A formal computational framework for the study of molecular evolution

Marek Kwiatkowski

Doctor of Philosophy

Laboratory for Foundations of Computer Science
School of Informatics
The University of Edinburgh

2010

Abstract

Over the past 10 years, multiple executable modelling formalisms for molecular biology have been developed in order to address the growing need for a system-level understanding of complex biological phenomena. An important class of these formalisms are biology-inspired *process algebras*, which offer—among other desirable properties—an almost complete separation of model specification (syntax) from model dynamics (semantics). In this thesis, the similarity between this separation and the genotype-phenotype duality in evolutionary biology is exploited to develop a process-algebraic approach to the study of evolution of biochemical systems.

The main technical contribution of this thesis is the *continuous π-calculus* ($c\pi$), a novel process algebra based on the classical π-calculus of Milner *et. al.* Its two defining characteristics are: continuous, compositional, computationally inexpensive semantics, and a flexible interaction structure of processes (molecules). Both these features are conductive to evolutionary analysis of biochemical systems by, respectively, enabling many variants of a given model to be evaluated, and facilitating *in silico* evolution of new functional connections. A further major contribution is a collection of *variation operators*, syntactic model transformation schemes corresponding to common evolutionary events. When applied to a $c\pi$ model of a biochemical system, variation operators produce its evolutionary neighbours, yielding insights into the local fitness landscape and neutral neighbourhood.

Two well-known biochemical systems are modelled in this dissertation to validate the developed theory. One is the KaiABC circadian clock in the cyanobacterium *S. elongatus*, the other is a mitogen-activated protein kinase cascade. In each case we study the system itself as well as its predicted evolutionary variants. Simpler examples, particularly that of a generic enzymatic reaction, are used throughout the thesis to illustrate important concepts as they are introduced.

Acknowledgements

I am indebted to my Ph.D. advisor Ian Stark, who helped me with my work in innumerable ways and on as many occasions. I could not have completed my studies without his guidance.

I thank my examiners, Wan Fokkink and Vincent Danos, for taking the time to read my thesis, for their insightful comments, and for focussing on the essence of my work rather than the technical details. My defence was organised and chaired by Stephen Gilmore and I would like to take this opportunity to thank him as well for his time and efficiency.

I am grateful to my friends Alessandro Romanel and Casey Helgeson, who read draft versions of this thesis and provided valuable feedback.

My office mates from IF-3.50 helped me to maintain my sanity or to shed it altogether, I am not sure anymore. Thanks anyway!

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification.

Chapters 3 and 4 are a refined and expanded version of [88]. Other material contained in this dissertation is being prepared for publication.

Figures 2.1 and 6.4 were composed by Ms Natalia Camelo Vergara specifically to illustrate this research. Figure 2.3 used with permission from the author. Figure 4.4(a) adapted from Ian Stark's original graphs in [88].

(Marek Kwiatkowski)

This is likely a groundbreaking paper
and it may very well be correct
but I completely fail to understand it.
—An anonymous referee of [88]

Table of Contents

Chapter 1

Introduction

The advancement of our understanding of living matter and of our ability to influence it depends more and more on efficient data analysis methods, predictive mathematical models and faithful simulation techniques. This advancement is therefore as much of a challenge to mathematics and informatics as it is to the traditionally understood biological sciences. The field of *systems biology* [72, 79, 82] attempts to rise to this challenge, particularly by seeking new abstractions and knowledge representations to organise, comprehend and learn from biological data. Many established mathematical and computational frameworks have been adapted by systems biology for this purpose, including differential equations [76], Petri Nets [59], process algebras [126], Statecharts [47], and stochastic simulations [91]; and many others have been developed from scratch in order to tackle specific aspects of biological complexity.

Development is the processing of the genetic information (*genotype*) as it travels through multiple levels of organisation—genes, proteins, networks, cells, tissues, organisms and populations—ultimately building a biological entity (*phenotype*). The realisation that this process is both central to and shaped by biological evolution lies at the heart of *evolutionary developmental biology* (EDB) [26, 102]. Very much in the spirit of systems biology, EDB researchers have identified several high-level properties characterising development across very diverse animal taxa [117]. Among these properties are *robustness* [94], *evolvability* [52], *canalisation* [135], *modularity* [14] and *plasticity* [116]. While their importance in a wider context, and especially the hypothesis that they themselves evolved by nat-

ural selection, is hotly debated [93], they remain the most promising theoretical devices for describing and studying development.

Interestingly, the two best established general mathematical frameworks for studying evolution offer little help when it comes to complex genotype-phenotype relationships. *Population genetics* [29, 61] is concerned with changes in allele frequencies; here, fitness is the only manifestation of phenotype. *Evolutionary game theory* [69, 95], on the other hand, has no notion of genotype at all. In view of these limitations, much of theoretical evolutionary developmental biology relies on ad-hoc computer simulations [6, 16, 137, 151]. A unifying approach would undoubtedly be useful, particularly by providing a basis for rigorous definitions of the high-level principles mentioned above, but also as a platform for comparing results relating to different biological systems.

The aim of this dissertation is to investigate one way in which a specific class of systems biology techniques—namely, process algebras—may serve as a general framework for evolutionary developmental biology. Several features of process algebras suggest considerable potential for such an application. First and foremost, system description and system function are treated separately by process algebras as *syntax* and *semantics*, mirroring the duality of genotype and phenotype. Second, molecules can be represented in process-algebraic models directly, and thus genetic variation can be modelled directly as well. Third, a single process-algebraic system description can be used to produce a range of fundamentally distinct analyses, making it possible to study development under different basic assumptions and at different resolutions. Finally, process algebras have an inherently computational character, well suited for automated processing and analysis, which in turn is necessary if sampling or exhaustive analysis of multiple evolutionary variants of the same system is required.

Overview of contributions The first major contribution of this thesis is the *continuous π-calculus* ($c\pi$ for short), a novel process algebra for the study of evolutionary properties of biochemical systems. It is based on the classical π-calculus of Milner *et. al.* [99], adapted so as to express fully quantitative and continuous dynamics. A further divergence from the π-calculus is to relax the basic structure of interaction channels from strict one-to-one linkage to arbitrary many-to-many connectivity. The first development makes $c\pi$ models relatively inexpensive com-

putationally, which is useful when many models (i.e. many related genotypes) are to be analysed; it also ties $c\pi$ closely to ordinary differential equations, often the preferred dynamical framework in systems biology applications. The other facilitates rewiring of the agents in the model in a manner similar to how evolution reshapes protein networks, thus paving the way for formal treatment of genetic mutations.

Exactly such treatment, in the form of eleven *variation operators*, is the other major contribution of this dissertation. Each variation operator is a model transformation scheme corresponding to a specific class of mutations. Crucially, operators are purely syntactic constructions, completely oblivious to the semantics of the models they act on. This corresponds to the basic neo-Darwinian assumption that mutations take place at the level of genotype and are blind to the effect they have on phenotype. Naturally, the eleven operators given in this thesis do not cover the entire spectrum of mutation classes, but they nevertheless form a reasonably expressive collection, capable of rigorous modelling complex patterns of molecular evolution.

The evaluation of this framework consists of three modelling exercises. The first one is a $c\pi$ model of a circadian clock of the cyanobacterium *Synechococcus elongatus*; here the focus is on showing that $c\pi$ is a sound general-purpose modelling language, and evolutionary applications are only touched upon. The second is a demonstration of the expressive power of variation operators; it consists of a construction of a $c\pi$ model of a complex biochemical process (competitive enzyme inhibition) from a trivial initial model by applications of variation operators alone. The last exercise is an operator-driven computational exploration of the evolutionary neighbourhood of the MAPK signalling cascade; it is an example of the kind of applications the $c\pi$/operators framework is primarily designed for. These exercises are secondary contributions of this dissertation.

Prior knowledge Every effort has been made to make this dissertation accessible to the widest readership possible. We assume, however, that the reader is familiar with the basic principles of cell biology and evolutionary theory. In addition, they should possess a certain degree of mathematical and computational literacy.

Data All models, software and experiment results reported in this thesis can be fetched from `http://homepages.ed.ac.uk/stark/cpi/`, or requested directly from the author.

Overview of the thesis

Chapter 1 is this Introduction.

Chapter 2 contains a survey of existing research relevant to this thesis. Systems biology, process algebras and selected topics in evolutionary biology are covered. This chapter may be skipped if the reader feels comfortable with this material.

Chapter 3 introduces the continuous π-calculus ($c\pi$). Syntax, semantics and an algorithm for the extraction of differential equations are given. A running example of a simple enzymatic reaction is used throughout. This chapter is central to this thesis.

Chapter 4 contains a $c\pi$ model of a cyanobacterial circadian clock, adapted from [148]. Its purpose is to demonstrate the merits of $c\pi$ as a general-purpose biochemical modelling language, and so it can be skipped if the reader is only interested in evolutionary applications.

Chapter 5 introduces *variation operators*, formal model transformations corresponding to potential evolutionary changes of systems modelled with $c\pi$. The operators form the basis of the framework developed in this thesis. This chapter should be read together with the next one.

Chapter 6 contains two case studies. The first one demonstrates the expressive power of variation operators by building a trajectory of models leading to a $c\pi$ representation of competitive enzyme inhibition. The second one is an example of the possible use of the $c\pi$-based evolutionary framework: an investigation of the evolutionary neighbourhood of a well-known molecular system, the MAPK cascade.

Chapter 7 provides an evaluation of the work contained in this dissertation and discusses possible future research directions.

Chapter 2

Background

2.1 Introduction

The task of providing a concise yet comprehensive survey of the research underlying and motivating this thesis is not an easy one. The main difficulty lies in the fact that although the primary intended readership of the thesis is the theoretical computer science community, it should at least be accessible to a computationally-minded biologist. This requirement presents us with the non-trivial challenge of covering both the relevant biology and computer science research on two levels: basic for the non-specialist and state-of-the-art for the expert. This is what this chapter attempts to do.

One inevitable consequence of this approach is that any discussion has to very quickly focus on the most relevant aspects of each field. This leads to the unfortunate situation where some high-level concepts that are relevant, but not crucial, to the subsequent developments in this thesis, are mentioned in the high-level overview, but are not discussed in detail later. This is particularly true of the treatment of systems biology offered here.

The central development in this thesis is a process algebra for the modelling of evolutionary variation and dynamics in biochemical networks. This dictates the choice of topics covered in this chapter. The first one is systems biology, the broad scientific discipline to which our research belongs; here the focus is on different dynamical frameworks and specification languages, because process algebras are both. The second topic concerns existing process algebras for biology, as this

research area is the closest to ours. The third topic is neutrality and mutational robustness in evolutionary developmental biology, and in particular an abstract model of development [152], which we shall later attempt to recast in process-algebraic terms.

2.1.1 Overview of the chapter

Section 2.2 explains what systems biology is (§2.2.1) and reviews two of its aspects that are the most relevant for further discussion: dynamical models (§2.2.2) and model specification standards (§2.2.3).

Section 2.3 begins with a very general introduction to process algebras (§2.3.1), but quickly focuses on their recent application in biology (§2.3.2 and §2.3.3). It then discusses relevant biological applications of techniques related to process algebras (§2.3.4), and concludes with a high-level summary (§2.3.5).

Section 2.4 is devoted to recent advances in evolutionary theory concerning neutral evolution and mutational robustness. It contains a discussion of the genotype-phenotype distinction and its implications (§2.4.1), introduces the important concepts of robustness and evolvability (§2.4.2) and presents a general framework for the analysis of these properties (§2.4.3).

The cited works range from textbooks, review and seminal papers in case of the high-level overview sections to recent research articles in case of advanced topics.

2.2 Systems biology

2.2.1 Emergence and aims

The term *systems biology* [72, 79, 82] describes two related and overlapping concepts. On one hand, it is a field of research concerned primarily with the dynamic processes taking place in living cells. On the other, it is a way of doing science, where experimental developments are continuously supported by computational modelling and vice versa (see Fig. 2.1 and [80]). In both guises, systems biology is a response to the wealth of data collected in the post-genomic era by high-throughput experimental techniques. The fact that this data is the result of a

subtle interplay of many components defies reductionist attempts at the understanding of cellular processes. Systems biology is therefore a quest for suitable abstractions that would enable us to make sense of the experimental data and ultimately of the cell itself.

The most natural and most often employed abstraction is that of a dynamical process. The identity and physical details of cellular agents are neglected to the widest possible extent, and the only feature being studied is the dynamical behaviour of the entire system. This approach links the two faces of systems biology, for in order to study a living system in this way, one needs an abstract model that is both informed by experiment and drives it. Some of the mathematics used for this purpose is reviewed below (§2.2.2). Needless to say, this approach has the added benefit of moving some of the workload from a wet lab to a computer, which is almost always cheaper.

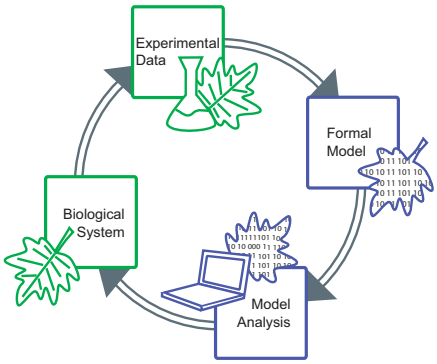


Figure 2.1: The virtuous circle of systems biology.

A different, though complementary, strategy is to seek organising principles, i.e. abstract characteristics that reappear in different systems and at different scales. The features often mentioned in this rôle are robustness [81], feedbacks and feed-forwards [2], modularity [2] and scale-free architecture of gene and protein networks [120]. The ultimate aim of this effort is to characterise the context in which Nature employs these mechanisms, the way they are implemented, and the objectives they fulfill in a manner that is generic enough to yield a unified understanding of biology.

2.2.2 Different modelling frameworks

We turn now to a discussion of mathematical formalisms most commonly used in systems biology to model dynamical processes. We focus on the modelling of molecular interactions, that is we assume that the task at hand is to predict transient and eventual behaviour of a system of molecules given their detailed interaction capabilities. The different methods we review make different simplifications of the physical reality in order to make the model analytically or com-

putationally tractable. We organise our discussion around these often conflicting assumptions.

Deterministic vs stochastic A model is *deterministic* if its initial state uniquely determines its behaviour at all time scales. Conversely, a *stochastic* model incorporates randomness, thereby making it possible for two executions of precisely the same model to differ. It is important, however, to understand that stochasticity does not necessarily imply non-determinism: a model with a non-trivial stochastic component can still consistently exhibit the same behaviour, especially when only high-level characteristics are considered.

Perhaps the most common dynamical formalism used in systems biology are coupled *Ordinary Differential Equations* (ODEs) [12]. ODEs specify rates of change of continuous real variables in a deterministic fashion. Each variable corresponds to the amount (concentration) of a molecular agent. The knowledge of all biochemical reactions in the system enables the modeller to write the rate of change of each variable as a function of that variable and the remaining ones (hence "coupled"). The resulting set of equations can rarely be solved analytically, but is usually dealt with easily by numerical integration methods.

Exactly the same basic principle underlies *Stochastic Differential Equations* (SDEs) [109], only now the rate of change of state variables may depend on a random variable, thereby yielding a stochastic model. Of particular interest here are the Langevin equations, where the stochastic component represents Brownian motion [57]. Another very important stochastic modelling framework is simply a set of biochemical reactions together with an initial state. It can be given stochastic dynamics by the Gillespie's Algorithm [56]. By drawing two real numbers from appropriate probability distributions, the Algorithm decides at each time step what single molecular event to simulate (i.e. which reaction to "fire"), and what time should elapse until the next one. The counts of molecules and the current time are then accordingly updated and the dice are rolled again.

Statistical mechanics teaches us that molecular kinetics should be seen as an essentially stochastic process and that their deterministic approximation holds provably only in the *thermodynamical limit*, that is when the number of molecules and the volume of the solution can be treated as practically infinite. This is often not the case for cellular systems, especially signalling and regulatory pathways

where molecules may be present in very few copies at any given time. Consequently, ODE models of many natural and synthetic biological systems give erroneous results, while stochastic simulations usually correctly predict the behaviour of the system [17]. Differential equations remain immensely useful, however, because in spite of the many improvements to the Gillespie Algorithm [55, 125], stochastic simulations are often computationally intractable for large models.

Continuous vs discrete A model is *continuous* if all state variables (such as amounts of molecules) admit values from dense, continuous sets (in practice the reals). Conversely, in a *discrete* model all state variables range over discrete sets (usually a finite set or the natural numbers). A model containing variables of both kinds is called *hybrid*. Of the frameworks discussed above, ODEs and SDEs are continuous and Gillespie-style simulations are discrete. Again, a discrete model usually requires more computational power than the corresponding continuous one, but it represents the physical reality better. In practice, hybrid models often offer more than acceptable accuracy at acceptable cost. Unfortunately, there is no widespread standard for hybrid modelling and as a result the possibility of a hybrid approach is often overlooked.

There is also a distinction between discrete-time and continuous-time models. In the former case, time is assumed to progress in discrete steps, or "jumps"; in the latter, time is a continuous real variable. Of the frameworks discussed above, ODEs and SDEs use continuous time, while time in Gillespie simulations progresses in a discrete fashion. It is important to realise, however, that discrete-time systems may retain an internal notion of real time and treat every step as occurring at a specific "real" time instant, with the intervals between steps having potentially different lengths. This is the case of Gillespie-style simulations.

Different kinetic laws A *kinetic law* is a function giving the rate at which a biochemical reaction proceeds. It depends on the amounts of substrates and inhibitors/catalysts. Strictly speaking, every reaction in a reaction set should be annotated with its kinetic law. In practice, unless specified differently, the *Law of Mass Action* is used; it mandates that the rate of a reaction is proportional to the amounts of its substrates. This principle follows from the kinetic theory and it is believed that all low-level chemical reactions obey it. Sometimes, however, it

is convenient to treat a few related reactions as a single one. In that case, while each of the constituent reactions may well follow the Law of Mass Action, their amalgamation often does not. An example of this situation are catalysed (enzymatic) reactions, which obey a non-linear dynamical law called *Michaelis-Menten kinetics*. Other kinetic laws of note include *Hill kinetics* (often used in models of gene regulation) and *linlog kinetics* (often used in models of metabolism).

2.2.3 Knowledge representation standards

Biologists have long used informal diagrams to represent the structure and dynamics of biological systems. One of the main shortcomings of this approach is that it introduces ambiguity of descriptions, because the same diagram may be interpreted differently by different people. Systems biology with its emphasis on computational methods made this problem particularly acute, because computers cannot yet be relied upon to resolve such ambiguities. Another problem has been the lack of a common format for model specification that would facilitate the exchange of models between different biological software packages. Several non-ambiguous biological specification languages have been proposed recently to address these issues and we briefly introduce three of the most influential.

Kohn maps *Molecular Interaction Maps* [83], or *Kohn Maps*, proposed in 1999, are the first model specification standard for systems biology. The primary concern of their author was the removal of ambiguity rather than ease of computational processing; hence, Kohn Maps are a purely graphical formalism without a standard textual representation. The diagrams are built from a limited set of graphical symbols and can be either "heuristic", meaning that the detailed interaction structure is unknown or omitted, or "explicit", meaning that all possible interactions are specified and computer simulation of the Map is possible [85]. For an example of a Molecular Interaction Map of a large, well-studied system, see the mammalian cell cycle model [84].

SBML The Systems Biology Markup Language (SBML) [71, 147] is a textual, machine-readable, XML-based language. Since its conception in the early 2000s it has become a *de facto* standard for storage and exchange of biological models.

From the point of view of this thesis, the most important feature of SBML is that it is deliberately agnostic about the dynamical framework in which the model is to be interpreted, making it possible to run the same model under different dynamical paradigms; we shall soon encounter the same feature with process algebras. While not being a modelling framework itself, SBML greatly extends the ability of systems biologists to investigate and communicate their models.

SBGN The Systems Biology Graphical Notation (SBGN) [89, 146] seeks to provide a standard for human-readable, diagrammatic representations of biological systems. It consists in fact of three graphical languages. The Process Diagrams specify state changes of individual agents (molecules). The Entity Relationship Diagrams, based largely on Kohn Maps, specify interactions between the agents and any constraints or conditions these interactions are subject to. Lastly, the Activity Flow Diagrams represent causal and temporal dependencies of biological events; because they are unsuitable for representation of state-based behaviour, they are not supposed to exist independently and it is expected that in the future they will be automatically generated from the other two. SBGN is a fairly recent development and it is impossible at the time of writing to adequately assess its impact on biology.

2.3 Process algebras and biology

2.3.1 Classical process algebras: models of concurrency

We provide now a very brief introduction to process algebras for the non-specialist reader. Experts are encouraged to skip to §2.3.2; otherwise they are requested to excuse the simplifications that follow.

Definition *Process algebras* or *process calculi* [48] are a family of mathematical formalisms for modelling of concurrent computations. The defining properties of process algebras are: use of *communication* (rather than e.g. shared memory) for synchronisation of otherwise independent computations; *parsimony*, i.e. use of very few basic constructs; and *compositionality*, a strong notion of modularity

$$\frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{\overline{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

Figure 2.2: An example SOS rule from the CCS process algebra. P, P', Q and Q' are metavariables referring to arbitrary processes; a is an action metavariable; $\overline{a}$ identifies the action complementary to a; τ is the distinguished "silent" action; finally, $|$ is the CCS parallel composition operator. The rule reads: **if** a process P **can** perform the a action and turn into P' **and** a process Q **can** perform the complementary $\overline{a}$ action and evolve into Q', **then** the process $P|Q$ **can** perform the special τ action and evolve into $P'|Q'$. Hence, the rule captures a synchronous step of concurrent computations. Observe that the evolution of the composite process $P|Q$ depends exclusively on the evolutions of its components P and Q, and thus compositionality is maintained.

where the complete behaviour of a model can be inferred from the behaviour of its constituents.

Syntax and semantics A definition of a typical process algebra consists of two elements. The first is an inductive definition of the *syntax*, that is a set of terms (called *processes*) representing different concurrent computations. Two processes can always be joined with a *parallel composition* operator and yield a valid process representing the computation consisting of the two threads denoted by the initial processes running in parallel. The second element is the definition of *semantics* of processes, i.e. their behaviour or meaning. This is usually accomplished using Structural Operational Semantics (SOS) [119], where a finite set of inference rules assigns to every process the set of its possible evolutions (or *transitions* or *actions*). The transitions of a composite process always depend exclusively on the evolutions of its components, yielding a *compositional* framework (Fig. 2.2).

Transition systems and behaviour The set of processes together with the semantic relation induced by SOS rules forms a *transition system*: a directed graph where every node is a process and every edge represents a potential evolution of a process. A different set of rules, even when operating on the same processes, will give rise to a different transition system. This separation of syntax

from semantics is important in the biological applications of process algebras (see §2.3.5). Another consequence of this setup is that different processes are observationally indistinguishable if their transition systems can mimic each other indefinitely; in this case we will call the two processes *bisimilar* (or, more generally, *behaviourally equivalent*). Hence, the process algebraic framework is able to recognise two computations as identical for all intents and purposes even if their syntactic specifications differ.

Compositionality A formal system is *compositional* if the semantics of a composite object can be fully inferred from the semantics of the components. It is important to realise, however, that compositionality does not preclude emergent behaviour: it merely requires that the semantic domain is rich enough to encompass potential, as well as actual, behaviour. In the case of process algebras, compositionality is usually achieved by having two kinds of transitions, corresponding precisely to potential for communication and to actual communication events (Fig. 2.2). Differential equations, on the other hand, are an example of a non-compositional framework: the solution of a catenation of ODE systems cannot be inferred from the isolated solutions of the initial components. Compositionality makes it possible to analyse a complex system by analysing each component in turn (semantic modularity) and guarantees that the semantics of each component remains valid in all possible contexts, and therefore does not need to be recomputed when other components are modified (encapsulation).

History Historically, process algebras can be organised into three schools of thought [4]. The first, due to R. Milner, led to the development of the Calculus of Communicating Systems [97] and its successor, the π-calculus [99], and can be credited with the use of SOS to separate syntax from semantics. C. A. R. Hoare introduced his Communicating Sequential Processes [68] as a programming language, using communication and parallel composition as its basic constructs. Finally, the Dutch school of J. Bergstra and J. W. Klop and their Algebra of Communicating Processes [7] introduced the algebraic approach, where combinators such as the parallel composition are thought of as algebraic operators and complex process expressions are simplified according to rules reminiscent of those of basic algebra. Today a plethora of process algebras exist, targeting specific as-

pects of concurrency including performance evaluation [67, 121], security [1, 13] and hybrid systems [8].

2.3.2 Seminal work of Regev et. al.

The process-as-molecule abstraction In the late 1990s, Aviv Regev and Ehud Shapiro realised that process calculi can be used to model molecular dynamics [126, 128, 129]. Under their interpretation, processes correspond to molecules, process interaction to biochemical reactions and parallel composition of processes to the spatial independence of individual molecules. The *name extrusion* mechanism specific to the π-calculus was used to represent formation of molecular complexes. Finally, a path through a transition system defined by a process corresponds to a possible dynamical evolution of the given molecular system. These insights made an entire field of concurrency research instantly applicable in systems biology.

Refinements Regev *et. al.* quickly identified two major deficiencies of their original work. The first was the lack of a fully quantitative semantics: the use of pure π-calculus forced all reactions to proceed at equal rates, in stark contrast with actual cellular dynamics. They addressed this problem by adapting the stochastic π-calculus [121], where every action is annotated with (the inverse of) its *expected* duration. Annotating every action with the mass-action rate constant of the reaction it represents makes executions of thusly modified stochastic π model equivalent to Gillespie simulations of the corresponding reaction set. The resulting Biochemical Stochastic π-Calculus [123] has been independently implemented as BioSpi [11] and Stochastic Pi-Machine (SPiM) [143] and has since been used in a number of non-trivial case studies [22, 86, 90].

The other problem was the lack of support for spatial aspects of cellular computations. Biochemical interactions are not only highly localised, but also actively modify the structure of the cellular compartments. The need to account for these phenomena led to the extension of the Biochemical Stochastic π-Calculus with BioAmbients [127], a notion of process location borrowed from the Ambient Calculus [23]. The reaction capabilities of processes were made dependent on their relative positions in a hierarchy of formal compartments ("ambients"), and ex-

tended with new primitives to dynamically change this hierarchy. BioAmbients are implemented as part of the BioSpi tool [11].

2.3.3 Modern calculi for biology

Since Regev's breakthrough, many existing process algebras have been used and many more designed for applications in biology. Here we discuss the most influential ones, focusing on their defining characteristics.

PEPA and Bio-PEPA The Performance Evaluation Process Algebra (PEPA) [67] is a well-established stochastic process calculus for the analysis of performance of concurrent systems. It has also been extensively used for biochemical modelling [18, 20, 21]. Although it is less expressive in general than the π-calculus, it usually yields models that are simpler than the corresponding π representations thanks to a more sophisticated parallel composition operator and lack of data passing. The recent variant designed especially for the modelling of biological systems, called Bio-PEPA [27], supports reactions with more than two substrates, arbitrary kinetic laws and cellular compartments. PEPA and Bio-PEPA also benefit from excellent tool support [144, 145].

Beta Binders/BlenX Beta Binders [122] were designed at the same time as BioAmbients to provide support for cellular compartments in the context of stochastic π-calculus. They do not offer hierarchical compartment structure and are thus inferior to BioAmbients in this respect. Beta Binders have recently evolved into BlenX [33], a general-purpose biological programming language, which was used in what is to the best of our knowledge the only molecular evolution research project based on a process algebra [34, 35, 130] (see also §6.3.1).

κ The κ (kappa) calculus [30] disposes of the strictly textual presentation typical of process calculi in favour of a very intuitive graphical notation. Furthermore, instead of modelling the states of molecules directly (i.e. using explicit processes to describe states), κ relies on sets of graph rewriting rules to specify state changes. This has a number of advantages, discussed in more detail below ("Rule-based modelling"). This original contribution has recently gained some exposure outside

the computer science community [45], which, as we argue below (§2.3.5) is a necessary condition for further advancement of the field.

Variants of π Numerous variants of the π-calculus have been created since Regev's work. The motivation is usually addressing a particular modelling need or simply making π-modelling more user-friendly. Here we mention a few of them: the spatial π-calculus [75] equips π-processes with a notion of position in space and conditions their interactions on their relative distance; Spico [86, 139] introduces sophisticated synchronisation patterns and uses them to develop an object-oriented perspective on process-algebraic modelling; the attributed π-calculus [74] allows the processes to carry arbitrary data and condition their behaviour on it; last but not least, π@ [149, 150] subsumes several location-focused calculi, including BioAmbients and Beta Binders.

2.3.4 Related topics

Model checking *Formal verification*, or *model checking* [5, 28], is a family of powerful analysis techniques for computational models. A model checking algorithm takes a logical formula and a formal model as inputs and verifies whether the formula is true of the model. This is not possible for arbitrary logics and model classes, due to decidability and/or tractability issues. The feasible combinations frequently involve transition systems and so process-algebraic models often benefit from this technique. Model checking is not limited to purely qualitative analysis, however. Stochastic model checkers such as PRISM [87, 124] can analyse stochastic models (e.g. Markov Chains [77]) and return quantitative measures, e.g. the probability of the property being satisfied by a random run of the model. Stochastic model checking has been applied in the biological context [63], but is still intractable for most non-trivial biological systems.

One application of formal verification to biological models and data that is of particular interest to us is model checking of time series. The model here is simply a linear transition system (the time series measurements arranged chronologically) and the logic of choice is usually the Linear Temporal Logic (LTL) [5, Ch. 5]. This setup avoids the problem of state space explosion but is of course of limited use for non-deterministic models, which may have many different yet equally valid traces.

Recent extensions of this technique have been used to estimate parameters [44] and to quantify robustness [43] of biological systems.

Petri Nets Petri Nets [103] are another prominent model of concurrent computing that has been used for biological modelling [15, 59, 64]. Petri Nets are mostly used as a simulation engine in this context, but work on model checking of biological systems exists [64]; furthermore, some of the basic structural properties of Petri Nets have direct biological interpretations [65]. Compared to process algebras, Petri Nets lack modularity, but their widespread recognition, abundance of existing software packages and, above all, intuitive graphical form makes them much more accessible to biologists.

Rule-based modelling Consider a protein with n independent, distinguishable phosphorylation sites; it has 2^n possible phosphorylation states. This is a simple instance of *combinatorial explosion*: the phenomenon of a huge number of possible states of relatively few initial components. When the dynamical behaviour has to be specified for each state separately—as is the case with Petri Nets and, to some extent, process calculi—the model becomes unnecessarily large. One solution is to specify dynamics by means of rewriting rules, so that a single rule is applicable in a number of states. As mentioned before, this is the approach adopted in κ [30]. Other rule-based formalisms include Biocham [10], a comprehensive model checking and simulation tool for biochemical modelling; similar, though less model-checking oriented Pathway Logic [41]; and the Language of Biochemical Systems [113, 114], a general-purpose biological programming language emphasising modularity and reusability of models.

2.3.5 Outlook

Strengths The process-algebraic abstraction of molecular interactions is attractive for several reasons. First of all, it provides a generic yet mathematically rigorous framework for description and simulation of biochemical systems. Second, the emphasis that process calculi place on concurrent behaviour reflects very well the spatial independence of molecules. Third, process calculi incorporate causality and non-determinism in a very natural way. Finally, they open the exciting

possibility of investigating biological systems by means of model checking and behavioural equivalences, as well as using compositionality to organise and execute models in a modular way.

One other major strength of process algebras, which is crucial to this thesis is that the same process-algebraic model may have different dynamical interpretations without losing its mathematical rigour (cf. [18]). This stems from the strong separation of formal syntax and formal semantics of process calculi: a process is a well-defined mathematical object which can in principle be given arbitrary semantics later on. Hence, any static information we have about a biological system, such as types of molecules and their interactions, can be formally and unambiguously specified without committing to any particular dynamical framework. This should be contrasted with, for example, ODE modelling, where the model *is* the dynamics and has to be completely rewritten if another kind of analysis needs to be performed.

Weaknesses The process-algebraic approach to biochemical modelling has two main weaknesses. The first is that is does not scale very well. When the molecule-as-process abstraction is applied strictly, the model has to account for every molecule. This is practical for systems with at most a couple of hundreds of molecules of every kind, which rules out large classes of systems, for example metabolic pathways. A new perspective, where a single agent in a process algebra corresponds to a larger quantity of molecules [20] has been developed recently to tackle this problem. Such coarse-grained approach makes it possible to analyse larger systems, but the stochastic dynamics derived from such models do not exactly match the underlying physics anymore. It is of course possible to abandon Regev's abstraction altogether, but it is unclear whether process algebras would then retain any advantage over other formal methods for biological modelling; to the best of our knowledge no sustained effort in this direction exists.

The other weakness is of a more sociological nature. The idiosyncrasies of process algebras, their steep learning curve and their textual form make them quite intimidating at the first glance. As a result, their visibility in biological journals is extremely poor and their adoption by biologists has been incidental. There are at least two solutions to this problem: the first is to develop sophisticated graphical notations and visualisation tools for process algebras; this has been done for κ

and SPiM. The other is to use them as a low-level language, a compilation target for a more high-level formalism. In any case, the impact of process algebras on biology will remain marginal unless this issue is addressed.

2.4 Selected topics in evolutionary theory

We now turn to a survey of selected concepts in theoretical evolutionary biology. We are especially interested in recent attempts at obtaining a unified theoretical perspective on evolutionary origins and properties of development. In this dissertation, we adopt one such perspective and provide a process-algebraic rendition of it; the material in this section is therefore highly relevant and important to subsequent discussions.

2.4.1 Genotype, phenotype and development

The *phenotype* of an organism is the totality of its observable characteristics. The *genotype* is all of its genetic information. Natural selection favours phenotypes that are fitter than others; variation and genetic drift randomly modify genotypes. The concepts of genotype and phenotype can be generalised to any setting where an evolutionary process underlies change: in the case of protein folding, for example, amino-acid sequences are genotypes and 3D protein structures are phenotypes. The genotype-phenotype distinction and the questions it raises is at the forefront of biological research [117] as well as at the heart of many issues addressed in this thesis and so we discuss it here in more detail.

Genotypes are transformed to phenotypes in a process called *development*. In all interesting cases it is a complex and highly degenerate mapping. Furthermore, in the case of genomes and organisms, development itself is guided by the genotype, and thus development is subject to the same evolutionary processes as any other feature of the organism. Evolutionary developmental biology (informally EDB or *evo-devo*) [26, 102] is a rapidly growing field of research seeking to understand the mechanisms and dynamics of the evolution of development.

Development and its evolution are tightly linked to a major outstanding problem in evolutionary biology: how does biological innovation emerge? It is often the

case that the degree of variation in a high-level phenotypical trait is significantly lower than the variability of the underlying genotypes, with animal body plans being the prime example [31]. On the other hand, sudden bursts of innovations such as the Cambrian radiation ostensibly defy the established gradualist view of evolution. The solution of this paradox may lie in the evolution of development and the way it translates random genotypical variation into apparently purposeful, discontinuous and constrained phenotypical change [50, 53, 54].

The importance of evo-devo notwithstanding, the situations where development can be treated as fixed or at least independent of the genotypes it acts on are common. Folding problems are a class of examples of fixed genotype-phenotype maps, and we discuss the case of RNA folding below (§2.4.3). In this dissertation we treat the syntactic representation of a biological system in a process algebra as genotype and the semantic interpretation of this model as phenotype. The development is thus the symbolic or numerical derivation of semantics and as such is fixed; from now on, therefore, we do not concern ourselves with variable genotype-phenotype maps.

2.4.2 Neutrality, robustness and evolvability

Neutral evolution A genetic event, such as mutation, is *neutral* if it does not change the fitness of the phenotype. Natural selection cannot distinguish between neutral mutants by definition, despite the fact that the phenotypes themselves may be different. The extent to which evolution advances by neutral changes was the subject of the selectionist-neutralist debate in 1960s and 1970s; see [106] for an overview of the debate and [78] for the definitive presentation of the neutralist point of view. Today the neutral theory serves mainly as a null model: natural selection is only invoked when the neutral theory fails to account for the given phenomenon. It has recently been argued, however, that this mode of thinking is not followed diligently enough and that many features of living organisms commonly ascribed to natural selection evolved in fact by neutral drift [93].

Robustness and evolvability We say that a biological system is *mutationally robust* (henceforth simply *robust*) if it continues to perform its function despite mutations. Many biological systems display robustness at many levels of organ-

isation [152]. Recently, robustness has received a lot of attention as a general organising principle of biological systems [81, 142, 152]. Robustness is closely linked to neutrality: if a system can perform its function despite mutational pressure, then most of these mutations have to be neutral, since advantageous mutations are in general very rare [42].

