = Organism[O] \underset{\{JEROn,ReproduceL,InternalDie\}}{\bowtie} (JEROff_2 \underset{\{JEOff,JEOn,JR\}}{\bowtie} JEOff_1 \underset{\{JVOff,JVOn,JG\}}{\bowtie} JVOff_1)$$

$$J_{10}[O] = Organism[O] \underset{\{JEROn,ReproduceL,InternalDie\}}{\bowtie} (JEROff_2 \underset{\{JEOff,JEOn,JR\}}{\bowtie} JEOff1 \underset{\{JVOff,JVOn,JG\}}{\bowtie} JVOff_1)$$

$$RJ_1[O] = Organism[O]$$

$$RJ_2[O] = Organism[O]$$

$$RJ_3[O] = Organism[O]$$

$$RJ_4[O] = Organism[O]$$

$$RJ_5[O] = Organism[O]$$

Initial marking of the net

$$(L_1[Organism_b], L_2[Organism_b], L_3[\_], L_4[\_], L_5[\_], L_6[\_], L_7[\_], L_8[\_], L_9[\_], L_{10}[\_],$$

$$J_1[Organism_e], J_2[Organism_e], J_3[Organism_e], J_4[\_], J_5[\_], J_6[\_], J_7[\_], J_8[\_], J_9[\_], J_{10}[\_],$$

$$RJ_1[Organism_{RandomJ}], RJ_2[Organism_{RandomJ}],$$

$$RJ_3[Organism_{RandomJ}], RJ_4[Organism_{RandomJ}], RJ_5[Organism_{RandomJ}])$$

Figure B.9: PEPA nets model Part 1 of 6.

Rates

$\lambda = 0.9 \;\; \mu = 0.9 \;\; \nu = 0.02$

$r_a = 0.9 \;\; r_b = 0.9 \;\; r_c = 0.25 \;\; r_d = 0.5 \;\; r_e = 0.25 \;\; r_f = 0.25$

$r_g = 0.3 \;\; r_h = 0.35 \;\; r_i = 0.6 \;\; r_j = 0.27 \;\; r_k = 0.27 \;\; r_l = 0.3$

$r_m = 0.02 \;\; r_n = 0.06 \;\; r_o = 0.1 \;\; r_1 = 0.9 \;\; r_2 = 0.9 \;\; r_3 = 1$

$r_4 = 1 \;\; r_5 = 1 \;\; r_6 = 1 \;\; r_7 = 1 \;\; r_8 = 1 \;\; r_9 = 1 \;\; r_{10} = 1$

PEPA definitions: Larva Static components

$LVOff_1 \;=\; (LEOff, r_3).LVOff_2$

$LVOff_2 \;=\; (LVOn, \tau).LVOn$

$LVOn \;=\; (LEOn, r_5).LVLow$

$LVZero \;=\; (LG, \tau).LVLow + (InternalDie, r_p).LVOff_1$

$LVLow \;=\; (LG, \tau).LVMedium + (LS, r_c).LVZero + (InternalDie, r_p).LVOff_1$

$LVMedium \;=\; (LSwitch, \tau).LVOff_1 + (LS, r_c).LVLow + (InternalDie, r_p).LVOff_1$

$LEOff_1 \;=\; (LEROff, r_4).LEOff_2$

$LEOff_2 \;=\; (LEOn, \tau).LEOn$

$LEOn \;=\; (LEROn, r_6).LELow$

$LEZero \;=\; (LA, r_d).LELow + (LEOff, \tau).LEOff_1 + (InternalDie, r_p).LEOff_1$

$LELow \;=\; (LA, r_d).LEMedium + (LG, r_e).LEZero + (LR, r_f).LEZero + (LEOff, \tau).LEOff_1 + (InternalDie, r_p).LEOff_1$

$LEMedium \;=\; (LA, r_d).LEHigh + (LG, r_e).LELow + (LR, r_f).LELow + (LEOff, \tau).LEOff_1 + (InternalDie, r_p).LEOff_1$

$LEHigh \;=\; (LG, r_e).LEMedium + (LR, r_f).LEMedium + (LEOff, \tau).LEOff_1 + (InternalDie, r_p).LEOff_1$

$LEROff_1 \;=\; (LEROn, \tau).LERZero$

$LERZero \;=\; (LR, \tau).LERLow + (LEROff, \tau).LEROff_1 + (InternalDie, r_p).LEROff_1$

$LERLow \;=\; (LR, \tau).LERMedium + (LJ, r_g).LERZero + (LEROff, \tau).LEROff_1 + (InternalDie, r_p).LEROff_1$

$LERMedium \;=\; (LR, \tau).LERHigh + (LJ, r_g).LERLow + (LEROff, \tau).LEROff_1 + (InternalDie, r_p).LEROff_1$

$LERHigh \;=\; (LJ, r_g).LERMedium + (LEROff, \tau).LEROff_1 + (InternalDie, r_p).LEROff_1$

Figure B.10: PEPA nets model Part 2 of 6.

