University of Insubria
Department of Theoretical and Applied Sciences (DiSTA)
Ph.D. in Computer Science and Computational Mathematics

Formal Models for Biological Systems

Supervisor: **Prof. Simone Tini**
Co-supervisor: **Prof. Ruggero Lanotte**
Co-supervisor: **Prof. Nicoletta Sabadini**

Doctoral Dissertation of
Desiree Manicardi
ID: 721203
XXXVI cycle

Abstract

In the last thirty years, formal models have been thoroughly employed in the realm of biological systems for many reasons: (i) preventing those ambiguities that may arise when informal notations are used for system description, (ii) supporting the development of simulators, (iii) supporting the development of tools, such as model checkers, allowing for verifying whether a system satisfies a given behavioural property, (iv) offering several instruments allowing for comparing the behaviour of different systems.

The work in this thesis can be divided into two contributions concerning formal models for biological systems.

The first contribution is related to the study of the robustness of biochemical networks. In particular, we take inspiration from the notion of *α-robustness* [1], which, intuitively, verifies how by varying the *initial* concentration of some species, called conventionally the *input species*, the concentration of other species of interest, called the *output species*, varies at *steady state*. Robustness in our sense captures random effects and temporary effects that are typical of the stochastic model.

We will employ: (i) the process calculi [2] approach for specifying systems of interest, (ii) the semantic model of *evolution sequences* [3, 4], which, intuitively, models the behaviour of a system as the sequence of probability measures over the attainable configurations, (iii) a formal notion of robustness, defined on the semantic model, and (iv) an algorithm allowing us to estimate the robustness of a system starting from its specification.

We validate our approach on three case studies, already considered in [1]: *EnvZ/OmpR Osmoregulatory Signaling System* in Escherichia Coli, which is an example of the regulatory network, the mechanism of *Bacterial Chemotaxis* of Escherichia Coli, and an abstract chemical reaction network, called *Enzyme Activity at Saturation*.

We have provided a Python implementation available at `https://github.com/dmanicardi/spebnr`, and described in Appendix A.

Our second contribution is showing how the features of CospanSpan(Graph) [5, 6] can be exploited in modelling biological systems. CospanSpan(Graph) offers an algebraic approach for the compositional description of variable topology networks that has been only partially exploited so far for the formalisation of that kind of systems.

In particular, we provide a simplified model of a human heart [7] and a model of a dual-chamber pacemaker [7] that can interact with the model of the heart. Then, we model a gene regulatory network, namely the *Lac Operon* of Escherichia Coli [8, 9, 10].

To my family members with the greatest thirst for knowledge
(my grandmom Lina, my dad Nelson and my cousin Alessandro),
and to all people who were, are and will be unable to study for economic reasons.

Acknowledgments

This thesis marks the end of my PhD in "Computer Science and Computational Mathematics".

First of all, I want to dedicate my heartfelt recognition to my supervisor and my two co-supervisors for their help. I would like to further extend my gratitude to my supervisor, Prof. Simone Tini; the spirit guide who has assisted and guided me throughout those years, not only in academics but also in psychological matters. I would also like to thank Prof. Ruggero Lanotte; for his precious and appreciated collaboration. I would like to thank the omnipresent Prof. Nicoletta Sabadini as well; who has helped me think differently and break the mould.

I would like to offer my special thanks to the reviewers Dr. Valentina Castiglioni and Prof. Paolo Milazzo; for their insightful comments and suggestions which have been precious for the review of this thesis.

I wish to extend my special thanks to every Professor, researcher and PhD student with whom I have had the opportunity to interact, not only academically but also spiritually. Their support helped me to not feel out of this academic world. I particularly thank the coordinator Prof. Barbara Carminati, Dr. Alessandra Rizzardi, Prof. Luigi Lavazza and Prof. Davide Tosi.

My gratitude goes also to all those who have believed in me outside the university walls and who have encouraged me to achieve more ambitious results.

I am deeply grateful to my parents, Emanuela and Nelson, valid collaborators throughout my career as a student and beyond. They have always accompanied me with love, sacrifices and a lot of patience. Words are not enough to thank them.

Last but not least, I would like to express my sincere gratitude to my British cousin, Davide, who always supports me and pushes me every day to be a better person. Particularly, in previous years, he was fundamental at helping me improve my English.

Thank you, Grazie, Teşekkür ederim, Dziękuję

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

In the last thirty years, formal models have been thoroughly employed in the realm of biological systems. The first reason is that formal models prevent those ambiguities that may arise when informal notations are used for system description. Then, formal modelling supports the development of simulators, that allow for understanding in silico how a system behaves, both in normal conditions and under the effect of perturbations that may alter its nominal functioning. Another important aspect is that formal models support the development of tools, such as model checkers, allowing for verifying whether a system satisfies a given behavioural property. Finally, formal models offer several instruments allowing for comparing the behaviour of different systems, which supports the ability to check whether two different versions of a system behave in the same way when some aspects of system behaviour are abstracted away.

Notably, those formal models that support compositional reasoning, have found applications in systems biology [11], where it is required that the functionality and the behaviour of a complex system are obtained from the interactions between its components. A typical example of a complex system, whose morphological and functional organization have been thoroughly investigated in the context of system biology, are living cells: the complex behaviour of cells arises from the interactions of their huge amount of components, that interact with each other, through chemical reaction networks. The importance of these interactions is witnessed also by the fact that their malfunctioning, or corruption may originate in severe diseases, such as cancer or diabetes.

The research in formal models applied to biological systems presents at least two directions. One is to focus on some relevant aspects or interesting properties of a class of systems and to study them by employing formalisms that are already used in the realm

of biological systems. The other is to focus on models that have been used, so far, only in different contexts, and discover that they offer some notions, concepts, and mechanisms that help to model biological systems. Below, we discuss the property of robustness and the model CospanSpan(Graph) as examples of, respectively, an interesting property deserving investigation, and a formalism offering relevant features not fully exploited, so far, in the modelling of biological systems.

Robustness as a relevant property of biological systems

In the case of living cells, one of the reasons for which it is relevant to study how the components of the cells interact with each other as a system is the necessity of being able to predict how perturbations can affect these interactions thus modifying cell behaviour. In particular, in some cases, it is interesting to predict in which case the nominal behaviour of the cell system can be maintained in the presence of those perturbations, which leads to the notion of robustness.

The notion of robustness has been widely used in several contexts, from control theory [12] to biology [13]. On one side, we observe no formal agreement on any "official" definition of robustness. For instance, according to the paper [14] published in 2013, there were more than 600 papers listed in the ACM Digital Library that propose an "original approach" to software robustness. On the other side, the intuitive meaning of this concept is shared in several areas: robustness is commonly meant as the ability of a system to maintain its functionalities against uncertainty and perturbations. Having adopted this intuitive meaning, the paper [15] argued that, in nature, several mechanisms support robustness against environmental and genetic perturbations, thus allowing complex systems to be evolvable:

(i) system control, which consists of negative and positive feedback to attain a robust dynamic response observed in a wide range of regulatory networks;

(ii) redundancy and diversity, which consists of having multiple means to achieve a specific function, because failure of one of them can be overcome by others;

(iii) modularity, which is an effective mechanism for containing perturbations and damage within subsystems in order to minimise the effects on the whole system;

(iv) decoupling, which consists of isolating low-level variation from high-level functionalities.

Clearly, robustness is more a quantitative than a qualitative property: saying that a system is robust, or not, is often uninformative, the point is that of quantifying both, the extent of the perturbations, and the impact that they have on system behaviour, then the value of robustness is a function of those two quantities.

The CospanSpan(Graph) model

The compositional model CospanSpan(Graph) [5, 6] has been shown to model a variety of phenomena, from asynchronous circuits to hierarchy, mobility and coordination. The

elements of the model are cospans and spans of graphs which here we shall call "Automata with interfaces". These automata are an extension of the classical finite state automata, that were introduced in the seminal work of McCulloch and Pitts [16] as a discrete model for threshold neurons and neural networks. Automata, since then, have become the standard model for the specification and verification of sequential discrete dynamical systems. Later, we have been assisting a paradigmatic shift from sequential systems, exemplified by Turing Machines, to networks of parallel, interacting components.

Hence, various models of automata with a product of states have been proposed to represent interactions (Zielonka [17], Petri [18]). These models are rather natural, but unfortunately are not compositional, that is they lack a proper algebra. On the contrary, compositionality is an essential feature of CospanSpan(Graph). This model offers explicit operations that combine automata with interfaces and their connectors. Hence, given a syntactic expression, its global semantics can be deduced only by the semantics of its constituents, and this is precisely what the algebraic approach guarantees. In order to fully achieve compositionality, also with respect to parallelism, CospanSpan(Graph) abandons both the classical (inherently sequential and closed) model of finite state automata and the well-established idea of input/output communication for a new paradigm, considering as fundamental the notion of open systems with communication interfaces. The operations in the algebra CospanSpan(Graph) can be interpreted in a very natural way as operations on automata with states and transitions, as well as interfaces and conditions. An expression (or even a recursive equation) in this algebra represents a hierarchical, reconfigurable network of interacting components.

From the beginning, it has been very natural to consider automata theory and biology as very close disciplines, with a long and very fruitful tradition of reciprocal influences, from robotics to biology-inspired models of computation. Unfortunately, whereas we could be reasonably satisfied with Turing Machines (and finite state automata) when dealing with discrete, isolated and sequential computation devices, in the literature there is a lack of a general model for compositional biology-inspired automata networks that could play an analogous role. This is crucial for performing verification tasks as well.

An interesting point is to focus on the importance of an algebraic approach for the compositional description of variable topology networks that leads to a natural formalisation of biological systems. In other words, investigating the expressivity of the CospanSpan model in the modelling of biological systems.

1.2 Our contribution

The work in this thesis can be divided into two contributions concerning formal models for biological systems, motivated by the discussion in the previous section.

The first contribution is related to the study of the robustness of biochemical networks. In particular, we take inspiration from the notion of α-*robustness* [1], which, intuitively, verifies how by varying the *initial* concentration of some species, called conventionally the *input species*, the concentration of other species of interest, called the *output species*, varies at *steady state*. Our contribution has two main differences with

respect to [1]:

(i) we work within the stochastic model [19], whereas α-robustness has been proposed for the deterministic model [20];

(ii) coherently with the choice of the model, instead of evaluating the concentration level of output species at steady state, we evaluate it step-by-step, up to a finite horizon.

Since the deterministic and the stochastic approach are complementary [20], we can argue that also our results and those in [1] are. Clearly, robustness in our sense captures random effects and temporary effects that are typical of the stochastic model.

We will employ:

(i) the process calculi approach [2] for specifying systems of interest,

(ii) the semantic model of *evolution sequences* [3, 4], which, intuitively, model the behaviour of a system as the sequence of probability measures over the attainable states,

(iii) a formal notion of robustness, defined on the semantic model, and

(iv) an algorithm allowing us to estimate the robustness of a system starting from its specification.

We validate our approach on three case studies, already considered in [1]: *EnvZ/OmpR Osmoregulatory Signaling System* in Escherichia Coli, which is an example of regulatory network, the mechanism of *Bacterial Chemotaxis* of Escherichia Coli, and an abstract chemical reaction network, called *Enzyme Activity at Saturation* in [1].

We have provided a Python implementation available at `https://github.com/dmanicardi/spebnr`. This is the tool SPEBNR, *a Simple Python Environment for statistical estimation of Biochemical Network Robustness*. Details on how to use SPEBNR are presented in Appendix A.

Our second contribution consists in showing how the features of CospanSpan(Graph) can be exploited in modelling biological systems. The aim is to fill the gap mentioned in the previous section: the lack of a general model for compositional (biology-inspired) automata networks that could play a role analogous to finite state automata in the context of discrete, isolated and sequential computation devices.

In particular, we provide a simplified model of a human heart [7] and a model of a dual-chamber pacemaker [7] that can interact with the model of the heart. Then, we model a gene regulatory network, namely the *Lac Operon* of Escherichia Coli [8, 9, 10].

1.3 Structure of the Thesis

The thesis is organised into five chapters, plus one introduction and a conclusion. The thesis consists of two main contributions concerning formal models for biological systems.

The former is related to the study of the robustness of biochemical networks, which is exposed in Chapters 3 and 4. The latter concerns the investigation of CospanSpan(Graph), exposed in Chapters 5 and 6. In detail:

- Chapter 2 offers an overview of classical formalisms for modelling biological systems;
- Chapter 3 shows how the process algebra approach for modelling biological systems well supports the study of a notion of *robustness* for biochemical networks, whose dynamics are formalised by adopting Gillespie's *stochastic model* [19]. As regards specification, we rely on *process algebras*, adopting in particular the *species as processes* [2] approach. In order to model the behaviour of a biological system, we follow Gillespie's approach [19], where computation steps represent single chemical reactions. Here, at each step there is a competition between all available reactions, giving rise to a probabilistic behaviour. In this context, we believe that it is convenient to adopt the semantic model of *evolution sequences* proposed in [3, 4]. Essentially, an evolution sequence is a sequence of probability measures over the attainable states of the system. In our case, by the state of the system, we mean the concentration level of all species. We define the *input distance* and the *output distance* between the original system and the system subject to variation in input species. Then we formalise a notion of *robustness* whose intuition is that small variations in input distance should give rise to smooth and limited variations in output distance;
- Chapter 4 applies the theory presented in Chapter 3 to study some robustness properties of three different systems that were already analysed in [1], within the deterministic model. Experiments are conducted in SPEBNR. In Section 4.1 we consider an example of a regulatory network, namely the *EnvZ/OmpR Osmoregulatory Signaling System* in Escherichia Coli. Then, in Section 4.2 we study the mechanism of *Bacterial Chemotaxis* of Escherichia Coli. Finally, in Section 4.3 we deal with an abstract chemical reaction network, called *Enzyme Activity at Saturation* in [1];
- Chapter 5 presents the algebra $\mathbf{Span}(\mathbf{C})$ and its dual counterpart $\mathbf{Cospan}(\mathbf{C})$, as introduced by Benabou in [21]. CospanSpan(Graph) is a categorical model proposed by Katis-Sabadini-Walters [22, 23] for the compositional description of reconfigurable hierarchical networks. In particular, we focus on *compositionality*, and on the possibility of describing the interactions among physical/biological systems. We also describe the *timed* version of this algebra;
- Chapter 6 applies the theory presented in Chapter 5 to describe compositionally three biological systems. In Section 6.1 we consider an example of the *Heart System*. Then, in Section 6.2 we study the *Pacemaker Dual Chamber DDD System*. Finally, in Section 6.3 we deal with the *Lac Operon System*. Moreover, we discuss the robustness of CospanSpan(Graph);
- Chapter 7 presents the conclusion of the thesis and describes possible future works.

At the end of the thesis, the Appendix A describes the procedure needed for installing and using the tool SPEBNR, *a Simple Python Environment for statistical estimation of Biochemical Network Robustness*, introduced in Chapter 3.

1.4 Published material

Part of the material presented in this thesis has appeared in some papers. In particular:

- The notion of robustness and the case study *EnvZ/OmpR Osmoregulatory Signalling* System, presented in Chapter 3 and in Section 4.1, appeared in [24].
- The work and the results presented in Chapter 4 – Section 4.2 and Section 4.3 – will be submitted in extend form with the title: "A framework for analysis of robustness in biochemical networks." (2024).
- The investigation of CospanSpan(Graph), presented in Chapter 5 and Chapter 6, has appeared in [25, 26, 27].
- The work presented in Chapter 5 – Section 6.4 – will be submitted in form with the title: "Robustness in CospanSpan(Graph)" (2024).

Part of the material is also been presented at conferences. In particular:

- R. Lanotte, D. Manicardi and S. Tini, "Step-by-step Robustness for Biochemical Networks", Italian Conference on Theoretical Computer Science 2023 (Palermo, Italy, September 13-15, 2023) [Conference Presentation], presented in Chapter 3 and in Section 4.1.
- D. Manicardi and N. Sabadini, "Cospan/Span(Graph): an algebra for open, reconfigurable automata networks.", Workshop in honour of R. F. C. Walters (Tallinn, Estonia, July 17-18, 2023) [Conference Presentation], presented in Chapter 5 and Chapter 6.
- D. Manicardi, N. Sabadini and S. Tini, "Automata and intelligent agents", NeuroSpine, V International Meeting, (online, November 16, 2020) [Conference Presentation], presented in Chapter 5 and Chapter 6.
- A. Gianola, S. Kasangian, D. Manicardi and N. Sabadini, "A Compositional Model of the Heart-Pacemaker System in CospanSpan(Graph)", Workshop iHeart, 2019 RISM Congress: Modelling the Cardiac Function, (Varese, Italy, July 22-24, 2019) [Poster], presented in Chapter 5 and Chapter 6.

2

Classical Formal Models for Biological Systems

Classical mathematical approaches have been successfully employed for years for modelling biological systems. Without claiming to be exhaustive, and only in order to provide some examples, we mention that ordinary differential equations and stochastic processes allow for describing systems at the level of biochemical reactions [28], partial differential equations have been used for representing higher level behaviours where also diffusion of molecules has to be taken into account [29], cellular automata [30] have been employed for modelling higher-level structures such as tissues [31].

In order to model and explain biological phenomena, some models have been proposed ad hoc, others were directly inspired by biology. One of the most prominent examples is that of Lindenmayer Systems [32]. Essentially, a Lindenmayer System is a formal grammar in which production rules are applied in parallel and that is equipped with mechanisms allowing to mapping of the generated strings into "geometric" structures. Originally, Lindenmayer Systems were employed to model the growth process of plants. Membrane Systems [33, 34, 35] was originally introduced as a universal computational paradigm inspired by living cells. The main idea is to have different "computation locations" where some subtasks of a more complex task can be performed. These locations are organised into a hierarchical structure, and each location is a sort of compartment surrounded by a membrane. As in a living cell, the tasks are different membranes run in parallel and independently, but some pieces of computation can conditionally cross the membranes, some membranes may dissolve, others can be joined, and others can be split. Even if Membrane Systems are essentially a computation paradigm, they have been also employed for the modelling of biological systems, for instance for modelling signalling pathways [36]. Also, Reaction Systems [37] were introduced as a computational paradigm, which capture two fundamental interactions typically present between biochemical entities, namely activation and inhibition: computation consists in trans-

forming reactant objects into product objects provided that activators are present and inhibitors are absent. This formalism has been then applied for modelling biological phenomena [38] and properties of biological systems [39].

When the aim has been that of maintaining and sharing models, descriptive languages such as SBML and graphical formalisms such as Kitano Map [40] have played a major role.

Formal models from computer science have been employed in the realm of biological systems for several reasons, already discussed in the Introduction: (i) as the formalisms described above, they prevent those ambiguities that may arise when informal notations are used for system description, (ii) they support the development of simulators, (iii) they support the development of tools, such as model checkers, (iv) they offer several instruments allowing for comparing the behaviour of different systems.

The analogies that exist between concurrent agent interactions and biochemical reactions motivated the application to formalisation of biological systems of process algebras [41], which are a family of calculi originally developed to represent and analyse formally the behaviour of concurrent systems. In this context, one of the first proposals [42, 43] was to use the π-calculus for describing biomolecular processes underlying protein networks. The main motivation was to exploit existing tools for simulation, analysis and verification of systems. Later, this approach was extended in order to capture also quantitative aspects of behaviour [44]. In κ-calculus [45], protein interactions are idealised as a particular kind of graph rewriting, and two different types of rewriting rules model two types of biological reactions, that is complexation and decomplexation. Bio Ambients [46] is a calculus that allows for representing various aspects of molecular localisation and compartmentalisation, such as the movement of molecules between compartments, the dynamic rearrangement of cellular compartments, and the interaction between molecules in a compartmentalised setting. Brane Calculi [47] are a family of formalisms where the main idea is to have a structure of membranes which are main actors of interactions such as phagocytosis and exocytosis, then computation happens on membranes. Beta-binders [48] is a formalism where processes are encapsulated in boxes with some sites (beta-binders) expressing some interaction capabilities, such as the joining between two processes, the split of one process into two, the change of the process interface by hiding, unhiding and exposing a site. The calculus of looping sequences [49] allows for describing micro-biological systems and their evolution. In particular, this calculus interprets membranes as sequences of elements able to interact, which has been demonstrated to be useful for describing real biological phenomena. Bio-PEPA [50] was proposed for the formalisation of biochemical networks and focuses on the possibility of representing explicitly some quantitative features of biochemical models, such as stoichiometry and levels of concentration. In Bio-PEPA there are two different styles of modelling, namely reagent-centric view and pathway-centric view. In the first approach processes are species at a given level of concentration, in the second approach processes represent subpathways. Process algebra with hooks [51] is a formalism where processes represent biological entities that belong to different structural levels, such as molecules, cells and tissues. The calculus allows for modelling behaviours that happen on several

levels and for formalising how events in each level can affect the behaviour at other levels.

The ability to combine discrete and continuous behaviours offered by hybrid formalisms, such as hybrid automata, allows for modelling, at the same time, continuous evolutions such as those of biological systems and discrete interactions such as the scheduling of a medical treatment [52]. Hybrid Functional Petri Nets [53] extend Petri Nets with discrete and continuous places, transitions and arcs and allows for the definition and management of biological issues, particularly dynamic network structure changes considering discrete and continuous amounts. Moreover, this version of Petri Nets uses hierarchisation, which is beneficial in describing complex network structures.

3

Robustness in Biochemical Networks: a Process Algebra Approach

In this chapter, we show how the process algebra approach for modelling biological systems well supports the study of a notion of *robustness* for biochemical networks, whose dynamics are formalised by adopting Gillespie's *stochastic model* [19].

We propose a notion of robustness inspired by the notion of α-*robustness* introduced in [1] for the *deterministic model* [20] of the dynamic of biochemical networks. Intuitively, our notion of robustness quantifies how much the concentration of some species of interest, called the *output species*, varies step-by-step in consequence of the varying concentration of other species, called conventionally the *input species*. Essentially, our approach differs from that in [1] for two reasons:

(i) we work within the stochastic model, whereas α-robustness has been proposed in the deterministic case;

(ii) coherently with the choice of the model, instead of evaluating the concentration level of output species at *steady state* as in [1], we evaluate it step-by-step, up to a finite horizon.

Since the deterministic and the stochastic approach are complementary [20], we can argue that this holds also for our results and those in [1]. Clearly, robustness in our sense captures random effects and temporary effects that are typical of the stochastic model.

In order to study the robustness of the system's behaviour in the stochastic model, we need:

(i) a language allowing us to specify systems of interest;

(ii) a semantic model, which must capture the probabilistic behaviour that characterises the stochastic approach;

(iii) a formal notion of robustness, defined on the semantic model;

(iv) an algorithm allowing us to estimate the robustness of a system starting from its specification.

As regards specification, we rely on *process algebras*, adopting in particular the *species as processes* [2] approach. Essentially, a solution with n species is modelled by the parallel composition of n processes, where each process represents one species and its concentration level. We express the concentration level in terms of the number of molecules, but this can be generalized.

In order to model the behaviour of a biological system, we follow Gillespie's approach [19], where computation steps represent single chemical reactions. Here, at each step there is a competition between all available reactions, giving rise to a probabilistic behaviour. In this context, we believe that it is convenient to adopt the semantic model of *evolution sequences* proposed in [3, 4]: essentially, an evolution sequence is a sequence of probability measures over the attainable states of the system. In our case, by the state of the system, we mean the concentration level of all species.

By adopting evolution sequences, one can exploit the whole theory developed in [3, 4] to measure the *differences* between system behaviours, which supports the study of robustness properties. In [3, 4] the focus is on cyber-physical systems and the starting idea is to provide a notion of *distance* between computation states that quantifies to which extent they perform differently with respect to some targets that are fixed initially. In other words, to each computation state, one assigns a *penalty* quantifying how bad the system is working with respect to the targets, and the distance between two computation states is given by the difference of the penalties that are assigned to them. Then, the notion of distance between computation states is lifted first to probability measures over computation states and, then, to evolution sequences. In the present thesis, we customize this approach to support the study of a robustness property for biochemical networks inspired by the α-robustness of [1]. Essentially, since we are interested in studying the difference between the behaviour of two systems by focusing on the quantity of a given set of species, we can assign to each system state a *rank* that depends on the available quantity of these species. Then, the distance between two states coincides with the difference between their ranks, and this can be lifted to probability measures and evolution sequences. Since α-robustness based on how much the output species vary depending on the variation of the initial concentration of input species, we can consider two different ranks, which capture the input and the output species, respectively, and that allow us to define the *input distance* and the *output distance* between the original system and the system subject to variation on input species. Then we formalise a notion of robustness whose intuition is that small variations in input distance should give rise to smooth and limited variations in output distance.

Then, following [3, 4], we provide a randomized algorithm that permits estimating the evolution sequences of systems and thus for the evaluation of the distances between

them. This allows us to estimate the robustness of a nominal system, by sampling *perturbed* systems at a fixed maximal input distance from it and by estimating their output distance, whereby perturbed system we mean a system affected by a variation of the quantity of input species.

In order to validate our proposal, we have provided a Python implementation available at `https://github.com/dmanicardi/spebnr`. This is the tool SPEBNR, *a Simple Python Environment for statistical estimation of Biochemical Network Robustness*, which can be viewed as a customisation of the tool SPEAR (`https://github.com/quasylab/spear`) implementing the algorithms in [4]. Details on how to use SPEBNR are in Appendix A.

In Chapter 4, we will apply our theory to three case studies that were already analysed in [1] within the deterministic model. A comparison between our results and those in that paper will be provided.

3.1 Related Work on Robustness in Biology

Robustness in biology has been extensively studied in recent years. A very general approach has been proposed in [54, 55], where the nominal behaviour of a system subject to perturbations is expressed in terms of a formula specified with a *linear temporal logic* equipped with a *quantitative semantics*, then the robustness of the system is quantified as the average satisfaction degree of that property over all admissible perturbations, possibly weighted by their probabilities. Several notions of robustness proposed in the literature are less general and focus on steady-state behaviour. As an example, the notion of *adaptability* in [56] captures the idea that the behaviour at the steady state of a system is insensitive with respect to the initial concentration of some species. *Absolute concentration robustness* [57, 58] with respect to a given species requires that the system admits at least one positive steady state and that the concentration of that species is the same in all of the positive steady states that the system might admit. The notion of *α-robustness* has been proposed in [1] on the Continuous Petri Nets model. Intuitively, this notion captures the idea that by varying the initial concentration of input species under suitable constraints, namely by remaining within a so-called *interval marking* of the net, then at steady state one observes a bounded variation of the concentration of output species, namely that concentration is in a ball of radius α.

3.2 The Model

In this section, we propose our model for biochemical networks adopting the *species as processes* [2] approach.

We assume a set $\mathcal{N}$ of *names* for species and a set $\mathcal{R}$ of (bio)chemical *reactions*. Then, we use a set of *actions* $\mathcal{A}$ describing how a species can take part in a reaction. More precisely, if a is a reaction in $\mathcal{R}$, then the following actions are in $\mathcal{A}$:

- $a?^r$, denoting that the species participates in reaction a as a reactant, consuming

r molecules,

- $a!^p$, denoting that the species participates in reaction a as a product, producing p molecules,
- $a?^r!^p$, denoting that the species participates in reaction a as both a reactant and a product, like in the case of enzymatic reactions. Here, r molecules are consumed and p molecules are produced.

Elements in $\mathcal{A}$ will be denoted with α, α', α_1, ... For an action $\alpha \in \mathcal{A}$, the associated reaction $\mathsf{r}(\alpha)$ is defined by $\mathsf{r}(a?^r) = \mathsf{r}(a!^p) = \mathsf{r}(a?^r!^p) = a$. For a set of actions $A \subseteq \mathcal{A}$, we let $\mathsf{r}(A)$ denote $\mathsf{r}(A) = \{\mathsf{r}(\alpha) \mid \alpha \in A\}$.

Definition 1 (System) *The set $\mathcal{S}$ of* systems *over $\mathcal{N}$, $\mathcal{R}$ and $\mathcal{A}$ is defined by*

$$S \quad ::= \quad n[A]_L \quad | \quad S \parallel S$$

where: (i) $n \in \mathcal{N}$, (ii) $A \subseteq \mathcal{A}$, and (iii) L is a natural.

Intuitively, the system $n[A]_L$ represents that L molecules of species n are in the mixture and have a potential behaviour described by A, and $\parallel$ is the (commutative and associative) parallel composition operator, which allows us to model that several species coexist in the same mixture.

Given systems $n_i[A_i]_{L_i}$, for $i = 1, \ldots, k$, we will say that the system $n_1[A_1]_{L_1} \parallel \cdots \parallel n_k[A_k]_{L_k}$ is *well-formed* if for all $1 \leqslant i < j \leqslant k$ it holds that $n_i \neq n_j$. We will always assume to work with well-formed systems. Intuitively, well-formedness ensures that each species is represented by precisely one sequential component in the parallel composition.

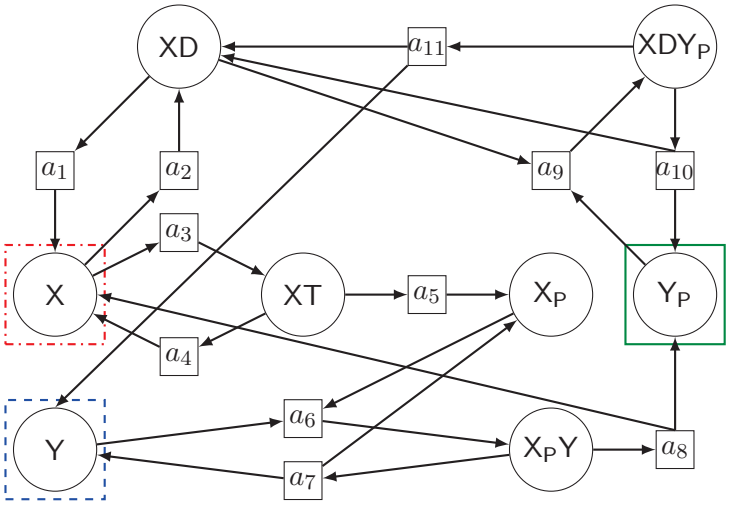


Figure 3.1: The Continuous Petri Net model for EnvZ/OmpR [1]

Example 1 (EnvZ/OmpR Osmoregulatory Signaling System) Escherichia Coli *is a bacterium whose cell contains a few million proteins of different types [11]. The* EnvZ/OmpR Osmoregulatory Signaling System *regulates two of those proteins. In order to support our exposition, in Figure 3.1 we report the graphical representation of this regulatory network as given in [1] by exploiting the formalism of Continuous Petri Nets. The main components of this network are EnvZ (*histine kinase*) and OmpR (*response regulator*), denoted, respectively, as* X *and* Y *in the figure. EnvZ phosphorylates OmpR (denoted* $\mathsf{Y_P}$ *in the figure) and itself (denoted* $\mathsf{X_P}$*), by binding and breaking down ATP. In the picture, we can see that this network is characterized by eight species, represented by places in the net, and eleven reactions, represented by transitions. More in detail, the reactants and the products of a reaction are those that are represented by the places that are the sources and the targets, respectively, of the transition represented by that reaction. For instance, the species* XT *is a reactant for the reactions* a_4 *and* a_5*, while is a product for* a_3*. In the Continuous Petri Nets representation, stoichiometric coefficients are represented by labels decoring transitions. Labels are omitted when the stoichiometric coefficient is 1. Indeed, in this example, each reaction uses precisely one molecule for each of its reactants and produces precisely one molecule for each of its products. Below we give the system* S *that represents the regulatory network with species* X*,* Y*,* XD *and* $\mathsf{Y_P}$ *having several molecules corresponding to 25, 150, 50 and 10, respectively, and the other species having no molecule.*

$$\begin{aligned} S \;=\; & \mathsf{X}[\{a_2?^1, a_3?^1, a_1!^1, a_4!^1, a_8!^1\}]_{25} \;\;\|\;\; \mathsf{Y}[\{a_6?^1, a_7!^1, a_{11}!^1\}]_{150} \\ & \|\; \mathsf{XT}[\{a_3!^1\}]_0 \;\|\; \mathsf{X_PY}[\{a_5!^1, a_7!^1\}]_0 \;\|\; \mathsf{X_PY}[\{a_6!^1\}]_0 \\ & \|\; \mathsf{XD}[\{a_1?^1, a_9?^1, a_2!^1, a_{10}!^1, a_{11}!^1\}]_{50} \;\|\; \mathsf{XDY_P}[\{a_9!^1\}]_0 \\ & \|\; \mathsf{Y_P}[\{a_9?^1, a_8!^1, a_{10}!^1\}]_{10} \end{aligned} \tag{3.1}$$

3.2.1 Behavioural Model

We describe systems' dynamics by means of two types of transitions. Those of the first type represent the *potential* behaviour of systems. They are of the form $S \xrightarrow{(a,w)} S'$, which models that the reaction a can take S to S'. Here, w is the *weight* of the transition and is a real number that will allow us to calculate the rate of the reaction a, which is needed in order to calculate the probability of all enabled reactions. The transitions of the second type describe how all enabled reactions for a system S compete and take it to a probability distribution over systems. They are of the form $S \Longrightarrow \pi$, where π is a discrete distribution over systems, namely a mapping $\pi : \mathcal{S} \to [0, 1]$ with $\sum_{S \in \mathcal{S}} \pi(S) = 1$. From the transitions of the form $S \Longrightarrow \pi$ we will derive transitions of the form $\pi \Longrightarrow \pi'$ that model the evolution of distributions of systems.