A system is *evolvable* if it can easily achieve new functionality through mutations. At the first glance evolvability is the exact opposite of robustness. It is important to realise that this is only the case on the level of genotypes [153]. There is evidence that phenotypical robustness and phenotypical evolvability are positively correlated (see §2.4.3). Evolvability is of course tightly linked to the question of the origins of novelty and so if robustness promotes evolvability, it also promotes biological innovation.

Second-order evolution Robustness can evolve, albeit via an indirect evolutionary process [152, Ch. 16–17]. Two organisms with equal fitness are identical from the perspective of immediate natural selection, even if they differ in their robustness. When faced with mutation pressure, however, the more robust phenotype has a greater chance to withstand it, and to pass its—presumably still robust—architecture to its offspring. Thus, the frequency of robust phenotypes can increase in the population. However, theoretical considerations suggest that this trend is very weak unless the population size is large or the mutation rate is high. This points to viruses as ideal candidates for experiments, and some evidence of selection for robustness in RNA viruses has indeed been found [101, 131]. Evolvability may also evolve along similar lines, but experimental evidence for this process is somewhat weaker [112].

2.4.3 Neutral spaces and RNA folding

We now present an abstract framework for studying neutrality, robustness and evolvability in biological systems. It stems from influential work on RNA folding, which we use here as an example. Our main reference for this section is [152], which contains this and many more case studies as well as a comprehensive treatment of the concept of neutral spaces.

Genotypes, phenotypes and accessibility Consider two abstract mathematical spaces: the space of genotypes and the space of phenotypes, connected by the genotype-phenotype mapping. No assumption is made about the structure of the phenotype space. The space of genotypes, however, is equipped with a binary *accessibility relation*, which links two genotypes if the first one can be turned into the other by a single mutational event. The exact characteristics of the spaces, the mapping and the accessibility relation may vary, but we usually find that genotype space is the set of sequences (DNA/RNA bases or more abstract letters), and hence is discrete or even finite. In this case the accessibility relation usually represents point mutation and is therefore symmetric. The phenotype space, on the other hand, usually lacks an obvious evolutionarily meaningful structure; indeed, it is often the aim of such studies to discover it. Finally, the genotype-phenotype map is always rather complex and many-to-one.

Neutral spaces and robustness The set of all genotypes mapped to the same phenotype is called the *neutral space* of this phenotype. Neutral spaces constitute a partition of the genotype space and so every genotype belongs to a unique neutral space. Robustness of a phenotype is defined as the size (cardinality) of its neutral space. This corresponds to the simple observation that phenotypes that can withstand a lot of changes to their genetic makeup, must in fact be encoded by many different genotypes. The structure of the neutral space w.r.t. the accessibility relation does *not* bear on the robustness of the corresponding phenotype in this setting. Also, note that no concept of fitness is used in this definition.

Definition of evolvability requires an evolutionary meaningful accessibility structure on the space of phenotypes. Stadler *et. al.* have demonstrated how to induce an evolutionary pre-topology on this space using only the accessibility of genotypes [140]. However elegant, this construction is too general to yield a reliably meaningful quantitative notion of evolvability, and in practice one uses additional information about the problem to define an evolvability measure. We now give one example of such an approach.

RNA folding An RNA strand folds onto itself using the Watson-Crick pairing of individual bases. The resulting three-dimensional structure is difficult to pre-

dict from the sequence. However, the set of base pairings that minimises the free energy of the molecule is computationally tractable through dynamic programming [155]. This set is a reasonable approximation of the actual 3D form; it is called a *shape*, because it gives rise to a unique 2D layout of the strand (Fig. 2.3).

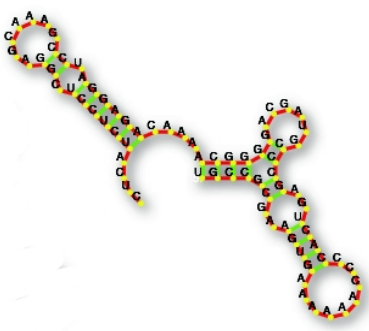


Figure 2.3: A shape

By treating RNA sequences of fixed length as genotypes, the shapes as phenotypes, and deeming two sequences mutually accessible if they differ by a single base (i.e. have Hamming distance equal to 1), one obtains an instance of the framework discussed above. This model and its implications were analysed extensively by Fontana *et. al.* [3, 49–51, 133]. The distribution of neutral space sizes (and thus robustness) was found to be highly non-random, with many sequences folding into the same few shapes. Furthermore, neutral spaces of these *frequent* shapes are connected, meaning that any sequences folding into a frequent shape can evolve into any other such sequence by means of point mutations alone without changing the resulting shape on the way. Finally, any random sequence is relatively few steps away from one that encodes a frequent shape (e.g. 15 steps in case of sequences of length 100).

Evolvability revisited As remarked before, the definition of phenotype evolvability requires a notion of evolutionary accessibility of phenotypes. In the case of RNA sequences and shapes, a shape p can be declared accessible from another one, say q, if the likelihood that a random mutation step away from the neutral space of q leads into the neutral space of p is greater than a threshold value. The distribution of these likelihoods follows a two-regime power law, suggesting a natural threshold value, namely the one corresponding to regime change [50]. Evolvability of a shape is then defined as the number of different shapes accessible from it.

The central result of the study of RNA folding is that phenotype robustness and phenotype evolvability are positively correlated. It is a strong indication that neutral evolution plays an important rôle in enabling biological innovation. Apart from theoretical analysis, this view is also supported by laboratory experiments [132]. Evidence of this effect has also been found in the case of protein evolution [46], and many believe it is a central feature of evolution in general.

Summary The notion of neutral space is applicable to many systems and at many levels of organisation [152]. The definition of robustness it promotes is exceptionally simple and yet without doubt meaningful from the biological perspective. The application of these concepts to the study of RNA folding reveals a deep connection between neutral evolution and biological innovation. This framework is not free from limitations, the most important being that development is assumed to be fixed. Furthermore, it is not clear whether it admits a representation of more complex kinds of genetic variation, such as recombination and horizontal gene transfer.

Chapter 3

The continuous π-calculus

3.1 Introduction

In this chapter we introduce the continuous π-calculus, a novel formal language for modelling of evolutionary variation of biochemical systems. It is a process algebra and so it enjoys a number of features useful for biological modelling, including formality, compositionality and succinctness. Formality removes ambiguity, which is quite common in conventional descriptions of biological systems, and thus facilitates computational processing and analysis of models. Compositionality provides strong support for modular modelling: models of subsystems can be put together with minimal effort to form a model of a larger system. Finally, succinctness facilitates model analysis and encourages identification of the key biological principles governing modelled systems and thus fully conforms to the spirit of systems biology [79].

In addition to these properties, we want the continuous π-calculus (henceforth $c\pi$) to support continuous-state and -time modelling and to facilitate expression of evolutionary variability. The support for variability is already partly provided by the process-algebraic paradigm, because every model has a clearly defined and easy to manipulate syntax; we enhance it by introducing *affinity networks* to explicitly represent flexible functional connections between different agents in the model. The support for continuous modelling is achieved by defining the $c\pi$ semantics in terms of real vector spaces. The main motivation for this development is to have a fully quantitative dynamical framework that is relatively

$$\dot{S} = -k_b \cdot S \cdot E + k_u \cdot C$$
$$\dot{P} = k_t \cdot C - k_d \cdot P$$
$$\dot{E} = -k_b \cdot E \cdot S + (k_t + k_u) \cdot C$$
$$\dot{C} = k_b \cdot S \cdot E - (k_t + k_u) \cdot C$$

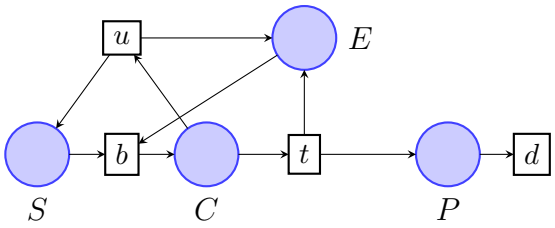


Figure 3.1: Two different models of a simple enzymatic reaction: a set of Ordinary Differential Equations (left) and a Petri Net (right). Both are formal and succinct but neither is compositional. ODEs are continuous, but offer no support for expressing evolutionary variability; Petri Nets could in principle express limited evolutionary variation, but their native semantics is not continuous.

inexpensive from the computational point of view, because any given system has potentially very many evolutionary variants whose behaviour we may want to analyse in detail.

3.1.1 The running example

Throughout this chapter we illustrate our constructions with the help of a simple molecular system: an abstract enzyme-catalysed biochemical reaction. The initial state is a solution of three agents: substrate S, enzyme E and product P. The enzyme can bind the substrate and form the complex C at the basal rate k_b. The complex can either dissociate and release the two original components (at the rate k_u), or the substrate can be converted into product and then released (at the rate k_t), leaving the enzyme unchanged. Finally, the product P can spontaneously degrade at the rate k_d. Figure 3.1 contains two models of this system in well-established formalisms, while Fig. 3.2 contains a $c\pi$ model complete with syntax, semantics and execution.

The syntactic part of the model consists of three separate parts: the *species definitions*, i.e. the specification of the kinds of molecules involved in the system; the *process term*, that is the definition of the initial state of the system; and the *affinity network* specifying the complementarity of the interaction sites of the molecules modelled by the species. The semantics of this model consists of the *transition system* encoding potential interaction capabilities of species, and the

process behaviour encapsulating the *immediate* dynamics of the system and its *potential* for interaction in other contexts. Together, these constructs allow us to compute the dynamical evolution of the system in a compositional manner.

3.1.2 Overview of the chapter

Section 3.2 is devoted to the presentation of the syntax and §3.3 defines the semantics of $c\pi$. The section on syntax defines the key notions of *species* (§3.2.1) and *processes* (§3.2.2) and discusses in detail their intended meaning. The presentation of semantics associates a transition system with species (§3.3.2) and a non-standard vector-based semantics with processes (§3.3.3). The chapter concludes with an algorithm for extracting Ordinary Differential Equations from $c\pi$ models (§3.4).

3.2 The syntax

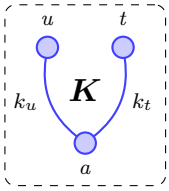
3.2.1 Species

The syntax of *species* is closely modelled on that of the π-calculus [98, 99, 111]. Familiarity with π-calculus is not required to follow this chapter, but the expert reader may find it useful to think of the syntax of $c\pi$ as that of π with symmetric polyadic communication, guarded definitions and choice, the ability to restrict multiple names at once, but without co-names.

Several of the definitions below use arbitrary real numbers to construct syntactic objects. The reader may think it prudent to mentally redefine these notions in terms of a countable approximation of $\mathbb{R}$ such as the computable real numbers or even the rationals. We choose not to do so explicitly because we do not rely on the countable nature of the syntax in any crucial way.

Definition 3.2.1. (name) A *name* is an element of the fixed, countably infinite, totally ordered set $\mathcal{N}$. We use lowercase Roman letters like a, b, x, y to denote names, and $\vec{a}, \vec{b}, \vec{x}, \vec{y}$, etc., to denote finite vectors of names.

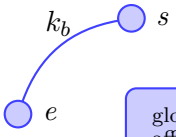
Definition 3.2.2. (prefix) A *prefix* is an expression of the form $a(\vec{x};\vec{y})$ (a *communication prefix*; a is a name, $\vec{x}, \vec{y}$ are vectors of names) or $\tau@k$ (an *autonomous*



$S(s) \triangleq s(x,y).x.S(s) + y.P$

$E(e) \triangleq (\boldsymbol{\nu} K) e\,\langle u,t\rangle\,.a.E(e)$

$P() \triangleq \tau@k_d.\mathbf{0}$ species definitions



global affinity network

$\boldsymbol{\Pi \triangleq c_1 \cdot E(e) \;||\; c_2 \cdot S(s) \;||\; c_3 \cdot P}$ process

$\{\!|\; S(s) \xrightarrow{s} (;x,y)(x.S(s)+y.P),$
$E(e) \xrightarrow{e} (\boldsymbol{\nu} K)(u,t;)(a.E(e)),$
$C \xrightarrow{\tau@k_u} (E(e) \mid S(s)),$
$C \xrightarrow{\tau@k_t} (E(e) \mid P),$
$P \xrightarrow{\tau@k_d} \mathbf{0}\;|\!\}$ multi-transition system *Trans*

$\lambda a.[\,$ **if** $a \equiv S(s)$ **then** $-k_b \cdot c_1 \cdot c_2$
else if $a \equiv E(e)$ **then** $-k_b \cdot c_1 \cdot c_2$
else if $a \equiv P$ **then** $-k_d \cdot c_3$
else $0\,]$ immediate behaviour $\frac{d\Pi}{dt}$

$\lambda a f x.[\,$ **if** $a \equiv S(s)$ **and** $f \equiv (;x,y)(x.S(s)+y.P)$ **and** $x = s$ **then** c_2
else if $a \equiv E(e)$ **and** $f \equiv (u,t;)(\boldsymbol{\nu} K)(a.E(e))$ **and** $x = e$ **then** c_1
else $0\,]$ potential behaviour $\partial\Pi$

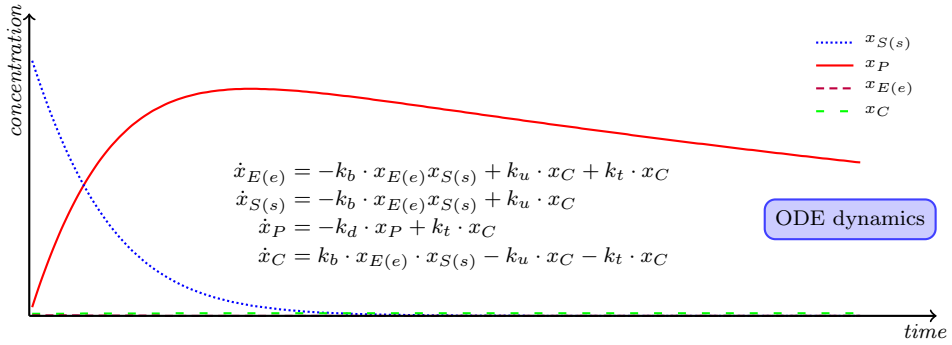


Figure 3.2: The syntax (top), semantics (middle) and execution (bottom) of the $c\pi$ model of an abstract enzymatic reaction. The syntax consists of: species definition, including the local affinity network K (referenced in the definition of $E(e)$); the $c\pi$ process; and the global affinity network giving the interaction structure of all free names of the process. The semantics of species is a multiset of transitions, while the semantics of the process consists of two real-valued functions. The system defines a set of ODEs describing its dynamic behaviour. Throughout the figure, C is used as an abbreviation for $(\boldsymbol{\nu} K)(a.E(e) \mid u.S(s) + t.P)$.

or *silent prefix*; k is a real number). We use the letter π and its derivatives such as π' or π_i to denote prefixes.

In the biochemical context, a prefix present in a species (defined below) represents the ability of the molecule modelled by this species to perform a specific action. In the case of a silent prefix $\tau@k$, the action is an autonomous (i.e. without an interacting partner) state change of a molecule; k is the mass-action kinetic rate constant of this transformation. A communication prefix $a(\vec{x};\vec{y})$ denotes the presence of an active site a on the surface of the molecule; if the molecule interacts with another using this site, it makes a state transition. The transition happens when another species with a complementary prefix is encountered. During this interaction the name vector $\vec{x}$ is passed to the partner species and another vector (represented by $\vec{y}$) is received. This scheme raises the following two issues: firstly, the use of name-passing in this context needs justification, because sites are not physically exchanged between interacting proteins. While name-passing is indeed a less-than-perfect abstraction for biochemical interaction (see also §7.1.2), together with name restriction it yields a very powerful mechanism for modelling of dynamical formation of complexes, and for this reason we choose to retain it. The second issue is that we need a notion of complementarity of sites; we choose to abandon the classical π-calculus concept of *co-names* and introduce instead a more general—and quantitative—construct of *affinity networks*, where any name can in principle communicate with any other and at different intensities.

Definition 3.2.3. (affinity network) An *affinity network* is a finite undirected graph whose vertices are names and whose edges are labelled with real numbers. We use uppercase letters such as M, N and K to refer to affinity networks.

Definition 3.2.4. (species) The set SPEC of *species* is defined by the following grammar:

$$A, B \;::=\; \mathbf{0} \;\mid\; D(\vec{a}) \;\mid\; \Sigma_{i=0}^{n}\pi_i.A_i \;\mid\; A|B \;\mid\; (\boldsymbol{\nu} M)A \tag{3.1}$$

For every *invocation* $D(\vec{a})$ there is an associated *species definition* $D(\vec{y}) \triangleq B$, where $\vec{y}$ coincides with the set of free names of B (see below). We require that every definition invocation in a species is *prefix-guarded*, that is appears below a Σ in the term tree.

A species represents interaction capabilities and state changes of a molecule. The intended meaning of the above syntactic constructs is as follows: The *null process* $\mathbf{0}$ denotes a molecule incapable of any action. The invocation $D(\vec{a})$ behaves

in the same way as the body B of its definition. The *choice* $\Sigma_{i=0}^{n}\pi_i.A_i$ denotes mutually exclusive interaction options and their consequences: upon performing the action specified by π_i, the molecule changes its state to A_i. The *parallel composition* $A|B$ of species enjoys the interaction capabilities of both components and thus denotes juxtaposition of two molecules or domains. Finally, *name restriction* $(\boldsymbol{\nu} M)A$ restricts the scope of the names mentioned in the affinity network M to A; it has no direct biochemical denotation, but is used as a device to model dynamical formation of complexes.

The above definitions are central to this dissertation, and so we use a number of abbreviations and auxiliary notions to manage them more effectively. They are collected below.

Notations We write $|\vec{x}|$ for the length of the vector $\vec{x}$, and $\vec{x}$**++**$\vec{y}$ for the catenation of the vectors $\vec{x}$ and $\vec{y}$. When A is a set of names, we write $\vec{A}$ for the vector consisting of the elements of A ordered according to the canonical ordering of $\mathcal{N}$. We sometimes abuse the vector notation and write $\vec{x}$ for the *set* of unique elements of the vector $\vec{x}$—see e.g. Def. 3.2.5 immediately below; the intended interpretation should be clear from the context. For a communication prefix $a(\vec{x};\vec{y})$, we write $a(\vec{y})$ if $|\vec{x}| = 0$, $a\langle\vec{x}\rangle$ if $|\vec{y}| = 0$ and simply a if $|\vec{x}| = |\vec{y}| = 0$.
Any set-theoretical predicate on an affinity network M refers to the set of its vertices; for example $x \in M$ asserts that the name x is a vertex of M. When a and b are names, we write $M(a, b)$ for the label of the edge linking a and b in M; if there is no such edge, or one of the names is not a vertex of M, $M(a, b)$ is defined be equal to 0.
When $D(\vec{y}) \triangleq B$ is a species definition, we call D the *handle*, $\vec{y}$ the *arguments* of *parameters*, B the *body* and $|\vec{y}|$ the *arity* of this definition. When D is a handle and $\mathcal{D}$ is a set of species definitions, we write $D\#\mathcal{D}$ to indicate that D *is fresh for* $\mathcal{D}$, that is no definition with handle D is, or is invoked by, an element of $\mathcal{D}$. We assume that there is a countably infinite set $\mathcal{H}$ of handles, and so there are always fresh handles available. For species of the form $\Sigma_{i=0}^{n}\pi_i.A_i$ we use the $+$-notation in the case of low n, so e.g. $\Sigma_{i=0}^{2}\pi_i.A_i$ may be rendered as $\pi_0.A_0 + \pi_1.A_1 + \pi_2.A_2$ and $\Sigma_{i=0}^{0}\pi_i.A_i$ as $\pi_0.A_0$. We omit the trailing $\mathbf{0}$ in species, so e.g. $a(x;y).\mathbf{0}$ becomes $a(x;y)$, and empty parentheses in case of invocation of definitions with no arguments, so e.g. $a.D()$ is now $a.D$. Finally, we write $A \subset B$ to indicate that A is a *subspecies* (i.e. subterm) of B.

Definition 3.2.5. (free and bound names) The sets of *free names* (fn) and *bound names* (bn) of a species are defined as follows:

$$\begin{aligned}
\mathsf{fn}(\mathbf{0}) &\stackrel{\text{df}}{=} \varnothing & \mathsf{bn}(\mathbf{0}) &\stackrel{\text{df}}{=} \varnothing \\
\mathsf{fn}(D(\vec{a})) &\stackrel{\text{df}}{=} \vec{a} & \mathsf{bn}(D(\vec{a})) &\stackrel{\text{df}}{=} \varnothing \\
\mathsf{fn}(A|B) &\stackrel{\text{df}}{=} \mathsf{fn}(A) \cup \mathsf{fn}(B) & \mathsf{bn}(A|B) &\stackrel{\text{df}}{=} \mathsf{bn}(A) \cup \mathsf{bn}(B) \\
\mathsf{fn}((\boldsymbol{\nu} M)A) &\stackrel{\text{df}}{=} \mathsf{fn}(A) \setminus M & \mathsf{bn}((\boldsymbol{\nu} M)A) &\stackrel{\text{df}}{=} \mathsf{bn}(A) \cup (\mathsf{fn}(A) \cap M) \\
\mathsf{fn}(\tau@k.A) &\stackrel{\text{df}}{=} \mathsf{fn}(A) & \mathsf{bn}(\tau@k.A) &\stackrel{\text{df}}{=} \mathsf{bn}(A) \\
\mathsf{fn}(a(\vec{x};\vec{y}).A) &\stackrel{\text{df}}{=} \{a\} \cup \vec{x} \cup (\mathsf{fn}(A) \setminus \vec{y}) & \mathsf{bn}(a(\vec{x};\vec{y}).A) &\stackrel{\text{df}}{=} \mathsf{bn}(A) \cup (\mathsf{fn}(A) \cap \vec{y}) \\
\mathsf{fn}(\Sigma_{i=0}^{n} \pi_i.A_i) &\stackrel{\text{df}}{=} \bigcup_i \mathsf{fn}(\pi_i.A_i) & \mathsf{bn}(\Sigma_{i=0}^{n} \pi_i.A_i) &\stackrel{\text{df}}{=} \bigcup_i \mathsf{bn}(\pi_i.A_i)
\end{aligned}$$

When X, Y are sets of names and A a species, we write $X\#A$ and say that X *is fresh for* A to indicate that $X \cap \mathsf{fn}(A) = \varnothing$; similarly, $X\#Y$ means $X \cap Y = \varnothing$. If x is a name, $x\#A$ stands for $x \notin \mathsf{fn}(A)$. Observe that the same name can be free and bound in different parts of a species at the same time; in this case, of course, it is not fresh for this species.

A name bound by the communication prefix can be seen as a placeholder for actual names to be received from the communicating partner. A formal device to manage the replacement of placeholders with actual (received) names is called *substitution*. We write $A\{\vec{a}/\vec{x}\}$ for the species arising when we take A and replace all *free* occurrences of elements of $\vec{x}$ with the corresponding ones of $\vec{a}$, and call it the *substitution of* $\vec{x}$ *by* $\vec{a}$ *in* A. Note that this requires $|\vec{a}| = |\vec{x}|$ and all elements of $\vec{x}$ to be distinct.

Another consequence of having name binding, either by communication prefixes or by name restrictions, is that the actual identity of a bound name is irrelevant: if y is just a placeholder, any $x \neq y$ would do the job just as well. This leads to the notion of α-*equivalence*: two species are α-equivalent if one can be obtained from the other by a consistent replacement of binders and bound names, for example $a(x;y).D(y)$ is equivalent to $a(x;z).D(z)$ (because y is bound) but not to $z(x;y).A(y)$ (because a is free). From now on we do not distinguish between α-equivalent species and thus the word "species" shall in fact mean "α-equivalence class of species".

For further discussion of substitution and α-equivalence (albeit in the context of the λ-calculus) we refer to [115].

$\mathbf{0}\|A \equiv A$	$(c \cdot \mathbf{0})\|\|P \equiv P$
$A\|B \equiv B\|A$	$P\|\|Q \equiv Q\|\|P$
$(A\|B)\|C \equiv A\|(B\|C)$	$(P\|\|Q)\|\|R \equiv P\|\|(Q\|\|R)$
$\Sigma_{i=0}^{n}\pi_i.A_i \equiv \Sigma_{i=0}^{n}\pi_{\sigma_i}.A_{\sigma_i}$ perm. σ	$(c+d)\cdot A \equiv (c\cdot A)\|\|(d\cdot A)$
$(\boldsymbol{\nu}M)A \equiv A$ $M\#A$	$c\cdot(A\|B) \equiv (c\cdot A)\|\|(c\cdot B)$
$(\boldsymbol{\nu}M)(\boldsymbol{\nu}N)A \equiv (\boldsymbol{\nu}N)(\boldsymbol{\nu}M)A$ $M\#N$	$c\cdot A \equiv c\cdot B$ $A \equiv B$
$(\boldsymbol{\nu}M)(A\|B) \equiv A\|(\boldsymbol{\nu}M)B$ $M\#A$	

Figure 3.3: Axioms generating structural congruence on species (left) and processes (right). Note how the congruence of species is embedded in that of processes.

Definition 3.2.6. (structural congruence of species) The *structural congruence* of species is the smallest congruence $\equiv$ on SPEC satisfying the rules in Fig. 3.3(l) and containing the α-equivalence of species. We write $\mathcal{S}$ for SPEC modulo $\equiv$.

The rôle of structural congruence is to equate species that are meant to represent the same objects but are syntactically distinct. The rules in Fig. 3.3 all have a simple interpretation in this context. In what follows, however, we maintain a careful distinction between SPEC and $\mathcal{S}$ in order to precisely state the correctness results of $c\pi$ semantics.

Recall that the intended meaning of the parallel composition of species $A|B$ is the juxtaposition of molecules denoted by A and B. A non-trivial parallel composition is therefore a species that models two or more molecules, not one. We call all species that cannot be represented as a parallel composition of non-trivial terms *prime*; thus a prime species is guaranteed not to model independent molecules. This distinction is crucial to the proper development of process semantics, because a biochemical reaction proceeds at different rates depending on whether the substrates are independent of each other.

Definition 3.2.7. (prime species) A species $A \in \mathcal{S}$ is called *prime* if $A \not\equiv \mathbf{0}$ and whenever $A \equiv B|C$, we have either $B \equiv \mathbf{0}$ or $C \equiv \mathbf{0}$. The set of prime species is denoted $\mathcal{S}^{\#}$. Note that $\mathcal{S}^{\#}$ is a proper subset of $\mathcal{S}$.

The following result allows us to represent any species in terms of prime species:
Theorem 3.2.8. (prime species decomposition) For every $A \in \text{SPEC}$, there exists a unique multiset $\mathsf{primes}(A) = \{\!|A_1, \ldots, A_n|\!\} \subseteq \mathcal{S}^{\#}$ called the *prime species decomposition of* A, such that $A \equiv A_1 \mid \ldots \mid A_n$. Observe that $\mathsf{primes}(\mathbf{0}) = \varnothing$. Furthermore, for any A and B, $\mathsf{primes}(A|B) = \mathsf{primes}(A) \cup \mathsf{primes}(B)$ (union of multisets), and if $A \equiv B$, then $\mathsf{primes}(A) = \mathsf{primes}(B)$.

Proof. See Appendix. □

Similar decompositions are possible w.r.t. more complex equivalences, especially bisimulations [100], but as we are interested in biological applications of the calculus, we do not refine the above result beyond the structural congruence.

Example Our example system (§3.1.1) is a solution of four molecular agents: the substrate S, the enzyme E, the product P and the enzyme-substrate complex C. Usually in such a situation we give one $c\pi$ species per molecular species, and we could do so here. We take a slightly more roundabout route instead, starting with the observation that the complex C is not a first-class citizen: it does not have its "own" interaction capabilities, but is entirely driven by its constituents: enzyme and substrate. We reflect this in our $c\pi$ model by defining the species in a way where the complex can arise dynamically from the interaction of the enzyme and substrate species. Note that the same reasoning can be applied to the product: all of its behaviour has to be already encoded by the substrate, since the latter is transformed into the former. For the sake of clarity, however, we do give a separate species for the product. Recall that we assumed that the only action the product can undertake is to spontaneously degrade at the rate k_d. We model this by setting

$$P() \stackrel{\Delta}{=} \tau@k_d.\mathbf{0} \tag{3.2}$$

Note the empty brackets indicating that this definition has no parameters—a direct consequence of the fact that $\mathsf{fn}(\tau@k_d.\mathbf{0}) = \varnothing$.

Now we turn to the substrate. It binds the enzyme and then either unbinds or transforms into the product. We model the act of binding by a communication event on a name s. Unbinding and transformation are further (mutually exclusive) communication events, this time using the names x and y received during binding. This natural language specification translates to $c\pi$ as:

$$S(s) \stackrel{\Delta}{=} s(x, y).(x.S(s) + y.P) \tag{3.3}$$

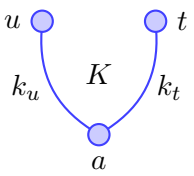


Figure 3.4: The local affinity network K.

The enzyme reacts with the substrate to form a complex and after the dissociation it returns to its original state. We have already decided to model the binding as a communication event in which the substrate receives two names. The enzyme must therefore send two names; after that, it has to further synchronise with the substrate in order to either unbind from it or transform it into the product. We use the name e to model the binding site of the enzyme. We send names u (for *unbind*) and t (for *transform*) to the substrate and use the name a (for *act*) for synchronisation within the complex; hence, a has to communicate with both u and t in order to ensure that both fates of the bound substrate are possible.

Before we give the definition of the enzyme, we have to take care of one more subtle issue: we need different copies of t, u and a for every complex in the solution, otherwise there would be unwanted cross-talk between distinct complexes. We use name restriction (cf. Eq. 3.1) to obtain fresh names for every enzyme, and therefore for every complex. In $c\pi$ the name restriction involves specifying the interaction topology of the restricted names in the form of a local affinity network; in our case this is the simple three-point network K in Fig. 3.4. Hence, the final form of the enzyme species is:

$$E(e) \stackrel{\text{df}}{=} (\boldsymbol{\nu} K) e\,\langle u, t\rangle\,.a.E(e) \tag{3.4}$$

Lastly, we specify the connections between the remaining (i.e. free) names. There are only two of them in the three definitions we have given: e and s. We assumed that their interaction models binding of enzyme to substrate, and hence we set their communication rate to k_b.

Figure 3.2(top) contains all the species definitions we have just given as well as the global affinity network.

3.2.2 Processes

Processes are simply collections of species, each labelled with a real number interpreted as the concentration of the molecular species. A process is therefore a model of a solution of many biochemical substances.

Definition 3.2.9. (process) The set PROC of *processes* is defined by the following grammar:

$$P, Q \quad ::= \quad c \cdot A \;\mid\; P||Q \tag{3.5}$$

where c is a real number and $A \in$ SPEC. We use letters P, Q, R, etc. to denote processes. When B is a species, we write $B \subset P$ to indicate that P contains a subterm $c \cdot A$ such that $B \subset A$ (defined previously). Note that it is meaningful to speak about the sets of free and bound names of a process—they are unions of respective sets defined for the species appearing in the given process.

The interpretation of Eq. 3.5 is straightforward: $c \cdot A$ is the solution consisting of a single agent A whose concentration is c, while $P||Q$ is the mixture of solutions P and Q. Just as in the case of species, there are processes that are technically different, but are supposed describe the same system (solution); we identify them with the help of a structural congruence, as before:

Definition 3.2.10. (structural congruence of processes) The *structural congruence of processes* is the smallest congruence on PROC satisfying the rules in Fig. 3.3(r). We write $P \equiv Q$ to indicate that the processes P and Q are structurally congruent, and $\mathcal{P}$ for PROC modulo $\equiv$. We ensure that the distinction between the congruence of processes and the congruence of species is evident from the context.

We now give the process corresponding to the initial state of our example system:

Example Assume the initial concentrations of enzyme, substrate and product are c_1, c_2 and c_3, respectively. The process describing this situation is

$$c_1 \cdot E(e) \;||\; c_2 \cdot S(s) \;||\; c_3 \cdot P() \tag{3.6}$$

This process, together with the species definitions and the global affinity network given before, forms the complete syntax of the $c\pi$ model of our example system (cf. Fig. 3.2(top)).

3.3 The semantics

We move now to define semantics of the syntactic objects defined in the previous section. Our first goal is to define semantics of species. They depend crucially on the concept of *concretions*.

$$
\begin{array}{lll|lll}
(\boldsymbol{\nu}M)(A|F) \equiv A|(\boldsymbol{\nu}M)F & M\#A & & F|\mathbf{0} \equiv F & & \\
(\boldsymbol{\nu}M)(F|A) \equiv F|(\boldsymbol{\nu}M)A & M\#F & & F|A \equiv A|F & & \\
(\boldsymbol{\nu}M)F \equiv F & M\#F & & (F|A)|B \equiv F|(A|B) & & \\
(\boldsymbol{\nu}M)(\boldsymbol{\nu}N)F \equiv (\boldsymbol{\nu}N)(\boldsymbol{\nu}M)F & M\#N & & (A|F)|B \equiv A|(F|B) & & \\
(\vec{b};\vec{y})(A|B) \equiv A|(\vec{b};\vec{y})B & \vec{y}\#A & & F|A \equiv F|B & A \equiv B &
\end{array}
$$

$$
(\vec{b};\vec{y})A \equiv (\vec{b};\vec{y})B \qquad A \equiv B
$$

Figure 3.5: The structural congruence of concretions is the smallest congruence on Conc containing α-equivalence and satisfying the above 11 rules.

3.3.1 Concretions

Definition 3.3.1. (concretion) A *concretion* is any term defined by the following grammar:

$$F \;::=\; (\vec{b};\vec{y})A \;\mid\; F|A \;\mid\; A|F \;\mid\; (\boldsymbol{\nu}M)F \tag{3.7}$$

where A is a species, $\vec{b},\vec{y}$ are vectors of names and M is an affinity network. Symbols $(\boldsymbol{\nu}–)$ and $|$ are distinct from the corresponding ones for species. We use letters F and G to range over concretions. Names occurring in concretions can be free or bound, and the appropriate definitions are analogous to the case of species. Concretions have therefore their own α-equivalence, and from now on we do not distinguish between α-equivalent concretions. We have a structural congruence $\equiv$ for concretions as well, induced by the rules on Fig. 3.5. We write Conc for the set of concretions and $\mathcal{C}$ for Conc modulo $\equiv$, and do distinguish between equivalent concretions in order to precisely define the semantics of species.

A concretion should be seen as a species that has committed itself to taking part in a specific interaction that has not yet taken place. When two *compatible* concretions meet, they interact and form a species—the parallel composition of the post-interaction residues. This scheme is formalised by the notion of *pseudo-application*, defined below. For another example of a concretion style presentation of a process algebra, see e.g. [98].

Definition 3.3.2. (pseudo-application) The *pseudo-application* is a binary partial function $– \circ –$: Conc × Conc ⇀ Spec, defined by structural induction over its arguments. For the base case, $(\vec{a};\vec{x})A \circ (\vec{b};\vec{y})B$ is defined if and only if

$|\vec{a}| = |\vec{y}|$ and $|\vec{b}| = |\vec{x}|$, in which case the result is $A\{\vec{b}/\vec{x}\}|B\{\vec{a}/\vec{y}\}$. The inductive clauses are as follows:

$$(\vec{a};\vec{x})A \circ (F|B) \stackrel{\text{df}}{=} ((\vec{a};\vec{x})A \circ F)|B \qquad (A|F) \circ G \stackrel{\text{df}}{=} A|(F \circ G)$$
$$(\vec{a};\vec{x})A \circ (B|F) \stackrel{\text{df}}{=} B|((\vec{a};\vec{x})A \circ F) \qquad (F|A) \circ G \stackrel{\text{df}}{=} (F \circ G)|A$$
$$(\vec{a};\vec{x})A \circ (\boldsymbol{\nu}M)F \stackrel{\text{df}}{=} (\boldsymbol{\nu}M)((\vec{a};\vec{x})A \circ F) \qquad (\boldsymbol{\nu}M)(F) \circ G \stackrel{\text{df}}{=} (\boldsymbol{\nu}M)(F \circ G)$$

For the two clauses in the bottom line we assume that M is fresh for the other concretion involved. In the presence of α-equivalence this condition can always be met, and hence the only reason for a pseudo-application to be undefined is the arity mismatch of the concretions in the base case. When the pseudo-application $F \circ G$ is defined, we say that F and G are *compatible* and write $F{\downarrow}G$.