PEPA definitions: Juvenile Static components

$\mathrm{JEROff}_1 \; = \; (\mathrm{JEROff}, r_7).\mathrm{JEROff}_2$

$\mathrm{JEROff}_2 \; = \; (\mathrm{JEROn}, \tau).\mathrm{JEROn}$

$\mathrm{JEROn} \; = \; (\mathrm{JEOn}, r_8).\mathrm{JERLow}$

$\mathrm{JERZero} \; = \; (\mathrm{JR}, \tau).\mathrm{JERLow} + (\mathrm{InternalDie}, r_p).\mathrm{JEROff}_1$

$\mathrm{JERLow} \; = \; (\mathrm{JR}, \tau).\mathrm{JERMedium} + (\mathrm{JJ}, r_h).\mathrm{JERZero} + (\mathrm{InternalDie}, r_p).\mathrm{JEROff}_1$

$\mathrm{JERMedium} \; = \; (\mathrm{JR}, \tau).\mathrm{JERHigh} + (\mathrm{JJ}, r_h).\mathrm{JERLow} + (\mathrm{InternalDie}, r_p).\mathrm{JEROff}_1$

$\mathrm{JERHigh} \; = \; (\mathrm{Reproduce}, \tau).\mathrm{JEROff}_1 + (\mathrm{JJ}, r_h).\mathrm{JER}_M\mathrm{edium} + (\mathrm{InternalDie}, r_p).\mathrm{JEROff}_1$

$\mathrm{JEOff}_1 \; = \; (\mathrm{JVOff}, r_9).\mathrm{JEOff}_2$

$\mathrm{JEOff}_2 \; = \; (\mathrm{JEOn}, \tau).\mathrm{JEOn}$

$\mathrm{JEOn} \; = \; (\mathrm{JVOn}, r_{10}).\mathrm{JEMedium}$

$\mathrm{JEZero} \; = \; (\mathrm{JA}, r_i).\mathrm{JELow} + (\mathrm{JEOff}, \tau).\mathrm{JEOff}_1 + (\mathrm{InternalDie}, r_p).\mathrm{JEOff}_1$

$\mathrm{JELow} \; = \; (\mathrm{JA}, r_i).\mathrm{JEMedium} + (\mathrm{JG}, r_j).\mathrm{JEZero} + (\mathrm{JR}, r_k).\mathrm{JEZero} + (\mathrm{JEOff}, \tau).\mathrm{JEOff}_1 + (\mathrm{InternalDie}, r_p).\mathrm{JEOff}_1$

$\mathrm{JEMedium} \; = \; (\mathrm{JA}, r_i).\mathrm{JEHigh} + (\mathrm{JG}, r_j).\mathrm{JELow} + (\mathrm{JR}, r_k).\mathrm{JELow} + (\mathrm{JEOff}, \tau).\mathrm{JEOff}_1 + (\mathrm{InternalDie}, r_p).\mathrm{JEOff}_1$

$\mathrm{JEHigh} \; = \; (\mathrm{JG}, r_i).\mathrm{JEMedium} + (\mathrm{JR}, r_j).\mathrm{JEMedium} + (\mathrm{JEOff}, \tau).\mathrm{JEOff}_1 + (\mathrm{InternalDie}, r_p).\mathrm{JEOff}_1$

$\mathrm{JVOff}_1 \; = \; (\mathrm{JVOn}, \tau).\mathrm{JVMedium}$

$\mathrm{JVZero} \; = \; (\mathrm{JG}, \tau).\mathrm{JVLow} + (\mathrm{JVOff}, \tau).\mathrm{JVOff}_1 + (\mathrm{InternalDie}, r_p).\mathrm{JVOff}_1$

$\mathrm{JVLow} \; = \; (\mathrm{JG}, \tau).\mathrm{JVMedium} + (\mathrm{JS}, r_l).\mathrm{JVZero} + (\mathrm{JVOff}, \tau).\mathrm{JVOff}_1 + (\mathrm{InternalDie}, r_p).\mathrm{JVOff}_1$

$\mathrm{JVMedium} \; = \; (\mathrm{JG}, \tau).\mathrm{JVHigh} + (\mathrm{JS}, r_l).\mathrm{JVLow} + (\mathrm{JVOff}, \tau).\mathrm{JVOff}_1 + (\mathrm{InternalDie}, r_p).\mathrm{JVOff}_1$

$\mathrm{JVHigh} \; = \; (\mathrm{JS}, r_l).\mathrm{JV}_M\mathrm{edium} + (\mathrm{JVOff}, \tau).\mathrm{JVOff}_1 + (\mathrm{InternalDie}, r_p).\mathrm{JVOff}_1$

PEPA definitions: Dynamic component

$\mathrm{Organism}_a \; = \; (\mathrm{LVOn}, r_1).\mathrm{Organism}_b \; + \; (\mathrm{DieL}, r_m).\mathrm{OrganismDead} \; + \; (\mathrm{InternalDie}, \tau).\mathrm{OrganismDead}$

$\mathrm{Organism}_b \; = \; (\mathrm{LSwitch}, r_a).\mathrm{Organism}_c \; + \; (\mathrm{DieL}, r_m).\mathrm{OrganismDead} \; + \; (\mathrm{InternalDie}, \tau).\mathrm{OrganismDead}$