The transitions of the first type are derived by means of the inference rules in Table 3.1. The first rule represents that r molecules of species n are consumed since n takes part in reaction a as a reactant. The weight $\binom{L}{r}$ coincides with the number of ways r molecules of n can be taken out from the available L, namely, it is the number of sets

$$\frac{a?^r \in A \qquad L \geqslant r}{n[A]_L \xrightarrow{(a,\binom{L}{r})} n[A]_{L-r}} \qquad \frac{a!^p \in A}{n[A]_L \xrightarrow{(a,1)} n[A]_{L+p}} \qquad \frac{a?^r!^p \in A \qquad L \geqslant r}{n[A]_L \xrightarrow{(a,\binom{L}{r})} n[A]_{L-r+p}}$$

$$\frac{a \notin \mathsf{r}(A)}{n[A]_L \xrightarrow{(a,\#)} n[A]_L} \qquad \frac{S_1 \xrightarrow{(a,w_1)} S_1' \quad S_2 \xrightarrow{(a,w_2)} S_2'}{S_1 \parallel S_2 \xrightarrow{(a,w_1 \cdot w_2)} S_1' \parallel S_2'}$$

Table 3.1: Potential behaviour of systems

$$\frac{\mathsf{trgt}(S) = \{(a_i, w_i, S_i) \mid i \in I\}}{S \Longrightarrow \sum_{i \in I} \frac{(c_{a_i} \cdot w_i)}{\sum_{j \in I}(c_{a_j} \cdot w_j)} \delta(S_i)} \qquad \frac{S_i \Longrightarrow \pi_i}{\sum_{i \in I} p_i \delta(S_i) \Longrightarrow \sum_{i \in I} p_i \pi_i}$$

Table 3.2: Probabilistic behaviour of systems

of molecules of n that can take part to the reaction a. The second rule represents that p molecules of n are produced since n takes part in reaction a as a product. The weight 1 reflects that the number of molecules of the species does not impact the rate of the reaction. The third rule represents that r molecules of n are consumed and p molecules of n are produced by taking part in reaction a both as a reactant and as a product. The fourth rule is applied when n is not involved in reaction a. The inferred transition allows for composing the behaviour of $n[A]_L$ with that of the species that participate in reaction a as reactant and/or product. The last rule allows for combining the behaviour of systems running in parallel. Clearly, they have to agree upon the reaction to be taken and the resulting weight is the product of the weights of the composed transitions. Here, we extend classic product by $\# \cdot w = w \cdot \# = w$ for all reals w, and $\# \cdot \# = \#$.

For a system S we denote by $\mathsf{trgt}(S)$ the set of the triples $\{(a_i, w_i, S_i) \mid S \xrightarrow{(a_i,w_i)} S_i \text{ and } w_i \neq \#\}$. Notice that for $S = n_1[A_1]_{L_1} \parallel \cdots \parallel n_k[A_k]_{L_k}$, we have $(a_i, w_i, S_i) \in \mathsf{trgt}(S)$ with $w_i \neq \#$ if and only if $w_i \geqslant 1$ and S contains w_i different sets of reagents that can take part to reaction a_i.

In order to define the transitions of the form $S \Longrightarrow \pi$, we need some notation for distributions over systems. By $\delta(S)$ we denote the *point distribution* giving probability 1 to S and probability 0 to all S' different from S. Then, for a countable set of indexes I and distributions π_i with $i \in I$, the distribution $\sum_{i \in I} p_i \pi_i$ with all $p_i \geqslant 0$ and $\sum_{i \in I} p_i = 1$ is the convex distribution defined by $(\sum_{i \in I} p_i \pi_i)(S) = \sum_{i \in I} p_i \cdot \pi_i(S)$.

The first rule in Table 3.2 represents the competition between all enabled reactions of S. For each reaction a, we consider the *constant reaction* c_a associated with the reaction, where, intuitively, $c_a\, dt$ is the probability that a particular combination of reactants gives rise to reaction a in an infinitesimal time interval dt. The probability that S behaves as described by (a_i, w_i, S_i) is the ratio between the rate $c_{a_i} \cdot w_i$ of the reaction a_i and the sum of the rates of all reactions $\sum_{j \in I}(c_{a_j} \cdot w_j)$. To explain this point, we recall

that $c_{a_i} \cdot w_i$ is the parameter of the exponential distribution modelling the time elapsing between two consecutive occurrences of reaction a_i and $\sum_{j\in I}(c_{a_j} \cdot w_j)$ is the parameter of the exponential distribution modelling the time elapsing between two consecutive occurrences of arbitrary reactions. Summarizing, S reaches the convex distribution of systems $\sum_{i\in I} p_i\pi_i$ with $p_i = \frac{(c_{a_i}\cdot w_i)}{\sum_{j\in I}(c_{a_j}\cdot w_j)}$ and $\pi_i = \delta(S_i)$.

The second rule in Table 3.2 lifts the behaviour of systems to that of distributions over systems. We note that if there is a sequence of transitions $\pi_1 \Longrightarrow \pi_2 \Longrightarrow \ldots \pi_i \ldots$, for π_1 the point distribution $\delta(n_1[A_1]_{L_1} \parallel \cdots \parallel n_m[A_m]_{L_m})$, then all systems S in the support of any π_i are of the form $S = n_1[A_1]_{L'_1} \parallel \cdots \parallel n_m[A_m]_{L'_m}$, for suitable $L'_1, \ldots, L'_m$. Following [3, 4], the sequence $\pi_1 \Longrightarrow \pi_2 \Longrightarrow \ldots \pi_i \ldots$ is called an *evolution sequence*. In particular, it is the evolution sequence of S if $\pi_1 = \delta(S)$.

In a transition $S \Longrightarrow \pi$ the probability weight assigned to each element in the support of π depends on the rate of all reactions that are possible in S. This is no longer true in a transition $\pi \Longrightarrow \pi'$ since the rates of the chemical reactions from two different systems in the support of π are unrelated.

3.2.2 Remarks on the Semantic Model

We have described the behaviour of a system by means of an evolution sequence, namely a sequence of distributions of systems. Clearly, computing exactly these distributions is undoable. However, by adapting to our context the work in [3, 4, 59], we can easily obtain a randomized algorithm allowing us to estimate the evolution sequence of a system S. This randomized algorithm, detailed in Section 3.4, works as follows:

(i) we apply N times a procedure allowing us to compute a h-steps trajectory. Essentially, this coincides with applying N times the classical Gillespie algorithm [19]. For $i = 1, \ldots, N$, we obtain the trajectory $S_0^i, S_1^i, \ldots, S_h^i$, where all S_0^i coincide with S. Clearly, for all $j = 1, \ldots, h$ it holds that the samples $S_j^1, \ldots, S_j^N$ obtained at step j are independent and identically distributed;

(ii) for each $j = 1, \ldots, h$, we use $S_j^1, \ldots, S_j^N$ to derive the empirical distribution $\hat{\pi}_j$ defined simply by $\hat{\pi}_j(S') = \frac{|\{i | S_j^i = S' \text{ and } 1 \leqslant i \leqslant N\}|}{N}$;

(iii) by applying the weak law of large numbers to i.i.d. samples we infer that when N goes to infinite then $\hat{\pi}_j$ converges weakly to π_j, where π_j are the distributions such that $\delta(S) \Longrightarrow \pi_1 \Longrightarrow \pi_2 \ldots$.

3.3 Behavioural Distances over Systems

In [3, 4, 59] a notion of *behavioural distance* over cyber-physical systems was proposed, aiming to express how well the cyber components fulfil their tasks. In this section, we customize that theory for our model, clearly with a different target. In particular, given two systems that differ by the number of molecules of species, say $S_1 = n_1[A_1]_{L_1^1} \parallel \cdots \parallel$

$n_k[A_k]_{L_k^1}$ and $S_2 = n_1[A_1]_{L_1^2} \parallel \cdots \parallel n_k[A_k]_{L_k^2}$, we aim to *quantify the differences over their evolution sequences* $S_1 \Longrightarrow \pi_1^1 \Longrightarrow \pi_2^1 \ldots$ and $S_2 \Longrightarrow \pi_1^2 \Longrightarrow \pi_2^2 \ldots$. To this purpose, we proceed as follows:

(i) first, we introduce the concept of *distance between systems*, which relies on the difference between the number of molecules of the species;

(ii) then we lift this notion to a notion of *distance between distributions over systems*, so that we can quantify the distance between the distributions π_i^1 and π_i^2 that are reached by S_1 and S_2, respectively, at step i, and, finally,

(iii) we lift this distance to a *distance between the two evolution sequences*.

We start with a notion of distance between systems that focuses on a single species. We assume to know the minimum and maximum level $\min(n)$ and $\max(n)$ that a given species n may reach during the computation. Notice that these values can be estimated by applying our randomized algorithm quickly described above in Section 3.2.2 and detailed below in Section 3.4.

Definition 2 (n_i-distance over systems) *Assume two systems* $S_1 = n_1[A_1]_{L_1^1} \parallel \cdots \parallel n_k[A_k]_{L_k^1}$ *and* $S_2 = n_1[A_1]_{L_1^2} \parallel \cdots \parallel n_k[A_k]_{L_k^2}$. *Then the* distance between S_1 and S_2 with respect to species n_i, *or* n_i*-distance between* S_1 *and* S_2 *for short, is*

$$\mathbf{d}_{n_i}(S_1, S_2) = \frac{\mid L_i^1 - L_i^2 \mid}{\max(n_i) - \min(n_i)} \,.$$

Clearly, $\mathbf{d}_{n_i}(S_1, S_2)$ is a value in the interval $[0, 1]$. Moreover, $\mathbf{d}_{n_i}$ is a *1-bounded pseudometric*, namely a function $\mathbf{d}_{n_i} : \mathcal{S} \times \mathcal{S} \to [0, 1]$ such that for all systems S_1, S_2, S_3 the following properties hold:

- $\mathbf{d}_{n_i}(S_1, S_1) = 0$,
- $\mathbf{d}_{n_i}(S_1, S_2) = \mathbf{d}_{n_i}(S_2, S_1)$, and
- $\mathbf{d}_{n_i}(S_1, S_3) \leqslant \mathbf{d}_{n_i}(S_1, S_2) + \mathbf{d}_{n_i}(S_2, S_3)$.

Notice that pseudometric $\mathbf{d}_{n_i}$ is not a *metric*, since, in general, it does not hold that $\mathbf{d}_{n_i}(S_1, S_2) = 0$ implies $S_1 = S_2$.

The second step to obtain the evolution distance consists of lifting distances on systems to distances on distributions over systems. Among the several notions of lifting for pseudometric proposed in the literature (see [60] for a survey), following [4] we opt for that known as *Wasserstein distance* or *Kantorovich-Rubinstein pseudometric* since it preserves the properties of the ground pseudometric and is computationally tractable via statistical inference.

In order to recall formally that notion, we need to introduce the notion of *matching*, also known as *measure coupling* [61], for pairs distributions.

Definition 3 (Matching, [61]) *Assume a set X. A* matching *for a pair of distributions (π, π') over X is a distribution ω over the product state space $X \times X$ with left marginal π, i.e. $\sum_{x' \in X} \omega(x, x') = \pi(x)$ for all $x \in X$, and right marginal π', i.e. $\sum_{x \in X} \omega(x, x') = \pi'(x')$ for all $x' \in X$. We let $\Omega(\pi, \pi')$ denote the set of all matchings for (π, π').*

According to [61], a matching in $\Omega(\pi, \pi')$ may be understood as a transportation schedule for the shipment of probability mass from π to π'. Then, by exploiting that notion, we can introduce the Kantorovich lifting.

Definition 4 (Kantorovich metric, [62]) *The* Kantorovich lifting *of a pseudometric d over a set X is the pseudometric $\mathbf{K}(d)$ over distributions over X defined for all distributions π, π' by*

$$\mathbf{K}(d)(\pi, \pi') = \min_{\omega \in \Omega(\pi, \pi')} \sum_{x, x' \in X} \omega(x, x') \cdot d(t, t').$$

The reader familiar with probability theory might have recognized the Kantorovich lifting $\mathbf{K}(d)(\pi, \pi')$ as the minimum expected value of the ground distance d, over the supports of π and π', with respect to the distribution ω. According to [61], $\mathbf{K}(d)(\pi, \pi')$ gives the optimal cost for shipping probability mass from π to π' when $d(x, x')$ is the unit cost for shipping mass from x to x'.

We notice that pseudometric $\mathbf{K}(\mathbf{d}_{n_i})$ preserves the 1-boundedness property of $\mathbf{d}_{n_i}$.

From the distances with respect to all species $\mathbf{K}(\mathbf{d}_{n_1}), \ldots, \mathbf{K}(\mathbf{d}_{n_k})$, we can derive a unique notion of distance by exploiting weights that quantify the *relevance* that we want to assign to each species.

Definition 5 (Distance between distributions over systems) *Assume the* weights $w_1, \ldots,$
$w_k \geqslant 0$ *such that* $\sum_{j=1}^{k} w_j = 1$. *The distance* $\mathbf{dd}$ *between distributions over systems induced by distances $\mathbf{d}_{n_1}, \ldots, \mathbf{d}_{n_k}$ and weights $w_1, \ldots, w_k$ is defined by*

$$\mathbf{dd}(\pi, \pi') = \sum_{i=1}^{k} w_i \cdot \mathbf{K}(\mathbf{d}_{n_i})(\pi, \pi') \ .$$

Clearly, being a convex combination of 1-bounded pseudometrics, the function $\mathbf{dd}$ is a 1-bounded pseudometric. We argue that $\mathbf{dd}$ can be computed efficiently, by following [63]. In detail, computing $\mathbf{dd}(\pi, \pi')$ requires computing the Kantorovich liftings $\mathbf{K}(\mathbf{d}_{n_1})(\pi, \pi'), \ldots, \mathbf{K}(\mathbf{d}_{n_k})(\pi, \pi')$. Both π and π' are discrete distributions, let us assume that their cardinality is N. If π and π' are estimated by our randomized algorithm, then N will be a parameter of the algorithm. In computing each lifting $\mathbf{K}(\mathbf{d}_{n_i})(\pi, \pi')$, each system in the support of π and π' can be viewed as a real, obtained as the cardinality of the molecules of n_i divided by $\max(n_i) - \min(n_i)$. Therefore, π and π' can be viewed as multisets of reals of cardinality N. This property is exploited in [63] as follows. In $O(N \log N)$ these two multisets can be ordered. Let $u_1 \leqslant \cdots \leqslant u_N$ and $v_1 \leqslant \cdots \leqslant v_N$ be the ordered

sequences. Notice that the ground distance $\mathbf{d}_{n_i}$ between two reals in the support of π and π' is a real (the absolute value of their difference, namely $\mathbf{d}_{n_i}(u_i, v_j) = |u_i - v_j|$). Following [63], we have that $\mathbf{K}(\mathbf{d}_{n_i})(\pi, \pi') = \frac{1}{N}\sum_{j=1}^{N} \mathbf{d}_{n_i}(u_j, v_j)$. Therefore, computing $\mathbf{dd}$ can be done in $O(k \cdot N \log N)$.

We now need to lift $\mathbf{dd}$ to a distance on evolution sequences. To this end, we observe that the evolution sequence of a system includes the distributions over systems induced after each computation step. Following [4] we define the evolution distance as a sort of *weighted infinity norm* of the tuple of the Kantorovich distances between the distributions in the evolution sequences.

Definition 6 (Evolution distance) *For a pseudometric* $\mathbf{dd}$ *over distributions over* $\mathcal{S}$ *and an interval* $[\tau_1, \tau_2]$*, the* evolution distance *over* $[\tau_1, \tau_2]$ *is the mapping* $\mathbf{E}(\mathbf{dd})_{[\tau_1,\tau_2]}$ *such that, given systems* S_1 *and* S_2 *and their evolution sequences* $S_1 \Longrightarrow \pi_1^1 \Longrightarrow \pi_2^1 \Longrightarrow \ldots \pi_i^1 \ldots$ *and* $S_2 \Longrightarrow \pi_1^2 \Longrightarrow \pi_2^2 \Longrightarrow \ldots \pi_i^2 \ldots$ *we have*

$$\mathbf{E}(\mathbf{dd})_{[\tau_1,\tau_2]}(S_1, S_2) = \max_{\tau_1 \leqslant i \leqslant \tau_2} \mathbf{dd}(\pi_i^1, \pi_i^2)$$

Since $\mathbf{dd}$ is a 1-bounded pseudometric, we can easily derive the same property for $\mathbf{E}(\mathbf{dd})_{[\tau_1,\tau_2]}$.

We remark that, as argued in [59], the choice of defining the evolution metric as the pointwise maximal distance in time between the evolution sequences of systems is natural and reasonable when one aims to evaluate the highest distance on the behaviour of systems. However, in different contexts, it would be more appropriate to use a different *aggregation function* over the tuple of Kantorovich distances instead of max. The definition of $\mathbf{E}(\mathbf{dd})_{[\tau_1,\tau_2]}$ in Definition 6 could be replaced by a definition parametric with respect to aggregation functions. In order to avoid a too-heavy notation, we decided to opt for the present formulation.

3.3.1 Robustness

Technically, in order to formalise the notion of robustness we need two distances, one taking into account input species and the other taking into account the output species. These will be the input distance $\mathbf{dd}_i$ and the output distance $\mathbf{dd}_o$ over distributions and will be defined as in Definition 5 by giving positive weights to only input and only output species, respectively. The definition of robustness is parametric with respect to these two distances, two thresholds η_1 and η_2 and two intervals I_1 and I_2: a system S is robust with respect to these parameters if whenever we consider a system S' whose input distance from S in the interval I_1 remains below η_1, then the output distance observed in the interval I_2 remains below η_2.

Definition 7 (Robustness) *A system* S *is* $(\mathbf{dd}_i, \mathbf{dd}_o, I_1, I_2, \eta_1, \eta_2)$-robust *if for all systems* S'*:*

$$\mathbf{E}(\mathbf{dd}_i)_{I_1}(S, S') \leqslant \eta_1 \text{ implies } \mathbf{E}(\mathbf{dd}_o)_{I_2}(S, S') \leqslant \eta_2$$

Clearly, we will use $I_1 = [0, 0]$ if the input distance is relevant only in the initial state of systems and we use $I_2 = [\tau_1, h]$ if we want to observe the behaviour of systems up to a time horizon h.

3.4 Estimating the Evolution Distance

In this section, we follow [3, 4, 59] and propose a procedure to estimate the evolution distance via statistical techniques. First, in Subsection 3.4.1, we show how we can estimate the evolution sequence of a given system S. The idea was quickly anticipated in Section 3.2.2. Then, in Subsection 3.4.2 we evaluate the distance between two systems S_1 and S_2 on their estimated evolution sequences.

All the procedures presented in this section have been implemented in Python in the tool SPEBNR, and are available at `https://github.com/dmanicardi/spebnr`. This implementation can be viewed as a customisation of the tool SPEAR (`https://github.com/quasylab/spear`) implementing the algorithms in [59]. Details on SPEBNR are in Appendix A.

3.4.1 Estimating the Evolution Sequence

Given a system S_0 and an integer h we can use the function SIMULATE(S_0, h), defined in [59] and adapted to our purpose in Figure 3.2, to sample a sequence of systems of the form $seq = (S_0, S_1, \ldots, S_h)$ that represents h-steps of a computation starting from S_0. Each step of the sequence is computed by using the function SIMSTEP, also defined in Figure 3.2. Here, we assume that $\mathcal{R} = \{a_1, \ldots, a_{|\mathcal{R}|}\}$ and we let RAND be a function that allows us to get a random number uniformly distributed in in $(0, 1]$. Essentially, SIMSTEP computes one step of Gillespie's algorithm. Our version of SIMSTEP is significantly different with respect to that provided in [59], which was proposed for cyber-physical systems and had to deal with the interactions between the cyber and the physical components of the systems. Our version of SIMSTEP returns also the minimum and maximum level of each species n in all generated systems. In computing the arrays m and M with the minimal and maximal levels of species we exploit the function proj_i mapping a system to the level of species n_i, namely $\mathsf{proj}_i(n_1[A_1]_{L_1} \parallel \cdots \parallel n_k[A_k]_{L_k}) = L_i$.

To compute the empirical evolution sequence of a system S, we used the function ESTIMATE in Figure 3.3, which is a customisation of the ESTIMATE procedure proposed in [59]. The function ESTIMATE(S, h, N) invokes N times the function SIMULATE(S, h) in Figure 3.2 in order to sample N sequences of systems $(S_0^i, \ldots, S_h^i)$, for $i = 1, \ldots, N$, each modelling h steps of a computation from $S = S_0$. Thus, a sequence of tuples of samples $\{E_0, \ldots, E_h\}$ is computed, where each E_j is the tuple $(S_j^1, \ldots, S_j^N)$ of systems observed at time j in each of the N sampled computations. Notice that, for each $j \in \{0, \ldots, h\}$, the samples $S_j^1, \ldots, S_j^N$ are independent and identically distributed. Each E_j can be used to estimate the distribution $\pi_{S,j}$, namely the j^{th} element of the evolution sequence $\delta(S) = \pi_{S,0} \Longrightarrow \pi_{S,1} \Longrightarrow \ldots \Longrightarrow \pi_{S,k}$. For any j, with $0 \leqslant j \leqslant h$, we let $\hat{\pi}_{S,j}^N$ be the

function SIMULATE($n_1[A_1]_{L_1} \parallel \cdots \parallel n_k[A_k]_{L_k}, h$)
  $step \leftarrow 1$
  $m \leftarrow [0, \ldots, 0]$
  $M \leftarrow [0, \ldots, 0]$
  $S \leftarrow n_1[A_1]_{L_1} \parallel \cdots \parallel n_k[A_k]_{L_k}$
  $seq \leftarrow (S)$
  while $step \leqslant h$ **do**
    $S \leftarrow$ SIMSTEP(S)
    $seq.append(S)$
    for $i = 1, \ldots, k$ **do**
      $m(i) = \min(m(i), \mathsf{proj}_i(S))$
      $M(i) = \max(M(i), \mathsf{proj}_i(S))$
    end for
    $step \leftarrow step + 1$
  end while
  return (seq, m, M)
end function

function SIMSTEP($n_1[A_1]_{L_1} \parallel \cdots \parallel n_k[A_k]_{L_k}$)
  for $j = 1, \ldots, |\mathcal{R}|$ **do**
    for $i = 1 \ldots k$ **do**

$$w_{j,i} \leftarrow \begin{cases} \binom{L_i}{r} & \text{if } a_j?^r \in A_i \text{ or } a_j?^r!^p \in A_i \\ 1 & \text{if } a_j!^p \in A_i \\ \# & \text{otherwise} \end{cases}$$

    end for
    $w_j = c_{a_j} \cdot \prod_{i=1}^{k} w_{j,i}$
  end for
  for $j = 1, \ldots, |\mathcal{R}|$ **do**

$$p_j = \begin{cases} \frac{w_j}{\sum_{j=1}^{|\mathcal{R}|} w_j} & \text{if } w_j \neq \# \\ 0 \text{ otherwise} \end{cases}$$

  end for
  $u \leftarrow$ RAND()
  let j s.t. $\sum_{l=1}^{j-1} p_l < u \leqslant \sum_{l=1}^{j} p_l$
  for $i = 1, \ldots, k$ **do**

$$L'_i \leftarrow L_i + \begin{cases} -r & \text{if } a_j?^r \in A_i \\ -r + p & \text{if } a_j?^r!^p \in A_i \\ +p & \text{if } a_j!^p \in A_i \\ 0 & \text{otherwise} \end{cases}$$

  end for
  return $n_1[A_1]_{L'_1} \parallel \cdots \parallel n_k[A_k]_{L'_k}$
end function

Figure 3.2: Functions for simulating a computation

1: **function** ESTIMATE(S, h, N)
2: $\mathsf{m} = [0, \ldots, 0]$
3: $\mathsf{M} = [0, \ldots, 0]$
4: $\forall j : (0 \leqslant j \leqslant h) : E_j \leftarrow \emptyset$
5: $counter \leftarrow 0$
6: **while** $counter < N$ **do**
7: $((S_0, \ldots, S_h), m, M) \leftarrow$ SIMULATE(S, h)
8: $\forall j : (0 \leqslant j \leqslant h) : E_j \leftarrow E_j \cup \{S_j\}$
9: $\forall l : (0 \leqslant l \leqslant k) : \mathsf{m}(l) = \min(\mathsf{m}(l), m(l))$
10: $\forall l : (0 \leqslant l \leqslant k) : \mathsf{M}(l) = \max(\mathsf{M}(l), M(l))$
11: $counter \leftarrow counter + 1$
12: **end while**
13: **return** $(\{E_0, \ldots, E_h\}, \mathsf{m}, \mathsf{M})$
14: **end function**

Figure 3.3: Function allowing for deriving an empirical evolution sequence

distribution such that for any system S' we have

$$\hat{\pi}^N_{S,j}(S') = \frac{|E_j \cap \{S'\}|}{N}.$$

Finally, we call $\hat{\pi}^N_S = \hat{\pi}^N_{S,0} \ldots \hat{\pi}^N_{S,h}$ as the *empirical evolution sequence of* S. Notice that by applying the weak law of large numbers to the i.i.d samples, we get that when N goes to infinite $\hat{\pi}^N_{S,j}$ converges weakly to $\pi_{S,j}$.

The algorithms in Figures 3.2 and 3.3 have been implemented in Python in the tool SPEBNR, and are available at `https://github.com/dmanicardi/spebnr`.

3.4.2 Estimating the Evolution Distance between Evolution Sequences

We have seen that function ESTIMATE allows us to collect independent samples at each time step from 0 to a deadline h. We proceed to show how these samples can be used to estimate the distance between two systems S_1 and S_2.

Following [63], to estimate the Kantorovich lifting $\mathbf{K}(d)$ of a distance d between two (unknown) distributions over reals π_1 and π_2, one can use N independent samples $\{e^1_1, \ldots, e^N_1\}$ taken from π_1 and $\ell \cdot N$ independent samples $\{e^1_2, \ldots, e^{\ell \cdot N}_2\}$ taken from π_2, with ℓ a natural suitably chosen. After that, one orders $\{e^1_1, \ldots, e^N_1\}$ and $\{e^1_2, \ldots, e^{\ell \cdot N}_2\}$, thus obtaining the two sequences of values $u_1 \leqslant \cdots \leqslant u_N$ and $v_1 \leqslant \cdots \leqslant v_{\ell \cdot N}$, respectively. The value $\mathbf{K}(d)(\pi_1, \pi_2)$ can be approximated as $\frac{1}{\ell N} \sum_{z=1}^{\ell N} \max\{v_z - u_{\lceil \frac{z}{\ell} \rceil}, 0\}$.

In our case, since we need to estimate the Kantorovich lifting $\mathbf{K}(\mathbf{d}_{n_i})$ for species n_i between the distributions over systems $\pi_{S_1,j}$ and $\pi_{S_2,j}$ that are reached in j steps from S_1 and S_2, respectively, we have to proceed as follows. We obtain the set of reals $\{e^1_1, \ldots, e^N_1\}$ by extracting N samples from $\pi_{S_1,j}$, and, then, by taking from each sample the level of species n_i and by dividing it by $\max(n_i) - \min(n_i)$. Analogously, we obtain the reals $\{e^1_2, \ldots, e^{\ell \cdot N}_2\}$ by extracting $\ell \cdot N$ samples from $\pi_{S_2,j}$, and, then, by taking from each sample the level of species n_i and by dividing it by $\max(n_i) - \min(n_i)$.

Clearly, having the estimation of $\mathbf{K}(\mathbf{d}_{n_i})(\pi_{S_1,j}, \pi_{S_2,j})$ for all species, we can derive an estimation for $\mathbf{dd}(\pi_{S_1,j}, \pi_{S_2,j})$ and, then, for $\mathbf{E}(\mathbf{dd})_{[a,b]}(S_1, S_2)$. The whole procedure is realized by functions DISTANCE and COMPUTEH in Figure 3.4, available at `https://github.com/dmanicardi/spebnr` in SPEBNR. The former takes as input the two systems S_1 and S_2 to compare, the weights assigned to species $W = (w_1, \ldots, w_k)$, the parameter h giving the number of computation steps that are observed, the parameters N and ℓ used to obtain the samplings of computation, and the bounds of the interval a and b. Function DISTANCE collects the samples $E_{1,0}, \ldots, E_{1,h}$ and $E_{2,0}, \ldots, E_{2,h}$ of possible computations of length h from S_1 and S_2. Then, for each $step \in [a, b]$, the n_i distance at time $step$ is computed via the function COMPUTEH$(E_{1,step}, E_{2,step}, \mathsf{proj}_i/(\mathsf{M}(i) - \mathsf{m}(i)))$, where $\mathsf{m}(i)$ and $\mathsf{M}(i)$ are the minimum and maximum level of species n_i in all systems that are generated.

Due to the sorting of $\{\nu_h \mid h \in [1, \ldots, \ell N]\}$ the complexity of function COMPUTEH is $O(\ell N \log(\ell N))$ (cf. [63]). We refer the interested reader to [64, Corollary 3.5, Equation (3.10)] for an estimation of the approximation error given by the evaluation of the Kantorovich distance over N samples.

function DISTANCE$(S_1, S_2, W, h, N, \ell, a, b)$
  $(\{E_{1,0}, \ldots, E_{1,h}\}, \mathsf{m}_1, \mathsf{M}_1) \leftarrow$ ESTIMATE(S_1, h, N)
  $(\{E_{2,0}, \ldots, E_{2,h}\}, \mathsf{m}_2, \mathsf{M}_2)) \leftarrow$ ESTIMATE$(S_2, h, \ell N)$
  $\forall l : (l = 1, \ldots, k) : \mathsf{m}(l) = \min(\mathsf{m}_1(l), \mathsf{m}_2(l))$
  $\forall l : (l = 1, \ldots, k) : \mathsf{M}(l) = \max(\mathsf{M}_1(l), \mathsf{M}_2(l))$
  $dist \leftarrow 0$
  for all $step \in [a, b]$ **do**
    for all $w_i \neq 0$ in W **do**
      $\rho = \mathsf{proj}_i/(\mathsf{M}(i) - \mathsf{m}(i))$
      $dist_{step,i} \leftarrow$ COMPUTEH$(E_{1,step}, E_{2,step}, \rho)$
    end for
    $dist_{step} = \sum_{i|w_i \neq 0} w_i \cdot dist_{step,i}$
    $dist \leftarrow \max\{dist, dist_{step}\}$
  end for
  return $dist$
end function

function COMPUTEH(E_1, E_2, ρ)
  $(S_1^1, \ldots, S_1^N) \leftarrow E_1$
  $(S_2^1, \ldots, S_2^{\ell N}) \leftarrow E_2$
  $\forall j : (1 \leqslant j \leqslant N) : u_j \leftarrow \rho(S_1^j)$
  $\forall z : (1 \leqslant z \leqslant \ell N) : v_z \leftarrow \rho(S_2^z)$
  re index $\{u_j\}$ s.t. $u_j \leqslant u_{j+1}$
  re index $\{v_z\}$ s.t. $v_z \leqslant v_{z+1}$
  return $\frac{1}{\ell N} \sum_{z=1}^{\ell N} \max\{v_z - u_{\lceil \frac{z}{\ell} \rceil}, 0\}$
end function

Figure 3.4: Functions used to estimate the evolution distance on systems.