Proposition 3.3.3. The following hold for any species B, compatible concretions F and G, and any affinity network $M\#F$:

(i) $F \circ (B|G) \equiv B|(F \circ G)$,

(ii) $F \circ (G|B) \equiv (F \circ G)|B$,

(iii) $F \circ (\boldsymbol{\nu}M)G \equiv (\boldsymbol{\nu}M)(F \circ G)$.

Proof. We prove the first claim by induction on the structure of F. The other two have analogous proofs. For the base case, assume $F = (\vec{a};\vec{x})C$ for some $\vec{a}$, $\vec{x}$ and C. We have: $F \circ (B|G) = ((\vec{a};\vec{x})C) \circ (B|G) = B|((\vec{a};\vec{x})C \circ G) = B|(F \circ G)$. For the inductive case there are the following three possibilities:

- $F = F'|A$ for a species A and a concretion F'. We have: $F \circ (B|G) = (F'|A) \circ (B|G) = (F' \circ (B|G))|A \stackrel{\text{IH}}{\equiv} (B|(F' \circ G))|A \equiv B|((F' \circ G)|A) = B|((F'|A) \circ G) = B|(F \circ G)$.
- $F = A|F'$ for a species A and a concretion F'. Analogously to the above.
- $F = (\boldsymbol{\nu}N)F'$ for an affinity network N and a concretion F'. We make sure through α-conversion that N is fresh for both B and G, and we have: $F \circ (B|G) = ((\boldsymbol{\nu}N)F') \circ (B|G) = (\boldsymbol{\nu}N)(F' \circ (B|G)) \stackrel{\text{IH}}{\equiv} (\boldsymbol{\nu}N)(B|(F' \circ G)) \equiv B|((\boldsymbol{\nu}N)(F' \circ G)) = B|((\boldsymbol{\nu}N)(F') \circ G) = B|(F \circ G)$.

□

Theorem 3.3.4. For any compatible concretions F and G:

(i) $F \circ G \equiv G \circ F$,

(ii) If $F' \equiv F$ and $G' \equiv G$ then $F'{\downarrow}G'$ and $F' \circ G' \equiv F \circ G$.

Proof. (sketch) The first claim follows from Prop. 3.3.3 by induction on the structure of F. Once (i) is established, it is enough to consider the case $F' = F'$ in (ii) and the proof becomes a straightforward induction on the derivation of $G \equiv G'$. □

3.3.2 The transition system of species

One of the hallmarks of process algebra is a transition system semantics defined using Structural Operational Semantics (SOS) [118, 119]. In this approach, a fixed set of rules is used to infer the steps that the given syntactic object (a process-algebraic term) can make. The set of these steps constitutes the behaviour of the object (cf. §2.3.1).

In the case of process algebras designed to model quantitative aspects of systems, including process algebras for biology, it is important to account for multiple capabilities for the same behaviour. Consider for example the $c\pi$ species $a.\mathbf{0}$. It models a molecule with one interaction site (a); after the interaction on this site, the molecule disappears (degrades). A conventional SOS system for π-like calculi would assign the transition set $\{a.\mathbf{0} \xrightarrow{a} \mathbf{0}\}$ to this species. The same transition set would be assigned to the species $a.\mathbf{0} + a.\mathbf{0}$, which models a molecule with two a sites. However, assuming standard stochastic kinetics, the latter molecule is two times more likely to engage in an interaction than the former. As the two molecules exhibit different quantitative behaviour, it is a mistake to assign the same semantic object to both.

There are at least two solutions to this problem. The first is to meticulously label the transitions with the information about the context of their derivation. In this setting, the semantics of $a.\mathbf{0} + a.\mathbf{0}$ is the set $\{a.\mathbf{0} + a.\mathbf{0} \xrightarrow{a,+_L} \mathbf{0}, a.\mathbf{0} + a.\mathbf{0} \xrightarrow{a,+_R} \mathbf{0}\}$, with the subscripts $-_L$ and $-_R$ indicating the parts of the species responsible for the transition; it is different from the semantics of $a.\mathbf{0}$, which is now the singleton set $\{a.\mathbf{0} \xrightarrow{a,\cdot} \mathbf{0}\}$. This labelling approach is used by the stochastic π-calculus [121]. The other solution is to switch to multisets of transitions; the semantics of $a.\mathbf{0}+a.\mathbf{0}$ now becomes the multiset $\{\!|a.\mathbf{0}+a.\mathbf{0} \xrightarrow{a} \mathbf{0}, a.\mathbf{0}+a.\mathbf{0} \xrightarrow{a} \mathbf{0}|\!\}$. This is the approach adopted by PEPA [67] and we use it here for $c\pi$ as well: the semantics of a $c\pi$ species shall be a multiset of transitions, with every transition belonging to one of three classes:

Class 1: From a species to a concretion, labelled by a name. Transitions of this kind represent potential for interaction; more precisely, a transition $A \xrightarrow{a} (\vec{b};\vec{y})B$ means that the species A can interact with another by sending $\vec{b}$ on the channel a and then evolve to B, with $\vec{y}$ replaced by the data received.

Class 2: From a species to another species, labelled by $\tau@k$ where k is a real number, for example $A \xrightarrow{\tau@1.5} B$. This transition indicates the ability of the species A to evolve into B at the basal rate of k (here 1.5) without interaction with another species. Examples of such behaviour include degradation, where B is $\mathbf{0}$, or complex dissociation, with B of the form $B'|B''$.

Class 3: From one species to another, labelled by a term $\tau\langle a,b\rangle$ where a,b are names, for example $A \xrightarrow{\tau\langle a,b\rangle} B$. This transition also denotes the potential for spontaneous transformation of A into B, but now the basal rate of transformation is the affinity between the names a and b. This affinity is determined either by the global affinity network of the model or by a local network, say M, which is yet to be introduced by restriction $(\boldsymbol{\nu}M)$. Transitions of this kind can be viewed as provisional—they represent the same behaviour as the transitions of the second kind, but when the basal rate of transformation is not yet known; they are necessary, however, for the semantics of species to be fully compositional.

Formal definitions of a multi-transition system and of the $c\pi$ multi-transition system can be found below.

Definition 3.3.5. (multi-transition system) A *multi-transition* system is a tuple $(\mathcal{A},\mathcal{L},\mathcal{B},\mathcal{T})$, where the first three elements are non-empty sets (called *sources*, *labels* and *targets*, respectively) and the last one is a multiset of triples (called *transitions*) of the form (α,λ,β), where $\alpha\in\mathcal{A}, \lambda\in\mathcal{L}$ and $\beta\in\mathcal{B}$. Instead of the tuple (α,λ,β) we write $\alpha \xrightarrow{\lambda} \beta$ and call it a *transition from α to β labelled with λ*.

Definition 3.3.6. (the $c\pi$ multi-transition system) The $c\pi$ *multi-transition system* is the multi-transition system $(\textsc{Spec}, \mathcal{L}, \textsc{Spec}\cup\textsc{Conc}, \mathit{Trans})$, where $\mathcal{L} \stackrel{\text{df}}{=} \mathcal{N}\cup\{\tau@k : k\in\mathbb{R}_{\geq 0}\}\cup\{\tau\langle a,b\rangle@k : a,b\in\mathcal{N}, k\in\mathbb{R}_{\geq 0}\}$ and the multiset *Trans* of transitions is defined by the SOS rules in Fig. 3.6. More precisely, there are exactly as many transitions of a given form in *Trans* as many distinct derivations of it can be performed using the rules in Fig. 3.6. When $A\in\textsc{Spec}$, we write $\mathit{Trans}(A)$ for the multiset of all transitions with source A.

$$\frac{\pi_j = a_j(\vec{b}_j;\vec{y}_j)}{\Sigma_{i=0}^{n}\pi_i.A_i \xrightarrow{a_j} (\vec{b}_j;\vec{y}_j)A_j}\ \textsc{Choice-1}(j,n) \qquad \frac{A \xrightarrow{\alpha} E}{A|B \xrightarrow{\alpha} E|B}\ \textsc{Par-Left}$$

$$\frac{\pi_j = \tau@k}{\Sigma_{i=0}^{n}\pi_i.A_i \xrightarrow{\tau@k} A_j}\ \textsc{Choice-2}(j,n) \qquad \frac{B \xrightarrow{\alpha} E}{A|B \xrightarrow{\alpha} A|E}\ \textsc{Par-Right}$$

$$\frac{A \xrightarrow{a} F \quad B \xrightarrow{b} G \quad F\downarrow G}{A|B \xrightarrow{\tau\langle a,b\rangle} F \circ G}\ \textsc{Com-1} \qquad \frac{A \xrightarrow{\alpha} E \quad \alpha \notin M}{(\boldsymbol{\nu}M)A \xrightarrow{\alpha} (\boldsymbol{\nu}M)E}\ \textsc{Res-1}$$

$$\frac{A \xrightarrow{\tau\langle a,b\rangle} B \quad a,b \in M}{(\boldsymbol{\nu}M)A \xrightarrow{\tau@M(a,b)} (\boldsymbol{\nu}M)B}\ \textsc{Com-2} \qquad \frac{A \xrightarrow{\tau\langle a,b\rangle} E \quad a,b \notin M}{(\boldsymbol{\nu}M)A \xrightarrow{\tau\langle a,b\rangle} (\boldsymbol{\nu}M)E}\ \textsc{Res-2}$$

$$\frac{B \xrightarrow{\alpha} E \quad D(\vec{y}) \stackrel{\text{df}}{=} B}{D(\vec{b}) \xrightarrow{\alpha\{\vec{b}/\vec{y}\}} E\{\vec{b}/\vec{y}\}}\ \textsc{Defn}$$

Figure 3.6: SOS rules generating the transition systems for species. The CHOICE rules are in fact rule schemes that can be instantiated for any $n \in \mathbb{N}$ and $0 \leq j \leq n$. The letter α ranges over all possible transition labels and E over all possible targets.

The following crucial result states that species equivalence is a behavioural equivalence: congruent species have equivalent transitions:

Theorem 3.3.7. Let $A \equiv B$. There exists a one-to-one multiset function ϕ from *Trans*(A) onto *Trans*(B) such that if $\phi(A \xrightarrow{\alpha} E) = (B \xrightarrow{\alpha'} E')$ then $\alpha = \alpha'$ and $E \equiv E'$.

Proof. (sketch) The proof proceeds by induction on the derivation of $A \equiv B$. For every transition in *Trans*(A) we exhibit a corresponding one in *Trans*(B) through case-analysis of the derivation tree. We then argue that this association is a bijection between the multisets of transitions. □

From now on it is completely safe not to distinguish between equivalent species and therefore, unless otherwise stated, the word "species" stands for "species equivalence class". Moreover, we do not distinguish between transitions whose labels are equal and sources and targets are both equivalent.

Below we derive the transitions for the species of the running example.

Example Recall that there are four species in our example system. Three of them are defined by Eqs. (3.2)–(3.4) and the fourth one (the enzyme-substrate complex) arises dynamically from the interaction of enzyme and substrate. We give the transitions for the first three before turning to the complex. The product species has only one transition:

$$\dfrac{\dfrac{}{\tau@k \xrightarrow{\tau@k} \mathbf{0}}\textsc{Choice-2(0,0)} \qquad P() \stackrel{\Delta}{=} \tau@k}{P \xrightarrow{\tau@k} \mathbf{0}}\textsc{Defn} \tag{3.8}$$

The only transition of the substrate species is derived similarly:

$$\dfrac{\dfrac{}{s(x,y).(x.S(s)+y.P) \xrightarrow{s} (;x,y)(x.S(s)+y.P)}\textsc{Choice-1(0,0)} \qquad S(s) \stackrel{\Delta}{=} s(x,y).(x.S(s)+y.P)}{S(s) \xrightarrow{s} (;x,y)(x.S(s)+y.P)}\textsc{Defn} \tag{3.9}$$

The enzyme species has also just one transition, but the derivation is slightly longer due to the use of name restriction:

$$\dfrac{\dfrac{\dfrac{}{e\,\langle u,t\rangle\,.a.E(e) \xrightarrow{e} (u,t;)a.E(e)}\textsc{Choice-1(0,0)} \qquad e \notin K}{(\boldsymbol{\nu} K)e\langle u,t\rangle\,.a.E(e) \xrightarrow{e} (\boldsymbol{\nu} K)(u,t;)a.E(e)}\textsc{Res-1} \qquad E(e) \stackrel{\Delta}{=} (\boldsymbol{\nu} K)e\langle u,t\rangle\,.a.E(e)}{E(e) \xrightarrow{b} (\boldsymbol{\nu} K)(u,t;)a.E(e)}\textsc{Defn} \tag{3.10}$$

Finally, let us turn to the enzyme-substrate complex. This is the species that emerges from the interaction of the substrate and enzyme species. The technical device to manage interaction is the pseudo-application and so the complex arises as a result of pseudo-application of two concretions: one provided by the substrate, the other by the enzyme. The appropriate concretions are derived in Eqs. (3.9) and (3.10) above: they are $(;x,y)(x.S(s)+y.P)$ and $(\boldsymbol{\nu} K)(u,t;).a.E(e)$. We compute their pseudo-application:

$$\begin{aligned}
& (;x,y)(x.S(s)+y.P) \circ (\boldsymbol{\nu} K)(u,t;).a.E(e) \\
= \;& (\boldsymbol{\nu} K)((;x,y)(x.S(s)+y.P) \circ (u,t;).a.E(e)) \\
= \;& (\boldsymbol{\nu} K)((x.S(s)+y.P)\{(u,t)/(x,y)\} \mid a.E(e)\{()/()\}) \\
= \;& (\boldsymbol{\nu} K)((u.S(s)+t.P) \mid a.E(e))
\end{aligned} \tag{3.11}$$

Thus (3.11) is the species describing the enzyme-substrate complex molecule. We now move to the derivation of its transitions. Recall that the complex should be able to split and release the product P and the enzyme $E(e)$. This is evidenced by the following

derivation (observe that $(;)P \circ (;)E(e) = P|E(e)$):

$$\dfrac{\dfrac{\dfrac{}{u.S(s)+t.P \xrightarrow{t} (;)P}\textsc{Choice-1(1,1)} \quad \dfrac{}{a.E(e) \xrightarrow{a} (;)E(e)}\textsc{Choice-1(0,0)}}{(u.S(s)+t.P) \mid a.E(e) \xrightarrow{\tau\langle t,a\rangle} P \mid E(e)}\textsc{Com-1} \quad t,a \in K}{(\boldsymbol{\nu} K)((u.S(s)+t.P) \mid a.E(e)) \xrightarrow{\tau@k_t} (\boldsymbol{\nu} K)(P \mid E(e))}\textsc{Com-2} \tag{3.12}$$

The complex has also one other transition

$$(\boldsymbol{\nu} K)((u.S(s)+t.P) \mid a.E(e)) \xrightarrow{\tau@k_u} (\boldsymbol{\nu} K)(S(s) \mid E(e)) \tag{3.13}$$

which can be derived analogously to the previous one, but using the rule Choice-1(0,1) instead of Choice-1(1,1). Also, observe that the restriction $(\boldsymbol{\nu} K)$ in the targets of the last two transitions can safely be dropped since we do not distinguish between equivalent species anymore, and K binds neither e nor s.

The transitions derived above are recorded in Fig. 3.2(middle).

3.3.3 The vector semantics of processes

The semantics of a $c\pi$ process should compositionally specify the dynamics of the modelled system in terms of continuous time and state space. Assuming that all reactions are governed by the Law of Mass Action, the syntax of a $c\pi$ model holds just enough information to define an Initial Value Problem (IVP) [12], i.e. a set of Ordinary Differential Equations together with an initial state. Unfortunately, ODE descriptions are non-compositional: unless we have extra information, we are unable to derive the set of ODEs that govern the behaviour of a system from the ODEs that govern the subsystems. For that reason we do not give semantics in terms of IVPs, and construct instead a less standard description of behaviour. Of course, differential equations can still be extracted from $c\pi$ models and we give a suitable algorithm later (§3.4).

An important feature of π-related process algebras is the ability to dynamically generate an unbounded variety of different behaviours. In terms of $c\pi$ this means that there are species such that the closure of their multi-transition system (i.e. their transitions, transitions of the targets, and so on) is infinite. Since $c\pi$ processes are built from species, the semantics of processes has to deal with these infinities. Admittedly, systems with infinitely many components do not occur in nature, but such an abstraction may nevertheless be useful, e.g. when studying polymerisation.

Unfortunately, even if the reader is concerned neither with compositionality nor with accounting for infinite and potentially infinite behaviours, they should not skip this section as the ODE extraction algorithm relies on the concepts defined here.

Definition 3.3.8. (the process space) The *process space* $\mathbb{P}$ is the vector space $(\mathbb{R}^{(\mathcal{S}^\#)}, +, \times, 0_\mathbb{P})$ equipped with the product topology. The set $\{\mathbf{1}_A\}_{A\in\mathcal{S}^\#}$, where $\mathbf{1}_A(A) \stackrel{\text{df}}{=} 1$ and $\mathbf{1}_A(B) \stackrel{\text{df}}{=} 0$ when $A \not\equiv B$ forms a basis of $\mathbb{P}$.

Definition 3.3.9. (support of a process) Let $P \in \mathbb{P}$. The set $\mathsf{supp}(P) \subseteq \mathcal{S}^\#$ is called the *support of* P and is given by

$$\mathsf{supp}(P) \stackrel{\text{df}}{=} \{A \in \mathcal{S}^\# : P(A) > 0\} \tag{3.14}$$

We extended this definition to operate on Proc by defining the support of a syntactic process by induction on the syntax as:

$$\mathsf{supp}(c \cdot A) \stackrel{\text{df}}{=} \{B \in \mathcal{S}^\# : B \in \mathsf{primes}(A)\} \tag{3.15}$$

$$\mathsf{supp}(P||Q) \stackrel{\text{df}}{=} \mathsf{supp}(P) \cup \mathsf{supp}(Q) \tag{3.16}$$

Clearly, $\mathsf{supp}(P) = \mathsf{supp}(Q)$ for any $P \equiv Q$.

Definition 3.3.10. (the space of potentials) The *space of potentials* $\mathbb{D}$ is the vector space $(\mathbb{R}^{(\mathcal{S}^\#\times\mathcal{C}\times\mathcal{N})}, +, \times, 0_\mathbb{D})$ equipped with the product topology. The collection $\{\mathbf{1}_{(A,F,x)}\}_{(A,F,x)\in\mathcal{S}^\#\times\mathcal{C}\times\mathcal{N}}$, where $\mathbf{1}_{(A,F,x)}(B,G,y) \stackrel{\text{df}}{=} 1$ iff $A \equiv B$, $F \equiv G$ and $x = y$, and 0 otherwise, is a basis of this space.

First of all, observe that every process can be identified with a vector in $\mathbb{P}$; in fact $\mathbb{P}$ is the *phase space* of $c\pi$ systems, where every dimension (prime species) corresponds to a type of agent, and every point (process) uniquely determines the state of the system. The dynamical evolution of a $c\pi$ model, therefore, is a (continuous) trajectory in $\mathbb{P}$ starting with the process encoding the initial state; unless it contains points with infinite support, it is also a trajectory in $\mathcal{P}$. We can specify this trajectory by giving the gradient vector associated with every point. This is the meaning of the $\mathbb{P}$-object $\frac{dP}{dt}$ we associate with every process $P \in$ Proc (Def. 3.3.14). As we shall soon see, this is not very different from specifying an Initial Value Problem.

However, as argued before, the dynamical evolution of a model cannot be specified compositionally unless we know more than just the trajectory. This extra information is provided by the $\mathbb{D}$-object ∂P we associate with every process $P \in$ Proc

(Def. 3.3.13). It is essentially an encoding of the Class 1 transitions of the constituent species enhanced with information about species concentrations. Hence, it quantitatively records the kinds of interaction the process can engage in.

Notation The elements of $\mathbb{P}$ and $\mathbb{D}$ are functions, and we sometimes use the λ notation to describe them. Thus for example $\lambda a.0$ is the origin of $\mathbb{P}$, $\lambda a.[\texttt{if } a \equiv A \texttt{ then 1 else 0}]$ is the basis vector $\mathbf{1}_A$ and $\lambda afx.[\texttt{if } a \equiv A \texttt{ and } f \equiv F \texttt{ and } x \in \{u, v\} \texttt{ then 1 else 0}]$ is the sum $\mathbf{1}_{(A,F,u)} + \mathbf{1}_{(A,F,v)}$. Similarly, we sometimes treat syntactic processes (i.e. elements of $\mathcal{P}$) as functions rather than terms and write for example $P(A)$ for the concentration of the prime species A in the process $P \in \mathcal{P}$.

If X is a multiset and x is its putative element, we write $\mathsf{card}(x, X)$ for the multiplicity of x in X (i.e. the number of times x appears in X). In addition, note that when we iterate over multisets—see e.g. Def. 3.3.11 immediately below—we visit every element as many times as it appears in the multiset. To differentiate between multisets and ordinary sets, we use the double brackets $\{\!|\cdots|\!\}$ for the former.

Definition 3.3.11. (the species embedding) The *species embedding* is the function $\langle - \rangle : \mathcal{S} \to \mathbb{P}$ defined by:

$$\langle A \rangle \stackrel{\text{df}}{=} \sum_{B \in \mathsf{primes}(A)} \mathbf{1}_B \tag{3.17}$$

Note that $\langle \mathbf{0} \rangle = 0$ and $\langle A | B \rangle = \langle A \rangle + \langle B \rangle$.

Definition 3.3.12. (the interaction tensor) Let M be an affinity network. The *interaction tensor* $- \mathbin{⦶_M} - : \mathbb{D} \times \mathbb{D} \multimap \mathbb{P}$ is defined as the bilinear extension of the following clause on basis vectors:

$$\mathbf{1}_{(A,F,x)} \mathbin{⦶_M} \mathbf{1}_{(B,G,y)} \stackrel{\text{df}}{=} \begin{cases} M(x,y) \times (\langle F \circ G \rangle - \mathbf{1}_A - \mathbf{1}_B) & x, y \in M \text{ and } F \downarrow G \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

The tensor takes two potentials (elements of $\mathbb{D}$) and returns a behaviour (element of $\mathbb{P}$) emerging from their interaction. As a consequence of the definition by (bi)linear extension, the result scales linearly with the arguments, and therefore the tensor reproduces the kinetic law of mass-action. The proportionality coefficient is the affinity of the two names mediating the given interaction, and thus name affinities correspond to reaction rate constants. Finally, observe that the interaction tensor is a *partial* function: there are potentials whose interaction as defined in above gives rise to infinite divergent sums. Possible improvements of the definitions of $\mathbb{P}$ and $\mathbb{D}$ are discussed towards the end of this dissertation (§7.2.2).

Definition 3.3.13. (interaction potential) Let $P \in \textsc{Proc}$ be a process. The *interaction potential of* P is the vector $\partial P \in \mathbb{D}$ defined by structural induction on P in the following way:

$$\partial(c \cdot A) \stackrel{\text{df}}{=} \lambda afx.[c \cdot \mathsf{card}(a \xrightarrow{x} f, \mathit{Trans}) \cdot \mathsf{card}(a, \mathsf{primes}(A))] \quad (3.19)$$

$$\partial(Q||R) \stackrel{\text{df}}{=} \partial Q + \partial R \quad (3.20)$$

Definition 3.3.14. (immediate behaviour) Let M be an affinity network and let $P \in \textsc{Proc}$ be a process. The *immediate behaviour of* P *in the context of* M is the vector $\frac{d_M P}{dt} \in \mathbb{P}$ defined inductively by the following clauses:

$$\frac{d_M(c \cdot A)}{dt} \stackrel{\text{df}}{=} \sum_{\substack{B \in \mathsf{primes}(A) \\ B \xrightarrow{\tau @ k} C}} \left(k \times c \times (\langle C \rangle - \mathbf{1}_B)\right) + \frac{1}{2} \times (\partial(c \cdot A) \oplus_M \partial(c \cdot A)) \quad (3.21)$$

$$\frac{d_M(P||Q)}{dt} \stackrel{\text{df}}{=} \frac{d_M P}{dt} + \frac{d_M Q}{dt} + \partial P \oplus_M \partial Q \quad (3.22)$$

Note that the interaction potential of a process does not depend on the global affinity network, but the immediate behaviour does; we always assume that a global affinity network has been defined and is fixed, and thus we do not mention it explicitly when discussing immediate behaviours.

The immediate behaviour $\frac{dP}{dt}$ associates with P the gradient of the flow line (system trajectory) at this point. The equations (3.21) and (3.22) reflect this interpretation: in (3.21) all τ-transitions are translated to vectors with positive contribution of transition targets (products) and negative contributions of sources (substrates), appropriately weighted and added together; the term $\frac{1}{2} \times (\partial(c \cdot A) \oplus_M \partial(c \cdot A))$ records the behaviour emerging from the interaction of two A molecules at the correct mass-action rate. In (3.22), the immediate behaviour of a composition of two processes is simply the sum of immediate behaviours of the components and the behaviour that emerges from their interaction.

Computing the interaction potential ∂P of a process is more straightforward: equation (3.19) lifts all the transitions involving concretions from the multi-transition system and weighs them with the concentrations of their sources, while (3.20) reflects the intuition that the interaction potential of a composition of processes is simply the sum of the interaction potentials of the components, with no cancellation or further emergent interaction.

The following two important results state that the decomposition into prime species is not necessary for the definition of immediate behaviour (Thm. 3.3.15), and that congruent processes have equal semantics (Thm. 3.3.16).

Theorem 3.3.15. For any species A, real number c and network M we have

$$\frac{d_M(c \cdot A)}{dt} = \sum_{A \xrightarrow{\tau@k} C} \big(k \times c \times (\langle C \rangle - \langle A \rangle)\big) \quad + \quad \frac{1}{2} \times (\partial(c \cdot A) \oplus_M \partial(c \cdot A)) \tag{3.23}$$

Proof. (sketch) Easy: every $\tau@k$ transition of A can be attributed to one of A's prime components (cf. Fig. 3.6), and the remaining components are present both in A and C, and therefore cancel each other out. □

Theorem 3.3.16. For every $P \equiv Q$, $\frac{dP}{dt} = \frac{dQ}{dt}$ and $\partial P = \partial Q$.

Proof. (sketch) Induction on the derivation of $P \equiv Q$, using bilinearity of the interaction tensor in the crucial case of the rule $c \cdot (A|B) \equiv (c \cdot A) \mid (c \cdot B)$. □

We conclude the presentation of semantics by deriving the interaction potential and the immediate behaviour of the process describing the initial state of the simple enzyme model.

Example The process describing the initial state of the example system is $c_1 \cdot E(e) \;||\; c_2 \cdot S(s) \;||\; c_3 \cdot P$; let us abbreviate this as Π. The global affinity network consists of just two names: e and s, able to communicate at the basal rate k_b; let us call it *Aff*. As usual, we write C for the species $(\boldsymbol{\nu} K)(a.E(e) \mid (u.S(s) + t.P))$ representing the enzyme-substrate complex.

We begin with the derivation of the interaction potential of Π:

$$\begin{aligned}
\partial\Pi &= \partial(c_1 \cdot E(e) \;||\; c_2 \cdot S(s) \;||\; c_3 \cdot P) && (3.24)\\
&= \partial(c_1 \cdot E(e)) + \partial(c_2 \cdot S(s)) + \partial(c_3 \cdot P) && (3.25)\\
&= \lambda afx.[c_1 \cdot \mathsf{card}(a \xrightarrow{x} f, \mathit{Trans}) \cdot \mathsf{card}(a, \mathsf{primes}(E(e)))]\\
&\quad + \lambda afx.[c_2 \cdot \mathsf{card}(a \xrightarrow{x} f, \mathit{Trans}) \cdot \mathsf{card}(a, \mathsf{primes}(S(s)))]\\
&\quad + \lambda afx.[c_3 \cdot \mathsf{card}(a \xrightarrow{x} f, \mathit{Trans}) \cdot \mathsf{card}(a, \mathsf{primes}(P))] && (3.26)\\
&= \lambda afx.[c_1 \cdot \mathsf{card}(a \xrightarrow{x} f, \mathit{Trans}) \cdot \mathsf{card}(a, \{\!|E(e)|\!\})]\\
&\quad + \lambda afx.[c_2 \cdot \mathsf{card}(a \xrightarrow{x} f, \mathit{Trans}) \cdot \mathsf{card}(a, \{\!|S(s)|\!\})]\\
&\quad + \lambda afx.[c_3 \cdot \mathsf{card}(a \xrightarrow{x} f, \mathit{Trans}) \cdot \mathsf{card}(a, \{\!|P|\!\})] && (3.27)\\
&= \lambda afx.[c_1 \cdot \texttt{if } a \equiv E(e) \texttt{ then } \mathsf{card}(E(e) \xrightarrow{x} f, \mathit{Trans}) \texttt{ else } 0]\\
&\quad + \lambda afx.[c_2 \cdot \texttt{if } a \equiv S(s) \texttt{ then } \mathsf{card}(S(s) \xrightarrow{x} f, \mathit{Trans}) \texttt{ else } 0]\\
&\quad + \lambda afx.[c_3 \cdot \texttt{if } a \equiv P \texttt{ then } \mathsf{card}(P \xrightarrow{x} f, \mathit{Trans}) \texttt{ else } 0] && (3.28)
\end{aligned}$$

$$\overset{\text{Fig. 3.2}}{=} \lambda afx.[\texttt{if } a \equiv E(e) \texttt{ and } f \equiv (\boldsymbol{\nu}K)(u,t;)a.E(e) \texttt{ and } x = e \texttt{ then } c_1 \texttt{ else } 0]$$
$$+ \lambda afx.[\texttt{if } a \equiv S(s) \texttt{ and } f \equiv (;x,y)(x.S(s) + y.P) \texttt{ and } x = s \texttt{ then } c_2 \texttt{ else } 0]$$
$$+ 0_{\mathbb{D}} \tag{3.29}$$
$$= \lambda afx.[\texttt{if } a \equiv E(e) \texttt{ and } f \equiv (\boldsymbol{\nu}K)(u,t;)a.E(e) \texttt{ and } x = e \texttt{ then } c_1$$
$$\texttt{else if } a \equiv S(s) \texttt{ and } f \equiv (;x,y)(x.S(s) + t.P) \texttt{ and } x = s \texttt{ then } c_2$$
$$\texttt{else } 0] \tag{3.30}$$

The above function can also be written as a linear combination of basis vectors:

$$\partial\Pi = c_1 \times \mathbf{1}_{(E(e),(\boldsymbol{\nu}K)(u,t;)a.E(e),e)} \; + \; c_2 \times \mathbf{1}_{(S(s),(;x,y)(x.S(s)+y.P),s)} \tag{3.31}$$

with the two components of the sum corresponding to $\partial(c_1 \cdot E(e))$ and $\partial(c_2 \cdot S(s))$, respectively, and $\partial(c_3 \cdot P) = 0_{\mathbb{D}}$.

In order to simplify the derivation of the immediate behaviour of Π, we first compute $\frac{d(c_3 \cdot P)}{dt}$ using the knowledge of the transitions of P obtained in the previous section:

$$\frac{d(c_3 \cdot P)}{dt} = \sum_{\substack{A \in \mathsf{primes}(P) \\ A \xrightarrow{\tau @ k} B}} \big(k \times c_3 \times (\langle B \rangle - \mathbf{1}_A)\big) \; + \; \frac{1}{2} \times (\partial(c_3 \cdot P) \oplus_{Aff} \partial(c_3 \cdot P)) \tag{3.32}$$
$$= \sum_{P \xrightarrow{\tau @ k} B} \big(k \times c_3 \times (\langle B \rangle - \mathbf{1}_P)\big) \; + \; \frac{1}{2} \times (0_{\mathbb{D}} \oplus_{Aff} 0_{\mathbb{D}}) \tag{3.33}$$
$$\overset{(3.8)}{=} k_d \times c_3 \times (\langle \mathbf{0} \rangle - \mathbf{1}_P) + 0_{\mathbb{P}} \tag{3.34}$$
$$= (-k_d \cdot c_3) \times \mathbf{1}_P \tag{3.35}$$

We ask the reader to convince themselves that $\frac{d(c_1 \cdot S(s))}{dt} = \frac{d(c_2 \cdot E(e))}{dt} = 0_{\mathbb{P}}$ and to recall from the previous derivations the values of $\partial(c_1 \cdot E(e))$, $\partial(c_2 \cdot S(s))$ and $\partial(c_3 \cdot P)$. The derivation of $\frac{d\Pi}{dt}$ follows:

$$\frac{d\Pi}{dt} = \frac{d(c_1 \cdot E(e) \;||\; c_2 \cdot S(s) \;||\; c_3 \cdot P)}{dt} \tag{3.36}$$
$$= \frac{d(c_1 \cdot E(e) \;||\; c_2 \cdot S(s))}{dt}$$
$$+ \frac{d(c_3 \cdot P)}{dt} + \partial(c_1 \cdot E(e) \;||\; c_2 \cdot S(s)) \oplus_{Aff} \partial(c_3 \cdot P) \tag{3.37}$$
$$= \frac{d(c_1 \cdot E(e))}{dt} + \frac{d(c_2 \cdot S(s))}{dt} + \partial(c_1 \cdot E(e)) \oplus_{Aff} \partial(c_2 \cdot S(s))$$
$$+ \frac{d(c_3 \cdot P)}{dt} + \partial(c_1 \cdot E(e) \;||\; c_2 \cdot S(s)) \oplus_{Aff} 0_{\mathbb{D}} \tag{3.38}$$
$$= (c_1 \times \mathbf{1}_{(E(e),(\boldsymbol{\nu}K)(u,t;)a.E(e),e)}) \oplus_{Aff} (c_2 \times \mathbf{1}_{(S(s),(;x,y)(x.S(s)+y.P),s)})$$
$$- (k_d \cdot c_3) \times \mathbf{1}_P + 0_{\mathbb{P}} \tag{3.39}$$
$$\overset{(3.11)}{=} (c_1 \cdot c_2 \cdot Aff(e,s)) \times (\langle C \rangle - \mathbf{1}_{S(s)} - \mathbf{1}_{E(e)}) - (k_d \cdot c_3) \times \mathbf{1}_P \tag{3.40}$$
$$= (k_b \cdot c_1 \cdot c_2) \times \mathbf{1}_C - (k_b \cdot c_1 \cdot c_2) \times (\mathbf{1}_{E(e)} + \mathbf{1}_{S(s)}) - (k_d \cdot c_3) \times \mathbf{1}_P \tag{3.41}$$

The gradient vector $\frac{d\Pi}{dt}$ derived above indicates that the system produces complexes at the rate $k_b \cdot c_1 \cdot c_2$; loses enzyme and substrate at the same rate ($k_b \cdot c_1 \cdot c_2$ again); and loses product at the rate $k_d \cdot c_3$. Note that no conversion of substrate to product occurs because no complexes are present in the initial state.

Figure 3.2 summarises the syntax and semantics of the enzyme model.

3.4 Extraction of Ordinary Differential Equations

We conclude this chapter by giving an algorithm for translating $c\pi$ models to sets of coupled Ordinary Differential Equations. Coupled ODEs are an established formalism for describing dynamical systems, and thus the ability to extract them from $c\pi$ models allows us to use the variety of ODE tools to study $c\pi$ models. Such translation algorithms exist for other process algebras [21], invariably exploiting the fact that the description of a system in a process algebra is decoupled from dynamics.

In the previous section we showed how to compute the gradient vector encoding the immediate behaviour of a particular $c\pi$ process. Differential equations can be seen as expressions enabling us to compute the gradient vector in *any* point of the process space, and their solutions are simply the flow lines of the resulting vector field on $\mathbb{P}$. As there is one differential equation per dimension (i.e. per prime species), the set of ODEs defining this vector field is infinite. Most systems of interest, however, visit only finitely many dimensions, and so it is important to produce a finite set of ODEs whenever possible. The initial state of the system determines if this is the case.

These intuitions lie at the heart of the algorithm for extracting Ordinary Differential Equations from $c\pi$ processes presented in Fig. 3.7. It abstracts from actual processes to *symbolic* ones, by substituting variables for actual concentrations. The procedure for computing the immediate behaviour $\frac{d-}{dt}$ applied to a symbolic process yields a vector of algebraic formulae rather than an actual vector in $\mathbb{P}$. Observe that this is exactly what we did in the previous section, where we used literals such as c_1, c_2 and c_3 rather than actual real numbers when computing the gradient vector $\frac{d\Pi}{dt}$ for the initial state of the enzyme model. If we are able to guarantee that the symbolic process mentions all the prime species ever encoun-

```
1   input Π;  // process
2
3   Π_s := x_0 · 0;
4   Γ := supp(Π);
5
6   while Γ ≠ ∅
7     enumerate Γ as {A_1, ..., A_n};
8     Π_s := Π_s || x_{A_1} · A_1 || ··· || x_{A_n} · A_n;
9     Π'_s := dΠ_s/dt;
10    Γ := supp(Π'_s) \ supp(Π_s);
11  endwhile
12
13  output {ẋ_A = Π'_s(A)}_{A∈supp(Π'_s)};  // equations
```

Figure 3.7: The ODE extraction algorithm.

tered in the system, the formulae obtained by the computation of its immediate behaviour are precisely the ODEs we seek.