$\mathrm{Organism}_c \; = \; (\mathbf{Switch}, \lambda).\mathrm{Organism}_d \; + \; (\mathrm{DieL}, r_m).\mathrm{OrganismDead} \; + \; (\mathrm{InternalDie}, \tau).\mathrm{OrganismDead}$

$\mathrm{Organism}_d \; = \; (\mathrm{JEROn}, r_2).\mathrm{Organism}_e \; + \; (\mathrm{DieJ}, r_n).\mathrm{OrganismDead} \; + \; (\mathrm{InternalDie}, \tau).\mathrm{OrganismDead}$

$\mathrm{Organism}_e \; = \; (\mathrm{ReproduceL}, r_b).\mathrm{Organism}_f \; + \; (\mathrm{DieJ}, r_n).\mathrm{OrganismDead} \; + \; (\mathrm{InternalDie}, \tau).\mathrm{OrganismDead}$

$\mathrm{Organism}_f \; = \; (\mathbf{Reproduce}, \mu).\mathrm{Organism}_a \; + \; (\mathrm{DieJ}, r_n).\mathrm{OrganismDead} \; + \; (\mathrm{InternalDie}, \tau).\mathrm{OrganismDead}$

$\mathrm{OrganismDead} \; = \; (\mathrm{Dead}, r_o).\mathrm{OrganismDead}$

$\mathrm{Organism}_{RandomJ} \; = \; (\mathbf{AddJ}, \nu).\mathrm{Organism}_d$

Figure B.11: PEPA nets model Part 3 of 6.

Arcs of the Net

$L_1 - (Switch, \lambda) > - J_1$ $L_2 - (Switch, \lambda) > - J_1$ $L_3 - (Switch, \lambda) > - J_1$ $L_4 - (Switch, \lambda) > - J_1$

$L_5 - (Switch, \lambda) > - J_1$ $L_6 - (Switch, \lambda) > - J_1$ $L_7 - (Switch, \lambda) > - J_1$ $L_8 - (Switch, \lambda) > - J_1$

$L_9 - (Switch, \lambda) > - J_1$ $L_{10} - (Switch, \lambda) > - J_1$ $L_1 - (Switch, \lambda) > - J_2$ $L_2 - (Switch, \lambda) > - J_2$

$L_3 - (Switch, \lambda) > - J_2$ $L_4 - (Switch, \lambda) > - J_2$ $L_5 - (Switch, \lambda) > - J_2$ $L_6 - (Switch, \lambda) > - J_2$

$L_7 - (Switch, \lambda) > - J_2$ $L_8 - (Switch, \lambda) > - J_2$ $L_9 - (Switch, \lambda) > - J_2$ $L_{10} - (Switch, \lambda) > - J_2$

$L_1 - (Switch, \lambda) > - J_3$ $L_2 - (Switch, \lambda) > - J_3$ $L_3 - (Switch, \lambda) > - J_3$ $L_4 - (Switch, \lambda) > - J_3$

$L_5 - (Switch, \lambda) > - J_3$ $L_6 - (Switch, \lambda) > - J_3$ $L_7 - (Switch, \lambda) > - J_3$ $L_8 - (Switch, \lambda) > - J_3$

$L_9 - (Switch, \lambda) > - J_3$ $L_{10} - (Switch, \lambda) > - J_3$ $L_1 - (Switch, \lambda) > - J_4$ $L_2 - (Switch, \lambda) > - J_4$

$L_3 - (Switch, \lambda) > - J_4$ $L_4 - (Switch, \lambda) > - J_4$ $L_5 - (Switch, \lambda) > - J_4$ $L_6 - (Switch, \lambda) > - J_4$

$L_7 - (Switch, \lambda) > - J_4$ $L_8 - (Switch, \lambda) > - J_4$ $L_9 - (Switch, \lambda) > - J_4$ $L_{10} - (Switch, \lambda) > - J_4$

$L_1 - (Switch, \lambda) > - J_5$ $L_2 - (Switch, \lambda) > - J_5$ $L_3 - (Switch, \lambda) > - J_5$ $L_4 - (Switch, \lambda) > - J_5$

$L_5 - (Switch, \lambda) > - J_5$ $L_6 - (Switch, \lambda) > - J_5$ $L_7 - (Switch, \lambda) > - J_5$ $L_8 - (Switch, \lambda) > - J_5$

$L_9 - (Switch, \lambda) > - J_5$ $L_{10} - (Switch, \lambda) > - J_5$ $L_1 - (Switch, \lambda) > - J_6$ $L_2 - (Switch, \lambda) > - J_6$

$L_3 - (Switch, \lambda) > - J_6$ $L_4 - (Switch, \lambda) > - J_6$ $L_5 - (Switch, \lambda) > - J_6$ $L_6 - (Switch, \lambda) > - J_6$

$L_7 - (Switch, \lambda) > - J_6$ $L_8 - (Switch, \lambda) > - J_6$ $L_9 - (Switch, \lambda) > - J_6$ $L_{10} - (Switch, \lambda) > - J_6$