4
Applying Robustness Analysis

In this chapter, we apply the theory presented in Chapter 3 to study some robustness properties of three different systems that were already analysed in [1], within the deterministic model. Experiments are conducted in SPEBNR.

In Section 4.1 we consider an example of a regulatory network, namely the *EnvZ/OmpR Osmoregulatory Signaling System* in Escherichia Coli. Then, in Section 4.2 we study the mechanism of *Bacterial Chemotaxis* of Escherichia Coli. Finally, in Section 4.3 we deal with an abstract chemical reaction network, called *Enzyme Activity at Saturation* in [1].

The steady-state analysis conducted in [1] concluded that all these systems are robust. Our analysis will show that these systems are also step-by-step robust.

4.1 The *EnvZ/OmpR Osmoregulatory Signalling* System

In this section, we apply the approach of Chapter 3 to study the robustness property of a regulatory network. In particular, we consider the *EnvZ/OmpR Osmoregulatory Signaling* System in Escherichia Coli, earlier introduced in Example 1. This system was already analysed in [1], within the deterministic model. As reported in [1], an interesting question is whether the initial concentrations of input species EnvZ and OmpR, which are represented in Figure 3.1 by X and Y, respectively, impact on the long-run concentration of output species phosphorylated OmpR, represented by $\mathsf{Y_P}$. We can answer to that question by studying the robustness of the system with respect to an input distance $\mathbf{dd}_i$ and an output distance $\mathbf{dd}_o$ defined so that they capture the differences over both X and Y, and over $\mathsf{Y_P}$, respectively.

The analysis in [1] concluded that the concentration of output species *at steady state* does not depend on the initial value of input species. In other words, at steady state, an

original system and its perturbed versions obtained by varying the initial concentration of input species X and Y do not exhibit any difference in the concentration of output species $\mathsf{Y_P}$. Our analysis will allow us to study the step-by-step distances between the original system and the perturbed one, which is abstracted away in the steady-state analysis.

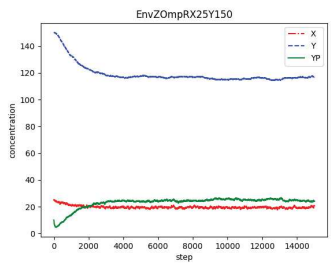
Initial concentration	Reaction constants	Chemical reactions
$\mathsf{X} = 25\diamondsuit$	$c_{a_1}, c_{a_2}, c_{a_3}, c_{a_4}, = 0.5$	$\mathsf{XD} \underset{a_2}{\overset{a_1}{\rightleftharpoons}} \mathsf{X}$
$\mathsf{Y} = 150\diamondsuit$	$c_{a_5}, c_{a_{11}} = 0.1$	$\mathsf{XT} \underset{a_4}{\overset{a_3}{\rightleftharpoons}} \mathsf{X}$
$\mathsf{XT} = 0$	$c_{a_6}, c_{a_9} = 0.02$	$\mathsf{XT} \xrightarrow{a_5} \mathsf{X_P}$
$\mathsf{X_P} = 0$	$c_{a_7}, c_{a_8}, c_{a_{10}} = 0.5$	$\mathsf{X_P} + \mathsf{Y} \underset{a_7}{\overset{a_6}{\rightleftharpoons}} \mathsf{X_PY}$
$\mathsf{X_PY} = 0$		$\mathsf{X_PY} \xrightarrow{a_8} \mathsf{X} + \mathsf{Y_P}$
$\mathsf{XD} = 50$		$\mathsf{XD} + \mathsf{Y_P} \xrightarrow{a_9} \mathsf{XDY_P}$
$\mathsf{XDY_P} = 0$		$\mathsf{XDY_P} \xrightarrow{a_{10}} \mathsf{XD} + \mathsf{Y_P}$
$\mathsf{Y_P} = 10$		$\mathsf{XDY_P} \xrightarrow{a_{11}} \mathsf{XD} + \mathsf{Y}$

Table 4.1: The initial concentrations, the reaction constants and the chemical reactions of the EnvZ/OmpR Osmoregulatory Signalling system. The concentration of input species is marked by $\diamondsuit$ and varies to estimate robustness.

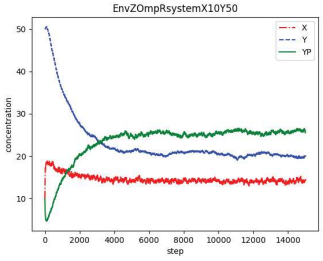
Let us focus on system S in Equation 3.1, whose initial species concentration are the same as in [1] and are reported in Table 4.1, together with the set $\mathcal{R}$ of reactions and the reaction constants. We start our analysis by studying the distances between S and two systems that were obtained in [1] by perturbing its initial state: system S_1 starts with $\mathsf{X} = 10$ and $\mathsf{Y} = 50$, system S_2 starts with $\mathsf{X} = 250$ and $\mathsf{Y} = 1000$. In [1] evidence is given that, at steady state, S, S_1 and S_2 have the same level of $\mathsf{Y_P}$.

Before analysing robustness, we focus on the simulation of S, S_1 and S_2. By applying function Estimate in Figure 3.3, we can simulate the evolution of these systems. For each system, we have applied function Estimate with parameters $h = 15000$ and $N = 100$. In Figures 4.1a, 4.1c, 4.1e we report the step-by-step average value obtained for all relevant species, namely X, Y and $\mathsf{Y_P}$, whereas in Figures 4.1b, 4.1d, 4.1f we have highlighted the step-by-step average value obtained for $\mathsf{Y_P}$.

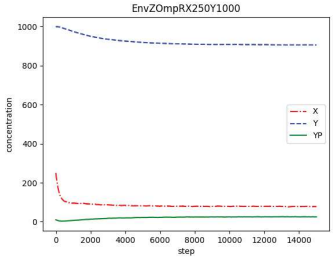
By applying function Distance in Figure 3.4 we can estimate the evolution distance between the original system S and the two perturbed ones. We define the input distance $\mathbf{dd}_i$ so that only the weights associated with X and Y are positive, in particular, we decided that both weights are 0.5 to reflect that the two species have the same relevance. Then, we define the output distance $\mathbf{dd}_o$ so that only the weight associated with $\mathsf{Y_P}$ is positive. In Table 4.2 we report the results obtained by applying Distance with the



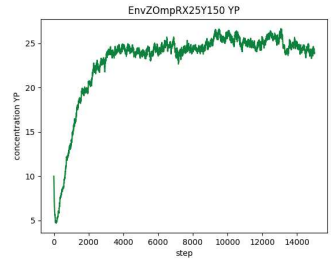
(a) Evolution of $\mathsf{X}, \mathsf{Y}, \mathsf{Y_P}$ in S



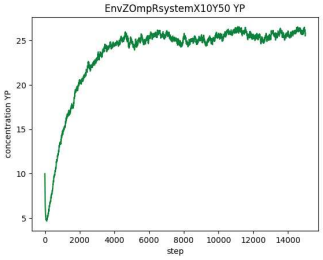
(c) Evolution of $\mathsf{X}, \mathsf{Y}, \mathsf{Y_P}$ in S_1



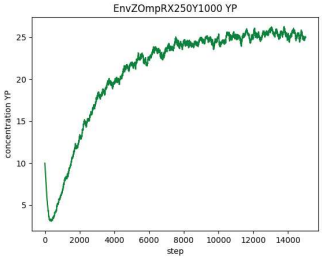
(e) Evolution of $\mathsf{X}, \mathsf{Y}, \mathsf{Y_P}$ in S_2



(b) Evolution of $\mathsf{Y_P}$ in S



(d) Evolution of $\mathsf{Y_P}$ in S_1



(f) Evolution of $\mathsf{Y_P}$ in S_2

Figure 4.1: Simulation of system S and its perturbed versions S_1 and S_2

following parameters: $N = 50$ runs for the original system, $\ell \cdot N = 100$ runs for the perturbed systems, $h = 15000$ steps for each run. In detail, for each perturbed system, in Table 4.2 we report:

(i) the input distance between the perturbed system and S, which is simply computed by focusing on the initial level of input species in S and the perturbed systems, and the minimal and maximal levels that are stored in m and M by DISTANCE;

(ii) the maximal output distance $\mathbf{dd}_o$ computed by DISTANCE in interval $[7500, 15000]$;

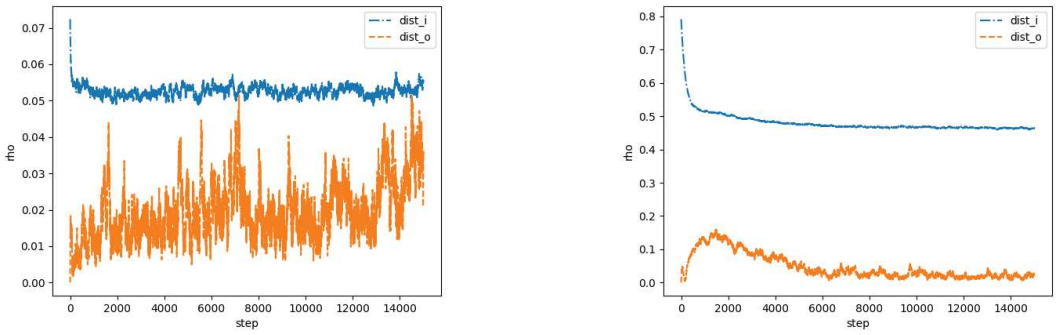
(iii) a pointer to a figure describing the step-by-step evolution of input and output distance between S and the perturbed system.

We noted that $\mathbf{dd}_i(S, S_2)$ is one order of magnitude higher than $\mathbf{dd}_i(S, S_1)$, whereas $\mathbf{E}(\mathbf{dd}_o)_{[7500,15000]}(S, S_1)$ and $\mathbf{E}(\mathbf{dd}_o)_{[7500,15000]}(S, S_2)$ are close each other and are quite low numbers. This suggests that even if one changes the initial level of X and Y in a light or a heavy way, the step-by-step variation of $\mathsf{Y_P}$ is light in all cases.

However, in our setting, a more systematic analysis, for studying how the initial concentrations of X and Y in a system S influence the concentration of $\mathsf{Y_P}$ in a temporal interval I, can be conducted as follows. We fix the maximal input distance η_1 between system S and its perturbed versions and, then, we estimate for which η_2 the system S is $(\mathbf{dd}_i, \mathbf{dd}_o, I_1 = [0, 0], I, \eta_1, \eta_2)$-robust. More precisely, in order to estimate η_2, for a suitable n we sample n systems $S_1, \ldots, S_n$ satisfying $\mathbf{E}(\mathbf{dd}_i)_{[0,0]}(S, S_i) \leqslant \eta_1$ and we fix $\eta_2 = \max_{i=1,\ldots,n} \mathbf{E}(\mathbf{dd}_o)_I(S, S_i)$.

System	Initial X	Initial Y	$\mathbf{dd}_i$ from S	$\mathbf{E}(\mathbf{dd}_o)_{[7500,15000]}$ from S	Figure showing $\mathbf{dd}_i$ and $\mathbf{dd}_o$ step-by-step
S_1	10	50	0.072407	0.051207	Figure 4.2a
S_2	250	1000	0.79145	0.056034	Figure 4.2b

Table 4.2: Input and output distance between S and its perturbed versions S_1 and S_2



(a) $\mathbf{dd}_i(S, S_1)$ and $\mathbf{dd}_o(S, S_1)$, step-by-step

(b) $\mathbf{dd}_i(S, S_2)$ and $\mathbf{dd}_o(S, S_2)$, step-by-step

Figure 4.2: Evolution of $\mathbf{dd}_i$ and $\mathbf{dd}_o$ between S and its perturbed versions S_1 and S_2

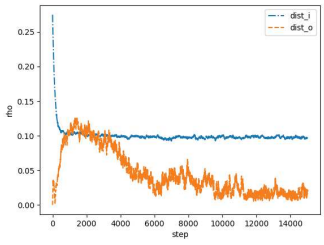
We have considered five different values for η_1: 0.3, 0.4, 0.5, 0.6, 0.7. For each choice for η_1 we have sampled $n = 20$ systems at input distance $\mathbf{dd}_i$ from S bounded by η_1. More precisely, we decided to sample these 20 systems so that they are at a $\mathbf{dd}_i$ distance from S in the interval $(\eta_1 - 0.1, \eta_1]$. Then, we have estimated the η_2 for which S is $(\mathbf{dd}_i, \mathbf{dd}_o, [0, 0], I, \eta_1, \eta_2)$-robust, for $I = [7500, 15000]$. In each experiment, we simulated runs consisting of $h = 15000$ reactions, and we considered $N = 50$ runs for the original system S and $\ell \cdot N = 100$ runs for the $n = 20$ perturbed systems. For each of the five choices for η_1, in Table 4.3 we report the value η_2 for which we have estimated the $(\mathbf{dd}_i, \mathbf{dd}_o, [0, 0], [7500, 15000], \eta_1, \eta_2)$-robustness, a pointer to the picture showing the step-by-step evolution of $\mathbf{dd}_i$ and $\mathbf{dd}_o$ between S and the perturbed system realizing η_2 and, finally, the initial concentration levels of X and Y of that system. Summarizing, at varying of η_1 in $[0.3, 0.7]$ we get values for η_2 that are close to each other and are one order of magnitude lower than η_1. This suggests that S, which was classified as robust in the steady state analysis in [1], has a good level of robustness also in the step-by-step approach. The analysis in [1] allows one to conclude that at steady state the level of $\mathsf{Y_P}$ does not depend on the initial level of X and Y, our analysis allows to conclude that if one varies the initial level of X and Y then, step-by-step, the changes in $\mathsf{Y_P}$ are limited and smooth.

The SPEBNR code used for the analysis can be found at `https://github.com/`

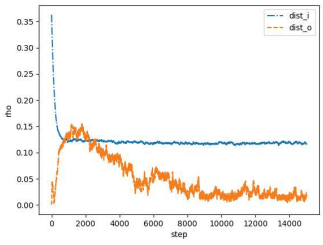
`dmanicardi/spebnr/tree/master/caseStudies/envZOmpR`. For each value for η_1, the analysis took 150 minutes on a 1.70 GHz i7-1255U, with 16.00 GB RAM.

η_1	η_2	$\mathbf{dd}_i$ and $\mathbf{dd}_o$ between S and the system realizing η_2	Initial concentration of X in system realizing η_2	Initial concentration of Y in system realizing η_2
0.3	0.065862	Figure 4.3a	165	101
0.4	0.053621	Figure 4.3b	222	134
0.5	0.059483	Figure 4.3c	57	1033
0.6	0.078448	Figure 4.3d	269	500
0.7	0.059138	Figure 4.3e	263	669

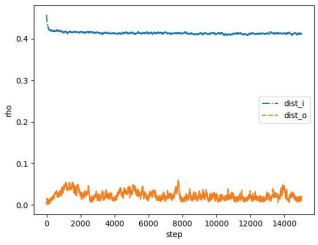
Table 4.3: EnvZ/OmpR Osmoregulatory Signaling System: robustness at varying of η_1



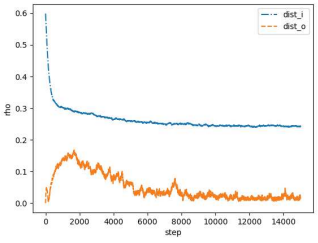
(a) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.3$



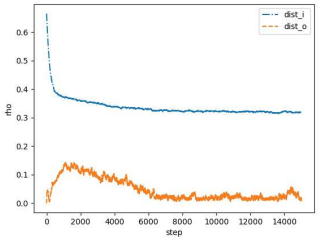
(b) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.4$



(c) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.5$



(d) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.6$



(e) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.7$

Figure 4.3: EnvZ/OmR Osmoregulatory Signaling System: evolution of $\mathbf{dd}_i$ and $\mathbf{dd}_o$ at varying of η_1

4.2 The *Bacterial Chemotaxis* System

In this section, we consider the mechanism of Bacterial Chemotaxis in Escherichia Coli, which was originally studied in [11, 56] and whose robustness was already analysed in

[1] within the deterministic model.

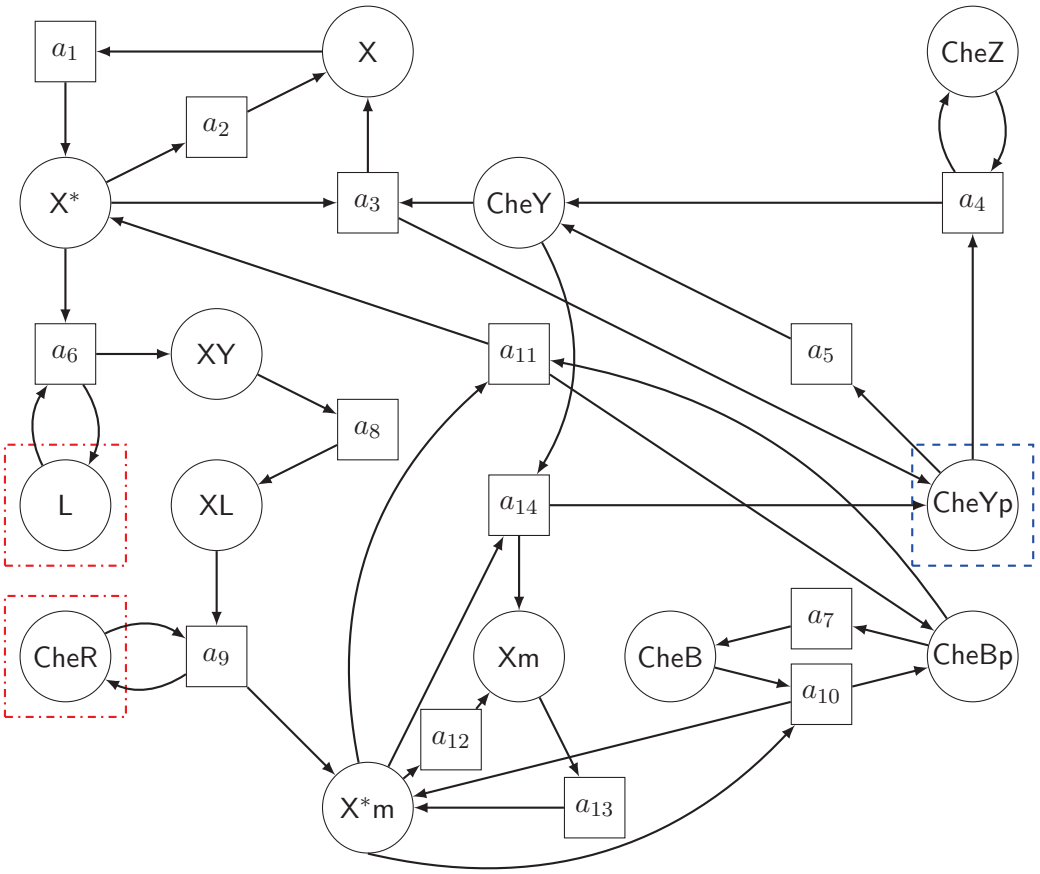


Figure 4.4: The Petri Nets model for the Bacterial Chemotaxis [1]

In order to support our exposition, in Figure 4.4 we report the graphical representation of this process as given in [1] by exploiting the formalism of Continuous Petri Nets. The components of this mechanism on which we put our attention are an attractant L, an enzyme CheR, and a regulatory protein CheYp. The main idea here is that E. Coli senses the concentration level of attractant L, in order to decide whether to keep moving in the current direction or to make a tumble, thus changing direction randomly. Tumbling frequency is reduced if the detected concentration of the attractant has increased with respect to the past. The concentration of L is sensed through receptors on the external membrane. Each receptor is bound to a protein kinase, forming a group denoted X, which passes from inactive state X to active state X* adding a phosphate group to a regulatory protein, denoted CheY, which becomes CheYp. Since CheYp is the main responsible for tumbling, the higher the concentration of CheYp is, the higher the tumbling frequency is. However, the binding of X reduces the probability of activating X. As a consequence, the probability of adding the phosphate to CheY diminishes, and

the tumbling frequency is lowered. Another relevant component of this chemical network is an enzyme, denoted as CheR. This enzyme works oppositely than the attractant L: it increases the activity of X. Indeed, CheR adds at a constant rate a methyl group to the XL complex, which becomes Xm and restarts behaving as X. The methyl group is removed by the enzyme CheB, which is influenced by X that, by adding a phosphoryl group to CheB makes it more active, constituting a negative feedback loop: the higher is the activity of X, the higher is that of CheB, which, in turn, reduces the activity of X. Below we give the system S that represents the network with the initial number of molecules as in Table 4.4.

$$\begin{aligned} S \;=\; & \mathsf{L}[\{a_6?^1!^1\}]_{1000} \;\|\; \mathsf{CheR}[\{a_9?^1!^1\}]_{1000} \;\|\; \mathsf{X}[\{a_1?^1, a_2!^1, a_3!^1\}]_{10} \\ \| \; & \mathsf{X}^*[\{a_2?^1, a_3?^1, a_6?^1, a_1!^1, a_{11}!^1\}]_{10} \;\|\; \mathsf{CheZ}[\{a_4?^1!^1\}]_1 \\ \| \; & \mathsf{Xm}[\{a_{12}!^1, a_{14}!^1\}]_0 \;\|\; \mathsf{XL}[\{a_8!^1\}]_0 \\ \| \; & \mathsf{X}^*\mathsf{m}[\{a_{10}?^1, a_{11}?^1, a_{12}?^1, a_{14}?^1, a_9!^1, a_{13}!^1\}]_1 \\ \| \; & \mathsf{CheY}[\{a_3?^1, a_{14}?^1, a_4!^1, a_5!^1\}]_{10} \;\|\; \mathsf{XY}[\{a_6!^1\}]_0 \\ \| \; & \mathsf{CheB}[\{a_{10}?^1, a_7!^1\}]_2 \;\|\; \mathsf{CheB_P}[\{a_{10}!^1\}]_0 \\ \| \; & \mathsf{CheY_P}[\{a_4?^1, a_5?^1, a_3!^1, a_{14}!^1\}]_1 \end{aligned} \tag{4.1}$$

As reported in [1], two interesting questions are whether the initial concentration of L, or, independently, CheR, impacts on the long-run concentration of CheYp. We can answer to each of those questions by studying the robustness of systems with respect to an input distance $\mathbf{dd}_i$ and an output distance $\mathbf{dd}_o$ defined so that they capture the differences over L, or CheR, and over CheYp, respectively.

4.2.1 Robustness at varying of L

The analysis in [1] concluded that the concentration of output species CheYp *at steady state* does not depend on the initial value of the input species L. In other words, at steady state, an original system and its perturbed versions obtained by varying the initial concentration of input species L do not exhibit any difference in the concentration of output species CheYp. Our analysis will allow us to study the step-by-step distances between the original system and the perturbed one, which is abstracted away in the steady-state analysis.

Let us focus on system S in Equation 4.1, whose initial species concentration is the same as in [1] and is reported in Table 4.4, together with the set $\mathcal{R}$ of reactions and the reaction constants.

We start our analysis by studying the distances between S and a system S_1 that was obtained in [1] by perturbing its initial state: system S_1 starts with $\mathsf{L} = 100000$. In [1] evidence is given that, at steady state, S and S_1 have the same level of CheYp.

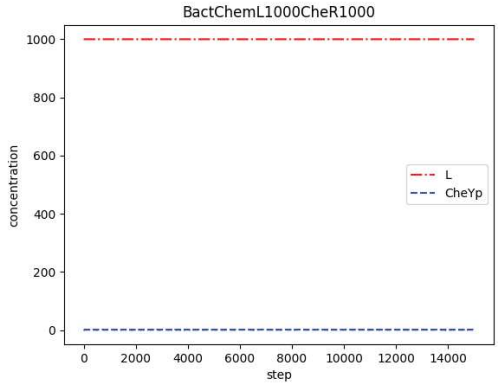
Initial concentration	Reaction constants	Chemical reactions
$\mathsf{L} = 1000\diamondsuit$	$c_{a_1}, c_{a_{13}} = 1.15$	$\mathsf{X} \underset{a_2}{\overset{a_1}{\rightleftharpoons}} \mathsf{X}^*$
$\mathsf{CheR} = 1000\diamondsuit$	$c_{a_2}, c_{a_{12}} = 0.25$	$\mathsf{X}^* + \mathsf{CheY} \xrightarrow{a_3} \mathsf{CheYp} + \mathsf{X}$
$\mathsf{X} = 10$	$c_{a_3} = 0.1$	$\mathsf{CheYp} + \mathsf{CheZ} \xrightarrow{a_4} \mathsf{CheY} + \mathsf{CheZ}$
$\mathsf{X}^* = 10$	$c_{a_4} = 10$	$\mathsf{CheYp} \xrightarrow{a_5} \mathsf{CheY}$
$\mathsf{CheZ} = 1$	$c_{a_5} = 0.002$	$\mathsf{L} + \mathsf{X}^* \xrightarrow{a_6} \mathsf{L} + \mathsf{XY}$
$\mathsf{Xm} = 0$	$c_{a_6}, c_{a_7}, c_{a_{11}} = 1$	$\mathsf{CheBp} \xrightarrow{a_7} \mathsf{CheB}$
$\mathsf{XL} = 0$	$c_{a_8} = 80$	$\mathsf{XY} \xrightarrow{a_8} \mathsf{XL}$
$\mathsf{X}^*\mathsf{m} = 1$	$c_{a_9} = 0.01$	$\mathsf{CheR} + \mathsf{XL} \xrightarrow{a_9} \mathsf{X}^*\mathsf{m} + \mathsf{CheR}$
$\mathsf{CheY} = 10$	$c_{a_{10}} = 0.2$	$\mathsf{X}^*\mathsf{m} + \mathsf{CheB} \xrightarrow{a_{10}} \mathsf{CheBp} + \mathsf{X}^*\mathsf{m}$
$\mathsf{XY} = 0$	$c_{a_{14}} = 0.18$	$\mathsf{X}^*\mathsf{m} + \mathsf{CheBp} \xrightarrow{a_{11}} \mathsf{X}^* + \mathsf{CheBp}$
$\mathsf{CheB} = 2$		$\mathsf{X}^*\mathsf{m} \underset{a_{13}}{\overset{a_{12}}{\rightleftharpoons}} \mathsf{Xm}$
$\mathsf{CheBp} = 0$		$\mathsf{X}^*\mathsf{m} + \mathsf{CheY} \xrightarrow{a_{14}} \mathsf{CheYp} + \mathsf{Xm}$
$\mathsf{CheYp} = 1$		

Table 4.4: The initial concentrations, the reaction constants and the chemical reactions of the chemotaxis phenomenon. The concentration of input species is marked by $\diamondsuit$ and varies to estimate robustness

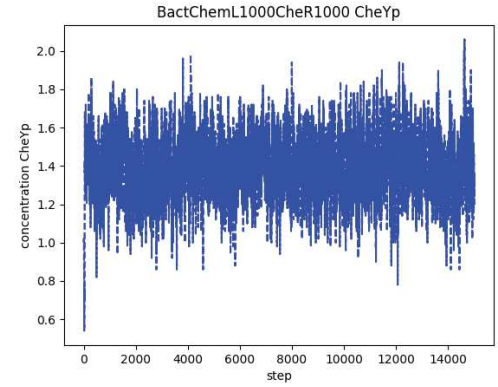
Before analysing robustness, we focus on the simulation of S and S_1. By applying function ESTIMATE in Figure 3.3, we can simulate the evolution of these systems. We have applied the function ESTIMATE with parameters $h = 15000$ and $N = 50$ on the original system S, and $h = 15000$ and $\ell \cdot N = 100$ on the perturbed system S_1. In Figures 4.5a and 4.5c we report the step-by-step average value obtained for all relevant species, namely L and CheYp, for S and S_1, respectively, and in Figures 4.5b and 4.5d we highlight the average value obtained for CheYp.

Then, by applying function DISTANCE in Figure 3.4 we can estimate the evolution distance between S and S_1. We define the input distance $\mathbf{dd}_i$ so that the weight associated with L is 1.0 and the output distance $\mathbf{dd}_o$ so that the weight associated with CheYp is 1.0. In Table 4.5 we report the results obtained by applying DISTANCE with the following parameters: $N = 50$ runs for the original system, $\ell \cdot N = 100$ runs for the perturbed systems, $h = 15000$ steps for each run. In detail, in Table 4.5 we report:

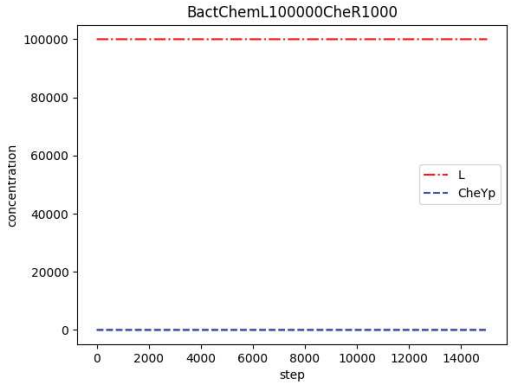
(i) the input distance between S and S_1, which is simply computed by focusing on the initial level of input species in those systems, and the minimal and maximal level that are stored in m and M by DISTANCE;



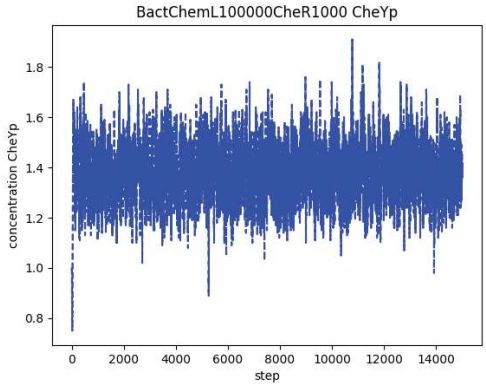
(a) Evolution of L and $\mathsf{CheY_P}$ in S



(b) Evolution of $\mathsf{CheY_P}$ in S



(c) Evolution of L and $\mathsf{CheY_P}$ in S_1



(d) Evolution of $\mathsf{CheY_P}$ in S_1

Figure 4.5: Simulation of system S and its perturbed version S_1

(ii) the maximal output distance $\mathbf{dd}_o$ computed by DISTANCE in interval $[7500, 15000]$;

(iii) a pointer to a figure describing the step-by-step evolution of input and output distance between S and S_1.

Notice that $\mathbf{dd}_o(S, S_1)$ is significantly smaller than the input one.