The algorithm takes a $c\pi$ process Π as input and forms the initial null symbolic process Π_s. It also sets the set of freshly visited dimensions (prime species) Γ to the support of Π: these are the species that are present in the initial state of the system. The main loop of the algorithm extends the symbolic process Π_s with the mentions of newly visited dimensions (8), derives its symbolic immediate behaviour Π'_s (9) and sets the freshly visited dimensions to the set of new prime species appearing in Π'_s (10). The loop continues until no new dimensions are visited. Naturally, the algorithm terminates if and only if the system visits finitely many dimensions. If this is not the case, it is still possible to extract a finite set of ODEs from Π'_s at every iteration of the loop to obtain a sequence of finite ODE systems approximating the perfect infinite one. Finally, observe that compositionality of $c\pi$ semantics facilitates the computation of the symbolic immediate behaviour Π'_s in line (9): $(\frac{d\Pi_s}{dt})^{n+1}$ is by definition equal to $(\frac{d\Pi_s}{dt})^n + \frac{d(x_{A_1}\cdot A_1 \,||\, \cdots \,||\, x_{A_n}\cdot A_n)}{dt} + (\partial\Pi_s)^n \oplus \partial(x_{A_1} \cdot A_1 || \cdots || x_{A_n} \cdot A_n)$, with the superscripts giving the index of the iteration.

Example We derive ODEs for our example system. The initial process is $c_1 \cdot E(e) \;||\; c_2 \cdot S(s) \;||\; c_3 \cdot P$; thus, the initial Γ is $\{E(e), S(s), P\}$, and the first execution of the computation in line (9) is essentially the derivation of the immediate behaviour we have performed in the previous section; the result, with literals replaced by variables, is $(k_b \cdot x_{E(e)} \cdot x_{S(s)})\times\mathbf{1}_C - (k_b \cdot x_{E(e)} \cdot x_{S(s)})\times(\mathbf{1}_{E(e)} + \mathbf{1}_{S(s)}) - (k_d \cdot x_P)\times\mathbf{1}_P$. The only new dimension here is C, and so the loop restarts with $\Gamma = \{C\}$, and the computation

of the immediate behaviour is now

$$\frac{d\Pi_s}{dt} = \frac{d(x_{E(e)} \cdot E(e) || x_{S(s)} \cdot S(s) || x_P \cdot P || x_C \cdot C)}{dt} \tag{3.42}$$

$$= \frac{d(x_{E(e)} \cdot E(e) || x_{S(s)} \cdot S(s) || x_P \cdot P)}{dt} + \frac{d(x_C \cdot C)}{dt} + \partial(x_{E(e)} \cdot E(e) || x_{S(s)} \cdot S(s) || x_P \cdot P) \oplus \partial(x_C \cdot C) \tag{3.43}$$

$$= (k_b \cdot x_{E(e)} \cdot x_{S(s)}) \times \mathbf{1}_C - (k_b \cdot x_{E(e)} \cdot x_{S(s)}) \times (\mathbf{1}_{E(e)} + \mathbf{1}_{S(s)}) - (k_d \cdot x_P) \times \mathbf{1}_P + (k_u \cdot x_C) \times (\mathbf{1}_{E(e)} + \mathbf{1}_{S(s)} - \mathbf{1}_C) + (k_t \cdot x_C) \times (\mathbf{1}_{E(e)} + \mathbf{1}_P - \mathbf{1}_C) + 0 \tag{3.44}$$

As there are no new prime species mentioned by this process ($\Gamma = \varnothing$), the loop terminates, and the algorithm outputs the set of four differential equations—projections of (3.44) on each of the visited dimensions:

$$\dot{x}_{E(e)} = -k_b \cdot x_{E(e)} \cdot x_{S(s)} + k_u \cdot x_C + k_t \cdot x_C \tag{3.45}$$

$$\dot{x}_{S(s)} = -k_b \cdot x_{E(e)} \cdot x_{S(s)} + k_u \cdot x_C \tag{3.46}$$

$$\dot{x}_P = -k_d \cdot x_P + k_t \cdot x_C \tag{3.47}$$

$$\dot{x}_C = k_b \cdot x_{E(e)} \cdot x_{S(s)} - k_u \cdot x_C - k_t \cdot x_C \tag{3.48}$$

The reader should confront these equations with those in Fig. 3.1(l).

Summary

We have introduced the continuous π-calculus ($c\pi$), a process algebra for biochemical modelling. It is based on the π-calculus, but with two crucial differences: the connectivity of names is relaxed from the strict name-coname relation to arbitrary many-to-many *affinity networks*, and processes are given semantics in terms of real vector spaces. The concept of affinity network drives a lot of the further research presented in this dissertation. On the other hand, most of the semantic notions treated here in such detail are not revisited in the future chapters, where we rely mostly on extraction of ODEs to study our models; as we have seen, however, they are the heart of $c\pi$.

Chapter 4

Modelling a circadian clock

4.1 Introduction

In this chapter we use $c\pi$ to model a non-trivial, real-life biological system. The purpose of this exercise is three-fold. First and foremost, it is to convince the reader that $c\pi$ can indeed handle real world systems. Second, it is to showcase the features of $c\pi$ that set it apart from other process algebras for biochemical modelling, in particular the flexible wiring of agents in the affinity network. Finally, it is to make first steps towards modelling of evolutionary variability.

What we are *not* doing is attempting to discover new facts about the modelled system. The $c\pi$ representation constructed here is entirely based on a published ODE model and is not directly driven by actual biological data. Parts of the upcoming discussion may superficially resemble biological research writing, but it should not be mistaken for such; the purpose of this chapter is to learn about the modelling method, not the modelled system. Moreover, since we build on an existing representation, most—perhaps all—of the credit for the explanatory power of the $c\pi$ model rests with the authors of the original work.

As well as demonstrating the strengths of $c\pi$ and process algebras in the context of systems biology, this chapter also exposes their weaknesses. Their discussion is postponed until the end of the dissertation (§7.1.2).

4.1.1 Overview of the chapter

The chapter commences with a brief introduction to the biological system modelled here (§4.2). We then describe the model we base this work on (§4.3.1) and translate it into $c\pi$ (§4.3.2). Finally, we use the prototype $c\pi$ software tool (§4.4.1) to analyse the $c\pi$ model (§4.4.2) and some of its variants (§4.4.3), including evolutionary ones.

4.2 The system

The system we model here is the circadian clock of the cyanobacterium *Synechococcus elongatus*. For a gentle introduction to this system—albeit not a recent one—see [58]; the key research papers are [60, 73, 104, 105]; the recently published collection [36] contains useful surveys of different aspects of the system. Below we survey the research milestones and describe the clock in very general terms, hopefully providing just enough information so that the reader is able to follow the rest of the chapter.

A circadian clock is a molecular system which oscillates with an approximately 24-hour period, thus allowing the organism it is contained in to synchronise its behaviour with the time of day. In order to be recognised as a circadian clock, a molecular oscillator has to exhibit three features: the oscillation has to persist without external stimuli for at least 24 hours; the period of oscillation has to be stable under a range of temperatures (*temperature compensation*); and the oscillations have to adapt to the local time, as defined by the external light and darkness periods (*entrainment*).

For many years it was believed that only eukaryotes have circadian clocks. The discovery of the circadian rhythms in cyanobacteria in mid-80s [60] invalidated this view and started a whole new research field of bacterial circadian clocks [36]. It also changed the perspective on the evolutionary history of circadian rhythms—cyanobacteria are a phylum which is at least 2.5 billion year old—and uncovered a new class of molecular mechanism capable of acting as circadian oscillators.

The core oscillator of *S. elongatus* has been found to consist of a cluster of three proteins: *KaiA*, *KaiB* and *KaiC* [73] ("kai" means "cycle" in Japanese). KaiC molecules have phosphorylation sites; the oscillating quantity is the average phosphorylation level of KaiC. Furthermore, it is known that KaiC forms hexamers

while KaiA and KaiB form dimers (complexes of respectively 6 and 2 identical molecules). Remarkably, the oscillatory behaviour of the KaiABC system has been reproduced *in vitro* [104, 105] using a purified solution of Kai proteins, proving that it depends neither on transcriptional feedback nor on intracellular compartments. This feature sets it apart from other clocks and further cements its reputation as the oldest and simplest.

Despite great interest in the KaiABC system, its precise mechanism is not yet fully understood [36].

4.3 The model

4.3.1 The original model of van Zon et. al.

Jeroen van Zon *et. al.* proposed an elegant model of the KaiABC system [148] based on the key assumption of *allostery* of KaiC. A protein is allosteric if it can assume two distinct 3D shapes. The two postulated conformations of KaiC are called *active* and *inactive* and differ in their biochemical and kinetic properties. Specifically, van Zon *et. al.* assume that each individual KaiC hexamer has an inherent propensity to oscillate between 14 phosphorylation states (2 hexamer forms $\times$ 7 possible phosphorylation levels per hexamer), as shown in Fig. 4.1.

A further assumption is the catalytic activity of KaiA, promoting the phosphorylation of active forms of KaiC with preference for hexamers at low phosphorylation levels. The authors call this mechanism *differential affinity* and suggest it is responsible for synchronising the oscillation cycles of individual KaiC hexamers. Finally, KaiB is given the rôle of stabilising the inactive forms of KaiC. According to the model, inactive KaiC hexamers are prone to flipping back to their active form, prematurely ending their autonomous oscillation cycle. The modellers assume that two KaiB monomers can bind an inactive KaiC and prevent it from flipping; this complex is then further stabilised by binding two KaiA monomers, increasing the competition for KaiA between the active KaiC hexamers as a side effect.

These assumptions allow van Zon *et. al.* to write out 34 coupled ODEs describing the behaviour of every possible form of KaiC. For easy comparison with the $c\pi$ model, we do not track the concentrations of all 34 species, but use a simpler metric instead: the mean phosphorylation level of KaiC. It is defined as the fraction

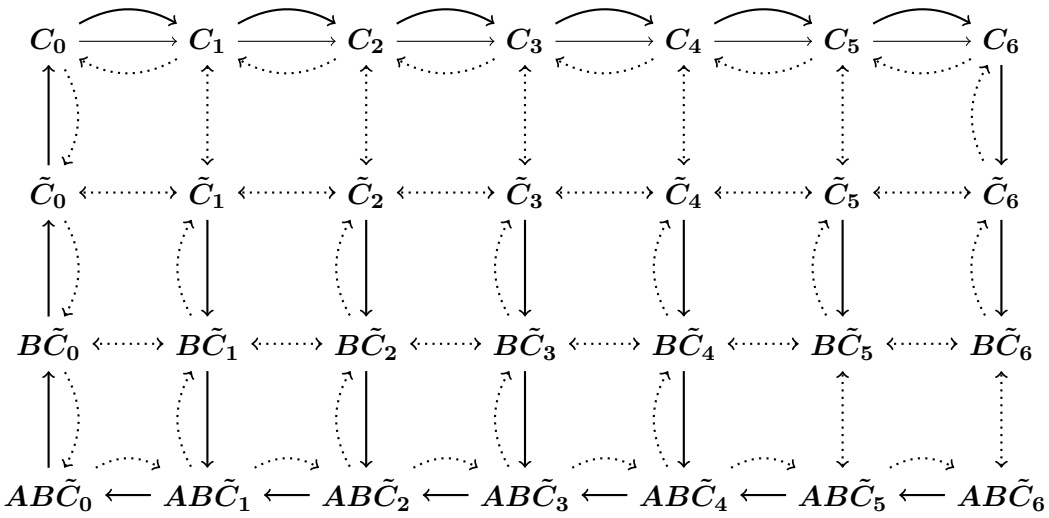


Figure 4.1: The state space of possible forms of KaiC according to [148], with arrow thickness giving the preference of transitions. The four rows of different configurations of KaiC hexamers are (top to bottom): active forms, inactive forms, KaiB-KaiC complexes and KaiA-KaiB-KaiC complexes. The transient complexes of KaiA and active KaiC are left out of this figure. Observe how every KaiC hexamer has an inherent tendency to oscillate.

of all phosphorylation sites that are switched on at any given time. The mean phosphorylation level as predicted by van Zon *et. al.* is graphed in Fig. 4.4(a, top).

The main merit of this model, according to its authors, is that it explains the oscillatory behaviour of the system without assuming any direct synchronisation between individual KaiC hexamers. From the point of view of $c\pi$ and this chapter the main appeal is the concept of differential affinity, which allows us to demonstrate the potential of our own notion of affinity networks.

4.3.2 The $c\pi$ translation

We recast the model discussed above in terms of the continuous π-calculus. As in the case of the running example in the previous chapter, the $c\pi$ representation has three parts: the definitions of species, the affinity network and the process. The complete model is presented in Fig. 4.2; we build it incrementally in this section.

Definitions of species We begin with the definitions of the species representing KaiA and KaiB. We assume each of them has one active site, termed a and b respectively. Upon interaction on this site (i.e. binding) the species awaits a further communication event (i.e. a command to unbind) on the freshly received name, after which it returns to its original state. This gives rise to the twin definitions:

$$A(a) \triangleq a(x).x.A(a) \tag{4.1}$$

$$B(b) \triangleq b(x).x.B(b) \tag{4.2}$$

There is no a priori reason why the definitions of KaiA and KaiB should be analogous. This is a result of our (implicit) decision to have KaiC drive its own state changes, rather than be driven by KaiA and KaiB: for example, the choice whether a KaiA-KaiC complex should dissociate with or without a KaiC phosphorylation event occurring is encoded within KaiC, not KaiA (cf. Eq. 4.3). Note however, that the species $A(a)$ and $B(b)$ are not identical nor equivalent in any formal sense: they have different free names, and the subsequent wiring of these names in the global affinity network causes A and B to behave quite differently.

For a KaiC hexamer with k phosphorylated sites ($k = 0 \ldots 6$) we have four species: the free active form C_k; the free inactive form $\tilde{C}_k$; the inactive form bound by two KaiB monomers $B\tilde{C}_k$; and the inactive form bound by two KaiB and two KaiA monomers $AB\tilde{C}_k$. It is possible to give just one definition for a k-fold phosphorylated hexamer, or indeed one to cover all 28 possible states of KaiC. There are benefits to doing so, for example the resulting strict correspondence of species definitions and gene products. Here we opt for a more extensive set of definitions instead, in order to keep the model more readable.

Before we proceed to define the KaiC species, there is one more issue to consider. Recall that we require every species definition to mention all the free names appearing in its body (Def. 3.2.4). The definitions we set up for KaiC are tightly linked (cf. Fig. 4.1) and one consequence is that each of them has in fact to mention every free name appearing in any of them. The vector of all free names in the 21 species that follow is going to be a long one, so we abbreviate it by σ for now. After we give all the definitions, we shall give σ directly and the reader should then be able to convince themselves that all the definitions indeed have to carry it around.

We now give the four definitions of C_3, $\tilde{C}_3$, $B\tilde{C}_3$ and $AB\tilde{C}_3$. The definitions of the species for the forms in different phosphorylation states (i.e. with a different subscript) are analogous, except for the boundary cases, which we discuss later. We start with C_3:

$$C_3(\sigma) \stackrel{\Delta}{=} (\boldsymbol{\nu} M_3)(\tau@k_{ps}.C_4(\sigma) + \tau@k_{dps}.C_2(\sigma) + \tau@\mathit{flip}_3.\tilde{C}_3(\sigma) + a_3\langle act\rangle.(u.C_3(\sigma) + r.C_4(\sigma))) \tag{4.3}$$

An active form of KaiC can therefore perform three spontaneous actions (phosphorylation, dephosphorylation and flip to the inactive form), or, assuming a_3 can react with a, bind a KaiA monomer and either undergo catalysed phosphorylation or unbind and return to the original state. The binding of KaiA is modelled in the usual way, by passing a fresh name $act \in M_3$ to the species $A(a)$—see Fig. 4.2 for the definition of M_3. Observe that this is in fact a concrete instance of the abstract enzymatic reaction modelled in the previous chapter.

The inactive form $\tilde{C}_3$ is rather similar:

$$\tilde{C}_3(\sigma) \stackrel{\text{df}}{=} (\boldsymbol{\nu} N)(\tau@\tilde{k}_{ps}.\tilde{C}_4(\sigma) + \tau@\tilde{k}_{dps}.\tilde{C}_2(\sigma) + \tau@\mathit{bflip}_3.C_3(\sigma) + \hat{b}_3\langle act_b\rangle.\hat{b}\langle \hat{act}_b\rangle.B\tilde{C}_3(\sigma, u_0, u, \hat{u})) \tag{4.4}$$

Here we model the binding of two KaiB molecules as two sequential communication events on the names $\tilde{b}_3$ and $\hat{b}$, both of which can communicate with b. Two fresh names, act_b and $\hat{act}_b$, are passed to two KaiB molecules, one to each; a communication event on any of these names models the unbinding of the corresponding monomer. Accordingly, u_0, u and $\hat{u}$, the names linked to act and $\hat{act}_b$ in the local affinity network N, are passed to $B\tilde{C}_3$ to be used for this purpose. Both u and u_0 can trigger the unbinding of the same KaiB monomer, but they do so at different rates (Fig. 4.2). The slow trigger u is used by $B\tilde{C}_1$ up to $B\tilde{C}_6$, while the fast one u_0, exclusively by $B\tilde{C}_0$; still, we pass u_0 to $B\tilde{C}_3$ because $B\tilde{C}_0$ can be reached from there. The differential unbinding of KaiB from the B-C complexes is a feature of the original model (Fig. 4.1).

Modelling of a three-substrate reaction as a pair of binary ones is a deviation from [148]. We are forced to do so because $c\pi$ admits only two-way communication. We demonstrate in due course that this modification does not alter the dynamics significantly.

Now, the definition of $B\tilde{C}_3$:

$$\begin{aligned} B\tilde{C}_3(\sigma, x_0, x, \hat{x}) \triangleq (\boldsymbol{\nu} K)(&\tau @ \tilde{k}_{ps}.B\tilde{C}_4(\sigma, x_0, x, \hat{x}) + \tau @ \tilde{k}_{dps}.B\tilde{C}_2(\sigma, x_0, x, \hat{x}) \\ &+ \hat{a}_3 \left\langle act_a \right\rangle .\hat{a} \left\langle \hat{act}_a \right\rangle .AB\tilde{C}_3(\sigma, x_0, x, \hat{x}, v, \hat{v}) \\ &+ x.\hat{x}.\tilde{C}_3(\sigma)) \end{aligned} \tag{4.5}$$

Again, we have the expected autonomous capabilities and the sequential binding of two KaiA molecules. This species can also use the last two names passed to it in order to unbind KaiB and return to the free inactive form. Only two triggers for the unbinding of KaiA (v and $\hat{v}$) have to be created and passed on this time, because the rate of unbinding is the same at each of the 7 phosphorylation stages.

The definition of $AB\tilde{C}_3$ follows:

$$\begin{aligned} AB\tilde{C}_3(\sigma, x_0, x, \hat{x}, y, \hat{y}) \triangleq\ &\tau @ \tilde{k}_{ps}.AB\tilde{C}_4(\sigma, x_0, x, \hat{x}, y, \hat{y}) \\ &+ \tau @ \tilde{k}_{dps}.AB\tilde{C}_2(\sigma, x_0, x, \hat{x}, y, \hat{y}) \\ &+ y.\hat{y}.B\tilde{C}_3(\sigma, x_0, x, \hat{x}) \end{aligned} \tag{4.6}$$

There is no further binding possible at this stage, so $AB\tilde{C}_3$ performs no name passing. Its capabilities are simply the autonomous (de-)phosphorylation and the unbinding of KaiA.

We now comment on the boundary cases. Obviously, C_0, $\tilde{C}_0$, etc. cannot be further dephosphorylated, so they have no autonomous dephosphorylation capability. Dually, the 6-fold phosphorylated species (C_6, $\tilde{C}_6$, etc.) have no phosphorylation capability. In the case of C_6 this also means that there is no local network M_6 and no a_6 site at all. Finally, because $B\tilde{C}_0$ binds KaiB more weakly than the other inactive forms of KaiC, it uses x_0 rather than x to trigger the unbinding.

We are now able to explicitly give σ. It is the catenation of the following vectors of names: $(a_i)_{i=0}^5$, the names used to bind KaiA on the active branch; $(\hat{a}_i)_{i=0}^6$ and $(\hat{b}_i)_{i=0}^6$, the names used to bind the first monomer during the double binding of KaiA or KaiB, respectively; and $(\hat{a}, \hat{b})$, the names used to bind the second.

The global affinity network The set of vertices of the global affinity network consists of a, b and the collection of names, previously called σ, that communicate

exclusively with either a or b, depending on whether they are responsible for KaiC's interactions with KaiA or KaiB. Hence, the global affinity network has the topology of two separated fans, with bases a and b. The affinities can be taken directly to be the appropriate rate constants, except for the affinities for the connections modelling the sequential binding of two KaiA and two KaiB molecules on the inactive side. Here, the affinity for the first binding (i.e. the first KaiA or the first KaiB to be bound) is set to the corresponding rate constant, while the affinity for the second—to a very high number, in order to create the effect of an immediate binding. We choose an arbitrary number $\hat{k}$ for this purpose, making sure it is a couple of orders of magnitude greater than the biggest rate constant (in what follows, $\hat{k} \stackrel{\text{df}}{=} 10^{20}$). However non-rigorous, such arbitrary manipulations are very common in dynamical modelling of biochemical systems, and are often necessary due to lack of actual parameter values or, as in our case, limitations of the particular modelling framework. Figure 4.4(a) vindicates our design.

The affinity network thus constructed is pictured in Fig. 4.2.

The process The $c\pi$ process representing the initial state of the system is simply

$$0.58 \cdot C_0(\sigma) \;||\; 0.58 \cdot A(a) \;||\; 1.72 \cdot B(b) \tag{4.7}$$

where the initial amounts are taken from [148].

We now proceed to execute our model and analyse the results.

4.4 The analysis

4.4.1 The $c\pi$ software tool

We have implemented a prototype software tool for automatic analysis of $c\pi$ specifications. The tool parses input files containing $c\pi$ models written in a human-readable syntax (Fig. 4.3). It then computes the multi-transition systems associated with all species in the file. Once all transitions are known, the program computes the immediate behaviour and interaction potentials of the model

$$
\begin{aligned}
C_0(\sigma) &\triangleq (\boldsymbol{\nu} M_0)(\tau@k_{ps}.C_1(\sigma) + \tau@flip_0.\tilde{C}_0(\sigma) + a_0\,\langle act\rangle\,.(u.C_0(\sigma) + r.C_1(\sigma)))\\
C_i(\sigma) &\triangleq (\boldsymbol{\nu} M_i)(\tau@k_{ps}.C_{i+1}(\sigma) + \tau@flip_i.\tilde{C}_i(\sigma) + a_i\,\langle act\rangle\,.(u.C_i(\sigma) + r.C_{i+1}(\sigma)) + \tau@k_{dps}.C_{i-1}(\sigma))\\
C_6(\sigma) &\triangleq \tau@flip_6.\tilde{C}_6(\sigma) + \tau@k_{dps}.C_5(\sigma)
\end{aligned}
$$

$$
\begin{aligned}
\tilde{C}_0(\sigma) &\triangleq (\boldsymbol{\nu} N)(\tau@\tilde{k}_{ps}.\tilde{C}_1(\sigma) + \tau@bflip_0.C_0(\sigma) + \hat{b}_0\,\langle act_b\rangle\,.\hat{b}\,\langle \hat{act}_b\rangle\,.B\tilde{C}_0(\sigma))\\
\tilde{C}_i(\sigma) &\triangleq (\boldsymbol{\nu} N)(\tau@\tilde{k}_{ps}.\tilde{C}_{i+1}(\sigma) + +\tau@bflip_i.C_i(\sigma) + \hat{b}_i\,\langle act_b\rangle\,.\hat{b}\,\langle \hat{act}_b\rangle\,.B\tilde{C}_i(\sigma,u_0,u,\hat{u}) + \tau@\tilde{k}_{dps}.\tilde{C}_{i-1}(\sigma))\\
\tilde{C}_6(\sigma) &\triangleq (\boldsymbol{\nu} N)(\tau@\tilde{k}_{dps}.\tilde{C}_5(\sigma) + \tau@bflip_6.C_6(\sigma) + \hat{b}_6\,\langle act_b\rangle\,.\hat{b}\,\langle \hat{act}_b\rangle\,.B\tilde{C}_6(\sigma,u_0,u,\hat{u}))
\end{aligned}
$$

$$
\begin{aligned}
B\tilde{C}_0(\sigma,x_0,x,\hat{x}) &\triangleq (\boldsymbol{\nu} K)(\tau@\tilde{k}_{ps}.B\tilde{C}_1(\sigma,x_0,x,\hat{x}) + x_0.\hat{x}.\tilde{C}_0(\sigma) + \hat{a}_0\,\langle act_a\rangle\,.\hat{a}\,\langle \hat{act}_a\rangle\,.AB\tilde{C}_0(\sigma,x_0,x,\hat{x},v,\hat{v}))\\
B\tilde{C}_i(\sigma,x_0,x,\hat{x}) &\triangleq (\boldsymbol{\nu} K)(\tau@\tilde{k}_{ps}.B\tilde{C}_{i+1}(\sigma,x_0,x,\hat{x}) + x.\hat{x}.\tilde{C}_i(\sigma) + \hat{a}_i\,\langle act_a\rangle\,.\hat{a}\,\langle \hat{act}_a\rangle\,.AB\tilde{C}_i(\sigma,x_0,x,\hat{x},v,\hat{v}) + \tau@\tilde{k}_{dps}.B\tilde{C}_{i-1}(\sigma,x_0,x,\hat{x}))\\
B\tilde{C}_6(\sigma,x_0,x,\hat{x}) &\triangleq (\boldsymbol{\nu} K)(\tau@\tilde{k}_{dps}.B\tilde{C}_5(\sigma,x_0,x,\hat{x}) + x.\hat{x}.\tilde{C}_6(\sigma) + \hat{a}_6\,\langle act_a\rangle\,.\hat{a}\,\langle \hat{act}_a\rangle\,.AB\tilde{C}_6(\sigma,x_0,x,\hat{x},v,\hat{v}))
\end{aligned}
$$

$$
\begin{aligned}
AB\tilde{C}_0(\sigma,x_0,x,\hat{x},y,\hat{y}) &\triangleq \tau@\tilde{k}_{ps}.AB\tilde{C}_1(\sigma,x_0,x,\hat{x},y,\hat{y}) + y.\hat{y}.B\tilde{C}_0(\sigma,x_0,x,\hat{x})\\
AB\tilde{C}_i(\sigma,x_0,x,\hat{x},y,\hat{y}) &\triangleq \tau@\tilde{k}_{ps}.AB\tilde{C}_{i+1}(\sigma,x_0,x,\hat{x},y,\hat{y}) + y.\hat{y}.B\tilde{C}_i(\sigma,x_0,x,\hat{x}) + \tau@\tilde{k}_{dps}.AB\tilde{C}_{i-1}(\sigma,x_0,x,\hat{x},y,\hat{y})\\
AB\tilde{C}_6(\sigma,x_0,x,\hat{x},y,\hat{y}) &\triangleq \tau@\tilde{k}_{dps}.AB\tilde{C}_5(\sigma,x_0,x,\hat{x},y,\hat{y}) + y.\hat{y}.B\tilde{C}_6(\sigma,x_0,x,\hat{x})
\end{aligned}
$$

$$
\begin{aligned}
A(a) &\triangleq a(x).x.A(a)\\
B(b) &\triangleq b(x).x.B(b)
\end{aligned}
$$

$$
\boldsymbol{\Pi} \triangleq \mathbf{0.58 \cdot C_0(\sigma)} \;||\; \mathbf{0.58 \cdot A(a)} \;||\; \mathbf{1.72 \cdot B(b)}
$$

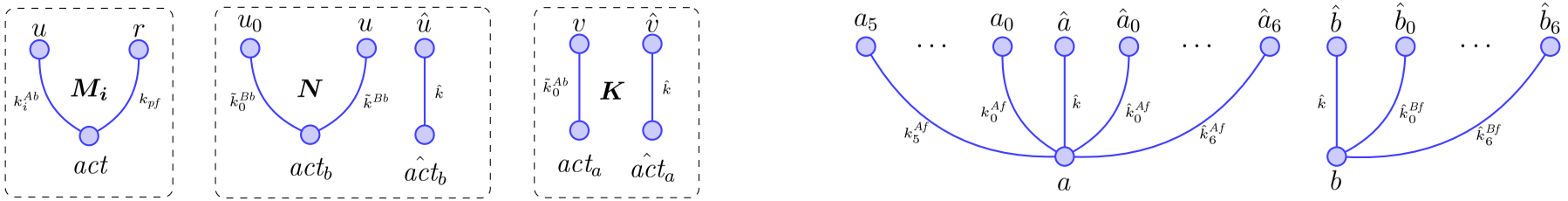


Figure 4.2: The $c\pi$ model of the KaiABC system: species definitions (top), including the local affinity networks (bottom left); the global affinity network (bottom right); and the process (bold).

```
1   species C3(sigma)
2   init 0.0;
3   network
4       site act, r, u;
5       react (act, r) @ kpf;
6       react (act, u) @ kAb(3);
7   end
8   body choice
9           tau<kps>.C4(sigma);
10          tau<kdps>.C2(sigma);
11          tau<flip(3)>.CC3(sigma);
12          a3[act;].choice
13                      r.C4(sigma);
14                      u.C3(sigma);
15                  end;
16      end
17  end
```

Figure 4.3: A sample from the $c\pi$ tool input file; compare with Eq. 4.3.

according to Defs. 3.3.13 and 3.3.14. Finally, using the precursor of the algorithm outlined in the preceding chapter (§3.4), it extracts the appropriate set of ODEs and writes them out in a format suitable for immediate numerical analysis with Octave [37, 108], the leading open source numerical computation software. In order to speed up Octave computations and make them more robust to changes in solver parameters, the $c\pi$ tool symbolically differentiates the ODEs and passes the Jacobian to the solver.

The $c\pi$ tool is written in Haskell [62]. The implementation follows the theory developed in the previous chapter quite closely (but see below). One technical detail that seems worth mentioning is the extensive memoization of transitions, ensuring that no transition is computed twice. This natural improvement over a naïve implementation led to speedups of up to three orders of magnitude.

The tool has at least one serious theoretical limitation: it recognises only a subset of the structural congruence of species. More precisely, although made aware of α-conversion in order to avoid name capture, it does not recognise α-convertible species as equivalent. Fortunately, this shortcoming does not matter for our case study. Future versions of the tool may overcome it by using the well-established de Bruijn indices [32] or building on nominal techniques such as FreshML [134].

4.4.2 The base model

The 30 species definitions and the initial state give rise to 64 prime $c\pi$ species. They are:

- The free KaiA and KaiB monomers, i.e. $A(a)$ and $B(b)$,
- The 7 active KaiC forms $C_0(\sigma), \ldots, C_6(\sigma)$,
- The 7 free inactive KaiC forms $\tilde{C}_0(\sigma), \ldots, \tilde{C}_6(\sigma)$
- The 6 KaiC-KaiA complexes, e.g. $(\boldsymbol{\nu} M_3)(act.A(a) \mid (u.C_3(\sigma) + r.C_4(\sigma)))$
- The 7 complexes of inactive KaiC and two KaiB monomers, e.g. $(\boldsymbol{\nu} N)(B\tilde{C}_3(\sigma, u_0, u, \hat{u}) \mid \hat{act_b}.B(b) \mid act_b.B(b))$,
- The 7 complexes of inactive KaiC, two KaiB and two KaiA monomers, e.g. $(\boldsymbol{\nu} K)(\boldsymbol{\nu} N)(AB\tilde{C}_3(\sigma, u_0, u, \hat{u}, v, \hat{v})|\hat{act_b}.B(b)|act_b.B(b)|\hat{act_a}.A(a)|act_a.A(a))$,
- 28 transitional species, representing complexes of inactive KaiC hexamers with either single KaiB monomer or two KaiB monomers and a single KaiA. Examples include $(\boldsymbol{\nu} N)(\hat{b}\left\langle \hat{act_b} \right\rangle .B\tilde{C}_3(\sigma, u_0, u, \hat{u}) \mid act_b.B(b))$ and $(\boldsymbol{\nu} K)(\boldsymbol{\nu} N)(\hat{v}.B\tilde{C}_3(\sigma, u_0, u, \hat{u}) \mid \hat{act_a}.A(a) \mid \hat{act_b}.B(b) \mid act_b.B(b))$.

Based on this set of species, a multi-transition system is computed, consisting of 64 states and 138 transitions. Finally, the tool outputs a system of 64 coupled ODEs and a symbolic representation of their Jacobian. The ODEs are similar to those found in [148], with all differences due to the sequential, rather than simultaneous, binding of KaiA and KaiB to the inactive forms of KaiC. In order to demonstrate that these differences do not alter the dynamics of the model, we juxtapose in Fig. 4.4(a) the original graph from [148] with one generated by Octave from the $c\pi$ model.

4.4.3 Perturbation experiments

So far we have done a fair amount of work to obtain a dynamical model that matches an arguably simpler one: a set of coupled ODEs. The difference is that in the case of an ODE model, the model and the dynamics are the same thing. In contrast, the $c\pi$ representation allows us to *derive* the dynamics. We now take advantage of this fact by generating models of several variants of the KaiABC system with simple manipulations of the base $c\pi$ model. To achieve the same

goal with ODEs, one would in principle have to rewrite them completely, or at least carefully alter them by hand.

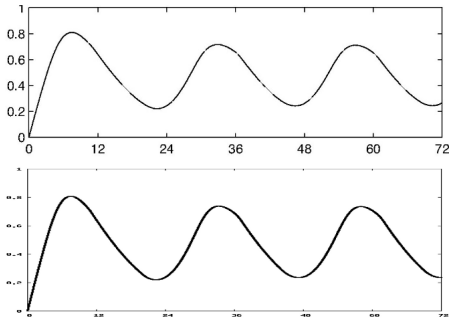
Stabilisation of the inactive branch Let us consider the situation where the KaiB molecules are no longer allowed to bind the inactive KaiC hexamers. There is therefore no stabilisation of the inactive branch and the competition for KaiA between the active KaiC hexamers is seriously weakened, because no KaiA is sequestered on the inactive branch either. The introduction of this perturbation to the $c\pi$ model is simple: we sever all connections to the free name b in the global affinity network. The now-superfluous species definitions like $B\tilde{C}_k$ or $AB\tilde{C}_k$ may also be discarded, but this is by no means necessary.

As expected, this modification completely destroys the clock (Fig. 4.4(b)). Further investigation reveals that C_6 acts as an absorbing state for KaiC, which is again something that should be expected given the instability of the inactive branch and the overabundance of KaiA.

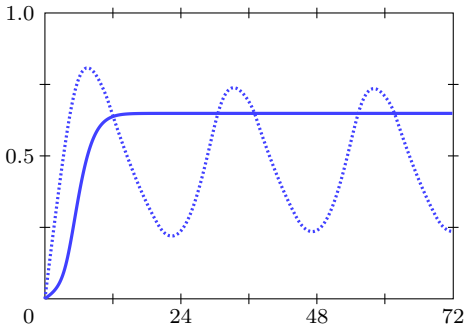
Synchronisation In order for the average phosphorylation level of KaiC to oscillate, individual hexamers have to oscillate in phase. The main mechanism responsible for the synchronisation of KaiC hexamers proposed in [148], and hence in our base model, is *differential affinity*: KaiA, the phosphorylating agent, has a greater affinity for weakly phosphorylated KaiC hexamers. We expect therefore that the removal of this feature results in a broken clock again. We test it by setting the affinities between a and a_0 through a_6 to a constant, intermediate value $k^{\mathrm{Ab}} \stackrel{\mathrm{df}}{=} k_3^{Ab}$.

Somewhat surprisingly, the oscillatory behaviour of the system remains almost unchanged (Fig. 4.4(c)). It appears that another synchronisation mechanism is at work. One obvious candidate is the differential unbinding of KaiB on the inactive branch. We can test this hypothesis by making $B\tilde{C}_0$ use x instead of x_0 to unbind the first KaiB monomer. The average phosphorylation level in this variant is graphed in Fig. 4.4(d). The oscillations are now less pronounced and their period appears to be less stable, but they are clearly still present.