$L_1 - (Switch, \lambda) > - J_7$ $L_2 - (Switch, \lambda) > - J_7$ $L_3 - (Switch, \lambda) > - J_7$ $L_4 - (Switch, \lambda) > - J_7$

$L_5 - (Switch, \lambda) > - J_7$ $L_6 - (Switch, \lambda) > - J_7$ $L_7 - (Switch, \lambda) > - J_7$ $L_8 - (Switch, \lambda) > - J_7$

$L_9 - (Switch, \lambda) > - J_7$ $L_{10} - (Switch, \lambda) > - J_7$ $L_1 - (Switch, \lambda) > - J_8$ $L_2 - (Switch, \lambda) > - J_8$

$L_3 - (Switch, \lambda) > - J_8$ $L_4 - (Switch, \lambda) > - J_8$ $L_5 - (Switch, \lambda) > - J_8$ $L_6 - (Switch, \lambda) > - J_8$

$L_7 - (Switch, \lambda) > - J_8$ $L_8 - (Switch, \lambda) > - J_8$ $L_9 - (Switch, \lambda) > - J_8$ $L_{10} - (Switch, \lambda) > - J_8$

$L_1 - (Switch, \lambda) > - J_9$ $L_2 - (Switch, \lambda) > - J_9$ $L_3 - (Switch, \lambda) > - J_9$ $L_4 - (Switch, \lambda) > - J_9$

$L_5 - (Switch, \lambda) > - J_9$ $L_6 - (Switch, \lambda) > - J_9$ $L_7 - (Switch, \lambda) > - J_9$ $L_8 - (Switch, \lambda) > - J_9$

$L_9 - (Switch, \lambda) > - J_9$ $L_{10} - (Switch, \lambda) > - J_9$ $L_1 - (Switch, \lambda) > - J_{10}$ $L_2 - (Switch, \lambda) > - J_{10}$

$L_3 - (Switch, \lambda) > - J_{10}$ $L_4 - (Switch, \lambda) > - J_{10}$ $L_5 - (Switch, \lambda) > - J_{10}$ $L_6 - (Switch, \lambda) > - J_{10}$

$L_7 - (Switch, \lambda) > - J_{10}$ $L_8 - (Switch, \lambda) > - J_{10}$ $L_9 - (Switch, \lambda) > - J_{10}$ $L_{10} - (Switch, \lambda) > - J_{10}$

Figure B.12: PEPA nets model Part 4 of 6.

Arcs of the Net continued

$J_1 -(Reproduce, \mu) >- L_1$ $J_2 -(Reproduce, \mu) >- L_1$ $J_3 -(Reproduce, \mu) >- L_1$

$J_4 -(Reproduce, \mu) >- L_1$ $J_5 -(Reproduce, \mu) >- L_1$ $J_6 -(Reproduce, \mu) >- L_1$

$J_7 -(Reproduce, \mu) >- L_1$ $J_8 -(Reproduce, \mu) >- L_1$ $J_9 -(Reproduce, \mu) >- L_1$

$J_{10} -(Reproduce, \mu) >- L_1$ $J_1 -(Reproduce, \mu) >- L_2$ $J_2 -(Reproduce, \mu) >- L_2$

$J_3 -(Reproduce, \mu) >- L_2$ $J_4 -(Reproduce, \mu) >- L_2$ $J_5 -(Reproduce, \mu) >- L_2$

$J_6 -(Reproduce, \mu) >- L_2$ $J_7 -(Reproduce, \mu) >- L_2$ $J_8 -(Reproduce, \mu) >- L_2$

$J_9 -(Reproduce, \mu) >- L_2$ $J_{10} -(Reproduce, \mu) >- L_2$ $J_1 -(Reproduce, \mu) >- L_3$

$J_2 -(Reproduce, \mu) >- L_3$ $J_3 -(Reproduce, \mu) >- L_3$ $J_4 -(Reproduce, \mu) >- L_3$

$J_5 -(Reproduce, \mu) >- L_3$ $J_6 -(Reproduce, \mu) >- L_3$ $J_7 -(Reproduce, \mu) >- L_3$

$J_8 -(Reproduce, \mu) >- L_3$ $J_9 -(Reproduce, \mu) >- L_3$ $J_{10} -(Reproduce, \mu) >- L_3$

$J_1 -(Reproduce, \mu) >- L_4$ $J_2 -(Reproduce, \mu) >- L_4$ $J_3 -(Reproduce, \mu) >- L_4$

$J_4 -(Reproduce, \mu) >- L_4$ $J_5 -(Reproduce, \mu) >- L_4$ $J_6 -(Reproduce, \mu) >- L_4$

$J_7 -(Reproduce, \mu) >- L_4$ $J_8 -(Reproduce, \mu) >- L_4$ $J_9 -(Reproduce, \mu) >- L_4$

$J_{10} -(Reproduce, \mu) >- L_4$ $J_1 -(Reproduce, \mu) >- L_5$ $J_2 -(Reproduce, \mu) >- L_5$

$J_3 -(Reproduce, \mu) >- L_5$ $J_4 -(Reproduce, \mu) >- L_5$ $J_5 -(Reproduce, \mu) >- L_5$