System	Initial L	$\mathbf{dd}_i$ from S	$\mathbf{E}(\mathbf{dd}_o)_{[7500,15000]}$ from S	Figure showing $\mathbf{dd}_i$ and $\mathbf{dd}_o$ step-by-step
S_1	100000	0.9	0.079091	Figure 4.6

Table 4.5: Input and output distance between S and perturbed version S_1

As in Section 4.1, we can conduct a more systematic analysis for studying how the initial concentration of L in a system S influences the concentration of CheYp in a

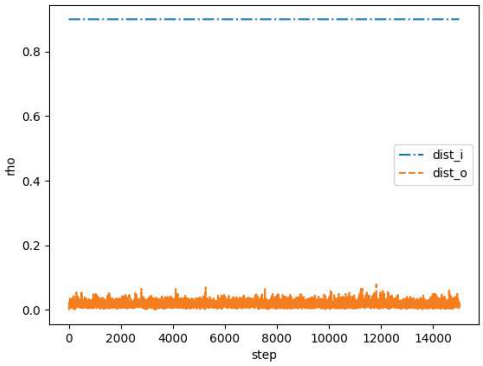


Figure 4.6: Evolution of $\mathbf{dd}_i$ and $\mathbf{dd}_o$ between S and perturbed version S_1

temporal interval I. We fix the maximal input distance η_1 between system S and its perturbed versions and, then, we estimate for which η_2 the system S is $(\mathbf{dd}_i, \mathbf{dd}_o, I_1 = [0,0], I, \eta_1, \eta_2)$-robust. More precisely, in order to estimate η_2, for a suitable n we sample n systems $S_1, \ldots, S_n$ satisfying $\mathbf{E}(\mathbf{dd}_i)_{[0,0]}(S, S_i) \leqslant \eta_1$ and we fix

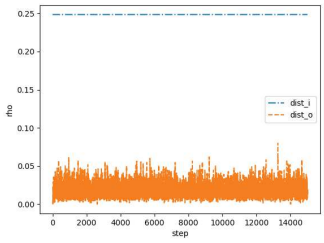
$$\eta_2 = \max_{i=1,\ldots,n} \mathbf{E}(\mathbf{dd}_o)_I(S, S_i).$$

We have considered five different values for η_1: 0.3, 0.4, 0.5, 0.6, 0.7. For each choice for η_1 we have sampled $n = 20$ systems at input distance $\mathbf{dd}_i$ from S bounded by η_1. More precisely, we decided to sample these 20 systems so that they are at a $\mathbf{dd}_i$ distance from S in the interval $(\eta_1 - 0.1, \eta_1]$. Then, we have estimated the η_2 for which S is $(\mathbf{dd}_i, \mathbf{dd}_o, [0,0], I, \eta_1, \eta_2)$-robust, for $I = [7500, 15000]$. In each experiment, we simulated runs consisting of $h = 15000$ reactions, and we considered $N = 50$ runs for the original system S and $\ell \cdot N = 100$ runs for the $n = 20$ perturbed systems. For each of the five choices for η_1, in Table 4.6 we report the value η_2 for which we have estimated the $(\mathbf{dd}_i, \mathbf{dd}_o, [0,0], [7500, 15000], \eta_1, \eta_2)$-robustness, a pointer to the picture showing the step-by-step evolution of $\mathbf{dd}_i$ and $\mathbf{dd}_o$ between S and the perturbed system realizing η_2 and, finally, the initial concentration level of L of that system. Summarizing, at varying of η_1 in $[0.3, 0.7]$ we get values for η_2 that are close to each other and are one order of magnitude lower than η_1. This suggests that S, which was classified as robust in the steady state analysis in [1], has a good level of robustness also in the step-by-step approach. The analysis in [1] allows one to conclude that at steady state the level of CheYp does not depend on the initial level of L, our analysis allows to conclude that if one varies the initial level of L then, step-by-step, the changes in CheYp are limited and smooth.

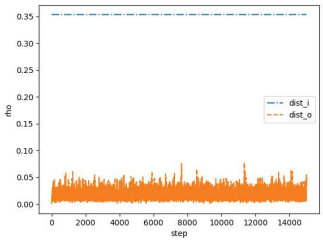
The SPEBNR code used for the analysis can be found at `https://github.com/dmanicardi/spebnr/tree/master/caseStudies/bactChem`. For each estimation – therefore, for each value for η_1 –, the analysis took 125 minutes on a 1.70 GHz i7-1255U, with 16.00 GB RAM.

η_1	η_2	$\mathbf{dd}_i$ and $\mathbf{dd}_o$ distance between S and the system realising η_2	Initial concentration of L of system realising η_2
0.3	0.08	Figure 4.7a	28328
0.4	0.077273	Figure 4.7b	39882
0.5	0.079091	Figure 4.7c	45510
0.6	0.080909	Figure 4.7d	63774
0.7	0.077273	Figure 4.7e	70277

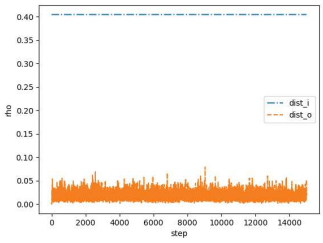
Table 4.6: Bacterial Chemotaxis: robustness at varying of η_1 for input L



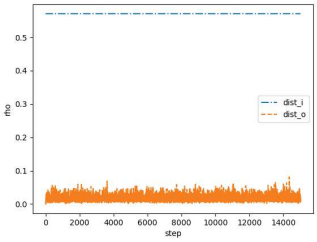
(a) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.3$



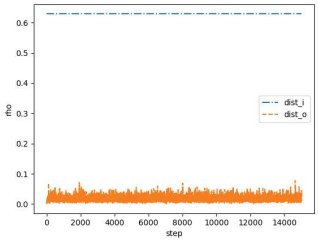
(b) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.4$



(c) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.5$



(d) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.6$



(e) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.7$

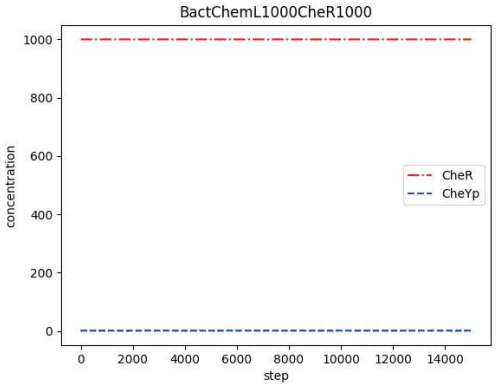
Figure 4.7: Bacterial Chemotaxis: evolution of $\mathbf{dd}_i$ and $\mathbf{dd}_o$ at varying of η_1, for input L

4.2.2 Robustness at varying of CheR

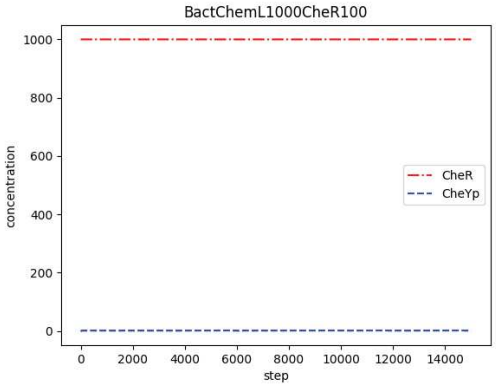
Let us focus again on system S in Equation 4.1. We start our analysis by studying the distances between S and a system S_2 that was obtained in [1] by perturbing its initial state: system S_2 starts with CheR $= 100$. In [1] evidence is given that, at steady state, S and S_2 exhibit a difference on the level of CheYp that leads to conclude that the system S cannot be α-robust for any $\alpha > 0.3$. This is a consequence of the fact that the absolute value of the difference of concentration of CheYp between S and S_2 is around $\frac{\alpha}{2} = 0.15$.

By applying function SIMULATE in Figure 3.2, we can simulate the evolution of these systems. We have applied the function SIMULATE with parameter $h = 15000$ on the original system for $N = 50$ times and for the perturbed one $\ell \cdot N = 100$ times. In

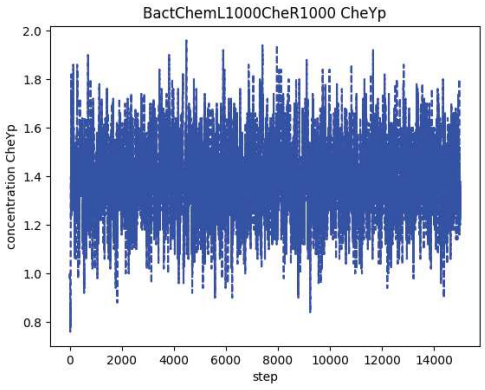
Figure 4.8 we report the step-by-step average value obtained for all relevant species, namely CheR and CheYp, and we highlight the values obtained for CheYp.



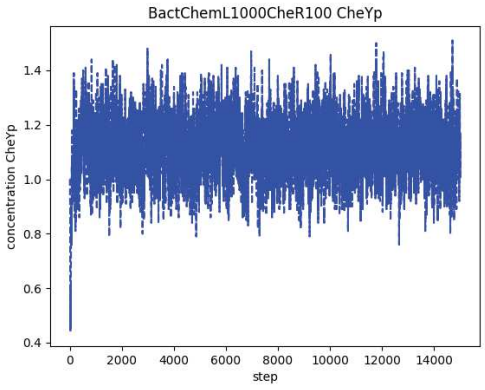
(a) Evolution of CheR, $\mathsf{CheY_P}$ in S



(c) Evolution of CheR, $\mathsf{CheY_P}$ in S_2



(b) Evolution of $\mathsf{CheY_P}$ in S



(d) Evolution of $\mathsf{CheY_P}$ in S_2

Figure 4.8: Simulation of system S and its perturbed version S_2

System	Initial CheR	$\mathbf{dd}_i$ from S	$\mathbf{E}(\mathbf{dd}_o)_{[7500,15000]}$ from S	Figure showing $\mathbf{dd}_i$ and $\mathbf{dd}_o$ step-by-step
S_2	100	0.818182	0.084545	Figure 4.9

Table 4.7: Input and output distance between S and the perturbed system S_2

By applying function DISTANCE in Figure 3.4 we can estimate the evolution distance between S and S_2 We define the input distance $\mathbf{dd}_i$ so that the weight associated with CheR is 1.0 and the output distance $\mathbf{dd}_o$ so that the weight associated with CheYp is 1.0. In Table 4.7 we report the results obtained by applying DISTANCE with the following

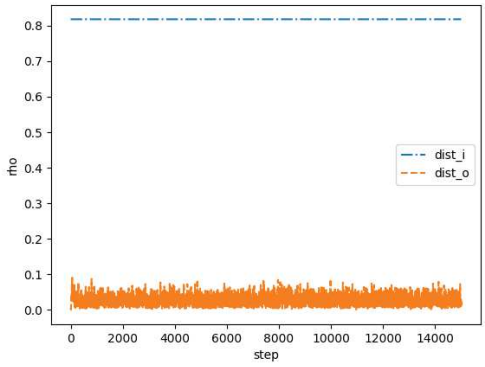


Figure 4.9: Evolution of $\mathbf{dd}_i$ and $\mathbf{dd}_o$ between S and its perturbed version S_2

parameters: $N = 50$ runs for the original system, $\ell \cdot N = 100$ runs for the perturbed systems, $h = 15000$ steps for each run. In particular, in Table 4.7 we report:

(i) the input distance between the perturbed system and S, which is simply computed by focusing on the initial level of input species in S and S_2 and the minimal and maximal level that are stored in m and M by DISTANCE;

(ii) the maximal output distance $\mathbf{dd}_o$ computed by DISTANCE in interval $[7500, 15000]$;

(iii) a pointer to a figure describing the step-by-step evolution of input and output distance between S and the perturbed system.

Notice that $\mathbf{dd}_o(S, S_2)$ is significantly smaller than the input one. We remark that having an output distance of 0.084545 in our analysis is in line with having 0.3-robustness in [1]. Indeed, 0.3-robustness in [1] means that at steady state the level of CheYp in the two systems differs by at most $\frac{0.3}{2} = 0.15$. In our case, an output distance of 0.084545 means that the level of CheYp in the two systems differs, step by step, by at most 0.084545 multiplied by the difference between the max and the min level of CheYp that are stored in m and M by DISTANCE, which is slightly above 2.0.

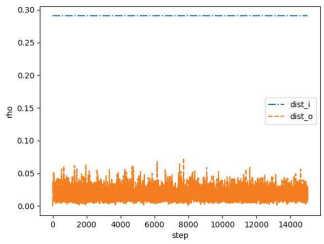
Then, we conducted a more systematic analysis as follows: we fix the maximal input distance η_1 between system S and its perturbed versions and, then, we estimate for which η_2 the system S is $(\mathbf{dd}_i, \mathbf{dd}_o, I_1 = [0, 0], I, \eta_1, \eta_2)$-robust. More precisely, in order to estimate η_2, for a suitable n we sample n systems $S_1, \ldots, S_n$ satisfying $\mathbf{E}(\mathbf{dd}_i)_{[0,0]}(S, S_i) \leqslant \eta_1$ and we fix $\eta_2 = \max_{i=1,\ldots,n} \mathbf{E}(\mathbf{dd}_o)_I(S, S_i)$.

We have considered five different values for η_1: 0.3, 0.4, 0.5, 0.6, 0.7. For each choice for η_1 we have sampled $n = 20$ systems at input distance $\mathbf{dd}_i$ from S bounded by η_1. More precisely, we decided to sample these 20 systems so that they are at a $\mathbf{dd}_i$ distance from S in the interval $(\eta_1 - 0.1, \eta_1]$. Then, we have estimated the η_2 for which S is $(\mathbf{dd}_i, \mathbf{dd}_o, [0, 0], I, \eta_1, \eta_2)$-robust, for $I = [7500, 15000]$. In each experiment, we simulated runs consisting of $h = 15000$ reactions, and we considered $N = 50$ runs for the original

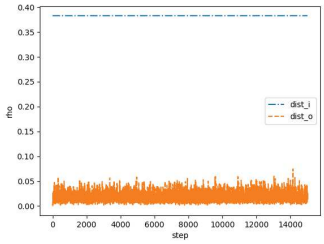
system S and $\ell \cdot N = 100$ runs for the $n = 20$ perturbed systems. For each of the five choices for η_1, in Table 4.8 we report the value η_2 for which we have estimated the $(\mathbf{dd}_i, \mathbf{dd}_o, [0,0], [7500, 15000], \eta_1, \eta_2)$-robustness, a pointer to the picture showing the step-by-step evolution of $\mathbf{dd}_i$ and $\mathbf{dd}_o$ between S and the perturbed system realizing η_2 and, finally, the initial concentration level of CheR of that system. Summarizing, at varying of η_1 in $[0.3, 0.7]$ we get values for η_2 that are close to each other and are one order of magnitude lower than η_1. This suggests that S has a good level of robustness in the step-by-step approach. As already discussed above, our results are in line with those obtained in the steady state analysis conducted in [1].

η_1	η_2	$\mathbf{dd}_i$ and $\mathbf{dd}_o$ distance between S and the system realising η_2	Initial concentration of CheR of system realising η_2
0.3	0.073636	Figure 4.10a	680
0.4	0.074545	Figure 4.10b	579
0.5	0.08	Figure 4.10c	543
0.6	0.084545	Figure 4.10d	345
0.7	0.084545	Figure 4.10e	232

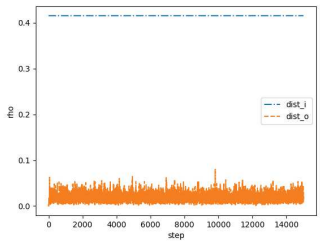
Table 4.8: Bacterial Chemotaxis: robustness at varying of η_1



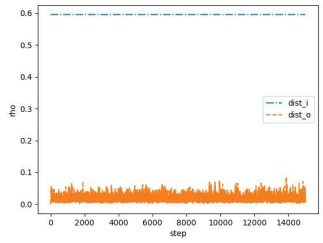
(a) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.3$



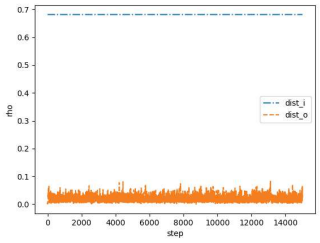
(b) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.4$



(c) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.5$



(d) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.6$



(e) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.7$

Figure 4.10: Bacterial Chemotaxis: evolution of $\mathbf{dd}_i$ and $\mathbf{dd}_o$ at varying of η_1, for input CheR

4.3 The *Enzyme Activity at Saturation* System

In this section, we consider another case study analysed in [1] within the deterministic model, the *Enzyme Activity at Saturation System*. In Figure 4.11 we report the graphical representation from [1]. This is an *abstract* chemical reaction network inspired by *Lotka-Volterra reactions* [65, 66] and by *logistic equation* [67]. Differently with respect to Example 3.1 and Example 4.4, in this network some stoichiometric coefficients are not 1 and that label the corresponding transitions. In this system, an enzyme R produces a molecule X. To ensure mass conservation, the species Z is added to this idealised example to preserve the concentration of R, which is therefore never consumed nor produced, but transformed into Z and back. The production of X is autocatalytic, meaning that the more X are present, the higher the production rate is, but the concentration of enzymes R is limited. Hence, the enzyme activity can easily reach saturation. This system is expected to reach a dynamic equilibrium in which the concentration of X does not depend on its initial concentration, but only on the concentration of R. However, a molecular species P acting as a "predator" for X is added, where X can be consumed and transformed into P, by another autocatalytic reaction. In [1] it is argued that it can be interesting to investigate how the initial concentration of P influences the steady-state concentration of X. The analysis in [1] concluded that the concentration of X at steady state loosely depends on the initial value of P.

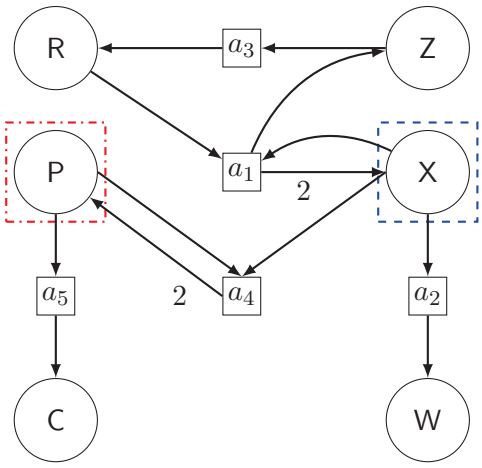


Figure 4.11: The Petri Nets model for the Enzyme Activity at Saturation System [1]

In our approach, we study the robustness of this system with respect to an input distance $\mathbf{dd}_i$ and an output distance $\mathbf{dd}_o$ defined so that they capture the differences over P, and over X, respectively.

Below we give the system S that represents the network with the initial number of molecules as in Table 4.9.

$$\begin{aligned} S \;=\; & \mathsf{P}[\{a_5?^1, a_4?^1!^2\}]_1 \;\;\|\;\; \mathsf{R}[\{a_1?^1, a_3!^1\}]_{1000} \;\;\|\;\; \mathsf{Z}[\{a_1!^1\}]_0 \\ & \|\;\; \mathsf{C}[\{a_5!^1\}]_0 \;\;\|\;\; \mathsf{W}[\{a_2!^1\}]_{10} \;\;\|\;\; \mathsf{X}[\{a_2?^1, a_4?^1, a_1?^1!^2\}]_{30} \end{aligned} \tag{4.2}$$

Initial concentration	Reaction constants	Chemical Reactions
$\mathsf{P} = 1\diamondsuit$	$c_{a_1} = 100$	$\mathsf{R} + \mathsf{X} \xrightarrow{a_1} \mathsf{X} + \mathsf{X} + \mathsf{Z}$
$\mathsf{R} = 1000$	$c_{a_2} = 10$	$\mathsf{X} \xrightarrow{a_2} \mathsf{W}$
$\mathsf{Z} = 0$	$c_{a_3} = 0.5$	$\mathsf{Z} \xrightarrow{a_3} \mathsf{R}$
$\mathsf{C} = 0$	$c_{a_4} = 0.01$	$\mathsf{X} + \mathsf{P} \xrightarrow{a_4} \mathsf{P} + \mathsf{P}$
$\mathsf{W} = 10$	$c_{a_5} = 0.5$	$\mathsf{P} \xrightarrow{a_5} \mathsf{C}$
$\mathsf{X} = 30$		

Table 4.9: The initial concentrations, the reaction constants and the chemical reactions of Enzyme Activity at Saturation system. The concentration of input species is marked by $\diamondsuit$ and varies to estimate robustness

We start our analysis by studying the distances between S and two systems that were obtained in [1] by perturbing its initial state: system S_1 starts with $\mathsf{P} = 1000$, while system S_2 starts with $\mathsf{P} = 20000$. In [1] evidence is given that, at steady state, both S_1 and S_2 have a level of X that loosely differs from that of S: while the initial level of P in S, S_1 and S_2 is 1, 1000 and 20000, respectively, at steady state the level of X in these three systems is always in the interval $[47, 50]$.

Before dealing with the behavioural distance, by applying the function ESTIMATE in Figure 3.3, we can simulate the evolution of these systems. For each system, we have applied the function ESTIMATE with parameters $h = 30000$ and $N = 30$ for S, and $h = 30000$ and $\ell \cdot N = 60$ for S_1 and S_2. In Figures 4.12a, 4.12c, 4.12e we report the step-by-step average value obtained for P and X for S, S_1 and S_2, respectively, then in Figures 4.12b, 4.12d, 4.12f we highlight the values obtained for X. Intuitively, these pictures suggest that the level of X in the three considered systems loosely differ not only at steady state but also step-by-step.

Then, by applying function DISTANCE in Figure 3.4 we can estimate the evolution distance between the original system S and the two perturbed ones, thus confirming such an intuition. We define the input distance $\mathbf{dd}_i$ so that the weight associated with P is 1.0 and the output distance $\mathbf{dd}_o$ so that the weight associated with X is 1.0. In Table 4.10 we report the results obtained by applying DISTANCE with the following parameters: $N = 30$ runs for the original system, $\ell \cdot N = 60$ runs for the perturbed systems, $h = 30000$ steps for each run. In detail, for each perturbed system, in Table 4.10 we report:

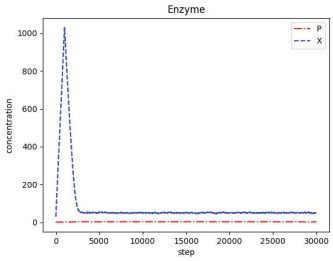
(i) the input distance between the perturbed system and S, which is simply computed by focusing on the initial level of input species in S and the perturbed systems,

and the minimal and maximal levels that are stored in m and M by DISTANCE;

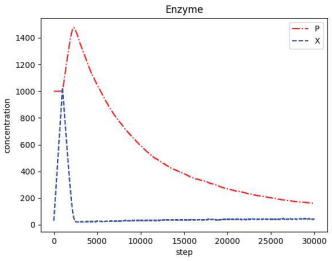
(ii) the maximal output distance $\mathbf{dd}_o$ computed by DISTANCE in interval $[15000, 30000]$;

(iii) a pointer to a figure describing the step-by-step evolution of input and output distance between S and the perturbed system.

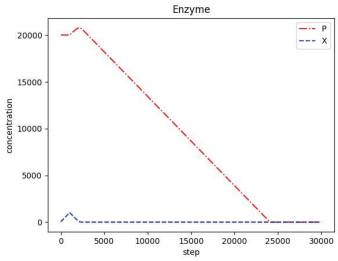
Notice that $\mathbf{dd}_o(S, S_1)$ and $\mathbf{dd}_o(S, S_2)$ are significantly smaller than $\mathbf{dd}_i(S, S_1)$ and $\mathbf{dd}_i(S, S_2)$, respectively.



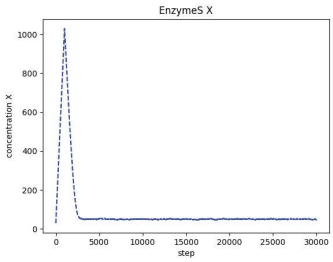
(a) Evolution of P, X in S



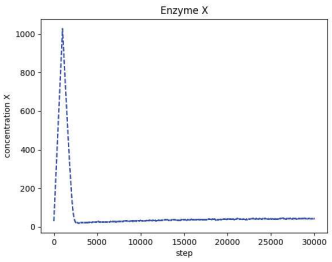
(c) Evolution of P, X in S_1



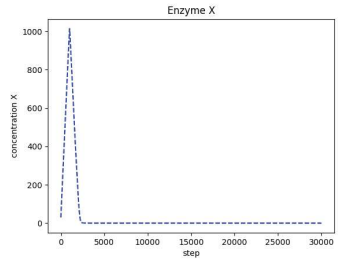
(e) Evolution of P, X in S_2



(b) Evolution of X in S



(d) Evolution of X in S_1



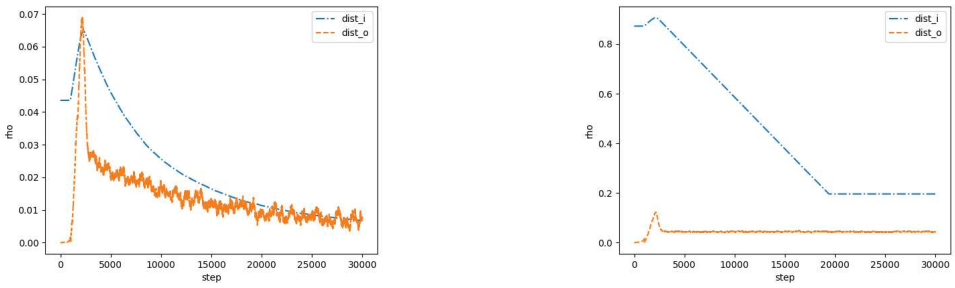
(f) Evolution of X in S_2

Figure 4.12: Simulation of system S and its perturbed versions S_1 and S_2

System	Initial P	$\mathbf{dd}_i$ from S	$\mathbf{E}(\mathbf{dd}_o)_{[15000,30000]}$ from S	Figure showing $\mathbf{dd}_i$ and $\mathbf{dd}_o$ step-by-step
S_1	1000	0.064714	0.014302	Figure 4.13a
S_2	20000	0.872062	0.048517	Figure 4.13b

Table 4.10: Input and output distance between S and its perturbed versions S_1 and S_2

In order to make a systematic analysis, we proceed as follows: we fix the maximal input distance η_1 between system S and its perturbed versions and, then, we estimate for which η_2 the system S is $(\mathbf{dd}_i, \mathbf{dd}_o, [0, 0], I, \eta_1, \eta_2)$-robust. More precisely, in order to estimate η_2, for a suitable n we sample n systems $S_1, \ldots, S_n$ satisfying $\mathbf{E}(\mathbf{dd}_i)_{[0,0]}(S, S_i) \leqslant \eta_1$ and we fix $\eta_2 = \max_{i=1,\ldots,n} \mathbf{E}(\mathbf{dd}_o)_I(S, S_i)$.



(a) $\mathbf{dd}_i(S, S_1)$ and $\mathbf{dd}_o(S, S_1)$, step-by-step

(b) $\mathbf{dd}_i(S, S_2)$ and $\mathbf{dd}_o(S, S_2)$, step-by-step

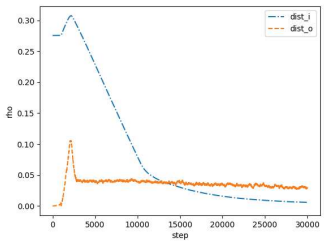
Figure 4.13: Evolution of $\mathbf{dd}_i$ and $\mathbf{dd}_o$ between S and its perturbed versions S_1 and S_2

We have considered five different values for η_1: 0.3, 0.4, 0.5, 0.6, 0.7. For each choice for η_1 we have sampled $n = 20$ systems at input distance $\mathbf{dd}_i$ from S bounded by η_1. More precisely, we decided to sample these 20 systems so that they are at a $\mathbf{dd}_i$ distance from S in the interval $(\eta_1 - 0.1, \eta_1]$. Then, we have estimated the η_2 for which S is $(\mathbf{dd}_i, \mathbf{dd}_o, [0, 0], I, \eta_1, \eta_2)$-robust, for $I = [15000, 30000]$. In each experiment, we simulated runs consisting of $h = 30000$ reactions, and we considered $N = 20$ runs for the original system S and $\ell \cdot N = 40$ runs for the $n = 10$ perturbed systems. For each of the five choices for η_1, in Table 4.11 we report the value η_2 for which we have estimated the $(\mathbf{dd}_i, \mathbf{dd}_o, [0, 0], [15000, 30000], \eta_1, \eta_2)$-robustness, a pointer to the picture showing the step-by-step evolution of $\mathbf{dd}_i$ and $\mathbf{dd}_o$ between S and the perturbed system realizing η_2 and, finally, the initial concentration level of P of that system. Summarising, at varying of η_1 in $[0.3, 0.7]$ we get values for η_2 that are close to each other and are one order of magnitude lower than η_1. This suggests that S, which was classified as robust in the steady state analysis in [1], has a good level of robustness also in the step-by-step approach.

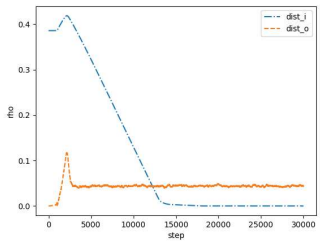
η_1	η_2	$\mathbf{dd}_i$ and $\mathbf{dd}_o$ distance between S and the system realising η_2	Initial concentration of P of system realising η_2
0.3	0.041828	Figure 4.14a	6319
0.4	0.049515	Figure 4.14b	8852
0.5	0.048458	Figure 4.14c	9947
0.6	0.04859	Figure 4.14d	11575
0.7	0.049339	Figure 4.14e	14766

Table 4.11: Enzyme Activity at Saturation System: robustness at varying of η_1

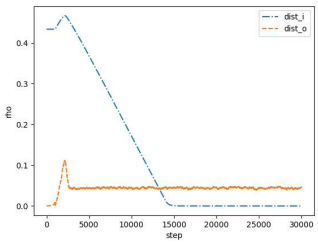
The SPEBNR code used for the analysis can be found at `https://github.com/dmanicardi/spebnr/tree/master/caseStudies/enzyme`. Execution time (for each value for η_1): the analysis took 1,010 minutes on a 1.70 GHz i7-1255U, with 16.00 GB RAM.



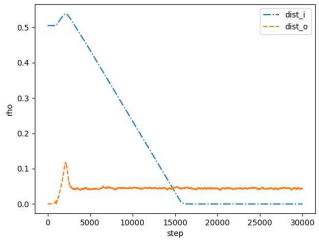
(a) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.3$



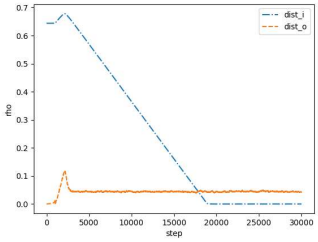
(b) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.4$



(c) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.5$



(d) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.6$



(e) $\mathbf{dd}_i$ and $\mathbf{dd}_o$, $\eta_1 = 0.7$

Figure 4.14: Enzyme Activity at Saturation System: evolution of $\mathbf{dd}_i$ and $\mathbf{dd}_o$ at varying of η_1

5

Modelling Reconfigurable Networks using CospanSpan(Graph)

Historically, automata had a fundamental role both in technical-scientific and philosophic-literary areas. From ancient times, we have examples of mechanical devices simulating some of the aspects of living beings, from movement to speech. Let us recall, for example, the wonderful automata of Vaucanson and Jacquet Droz (the duck, the flute player, the writer, ...) or Von Kempelen's speaking machine, less famous than his fake chess player "the Turk".

At the beginning of the 20th century, the fundamental results of mathematicians like A. Turing, J. Von Neumann, and A. Church, provided a solid mathematical foundation for Automata Theory, giving rise to a paradigmatic shift from automata/machines as purely mechanical-electronic devices towards automata as mathematical entities. This fundamental step, opening the way to Cybernetics and Artificial Intelligence, reconciles their technical, cognitive and symbolic aspects. Indeed, Turing Machines became the standard model to formalise, for the first time in history, the concepts of the sequential machine and of "mechanically" solvable and non-solvable problems. In the same period, starting from McCulloch and Pitts' seminal work on *finite state automata* [16] as a discrete formalisation of neural networks and from Von Neumann's *cellular automata* [30], the idea of *automata* became fundamental.