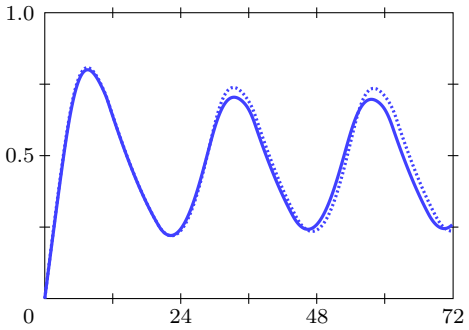
We conclude that neither the differential affinity nor the differential unbinding of KaiB is *the* synchronisation mechanism. This is further supported by the fact that knocking out both these features simultaneously still does not break the oscillations (not shown). It is possible that there is no synchronisation mechanism



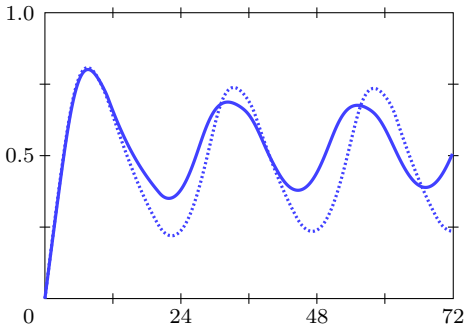
(a) The avg. phosph. level according to [148] (top) and the base $c\pi$ model (bottom).



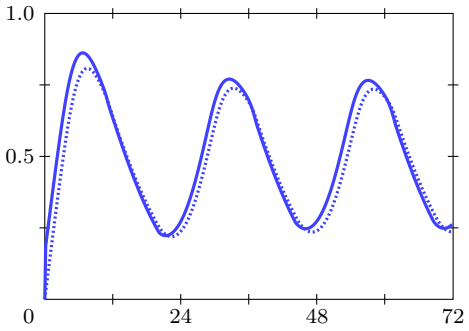
(b) The clock does not function when the inactive branch is not stabilised.



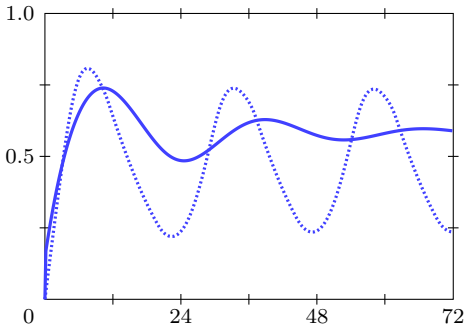
(c) The differential affinity is not necessary for sustained oscillations.



(d) The differential unbinding of KaiB does not fully explain oscillations either.



(e) Duplication of KaiB has no major effect on the behaviour of the system.



(f) The evolution of weak affinity for KaiA by a copy of KaiB breaks the clock.

Figure 4.4: Validation (a) of the $c\pi$ model of the KaiABC system, and computational investigation of some of its aspects: stabilisation of the inactive branch (b); synchronisation of individual KaiC hexamers (c)&(d); and a potential evolutionary development (e)&(f). All graphs show the average phosphorylation of KaiC over 72 hours; the dotted line in (b)–(f) gives this level for the base model as reference.

as such, but the KaiC hexamers oscillate in phase because they all start in the same state (C_0). This hypothesis can in principle be easily tested by considering the process

$$0.083 \cdot C_0(\sigma) \;||\; 0.083 \cdot C_1(\sigma) \;||\; 0.083 \cdot C_2(\sigma) \;||\; 0.083 \cdot C_3(\sigma) \;||\; 0.083 \cdot C_4(\sigma) \;||$$
$$0.083 \cdot C_5(\sigma) \;||\; 0.083 \cdot C_6(\sigma) \;||\; 0.58 \;\cdot A(a) \;\;||\; 1.72 \cdot B(b) \tag{4.8}$$

where the initial amount of KaiC is equally spread over all 7 phosphorylation states. Unfortunately the resulting ODEs appear to be numerically unstable, and we have not succeeded in our attempts to solve them, which points to the need of developing a simulation algorithm for $c\pi$ models (§7.2).

Evolutionary variability Now let us examine the following evolutionary scenario: suppose that the gene coding for KaiB is duplicated and subsequently the product of one of the copies acquires the ability to weakly bind KaiA. We further postulate that the resulting complex does not have any special function, but a KaiA molecule is unable to engage in any interaction while it is bound. We are interested in knowing whether the oscillatory behaviour is preserved along this short evolutionary trajectory and thus manipulate and solve the $c\pi$ model accordingly.

The first step is the duplication of KaiB. In $c\pi$ terms this corresponds to a duplication of the corresponding species definition; in addition to the definitions in Fig. 4.2(a), we now have also

$$B'(b') \stackrel{\text{df}}{=} b'(x).x.B'(b') \tag{4.9}$$

where b' is a new free name with precisely the same connectivity as b. After adding it to the global affinity network and extending the $c\pi$ process to include $B'(b')$ thusly

$$0.58 \cdot C_0(\sigma) \;||\; 0.58 \cdot A(a) \;||\; 1.72 \cdot B(b) \;||\; 1.72 \cdot B'(b') \tag{4.10}$$

the model can be processed and executed just like the base one. The result, displayed in Fig. 4.4(e), suggests that duplication of *kaiB* preserves the oscillations.

The second step is the evolution of weak affinity for KaiA by one of the copies of KaiB. Without loss of generality, we choose the original KaiB gene to do this step. The evolution of the binding of the kind we discuss means in $c\pi$ terms that $B(b)$ should be replaced by

$$B(b,c) \stackrel{\text{df}}{=} (\boldsymbol{\nu} y \overset{k_-}{—} z)(b(x).x.B''(b,c) + c\langle z\rangle\,.y.B(b,c)) \tag{4.11}$$

where the new free name c is connected to a with the affinity $k_+ \stackrel{\text{df}}{=} k_3^{Af}/10$ (arbitrary low value), and $y \overset{k}{—} z$ is the rendering of the 2-point affinity network consisting of sites y and z which can communicate at the rate $k_- \stackrel{\text{df}}{=} k_3^{Ab}$ (arbitrary normal strength unbinding rate). Furthermore, all invocations of the form $B(b)$—but, importantly, not $B'(b')$—have to be updated to use the new handle and parameter.

We readily verify that the presence of B'' severely disrupts the clock (Fig. 4.4(f)).

Summary

We have shown that $c\pi$ can handle real-world biochemical systems and demonstrated the process of modelling with $c\pi$. Importantly, we are able to conclude that the effort of providing an algebraic presentation of a biological system pays off with the ease of model perturbation experiments. We have also made our first attempt at the modelling of evolutionary events with $c\pi$.

Chapter 5

Variation operators

5.1 Introduction

This chapter is devoted to the definition of special syntactical modifications of $c\pi$ models called *variation operators*. Each operator is a model transformation scheme corresponding to a potential evolutionary change of the modelled system. Variation operators play a central rôle in this dissertation. Recall that in §2.4.3 we introduced a general abstract setting for studying and quantifying evolutionary neutrality and related concepts, based on the notion of neutral space. The required elements were: a set of genotypes equipped with an accessibility relation, a set of phenotypes and a genotype-phenotype map. In our framework, $c\pi$ models play the part of genotypes and their dynamical behaviours are phenotypes. The mathematical definition of behaviour, given by Defs. 3.3.13 and 3.3.14 (or, in more practical terms, by the ODE extraction algorithm in Fig. 3.7) is the genotype-phenotype map. Hence, the only missing ingredient is the accessibility structure of the genotype space. It is provided in this chapter: a $c\pi$ model is accessible from another if there is a variation operator that transforms the latter into the former. In this way we complete the recasting of the framework of neutral spaces in process-algebraic terms. First, however, we need to address several conceptual and technical difficulties this general idea raises.

5.1.1 Key issues and design choices

Events vs effects Modelling of biological systems with process algebras takes place at the level of molecular interaction networks. Other levels of abstraction or application domains are of course possible, but are not considered in this thesis. However, the mutational events that we want to model by applications of variation operators to $c\pi$ models take place at the level of the genetic code. To say that the mapping between this code and protein networks is non-trivial would be an understatement. Ignoring the complexity of this translation entirely—as we are forced to do here because $c\pi$ models have no notion of genome—comes at a price, namely blurring the important distinction between mutations and their effects.

In some cases this distinction is immaterial, for example gene duplication (a mutation) results essentially in duplication of the corresponding part of the network (an effect). More often, however, there is no clear correspondence between classes of mutations and classes of effects. Consider, for example, a single nucleotide substitution (point mutation): it may have no effect on the network at all, it may alter the binding affinities of a protein's active site, it may up- or down-regulate a protein or it may disable it entirely. Any of these effects can also be produced by a mutational event other than a point mutation. One consequence of considering effects rather than events, therefore, is the distortion of the true accessibility relation of genotypes (and hence evolutionary trajectories in the phenotype space) in a potentially very significant way; moreover, we lose the ability to investigate those neutral genetic changes that do not alter the structure of the network at all.

On the other hand, molecular networks are a fairly low level of abstraction for complex biological phenomena, with a clear connection to the genome, and there are reasons to believe that patterns uncovered by *in silico* evolution of models of protein networks have evolutionary significance (cf. [6, 136–138, 151]). Process algebraic models of these networks offer even more promise thanks to their focus on agents rather than the function they perform, their capacity to capture concurrent and emergent behaviour in a modular way, and, above all, their inherent ability to decouple genetic (i.e. syntactic) variation from its phenotypic (i.e. semantic) consequences. At the risk of belabouring the point, we note that neither sets of biochemical reactions, nor systems of differential equations, nor even Petri Nets enjoy all of these advantages simultaneously.

The unbearable expressiveness of π The π-calculus, and hence $c\pi$, are very expressive languages and it is easy to write models that do not have any direct biological meaning, at least not under the process-as-molecule interpretation. As the variation operators modify the *syntax* of $c\pi$ models, it is of utmost importance to ensure that the existing biological *meaning* is preserved in the process. The (admittedly imperfect) solution adopted here is two-fold: first and foremost, we require that modellers represent the biological reality with $c\pi$ in a particular way. Specifically, we ask that:

(i) All free names correspond to protein interaction sites; sites encoded by different regions of DNA, even if functionally equivalent or evolutionarily related, are modelled by different free names.

(ii) Only restricted names are passed and all name passing results in scope extrusion; in all cases this models formation of complexes.

(iii) All prime species present in the model correspond to actual molecular species. Consequently, all definitions of species correspond to states of molecules.

(iv) All species present in the support of the process are definition invocations, unless they model protein complexes.

Observe that because the above requirements relate the model to biological reality, they cannot be made formal.

In addition, we endeavour to design the operators in such a way that it is intuitively clear that the variants they produce possess a biological interpretation consistent with that of the original model. We regard this to be the most important guiding principle in the design of variation operators. Many sensible operators are not implemented for the sole reason that they might yield a model whose biological interpretation is unclear.

To give a concrete example of the issues discussed here, consider the elaborate private name passing schemes in the KaiABC model from the previous chapter: while it does have a sound biological interpretation there, it is conceivable to use exactly the same constructs to make an otherwise biologically relevant model meaningless. Therefore, no variation operators shall introduce such complicated name passing. On the other hand, the KaiABC model itself is properly grounded in biology and thus is a valid starting point for the operator-based analysis. Furthermore, there are no reasons to believe that a functionally equivalent, but

syntactically simpler model of the KaiABC system could not be produced by variation operators acting on a different $c\pi$ system.

Few operators, many variants Only a handful of variation operators are given in this chapter and they by no means exhaust the variety of mutational effects observed in nature. Some of the omitted effects are simply not expressible in the process-algebraic context; some are not of the same importance as the ones that are implemented; and formalisation of others would yield an unwieldy theory or require compromising on the conservative design principle outlined in the previous paragraph. In short, the selection offered here attempts to strike a balance between a proof-of-concept development and a fully fledged process-algebraic treatment of mutational effects. The number of operators, however, does not affect the number of possible variants of a given model. In fact, several operators are parametrised by elements of infinite sets, and thus a single model may have infinitely many variants. Development of general mechanisms for effective and meaningful sampling and traversal of the resulting infinite and dense variation space, however, is not covered by this thesis.

Operators as rules In this chapter, $c\pi$ models are rendered as tuples of the form $(\mathcal{D}, N, P)$, where $\mathcal{D}$ is a set of species definitions, N is an affinity network and P is a $c\pi$ process. These elements should satisfy the usual necessary conditions to constitute a well-defined model: N should contain all the free names of P, all of the definitions invoked in P should be defined in $\mathcal{D}$ and $\mathcal{D}$ itself should be a set of well-formed definitions. The variation operators are formalised as inference rules, inducing a binary accessibility relation "$\longrightarrow$" on the set of $c\pi$ models (tuples). The successor model is always obtained via a syntactic manipulation of the original one, and its biological interpretation can be recovered by applying the rules (i)–(iv) above.

As a very simple example, consider the following operator:

$$\frac{}{(\mathcal{D}, N, P) \longrightarrow (\mathcal{D} \cup \{Z() \triangleq \mathbf{0}\}, N, P)}\,\text{GENE-NEW} \tag{5.1}$$

This rule states that any model can be extended with a new species definition. By applying (iii) to the rhs of the "$\longrightarrow$" symbol—observe that (i),(ii) and (iv) do not offer any relevant information here—we conclude that a new, non-functional protein was added to the system, but is not present in its initial state. Thus, the rule can be seen as a formalisation of the emergence of a new, non-functional,

not expressed gene. Throughout this chapter we assume that any newly introduced names and definition handles are fresh for the existing ones. Hence, the condition $Z\#\mathcal{D}$ was not mentioned in the above rule.

None of the operators we introduce here have been implemented in software. The evaluation of our approach is performed in the next chapter by manual application of operators (§6.2) and ad-hoc scripting (§6.3). We find it therefore necessary to stress that proper implementation of variation operators is not only feasible, but quite likely relatively straightforward thanks to their definition as inference rules.

Physical and virtual sites Names in $c\pi$ models are of two kinds: they either model physical interaction sites, or are used for internal synchronisation within a complex and do not correspond directly to any physical entity. The following definition makes this distinction formal at the level of parameters of species definitions. Names of the first type are termed *physical sites*, names of the other type are *virtual sites*:

Definition 5.1.1. Let $\mathcal{M} = (\mathcal{D}, N, P)$ be a $c\pi$ model. We inductively define the set $\mathsf{vs}_\mathcal{M} \subset \mathcal{N} \times \mathcal{H}$ as the smallest set satisfying the following:

(i) If $(A(x_1, \ldots, x_n) \stackrel{\Delta}{=} B) \in \mathcal{D}$, and $A(a_1, \ldots, a_n) \subset P$ and for some j this occurrence of a_j is bound in P, then $(x_j, A) \in \mathsf{vs}_\mathcal{M}$,

(ii) If $(A(x_1, \ldots, x_n) \stackrel{\Delta}{=} B), (C(\vec{y}) \stackrel{\Delta}{=} D) \in \mathcal{D}$, and $A(a_1, \ldots, a_n) \subset D$ and for some j this occurrence of a_j is bound in D, then $(x_j, A) \in \mathsf{vs}_\mathcal{M}$,

(iii) If $(A(x_1, \ldots, x_n) \stackrel{\Delta}{=} B), (C(y_1, \ldots, y_m) \stackrel{\Delta}{=} D) \in \mathcal{D}$, $A(a_1, \ldots, a_n) \subset D$ and for some k, j we have $a_j = y_k$ and $(y_k, C) \in \mathsf{vs}_\mathcal{M}$, then $(x_j, A) \in \mathsf{vs}_\mathcal{M}$ as well.

When $(x, A) \in \mathsf{vs}_\mathcal{M}$, we say that x *is a virtual site of* A. When $(x, A) \notin \mathsf{vs}_\mathcal{M}$, but x is an argument of the definition A, we say that x is a *physical* site of A. We write $\mathsf{vs}_\mathcal{M}(A)$ for the set of virtual sites of A, and $\mathsf{ps}_\mathcal{M}(A)$ for the set of physical ones.

The first clause identifies bound names that are passed in the process as parameters to an invocation of a species definition. The second does the same with invocations in bodies of definitions. The third (inductive) clause makes sure that if a formal parameter y_k of a species definition C is identified as virtual, then so are the corresponding formal parameters of the definition A, if they use y_k as an actual parameter in the body of C.

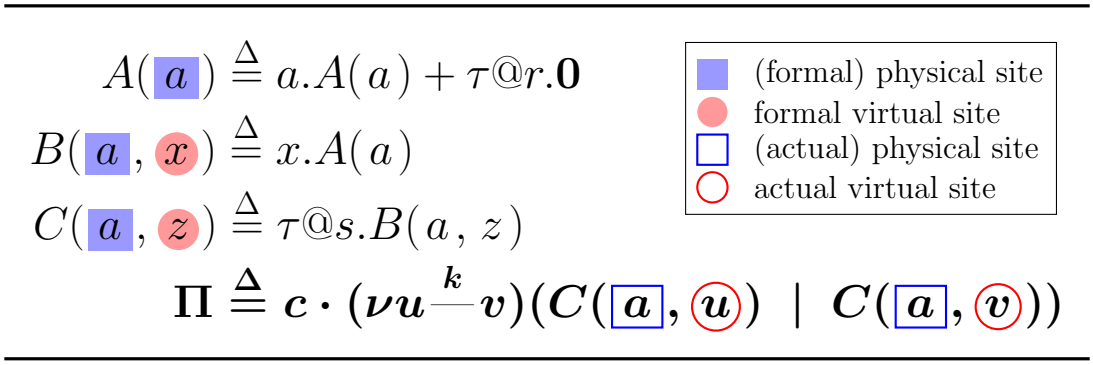


Figure 5.1: This example system models a complex of two C molecules, which can independently transform to B, after which they synchronously flip to an A form; A is capable of repeated interaction on a and of spontaneous degradation. The sites used to synchronise the intra-complex flip are virtual and a is physical. Observe that the distinction between formal and actual physical sites is superfluous. What names should the global affinity network of this model contain?

In general, it *does* make biological sense to invoke the same definition with different free names passed as actual physical sites: this can in principle be useful for tackling the combinatorial explosion of species. However, if combinatorial explosion is a major concern for the system under consideration, then rule-based modelling formalisms such as κ [30] or LBS [114] are a much better choice (§2.3.4). Hence, for the purposes of the evolutionary analysis techniques developed here, we shall assume that in any given model only one fixed free name is passed as an actual physical site to any given definition. An immediate consequence of this fact is that this name can be used as the canonical *formal* physical site in that definition, and we shall assume that this is the case as well. Thus, whenever we have a model $(\mathcal{D}, N, P)$ and $(A(\vec{x}) \triangleq B) \in \mathcal{D}$ and $\vec{x}$ are physical sites of A, we know that $\vec{x} \subseteq N$; furthermore, whenever an invocation $A(\vec{y})$ occurs, be it in P or a body of a definition in $\mathcal{D}$, it follows that $\vec{y} = \vec{x}$. The same applies even if in addition to $\vec{x}$, A has virtual parameters as well, although no similar inferences can be made about these. Figure 5.1 illustrates the concepts of physical and virtual sites with the help of a simple example.

5.1.2 Overview of the chapter

This section discussed important aspects of the design of variation operators. Section 5.2 contains preliminary technical definitions required for the development of variation operators. Three major groups of operators: gene-level events, state variation and rate changes are defined in §5.3–§5.5, respectively.

5.2 Preliminary definitions

Each definition in this section is accompanied with a brief comment on its intended meaning and use. While this section probably is not the most entertaining read to be found in this thesis, it is relied upon heavily by the remainder of the chapter. Most notions are defined in the context of a particular $c\pi$ model or set of species definitions, and thus their symbols carry subscripts like $-_{\mathcal{M}}$ or $-_{\mathcal{D}}$; these subscripts are omitted whenever possible.

Definition 5.2.1. (dependence of definitions) Let ξ, ζ be species definitions. We write $\xi \circ\!\!\rightarrow \zeta$ and say that ξ *depends on* ζ to indicate that the handle of ζ is mentioned in the body of ξ. When $\mathcal{D}$ is a set of definitions and $\xi \in \mathcal{D}$, we write $[\xi]_{\mathcal{D}}$ for the equivalence class of ξ w.r.t. to the smallest equivalence relation containing $\circ\!\!\rightarrow$. Moreover, when A is a species, we write $A \circ\!\!\rightarrow \zeta$ if ζ is invoked by A.

According to the informal assumptions outlined in §5.1.1, if two definitions are related via the equivalence generated by $\circ\!\!\rightarrow$, then both model states of the same molecule. The converse is not true in general: if a molecular species can exist in many different forms, but no transformation between these forms is possible, then there are multiple equivalence classes of definitions in the corresponding $c\pi$ model. The modeller should be aware, however, that because this case is formally indistinguishable from the more common one of many genuinely unrelated molecules, the variation operators treat it as the latter.

Definition 5.2.2. (substitution of definition handles) Let A be a species and let W, V be two definition handles. We write $A[W \mapsto V]$ to denote the species arising from A, where all occurrences of W are replaced by V. More precisely:

$$U(\vec{x})[W \mapsto V] \stackrel{\text{df}}{=} U(\vec{x}) \qquad \text{if } U \neq W \tag{5.2}$$

$$W(\vec{x})[W \mapsto V] \stackrel{\text{df}}{=} V(\vec{x}) \tag{5.3}$$

$$\mathbf{0}[W \mapsto V] \stackrel{\text{df}}{=} \mathbf{0} \tag{5.4}$$

$$((\boldsymbol{\nu} N)A)[W \mapsto V] \stackrel{\text{df}}{=} (\boldsymbol{\nu} N)(A[W \mapsto V]) \tag{5.5}$$

$$(\Sigma_{i=0}^{n} \pi_i.A_i)[W \mapsto V] \stackrel{\text{df}}{=} \Sigma_{i=0}^{n} \pi_i.(A_i[W \mapsto V]) \tag{5.6}$$

$$(A \mid B)[W \mapsto V] \stackrel{\text{df}}{=} A[W \mapsto V] \mid B[W \mapsto V] \tag{5.7}$$

Simultaneous substitution of multiple handles—denoted $A[\vec{W} \mapsto \vec{V}]$ where $\vec{W}$ and $\vec{V}$ are vectors of handles—is allowed provided that $\vec{W}$ contains no duplicates and $\vec{V}$ consists of fresh handles only.

Substitution of handles is used to define the variation operator modelling gene duplication.

Definition 5.2.3. (modifications of affinity networks) Let N be an affinity network, $a \in N$, $x \notin N$ and $f\colon N \to \mathbb{R}_{\geq 0}$. We define three affinity networks: $N \ominus a$, $N \oplus_f x$ and $N \odot_f a$, whose sets of vertices are $N \setminus \{a\}$, $N \cup \{x\}$ and N, respectively. The affinities of the three networks are given by:

$$(N \ominus a)(u, v) \stackrel{\text{df}}{=} N(u, v) \tag{5.8}$$

$$(N \oplus_f x)(u, v) \stackrel{\text{df}}{=} \begin{cases} N(u, v) & u, v \in N \\ f(v) & v \in N, u = x \\ f(u) & u \in N, v = x \\ 0 & u = v = x \end{cases} \tag{5.9}$$

$$(N \odot_f a)(u, v) \stackrel{\text{df}}{=} \begin{cases} N(u, v) & u \neq a, v \neq a \\ f(v) & u = a, v \neq a \\ f(u) & v = a \end{cases} \tag{5.10}$$

Furthermore, we introduce the following abbreviation:

$$N \oplus_a x \stackrel{\text{df}}{=} N \oplus_{\lambda z.N(a,z)} x \tag{5.11}$$

and vectorise is as follows:

$$N \oplus_{(a_1,\ldots,a_n)} (x_1, \ldots, x_n) \stackrel{\text{df}}{=} (N \oplus_{(a_1,\ldots,a_{n-1})} (x_1, \ldots, x_{n-1})) \oplus_{a_n} x_n \tag{5.12}$$

Finally, we set

$$N \ominus X \stackrel{\text{df}}{=} (N \ominus (x_1, \ldots, x_{n-1})) \ominus x_n \tag{5.13}$$

where $X = \{x_1, \ldots, x_n\} \subset N$. Note that this definition is insensitive to the choice of a particular ordering of X.

In short, $N \ominus X$ is the restriction of the affinity network to $N \setminus X$; $N \oplus_f x$ adds a single name x to the network N, and the affinities of the new name to the existing ones are given by f; finally, $N \odot_f a$ reconfigures the existing site a by changing all of its affinities according to f. The construction $N \oplus_{\vec{a}} \vec{x}$ extends the network N with exact duplicates of the names $\vec{a}$. These definitions are used whenever the global affinity network of a model needs to be reconfigured; for example, gene duplication usually requires addition of new names, while gene loss may lead to pruning of the network.

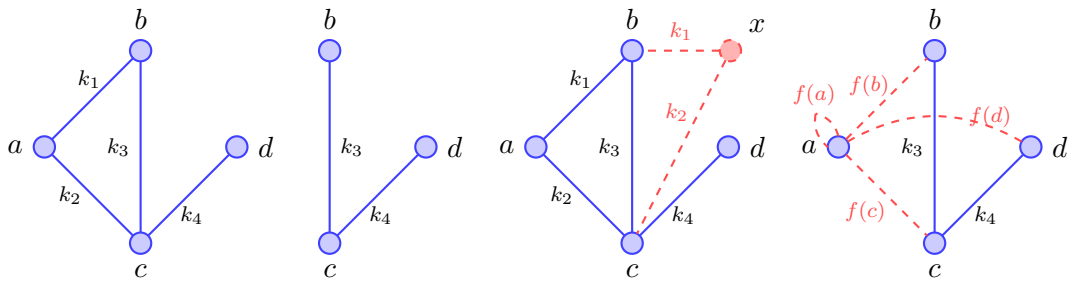


Figure 5.2: Modifications of affinity networks.

The following construction alters just one connection in the network. It is used with local affinity networks, for which the modifications set out in Def. 5.2.3 have little biological meaning, since local names do not model interaction sites.

Definition 5.2.4. Let N be an affinity network, $a, b \in N$ and $k \geq 0$. We define the affinity network $N(a, b, k)$ on the set of vertices of N by:

$$N(a,b,k)(x,y) \stackrel{\text{df}}{=} \begin{cases} k & x = a \text{ and } y = b, \text{ or } x = b \text{ and } y = a \\ N(a,b) & \text{otherwise} \end{cases} \tag{5.14}$$

Furthermore, we define the set of networks

$$N(-,-,k) \stackrel{\text{df}}{=} \{N(u,v,k) : u, v \in N\} \tag{5.15}$$

The set $\mathsf{rc}(k, A)$ contains all species arising from A through a modification of a single silent prefix rate or a single affinity in a local affinity network.

Definition 5.2.5. (rc: rate change) Let A be a species and let $k \geq 0$. We define the set of species $\mathsf{rc}(k, A)$ by induction on A as follows:

$$\mathsf{rc}(k, \mathbf{0}) \stackrel{\text{df}}{=} \varnothing \tag{5.16}$$

$$\mathsf{rc}(k, W(\vec{y})) \stackrel{\text{df}}{=} \varnothing \tag{5.17}$$

$$\begin{aligned} \mathsf{rc}(k, \Sigma_{i=0}^{n} \pi_i.A_i) \stackrel{\text{df}}{=} & \bigcup_{j=0}^{n} \{\Sigma_{i=0}^{n} \pi_i'.A_i : \pi_j = \tau@r, \pi_j' = \tau@k, \text{ and } \forall_{l \neq j}(\pi_l' = \pi_l)\} \\ & \cup \bigcup_{j=0}^{n} \{\Sigma_{i=0}^{n} \pi_i.A_i' : A_j' \in \mathsf{rc}(k, A_j) \text{ and } \forall_{l \neq j}(A_l' = A_l)\} \end{aligned} \tag{5.18}$$

$$\mathsf{rc}(k, B|C) \stackrel{\text{df}}{=} \{(B|X) : X \in \mathsf{rc}(k, C)\} \cup \{(X|C) : X \in \mathsf{rc}(k, B)\} \tag{5.19}$$

$$\mathsf{rc}(k, (\boldsymbol{\nu} N)B) \stackrel{\text{df}}{=} \{(\boldsymbol{\nu} M)B : M \in N(-,-,k)\} \cup \{(\boldsymbol{\nu} N)X : X \in \mathsf{rc}(k, B)\} \tag{5.20}$$

The set $\mathsf{cg}(\pi, Z, A)$ contains all possible ways of enriching the species A with the prefix π. If the position where π was inserted into A does not force any particular successor state, this state is set to be Z.

Definition 5.2.6. (cg: capability gain) Let A, Z be species and π be a prefix. We define the set of species $\mathsf{cg}(\pi, Z, A)$ by induction on A in the following way:

$$\mathsf{cg}(\pi, Z, \mathbf{0}) \stackrel{\text{df}}{=} \{\pi.\mathbf{0}, \pi.Z\} \tag{5.21}$$

$$\mathsf{cg}(\pi, Z, W(\vec{y})) \stackrel{\text{df}}{=} \{\pi.W(\vec{y})\} \tag{5.22}$$

$$\begin{aligned}\mathsf{cg}(\pi, Z, \Sigma_{i=0}^n \pi.A_i) \stackrel{\text{df}}{=}\ & \{\pi.\Sigma_{i=0}^n \pi_i.A_i\} \cup \{\pi_0.A_0 + \cdots + \pi_n.A_n + \pi.Z\} \\ & \cup \bigcup_{j=0}^{n} \{\Sigma_{i=0}^n \pi_i.X_i : X_j \in \mathsf{cg}(\pi, Z, A_i) \text{ and } \forall_{l\neq j} X_l = A_l\}\end{aligned} \tag{5.23}$$

$$\begin{aligned}\mathsf{cg}(\pi, Z, B|C) \stackrel{\text{df}}{=}\ & \{\pi.(B|C)\} \\ & \cup \{(B|X) : X \in \mathsf{cg}(\pi, Z, C)\} \\ & \cup \{(X|C) : X \in \mathsf{cg}(\pi, Z, B)\}\end{aligned} \tag{5.24}$$

$$\mathsf{cg}(\pi, Z, (\boldsymbol{\nu} N)B) \stackrel{\text{df}}{=} \{\pi.(\boldsymbol{\nu} N)B\} \cup \{(\boldsymbol{\nu} N)X : X \in \mathsf{cg}(\pi, Z, B)\} \tag{5.25}$$

The next definition is dual to the preceding one and describes the possible ways for the species A to lose a single capability.

Definition 5.2.7. (cl: capability loss) Let A be a species. We define the set of species $\mathsf{cl}(A)$ inductively by setting:

$$\mathsf{cl}(\mathbf{0}) \stackrel{\text{df}}{=} \varnothing \tag{5.26}$$

$$\mathsf{cl}(W(\vec{y})) \stackrel{\text{df}}{=} \varnothing \tag{5.27}$$

$$\mathsf{cl}(B|C) \stackrel{\text{df}}{=} \{(B|X) : X \in \mathsf{cl}(C)\} \cup \{(X|C) : X \in \mathsf{cl}(B)\} \tag{5.28}$$

$$\begin{aligned}\mathsf{cl}(\Sigma_{i=0}^n \pi_i.A_i) \stackrel{\text{df}}{=}\ & \{\pi_0.A_0 + \cdots \pi_{j-1}.A_{j-1} + \pi_{j+1}.A_{j+1} + \cdots + \pi_n.A_n\}_{j=0}^n \\ & \cup \bigcup_{j=0}^{n} \{\Sigma_{i=0}^n \pi_i.A'_i : A'_j \in \mathsf{cl}(A_j) \text{ and } \forall_{l\neq j}(A'_l = A_l)\}\end{aligned} \tag{5.29}$$

$$\mathsf{cl}((\boldsymbol{\nu} N)B) \stackrel{\text{df}}{=} \{(\boldsymbol{\nu} N)X : X \in \mathsf{cl}(B)\} \tag{5.30}$$

The set $\mathsf{succ}_{\mathcal{M}}(\xi)$ contains all species that may serve as successor states when adding a capability to the body of ξ (cf. Def. 5.2.6). A successor state may is an invocation of either a definition modelling a state of the same molecule (provided it does not have any virtual sites), or of a fresh definition Z.

Definition 5.2.8. Let $\mathcal{M} = (\mathcal{D}, N, P)$ be a model and let $\xi \in \mathcal{D}$. We define the set of species $\mathsf{succ}_{\mathcal{M}}(\xi)$ by:

$$\mathsf{succ}_{\mathcal{M}}(\xi) \stackrel{\text{df}}{=} \{A(\vec{x}) : (A(\vec{x}) \stackrel{\Delta}{=} B) \in [\xi]_{\mathcal{D}} \text{ and } \mathsf{vs}_{\mathcal{M}}(A) = \varnothing\} \cup \{Z() : Z\#\mathcal{D}\} \tag{5.31}$$

A pair of vectors of unique names uniquely defines a name substitution as given below. This definition is used to coordinate substitutions of physical and virtual sites in Def. 5.2.11.

Definition 5.2.9. Let $\vec{x}$ and $\vec{y}$ be vectors of names. Assume that no name appears twice in $\vec{x}$ and the same for $\vec{y}$. Define the function $\sigma_{\vec{x}\rightsquigarrow\vec{y}} : \mathcal{N}^{|\vec{x}|} \to \mathcal{N}^{|\vec{y}|}$ by:

$$(\sigma_{\vec{x}\rightsquigarrow\vec{y}}(\vec{u}))_i \stackrel{\text{df}}{=} \begin{cases} u_i & y_i \notin \vec{x} \\ u_j & y_i = x_j \end{cases} \tag{5.32}$$

In Defs. 5.2.10 and 5.2.11 we consider species definitions that may be ill-formed in a particular way. We still use the familiar $\stackrel{\Delta}{=}$ notation to render them.

Definition 5.2.10. (balanced and imbalanced definitions)
Let $\xi = (A(\vec{x}) \stackrel{\Delta}{=} B)$ be a species definition. We call ξ *imbalanced* if $\vec{x} \neq \mathsf{fn}(B)$. In this case, the *balanced version of* ξ is the definition $A(\vec{z}) \stackrel{\Delta}{=} B$, where $\vec{z} = \mathsf{fn}(\vec{B})$.

Definition 5.2.11. (replacement of definitions) Let $(\mathcal{D}, N, P)$ be a model and let ξ and ζ be species definitions such that:

(i) $\xi \in \mathcal{D}$, and

(ii) $\mathcal{D}$ contains no imbalanced definitions, and

(iii) If ζ invokes definitions from $\mathcal{D}$, it respects their arities, and

(iv) Handles and arities of ξ and ζ are equal.

We define the model $(\mathcal{D}, N, P)[\xi \mapsto \zeta]$ by

$$(\mathcal{D}, N, P)[\xi \mapsto \zeta] \stackrel{\text{df}}{=} \texttt{balance}((\mathcal{D} \setminus \{\xi\}) \cup \{\zeta\}, N, P) \tag{5.33}$$

where `balance` is the following algorithm:

```
1   input (E, M, Q);
2
3   while E contains imbalanced definitions
4     let (A(x) ≜ B) = η ∈ E an imbalanced definition in E;
5     let (A(y) ≜ B) be the balanced version of η;
6     replace every A(z) with A(σ_{x⇝y}(z)) in bodies of E;
7     replace every A(z) with A(σ_{x⇝y}(z)) in Q;
8   endwhile
9   M := M ⊖ (M \ fn(Q));
10
11  output (E, M, Q);
```

The above construction enables us to replace one species definition (ξ) in a model with another (ζ). Since their handles and arities are equal, this amounts to

substituting the bodies. It is seemingly a straightforward task, but surprisingly many things can go wrong here: in particular, the number and order of name parameters may differ between ξ and ζ. To counter this, we turn every invocation of ξ of the form $A(\vec{z})$ into $A(\sigma_{\vec{x}\rightsquigarrow\vec{y}}(\vec{z}))$ (lines 6 and 7), where $\sigma_{\vec{x}\rightsquigarrow\vec{y}}$ encodes the reordering and forgetting/addition of parameters derived from comparing $\vec{x}$ with $\vec{y}$. Unfortunately, in doing so we may change the free name sets of the bodies of some definitions and hence unbalance them, so it is necessary to redo the replacement, this time of the new unbalanced definition with its balanced version, and so on until all definitions are balanced.

Strictly speaking, it is in order now to prove the termination of the algorithm and its insensitivity to the order in which imbalanced definitions are picked. We leave it to the reader, and turn instead to finally defining variation operators.

5.3 Gene-level operators

We begin with operators that work on entire proteins rather than particular sites or states. All evolutionary effects modelled by the operators in this class correspond closely to actual mutations.