$J_6 -(Reproduce, \mu) >- L_5$ $J_7 -(Reproduce, \mu) >- L_5$ $J_8 -(Reproduce, \mu) >- L_5$

$J_9 -(Reproduce, \mu) >- L_5$ $J_{10} -(Reproduce, \mu) >- L_5$ $J_1 -(Reproduce, \mu) >- L_6$

$J_2 -(Reproduce, \mu) >- L_6$ $J_3 -(Reproduce, \mu) >- L_6$ $J_4 -(Reproduce, \mu) >- L_6$

$J_5 -(Reproduce, \mu) >- L_6$ $J_6 -(Reproduce, \mu) >- L_6$ $J_7 -(Reproduce, \mu) >- L_6$

$J_8 -(Reproduce, \mu) >- L_6$ $J_9 -(Reproduce, \mu) >- L_6$ $J_{10} -(Reproduce, \mu) >- L_6$

$J_1 -(Reproduce, \mu) >- L_7$ $J_2 -(Reproduce, \mu) >- L_7$ $J_3 -(Reproduce, \mu) >- L_7$

$J_4 -(Reproduce, \mu) >- L_7$ $J_5 -(Reproduce, \mu) >- L_7$ $J_6 -(Reproduce, \mu) >- L_7$

$J_7 -(Reproduce, \mu) >- L_7$ $J_8 -(Reproduce, \mu) >- L_7$ $J_9 -(Reproduce, \mu) >- L_7$

$J_{10} -(Reproduce, \mu) >- L_7$ $J_1 -(Reproduce, \mu) >- L_8$ $J_2 -(Reproduce, \mu) >- L_8$

$J_3 -(Reproduce, \mu) >- L_8$ $J_4 -(Reproduce, \mu) >- L_8$ $J_5 -(Reproduce, \mu) >- L_8$

$J_6 -(Reproduce, \mu) >- L_8$ $J_7 -(Reproduce, \mu) >- L_8$ $J_8 -(Reproduce, \mu) >- L_8$

$J_9 -(Reproduce, \mu) >- L_8$ $J_{10} -(Reproduce, \mu) >- L_8$ $J_1 -(Reproduce, \mu) >- L_9$

$J_2 -(Reproduce, \mu) >- L_9$ $J_3 -(Reproduce, \mu) >- L_9$ $J_4 -(Reproduce, \mu) >- L_9$

$J_5 -(Reproduce, \mu) >- L_9$ $J_6 -(Reproduce, \mu) >- L_9$ $J_7 -(Reproduce, \mu) >- L_9$

$J_8 -(Reproduce, \mu) >- L_9$ $J_9 -(Reproduce, \mu) >- L_9$ $J_{10} -(Reproduce, \mu) >- L_9$

$J_1 -(Reproduce, \mu) >- L_{10}$ $J_2 -(Reproduce, \mu) >- L_{10}$ $J_3 -(Reproduce, \mu) >- L_{10}$

$J_4 -(Reproduce, \mu) >- L_{10}$ $J_5 -(Reproduce, \mu) >- L_{10}$ $J_6 -(Reproduce, \mu) >- L_{10}$

$J_7 -(Reproduce, \mu) >- L_{10}$ $J_8 -(Reproduce, \mu) >- L_{10}$ $J_9 -(Reproduce, \mu) >- L_{10}$

$J_{10} -(Reproduce, \mu) >- L_{10}$

Figure B.13: PEPA nets model Part 5 of 6.

Arcs of the Net continued

$RJ_1 \; -(AddJ, \nu) >- J_1 \quad RJ_2 \; -(AddJ, \nu) >- J_1 \quad RJ_3 \; -(AddJ, \nu) >- J_1 \quad RJ_4 \; -(AddJ, \nu) >- J_1$

$RJ_5 \; -(AddJ, \nu) >- J_1 \quad RJ_1 \; -(AddJ, \nu) >- J_2 \quad RJ_2 \; -(AddJ, \nu) >- J_2 \quad RJ_3 \; -(AddJ, \nu) >- J_2$

$RJ_4 \; -(AddJ, \nu) >- J_2 \quad RJ_5 \; -(AddJ, \nu) >- J_2 \quad RJ_1 \; -(AddJ, \nu) >- J_3 \quad RJ_2 \; -(AddJ, \nu) >- J_3$

$RJ_3 \; -(AddJ, \nu) >- J_3 \quad RJ_4 \; -(AddJ, \nu) >- J_3 \quad RJ_5 \; -(AddJ, \nu) >- J_3 \quad RJ_1 \; -(AddJ, \nu) >- J_4$

$RJ_2 \; -(AddJ, \nu) >- J_4 \quad RJ_3 \; -(AddJ, \nu) >- J_4 \quad RJ_4 \; -(AddJ, \nu) >- J_4 \quad RJ_5 \; -(AddJ, \nu) >- J_4$

$RJ_1 \; -(AddJ, \nu) >- J_5 \quad RJ_2 \; -(AddJ, \nu) >- J_5 \quad RJ_3 \; -(AddJ, \nu) >- J_5 \quad RJ_4 \; -(AddJ, \nu) >- J_5$