Nowadays, this idea is pervasive and dominant in Computer Science as well as in our society, and automata have evolved from simple mechanisms to complex structures consisting of *networks of interacting and interconnected entities*, with variable topology, that can even exhibit "cognitive" capabilities, as calculus, speech and visual recognition, learning and self-organization. Indeed, the distinction between artificial and living entities is more and more "fuzzy". So, it is very natural to consider Automata Theory and Biology as very close disciplines, with a long and very fruitful tradition of reciprocal

influences, from robotics to biology-inspired models of computation [68, 69, 70].

Unfortunately, while we could be reasonably satisfied with Turing Machines (and finite state automata) when dealing with discrete, isolated and sequential computation devices, we do not have a general model that could play an analogous role for networks. A variety of models, and different mathematical approaches, have been proposed in literature to formalise "complex" systems or networks as described in previous chapters. The CospanSpan(Graph) model, introduced in [22, 23], has been shown to model, in a very clean way, a variety of phenomena from asynchronous circuits to hierarchy, mobility and coordination [71]. Intuitively, the elements of the model are *Automata with interfaces* described in a specific categorical algebra, that we will introduce in this chapter, as a *discrete* model for neural networks. Various models of automata with the product (of states) have been proposed to represent *interactions* (Zielonka's Asynchronous Automata [17], Arnold's model, Petri Nets [18]). These models are rather natural, but unfortunately, *they are not compositional*, that is they lack a proper algebra. On the contrary, *compositionality* is an essential feature of CospanSpan(Graph). In this approach, explicit operations are provided that combine automata with interfaces and their connectors. Hence, given a syntactic expression, its global semantics can be deduced *only* by the semantics of its constituents, and this is precisely what compositionality requires. In order to fully achieve compositionality, also with respect to parallelism, in CospanSpan(Graph) both the classical (inherently sequential and closed) model of finite state automata and the well-established idea of input/output communication are abandoned for a new paradigm, considering as fundamental the notion of *open systems* with communication interfaces. The operations in the algebra CospanSpan(Graph) can be interpreted in a very natural way as operations on automata with states and transitions, as well as interfaces and conditions. An expression (or even a recursive equation) in this algebra represents a hierarchical, reconfigurable *network of interacting components*.

5.1 CospanSpan(Graph): a Formalism for Automata Networks

In this section, we present the algebra **Span**(**C**) and its dual counterpart **Cospan**(**C**), as introduced by Benabou in [21].

We refer to [72] for basic notions and definitions of category theory.

We will see that CospanSpan(C) becomes a natural syntax for automata networks when $C := Graph$. In fact, the operations of the algebra correspond in a natural way to operations on automata with interfaces (and their connectors), hence providing an algebra that extends Kleene's algebra on "classical" Finite State Automata.

5.1.1 The Algebra of Spans

Span(C): an abstract view

Definition 8 *Given a category* **C** *with finite limits, we define a new category* **Span**(**C**) *by describing its objects and arrows. Objects of* **Span**(**C**) *are the same objects of* **C***;*

arrows of **Span**(**C**) *from A to B (with A, B objects) are* spans, *that is pairs of arrows* $(f : X \to A, g : X \to B)$ *of* **C** *with common domain, often written as in Figure 5.1.*

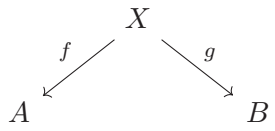


Figure 5.1: Objects of **Span**(**C**)

The composition *of spans* $(f : X \to A, g : X \to B)$ *and* $(h : Y \to B, k : Y \to C)$ *is by* pullback *(restricted product) as in Figure 5.2.*

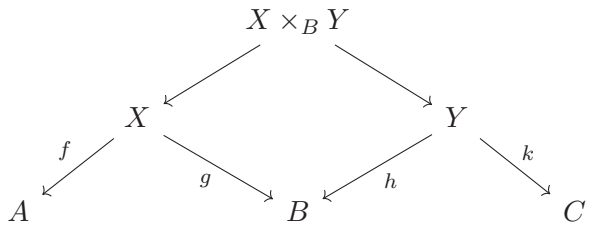


Figure 5.2: The *composition* of spans

A span $(f : X \to A, g : X \to B)$ *will also be written* $X : A \to B$*, and the composition of span will be indicated with the notation* $X \cdot Y : A \longrightarrow C$*. The* identity span *of the object A is* $(1_A, 1_A)$*. The category* **Span**(**C**) *is actually* symmetric monoidal *with the* tensor *product of two spans* $(f : X \to A, g : X \to B)$ *and* $(h : Y \to C, k : Y \to D)$ *being* $(f \times h : X \times Y \to A \times C, g \times k : X \times Y \to B \times D)$*, denoted by* $X \times Y$ *or* $X \otimes Y$*. The identity span of the object A is* $(1_A, 1_A)$*.*

In [22] an informal geometric description (in the style of monoidal category string diagrams) was introduced for the operations in **Span**(**C**). For example, spans $(X \to A \times B, X \to C \times D)$, $(Y \to C \times D, Y \to E)$ and $(Z \to F, X \to G)$ are represented, respectively, by the following three pictures of *components with ports* as in Figure 5.3. Then the composition of the first two spans is pictured as in Figure 5.4. while the tensor

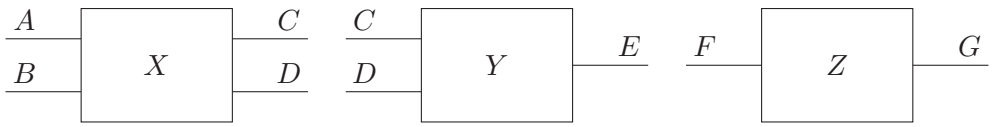


Figure 5.3: Three components with ports

of the first and third span is pictured as in Figure 5.5.

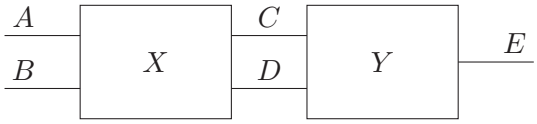


Figure 5.4: The composition of two spans

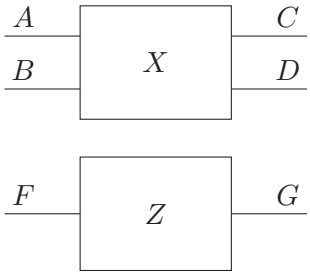


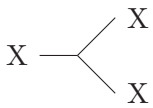
Figure 5.5: The tensor of two spans

In addition, there are constants of the algebra which are pictured as operation on wires which enable the depiction of fanning out of wires and feedback, and hence of general circuit diagrams. We recall some examples of constants in **Span**(**C**) (from [22, 71]):

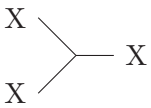
- The identity span $1_X : X \longrightarrow X$ has head X and two legs $1_X, 1_X$. It is denoted by a plain wire as in Figure 5.6.

Figure 5.6: The identity span 1_X

- The span with head X and legs $1_X : X \longrightarrow X$, $\Delta_X : X \longrightarrow X \times X$ is called the *diagonal* of X and is denoted also $\Delta : X \longrightarrow X \times X$, as in Figure 5.7a.
  The span with head X and legs $\Delta_X : X \longrightarrow X \times X$,$1_X : X \longrightarrow X$ is called the *reverse diagonal*, and is denoted $\nabla : X \times X \longrightarrow X$, as in Figure 5.7b.



(a) Diagonal



(b) Reverse diagonal

Figure 5.7: The diagonal and the reverse diagonal of X

- The span with head $X \times Y$ and legs $1_{X\times Y} : X \times Y \longrightarrow X \times Y$, $p_X : X \times Y \longrightarrow X$ is denoted p_X, is called a *projection* and is pictured by the termination of the wire

Y as in Figure 5.8a. There is also a similarly defined *reverse projection* denoted p^*_X as in Figure 5.8b.

(a) Projection (b) Reverse projection

Figure 5.8: The projection and the reverse projection of X

- Consider the terminal object, denoted I. The span with head X and two legs $! : X \longrightarrow I$, $\Delta_X : X \longrightarrow X \times X$ is called η_X. The span with head X and two legs $\Delta_X : X \longrightarrow X \times X$, $! : X \longrightarrow I$ is called ε_X. The two spans are pictured in the Figure 5.9.

(a) η_X (b) ε_X

Figure 5.9: The spans η_X and ε_X

The correspondence between constants and operations on the one hand, and their geometric representations, on the other hand, results in the fact that expressions in the algebra have corresponding circuit or system diagrams. This is clarified in the following example.

Example 2 *Given spans $S : X \longrightarrow X \times X$ and $C : X \longrightarrow I$, the expression*

$$\eta_X \cdot (S \otimes 1_X) \cdot (C \otimes 1_X \otimes 1_X) \cdot (S \otimes 1_X) \cdot (C \otimes 1_X \otimes 1_X) \cdot (S \otimes 1_X) \cdot (C \otimes 1_X \otimes 1_X) \cdot \varepsilon_X$$

has a system diagram (which graphically forms feedback) as in Figure 5.10.

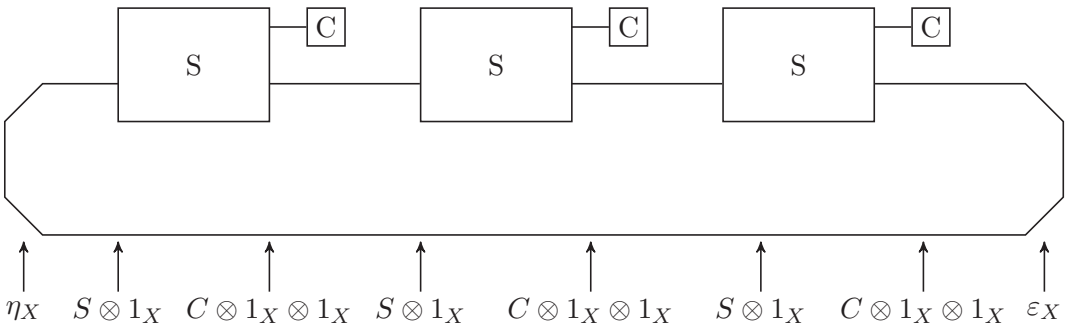


Figure 5.10: The system diagram of the expression in Example 2

As shown in the previous example, η and ε permit a natural feedback operation. More formally, in **Span**(**C**) it is possible to abstractly define the notion of *parallel feedback* [71]:

Definition 9 *Given a span $X : A \times B \longrightarrow C \times B$, we call* (abstract) parallel feedback *of X with respect to B, denoted by $AbPfb_B(X)$, the span denoted by the following algebraic expression is described by Figure 5.11.*

$$(1_A \otimes \eta_B) \cdot (X \otimes 1_B) \cdot (1_C \otimes \varepsilon_B)$$

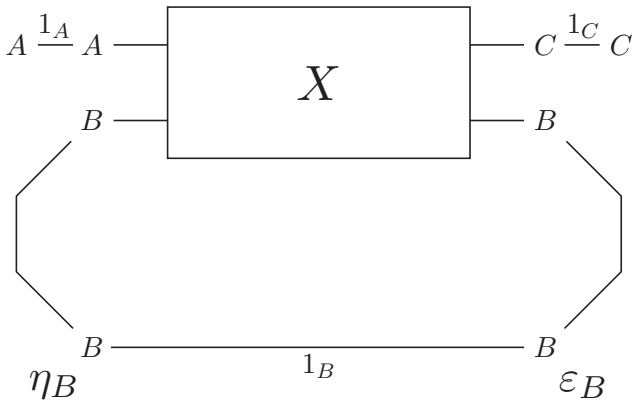


Figure 5.11: The span denoted by the algebraic expression $(1_A \otimes \eta_B) \cdot (X \otimes 1_B) \cdot (1_C \otimes \varepsilon_B)$

Note that the diagrammatic representation of $AbPfb_B(X)$ involves joining the right interface B to the left interface B.

Span(Graph), a parallel algebra of automata

Curiously, **Span**(**C**), when **C** is the category of (finite) directed graphs (**Span**(**Graph**)), provides a very natural mathematical framework for describing the composition of *automata with interfaces* (or communication ports, as in circuit theory).

Consider a span of graphs $(\delta_0 : X \to A, \delta_1 : X \to B)$. The graph X may be considered as the graph of states and transitions of an automaton (with interfaces), and it is called the *head* of the span. The graph A is the graph of states and transitions of the combined left ports and B is the graph of states and transitions of the combined right ports. The graph morphism δ_0 associates to a state and to a transition of the automaton X the corresponding state and transition of the left ports A; the morphism δ_1 does the same for the right ports.

For all the examples of this thesis, the left and right ports have only one state so we tend to ignore that; then δ_0 and δ_1 are double labelling of the transitions of the automaton X by transitions on the left ports and transitions on the right ports. More intuitively, *each transition of the component has an effect on all its interfaces*, maybe the null effect ε.

In the case that the left and right ports have one state, the operations of composition and tensor of spans have a simple description in terms of operations on automata. The tensor of two automata has states being pairs of states, one of each automaton, and has as transitions pairs of transitions between the corresponding pairs of states. The composition of automata has similar states being pairs of states, and transitions being pairs of transitions but *only those pairs of transitions whose labels on the connected ports are the same*. In the following, we will call the span composition *parallel composition with communication* and the tensor *parallel composition without communication*.

Initial and final states

If one wants a parallel algebra of automata which also includes the initial and final states of the automata, a slight variation of the above is possible: instead of the category of graphs take the category $(I+J)\backslash\mathbf{Graph}$ whose objects are graph morphisms $(I+J) \to X$, that consists of two graph morphisms $I \to X$ and $J \to X$, to be thought of as the initial states and final states of X. Then the parallel algebra is $\mathbf{Span}((I+J)\backslash\mathbf{Graph})$.

Quantitative aspects

The description of systems often requires modelling of quantitative aspects, such as time and probability.
In [73] *timed actions* with different durations have been considered. Composition is obtained by considering a linear (w.r.t. transitions) number of extra "internal" states. The intended meaning is that a component that interacts with a "faster" one could be still doing an action (hence being in an internal state) when the other one has completed the transition. We give a simple example.

Example 3 (Two actions which synchronise with an arbitrary duration) *Consider two automata $\mathcal{G}_1$, $\mathcal{G}_2$ and two "non-atomic" actions, one of $\mathcal{G}_1$, one of $\mathcal{G}_2$. The action of $\mathcal{G}_1$ is $\{a/b : 0 \to 1, \varepsilon/\varepsilon : 1 \to 1, d/e : 1 \to 2\}$; the action of $\mathcal{G}_2$ is $\{b/c : 0 \to 1, \varepsilon/\varepsilon : 1 \to 1, e/f : 1 \to 2\}$. In the restricted product $\mathcal{G}_1 \cdot \mathcal{G}_2$ these actions synchronise but with arbitrary duration. A typical behaviour is $(0,0) - a/c \to (1,1) - \varepsilon/\varepsilon \to (1,1) - \varepsilon/\varepsilon \to \cdots - \varepsilon/\varepsilon \to (1,1) - d/f \to (2,2)$.*

Further details about timing in this algebra are presented in Section 5.3 while a probabilistic version is described in [74, 75].

5.1.2 The Algebra of Cospans

Cospan(C): an abstract view

Assume categories $\mathbf{C}$ and $\mathbf{Span}(\mathbf{C})$ (described in Definition 8). There is a dual construction $\mathbf{Cospan}(\mathbf{C})$ for categories $\mathbf{C}$ with finite colimits.

Definition 10 *Given a category $\mathbf{C}$ with finite colimits, we define a new category, the $\mathbf{Cospan}(\mathbf{C})$, by describing its objects and arrows. Objects of $\mathbf{Cospan}(\mathbf{C})$ are the same*

as objects of $\mathbf{C}$*; arrows of* $\mathbf{Cospan}(\mathbf{C})$ *from* A *to* B *are* cospans*, that is, pairs of arrows* $(f : A \to X, g : B \to X)$ *of* $\mathbf{C}$ *with common codomain, also written as in Figure 5.12.*

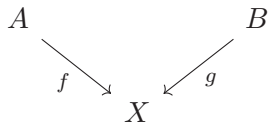


Figure 5.12: Objects of $\mathbf{Cospan}(\mathbf{C})$

The composition *(which is also called* restricted sum*) of* $(f : A \to X, g : B \to X)$ *and* $(h : B \to Y, k : C \to Y)$ *is by* pushout *(glued sum) as in Figure 5.13.*

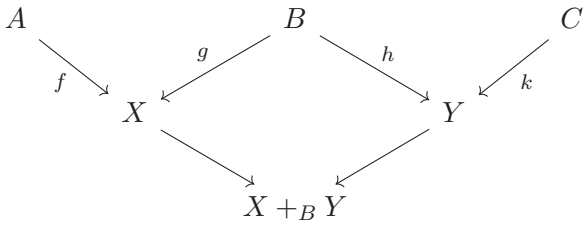


Figure 5.13: The *composition* of cospans

A cospan $(f : A \to X, g : B \to X)$ *will also be written* $X : A \to B$*, and the composition of cospan will be indicated with the notation* $X + Y : A \longrightarrow C$*. The identity cospan of the object* A *is* $(1_A, 1_A)$*.*

Again there are constants of the algebra which are pictured as operation on wires which enable the depiction of the joining of wires and sequential feedback. Graphically, we can present these operations in the Figure 5.14.

Figure 5.14: The operations of the algebra

Cospan(Graph), a sequential algebra of automata

When $\mathbf{C}$ is the category of (finite) directed graphs, $\mathbf{Cospan}(\mathbf{Graph})$ provides a sequential calculus for automata that generalises Kleene's one. Consider a cospan of graphs

$(\gamma_0 : E \to X, \gamma_1 : F \to X)$, the graph X may be considered as the graph of states and transitions of an (unlabelled) automaton. The graph E is the subgraph of *initial* states and transitions and F is the subgraph of *final* states and transitions. In all the examples considered E and F have only states and not transitions. The graph morphisms γ_0 and γ_1 are often inclusion morphisms of the initial and final states in X.

Labelled transitions

If one wants a sequential algebra of automata which have labelled transitions, a slight variation of the above is possible: instead of the category of graphs take the category $\mathbf{Graph}/(A \times B)$ whose objects are graph morphisms $X \to A \times B$, that consists of two graph morphisms $X \to A$ and $X \to B$, to be thought of as the left and right labellings of X. Then the sequential algebra is $\mathbf{Cospan}(\mathbf{Graph}/(A \times B))$.

5.1.3 Cospans and Spans of Graphs

The two algebras recalled above may be combined in a natural way following [76, 71]. Consider four graph morphisms $(\delta_0 : X \to A, \delta_1 : X \to B, \gamma_0 : E \to X, \gamma_1 : F \to X)$. From these we may obtain an arrow in the parallel algebra $\mathbf{Span}((E + F)\backslash\mathbf{Graph})$, and also in the sequential algebra $\mathbf{Cospan}(\mathbf{Graph}/(A \times B))$, and hence we may apply both sequential and parallel operations to such automata, obtaining hierarchical nets of automata with evolving geometry. There is a distributive law of parallel composition over sequential, that will be used in the following example. For some details of this see [23].

Example 4 (Distributed Sort Algorithm) *In [23] has been described in full detail an example of a reconfigurable network, that is a Distributed Sort Algorithm: an atomic sort* A *receives a stream of items to be sorted; if the atomic sort gets full, a new network gets activated in which a divert component* D*, two atomic sorts* A *and the merge component* M *act in parallel. The whole system is the solution of a recursive equation:*

$$S = A + D \cdot (A \times S) \cdot M$$

which is expanded using the distributive laws *(between products and the restricted sums) in the following way:*

$$\begin{aligned} S &= A + D \cdot (A \times S) \cdot M \\ &= A + D \cdot (A \times (A + D \cdot (A \times S) \cdot M)) \cdot M \\ &= A + D \cdot (A \times A) \cdot M + D \cdot (A \times (D \cdot (A \times S) \cdot M)) \cdot M \\ &= \dots \\ &= A + D \cdot (A \times A) \cdot M + D \cdot (A \times (D \cdot (A \times A) \cdot M)) \cdot M + \dots \end{aligned}$$

In this equation, the variables are Cospan/Span automata. It gives rise to a network that can be graphically represented as in Figure 5.15.

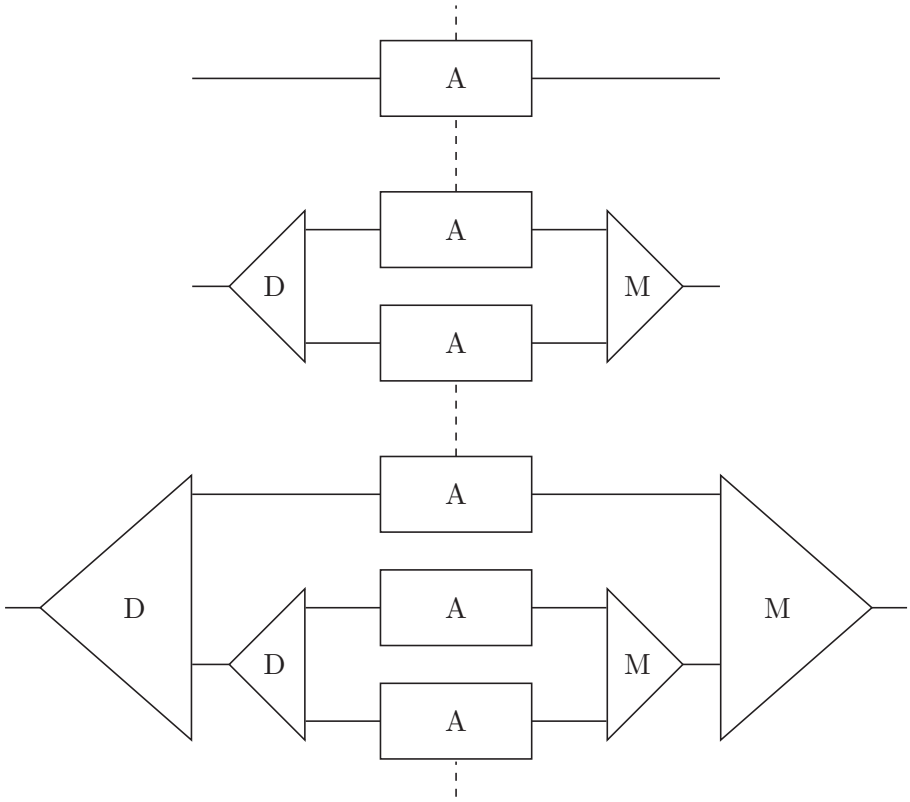


Figure 5.15: The graphical representation of the network for the recursive equation $S = A + D \cdot (A \times S) \cdot M$

A different example of automaton in CospanSpan(Graph) with parallel and sequential feedback is given by *Sofia's Birthday Party*, presented in [23, 71] (see in particular [71] for a global representation of the whole automaton) and in Example 7.

5.2 Compositionality

A distinctive feature of **CospanSpan(Graph)** is *compositionality*. Intuitively, compositionality means that, given a syntactic expression, its global semantics can be *deduced* (using the algebra) *only* by the semantics of its constituents, and this is precisely what the algebraic approach recalled in this chapter provides.

We remark that Kleene's algebra, with corresponding automata operations, gives another example of compositional description, working for classical *sequential* automata. Petri Nets [18] and various models based on a notion of *product of automata*, for example, Zielonka's Asynchronous Automata [17], have been considered in the literature in order to consider the possibility of "parallel communication" among components, but these models are *not* compositional. That is, given, for example, two Petri Nets, P_1 and P_2,

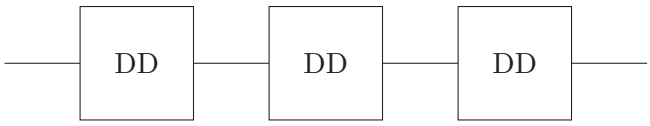


Figure 5.17: The span composition of three DD's

there is no standard way to compose them sequentially or in parallel. In fact, it is common to introduce new places and/or new transitions in an arbitrary way to connect them. The only way to compose them in a standard way is to use a tensor product, putting them aside as they are.

In order to fully achieve compositionality, also with respect to parallelism, the idea behind **CospanSpan(Graph)** is to abandon, both the classical (inherently sequential and closed) model of finite state automata and the well-established idea of input/output communication for a new paradigm, considering as fundamental the notion of *open systems with communication interfaces*. The operations in the algebra **CospanSpan(Graph)** can be interpreted in a very natural way as operations on automata with interfaces. Hence, it is quite easy to build complex systems starting from simpler components.

We start with an example of a decimal counter.

Example 5 (Decimal Counter) *In the following example, in Figure 5.16, the left and right ports are both graphs with one state and two transitions ε and* s *which are displayed on the ports. The automaton has ten states and ten transitions (in a circle through the states). The morphisms δ_0 and δ_1 are indicated by doubly-labelling the transitions of the automaton (so, for example, $\delta_0(0 \to 1) = \varepsilon$ and $\delta_1(0 \to 1) = s$).*

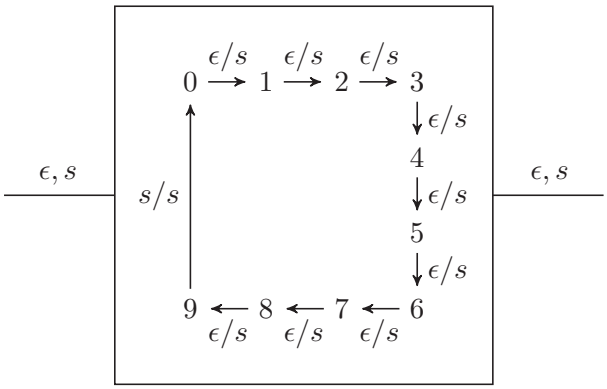


Figure 5.16: The span of a Decimal Counter

We are interested in a variation of this example which we shall call DecimalDigit or more shortly DD *which has the same ports but in addition to the ten transitions above also ten loops, one on each state, each labelled $(\varepsilon/\varepsilon)$.*

Using composition in **Span**(**Graph**) *we can form a new automaton DecimalCounter as the span composition of (for simplicity only) three* DD*'s as in Figure 5.17. or in more*

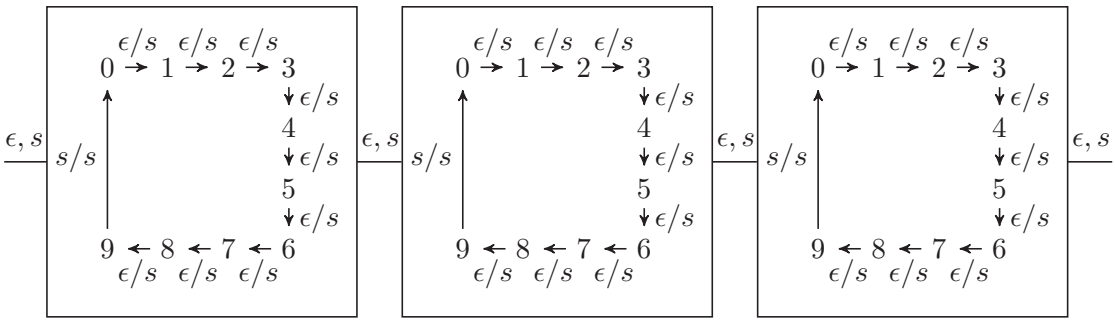


Figure 5.18: The span composition of three DD's (more detail)

detail (though omitting the loops) as in Figure 5.18.

We notice that a state of DecimalCounter is a triple of states, one of each DD*, and a transition of DecimalCounter is a triple of simultaneous transitions with the property that the* labels on the connected ports agree*.*

For example, starting in state $(0, 1, 8)$ *the following is a sequence of transitions of DecimalCounter:*

$$(0, 1, 8) \xrightarrow{(\varepsilon/s)} (0, 1, 9) \xrightarrow{(\varepsilon/\varepsilon)} (0, 1, 9) \xrightarrow{(\varepsilon/s)} (0, 2, 0)$$

(remember the ε/ε *loops on each state of* DD*) so a transition* s *on the right-most port increments the counter. In fact, starting from state* $(0, 0, 0)$*, after* 123 *transitions labelled* s *on the right, the state of the network is* $(1, 2, 3)$*.*

Notice that the transition

$$(9, 9, 9) \xrightarrow{(s/s)} (0, 0, 0)$$

occurs in one step.

We can notice that the previous example explicitly shows the compositionality of **Span**(**Graph**) algebraic model, and it does not seem that the Decimal Counter could be described in such a natural way as in **Span**(**Graph**) by using other formalisms known in the literature.