Gene emergence By gene emergence we mean either emergence of a new stretch of DNA and the corresponding protein or the inclusion of a preexisting gene and its product to the modelled network. The variation operator modelling gene emergence has the form:

$$\frac{}{(\mathcal{D}, N, P) \longrightarrow (\mathcal{D} \cup \{Z() \triangleq \mathbf{0}\}, N, P)}\ \text{GENE-NEW} \tag{5.34}$$

Here, we assume that the new protein cannot interact with the existing agents within the network. Thus, it serves as a "blank page", on which evolution can develop new functionalities and ultimately make it interact with the network in a meaningful way. We also assume that it is not expressed initially; the operator GENE-EXPR(1) (below) can be used to change the expression level.

Gene loss Existing genes can be lost in many ways, in particular by physical removal (deletion) of their DNA or by serious corruption of its regulatory or

coding sequence. As a result, the protein encoded by the gene disappears from the network. The operator rule modelling gene loss has the form:

$$\frac{\xi \in \mathcal{D}}{(\mathcal{D}, N, P) \longrightarrow (\mathcal{D} \setminus [\xi], N \ominus (\mathsf{fn}(P) \setminus \mathsf{fn}(Q)), Q)}\text{GENE-LOSS} \tag{5.35}$$

where

$$Q \stackrel{\text{df}}{=} \lambda A.\texttt{if}\ \ A \circ\!\!\rightarrow [\xi]\ \texttt{then}\ 0\ \texttt{else}\ P(A) \tag{5.36}$$

Hence, in $c\pi$ terms gene loss is modelled by the removal of all species definitions associated with the protein the gene coded for, as well as those components of the process that referred to them. The free names that model the interaction sites of the lost protein are also removed from the affinity network.

Change in gene expression A mutation can change the expression level of a gene by altering its promoter sequence or its regulatory pathway(s). In recent years this kind of evolutionary change has received a lot of attention as a potentially crucial component of evolution of development [31, 52, 54]. There are two operators modelling this kind of evolutionary events:

$$\frac{(A(\vec{x}) \stackrel{\Delta}{=} B) \in \mathcal{D} \qquad \mathsf{vs}(A) = \varnothing \qquad A(\vec{x}) \notin \mathsf{supp}(P) \qquad c \geq 0}{(\mathcal{D}, N, P) \longrightarrow (\mathcal{D},\ N,\ \lambda S.\texttt{if}\ S{=}A(\vec{x})\ \texttt{then}\ c\ \texttt{else}\ P(S))}\text{GENE-EXPR(1)} \tag{5.37}$$

$$\frac{A(\vec{x}) \in \mathsf{supp}(P) \qquad c \geq 0}{(\mathcal{D}, N, P) \longrightarrow (\mathcal{D},\ N,\ \lambda S.\texttt{if}\ S{=}A\ \texttt{then}\ c\ \texttt{else}\ P(S))}\text{GENE-EXPR(2)} \tag{5.38}$$

In $c\pi$ terms, therefore, it is only the process that changes, with the definitions and network remaining the same. Observe that any non-negative real number c is allowed as the new concentration, and thus any model has uncountably many variants via this rule. The difference between the two operators is minor: the latter deals with species already present in the process, the former with ones about to introduced.

Gene duplication A fragment of DNA containing a gene can be accidentally duplicated, for example in the process of recombination. From the evolutionary point of view, this usually results in weakened selective pressure on the gene: as long as one copy is functional, the other can be safely mutated. Furthermore,

the mutations of the new copy are likely to be of relevance to the network(s) the original gene was part of, since they result in variants of the same protein. Gene duplication is believed to be the single most important genetic mechanism of evolutionary innovation; see [154] for more information. The variation operator modelling gene duplication has the form:

$$\frac{[\xi] = \{A_i(\vec{x_i}) \stackrel{\Delta}{=} B_i\}_{i=0}^{n} \subseteq \mathcal{D}}{(\mathcal{D}, N, P) \longrightarrow (\mathcal{D} \cup \{A'_i(\vec{z_i}) \stackrel{\Delta}{=} B_i[\vec{A} \mapsto \vec{A'}]\{\vec{y}/\vec{p}\}\}_{i=0}^{n}, N \oplus_{\vec{p}} \vec{y}, P)}\ \text{GENE-DUP} \quad (5.39)$$

where

$\vec{A} \stackrel{\text{df}}{=} (A_0, \ldots, A_n)$ and $\vec{p} \stackrel{\text{df}}{=} \mathsf{ps}(A_0) \cup \ldots \cup \mathsf{ps}(A_n)$,
$\vec{A'} = (A'_0, \ldots, A'_n)$ is a vector of fresh handles and $\vec{y}$ is a vector of fresh names,
$\vec{z_i} \stackrel{\text{df}}{=} \vec{x_i}\{\vec{y}/\vec{p}\}$, i.e. $(z_i)_j \stackrel{\text{df}}{=} (\texttt{if}\ \ (x_i)_j = p_k\ \ \texttt{then}\ \ y_k\ \ \texttt{else}\ \ (x_i)_j)$.

Thus, in the target model the set of definition is extended with a copy of the $\circ\!\!\rightarrow$-induced equivalence class of ξ. The physical sites of the definitions from $[\xi]$ are duplicated in the affinity network, and the duplicated definitions use the duplicated sites rather than the original ones. In this way their subsequent variation is independent. Again, we assume that the copied gene is not expressed; this can be amended by applying the GENE-EXPR(1) operator immediately after the duplication.

5.4 State variation

Variation operators that affect the internal structure of $c\pi$ terms correspond to mutational effects rather than events. They are responsible for most of the expressive power of the collection of operators developed here. We begin with the operator modelling the evolution of an autonomous state transition capability:

$$\frac{k \geq 0 \quad \xi = (A(\vec{x}) \stackrel{\Delta}{=} B) \in \mathcal{D} \quad C \in \mathsf{succ}(\xi) \quad D \in \mathsf{cg}(\tau@k, C, B)}{(\mathcal{D}, N, P) \longrightarrow (\mathcal{D} \cup X, N, P)[\xi \mapsto (A(\vec{x}) \stackrel{\Delta}{=} D)]}\ \text{STATE-TAU} \quad (5.40)$$

where

$$X \stackrel{\text{df}}{=} \begin{cases} \{Z() \stackrel{\Delta}{=} \mathbf{0}\} & \text{if } C = Z() \text{ and } Z\#\mathcal{D} \\ \varnothing & \text{otherwise} \end{cases} \quad (5.41)$$

Here, C ranges over possible successor states of the capability $\tau@k$. These are invocations of definitions from the same $\circ\!\!\rightarrow$ class as ξ, or an invocation of an entirely fresh definition (cf. Def 5.2.8), which is then added to the model. An enriched version of the body of ξ is sourced from the set $\mathsf{cg}(\tau@k, C, B)$, and it replaces B in the model.

Precisely the same procedure is followed in the case of an emergence of an interaction site rather than autonomous capability (X has the same meaning here):

$$\frac{f : N \to \mathbb{R}_{\geq 0} \quad \xi = (A(\vec{x}) \triangleq B) \in \mathcal{D} \quad C \in \mathsf{succ}(\xi) \quad D \in \mathsf{cg}(a, C, B)}{(\mathcal{D}, N, P) \longrightarrow (\mathcal{D} \cup X, N \oplus_f a, P)[\xi \mapsto (A(\vec{x}) \triangleq D)]}\text{STATE-SITE} \tag{5.42}$$

The only difference here is that the new state carries a new free name, which is added to the affinity network. This is a direct consequence of the assumption that different sites are represented by different names. Also, note that the new definition $A(\vec{x}) \triangleq D$ is imbalanced.

Arguably the most complex operator models emergence of a binding site. Since binding is modelled as scope extrusion, the operator has to introduce a name-passing prefix. For the sake of simplicity and symmetry we only consider prefixes sending one private name, bound in a local two-point affinity network, and receiving only one name, presumably bound in the same way by the interaction partner. Furthermore, we assume that upon binding the molecule enters a state with two possible, mutually exclusive continuations; the actual path chosen depends on whether the internal synchronisation within the complex takes place using the name that was sent or the one that was received. In the next chapter (§6.2) we show that this setup is sufficient to evolve complex enzyme-mediated reactions.

$$\frac{\begin{array}{c} k \geq 0 \qquad f : N \to \mathbb{R}_{\geq 0} \qquad \xi = (A(\vec{x}) \triangleq B) \in \mathcal{D} \\ C \in \{\overline{x}.X + y.Y : X, Y \in \mathsf{succ}_{\mathcal{D}}(\xi)\} \quad D \in \mathsf{cg}(a(x;y), C, B) \end{array}}{(\mathcal{D}, N, P) \longrightarrow (\mathcal{D} \cup Z, N \oplus_f a, P)[\xi \mapsto (A(\vec{x}) \triangleq (\boldsymbol{\nu} x \overset{k}{\text{—}} \overline{x})D)]}\text{STATE-BIND} \tag{5.43}$$

where

$$Z \stackrel{\text{df}}{=} \{W() \triangleq \mathbf{0} : W() \subset C \text{ and } W \# \mathcal{D}\} \tag{5.44}$$

Apart from the increasing complexity of the three operators defined so far in this section, the reader is encouraged to detect similarities between them. All three focus on a single species definition ξ and enrich it with an extra capability: a silent prefix $\tau@k$, a simple prefix a, and finally a name-passing prefix $a(x;y)$. The resulting extended definition takes place of the original one in the model, and the affinity network is updated in the two latter cases to include the newly added name.

The last state-related operator models loss of an interaction or autonomous capability. The successor state disappears together with the capability (cf. Def. 5.2.7), and the operator has the self-explanatory form:

$$\frac{\xi = (A(\vec{x}) \triangleq B) \in \mathcal{D} \qquad \zeta \in \mathsf{cl}(B)}{(\mathcal{D}, N, P) \longrightarrow (\mathcal{D}, N, P)[\xi \mapsto A(\vec{x}) \triangleq C]}\text{STATE-LOSS} \tag{5.45}$$

5.5 Rate changes

Finally, we have two operators corresponding to changes in site binding or interaction affinities, and changes to the rates of autonomous capabilities. We begin with the first:

$$\frac{a \in N \qquad f\colon\ N \to \mathbb{R}_{\geq 0}}{(\mathcal{D}, N, P) \longrightarrow (\mathcal{D}, N \odot_f a, P)}\text{RATE-SITE} \tag{5.46}$$

This operator models a molecular event: change of the 3D shape of an active site of a protein. Such an event impacts the affinities of this site towards all other sites, because affinity is to a large extent a measure of 3D complementarity of sites. For that reason the operator reconfigures all connection of the site a in the global affinity network (cf. Def. 5.2.3).

In contrast, the next operator models a molecular effect—change of a single reaction/transformation rate of a molecule. Here, only a single silent prefix rate or a single affinity in a local affinity network is modified. There is no need to reconfigure all affinities in a local affinity network, because local names do not model actual sites (§5.1.1). The operator has the following form:

$$\frac{\xi = (A(\vec{x}) \triangleq B) \in \mathcal{D} \qquad k \geq 0 \qquad C \in \mathsf{rc}(k, B)}{(\mathcal{D}, N, P) \longrightarrow (\mathcal{D}, N, P)[\xi \mapsto (A(\vec{x}) \triangleq C)]}\text{RATE-AUTO} \tag{5.47}$$

Summary

We have defined 11 variation operators as inference rules inducing a binary relation on the space of $c\pi$ models. This structure completes our recasting of the framework of neutral spaces (§2.4.3) in process-algebraic terms. In the next chapter we study its expressiveness and show how to apply it to the study of important evolutionary properties of real biological systems.

Chapter 6

Evolutionary case studies

6.1 Introduction

This chapter is devoted to a demonstration of expressiveness and an application in evolutionary modelling of the variation operators defined in the previous chapter.

Expressiveness is showcased by building, from a trivial model, a $c\pi$ description of a simple enzymatic reaction exhibiting classical Michaelis-Menten dynamics; this model is very similar to the one that served as a running example in Ch. 3. The construction proceeds solely by application of variation operators, and every step has a plausible biological interpretation. The final description has the potential for further evolution, and we demonstrate this by extending the model with a competitive inhibition mechanism. As a final demonstration of the expressive power of the operators we briefly show how to build a fully functional model of a MAPK cascade from scratch. We stress that neither model populations nor model selection techniques are involved here, for the purpose of these exercises is to demonstrate that variation operators create a rich, evolutionarily meaningful accessibility structure on the space of $c\pi$ models, rather than to study evolutionary origins of enzymes and signalling cascades.

In the second part of the chapter we perform a computational exploration of the evolutionary neighbourhood of the previously developed MAPK model by analysing approx. 1 million of its variants obtained via applications of the RATE-SITE operator. We rely on the previously championed syntax/semantic separation (see e.g. Ch. 1 and §4.4.3) in process algebras to perform both qualitative (LTL checking) and quantitative (signal integration) analyses of these variants. The

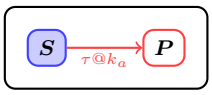
results are insights into neutral spaces and fitness landscape underlying MAPK signalling, and in particular evolutionary robustness of individual active sites of the cascade proteins.

6.1.1 Overview of the chapter

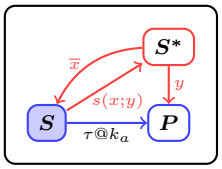
Section 6.2 contains the enzyme study. Section 6.3 is devoted to the analysis of the MAPK cascade, with §§6.3.1–6.3.3 dealing with background and previous work, the model itself, and the computational experiments, respectively.

6.2 Evolution of enzyme models

Beginning with the very simple $c\pi$ model $(\{S() \triangleq \mathbf{0}\}, \varnothing, c_S \cdot S)$ representing a network consisting of a single inert species S, it takes several steps (i.e. operator applications) to arrive at a model of a fully functional enzyme, and a further few to obtain a model of competitive enzyme inhibition. There are many trajectories meeting this specification; we show one of them, commenting on the use of variation operators as well as on the underlying evolutionary events. The complete trajectory of formal models is given in Fig. 6.1. In addition we provide a cartoon to illustrate each intermediate model. The changes introduced to the model in the given step are marked in red; filled nodes denote species present in the process part of the $c\pi$ model; affinity networks are not depicted, and so in the last step we only highlight the names affected by the RATE-SITE operator. The asterisk $-^*$ denotes a molecule bound in a complex.

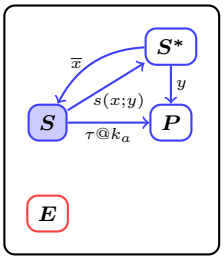


Step 1 The STATE-TAU operator is applied to equip the species S with a silent prefix. The successor state of this prefix is a new state P; a definition $P() \triangleq \mathbf{0}$ is added to the set of species definitions. This step corresponds to the molecular species S evolving a capability to autonomously turn into P. Naturally, S is the future substrate and P the future product of the modelled enzymatic reaction.

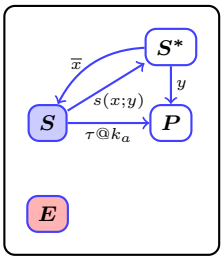


Step 2 The STATE-BIND operator is applied to the definition of S, resulting in the previously evolved autonomous capability $\tau@k_a$ entering in direct competition with the new binding capability $s(x;y)$. The successor state of the new capability,

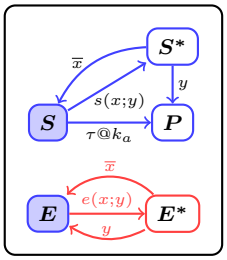
as mandated by the STATE-BIND operator, is a choice between two states of the S species; these states are arbitrarily chosen here to be S and P. The two virtual sites introduced are x and $\overline{x}$, and it is assumed that they can interact at the rate k_{-1}; this is the future enzyme-substrate complex backwards dissociation rate. The new name s is added to the affinity network, and the definition of S now carries a formal parameter s. In biological terms, this step corresponds to the emergence of an interaction site s on the surface of the protein S; upon binding to this site a complex is formed, and after it dissociates S is either unchanged or transformed into P. As there are no sites in the system able to interact with s, this addition has no effect on the dynamics of the model.



Step 3 An application of the GENE-NEW operator results in a new definition $E() \triangleq \mathbf{0}$ being added to the definition set. Observe that it is completely independent from both S and P, and thus is forms a singleton-sized equivalence class w.r.t. the formal dependence of definitions (Def. 5.2.1). As discussed in the previous chapter, this addition can be seen either as an emergence of a new non-functional gene, or an inclusion of an existing gene in the modelled system. In our case, this gene codes for the future enzyme catalysing the transformation of S into P.



Step 4 Through an application of the GENE-EXPR(1) operator, the newly created species E appears in the process at the concentration c_E. Only the process changes in this step, species definitions and the affinity network remain the same. The dynamical behaviour of the model does not change, because E is defined to be the inert process $\mathbf{0}$. The biological interpretation of this alteration is a change in the regulation of the gene coding for E, resulting in an increase of its basal concentration. It is also possible—but not essential, by far—to consider this step together with the preceding one, as one mutation event giving rise to a new, immediately expressed gene.

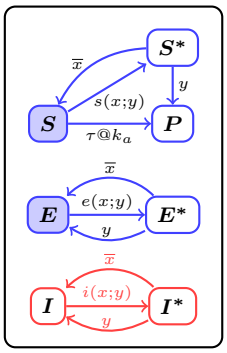


Step 5 The STATE-BIND operator is applied to the definition E. A single binding site e complementary to s appears in E, and their affinity k_r is the enzyme-substrate binding rate constant. The successor state of this site is a two-way choice leading back to E on both paths. The site e is added to the definition of E as a formal parameter and also appears in the

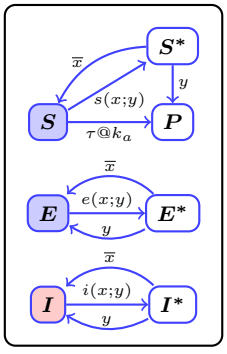
global affinity network. The affinity k_+ of the two virtual sites introduced in E is the enzyme-complex forward dissociation rate constant. In biological terms, this step should be interpreted analogously to Step 2: an emergence of a functional binding site on the surface of a protein.

We contrast this simple enzyme model with the running example of Ch. 3. Both exhibit exactly the same behaviour, save for the fact the product P could spontaneously degrade in the Ch. 3 version (the corresponding prefix can be easily added to the current model using the STATE-TAU operator). A more subtle difference is the arrangement of private names (virtual sites) in the model: previously, they were grouped in a single three-point local affinity network, now they are split between two two-point networks. This is a manifestation of the assumption made in §5.1.1 that only very simple local affinity networks can be introduced by variation operators.

The next three operator applications lead to a model of competitive enzyme inhibition. Following [82, p. 149], competitive inhibition takes place when a molecule called an *inhibitor* competes with the substrate for the binding site of the enzyme. When the inhibitor molecule binds the enzyme, it makes it inactive until the complex dissociates.



Step 6 A new species definition appears in the model through an application of the GENE-DUP operator to the definition $E(e)$. The definition-dependency class of $E(e)$ contains only $E(e)$ itself (cf. Def. 5.2.1), and so only this definition is duplicated. The duplicate is $I(i)$, where I is a fresh handle, and i is a fresh physical site with the same affinities as e; I plays the rôle of the inhibitor in the final model. This step models the duplication of the gene coding for the enzyme. Obviously it is not essential to fashion the inhibitor out of the copy of the enzyme: it is also possible to duplicate and rearrange the substrate or use the GENE-NEW operator instead.



Step 7 The copy of the enzyme appears in the process via an application of the GENE-EXPR(1) operator; its concentration is set to c_I. The biological interpretation of this step is analogous to that of Step 4. Interestingly, at this stage the conversion of substrate into product proceeds faster than previously because I is an exact copy of the enzyme, and thus can catalyse the reaction as well.

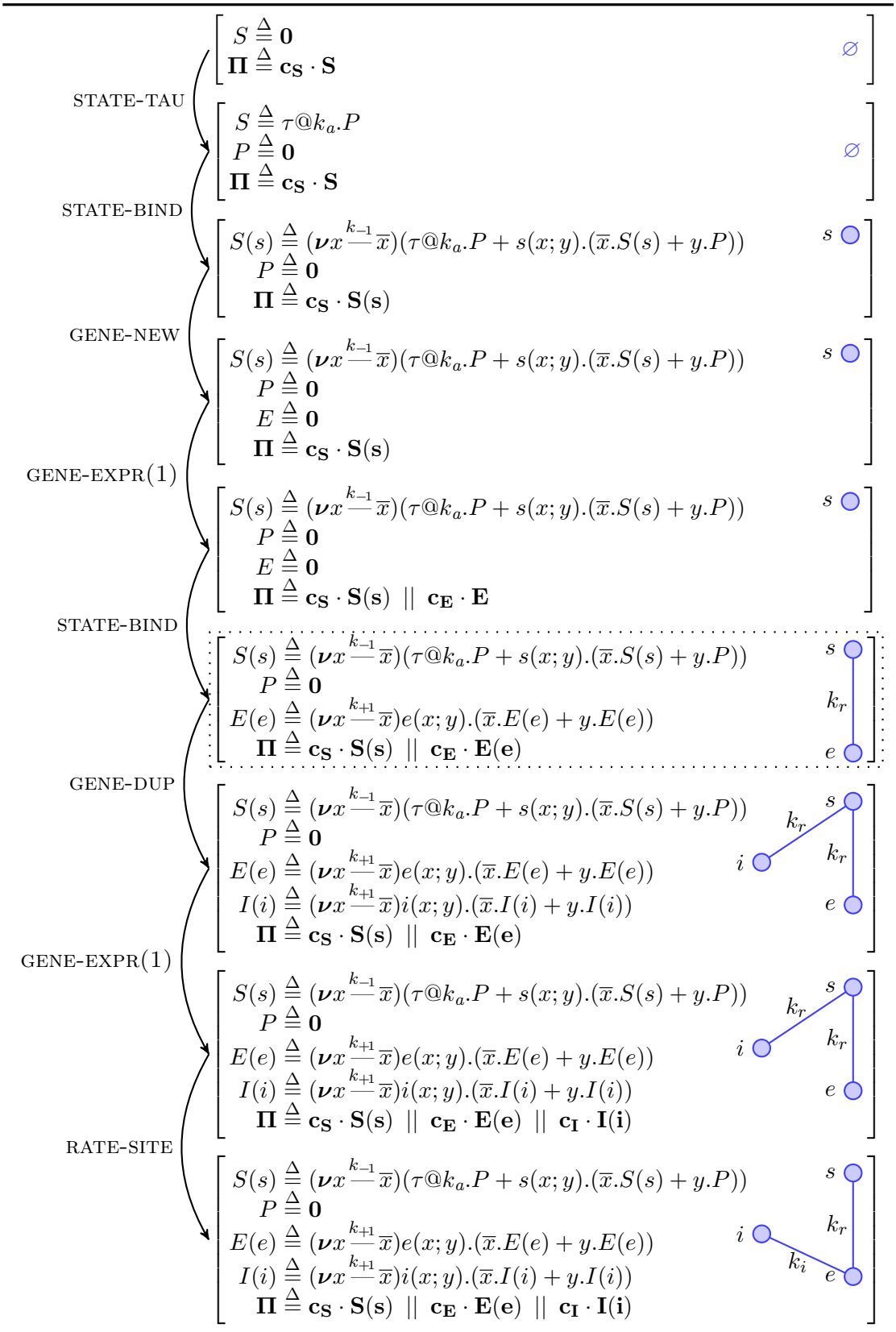
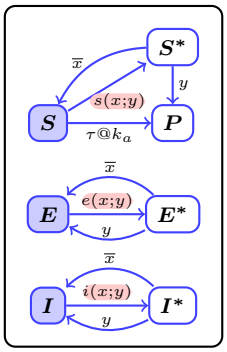


Figure 6.1: Evolution of simple enzyme models by means of variation operators. The model of a standard Michaelis-Menten enzyme is highlighted.



Step 8 The final step is an application of the RATE-SITE operator to the site i. The site affinities are reconfigured in the affinity network in such a way that i now has an affinity for the active site of E, rather than that of S. This switch results in I no longer taking part in the conversion of S into P; instead, it attempts to bind E and prevent it from interacting with S, thus fulfilling the definition of a competitive inhibitor. The underlying genetic cause of this transformation can be, for example, a point mutation in the fragment of inhibitor gene coding for the active site of the inhibition protein.

Summary We have presented a trajectory of $c\pi$ models of increasing complexity and functionality obtained strictly by applications of variation operators. Starting with a trivially simple model we have constructed a representation of an enzymatic reaction, and then further extended it with a competitive inhibition mechanism. Crucially, every step of this trajectory can be attributed to a genetic event (but see §5.1.1). We conclude that variation operators rigorously define a non-trivial, evolutionary meaningful accessibility structure on the space of $c\pi$ models.

6.3 Evolutionary properties of a signalling cascade

In this section we study evolutionary robustness and evolvability of a well-known biochemical system—the MAPK cascade—by means of massively parallel computational analysis driven by our variation framework. After giving a $c\pi$ model of the MAPK cascade, we systematically reconfigure it with the RATE-SITE operator (§5.5), yielding approximately 1 million variants, each immediately accessible from the base model. We then analyse the dynamical behaviour of these variants and draw conclusions regarding the distribution of fitness amongst them (using ODE extraction and solving) and perform a qualitative assessment of their signal processing characteristics (using formal verification of numerical traces with Linear Temporal Logic). Due to the solid genetic and mechanistic underpinning of the variation operators (and RATE-SITE in particular), the generated variants are likely to be representative of the actual evolutionary neighbourhood of the MAPK cascade. The same cannot be said with the same degree of confidence about more ad-hoc variation schemes such as single parameter sweeps.

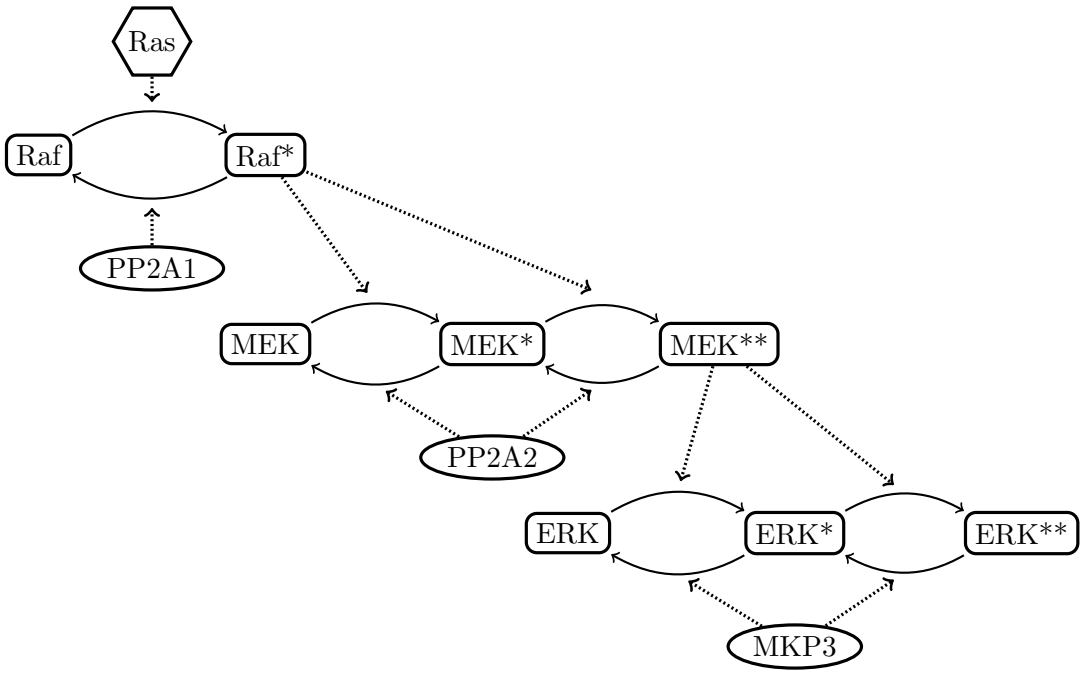


Figure 6.2: The human MAPK cascade modelled in this chapter. Ras is the input signal and ERK** is the output. Raf, MEK and ERK proteins form the MAPKKK, MAPKK and MAPK layers, respectively. Every box is a protein species. Protein kinases are depicted as rectangular boxes, and phosphotases as ovals. Solid arrows denote potential transformations and dotted ones represent catalytic activity. Diagram adapted from [113].

6.3.1 Background

Signal transduction is the family of molecular mechanisms by which cells recognise and process external stimuli, eventually converting them into a specific biochemical response, for example modification of expression of a specific gene or change in activity of a specific enzyme in the cytoplasm [82, Ch. 6]. In a typical signal transduction scenario, a small signalling molecule called a *ligand* binds the outside part of a large transmembrane *receptor* protein and causes it to change shape. The new conformation of the in-cell part of the receptor activates another molecule, which in turn reacts with yet another, and so on; eventually after such a *cascade* of interactions—which may involve two to a dozen different protein species as well as complex control mechanisms such as feedbacks and feed-forwards—the last protein in the cascade migrates to the nucleus, binds the DNA and changes the expression of one or more genes. More generally, signal transduction refers to all information processing in the cell, and thus its impor-

tance to biology and medicine cannot be underestimated. It is also the prime application area of process algebras in systems biology, as the copy numbers of signalling molecules tend to be relatively low, making stochastic simulations (and in some cases even model checking) computationally tractable.

MAPK cascades The mitogen-activated protein kinase (MAPK) cascades are an important component of many signal transduction pathways. Found in all eukaryotes, MAPK cascades help control a number of cellular processes, most notably cell growth and cell division. Here, we restrict our analysis to a subfamily of MAPK architectures considered in [70] (Fig. 6.2). The initial signal promotes the activation (phosphorylation) of an order 3 protein kinase (MAPKKK). Once activated, MAPKKK acts as a catalyst for the phosphorylation of an order 2 kinase (MAPKK). Doubly phosphorylated MAPKK activates (again, twice) an order 1 kinase (MAPK), which is considered the output signal of the cascade. Finally, every kinase has a corresponding phosphatase, which performs the opposite action, namely dephosphorylates its target. This multi-tiered architecture promotes sensitivity to the signal and reduces response time [70]; as a result, the pathway operates like a fast, sensitive, amplifying switch. The MAPK cascade is among the most often modelled and best-understood signalling systems [110] and often serves—as it does here—as a benchmark for new systems biology techniques.

Existing work An algebra-based programming language BlenX (see §2.3.3) has been used to evolve MAPK-like architectures *in silico* [34, 35, 130]. The initial population of BlenX models of a disconnected set of cascade proteins undergoes thousands of rounds of simulated evolution. In every round, an individual program can change along specific mutation schemes very similar to our own variation operators. After mutations, the fitness of every individual is assessed using a simple formula promoting speed and strength of cascade's response to changes in the input signal. Individuals with high fitness have a greater chance of progressing to the next round. The original cascade architecture was not reconstructed in this experiment, but several different architectures with comparably high fitness were found.

From the point of view of this dissertation, the BlenX experiment is notable not because of its results, but because the approach taken is related to our own. The authors have designed and implemented a number of syntactical modifications of

BlenX programs with the explicit purpose of modelling of genetic events; these include duplication and loss of proteins and domains as well mutations of individual domains. The BlenX variation scheme is specifically designed to model evolution of signalling pathways, and explicitly constrains syntactic variation in order to avoid the events-effects mismatch (§5.1.1). Hence, it is less general and almost certainly less expressive than the variation operators we use in this thesis, but it offers stronger guarantees of biological correctness and relevance in its chosen application domain.

6.3.2 A $c\pi$ model of the MAPK cascade

Trusting that the reader is by now familiar with the general structure of $c\pi$ models as well as with the basic shape of the MAPK cascade, we do not describe the $c\pi$ model of the MAPK cascade (Fig. 6.3) in detail, but focus on the aspects that are of importance for the variation experiments. First, however, we point out that this model can be built from the empty one $(\varnothing, \varnothing, 1 \cdot \mathbf{0})$ by applications of variation operators. One possible sequence consists of 7 applications of the GENE-NEW operator establishing ancestral forms of the kinases, phosphotases and Ras, followed by a single use of STATE-TAU introducing the degradation of Ras, followed by 16 applications of STATE-BIND introducing the 16 protein active sites as well as the correct complex formation/dissociation patterns, and terminated by 7 applications of GENE-EXPR(1) establishing the initial state of the system. Once more we see that variation operators are powerful enough to generate models of real biological significance.

Handles and sites The handles of species definitions in the MAPK model consist of a species name followed by zero, one or two asterisks denoting the phosphorylation level; thus MEK^{**} stands for the doubly-phosphorylated MEK protein. A similar convention is followed for the global names (physical sites): if a particular protein in a particular state exposes only one active site, the name of that site is the lowercase version of the corresponding definition handle. When two sites are exposed, one is invariably the kinase active site and the other the phosphotase-binding site; the former is again the lowercase of the definition handle, and the latter is the lowercase equipped with the subscript $-_b$. Thus, erk^*_b is the phosphotase-binding site on the surface of the singly-phosphorylated ERK.

Rates and dynamics All site affinities are set to the unit value 1.0; this applies to local affinity networks as well. The k_d degradation rate of *Ras* is also set to 1.0. The initial concentrations of species are: 2.0 for Ras, PP2A1, PP2A2, and MKP3; 10.0 for Raf, MEK and ERK; and trace amount 0.01 for all other species (not shown in Fig. 6.3). The model gives rise to a set of 23 ODEs, which we solve numerically over 72 time units, with 10 equally spaced integration points per unit. The result is, therefore, a 720-entries-long time series. The initial signal—defined as the combined concentrations of free Ras and the Ras-Raf complex—decays rapidly and is followed by a well-defined peak of the output (free ERK**); see Fig. 6.4 for the graph. The signal is not amplified and is not transduced particularly fast in this model, but we can conclude that the cellular behaviour is reproduced satisfactorily for our purposes.

6.3.3 Computational experiments

In this section we describe our study of robustness and evolvability of the MAPK cascade in the context of a particular kind of genetic perturbations—namely, mutations affecting individual active sites of cascade proteins. Recall from the previous chapter that the variation operator modelling this kind of genetic change is RATE-SITE (§5.5). Starting with the $c\pi$ model of the MAPK cascade described above (Fig. 6.3), we systematically reconfigure its global affinity network using the RATE-SITE operator and analyse the resulting variants. We limit ourselves to all possible reconfigurations that can be obtained by a single application of RATE-SITE to the initial model and where the affinities remain either 0.0 or 1.0; hence, we obtain $2^{20} = 1048576$ variants (16 individual sites to reconfigure $\times$ 2^{16} possible reconfigurations of each).

For every variant we detect its *phenotype class* and compute its *fitness correlate* (henceforth *fitness* for short). These are qualitative and quantitative analyses, respectively, and we perform them using different computational techniques.