$RJ_5 \; -(AddJ, \nu) >- J_5 \quad RJ_1 \; -(AddJ, \nu) >- J_6 \quad RJ_2 \; -(AddJ, \nu) >- J_6 \quad RJ_3 \; -(AddJ, \nu) >- J_6$

$RJ_4 \; -(AddJ, \nu) >- J_6 \quad RJ_5 \; -(AddJ, \nu) >- J_6 \quad RJ_1 \; -(AddJ, \nu) >- J_7 \quad RJ_2 \; -(AddJ, \nu) >- J_7$

$RJ_3 \; -(AddJ, \nu) >- J_7 \quad RJ_4 \; -(AddJ, \nu) >- J_7 \quad RJ_5 \; -(AddJ, \nu) >- J_7 \quad RJ_1 \; -(AddJ, \nu) >- J_8$

$RJ_2 \; -(AddJ, \nu) >- J_8 \quad RJ_3 \; -(AddJ, \nu) >- J_8 \quad RJ_4 \; -(AddJ, \nu) >- J_8 \quad RJ_5 \; -(AddJ, \nu) >- J_8$

$RJ_1 \; -(AddJ, \nu) >- J_9 \quad RJ_2 \; -(AddJ, \nu) >- J_9 \quad RJ_3 \; -(AddJ, \nu) >- J_9 \quad RJ_4 \; -(AddJ, \nu) >- J_9$

$RJ_5 \; -(AddJ, \nu) >- J_9 \quad RJ_1 \; -(AddJ, \nu) >- J_{10} \quad RJ_2 \; -(AddJ, \nu) >- J_{10} \quad RJ_3 \; -(AddJ, \nu) >- J_{10}$

$RJ_4 \; -(AddJ, \nu) >- J_{10} \quad RJ_5 \; -(AddJ, \nu) >- J_{10}$

Figure B.14: PEPA nets model Part 6 of 6.

BIBLIOGRAPHY

[1] Primary Industies and Resources South Australia, Pacific oyster aquaculture in South Australia, 1986.

[2] Bio-PEPA web page : http://homepages.inf.ed.ac.uk/jeh/Bio-PEPA/biopepa.html, 2012.

[3] ERSEM (the European Regional Seas Ecosystem Model) page : http://www.meece.eu/library/ersem.html, 2012.

[4] SBML Software Matrix web page: http://sbml.org/SBML_Software_Guide/SBML_Software_Matrix, 2012.

[5] SBML web page : http://sbml.org/Main_Page, 2012.

[6] Systems biology software infrastructure web page: http://www.sbsi.ed.ac.uk/, 2012.

[7] Agence France-Presse. Pacific oyster farmers see global warming in poor harvest, 2010.

[8] M. Ajmone Marsan, A. Bobbio, and S. Donatelli. Petri nets in performance analysis: An introduction. *Lectures on Petri Nets I: Basic Models*, pages 211–256, 1998.

[9] O. Akman, M. Guerriero, L. Loewe, and C. Troein. Complementary approaches to understanding the plant circadian clock. *Computing*, pages 1–19, 2010.

[10] J.I. Allen and E.A. Fulton. Top-down, bottom-up or middle-out? Avoiding extraneous detail and over-generality in marine ecosystem models. *Progress in Oceanography*, 84(1-2):129–133, January 2010.

[11] M. Avgeri and S. Gilmore. The Bio-PEPA Eclipse Plug-in User Manual, 2012.

[12] J.C.M. Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335(2-3):131–146, 2005.

[13] J.W. Baretta, W. Ebenhöh, and P. Ruardij. The European regional seas ecosystem model, a complex marine ecosystem model. *Netherlands Journal of Sea Research*, 33(3-4):233–246, July 1995.

[14] P. Barros, P. Sobral, P. Range, L. Chícharo, and D. Matias. Effects of sea-water acidification on fertilization and larval development of the oyster Crassostrea gigas. *Journal of Experimental Marine Biology and Ecology*, 440:200–206, 2013.

[15] E. Bartocci and P. Lió. Computational Modeling, Formal Analysis, and Tools for Systems Biology. *PLoS Computational Biology*, 12(1):1–22, 2016.

[16] S. Benkirane, R. Norman, E. Scott, and C. Shankland. Measles Epidemics and PEPA : an exploration of historic disease dynamics using process algebra. In *18th International Symposium on Formal Methods*, 2012.

[17] J.C. Blackford. A structure and methodology for marine ecosystem modelling. *Netherlands Journal of Sea Research*, 33:247–260, 1995.

[18] M. Boots and A. Sasaki. 'Small worlds' and the evolution of virulence: infection occurs locally and at a distance. *Proceedings of the Royal Society B-Biological Sciences*, 266(1432):1933–1938, 1999.

[19] T.A. Branch and B.M. DeJoseph. Impacts of ocean acidification on marine seafood. *Trends in Ecology and Evolution*, pages 1–9, 2012.

[20] R.V. Carvalho, J. Kleijn, and F.J. Verbeek. A multi-scale extensive Petri net model of the bacterial macrophage interaction. *BioPPN*, 1159, 2014.