Example 6 (Producer/Consumer) *In the following example, this well-known problem is described by a producer* P*, a consumer* C *and a buffer* B *of size 3. The producer* P *cannot produce anything when the buffer* B *is full. The consumer* C *cannot consume anything when the buffer* B *is empty. When a producer or a consumer cannot do an action, they need to wait; therefore, they do the null action,* ϵ*.*

Span(**Graph**) *allows us to describe the Producer/Consumer problem with three distinct spans, as in Figure 5.19. It is very easy to modify the system: for instance, we can add another buffer, as in Figure 5.20 or add a second producer, as in Figure 5.21. Another example could be a unique buffer with two producers and two consumers, as in Figure 5.22 where the transitions are:*

$$(1) \quad \epsilon, \epsilon/\epsilon, \epsilon : q_n \to q_n \quad (n = 0, \ldots, 3),$$

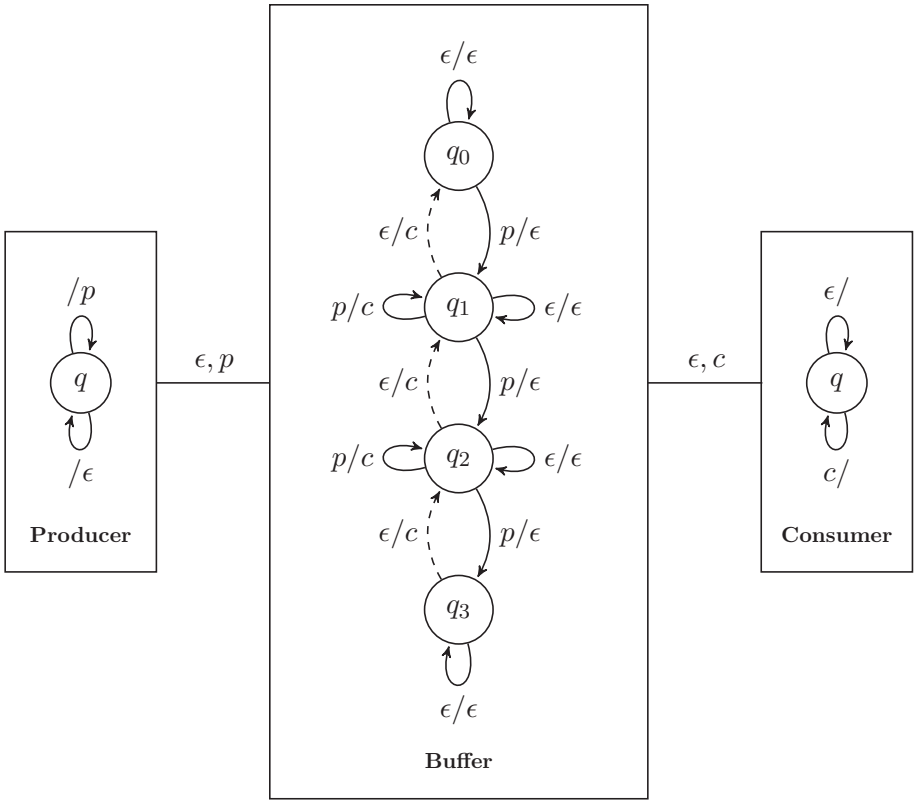


Figure 5.19: The span composition of the Producer/Consumer problem, with one producer, one consumer and one buffer

$$(2) \quad p, \epsilon/\epsilon, \epsilon : q_n \to q_{n+1} \quad or \quad \epsilon, p/\epsilon, \epsilon : q_n \to q_{n+1} \quad (n = 0, \ldots, 2),$$

$$(3) \quad \epsilon, \epsilon/c, \epsilon : q_n \to q_{n\text{-}1} \quad or \quad \epsilon, \epsilon/\epsilon, c : q_n \to q_{n\text{-}1} \quad (n = 1, \ldots, 3),$$

$$(4) \quad p, p/c, c : q_n \to q_n \quad (n = 1, 2),$$

$$(5) \quad p, \epsilon/c, \epsilon : q_n \to q_n \quad or \quad p, \epsilon/\epsilon, c : q_n \to q_n \quad or \quad \epsilon, p/c, \epsilon : q_n \to q_n \quad or \quad \epsilon, p/\epsilon, c : q_n \to q_n \quad (n = 1, 2),$$

$$(6) \quad p, p/\epsilon, \epsilon : q_n \to q_{n+2} \quad (n = 0, 1),$$

$$(7) \quad \epsilon, \epsilon/c, c : q_n \to q_{n\text{-}2} \quad (n = 2, 3).$$

Example 7 (Dining Philosophers) *In the following example, this well-known problem is described by* n *philosophers* P *and* n *forks* F*, as described respectively by Figure 5.23 and Figure 5.24. A philosopher can only* think *or* eat*. When a philosopher eats, needs to use the nearest two forks. For each fork, the philosopher can do nothing*

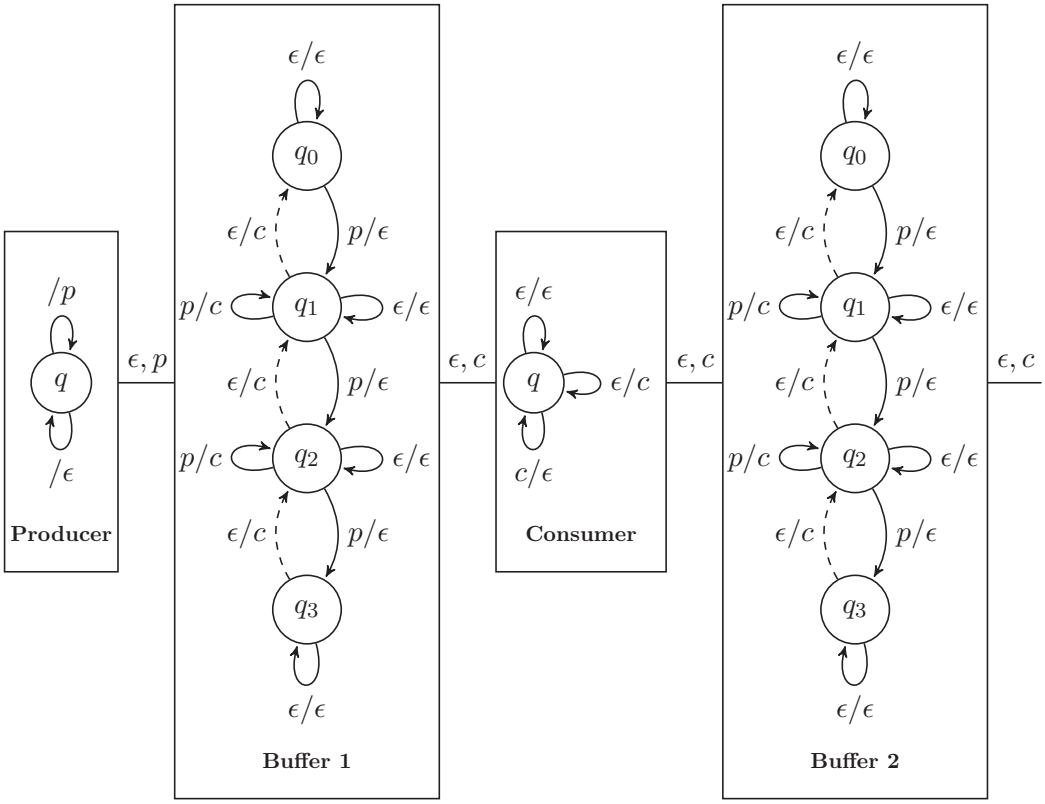


Figure 5.20: The span composition of the Producer/Consumer problem, with two buffers

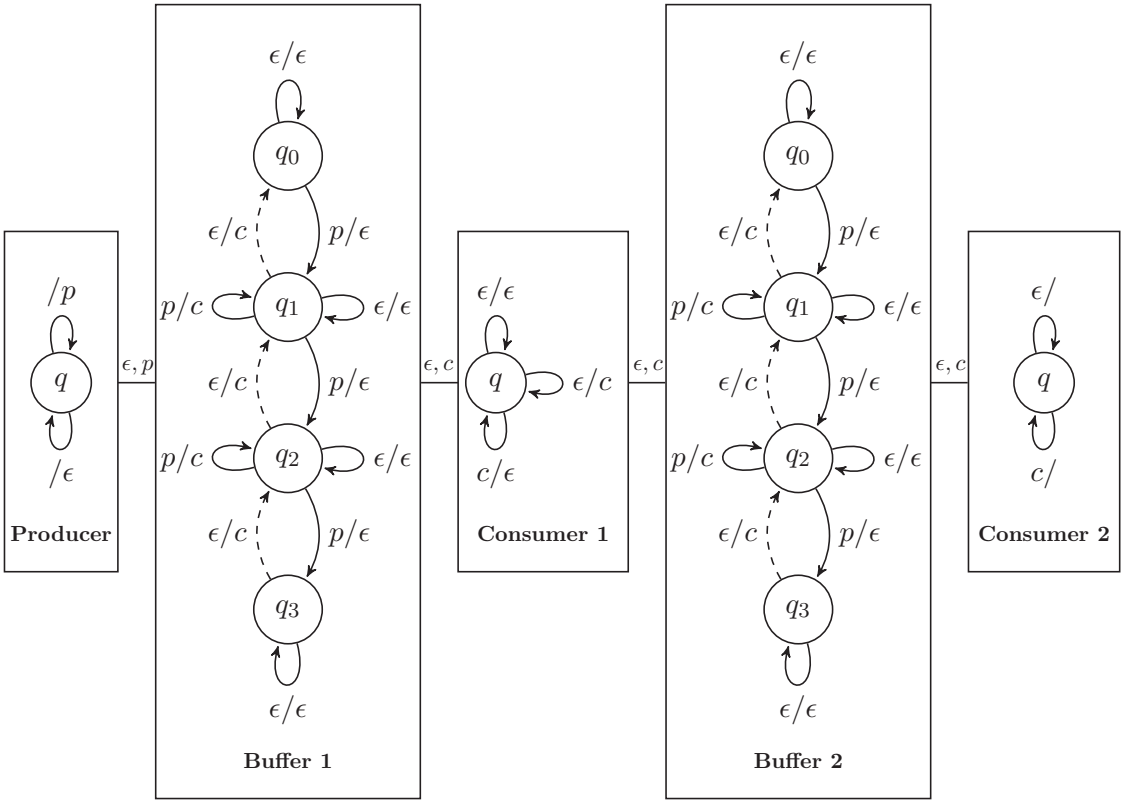


Figure 5.21: The span composition of the Producer/Consumer problem, with two buffers and two consumers

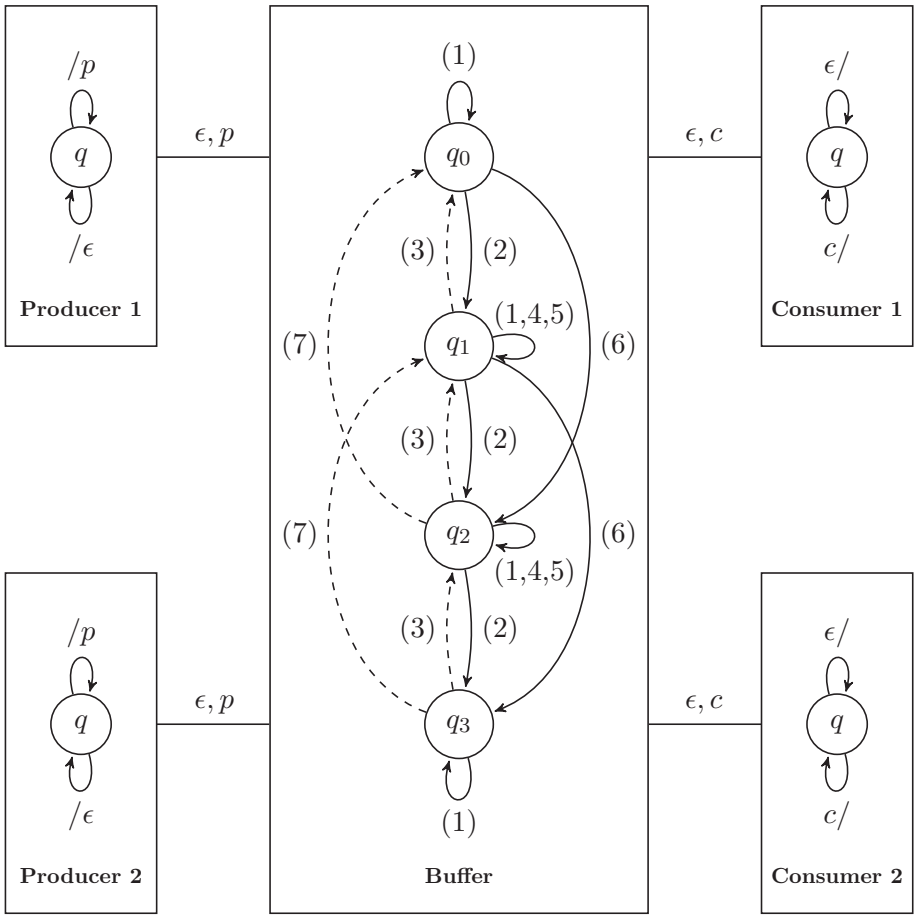


Figure 5.22: The span composition of the Producer/Consumer problem, with a unique buffer, two producers and two consumers

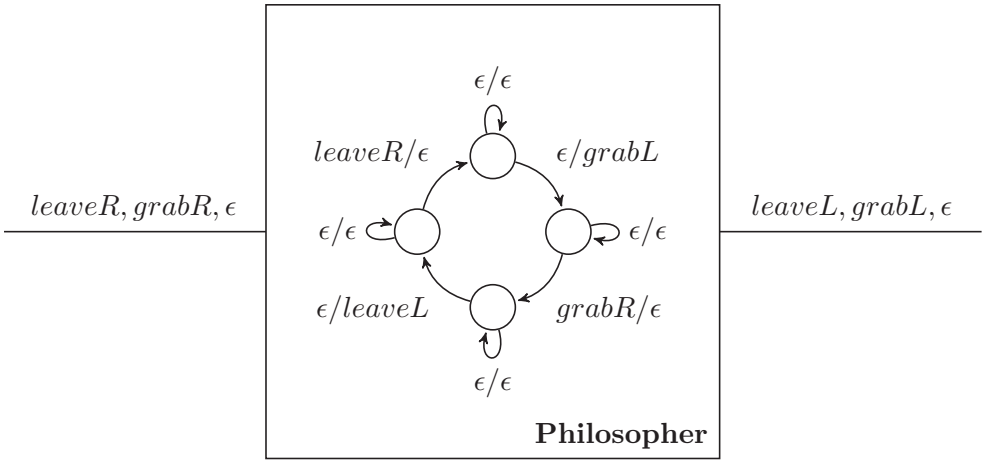


Figure 5.23: The span composition of the Philosopher element of the Dining Philosophers problem

– action: ϵ *which is thinking or eating –, grab the fork or leave it. Therefore, the states of* P *are four: fork on left, both forks, fork on right and no forks. When the philosopher grabs a fork, does not leave until both are grabbed; therefore, there are no intermediate states of communication to the left fork and no forks. The fork has three states: it could be grabbed by the right philosopher, by the left philosopher or by none.*

For instance, the span composition of only three elements of the Dining Philosophers problem is described in Figure 5.25. Notice that all components have all three actions on ϵ*.*

An interesting variation is Sofia's Birthday Party*, presented in [23, 71] (see in particular [71] for a global representation of the whole automaton). There are not philosophers, but* n *children that can move around a table with* k *seats,* $\mathrm{k} \leqslant \mathrm{n}$*. The protocol of each child is the same as a philosopher. In addition, if a child is not holding a fork and has an empty seat on the right, he can change seats.*

Figure 5.26 represents the Sofia's Birthday Party *problem. The basic components of the problem are represented in Figure 5.27. Therefore, Figure 5.28 represents the whole system, where* S *can either be a child (on a seat) or an empty seat.*

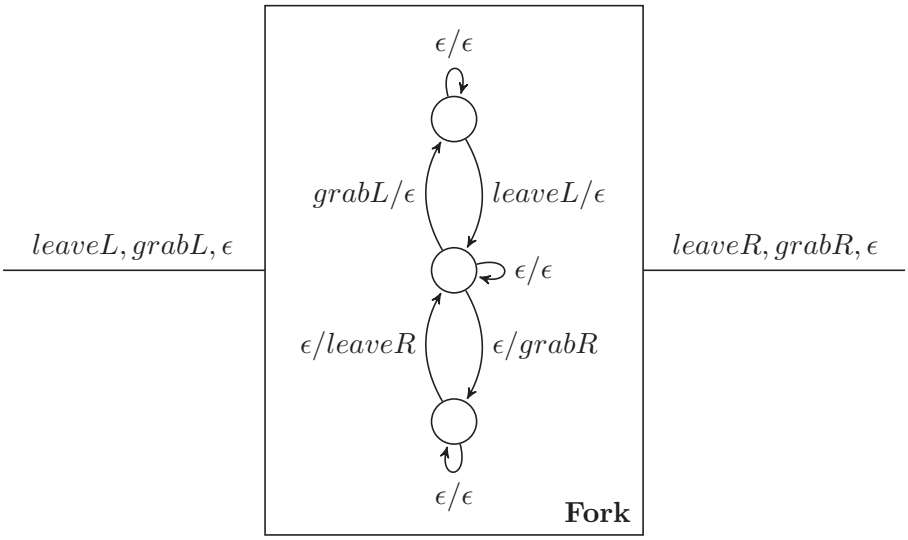


Figure 5.24: The span composition of the Fork element of the Dining Philosophers problem

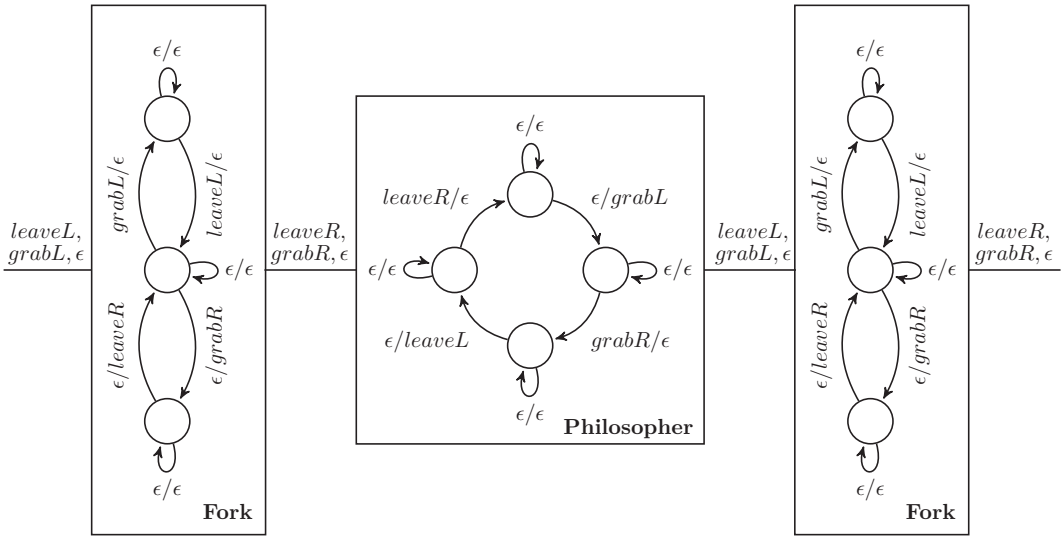
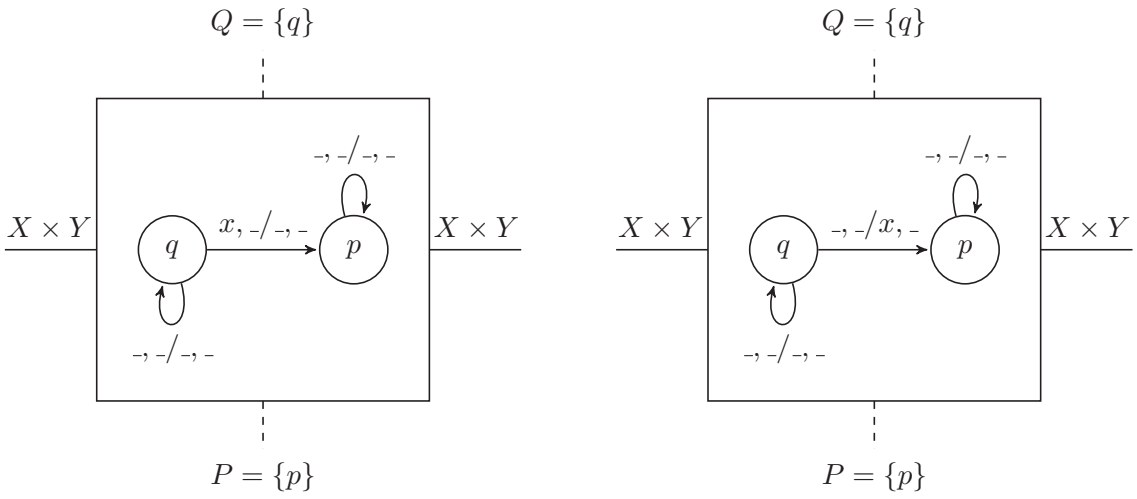


Figure 5.25: The span composition of three elements of the Dining Philosophers problem



(a) L (b) R

Figure 5.26: The basic components of Sofia's Birthday Party

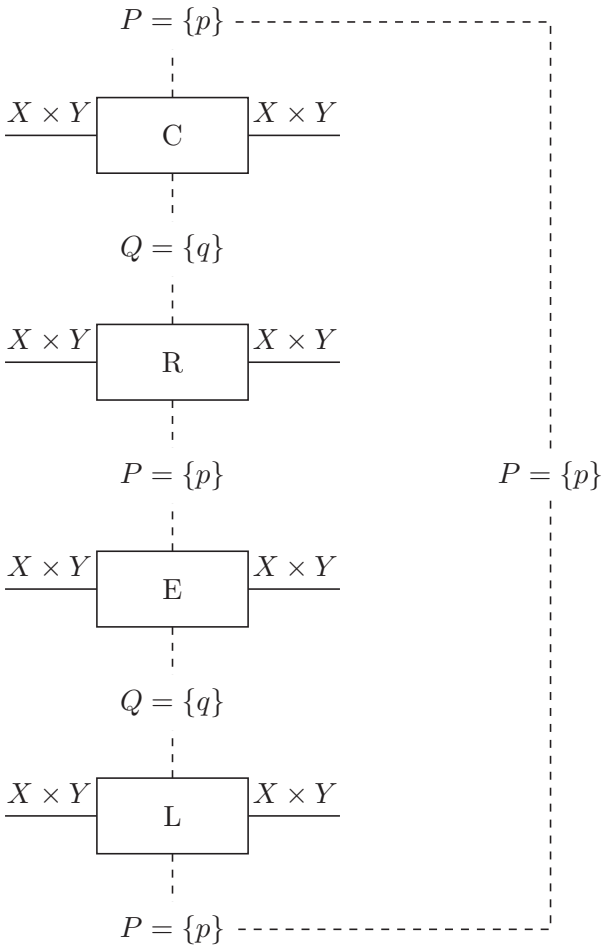


Figure 5.27: The span composition of Sofia's Birthday Party

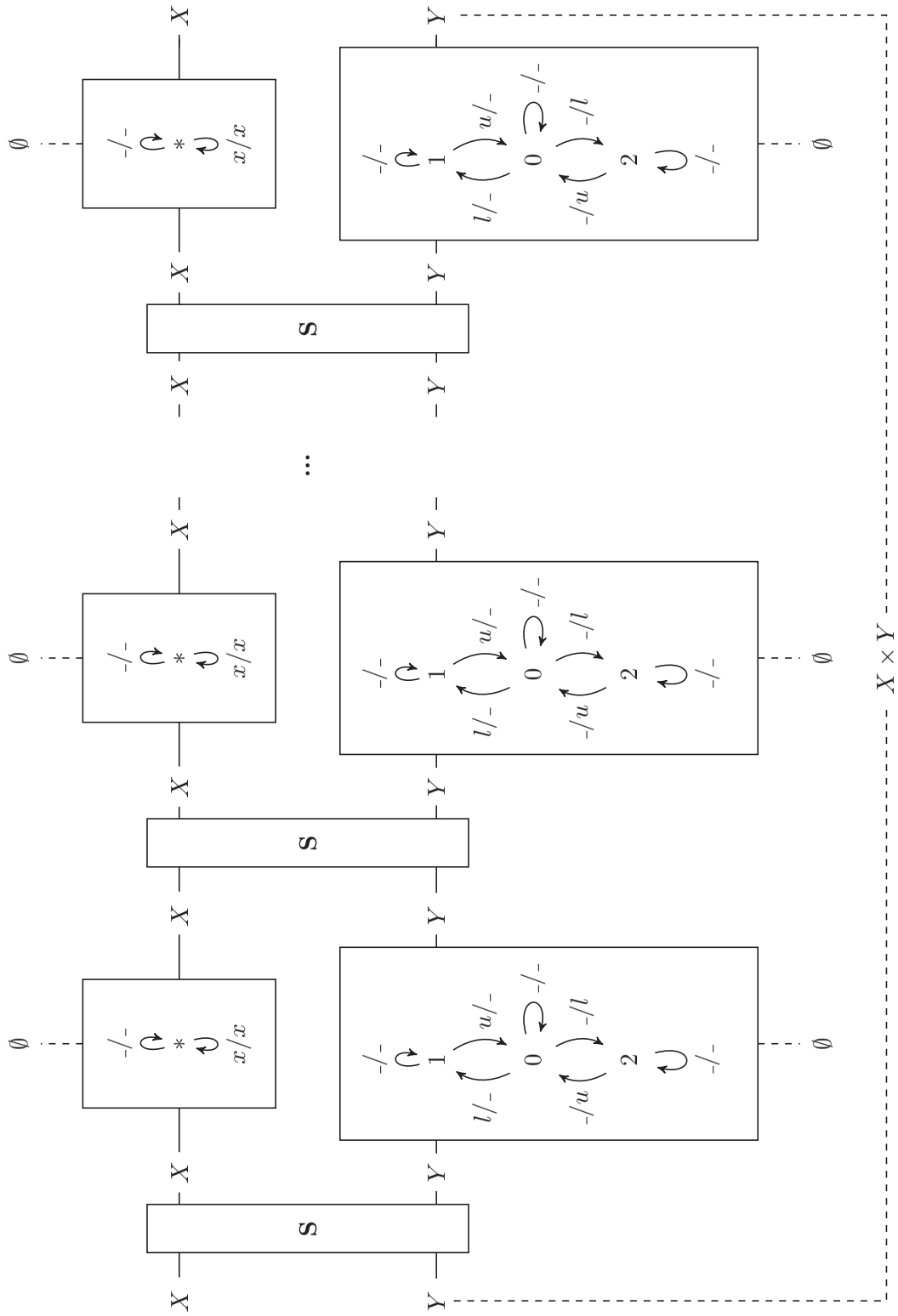


Figure 5.28: The whole span composition of Sofia's Birthday Party

5.3 Timing in CospanSpan(Graph)

In [73], CospanSpan(Graph) algebra is extended in order to model *discrete timed systems.*

As described in Section 5.1, CospanSpan(Graph) algebra can model a variety of distributed systems through spans and cospans of graphs and operations to compose them, respectively interpreted as *automata with states and transitions, interfaces and conditions*, and *parallel with communication.* Through the parallel with communication operation, two automata/span share common interfaces and can synchronise and evolve *simultaneously* by performing actions having the same effects on the common interfaces. Therefore, the movement for all spans of the model is mandatory at each step. *Asynchrony* is achieved by allowing the ϵ move – null, empty action – in all the states. For instance, in Figure 5.18 of the Example 5, a counter can move while the next one remains in the same state by performing the ϵ action.

CospanSpan(Graph) considers that all components have the same duration in time: they are *atomic* actions. Therefore, each transition has a *fixed unit duration*, and all behaviours have a duration which is a multiple of this fixed unit of time. Therefore, all behaviours have a duration which is a multiple of this fixed unit of time. However, in practical examples, the atomic time interval may need to be taken as very brief, and transitions of this duration may not have conceptual significance, being instead only parts of a higher-level action. A top-down description clearly needs to consider actions that have an extended, even variable, duration. The general notion of action therefore is modelled not by a single transition but by a summand in an expression, what we call a non-atomic action, which has a set of initial states and of final states as well as internal states, so the completion of an action does not necessarily require only a unit time interval. In addition, we may consider examples in which two automata synchronise on a non-atomic action - a protocol - rather than on single transitions.

Definition 11 (Discrete-time system and behaviour) *A* ***discrete-time system*** *is an expression of automata given using the parallel and sequential operations, and the constant's identity diagonal as described in Section 5.1. While a* ***behaviour*** *is a path in the automata resulting from evaluating the expression of the discrete-time system. The length of the path is the duration of the behaviour.*

Definition 12 (Non-atomic action) *Suppose* G *is given as an expression of automata involving only sequential operations, and* H *is one of the operands in the expression, then, we call* H *a non-atomic action of* G*. A behaviour of a non-atomic action consists of a path from an in condition to an out condition. The duration of a non-atomic action is hence variable.*

Example 8 (Two actions which synchronise with an arbitrary duration) *Consider two automata $\mathcal{G}_1$, $\mathcal{G}_2$ and two "non-atomic" actions, one of $\mathcal{G}_1$, one of $\mathcal{G}_2$. The action of $\mathcal{G}_1$ is $\{a/b : 0 \to 1, \varepsilon/\varepsilon : 1 \to 1, d/e : 1 \to 2\}$; the action of $\mathcal{G}_2$ is $\{b/c : 0 \to 1, \varepsilon/\varepsilon : 1 \to 1, e/f : 1 \to 2\}$. In the restricted product $\mathcal{G}_1 \cdot \mathcal{G}_2$ these actions synchronise but with arbitrary duration. A typical behaviour is $(0,0) - a/c \to (1,1) - \varepsilon/\varepsilon \to (1,1) - \varepsilon/\varepsilon \to \cdots - \varepsilon/\varepsilon \to (1,1) - d/f \to (2,2)$.*

There is construction of the *desynchronisation* of a system is based on this idea. Consider an expression in the parallel operations of automata representing a system. Supposing that, in each component automaton of the system, every transition is replaced by an action consisting of three transitions analogous to the actions in this example. Therefore, the components of the system do not synchronise with duration 1, but they synchronise with arbitrary duration. For instance, consider the Dining Philosopher problem in Example 7. In the desynchronised version, each of the philosopher and fork actions are divided into begin-action and end-action having the property that the duration of taking a fork may be arbitrarily long. Therefore, while one philosopher is taking a fork another may eat several meals.

Example 9 (A non-atomic action of duration between 1 and 3, which synchronises with one of duration between 2 and 4) *Consider two automata $\mathcal{G}_1$, $\mathcal{G}_2$ and two "non-atomic" actions, one of $\mathcal{G}_1$, one of $\mathcal{G}_2$. The action of $\mathcal{G}_1$ has initial state 0 and final state 1 and transitions are* $\{a_{1,1}/b_{1,1} : 0 \to 1, a_{2,1}/b_{2,1} : 0 \to 2, a_{2,2}/b_{2,2} : 2 \to 1, a_{3,1}/b_{3,1} : 0 \to 3, a_{3,2}/b_{3,2} : 3 \to 4, a_{3,3}/b_{3,3} : 4 \to 1\}$*; the action of $\mathcal{G}_2$ has initial state 0 and final state 1 and transitions are* $\{b_{2,1}/c_{2,1} : 0 \to 2, b_{2,2}/c_{2,2} : 2 \to 1, b_{3,1}/c_{3,1} : 0 \to 3, b_{3,2}/c_{3,2} : 3 \to 4, b_{3,3}/c_{3,3} : 4 \to 1, b_{4,1}/c_{4,1} : 0 \to 5, b_{4,2}/c_{4,2} : 5 \to 6, b_{4,3}/c_{4,3} : 6 \to 7, b_{4,4}/c_{4,4} : 7 \to 1\}$*. In the restricted product $\mathcal{G}_1 \cdot \mathcal{G}_2$ these actions synchronise with duration 2 or 3. The two possible terminating behaviours are* $(0,0) - a_{2,1}/c_{2,1} \to (2,2) - a_{2,2}/c_{2,2} \to (1,1)$*, and* $(0,0) - a_{3,1}/c_{3,1} \to (3,3) - a_{3,2}/c_{3,2} \to (4,4) - a_{3,3}/c_{3,3} \to (1,1)$*.*

Although Examples 8 and 9 are simple, they are suggestive examples of processes synchronising on a protocol.

Example 10 (Counter) *Considering the counter described previously, it provides exactly* an action with a settable duration*, and the input can set the duration of the whole action of the counter. Having finite state automata and counters, control decisions may be made based on timing.*

In the past decade, various automata-based models for describing and reasoning about timing-based systems have been proposed and *timed automata* introduced by Alur and Dill [77] have received enormous attention because they provide a natural way of expressing timing delays of real-time systems. A timed automaton is a finite automaton with a finite set of real-valued clocks. All clocks increase at a uniform rate counting time with respect to a fixed global time frame. The clocks can be reset to 0 (independently of each other) with the transitions of the automata, keeping track of the time elapsed since the last reset. The transitions of the automaton contain constraints on the clock values and a transition may be taken only if the current values of the clocks satisfy the associated constraints. The allowed constraints in the seminal paper of Alur and Dill were Boolean combinations of simpler constraints comparing clock values with time constants.

Comparing timed automata with Timed CospanSpan the first difference to be underlined is the time domain. Here CospanSpang(Graph) uses a discrete time domain

while clocks in timed automata assume real values (or more precisely dense time). Furthermore, it is assumed that each transition has a fixed a priori duration while in timed automata transitions are instantaneous, and time can elapse in a state. This choice limits the expressiveness of our model (for instance we cannot recognise the language of Example 3.16 in [77]). However, we believe that the assumption of discrete time domain is not a strong restriction with respect to a dense time in real systems because almost all physical devices are digital, or approximal by discrete time.

One of the most important advantages of CospanSpan(Graph) is also *compositionality*: which is not considered in Timed Automata. Considering, for instance, Example 5, modelling a one-digit or a three-digit designation is almost similar and easy, thanks to compositionality.

Moreover, the graphical representation in CospanSpan(Graph) is easier to understand than in Timed Automata. Timed Automata only use labels to represent the communications between components. On the other end, CospanSpan(Graph) model the counter – the clock – which became a subcomponent of the system.

6

Modelling Biological Systems and Robustness with CospanSpan(Graph)

In this chapter, **CospanSpan(Graph)** will be used for modelling three examples of biological systems. In Section 6.1 we consider an example of the *Heart System*. Then, in Section 6.2 we study the *Pacemaker Dual Chamber DDD System*. Finally, Section 6.3 deals with the *Lac Operon System*. Moreover, in Section 6.4, we discuss the robustness in CospanSpan(Graph).