Phenotype classes We classify the solutions of ODE systems (and thus the underlying models) in four exclusive groups according to the qualitative characteristic of the numerical trace corresponding to the output signal of the cascade variant. The classes correspond to oscillatory, peak, switch and noise responses. We are inspired here by an earlier study [136], where similar classification—albeit using different methods—was performed exhaustively on the space of small

$$
\begin{aligned}
Ras &\triangleq (\boldsymbol{\nu} x\text{—}\overline{x})\, ras(x;y).(\overline{x}.Ras + y.Ras) + \tau@k_d.\mathbf{0} \\
Raf &\triangleq (\boldsymbol{\nu} x\text{—}\overline{x})\, raf(x;y).(\overline{x}.Raf + y.Raf^{*}) \\
Raf^{*} &\triangleq (\boldsymbol{\nu} x\text{—}\overline{x})(\boldsymbol{\nu} z\text{—}\overline{z})(raf^{*}(x;y).(\overline{x}.Raf^{*} + y.Raf^{*}) + raf_b^{*}(z;y).(\overline{z}.Raf^{*} + y.Raf)) \\
PP2A1 &\triangleq (\boldsymbol{\nu} x\text{—}\overline{x})\, pp2a1(x;y).(\overline{x}.PP2A1 + y.PP2A1) \\
MEK &\triangleq (\boldsymbol{\nu} x\text{—}\overline{x})\, mek(x;y).(\overline{x}.MEK + y.MEK^{*}) \\
MEK^{*} &\triangleq (\boldsymbol{\nu} x\text{—}\overline{x})(\boldsymbol{\nu} z\text{—}\overline{z})(mek^{*}(x;y).(\overline{x}.MEK^{*} + y.MEK^{**}) + mek_b^{*}(z;y).(\overline{z}.MEK^{*} + y.MEK)) \\
MEK^{**} &\triangleq (\boldsymbol{\nu} x\text{—}\overline{x})(\boldsymbol{\nu} z\text{—}\overline{z})(mek^{**}(x;y).(\overline{x}.MEK^{**} + y.MEK^{**}) + mek_b^{**}(z;y).(\overline{z}.MEK^{**} + y.MEK^{*})) \\
PP2A2 &\triangleq (\boldsymbol{\nu} x\text{—}\overline{x})\, pp2a2(x;y).(\overline{x}.PP2A2 + y.PP2A2) \\
ERK &\triangleq (\boldsymbol{\nu} x\text{—}\overline{x})\, erk(x;y).(\overline{x}.ERK + y.ERK^{*}) \\
ERK^{*} &\triangleq (\boldsymbol{\nu} x\text{—}\overline{x})(\boldsymbol{\nu} z\text{—}\overline{z})(erk^{*}(x;y).(\overline{x}.ERK^{*} + y.ERK^{**}) + erk_b^{*}(z;y).(\overline{z}.ERK^{*} + y.ERK)) \\
ERK^{**} &\triangleq (\boldsymbol{\nu} x\text{—}\overline{x})\, erk^{**}(x;y).(\overline{x}.ERK^{**} + y.ERK^{*}) \\
MKP3 &\triangleq (\boldsymbol{\nu} x\text{—}\overline{x})\, mkp3(x;y).(\overline{x}.MKP3 + y.MKP3) \\
\boldsymbol{\Pi} &\triangleq \mathbf{10.0} \cdot \boldsymbol{Raf} \,||\, \mathbf{2.0} \cdot \boldsymbol{Ras} \,||\, \mathbf{10.0} \cdot \boldsymbol{MEK} \,||\, \mathbf{10.0} \cdot \boldsymbol{ERK} \,||\, \mathbf{2.0} \cdot \boldsymbol{PP2A1} \,||\, \mathbf{2.0} \cdot \boldsymbol{PP2A2} \,||\, \mathbf{2.0} \cdot \boldsymbol{MKP3}
\end{aligned}
$$

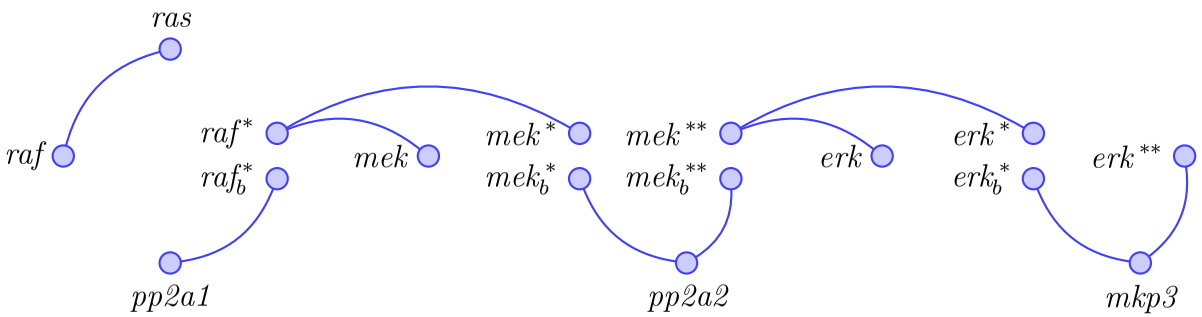


Figure 6.3: A $c\pi$ model of the MAPK cascade. Site affinities, arguments of definitions and initial (trace) concentrations of phosphorylated proteins omitted for readability.

abstract signalling networks. We characterise each class by a Linear Temporal Logic formula, and thus the classification amounts to checking whether the time series of ERK** satisfies it. The formulae are:

$$\mathsf{oscil}(e) \stackrel{\text{df}}{=} \mathsf{F}(e < 0.5 \wedge \mathsf{F}(e > 1.5 \wedge \mathsf{F}(e < 0.5 \wedge \mathsf{F}(e > 1.5 \wedge \mathsf{F}(e < 0.5))))) \quad (6.1)$$

$$\mathsf{peak}(e) \stackrel{\text{df}}{=} \mathsf{not\ oscil}(e) \wedge e < 0.5 \wedge \mathsf{F}(e > 1.5 \wedge \mathsf{F}(\mathsf{G}(e < 0.5))) \quad (6.2)$$

$$\mathsf{switch}(e) \stackrel{\text{df}}{=} e < 0.5 \wedge \mathsf{F}(\mathsf{G}(e > 1.5)) \quad (6.3)$$

$$\mathsf{noise}(e) \stackrel{\text{df}}{=} \mathsf{not\ oscil}(e) \wedge \mathsf{not\ peak}(e) \wedge \mathsf{not\ switch}(e) \quad (6.4)$$

The low and high threshold values 0.5 and 1.5 are calibrated to put the starting model comfortably into the peak category. A more sophisticated approach using the QFLTL logic [44] to infer the threshold values for which a given trace satisfies a formula, is also possible.

Fitness We use a simplified version of the fitness measure used in the BlenX studies (§6.3.1). Given a time series $e = (e_1, \ldots, e_{720})$ of the concentration of ERK** in the analysed variant of the cascade, the fitness of this variant is defined by

$$\mathsf{fitness}(e) \stackrel{\text{df}}{=} \sum_{i=1}^{135} e_i - \sum_{i=302}^{720} e_i \quad (6.5)$$

The contribution of the first sum rewards a quick and strong response to the input signal. Similarly, the second sum penalises slow or incomplete switching off of the output signal (Fig. 6.4). The cut-off values 135 and 302 are the time points where the rapidly decaying input signal reaches 1/16th and 1/256th, respectively, of the initial value. These values are chosen arbitrarily and represent instants when the signal is considered seriously weakened and completely absent, respectively. In the BlenX studies, the signal was constant and was switched off at a predefined point, giving rise to more natural cut-off points.

Execution The problem was split into 16 obvious subtasks, one for every site in the network. The procedure given below was the same for each subtask. A simple Haskell program generated all possible $2^{16} = 65536$ reconfigurations of the site. These were catenated, one-by-one, with a fixed species definitions file containing the definitions from Fig. 6.3 and passed to the prototype $c\pi$ tool (§4.4.1), which generated one ODE system for every variant. The resulting 65536

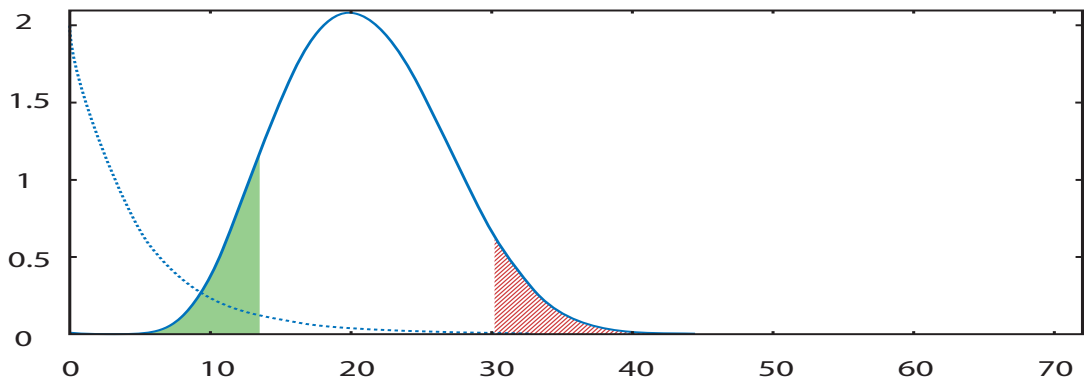


Figure 6.4: The input and output signal of the base MAPK model, and the fitness computation principle. The dotted line is the input (Ras) and the solid one is the output (ERK**). The fitness is the green area (left) minus the red area (right).

ODE systems were grouped in 128 batches of 512 scripts and solved using Octave on the Edinburgh Compute and Data Facility [40] parallel cluster. The solutions were then processed by another script containing a simple LTL checker and a fitness computation function.

Creation of Octave scripts took on average 2 hours, solving ODEs an hour, and LTL checking and fitness computations, which were not parallelised, further two hours (times per subtask). Given that all three steps of this procedure can be further optimised, it seems safe to conclude that 2 orders of magnitude more variants can be assessed effectively, which in case of this particular model means that a much more comprehensive network sampling could be performed.

Results and discussion Results of the experiment are summarised in Figs. 6.5–6.7. The first observation is that oscillatory behaviour is completely absent: not a single mutant satisfies the oscil formula in Eq. (6.1). There are indications that oscillatory behaviour, while rare, does arise in small protein networks via qualitative changes to the network topology only [136]. The fact that we obtained no oscillatory dynamics in our experiment underlines the contribution of species definitions to the overall structure of the network.

Secondly, the peak phenotype is maintained by a non-trivial fraction of mutants, indicating a degree of robustness of the cascade's basic function to changes in protein binding affinities. The robustness of individual sites varies, however (Fig. 6.7). Furthermore, 45% of all mutants have the switch phenotype,

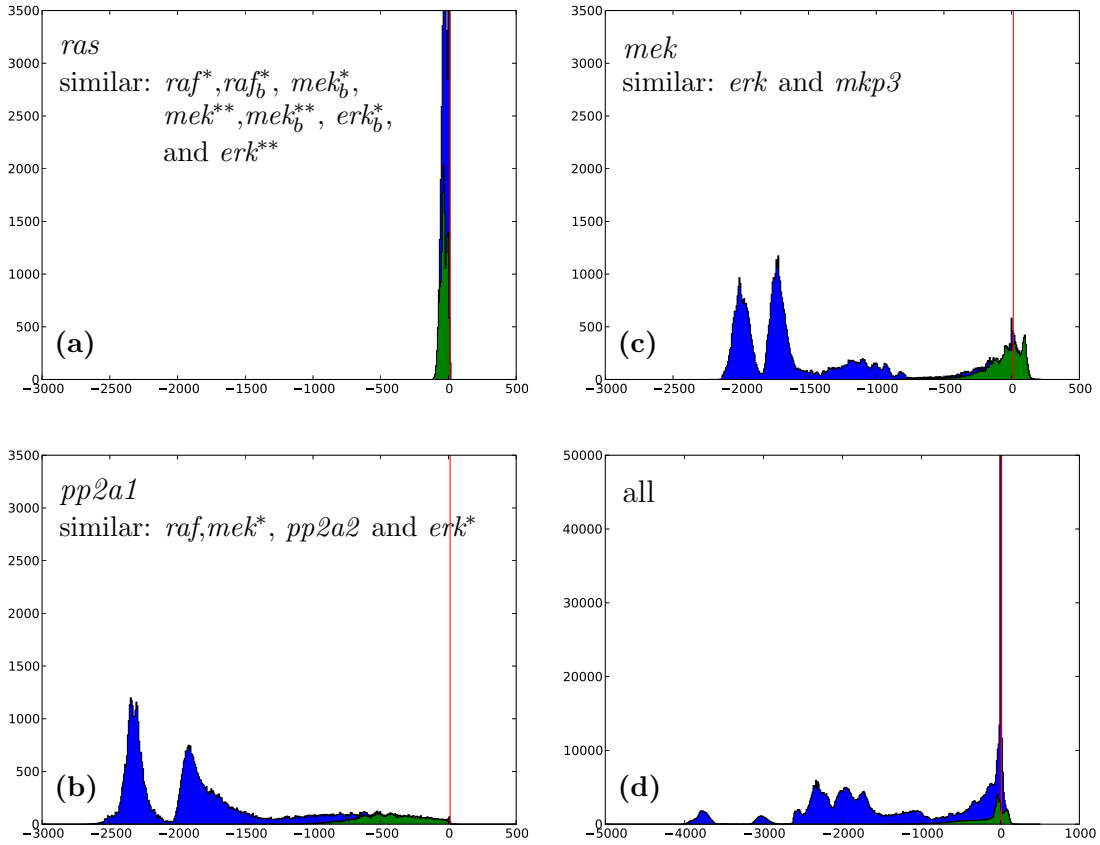


Figure 6.5: Three typical fitness distributions of variants of specific sites (a)–(c) and the distribution of fitness of all mutants (d). Regions of the distribution contributed by variants satisfying the peak formula (6.2) are marked in green. The red vertical line gives the position of the initial model. All histograms are made using 500 evenly sized bins; spikes in (a) and (d) reach 3.0e+4 and 3.0e+5, respectively; note the different scale used in (d).

meaning that this kind of response is readily accessible from the current architecture.

Thirdly, the 16 binding sites can be organised in three categories, according to the fitness distributions of associated mutants (Fig. 6.5). The sites of the first kind (Fig. 6.5a) are robust to reconfiguration and the corresponding histogram is concentrated close to the fitness of the initial model. Mutants of the sites of the second kind (Fig. 6.5b) typically belong to one of two very low fitness regimes. These low fitness groupings are also present in the distributions of the third kind (Fig. 6.5c), but here there is also a non-trivial fraction of mutants—composed exclusively of variants satisfying the peak formula in Eq. (6.2)—with fitness higher than that of the initial model.

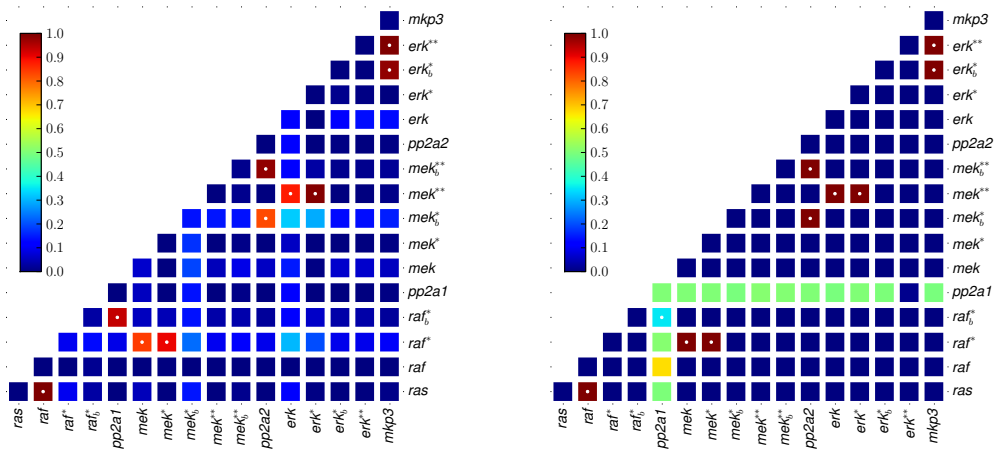


Figure 6.6: Distribution of edges in the advantageous mutants (left) and in the leftmost peak of the low-fitness mutants of *pp2a1* (right). Every square is an edge, and the colour gives its frequency in the analysed group of mutants. Edges present in the affinity network of the base model are marked with a small white dot.

In order to establish the cause of the two low-fitness peaks and the mechanisms conferring fitness advantage, we have filtered out the variants belonging to these fitness regimes and analysed the frequencies of individual edges in their affinity networks (Fig. 6.6). It turns out that the low fitness mutants of the *pp2a1* site are more likely to have the *pp2a1–raf* and less likely to have the *pp2a1–raf*$^*_b$ connection than average. Hence, low fitness of these mutants is due to PP2A1 not fulfilling its primary rôle as the Raf-specific phosphotase and/or preventing Raf from activating by binding it temporarily. The separation into two peaks is governed by the status of the *pp2a1–erk*** connection: the variants that evolved it belong to the higher-fitness peak, the ones that did not—to the other. There appears to be no single mechanism behind an increase in fitness, but feed-forward (e.g. *raf***–erk*) and extra dephosphorylation reactions (e.g. *erk***–mek*$^*_b$) are common.

Lastly, the histogram of fitness of all mutants (Fig. 6.5d) is again concentrated around the fitness of the initial model, but has also a non-trivial fraction of mutants of low fitness. Admittedly, this distribution does not resemble the gamma or log-normal distributions postulated in biological literature, but it does exhibit at least two properties generally expected of distributions of mutational effects: a vast majority of mutations are deleterious, and most have little impact on fitness [42].

Summary We have performed a large-scale computational exploration of evolutionary characteristics of the immediate evolutionary neighbourhood of an important biochemical system. We have used the framework of variation operators to generate a sample of this neighbourhood and relied on the separation of syntax and semantics in $c\pi$ to perform two very different kinds of analysis on it. We have exhibited fragments of neutral spaces of cascade models using LTL checking and provided a finer-grained, quantitative analysis of the same using ODE solving. The results we have obtained are a healthy mix of the expected and the surprising, which further increases our confidence in the soundness and usefulness of the framework of variation operators.

	phenotype class				fitness correlate					
	% mutants				peak mutants only			all mutants		
	peak	oscil	switch	noise	mean	median	std. dev.	mean	median	std. dev.
ras	24.21	0.00	0.00	75.79	-33.94	-34.16	24.34	-14.60	0.09	22.15
raf	0.00	0.00	99.90	0.10	N/A	N/A	N/A	-1992.59	-2001.40	259.08
raf^{*}	2.44	0.00	17.01	80.55	23.89	16.29	40.26	-246.10	0.19	584.34
raf^{*}_{b}	0.61	0.00	0.01	99.38	10.51	30.51	92.71	3.02	1.56	29.37
pp2a1	12.39	0.00	81.03	6.58	-438.29	-434.15	236.40	-1740.54	-1902.03	651.19
mek	19.82	0.00	75.86	4.32	-64.02	-28.72	155.47	-1305.92	-1688.53	743.11
mek^{*}	4.51	0.00	91.78	3.71	-686.69	-709.32	298.18	-1918.09	-1993.59	419.05
mek^{*}_{b}	0.52	0.00	11.53	87.95	23.08	33.02	40.15	-220.67	-26.47	387.81
mek^{**}	1.41	0.00	9.39	89.20	-175.59	-109.51	197.79	-143.04	0.10	413.45
mek^{**}_{b}	0.05	0.00	0.73	99.22	-69.35	-96.01	77.51	-15.47	2.09	116.58
pp2a2	9.25	0.00	86.48	4.27	-380.31	-357.05	237.78	-1912.88	-2065.28	632.26
erk	16.57	0.00	69.48	13.95	48.04	64.04	67.43	-717.68	-931.54	512.89
erk^{*}	9.50	0.00	87.89	2.62	-351.40	-323.49	268.31	-1945.92	-2155.53	647.28
erk^{*}_{b}	0.61	0.00	3.59	95.80	-123.26	-105.60	110.36	-168.05	-129.65	210.46
erk^{**}	0.05	0.00	0.55	99.40	-192.41	-195.73	157.44	-72.13	-40.21	115.26
mkp3	10.28	0.00	88.20	1.52	-230.63	-209.37	207.78	-2853.67	-3100.71	1198.07
overall	7.01	0.00	45.22	47.77	-172.08	-56.78	264.22	-954.08	-341.57	1078.10

Figure 6.7: Simple statistics of phenotype and fitness distributions.

Chapter 7

Conclusions

In this chapter we summarise and evaluate the work contained in this dissertation.

The main conclusion we offer is that a process algebra can successfully serve as a generic, formal and computational framework for the study of evolution of molecular networks. The features we identified in the Introduction as potentially conducive to this application have all proved very useful indeed. We have used the syntax/semantics separation to ensure that variation operators are oblivious to the effect they have on the dynamics of the model. The process-as-molecule abstraction has enabled us to model the evolutionary changes directly as formal transformations of species and processes. Finally, the ability to generate multiple analyses from a single description has allowed us to study variants of an important molecular pathway qualitatively and quantitatively in a single computational experiment. Overall, we are confident that the application of process algebras in theoretical evolutionary biology along the lines presented in this thesis is an idea worth pursuing well beyond our initial investigation.

This approach is not without problems, however. In particular, process-algebraic models of biochemical systems tend to be unnecessarily complicated and idiosyncratic, which has a negative impact on their accessibility and adoption by a wider scientific community. The technical manipulations necessary to rigorously define models of evolutionary variation are even more off-putting, as the reader probably agrees having seen Ch. 5 of this dissertation. It is unclear at present whether the unique features of process algebras, such as compositionality and behavioural equivalences, eventually outweigh these shortcomings; we discuss these issues in more detail below (§7.1.2). This chapter contains also a more detailed survey of the contributions of this work (§7.1.1) and proposes future research directions (§7.2).

7.1 Evaluation

7.1.1 Contributions

A novel process algebra The first major contribution of this thesis is a novel process algebra for biochemical modelling: the continuous π-calculus. Its usability for general-purpose dynamic modelling of molecular pathways is on a par with that of the π-calculus, and is certainly greater for model perturbation experiments. Both claims are well-supported by the KaiABC model and its analysis. The organisation of names in affinity networks affords a considerable flexibility to the interaction patterns of species, which in turn makes $c\pi$ a good platform for the development of a formal treatment of mutations. The novel continuous-state and time semantics developed to reduce the computational cost of quantitative evaluation of multiple variants of the same model serves this purpose well, as evidenced by the analysis of mutants of the MAPK cascade.

Two properties of $c\pi$ semantics are of additional interest from a more theoretical point of view. The first one is the ability to extract systems of ODEs from $c\pi$ models in a fully compositional fashion, potentially reducing the computational cost of extraction, and the related possibility of approximating of infinite sets of ODEs with sequences of finite ones (§3.4). The other is the separation of the overall dynamics of a composite process into the autonomous contributions of its components and the emergent behaviour due to their cross-interaction. This separation, best visible in Eq. (3.22), provides the modeller with extra information about the dynamics of his model.

A setup for evolutionary analysis The other major contribution of this dissertation is the framework of variation operators. The operators defined in this thesis cover a number of qualitatively distinct mutational effects and are expressive enough to generate models of considerable complexity and biological interest, such as the enzyme inhibition and MAPK cascade models. They also largely avoids the major pitfall of *in silico* evolution of complex models: confusion of mutational events with mutational effects. Every operator is associated with a genetic event, and thus the structure they give to the space of models has evolutionary relevance.

This structure is the key ingredient of the theory of neutral spaces [152], and thus we have succeeded in recasting this framework in process-algebraic terms.

Although we have not exhibited entire neutral spaces of MAPK models in our experiments, we have shown that it is both conceptually possible and computationally feasible. In the same experiment we have evaluated an abstract fitness function for variants of a MAPK cascade model, thus complementing the qualitative LTL analysis. The theoretical setup provided by $c\pi$ and variation operators is thus capable of analysing non-trivial evolutionary properties of complex genotype-phenotype relationships, and as such has its place in theoretical evolutionary developmental biology.

Models of biochemical systems We have built relatively large $c\pi$ models of two biochemical systems: the KaiABC circadian clock and the MAPK cascade. Although their primary purpose have been demonstration of the features of $c\pi$ and variation operators, they are fully functional and form a good starting point for a more advanced analysis of these systems. They should be viewed, therefore, as secondary contributions of this thesis.

7.1.2 Problems

We now highlight two major issues with process-algebraic modelling of biochemical systems. They are not specific to the continuous π-calculus or aggravated by the application to evolutionary biology. It is nevertheless important to make them explicit here, because they do have a bearing on our work.

Molecules do not pass names The ability to dynamically form complexes via private name passing is a major advantage of π-related process algebras over other calculi for biology. This mechanism, however, has two major negative consequences from the point of view of the modeller. The first one is that in order to ensure that the constituents of a complex are not treated separately, they both have to use the shared name(s) as communication capabilities, even if there is no other reason for doing so. While the modeller is often able to accommodate this requirement without distorting the behaviour of the model, the mere need for such an accommodation is an unnecessary complication. The second problem is that scope extrusion quickly leads to illegible models, which severely limits its usability as a primitive modelling concept, especially with complexation being such an ubiquitous molecular event. In short, complex formation should in future be modelled by a dedicated construct, preserving the advantages of private name

passing (in particular the capacity for dynamic formation of complexes), but doing away with the flaws outlined above.

Defining features of process algebras are underused Why use process algebras for biological modelling in the first place? We have argued previously that they offer an attractive combination of properties: formality, succinctness, compositionality, model checking opportunities, behavioural equivalences and the syntax-semantics separation. However, at least three of the above have had very little impact on biological modelling so far. Compositionality has an undeniable potential for knowledge organisation, and yet we are not aware of any modelling efforts where it played an essential rôle. Quantitative model checking of non-trivial biological models is currently intractable. Lastly, biologically relevant behavioural equivalences are still in their infancy. Formality and succinctness are of course very important and advantageous, and a lot of work—including this thesis—makes good use of the clear separation of syntax and semantics, even if it is not explicitly acknowledged. However, should only these three features prove truly important in biological modelling, we are likely to see process algebras replaced with less idiosyncratic, purpose-built formal languages for biology.

7.2 Future work

In this section we outline promising and important directions of future research. The main continuation of this dissertation should be, of course, development of additional variation operators and study of important evolutionary questions. Here, we focus on less obvious challenges instead.

7.2.1 A better continuous π-calculus

Better theory The presentation of the continuous π-calculus offered in this dissertation constitutes a major improvement over the early version [88], but we believe that further enhancements are possible and desired. As an example, consider the concept of compatibility of concretions (Def. 3.3.2): two concretions are compatible iff they have complementary lengths of their send/receive name vectors. This notion is required by the polyadic synchronisation paradigm we implemented. We did use polyadic name passing in our models, but it was never

essential. However, the mere fact that two concretions may be incompatible means that any variation in the connectivity structure of names (global affinity network) may be constrained for purely technical reasons, which is unacceptable. The obvious solution is to abandon polyadic name passing, but retain symmetric prefixes, thus forcing every name to send one and receive one name and making all concretions compatible. Similar imperfections and "frozen accidents" affect other aspects of $c\pi$, and the next version of the calculus should do away with them as well.

Better support The $c\pi$ software tool (§4.4.1) requires a complete reimplementation. The next version should properly recognise equivalent species, accept literate (programming-language style) and brief (process-calculus style) model descriptions, and, above all, compile $c\pi$ models to a range of target formalisms and analyses, necessarily including ODEs, stochastic simulations, sets of chemical reactions and SBML.

7.2.2 Infinitely supported processes

The π-calculus and its variants have the unique capacity to generate unbounded numbers of non-equivalent processes from finite descriptions. The corresponding set of ODEs, which consists of one equation per molecular species present at any point of the dynamic evolution of the system, may therefore be infinite. It is unclear how to handle infinite sets of nonlinear ODEs, in particular what to expect from their solutions, when there are any. This is why we chose conservatively $\mathbb{P} \stackrel{\text{df}}{=} \mathbb{R}^{(\mathcal{S}^\#)}$ as our process space rather than one of its better behaved subspaces: $\mathbb{P}_{\text{fin}} \stackrel{\text{df}}{=} \{p \in \mathbb{P} : \mathsf{supp}(p) \text{ is finite}\}$ or one of $\mathbb{P}_n \stackrel{\text{df}}{=} \{p \in \mathbb{P} : \sum_{A \in \mathcal{S}^\#} p(A)^n < \infty\}$. Admittedly, these issues have little importance in modelling, where the number of molecular species involved is usually finite; polymerisation stands out as the only possible exception to that rule. From the theoretical perspective, however, they are very important, particularly because the ability to generate unbounded numbers of species confers Turing universality on biochemistry [25].

In order to give a satisfactory account of infinite and potentially infinite processes, one has to find a suitably normed subset of $\mathbb{P}$ to serve as process space and to generalise $c\pi$ semantics to operate on it. It can be shown that neither $\mathbb{P}_{\text{fin}}$ nor any $\mathbb{P}_n$ with the standard norms can play that rôle: $\frac{d-}{dt}$ is nowhere continuous on the former, and $\partial- \oplus \partial-$ is not always defined on the latter. A recent article

addresses similar questions in the context of the stochastic π-calculus [24]. A complementary task is to provide syntactic conditions on definitions of species under which the corresponding set of ODEs is finite, or is guaranteed to have a solution. Here, the recent work on π to Petri Net translations [96] and the fluid spatial π-calculus [141] may be good places to start.

7.2.3 A dedicated $c\pi$ logic

for Chris Banks

The non-standard continuous semantics of $c\pi$ open intriguing opportunities for model checking of $c\pi$ models using continuous logics. Here we suggest a dedicated logic based on LTL that can greatly extend the applicability of $c\pi$ to the study of biochemical systems.

A proposal for the syntax of such logic can be found in Fig. 7.1. The formulae are built of polynomials over real variables, real constants, concentrations of species (e.g. $[S]$), their gradients ($[\dot{S}]$) and site affinities ($Aff(a, b)$). Apart from standard boolean connectives, first-order quantification over reals and names, and the bounded LTL "future" quantifier, the logic has two novel quantifiers. The first is similar to the "future" quantifier, but the behaviour of the system is assessed in the presence of an extra agent: the model $(\mathcal{D}, N, P)$ satisfies the formula $\mathsf{F}_t^{c \cdot a}\phi$ if P reaches, within t time units and in the continued presence of the process $c \cdot a.\mathbf{0}$, a state satisfying ϕ. The other quantifier is concerned with extensions of affinity networks: the model $(\mathcal{D}, N, P)$ satisfies the formula $\mathsf{И}_x\phi$ if, for any f, $(\mathcal{D}, N \oplus_f x, P)$ satisfies ϕ. Working in tandem, the two novel quantifiers can express a range of complex properties, e.g.

$$\forall_{c>100}\mathsf{И}_x(Aff(x, a) > 2.0 \implies \mathsf{F}_t^{c \cdot x}\phi) \tag{7.1}$$

Depending on the meaning of ϕ, the above formula can be interpreted e.g. in a general biochemical/drug design context ("If we introduce an agent interacting with the site a at a basal rate greater than 2.0 in a concentration greater than 100, then the system reaches the healthy state ϕ within time t"), but also in an evolutionary one ("If a protein with a site capable of interacting with a at a rate greater that 2.0 evolved, then when expressed above the concentration of 100 units, it would cause the protein network to reach the novel state ϕ within time t").

`name`	::=	$a, b, ..., x, y, ...$	
`prefix`	::=	$(\boldsymbol{\nu} M)a(\vec{x}, \vec{y})$	$M, \vec{x}, \vec{y}$ may be null
`real variable`	::=	$r, s, ...$	
`constant`	::=	$\underline{r}, \underline{s}, ...$	$\underline{r}, \underline{s} \in \mathbb{R}$
`quantity`	::=	$[S]$, $[\dot{S}]$, $Aff(a, b)$	S species, a, b names
`expr`	::=	`real variable`	
		| `constant`	
		| `quantity`	
		| f`(expr, ..., expr)`	$f : \mathbb{R}^n \to \mathbb{R}$ poly.
`formula`	::=	`(name = name)`	
		| `expr > 0`	
		| $\neg$ `formula`	
		| `formula` $\vee$ `formula`	
		| $\exists_{\texttt{real variable}}$ `formula`	
		| $\exists_{\texttt{name}}$ `formula`	
		| $\mathsf{F}_{\texttt{constant}}$ `formula`	
		| $\mathsf{F}^{\texttt{constant}\cdot\texttt{prefix}}_{\texttt{constant}}$ `formula`	
		| $\mathit{И}_{\texttt{name}}$ `formula`	

Figure 7.1: The syntax of the proposed $c\pi$ logic.

In order to formally define the satisfaction relation, we believe that $c\pi$ processes should be given semantics in terms of flow lines in a topological vector space. This can either be the Euclidean space $\mathbb{R}^n$, if the processes can be guaranteed to give rise to finitely many prime species, or an infinitely-dimensional Banach space. Thus, we see solving the problem posed in §7.2.2 as a prerequisite for developing a dedicated $c\pi$ logic. Once a satisfaction relation is defined, it may be possible to develop a model checking algorithm using the exact real arithmetic and interval analysis techniques, in particular the work of Edalat and Pattinson [38, 39].

7.2.4 Richer affinity networks

We have identified two possible improvements of the affinity network concept. As they are independent, we discuss them separately.

Typed sites The first improvement consists of requiring names to carry an abstract *type*. A type should be a reflection of the biochemistry of the protein

active site modelled by the name, or of similar domain-specific knowledge. Any two names can only be connected by an edge in an affinity network if they are of complementary types. This requirement would provide a mechanism for excluding biologically unrealistic or uninteresting interaction patterns from the analysis. To see why this may be useful, cast your mind back to the analysis of MAPK cascade variants. It turns out that in the fittest of all mutants the *mek* site is connected to the *erk** site; upon interaction of these two sites, both MEK and ERK* are phosphorylated, which is not biologically realistic. One can of course manually or semi-automatically remove the troublesome variants, as was done in the BlenX studies (§6.3.1), but a more general approach such as the one described above would definitely constitute an improvement.

More kinetic laws At present, all reactions in $c\pi$ follow the Law of Mass Action. While this is assumed to faithfully reflect the low-level molecular kinetics and does not preclude the emergence of more complex dynamical behaviour (see e.g. the Michaelis-Menten kinetics exhibited by our enzyme example), it is still a serious limitation, because the exact configuration of the underlying mass-action reactions may be unknown or simply unimportant (§2.2.2). The solution is to allow names to interact according to arbitrary kinetic laws; this *functional rate* approach has already been used in Bio-PEPA (see §2.3.3) for the same reason. It can be implemented in $c\pi$ by labelling the edges of affinity networks with indication and parameters of a kinetic function. While straightforward from the syntactic point of view, this modification would have far-reaching consequences for the semantics. In particular, the interaction function $- \oplus -$ would not be bilinear anymore and the crucial Thm. 3.3.16 would have to be proven anew.

Appendix A

Proof of Theorem 3.2.8

In this appendix we give a detailed proof of Thm. 3.2.8 (page 33).

Statement For every $A \in \text{SPEC}$, there exists a unique multiset $\mathsf{primes}(A) = \{\!|A_1, \ldots, A_n|\!\} \subset \mathcal{S}^{\#}$ called the *prime species decomposition of* A, such that $A \equiv A_1 \mid \ldots \mid A_n$.

Related work Moller and Milner [100] prove a similar decomposition result for the CCS process algebra. However, they decompose processes with respect to bisimulation, and not structural congruence. As a result, their proof is difficult to adapt to our case.

Luttik and van Oostrom [92] give general conditions for a commutative monoid (in our case $\langle \mathcal{S}, |, \mathbf{0} \rangle$) to have the unique decomposition property. While we would very much like to use this work, the necessary conditions (in particular a cancellation law: $A|B \equiv A|C$ implies $B \equiv C$) appear to be as hard to prove in our case as the main result.

In view of the abovementioned issues, we give a direct proof.

Proof idea and outline It is a common proof technique for equational specifications of syntactic relations (such as $\equiv$) to *direct* all the equations and thus turn the specification into a term rewriting system. If the system is terminating and confluent then every term has a unique normal form which can be taken as a canonical representative of the equivalence class of the relation under study.

In our case, we would like to have a rewrite system on SPEC such that its unique normal forms are parallel compositions of prime species. Unfortunately, it turns out that some of the rules defining $\equiv$, most notably the associativity of $|$, have to be used in both directions to properly extract prime components, while others (especially commutativity of Σ) are not necessary at all for this task. Unfortunately, if a rule can be used in both directions, the rewrite system is not terminating.

The main idea of the proof is to use these problematic rules to define a subrelation $\sim$ of $\equiv$ and direct the remaining ones to give a terminating and confluent rewrite system on SPEC modulo $\sim$. The normal forms of this system still correspond to prime decompositions and are unique, thus yielding the desired result. The price to pay is the technical nature of the proof, arising from juggling three congruences on SPEC: the equality $=$, the auxiliary relation $\sim$ and the structural congruence $\equiv$.

The entire discussion above applies to the proof of uniqueness of prime decompositions. Showing existence does not pose a major problem, and we give a simple proof in Thm. A.1.

The remainder of this appendix is structured as follows:

(i) **(existence)** We show (A.1) that every species has a prime decomposition.

(ii) We define (A.2) an auxiliary congruence $\sim$ on SPEC such that $\sim \subset \equiv$ (A.3).

(iii) We define (A.5) a terminating and confluent (A.6) reduction system on $\mathcal{S}$ modulo $\sim$.

(iv) We prove that if $A \equiv B$, then A and B have the same normal forms (A.8).

(v) **(uniqueness)** We prove that any two prime decompositions of a given species are equal by exploiting the fact that their normal forms are (A.9).

Theorem A.1. Let $A \in \text{SPEC}$. There exists a multiset $\{\!|A_1, \ldots, A_n|\!\} \subset \mathcal{S}^\#$ such that $A \equiv A_1| \cdots |A_n$.