[21] F. Ciocchetta and J. Hillston. Bio-PEPA: a framework for the modelling and analysis of biological systems. *Theoretical Computer Science*, 410:3065–3084, 2009.

[22] J. O. Dada and P. Mendes. Multi-scale modelling and simulation in systems biology. *Integrative biology : quantitative biosciences from nano to macro*, 3(2):86–96, 2011.

[23] E. De Maria, J. Despeyroux, and A. Felty. A Logical Framework for Systems Biology. *Formal Methods in Macro-Biology*, 8738:136–155, 2014.

[24] A. Degasperi. *Multi-scale modelling of biological systems in process algebra with multi-way synchronisation*. PhD thesis, University of Glasgow, 2011.

[25] A. Degasperi and M. Calder. A process algebra framework for multi-scale modelling of biological systems. *Theoretical Computer Science*, 488:15–45, June 2013.

[26] S. Doney, V. Fabry, R. Feely, and J. Kleypas. Ocean Acidification: The Other CO 2 Problem. *Annual Review of Marine Science*, 1(1):169–192, January 2009.

[27] A. Duguid, S. Gilmore, M. Smith, and M. Tribastone. The PEPA Eclipse Plug-in user manual, 2010.

[28] R. Fehling. A concept of hierarchical Petri nets with building blocks. *Advances in Petri Nets*, 674:148–168, 1993.

[29] F. Gazeau, C. Quiblier, J. M. Jansen, J. P. Gattuso, J. J. Middelburg, and C. H. R. Heip. Impact of elevated CO_2 on shellfish calcification. *Geophysical Research Letters*, 34(7):1–5, 2007.

[30] S. Gilmore, J. Hillston, L. Kloul, and M. Ribaudo. PEPA nets: a structured performance modelling formalism. *Performance Evaluation*, pages 111–130, 2003.

[31] S. Glasstone, K. J. Laidler, and H. Eyring. *The Theory of Rate Processes*. McGraw-Hill, London, 1941.

[32] E. Gosling. *Bivalve Molluscs: Biology, Ecology and Culture*. Wiley-Blackwell, 2003.

[33] M. L. L. Guerriero and J. K. Heath. Computational modeling of biological pathways by executable biology. *Methods in enzymology*, 487:217–251, 2011.

[34] M. Heiner, M. Herajy, F. Liu, C. Rohr, and M. Schwarick. Snoopy - A Unifying Petri Net Tool. In *Petri Nets 2012*, volume 7347, pages 398–407. Springer, 2012.

[35] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[36] J. Hillston. Tuning Systems: From Composition to Performance. *The Computer Journal*, 48(4):385–400, May 2005.

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

[38] isee systems. STELLA web page: http://www.iseesystems.com/softwares/Education/StellaSoftware.aspx, 2016.

[39] T. C. Johns, R. E. Carnell, J. F. Crossley, J. M. Gregory, J. F. B. Mitchell, C. A. Senior, S. F. B. Tett, R. A. Wood, and Low. The second Hadley Centre coupled ocean-atmosphere GCM: Model description, spinup and validation. *Climate Dynamics*, 13(2):103–134, 1997.

[40] S.A.L.M. Kooijman. *Dynamic Energy Budget theory for metabolic organisation*. Cambridge University Press, 3rd edition, 2010.

[41] S.A.L.M. Kooijman. Bibliography on Dynamic Energy Budget theory, 2012.

[42] J. R. Levine, T. Mason, and D. Brown. *lex & yacc*. O'Reilly, 2nd edition, 1992.

[43] R. Lintott, S. Mcmahon, K. Prise, C. Addie-lagorio, and C. Shankland. Using Process Algebra to Model Radiation Induced Bystander Effects. In *Proceedings of the 12th Conference on Computational Methods in Systems Biology*, pages 196–210, 2014.

[44] D. Machado, R. S. Costa, M. Rocha, E. C. Ferreira, B. Tidor, and I. Rocha. Modeling formalisms in Systems Biology. *AMB Express*, 1(1):45, 2011.

[45] D. Marco, D. Cairns, and C. Shankland. Optimisation of process algebra models using evolutionary computation. In *Proceedings of 2011 IEEE Congress on Evolutionary Computation*, pages 1296–1301. IEEE, 2011.

[46] D. Marco, E. Scott, D. Cairns, A. Graham, J. Allen, S. Mahajan, and C. Shankland. Investigating co-infection dynamics through evolution of Bio-PEPA model parameters: a combined process algebra and evolutionary computing approach. In *Proceedings of the 10th*

Conference on Computational Methods in Systems Biology, pages 227–246. Springer-Verlag, 2012.

[47] B. T. Martin, E. I. Zimmer, V. Grimm, and T. Jager. Dynamic Energy Budget theory meets individual-based modelling: a generic and accessible implementation. *Methods in Ecology and Evolution*, (3):445–449, 2012.

[48] C. McCaig, R. Norman, and C. Shankland. From individuals to populations: A symbolic process algebra approach to epidemiology. *Mathematics in Computer Science*, 2(3):535–556, 2009.