6.1 The *Heart* System

In this section we provide a simplified model of a human heart [7]. The heart is a muscular organ and it is the engine of the blood flow. Being a physical system, it could be modelled "globally" using continuous time and differential equations, but, we believe, with difficulty. In our approach, we model it *compositionally* as a discrete dynamical system. We observe that we focus on the regular behaviour of the heart. Atypical behaviour, such as, for instance, cardiac arrhythmia, could be described similarly. Considering that the heart system is quite complex, we concentrate first on formalising its basic components: atria, ventricula, tricuspid/mitral valve, sinoatrial node, atrioventricular node, HIS bundle and the heart's interfaces, such as veins, aorta and pulmonary arteries.

The heart comprehends arterial blood vessels, namely the aorta on the left and pulmonary artery on the right, and two veins. On the right side of the heart, blood continuously enters through the veins and comes out through the pulmonary artery, whereas on the left side, blood enters in the pulmonary vein and comes out from the aorta. Therefore, the heart's interfaces are:

(i) *pulmonary vein* on the left side of the heart;

(ii) *aorta* on the left side of the heart;

(iii) *superior vena cava* and *inferior vena cava* on the right side of the heart;

(iv) *pulmonary artery* on the right side of the heart.

Blood is *continuously* going through the heart system; therefore, it would be incorrect to describe its flux in terms of input/output events.

The heart system furthermore consists of four cavities, paired up in two similar subsystems, left and right. Each subsystem includes an atrium, on the top, and a ventricle, on the bottom. Atria related to ventricles through tricuspid – for the right side – and mitral – for the left side – valve. Atria are like a tank of blood coming from veins. After a beat, the blood continues to enter, increasing the internal pressure until a threshold is reached, that causes the tricuspid/mitral valve to open. The blood begins to drain through the tricuspid/mitral valve and at the same time, the internal pressure in the atria decreases. There is a further pressure peak in the atria due to the electric signal from the sinoatrial node which causes the atrium's contraction and the blood's flow through the tricuspid/mitral valve.

The sinoatrial node generates the normal rhythmic impulse and distributes this impulse to both the atria, in a simultaneous way. The normal range is 60, to 80 beats per minute. The atrioventricular node is primarily responsible for the delay in passing the signal from the atria to the ventricles, whereas the HIS bundle propagates the impulse to the ventricular heart mass.

It should be noted that the operation of the heart, shown below, is a first approximation that can be extended by considering for example other components, like semilunar valves – they are located in the tricuspid/mitral valve and prevent reflux –, time and probability. First, we give a high-level view of the heart in Figure 6.1. Next, we give more detail concerning the heart in Figure 6.2.

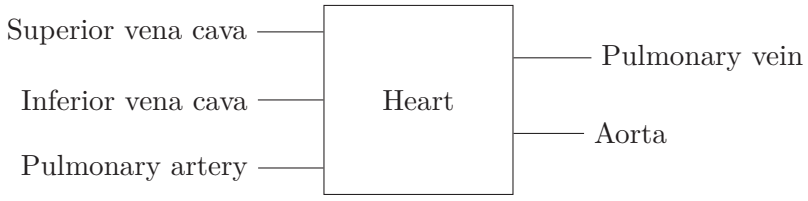


Figure 6.1: Heart

Figure 6.3 shows the functioning of the heart's interfaces – in our simplified example – which are very simple automata with alphabet $\{b\}$. Veins, aorta and pulmonary artery have one state and one transition, labelled with b. The idea is that a heart's interface – veins, aorta or pulmonary artery – is continuously broadcasting blood b in a way which does not impede blood's flow to and from the heart.

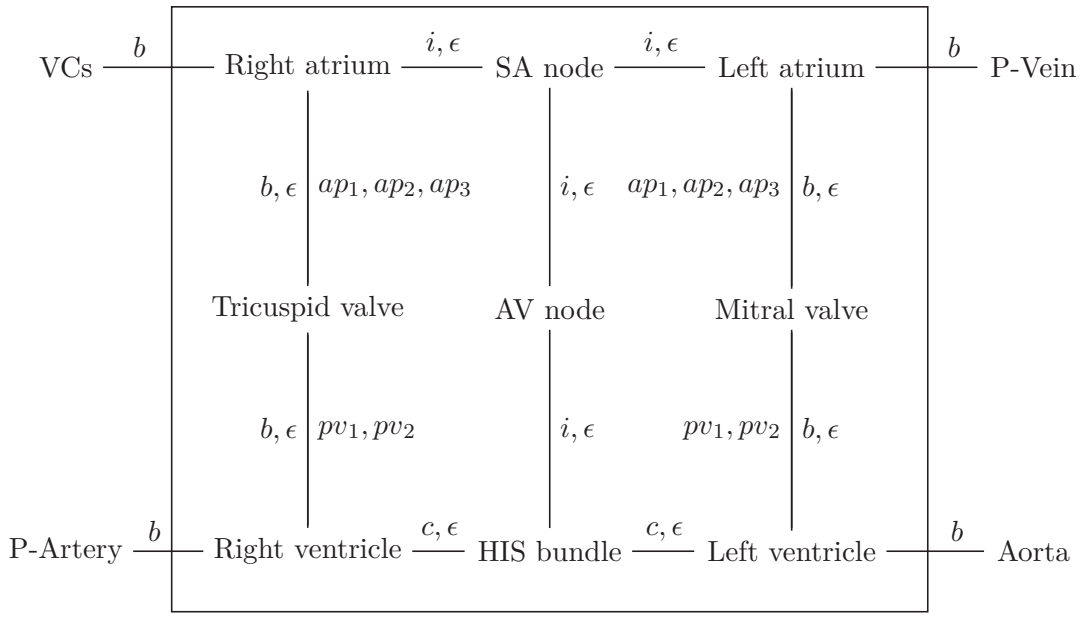


Figure 6.2: Heart Architecture

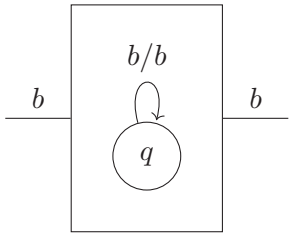


Figure 6.3: Veins, aorta or pulmonary artery

In the following, we adopt the convention to denote interfaces – by abuse of notation – with *Component*=$\{$*labels*$\}$, where *labels* are the proper labels of the interfaces and *Component* corresponds to the automaton to be connected. Now, we describe (an abstraction of) the two atria. For the sake of simplicity, we present only the right part in Figure 6.4: the other side and also the ventricles' modelling is very similar. It has one interface named and labelled by the cartesian product SVC $\times$ IVC = $\{$b$\}$ ("b" stands for "blood") – Superior vena cava $\times$ Inferior vena cava –, another interface named and labelled by the Sinoatrial (SA) node = $\{i, \epsilon\}$ ("i" stands for "impulse"), and two interfaces (linked to the tricuspid valve) $\{ap_1, ap_2, ap_3\}$ (atrial pressure) and $\{b, \epsilon\}$ (blood). The right atrial has states AP_1, AP_2, AP_3 which represent the atrial pressure: AP_1 represents the lowest pressure (when the tricuspid valve is closed), AP_2 relates to the blood flow inside the tricuspid valve and AP_3 is the highest state what you get when the SA node sends an impulse to the right atrium. The transitions are:

$$b, \epsilon / ap_1, \epsilon : AP_1 \rightarrow AP_1,$$

$$b, \epsilon / ap_2, b : AP_2 \to AP_2,$$

$$b, \epsilon / ap_2, b : AP_3 \to AP_2,$$

$$b, \epsilon / ap_2, \epsilon : AP_1 \to AP_2,$$

$$b, i / ap_3, b : AP_2 \to AP_3,$$

$$b, \epsilon / ap_1, \epsilon : AP_2 \to AP_1.$$

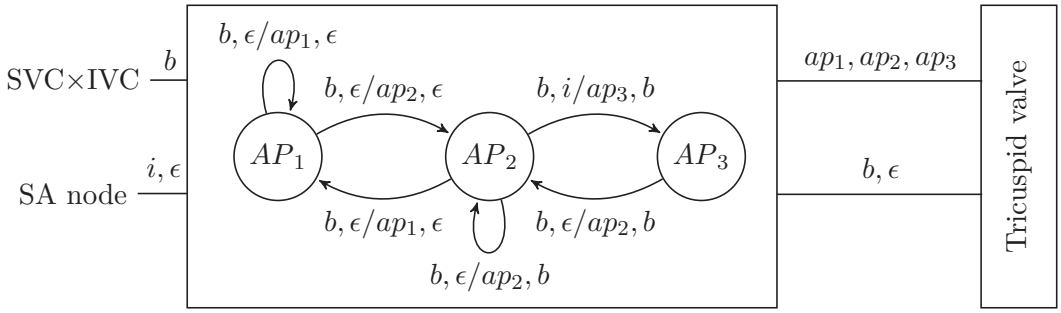


Figure 6.4: Right atrium

Next, we describe the tricuspid/mitral valve, in Figure 6.5. The interfaces are named and labeled by:

$$\text{Atrium}_1 = \{ap_1, ap_2, ap_3\} \text{ (atrial pressure)},$$

$$\text{Atrium}_2 = \{b, \epsilon\} \text{ (blood)},$$

$$\text{Ventricle}_1 = \{vp_1, vp_2\} \text{ (ventricular pressure)},$$

$$\text{Ventricle}_2 = \{b, \epsilon\} \text{ (blood)}.$$

The states are $Open, Close, Peak$, and the transitions are:

$$ap_1, \epsilon / pv_1, \epsilon : Close \to Close,$$

$$ap_2, b / pv_1, b : Open \to Open,$$

$$ap_3, b / pv_1, b : Peak \to Peak,$$

$$ap_2, \epsilon / pv_1, \epsilon : Close \to Open,$$

$$ap_3, b / pv_1, b : Open \to Peak,$$

$$ap_1, \epsilon / pv_2, \epsilon : Peak \to Close.$$

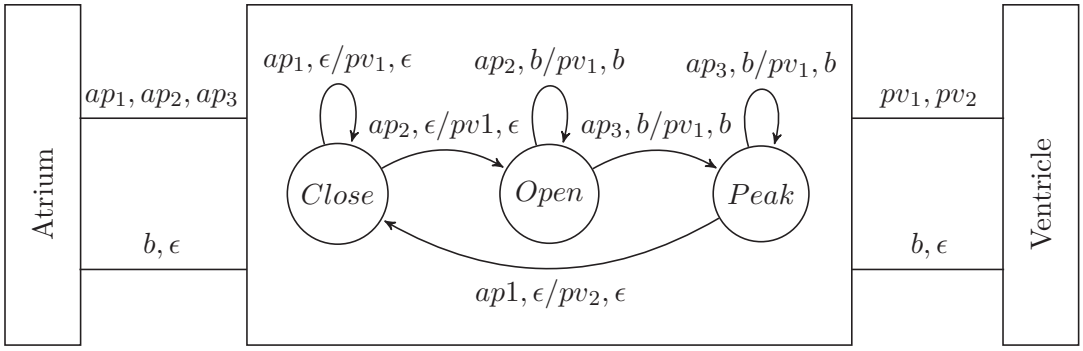


Figure 6.5: Tricuspid valve (Tri-Valve)

The sinoatrial node (SA node) regulates the heart rate through a signal that sends an impulse to the right atrium, the left atrium and the AV node. The normal rate is 60 or 80 beats per minute. We start with 60 bpm (see Figure 6.6), which is one beat per second. We want to temporize the discrete model used in our formalism, so we assume that the impulse i occurs with an atomic action that lasts for one second. Alternatively, we model it with a non-atomic action consisting of [start(impulse), waiting, end(action)], always of one second. It may be that the SA node transmits that impulse, the impulse of contractions, at a higher or lower frequency. By way of example, in Figure 6.7, we have a SA node at a frequency of 60 bpm (so the SA node transmits i at a frequency of 1 bps) or 80 bpm (which is 0.75 bps). The SA node has only three interfaces: the right atrium, the left atrium and the AV node. In Figure 6.6 there are $\{i, \epsilon\}$ (impulse) and there are 60 states and transitions: one transition is $/i, i, i : 0 \to 1$ and other transitions are $/\epsilon, \epsilon, \epsilon : s \to s'$. On the other hand, in Figure 6.7, there are 139 (60+80-1) states which refer to the two different heart rates (60 or 80 bpm), the interfaces are $\{i_1, i_2, \epsilon\}$ and its function is the same as Figure 6.7. It can be easily seen that the SA node component could be modelled in a more elegant way using CospanSpan(Graph), as in the example of Sofia's Birthday Party [71, 74, 75]. Intuitively, when a certain state is reached, the SA node component can activate a similar component working in a different modality, i.e. with a different beat rating.

Figure 6.8 shows the atrioventricular node. AV node sees the impulse generated by the SA node and retransmits it, after a delay, to HIS bundle. AV node sends the pulses at different frequencies, which we can indicate with i_1, i_2, ... It has one interface named and labelled by SA node = $\{i, \epsilon\}$ and one interface named and labelled by HIS bundle = $\{i, \epsilon\}$. The transitions are:

$$\begin{array}{cc} \epsilon/\epsilon : 0 \to 0, & i/\epsilon : 0 \to 1, \\ \epsilon/\epsilon : 1 \to 2, & \epsilon/\epsilon : 2 \to 3, \\ \cdots & \cdots \\ \epsilon/\epsilon : (n-1) \to n, & \epsilon/i : n \to 0. \end{array}$$

Finally, Figure 6.9 shows the HIS bundle. It has one interface named and labelled by AV node = $\{i, \epsilon\}$, and two other interfaces named and labelled by the right ventricle

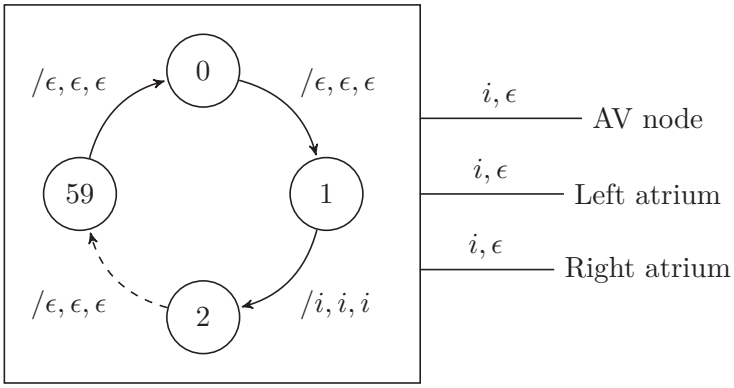


Figure 6.6: SA node (60 bpm)

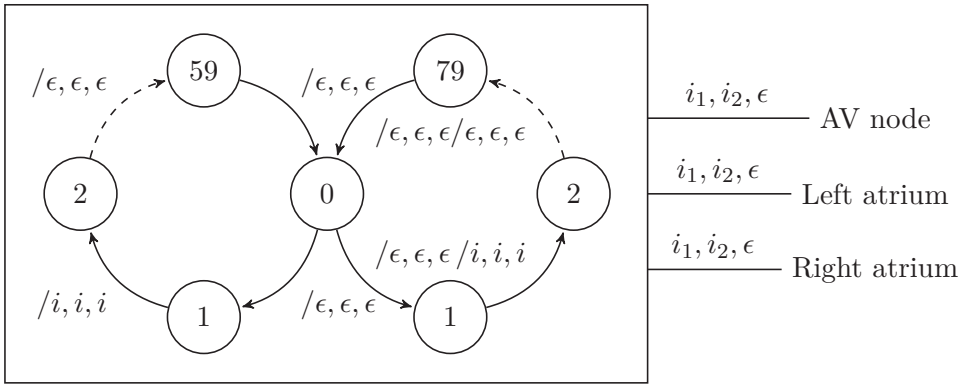


Figure 6.7: SA node (60/80 bpm)

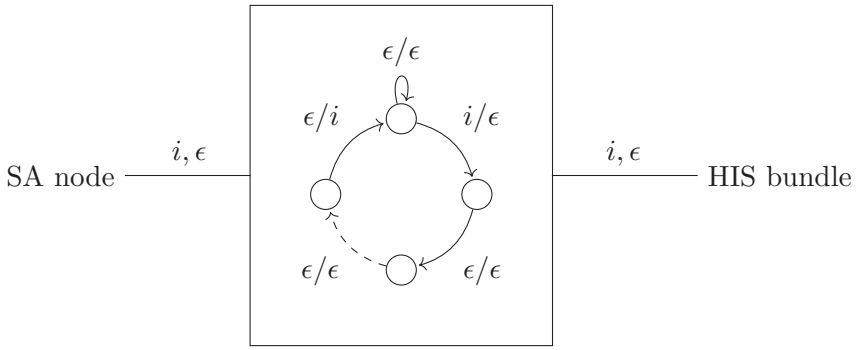


Figure 6.8: AV node

$= \{c, \epsilon\}$ and the left ventricle $= \{c, \epsilon\}$. It has one state and two different transitions: $i/c, c : s \to s$ and $\epsilon/\epsilon, \epsilon : s \to s$.

We give the algebraic representation of the whole automaton of the heart:

$$(\text{SVC} \otimes \text{IVC} \otimes \text{SA} \otimes \text{P-Vein}) \cdot (\text{R-Atrium} \otimes \text{AV} \otimes \text{L-Atrium}) \cdot$$

$$\cdot(\text{Tri-Valve} \otimes \text{HIS} \otimes \text{Mitr-Valve}) \cdot (\text{R-Ventr} \otimes \text{L-Ventr}) \cdot (\text{P-Artery} \otimes \text{Aorta})$$

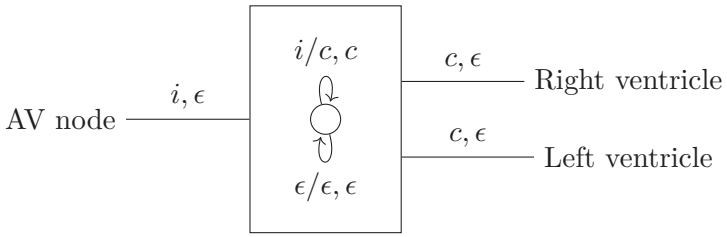


Figure 6.9: HIS bundle

This algebraic expression corresponds to the following geometric representation of the whole automaton in Figure 6.10 (as explained in Section 5.1.1).

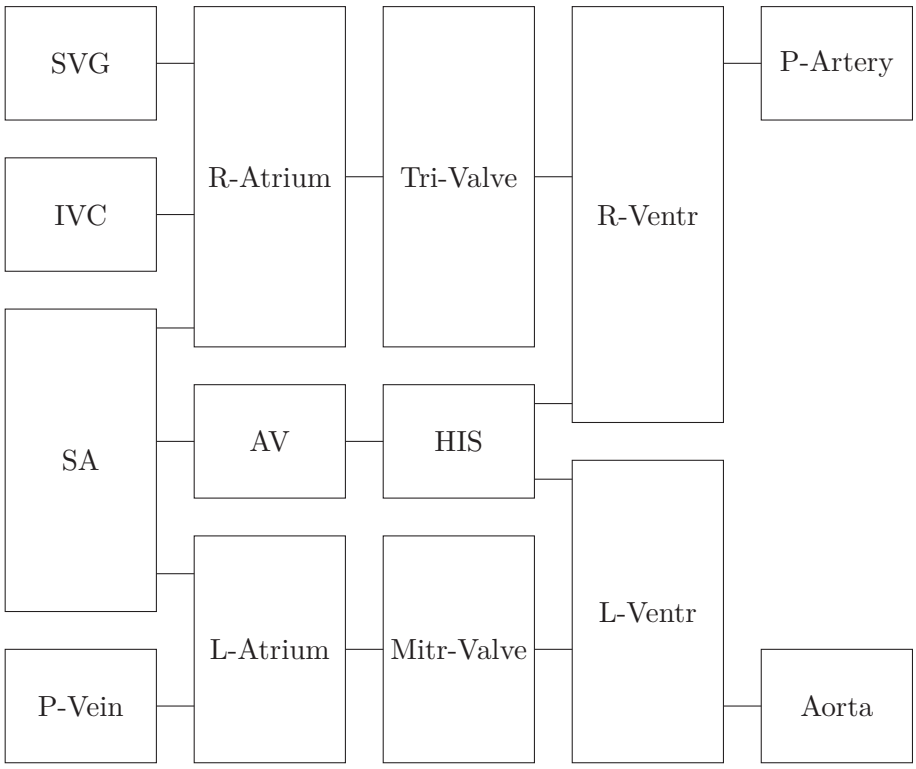


Figure 6.10: Heart automaton

Notice that the previous picture is basically the transpose of the one in Figure 6.2 since we assumed to always represent restricted products horizontally and tensor products vertically.

6.2 The *Pacemaker Dual Chamber DDD* System

In this section, through the use of a *timed* version of **CospanSpan(Graph)** [73], we provide the model of a pacemaker that communicates with the heart system described in the previous Section 6.1. A pacemaker is a system that promptly supplies electrical impulses to the heart in order to maintain an appropriate heart rate and also ventricular-atrial synchrony. Different cardiac problems can occur, hence modern pacemakers are used in different ways. In particular, we model a Dual Chamber Pacemaker DDD − formalised in [7] using UPPAAL − that stimulates both the atrium and the ventricle.

In [7], the Dual Chamber Pacemaker DDD is made up of five components:

(i) **LRI** *Lower Rate Interval*,

(ii) **AVI** *Atrio-Ventricular Interval*,

(iii) **URI** *Upper Rate Interval*,

(iv) **PVARP** *Post Ventricular Atrial Refractory Period* and **PVAB** *Post Ventricular Atrial Blanking*,

(v) **VRP** *Ventricular Refractory Period*.

Notice that the functioning of the modelled pacemaker is a first approximation that can be extended, e.g. also considering a different type of pacemaker or using the probabilistic version of **CospanSpan(Graph)** formalism from [74].

Figure 6.11 shows the pacemaker architecture and the communications with our heart system. Unlike [7], we add two components for broadcasting transmission: S1 and S2 - respectively for AP and VS - which transmits the signal to different other components simultaneously.

The pacemaker shown here was modelled considering the heart in bradycardia or with a regular beat − with 80 beats per minute. The time used in the timed version of **Span(Graph)** is discrete with $\Delta = 10$ milliseconds. The constants TAVI, TLRI, TPVARP, TVRP, TURI and TPVAB − described in [7] − which control the duration of the operations, have the following values: (i) **TAVI**: 150 ms; (ii) **TLRI**: 1000 ms; (iii) **TPVARP**: 100 ms; (iv) **TVRP**: 150 ms; (v) **TURI**: 400 ms; (vi) **TPVAB**: 50 ms.

In the following, we adopt the convention to denote interfaces – by abuse of notation – with $Component$=$\{labels\}$, where labels are the proper labels of the interfaces and $Component$ corresponds to the automaton to be connected. We describe the **LRI**, **AVI**, **URI**, **VRP** and **PVARP** components.

Let us focus on the **LRI** component – described by Figure 6.12 – which has the task of maintaining the heart rate: it models the cycle that defines the longest interval

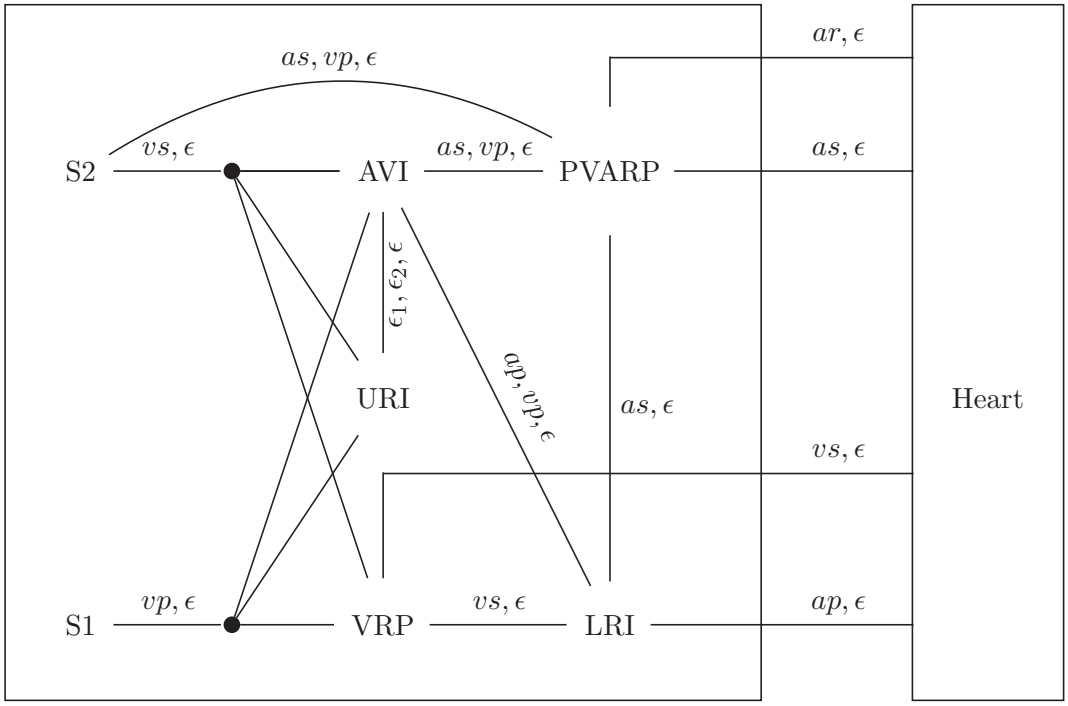


Figure 6.11: Pacemaker Architecture

between two ventricular events by resetting the clock when a ventricular event (VS, VP) is received. Furthermore, if it does not detect any atrial event AS, the component delivers the atrial pacing AP after TLRI-TAVI (850 milliseconds). As shown in Figure 6.12, the time starts with S0; in addition, LRI tracks the time through the passage from one state to the next. LRI has four interfaces: AVI = $\{vp, \epsilon\}$, VRP = $\{vs, \epsilon\}$, PVARP = $\{as, \epsilon\}$ and Atrium = $\{ap, \epsilon\}$. The transitions are:

vp,ϵ/ϵ,ϵ : $0 \rightarrow 1$	ϵ,vs/ϵ,ϵ : $0 \rightarrow 1$
ϵ,ϵ/ϵ,ϵ : $1 \rightarrow 2$	ϵ,ϵ/ϵ,ϵ : $2 \rightarrow 3$
...	...
ϵ,ϵ/ϵ,ϵ : $848 \rightarrow 849$	ϵ,ϵ/ϵ,ϵ : $849 \rightarrow 850$
ϵ,ϵ/as,ϵ : $850 \rightarrow 851_{\mathrm{r}}$	ϵ,ϵ/ϵ,ϵ : $851_{\mathrm{r}} \rightarrow 852$
ϵ,ϵ/ϵ,ϵ : $850 \rightarrow 851_{\mathrm{l}}$	ϵ,ϵ/ϵ,ap : $851_{\mathrm{l}} \rightarrow 852$
ϵ,ϵ/ϵ,ϵ : $852 \rightarrow 853$	ϵ,ϵ/ϵ,ϵ : $853 \rightarrow 0$

Next, Figure 6.14 describes the **AVI** which maintains the appropriate interval between atrial and ventricular activation so it defines the longest interval between an atrial event and a ventricular event. If AVI does not detect any ventricular event (VS) after an atrial event (AS, AP), within TAVI, then AVI delivers a ventricular stimulation (VP). AVI has five interfaces: LRI = $\{ap, \epsilon\}$, PVARP = $\{as, \epsilon\}$, S2 = $\{vs, \epsilon\}$, URI = $\{\epsilon, \epsilon_1, \epsilon_2\}$ and S1 = $\{vp, \epsilon\}$. The transitions are:

$$
\begin{array}{cc}
ap, \epsilon, \epsilon/\epsilon, \epsilon : \text{-1} \rightarrow 0 & \epsilon, as, \epsilon/\epsilon, \epsilon : \text{-1} \rightarrow 0 \\
\epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon : 0 \rightarrow 1 & \epsilon, \epsilon, \epsilon/\epsilon_2, \epsilon : 0 \rightarrow 1 \\
\epsilon, \epsilon, \epsilon/\epsilon_2, \epsilon : 1 \rightarrow 2 & \epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon : 1 \rightarrow 2 \\
\dots & \dots \\
\epsilon, \epsilon, \epsilon/\epsilon_2, \epsilon : 149 \rightarrow 150 & \epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon : 149 \rightarrow 150 \\
\epsilon, \epsilon, vs/\epsilon, \epsilon : 150 \rightarrow \text{-1} & \epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon : 150 \rightarrow 151 \\
\epsilon, \epsilon, vs/\epsilon, \epsilon : 151 \rightarrow \text{-1} & \epsilon, \epsilon, \epsilon/\epsilon_2, \epsilon : 151 \rightarrow \text{-1} \\
\epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon : 151 \rightarrow 151 & \epsilon, \epsilon, \epsilon/\epsilon, \epsilon : \text{-1} \rightarrow \text{-1}
\end{array}
$$

The **URI** – described by Figure 6.13 – prevents the pacemaker from simulating the ventricle too quickly thanks to a global clock used to track the time after a ventricular event (VS,VP). URI allows AVI to supply VP only when the global clock is TURI. URI has three interfaces: AVI = $\{\epsilon, \epsilon_1, \epsilon_2\}$, S1 = $\{vp, \epsilon\}$ and S2 = $\{vs, \epsilon\}$ The transitions are:

$$
\begin{array}{cc}
\epsilon_1/\epsilon,\text{vs} : 0 \rightarrow 1 & \epsilon/\text{vp},\epsilon : 1 \rightarrow 0 \\
\epsilon_1/\epsilon,\epsilon : 1 \rightarrow 2 & \epsilon_1/\epsilon,\epsilon : 2 \rightarrow 3 \\
\dots & \dots \\
\epsilon_1/\epsilon,\epsilon : 398 \rightarrow 399 & \epsilon_2/\epsilon,\epsilon : 399 \rightarrow 0
\end{array}
$$

Figure 6.16 shows the **PVARP**. After an atrial event AS, a ventricular event VS or VP must occur; therefore, when VS or VP is detected, in a small latency time (TPVAB) atrial events are ignored. After TPVAB, there is a second latency time (TPVARP), in which a signal is transmitted − AR! − outside the pacemaker. Finally, after TPVARP, the atrial event can be detected and sent to the LRI component. PVARP has five interfaces: Heart = $\{ar!, \epsilon\}$, LRI = $\{as, \epsilon\}$, AVI = $\{as, vp, \epsilon\}$, S2 = $\{as, vp, \epsilon\}$ and Atrium = $\{as!, \epsilon\}$. The transitions are:

$$
\begin{array}{cc}
\epsilon, \epsilon, \epsilon/\epsilon, \epsilon : 0 \rightarrow 0 & \epsilon, \epsilon, vp/\epsilon, \epsilon : 0 \rightarrow 1 \\
\epsilon, \epsilon, \epsilon/vs, \epsilon : 0 \rightarrow 1 & \epsilon, \epsilon, \epsilon/as, \epsilon : 1 \rightarrow 2 \\
\dots & \dots \\
ar!, \epsilon, \epsilon/\epsilon, as! : 50 \rightarrow 51 & \epsilon, \epsilon, \epsilon/\epsilon, \epsilon : 50 \rightarrow 51 \\
\dots & \dots \\
\epsilon, \epsilon, \epsilon/\epsilon_1, \epsilon : 51 \rightarrow n & \epsilon, as, as/\epsilon, as! : n \rightarrow 0
\end{array}
$$

Finally, Figure 6.15 shows **VRP** which monitors all ventricular events (VP, VS) and filters early events in the ventricular canal that could cause an inappropriate pacemaker behaviour. VRP has three interfaces: one named and labelled by our heart system = $\{vs!, \epsilon\}$, and two other interfaces named and labelled S1 = $\{vp, \epsilon\}$ and S2 = $\{vs, \epsilon\}$. The transitions are:

$$
\begin{array}{cc}
\epsilon/vp, \epsilon : \text{IDLE} \rightarrow 0 & \epsilon/\epsilon, \epsilon : 0 \rightarrow 1 \\
\dots & \dots \\
\epsilon/\epsilon, \epsilon : 149 \rightarrow \text{IDLE} & \epsilon/vp, \epsilon : \text{IDLE} \rightarrow \text{s} \\
\epsilon/\epsilon, vs : \text{IDLE} \rightarrow \text{s} & vs!/\epsilon, \epsilon : \text{s} \rightarrow 0
\end{array}
$$

Summarizing, **CospanSpan(Graph)** can model, clearly and simply, a complex system like the heart-pacemaker system.