Proof. There is a simple procedure for constructing a prime decomposition. If $A \equiv \mathbf{0}$, take $\varnothing$. If $A \not\equiv \mathbf{0}$, begin with $\{\!|A|\!\}$; if A is prime, we're done. If not, $A \equiv B|C$ for $B, C \not\equiv \mathbf{0}$, so we take $\{\!|B, C|\!\}$. If B and C are prime, we're done; if not, they can be decomposed again, and so on. It remains to show that this procedure terminates. Towards this goal, define a measure $\mu\colon \text{SPEC} \to \mathbb{N}$ by

$$\mu(A) \stackrel{\text{df}}{=} \begin{cases} 0 & A = \mathbf{0} \\ 1 & A = D(\vec{x}) \text{ or } A = \Sigma_{i=0}^{n} \pi_i.A_i \\ \mu(B) + \mu(C) & A = B|C \\ \mu(B) & A = (\boldsymbol{\nu} M)B \end{cases}$$

Observe that (1) $\mu(A) = 0$ iff $A \equiv \mathbf{0}$, and (2) if $A \equiv B$ then $\mu(A) = \mu(B)$. Thus, the size of any decomposition of A into non-zero parallel components is bounded by $\mu(A)$. But the procedure outlined above necessarily increases the size of the decomposition at each step, and thus must terminate. □

Definition A.2. Define $\sim$ as the smallest congruence on Spec containing α-equivalence and satisfying the following rules:

- $A|(B|C) \sim (A|B)|C$,
- $A|B \sim B|A$,
- $(\boldsymbol{\nu} M)(\boldsymbol{\nu} N)A \sim (\boldsymbol{\nu} N)(\boldsymbol{\nu} M)A$ when $M \# N$,
- $\Sigma_{i=0}^{n} \pi_i.A_i \sim \Sigma_{i=0}^{n} \pi_{\sigma i}.A_{\sigma i}$ when σ is a permutation.

We write $\tilde{\mathcal{S}}$ for Spec modulo $\sim$ and $[A]$ for the equivalence class of A w.r.t. $\sim$.

Proposition A.3. Let $A, B, A_i, B_j \in$ Spec for all i, j. The following statements hold ((ii) and (iii) under any bracketing):

(i) If $A \sim B$ then $A \equiv B$,

(ii) If $(A_1|\cdots|A_n) \sim (B_1|\cdots|B_k)$, and no A_is and B_js are parallel compositions, then $\{\!|[A_1], \ldots, [A_n]|\!\} = \{\!|[B_1], \ldots, [B_k]|\!\}$,

(iii) If $(A_1|\cdots|A_n) \sim (B_1|\cdots|B_k)$, and no A_is and B_js are parallel compositions, then $\{\!|A_1, \ldots, A_n|\!\}$ and $\{\!|B_1, \ldots, B_k|\!\}$ are equal as multisets over $\mathcal{S}$.

Proof. The first statement holds because any derivation of $A \sim B$ is also a derivation of $A \equiv B$; the second is proven easily by induction on the length of the proof of $(A_1|\cdots|A_n) \sim (B_1|\cdots|B_k)$; the third follows from the previous two. □

Definition A.4. Define the term rewriting system $\mathcal{T} \stackrel{\text{df}}{=} (\text{Spec}, \rightarrow)$ by the following rules:

(i) $A|\mathbf{0} \rightarrow A$,

(ii) $(\boldsymbol{\nu} M)A \rightarrow A$ when $M \# A$,

(iii) $(\boldsymbol{\nu} M)(A|B) \rightarrow A|(\boldsymbol{\nu} M)B$ when $M \# A$.

Definition A.5. Define the abstract rewrite system $\mathcal{Z} \stackrel{\text{df}}{=} (\mathcal{S}_\sim, \rightsquigarrow)$ by setting

$$[A] \rightsquigarrow [B] \text{ iff } \exists_{A' \sim A} \exists_{B' \sim B} (A' \to B').$$

Lemma A.6. System $\mathcal{Z}$ is terminating and confluent.

Proof. For termination, suppose that we have an infinite sequence $[A_0] \rightsquigarrow [A_1] \rightsquigarrow [A_2] \rightsquigarrow \cdots$. By definition, this implies an infinite sequence of species terms: $A_0 \sim A_0' \to A_1' \sim A_1 \sim A_1'' \to A_2' \sim A_2 \sim A_2'' \to \cdots$. Because rules (i) and (ii) of $\mathcal{T}$ remove symbols ($\mathbf{0}$ and ($\boldsymbol{\nu}$–), respectively) whose total number is preserved by $\sim$, after finitely many steps all reductions have to be applications of rule (iii). It is now relatively easy to reach a contradiction, for example by observing that each application of (iii) reduces the total number of | symbols under restriction (which again is preserved by $\sim$).

For confluence we use Newman's Lemma [107], stating that in terminating reduction systems, confluence is equivalent to weak confluence (the ability to join any two divergent one-step reductions). It is easy to verify this property for all six rule pairs, and we leave it to the reader. □

Definition A.7. Let $A \in \textsc{Spec}$. We write $\mathsf{nf}(A)$ for the $\mathcal{Z}$-normal form of $[A]$.

Lemma A.8. Let $A \equiv B$ for $A, B \in \textsc{Spec}$. Then $\mathsf{nf}(A) = \mathsf{nf}(B)$.

Proof. If $A \equiv B$, then there exists a proof: a sequence of terms $(u_i)_{i=0}^n \subset \textsc{Spec}$ such that $u_0 = A, u_n = B$ and for every $j \leq n$, $u_j \equiv u_{j+1}$ is an instance of one of the rules defining $\equiv$ (see Fig. 3.3(l)). But each rule is either used by $\sim$ or by $\to$, and thus for every $j < n$, we have either $u_j \sim u_{j+1}$ or $u_j \to u_{j+1}$ or $u_{j+1} \to u_j$. Moving to $\tilde{\mathcal{S}}$, we obtain the sequence $([u_i])_{i=0}^n \subset \tilde{\mathcal{S}}$, with $[A] = [u_0]$, $[B] = [u_n]$, and where for every $j < n$ either $[u_j] = [u_{j+1}]$ or $[u_j] \rightsquigarrow [u_{j+1}]$ or $[u_{j+1}] \rightsquigarrow [u_j]$. Hence, $[A]$ and $[B]$ are convertible in $\mathcal{Z}$, and as such have the same normal form. □

Theorem A.9. Let $A \in \mathcal{S}$ be a species. Suppose $\mathcal{A} = \{\!|A_1, \ldots, A_n|\!\} \subset \mathcal{S}^\#$ and $\mathcal{B} = \{\!|B_1, \ldots, B_k|\!\} \subset \mathcal{S}^\#$ are prime species decompositions of A. Then $\mathcal{A} = \mathcal{B}$.

Proof. Since $(A_1|\cdots|A_n) \equiv (B_1|\cdots|B_k)$, then by Lem. A.8 $\mathsf{nf}(A_1|\cdots|A_n) = \mathsf{nf}(B_1|\cdots|B_k)$, and as a consequence, $(\mathsf{nf}(A_1)|\cdots|\mathsf{nf}(A_n)) = (\mathsf{nf}(B_1)|\cdots|\mathsf{nf}(B_k))$. Pick a representative of $\mathsf{nf}(A_i)$; it can be written as $(A_i^0|\cdots|A_i^{s_i})$, with $s_i \geq 0$ and all parallel components shown. By primality of A_i, all but one of these

components are equivalent to $\mathbf{0}$. Assume wlog that only A_i^0 is not; hence $A_i^0 \equiv A_i$ and $A_i^j \equiv \mathbf{0}$ for $j > 0$. By performing the same construction on B_is we arrive at

$$(A_1^0|\cdots|A_1^{s_1})\,|\,\cdots\,|\,(A_n^0|\cdots|A_n^{s_n}) \;\sim\; (B_1^0|\cdots|B_1^{t_1})\,|\,\cdots\,|\,(B_k^0|\cdots|B_k^{t_k}).$$

By Prop. A.3(iii), therefore, we have

$$\{\!|A_1^0,\cdots,A_1^{s_1},\cdots,A_n^0,\cdots,A_n^{s_n}|\!\} \;=\; \{\!|B_1^0,\cdots,B_1^{t_1},\cdots,B_k^0,\cdots,B_k^{t_k}|\!\},$$

with both multisets over $\mathcal{S}$. As equal, both multisets must have the same number of $\mathbf{0}$-equivalent terms in them: they can be removed and two equal multisets will remain. But the terms equivalent to $\mathbf{0}$ are, by construction, precisely all A_i^js and B_i^js for $j \geq 1$. Removing them from both multisets simultaneously leaves us with:

$$\{\!|A_0^0,\cdots,A_n^0|\!\} = \{\!|B_0^0,\cdots,B_k^0|\!\},$$

but of course lhs is $\mathcal{A}$ and rhs is $\mathcal{B}$. □

Table of symbols

Chapter 3: The continuous π-calculus

$a, b, \ldots, x, y, z, \ldots$	names
$\vec{a}, \vec{b}, \ldots, \vec{x}, \vec{y}, \vec{z}, \ldots$	vectors of names
$\lvert\vec{x}\rvert$	length of the vector $\vec{x}$
$\mathcal{N}$	the set of names
$\pi, \pi_i, \pi', \ldots$	prefixes
$\tau@k$	silent/spontaneous prefix
$a(\vec{x}; \vec{y})$	communication prefix
$M, N, K \ldots$	affinity networks
$A, B, \ldots$	species
$A \subset B$	A is a subspecies of B
$\mathcal{H}$	the set of definition handles
Spec	the set of species
$\mathcal{S}$	Spec modulo species equivalence
$\mathcal{S}^{\#}$	the set of prime species
$\mathsf{primes}(A)$	the prime decomposition of A
$P, Q, \ldots$	processes
$A \subset P$	A or its superspecies appears in P
Proc	the set of processes
$\mathcal{P}$	Proc modulo process equivalence
fn	free names
bn	bound names
$x\#E$	x is fresh for E
$F, G, \ldots$	concretions
Conc	the set of concretions
$\mathcal{C}$	Conc modulo concretion equivalence
$F \circ G$	pseudo-application of F and G
$F \downarrow G$	F and G are compatible

Trans	the multiset of transitions of $c\pi$ species
$\mathit{Trans}(A)$	the multiset of transitions with source A
$\mathsf{supp}(P)$	the support of P
$\mathsf{card}(x, X)$	the multiplicity of x in the multiset X
$\mathbb{P}$	the process space
$\mathbb{D}$	the space of potentials
$\oplus_M$	the interaction tensor (in the context of M)
$\frac{dP}{dt}$	the immediate behaviour of P
∂P	the interaction potential of P

Chapter 5: Variation operators

$\xi, \zeta, \eta, \ldots$	definitions of species
$\mathcal{D}, \mathcal{E}, \ldots$	sets of species definitions
$(\mathcal{D}, N, P), (\mathcal{E}, M, Q), \ldots$	operator-ready $c\pi$ models
$\mathsf{vs}_{\mathcal{M}}$	virtual sites in model $\mathcal{M}$
$\mathsf{vs}_{\mathcal{M}}(A)$	virtual sites of the definition with handle A
$\mathsf{ps}_{\mathcal{M}}(A)$	physical sites of the definition with handle A
$\xi \circ\!\!\rightarrow \zeta$	ξ depends on ζ
$[\xi]_{\mathcal{D}}$	the class of ξ w.r.t. the equivalence induced by $\circ\!\!\rightarrow$
$A \circ\!\!\rightarrow \zeta$	species A invokes definition ζ
$A[W \mapsto V]$	substitution of handle W by V in species A
$N \ominus a$	N restricted to $N \setminus \{a\}$
$N \ominus X$	N restricted to $N \setminus X$
$N \oplus_f a$	N extended with a (affinites of a given by f)
$N \odot_f a$	N with a reconfigured according to f
$N \oplus_{\vec{a}} \vec{x}$	N extended with $\vec{x}$ as a carbon copy of $\vec{a}$
$N(a, b, k)$	N with the affinity of a and b changed to k
$N(-, -, k)$	a set of variants of N
$\mathsf{rc}(A, k)$	A with an internal rate changed to k
$\mathsf{cg}(\pi, Z, A)$	A enriched with π or $\pi.Z$
$\mathsf{cl}(A)$	A with a single prefix and its successor state lost
$\mathsf{succ}_{\mathcal{M}}(\xi)$	eligible successor states for variants of ξ
$\sigma_{\vec{x} \rightsquigarrow \vec{y}}$	name substitution emulating $\{\vec{y}/\vec{x}\}$
$\mathcal{M}[\xi \mapsto \zeta]$	$\mathcal{M}$ with ξ substituted by ζ (with balancing).

Bibliography

[1] M. Abadi and A. Gordon, *A calculus for cryptographic protocols: The spi calculus*, Proc. 4th ACM Conference on Computer and Communications Security, ACM Press, 1997.

[2] U. Alon, *An introduction to systems biology: design principles of biological circuits*, Chapman & Hall/CRC, 2006.

[3] L. W. Ancel and W. Fontana, *Plasticity, evolvability, and modularity in RNA*, J. Exp. Zool. (Mol Dev Evol) **288** (2000), 242–283.

[4] J. C. M. Baeten, *A brief history of process algebra*, Theor. Comput. Sci. **335** (2005), 131–146.

[5] C. Baier and J.-P. Katoen, *Principles of model checking*, MIT Press, 2008.

[6] A. Bergman and M. Siegal, *Evolutionary capacitance as a general feature of complex gene networks*, Nature **424** (2003), 549–552.

[7] J. A. Bergstra and J. W. Klop, *Process algebra for synchronous communication*, Inform. Control **60** (1984), 109–137.

[8] J. A. Bergstra and C. A. Middelburg, *Process algebra for hybrid systems*, Theor. Comput. Sci. **335** (2005), 215–280.

[9] M. Bernardo, P. Degano and G. Zavattaro (eds.), *Proc. Eighth International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM-08:Bio)*, Lect. N. Bioinformat., vol. 5016, 2008.

[10] Biocham, `http://contraintes.inria.fr/BIOCHAM/`.

[11] BioSpi, `http://www.wisdom.weizmann.ac.il/~biospi`.

[12] G. Birkhoff and G.-C. Rota, *Ordinary differential equations*, John Wiley and Sons, 1978.

[13] C. Bodei, M. Buchholtz, P. Degano, F. Nielson and H. Nielson, *Automatic validation of protocol narration*, Proc. 16th Computer Security Foundations Workshop (CSFW 03), IEEE Press, 2003, pp. 126–140.

[14] J. A. Bolker, *Modularity in development and why it matters to evo-devo*, Am. Zool. **40** (2000), 770–776.

[15] N. Bonzanni, E. Krepska, W. J. Fokkink, T. Kielmann, H. Bal and J. Heringa, *Executing multicellular differentiation: quantitative predictive modelling of C. elegans vulval development*, Bioinformatics **25** (2008), 2049–2056.

[16] E. Borenstein and D. C. Krakauer, *An end to endless forms: Epistasis, phenotype distribution bias and non-uniform evolution*, PLoS Comput. Biol. **4** (2008).

[17] L. Bortolussi and A. Policriti, *Dynamical systems and stochastic programming – to ordinary differential equations and back*, Transactions on Computational Systems Biology **5750** (2009), 216–267.

[18] M. Calder, A. Duguid, S. Gilmore and J. Hillston, *Stronger computational modelling of signalling pathways using both continuous and discrete-state methods*, Proc. Fourth International Conference on Computational Methods in Systems Biology (CMSB 2006) (C. Priami, ed.), Lect. N. Bioinformat., vol. 4210, 2006, pp. 63–77.

[19] M. Calder and S. Gilmore (eds.), *Proc. Fifth International Conference on Computational Methods in Systems Biology (CMSB 2007)*, Lect. N. Bioinformat., vol. 4695, 2007.

[20] M. Calder, S. Gilmore and J. Hillston, *Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA*, Proc. Second Workshop on Concurrent Models in Molecular Biology (BioConcur 2004) (A. Ingolfsdottir and H. R. Nelson, eds.), 2004.

[21] ———, *Automatically deriving ODEs from process algebra models of signalling pathways*, Proc. Third International Conference on Computational Methods in Systems Biology (CMSB 2005) (G. Plotkin, ed.), 2005, pp. 204–215.

[22] L. Cardelli, E. Caron, P. Gardner, O. Kahramanoğulları and A. Phillips,

A process model of Rho GTP-binding proteins, Theor. Comput. Sci. **410** (2009), 3166–3185.

[23] L. Cardelli and A. D. Gordon, *Mobile ambients*, Proc. First International Conference on Foundations of Software Science and Computation Structure (FoSSaCS 98) (M. Nivat, ed.), 1998.

[24] L. Cardelli and R. Mardare, *The measurable space of stochastic processes*, Proc. International Conference on Quantitative Evaluation of Systems (QEST 2010), 2010, to appear.

[25] L. Cardelli and G. Zavattaro, *Turing universality of the biochemical ground form*, Math. Struct. Comp. Sci. **20** (2010), 45–73.

[26] S. B. Carrol, *Endless forms most beautiful: The new science of evo devo and the making of the animal kingdom*, W. W. Norton and Company, 2005.

[27] F. Ciocchetta and J. Hillston, *Bio-PEPA: a framework for modelling and analysis of biochemical networks*, Theor. Comput. Sci. **410** (2009), 3065–3084.

[28] E. M. Clarke, O. Grumberg and D. A. Peled, *Model checking*, MIT Press, 1999.

[29] J. F. Crow and M. Kimura, *An introduction to population genetics theory*, Harper and Row, 1970.

[30] V. Danos and C. Laneve, *Formal molecular biology*, Theor. Comput. Sci. **325** (2004), 69–110.

[31] E. H. Davidson and D. H. Erwin, *Gene regulatory networks and the evolution of animal body plans*, Science **311** (2006), 796–800.

[32] N. G. de Bruijn, *Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church–Rosser theorem*, Indag. Math **34** (1972), 381–392.

[33] L. Dematté, C. Priami and A. Romanel, *The BlenX language: A tutorial*, in Bernardo et al. [9], pp. 313–365.

[34] L. Dematté, C. Priami, A. Romanel and O. Soyer, *A formal and intgrated framework to simulate evolution of biological pathways*, in Calder and Gilmore [19], pp. 106–120.

[35] ———, *Evolving BlenX programs to simulate the evolution of biological networks*, Theor. Comput. Sci. **408** (2008), no. 1, 83–96.

[36] J. L. Ditty, S. R. Mackey and C. H. Johnson (eds.), *Bacterial circadian programs*, Springer, 2009.

[37] J. W. Eaton, *GNU Octave manual*, Network Theory, 2002.

[38] A. Edalat and D. Pattinson, *A domain-theoretic account of Picard's theorem*, Proc. 31st International Colloquium on Automata, Languages and Programming (ICALP 2004) (J. Díaz et al., eds.), Lect. Notes Comp. Sci., vol. 3412, 2004, pp. 494–505.

[39] ———, *Domain theoretic solutions of initial value problems for unbounded vector fields*, Proc. 21st Annual Conference on Mathematical Foundations of Programming Semantics (MFPS XXI), Electronic Notes in Theoretical Computer Science, vol. 155, 2006, pp. 565–581.

[40] Edinburgh Compute and Data Facility (ECDF), `http://www.ecdf.ed.ac.uk`.

[41] S. Eker, M. Knapp, K. Laderoute, P. Lincoln, J. Meseguer and K. Sonmez, *Pathway Logic: Symbolic analysis of biological signaling*, Proc. Pacific Symposium on Biocomputing, 2002, pp. 400–412.

[42] A. Eyre-Walker and P. D. Keightley, *The distribution of fitness effects of new mutations*, Nat. Rev. Genet. **8** (2007), 610–618.

[43] F. Fages, G. Batt, A. Rizk and S. Soliman, *On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology*, in Heiner and Uhrmacher [66], pp. 251–268.

[44] F. Fages and A. Rizk, *On the analysis of numerical data time series in temporal logic*, in Calder and Gilmore [19], pp. 48–63.

[45] J. Feret, V. Danos, J. Krivine, R. Harmer and W. Fontana, *Internal coarse-graining of molecular systems*, P. Natl. Acad. Sci. USA **106** (2009), 6453–6458.

[46] E. Ferrada and A. Wagner, *Protein robustness promotes evolutionary innovations on large evolutionary time-scales*, P. Roy. Soc. B **275** (2008), 1595–1602.

[47] J. Fisher and D. Harel, *On statecharts for biology*, Symbolic Systems Bi-

ology: Theory and Methods (M. Sriram Iyengar, ed.), Jones & Bartlett, 2010.

[48] W. Fokkink, *Introduction to process algebra*, Texts in Theoretical Computer Science: An EATCS Series, Springer, 1999, also available from `http://www.cs.vu.nl/~wanf/books.html`.

[49] W. Fontana, *Modelling 'evo-devo' with RNA*, BioEssays **24** (2002), 1164–1177.

[50] W. Fontana and P. Schuster, *Continuity in evolution: On the nature of transitions*, Science **280** (1998), 1451–1455.

[51] ———, *Shaping space: the possible and the attainable in RNA genotype-phenotype mapping*, J. Theor. Biol. **194** (1998), 491–515.

[52] J. Gerhart and M. Kirschner, *Evolvability*, P. Natl. Acad. Sci. USA **95** (1998), 8420–8427.

[53] ———, *The plausibility of life: Resolving Darwin's dilemma*, Yale University Press, 2005.

[54] ———, *The theory of facilitated variation*, P. Natl. Acad. Sci. USA **104** (2007), 8582–8589.

[55] M. A. Gibson and J. Bruck, *Efficient exact stochastic simulation of chemical systems with many species and many channels*, J. Phys. Chem. A **104** (2000), 1876–1889.

[56] D. T. Gillespie, *Exact stochastic simulation of coupled chemical reactions*, J. Phys. Chem. **81** (1977), 2340–2361.

[57] ———, *The chemical Langevin equation*, J. Chem. Phys. **113** (2000), 297–306.

[58] S. S. Golden, C. H. Johnson and T. Kondo, *The cyanobacterial circadian system: A clock apart*, Curr. Opin. Microbiol. **1** (1998), no. 6, 669–673.

[59] P. J. E. Goss and J. Peccoud, *Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets*, P. Natl. Acad. Sci. USA **95** (1998), 6750–6755.

[60] N. Grobbelaar, T. Huang, H. Lin and T. Chow, *Dinitrogen-fixing endogenous rhythm in Synechococcus RF-1*, FEMS Microbiol. Lett. **37** (1986), 173–177.

[61] D. L. Hartl and A. G. Clark, *Principles of population genetics*, Sinauer Associates, 1997.

[62] Haskell, `http://www.haskell.org/`.

[63] J. Heath, M. Kwiatkowska, G. Norman, D. Parker and O. Tymchyshyn, *Probabilistic model checking of complex biological pathways*, Theor. Comput. Sci. **391** (2007), 239–257.

[64] M. Heiner, D. Gilbert and R. Donaldson, *Petri nets for systems and synthetic biology*, in Bernardo et al. [9], pp. 215–264.

[65] M. Heiner and I. Koch, *Petri net based model validation in systems biology*, Proc. 25th International Conference on Applications and Theory of Petri Nets (ICATPN 2004) (J. Cortadella and W. Reisig, eds.), Lect. Notes Comp. Sci., vol. 3099, 2004, pp. 216–237.

[66] M. Heiner and A. Uhrmacher (eds.), *Proc. Sixth International Conference on Computational Methods in Systems Biology (CMSB 2008)*, Lect. N. Bioinformat., vol. 5307, 2008.

[67] J. Hillston, *A compositional approach to performance modelling*, Cambridge University Press, 1996.

[68] C. A. R. Hoare, *Communicating sequential processes*, Commun. ACM **21** (1978), 666–677.

[69] J. Hofbauer and K. Sigmund, *Evolutionary games and population dynamics*, Cambridge University Press, 1998.

[70] C. Y. Huang and J. E. Ferrell, *Ultrasensitivity in the mitogen-activated protein kinase cascade*, P. Natl. Acad. Sci. USA **93** (1996), 10078–10083.

[71] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano et al., *The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models*, Bioinformatics **19** (2003), 524–531.

[72] T. Ideker, T. Galitski and L. Hood, *A new approach to decoding life: Systems biology*, Annu. Rev. Genomics Hum. Genet. **2** (2001), 343–372.

[73] M. Ishiura, S. Kutsuna, S. Aoki, H. Iwasaki, C. R. Andersson, A. Tanabe, S. S. Golden, C. H. Johnson and T. Kondo, *Expression of a gene clus-*

ter kaiABC as a circadian feedback process in cyanobacteria, Science **281** (1998), no. 5382, 1519–1523.

[74] M. John, C. Lhoussaine, J. Niehren and A. M. Uhrmacher, *The attributed pi-calculus*, in Heiner and Uhrmacher [66], pp. 83–102.

[75] M. John, R. Ewald and A. M. Uhrmacher, *A spatial extension to the pi calculus*, Electronic Notes in Theoretical Computer Science **194** (2008), 133–148.

[76] D. S. Jones and B. D. Sleeman, *Differential equations and mathematical biology*, Chapman & Hall, 2003.

[77] J. G. Kemeny and J. L. Snell, *Finite Markov chains*, Springer, 1976.

[78] M. Kimura, *The neutral theory of molecular evolution*, Cambridge University Press, 1983.

[79] H. Kitano, *Systems biology: Towards system-level understanding of biological systems*, Foundations of Systems Biology, MIT Press, 2001, pp. 1–36.

[80] ———, *Computational systems biology*, Nature **420** (2002), 206–210.

[81] ———, *Biological robustness*, Nat. Rev. Genet. **5** (2004), 826–837.

[82] E. Klipp, R. Herwig, A. Kowald, C. Wierling and H. Lehrach, *Systems biology in practice: Concepts, implementation and application*, Wiley-VCH, 2005.

[83] K. W. Kohn, M. I. Aladjem, J. N. Weinstein and Y. Pommier, *Molecular interaction maps of bioregulatory networks: a general rubric for system biology*, Mol. Biol. Cell **17** (2006), 1–13.

[84] K. W. Kohn, *Molecular interaction map of the mammalian cell cycle control and DNA repair systems*, Mol. Biol. Cell **10** (1999), 2703–2734.

[85] ———, *Molecular interaction maps as information organizers and simulation guides*, Chaos **11** (2001), 84–97.

[86] C. Kuttler, *Modelling bacterial gene expression in a stochastic pi-calculus with concurrent objects*, Ph.D. thesis, University of Lille 1, 2007.

[87] M. Kwiatkowska, G. Norman and D. Parker, *Stochastic model checking*, Proc. 7th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation

(SFM'07) (M. Bernardo and J. Hillston, eds.), Lect. Notes Comp. Sci., vol. 4486, 2007, pp. 220–270.

[88] M. Kwiatkowski and I. Stark, *The continuous π-calculus: A process algebra for biochemical modelling*, in Heiner and Uhrmacher [66], pp. 103–122.

[89] N. Le Novère et al., *The systems biology graphical notation*, Nat. Biotechnol. **27** (2009), 735–741.

[90] P. Lecca, C. Priami, P. Quaglia, B. Rossi, C. Laudanna and G. Constantin, *A stochastic process algebra approach to simulation of autoreactive lymphocyte recruitment*, Simulation **80** (2004), 273–288.

[91] H. Li, Y. Cao, L. R. Petzold and D. T. Gillespie, *Algorithms and software for stochastic simulation of biochemical reacting systems*, Biotechnol. Prog. **24** (2008), 56–62.

[92] B. Luttik and V. van Oostrom, *Decomposition orders—another generalisation of the fundamental theorem of arithmetic*, Theor. Comput. Sci. **335** (2005), 147–186.

[93] M. Lynch, *The origins of genome architecture*, first ed., Sinauer Associates, 2007.

[94] M.-A. Félix and A. Wagner, *Robustness and evolution: concepts, insights and challenges from a developmental model system*, Heredity **100** (2008), 132–140.

[95] J. Maynard Smith, *Evolution and the theory of games*, Cambridge University Press, 1982.

[96] R. Meyer, V. Khomenko and T. Strazny, *A practical approach to verification of mobile systems using net unfoldings*, Fundam. Inform. **94** (2009), 439–471.

[97] R. Milner, *A calculus of communicating systems*, Springer-Verlag, 1980.

[98] ———, *The polyadic π-calculus: A tutorial*, Tech. Report ECS-LFCS-91-180, LFCS, University of Edinburgh, 1991.

[99] ———, *Communicating and mobile systems: The π calculus*, Cambridge University Press, 1999.

[100] R. Milner and F. Moller, *Unique decomposition of processes*, Theor. Comput. Sci. **107** (1993), 357–363.

[101] R. Montville, R. Froissart, S. K. Remold, O. Tenaillon and P. E. Turner, *Evolution of mutational robustness in an RNA virus*, PLoS Biol. **3** (2005).

[102] G. B. Müller, *Evo-devo: extending the evolutionary synthesis*, Nat. Rev. Genet. **8** (2007), 949–949.

[103] T. Murata, *Petri nets: properties, analysis and applications*, P. IEEE **77** (1989), 541–580.

[104] M. Nakajima, T. Kondo, J. Tomita and H. Iwasaki, *No transcription-translation feedback in circadian rhythm of kaic phosphorylation*, Science **307** (2005), 251–254.

[105] M. Nakajima, K. Imai, H. Ito, T. Nishiwaki, Y. Murayama, H. Iwasaki, T. Oyama and T. Kondo, *Reconstitution of circadian oscillation of cyanobacterial KaiC phosphorylation in vitro*, Science **308** (2005), 414–415.

[106] M. Nei, *Selectionism and neutralism in molecular evolution*, Mol. Biol. Evol. **22** (2005), 2318–2342.

[107] M. H. A. Newman, *On theories with a combinatorial definition of "equivalence"*, Annals of Math. **43** (1942), 223–243.

[108] Octave, `http://www.gnu.org/software/octave/`.

[109] B. Oksendal, *Stochastic differential equations: An introduction with applications*, Springer, 2003.

[110] R. J. Orton, O. E. Sturm, V. Vyshemirsky, M. Calder, D. R. Gilbert and W. Koch, *Computational modelling of the receptor-tyrosine-kinase-activated MAPK pathway*, Biochem. J. **392** (2005), 249–261.

[111] J. Parrow, *An introduction to the π-calculus*, Handbook of Process Algebra, Elsevier, 2001, pp. 479–543.

[112] L. Partridge and N. H. Barton, *Natural selection: Evolving evolvability*, Nature **407** (2000), 457–458.

[113] M. Pedersen, *Modular languages for systems and synthetic biology*, Ph.D. thesis, The University of Edinburgh, 2010.

[114] M. Pedersen and G. Plotkin, *A language for biochemical systems: Design and formal specification*, Transactions on Computational Systems Biology **XII** (2010), 77–145.

[115] B. C. Pierce, *Types and programming languages*, MIT Press, 2002.

[116] M. Pigliucci, *Phenotypic plasticity: Beyond nature and nurture*, Syntheses in Ecology and Evolution, Johns Hopkins University Press, 2001.

[117] ———, *Genotype–phenotype mapping and the end of the 'genes as blueprint' metaphor*, Philos. T. Roy. Soc. B **365** (2010), 557–566.

[118] G. D. Plotkin, *The origins of structural operational semantics*, J. Logic Algebr. Progr. **60–61** (2004), 3–15.

[119] ———, *A structural approach to operational semantics*, J. Logic Algebr. Progr. **60–61** (2004), 17–139, Originally published as DAIMI FN-19, Aarhus University, 1981.

[120] J. Podani, Z. N. Oltvai, H. Jeong, B. Tombor, A.-L. Barabási and E. Szathmáry, *Comparable system-level organization of Archaea and Eukaryotes*, Nat. Genet. **29** (2001), 54–56.

[121] C. Priami, *Stochastic π-calculus*, Comput. J. **38** (1995), 578–589.

[122] C. Priami and P. Quaglia, *Beta binders for biological interactions*, Proc. Second International Conference on Computational Methods in Systems Biology (CMSB 2004) (V. Danos and V. Schächter, eds.), Lect. Notes Comp. Sci., vol. 3082, 2005, pp. 20–33.

[123] C. Priami, A. Regev, E. Shapiro and W. Silverman, *Application of a stochastic name-passing calculus to representation and simulation of molecular processes*, Inform. Process. Lett. **80** (2001), 25–31.

[124] PRISM Model Checker, `http://www.prismmodelchecker.org`.

[125] M. Rathinam, L. R. Petzold, Y. Chao and D. T. Gillespie, *Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method*, J. Chem. Phys. **119** (2003), 12784–12794.

[126] A. Regev, *Computational systems biology: A calculus for biochemical knowledge*, Ph.D. thesis, Tel Aviv University, 2002.

[127] A. Regev, E. M. Panina, W. Silverman, L. Cardelli and E. Shapiro, *Bioambients: An abstraction for biological compartments*, Theor. Comput. Sci. **325** (2004), 141–167.

[128] A. Regev and E. Shapiro, *Cellular abstractions: Cells as computations*, Nature **419** (2002), 343.

[129] A. Regev, W. Silverman and E. Shapiro, *Representation and simulation of biochemical processes using the pi-calculus process algebra*, Proc. Pacific Symposium on Biocomputing, 2001, pp. 459–470.

[130] A. Romanel, *Dynamic biological modelling: a language-based approach*, Ph.D. thesis, University of Trento, 2010.

[131] R. Sanjuán, J. M. Cuevas, V. Furió, E. C. Holmes and A. Moya, *Selection for robustness in mutagenized RNA viruses*, PLoS Genet. **3** (2007).

[132] E. A. Schultes and D. P. Bartel, *One sequence, two ribozymes: implications for the emergence of new ribozyme folds*, Science **289** (2000), 448–452.

[133] P. Schuster, W. Fontana, P. F. Stadler and I. L. Hofacker, *From sequences to shapes and back: A case-study in RNA secondary structures*, P. Roy. Soc. B **255** (1994), 279–284.

[134] M. R. Shinwell, A. M. Pitts and M. J. Gabbay, *FreshML: Programming with binders made simple*, Eighth ACM SIGPLAN International Conference on Functional Programming (ICFP 2003), 2003, pp. 263–274.

[135] M. L. Siegal and A. Bergman, *Canalization*, Evolutionary Genetics: Concepts and Case Studies (C. W. Fox and J. B. Wolf, eds.), Oxford University Press, 2006.

[136] O. Soyer, M. Salathé and S. Bonhoeffer, *Signal transduction networks: Topology, response, and biochemical reactions*, J. Theor. Biol. **238** (2006), 416–425.

[137] O. S. Soyer and S. Bonhoeffer, *Evolution of complexity in signaling pathways*, P. Natl. Acad. Sci. USA **103** (2006), 16337–16342.

[138] O. S. Soyer, T. Pfeiffer and S. Bonhoeffer, *Simulating the evolution of signal transduction pathways*, J. Theor. Biol. **241** (2006), 223–232.

[139] SpiCO, `http://spico.gforge.inria.fr/`.

[140] B. M. R. Stadler, P. F. Stadler, G. Wagner and W. Fontana, *The topology of the possible: Formal spaces underlying patterns of evolutionary change*, J. Theor. Biol. **213** (2001), 241–274.

[141] A. Stefanek, *Continuous and spatial extension of the stochastic pi-calculus*, Master's thesis, Imperial College, London, UK, 2009.

[142] J. Stelling, U. Sauer, Z. Sallasi, F. J. Doyle III and J. Doyle, *Robustness of cellular functions*, Cell **118** (2004), 675–685.

[143] Stochastic Pi Machine (SPiM), `http://research.microsoft.com/en-us/projects/spim/default.aspx`.

[144] The Bio-PEPA Workbench, `http://www.dcs.ed.ac.uk/home/stg/software/biopepa/`.

[145] The PEPA Plugin Project, `http://www.dcs.ed.ac.uk/pepa/tools/plugin`.

[146] The Systems Biology Graphical Notation (SBGN), `http://sbgn.org`.

[147] The Systems Biology Markup Language (SBML), `http://sbml.org`.

[148] J. S. van Zon, D. K. Lubensky, P. R. H. Altena and P. R. ten Wolde, *An allosteric model of circadian KaiC phosphorylation*, P. Natl. Acad. Sci. USA **104** (2007), 7420–7425.

[149] C. Versari, *A core calculus for a comparative analysis of bio-inspired calculi*, Proc. 16th European Symposium on Programming (ESOP 2007) (R. de Nicola, ed.), Lect. Notes Comp. Sci., vol. 4421, 2007, pp. 411–425.

[150] C. Versari and N. Busi, *Stochastic simulation of biological systems with dynamical compartment structure*, in Calder and Gilmore [19], pp. 80–95.

[151] A. Wagner, *Does evolutionary plasticity evolve?*, Evolution **50** (1996), 1008–1023.

[152] ———, *Robustness and evolvability in living systems*, Princeton University Press, 2005.

[153] ———, *Robustness and evolvability: a paradox resolved*, P. Roy. Soc. B **275** (2008), 91–100.

[154] J. Zhang, *Evolution by gene duplication: an update*, Trends in Ecology and Evolution **18** (2003), 292–298.

[155] M. Zuker and D. Sankoff, *RNA secondary structures and their prediction*, B. Math. Biol. **46** (1984), 591–621.