[49] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[50] R. Milner. *Communicating and Mobile Systems: the* π*-calculus,*. 1999.

[51] L. Miossec, R.M. Le Deuff, and P. Goulletquer. Alien species alert: Crassostrea gigas (Pacific oyster). Technical Report 299, ICES Cooperative Research, 2009.

[52] J. Nicol and J. Coulter. *Radiation Dose Enhancement: The development and application of radio consisting gold nanoparticles*. PhD thesis, Queen's University Belfast, 2016.

[53] D. Noble. *The Music of Life: Biology Beyond Genes*. Oxford University Press, 2006.

[54] N. H. Packard and S. Wolfram. Two-dimensional cellular automata. *Journal of Statistical Physics*, 38(5-6):901–946, 1985.

[55] G. Paun and F. J. Romero-Campero. Membrane Computing as a Modeling Framework . Cellular Systems Case Studies. *Formal Methods for Computational Systems Biology*, 5016:168–214, 2008.

[56] G. Paun and G Rozenberg. A guide to membrane computing. *Theoretical Computer Science*, 287(1):73–100, 2002.

[57] C.A. Petri and W. Reisig. Petri net. Scholarpedia, 2008.

[58] A. Pnueli. The Temporal Logic of Programs. *The 18th IEEE Symposium on Foundation of Computer Science*, pages 46–57, 1977.

[59] S. Pouvreau, Y. Bourles, S. Lefebvre, A. Gangnery, and M. Alunno-Bruscia. Application of a dynamic energy budget model to the Pacific oyster, Crassostrea gigas, reared under various environmental conditions. *Journal of Sea Research*, 56(2):156–167, August 2006.

[60] G. G Powathil, D. J. Adamson, and M. A. Chaplain. Towards predicting the response of a solid tumour to chemotherapy and radiotherapy treatments: clinical insights from a computational model. *PLOS Computational Biology*, 9(7), 2013.

[61] G. G. Powathil, K. E. Gordon, L. A. Hill, and M. A. J. Chaplain. Modelling the effects of cell-cycle heterogeneity on the response of a solid tumour to chemotherapy : Biological insights from a hybrid multiscale cellular automaton model. *Journal of Theoretical Biology*, 308:1–19, 2012.

[62] C. Priami. Algorithmic systems biology. *Communications of the ACM*, 2009.

[63] Z. Qi, M. Li, C. Fu, D. Shi, and J. You. Membrane Calculus: A formal method for Grid transactions. *Grid and Cooperative Computing*, 3251:73–80, 2004.

[64] S. F. Railsback and V. Grimm. *Agent-Based and Individual-Based Modeling A Practical Introduction*. Princeton University Press, 2012.

[65] 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.

[66] J. Ren and D. Schiel. A dynamic energy budget model: parameterisation and application to the Pacific oyster Crassostrea gigas in New Zealand waters. *Journal of Experimental Marine Biology and Ecology*, 361(1):42–48, June 2008.

[67] B. Rico-Villa, I. Bernard, R. Robert, and S. Pouvreau. A Dynamic Energy Budget (DEB) growth model for Pacific oyster larvae, Crassostrea gigas. *Aquaculture*, 305(1-4):84–94, 2010.

[68] F. J. Romero-Campero, J. Twycross, H. Cao, J. Blakes, and N. Krasnogor. A Multiscale Modelling Framework Based On P Systems. *Membrane Computing*, 5391:63–77, 2009.

[69] E. Scott. *PEPA or Bio-PEPA : A Comparison*. Dissertation, University of Stirling, 2011.

[70] E. Scott. PAL Parser source code: https://github.com/MissErinScott/PAL-Parser, 2016.

[71] E. Scott, A. Hoyle, and C. Shankland. PEPA'd Oysters: Converting Dynamic Energy Budget Models to Bio-PEPA, Illustrated by a Pacific Oyster Case Study. *Electronic Notes in Theoretical Computer Science*, 296:211–228, 2013.

[72] E. Timmins-Schiffman, M. J. O'Donnell, C. S. Friedman, and S. B. Roberts. Elevated pCO2 causes developmental delay in early larval Pacific oysters, Crassostrea gigas. *Marine Biology*, 160(8):1973–1982, oct 2012.

[73] J. J. Tyson and B. Novak. Regulation of the eukaryotic cell cycle: molecular antagonism, hysteresis, and irreversible transitions. *Journal of theoretical biology*, 210(2):249–263, 2001.

[74] W. M. P. van der Aalst, C. Stahl, and M. Westergaard. Strategies for Modeling Complex Processes Using Colored Petri Nets. *Transactions on Petri Nets and Other Models of Concurrency VII*, 7480:6–55, 2013.

[75] D. C. Walker and J. Southgate. The virtual cell–a candidate co-ordinator for 'middle-out' modelling of biological systems. *Briefings in bioinformatics*, 10(4):450–61, July 2009.

[76] U. Wilensky. NetLogo User Manual, 2012.

[77] T. Zhang, P. Brazhnik, and J. J. Tyson. Exploring mechanisms of the DNA-damage response: p53 pulses and their possible relevance to apoptosis. *Cell Cycle*, 6(1):85–94, 2007.