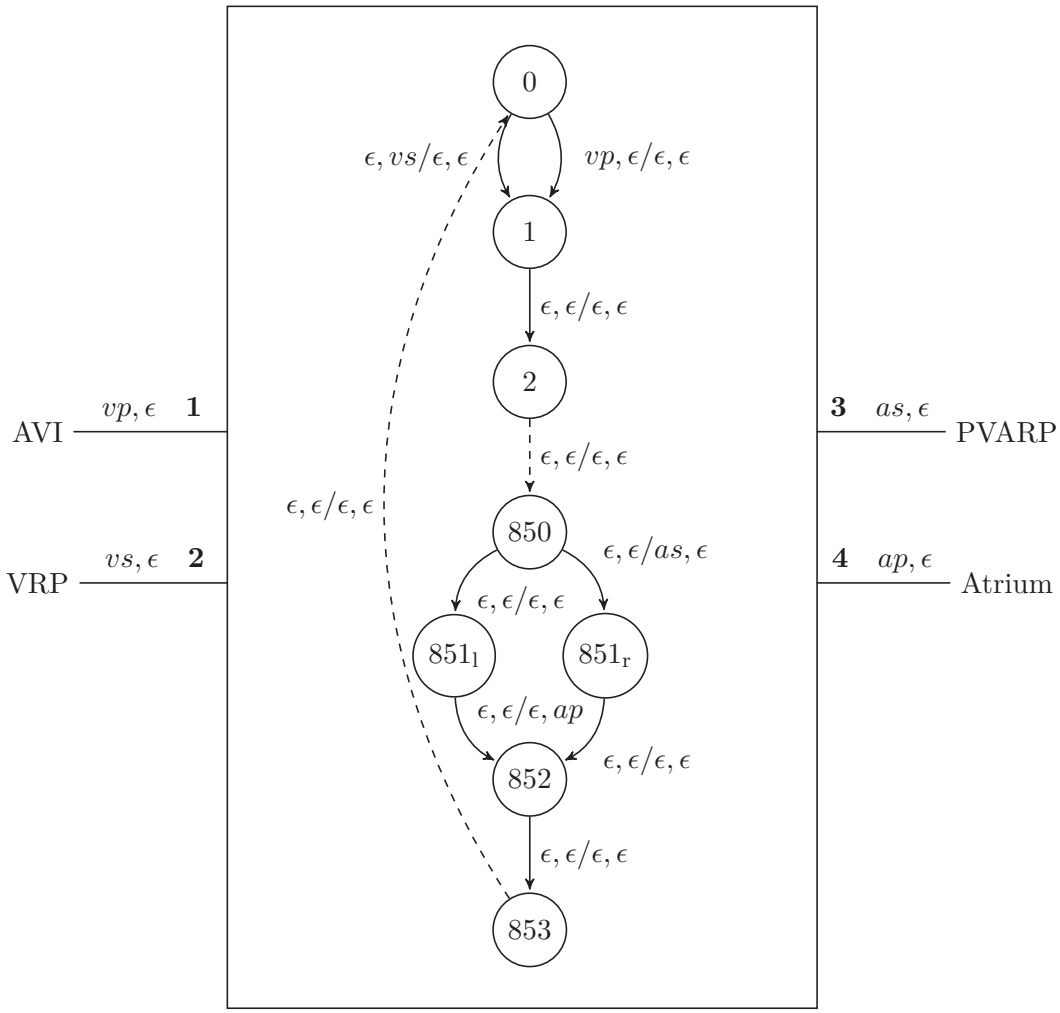


Figure 6.12: LRI (Lower Rate Interval)

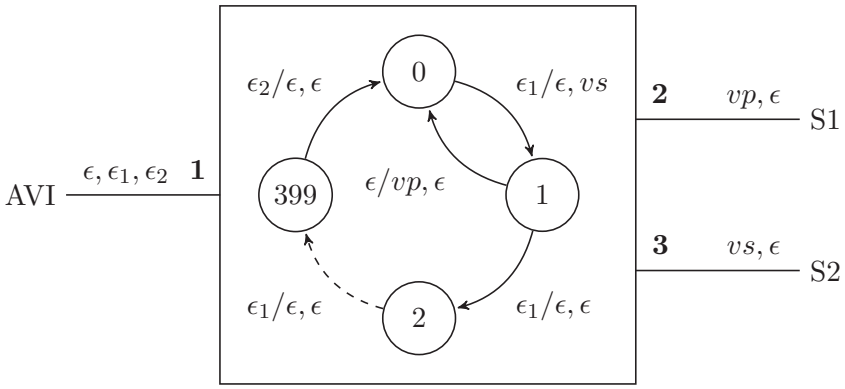


Figure 6.13: URI (Upper Rate Interval)

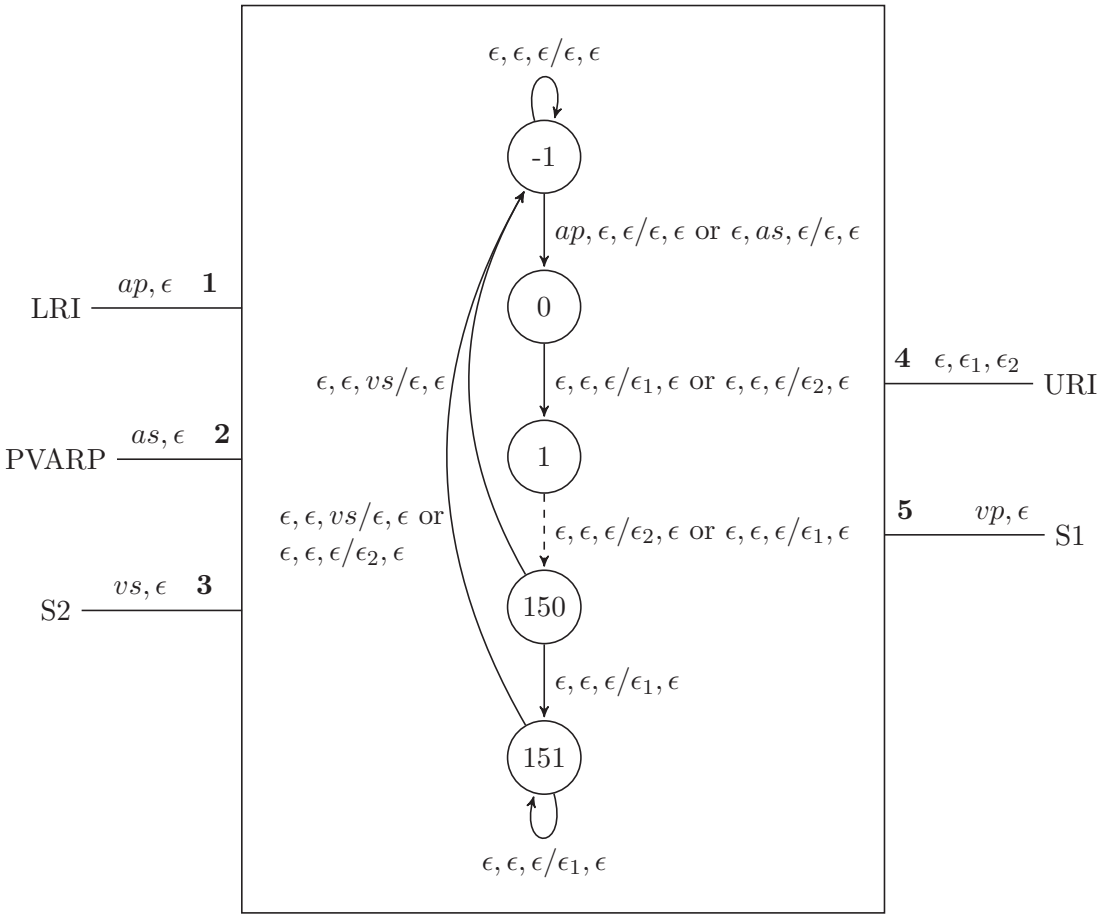


Figure 6.14: AVI (Atrio-Ventricular Interval)

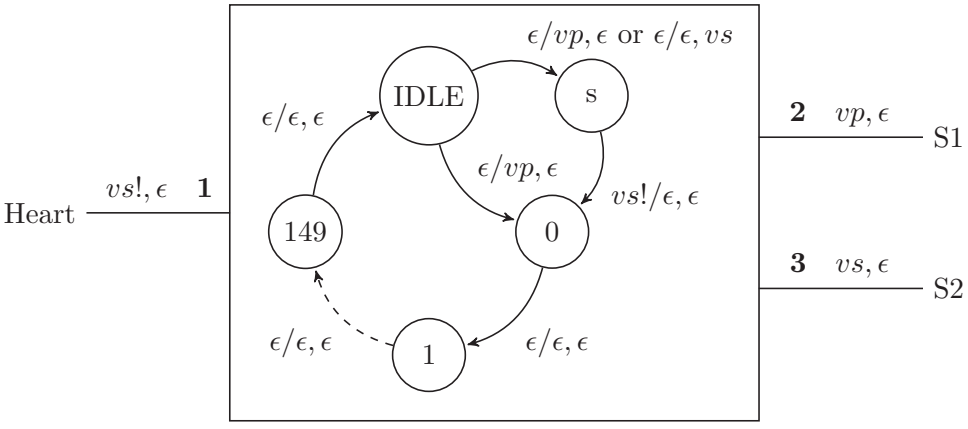


Figure 6.15: VRP (Ventricular Refractory Period)

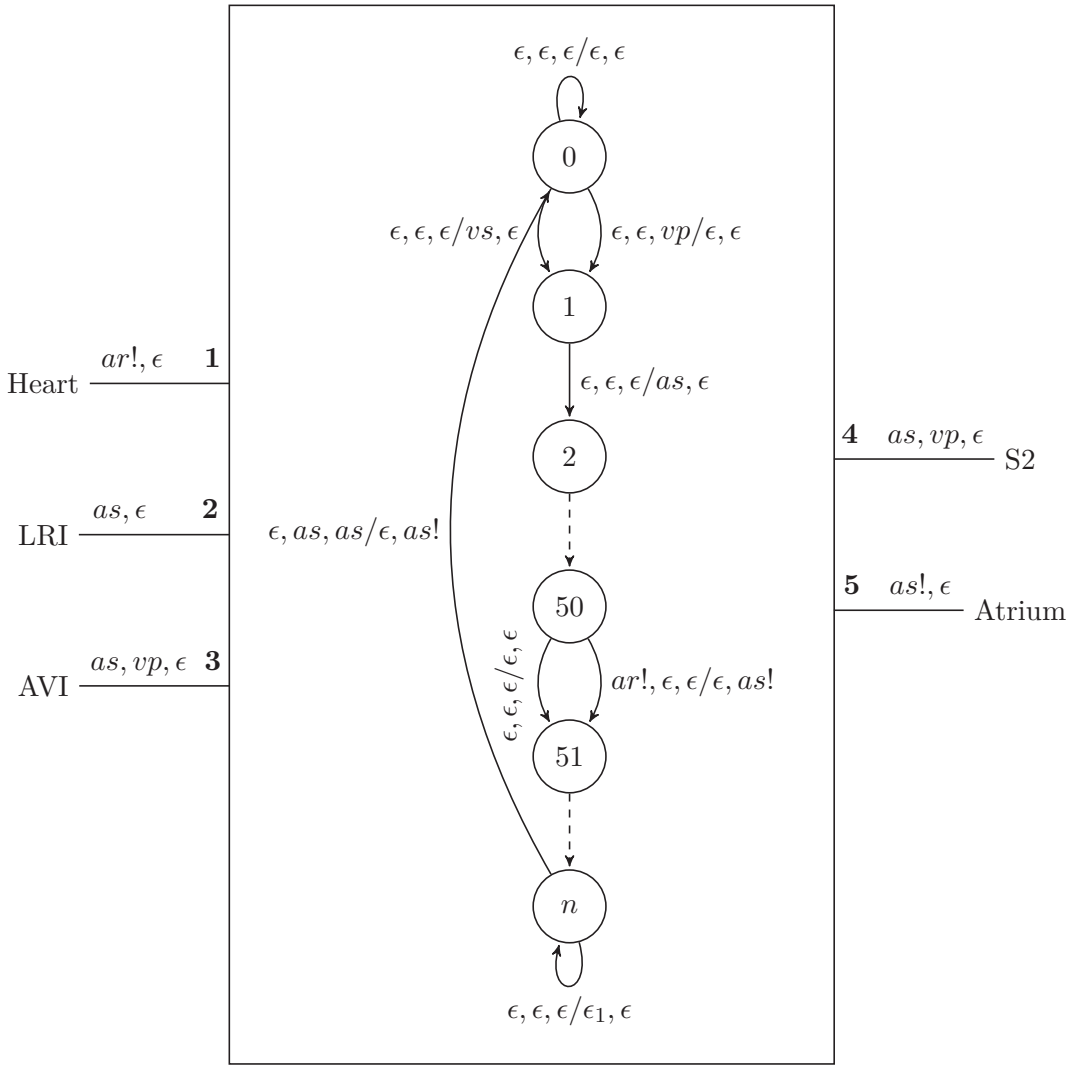


Figure 6.16: PVARP (Post Ventricular Atrial Refractory Period)

6.3 The *Lac Operon* System

In this section we provide a final biological application of the CospanSpan(Graph) algebra: we formalise the Lactose Operon in the Escherichia Coli bacterium, using for the first time a natively compositional framework.

The lactose operon in Escherichia Coli is composed of a sequence of genes that are responsible for producing three enzymes for lactose degradation, namely the lactose permease, which is incorporated in the membrane of the bacterium and actively transports the sugar into the cell, the β-galactosidase, which splits lactose into glucose and galactose, and the transacetylase, whose role is marginal. The Lac Operon functionality depends on the integration of two different control mechanisms, one mediated by lactose and the other by glucose. Since gene expression is an energy-consuming process, Escherichia Coli synthesises the proteins involved in the metabolism of lactose when this nutrient is present in the environment and the environment does not provide glucose, which is a more readily available source of energy.

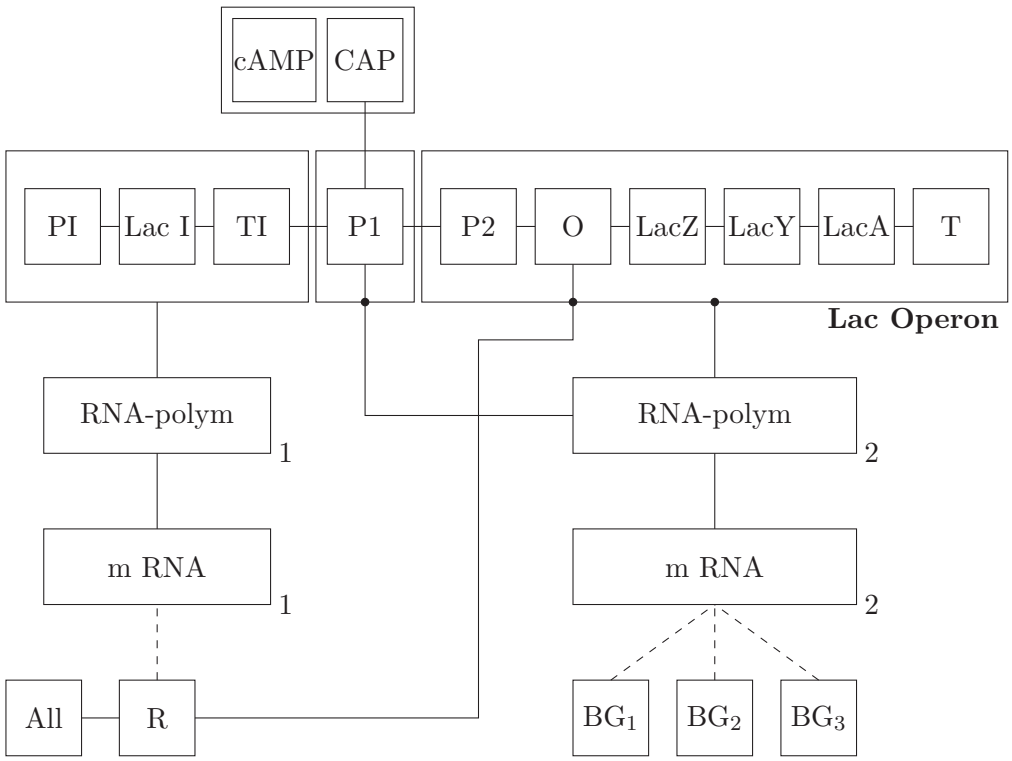


Figure 6.17: Lac Operon model

The model, from [8, 9, 10], that we consider is depicted in graphical form in Figure 6.17. The DNA sequence of the Lac Operon – depicted in Figure 6.17 – regulates the production of the enzymes, through the genes LacZ, LacY, LacA. The regulation process is as follows: gene LacI encodes the lac repressor R, which, in the absence of lactose, binds

to gene O (the operator). Transcription of structural genes into mRNA is performed by the RNA polymerase enzyme, which usually binds to gene P2 (the promoter) and scans the operon from left to right by transcribing the three structural genes LacZ, LacY and LacA into a single mRNA fragment. When the lac repressor R is bound to gene O (that is, the complex R-O is present) it becomes an obstacle for the RNA polymerase, and transcription of the structural genes is not performed. On the other hand, when lactose is present inside the bacterium, it binds to the repressor thus inhibiting the binding of R to O. This inhibition allows the transcription of genes LacZ, LacY, and LacA by the RNA polymerase.

A second mechanism is relevant: when glucose is not present, the complex cAMP-CAP, which is present and acting on P1, can increase significantly the expression of lac genes. Of course, also in the presence of the cAMP-CAP complex, the expression of the lac genes is inhibited by R-O.

Here, we just give two simple examples of the component O and of the protein R, in Figure 6.18 and Figure 6.19 respectively.

6.4 Robustness in CospanSpan(Graph)

As discussed previously, CospanSpan(Graph) is a compositional model for networks of automata with interfaces which provides new operations compared to the classical operations on automata given by Kleene's Theorem. One of the most significant operations is the *parallel with communication* which synchronises at least two components by allowing them to evolve simultaneously only when performing actions have the same effect of common interfaces.

In this section, we discuss robustness in CospanSpan(Graph). First of all, we consider the *system under perturbation*. CospanSpan(Graph) considers networks of finite automata. All components are finite: all states and transitions are finite sets. Considering timed and probabilistic CospanSpan(Graph), only discrete probabilities and discrete intervals of time are considered.

On the other end, biological systems are *continuous* dynamical systems, at least at a macroscopical level. So, we should argue if it is correct to provide discrete (even finite) models for them. From the perspective of control theory and automata theory, discrete and finite models are necessary; in fact, to control – or interact with – a dynamical system we have to consider a finite set of *threshold states* considered relevant. So, a discrete model such as CospanSpan(Graph) is still suitable in order to study the robustness of real systems.

Moreover, CospanSpan(Graph) is a very rich *algebra*, providing operations of the sequential and parallel composition of open (timed and probabilistic) components and their connectors. Complex systems of interacting components could be obtained by connecting simpler components, hierarchically. The algebra provides an expression for the resulting system, with an immediate geometric interpretation. Hence, in this approach, we could describe as a single network both the system under perturbation and the perturbing system.

In fact, CospanSpan(Graph) consider the *perturbation* agent as a part of the system. Therefore, it is described as a component or as a sub-component of the whole system. A perturbation agent communicates with other components via interfaces. Therefore, a perturbation is the result of a change of behaviour over interfaces. A perturbation is the result of:

- An *internal change of a component*. A change of state could result in a different behaviour on some interfaces. For instance, if a trap state is reached, the whole system could be blocked. In probabilistic CospanSpan(Graph), actions' probabilities in a component could change after an interaction.
  For example, in the Decimal Counter described in Example 5, a simple modification of the interface, by adding a new symbol t, leads to a block of the system's component, as in Figure 6.20.

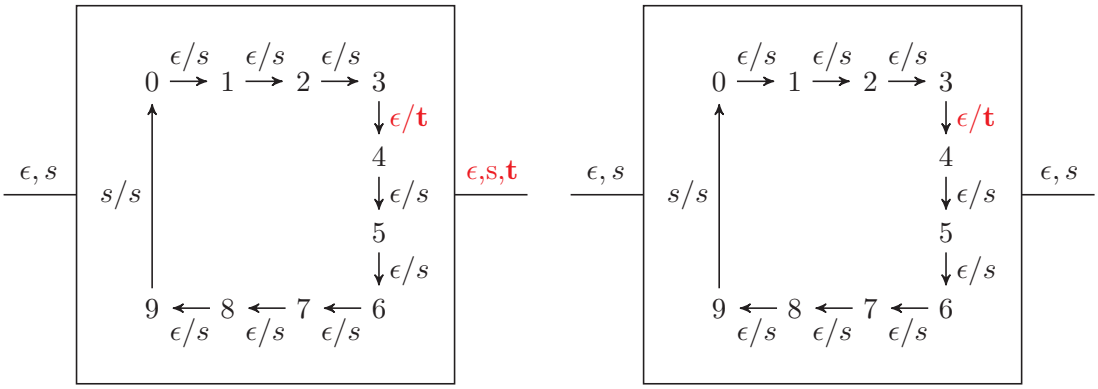


Figure 6.20: Decimal Counter with a perturbation

- A *substitution of a component (or a subnet)*. CospanSpan(Graph) describes networks of components with varying topology, as described in Sofia's Birthday Party example. Intuitively, when a component reaches a designed state (or set of states), it could be substituted with a different one having the same parallel interfaces, but a completely different internal structure. For instance, consider the Producer/Consumer problem where a consumer communicates with two buffers. If the second buffer is always empty, the system behaviour does not change. On the contrary, if the second buffer is not empty, the behaviour changes drastically.

Another interesting possibility open by the formalism CospanSpan(Graph) is the *static analysis of a system*. Since the system is described (algebraically and geometrically) as a network of components, we could consider a *topological distance* between any two components. So, we could analyse if a perturbation in component A will reach a component B (or not) and when. We could estimate the minimum time needed to reach a component, change its behaviour, or the impossibility of propagation for a perturbation.

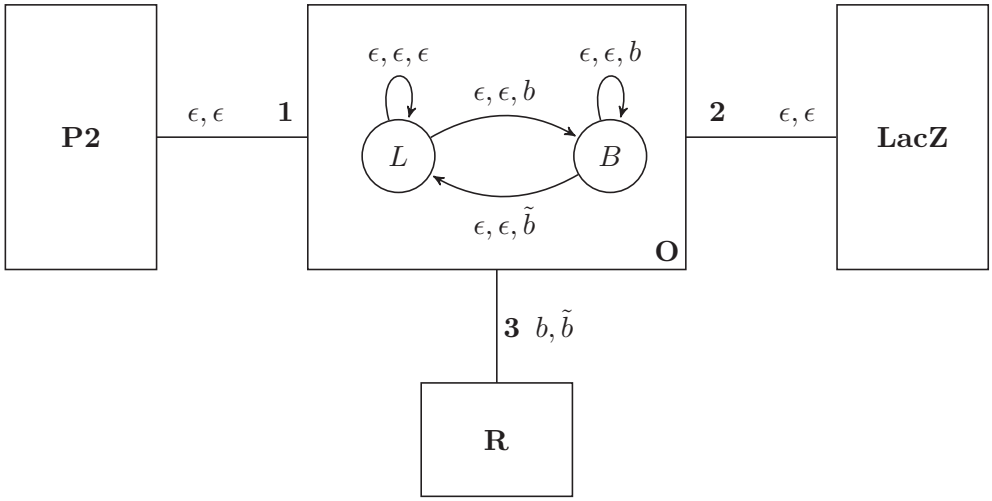


Figure 6.18: Lac Operon: component O

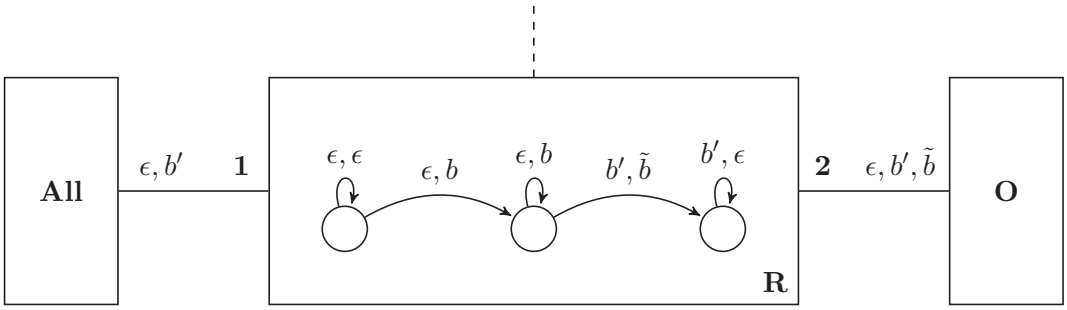


Figure 6.19: Lac Operon: protein R

7 Conclusion and future work

7.1 Biochemical Network Robustness

We have proposed a notion of robustness for biochemical networks that, essentially, evaluates the ability of a network to exhibit step-by-step limited variations on the quantity of a so-called output species at varying the initial concentration of some so-called input species.

Recently, *Robustness Temporal Logic* [78] (RobTL) has been proposed for the specification and analysis of distances between the behaviours of cyber-physical systems over a finite time horizon. Atomic propositions are defined using two simple languages: one to specify the effect of *perturbations* over an evolution sequence, and one to specify *distance expressions* between an evolution sequence and its perturbed version. In detail, atomic expressions are of the form $\Delta(\mathsf{exp}, \mathsf{pert}) \bowtie \eta$, with $\bowtie \in \{<, \leqslant, =, \neq, >, \geqslant\}$, and allow for comparing a threshold η with the distance, specified by an expression exp, between an evolution sequence and its perturbed version, obtained by applying a perturbation specified by pert, starting from a given time step. Boolean and temporal operators allow for extending these evaluations to the entire evolution sequence. Then, the tool STARK [79] offers: (i) languages for specifying systems, perturbations, distance expressions, and RobTL formulae; (ii) a module for the simulation of systems behaviours and their perturbed versions; (iii) a module for the evaluation of distances between behaviours; (iv) a statistical model checker for RobTL formulae. An interesting future work consists of extending RobTL in order to allow for specifying and analysing properties of biochemical networks. Then, enrich the STARK tool accordingly in order to use it for the robustness analysis of biochemical networks.

Although the simulations presented in Chapter 4 do not suffer from any problem, Gillespie's simulation approach suffers from scalability and computational cost problems

when applied to complex models. Therefore, we could consider approximate stochastic simulation algorithms, e.g. τ-leap [80].

7.2 Robustness and Hierarchy in CospanSpan(Graph)

We could analyse the relation between robustness and the hierarchical nature of a complex system. For a system, hierarchy is a crucial feature, but the notion of hierarchy itself is controversial. Models such as Harel's State Charts, Cardelli's Brane Calculi [47] and Paun's Membrane Systems [33, 34, 35] consider a hierarchical description of a system, where components could be encapsulated, or, on the contrary, refined.

In CospanSpan(Graph) a hierarchical description of a network is obtained gratis, in a very elegant way, thanks to the compositional nature of the approach.

However, hierarchy in a system has a different, deeper meaning: inside the system organisation components have *a different role*, and if an "important" component is changed, global behaviour could change dramatically. How could we measure the hierarchical weight or the importance of a component? A possibility is given by relating this concept to the system's robustness. The intuition is that a component is less hierarchically relevant iff its damage does not result in a lack of robustness.

A

SPEBNR – a Simple Python Environment for statistical estimation of Biochemical Network Robustness

In this Appendix, we describe the procedure needed for installing and using the tool SPEBNR, *a Simple Python Environment for statistical estimation of Biochemical Network Robustness*, introduced in Chapter 3. Then, we discuss what is needed in order to reproduce the analysis of the case studies considered in Chapter 4.

The tool is available at `https://github.com/dmanicardi/spebnr` and is implemented in Python. Essentially, the Python code consists of two components:

- **spebnr.py**, available at `https://github.com/dmanicardi/spebnr/blob/master/spebnr.py`, which implements the following procedures:
  - procedure SIMULATE (Figure 3.2), implemented as run(...)
  - procedure SIMSTEP (Figure 3.2), implemented as pstep(...)
  - procedure ESTIMATE (Figure 3.3), implemented as sample_element_from_list(...)
  - procedure DISTANCE (Figure 3.4), implemented as calculate_distance(...)
  - procedure COMPUTEH (Figure 3.4), implemented as wasserstein(...).
- **createSystem.py**, available at `https://github.com/dmanicardi/spebnr/blob/master/createSystem.py`. Although it does not implement any procedures presented in Chapter 3, it creates the map, which contains all possible species with their actions, generates the initial concentration of perturbed systems, and generates the plots.

In order to download SPEBNR, one can to clone the GitHub project as in Listing A.1:

```
git clone https://github.com/dmanicardi/spebnr.git
```

Listing A.1: cloning SPEBNR

This command has to be run in the directory where one desires to download the tool.

In order to run experiments, Python3 ($\geqslant$ 3.11) is needed. Moreover, the following Python packages are required:

- **numpy** `https://numpy.org` $\geqslant$ 1.23.4
- **scipy** `https://scipy.org/` $\geqslant$ 1.10.1
- **matplotlib** `https://matplotlib.org` $\geqslant$ 3.6.2
- **statistics** `https://github.com/digitalemagine/py-statistics` $\geqslant$ 1.0.3.5

In order to install all the required packages, one needs the command in Listing A.2:

```
pip install -r requirements.txt
```

Listing A.2: installing Python packages

Finally, in order to reproduce the experiments considered in Chapter 4, one has simply to run the Python codes available at `https://github.com/dmanicardi/spebnr/tree/master/caseStudies`. For instance, in order to reproduce the experiments in Section 4.1 on EnvZ/OmpR Osmoregulatory Signaling System in Escherichia Coli, the command in Listing A.3 below is enough:

```
python EnvZOmpR.py
```

Listing A.3: executing EnvZ/OmpR Osmoregulatory Signalling System in SPEBNR

Bibliography

[1] L. Nasti, R. Gori, and P. Milazzo, "Formalizing a notion of concentration robustness for biochemical networks," in *Software Technologies: Applications and Foundations - STAF 2018 Collocated Workshops, Toulouse, France, June 25-29, 2018, Revised Selected Papers* (M. Mazzara, I. Ober, and G. Salaün, eds.), vol. 11176 of *Lecture Notes in Computer Science*, pp. 81–97, Springer, 2018.

[2] M. Calder, S. Gilmore, and J. Hillston, "Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA," *Trans. Comp. Sys. Biology*, vol. 7, pp. 1–23, 2006.

[3] V. Castiglioni, M. Loreti, and S. Tini, "Measuring adaptability and reliability of large scale systems," in *Leveraging Applications of Formal Methods, Verification and Validation: Engineering Principles - 9th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2020, Rhodes, Greece, October 20-30, 2020, Proceedings, Part II* (T. Margaria and B. Steffen, eds.), vol. 12477 of *Lecture Notes in Computer Science*, pp. 380–396, Springer, 2020.

[4] V. Castiglioni, M. Loreti, and S. Tini, "A framework to measure the robustness of programs in the unpredictable environment," *CoRR*, vol. abs/2111.15319, 2021.

[5] P. Katis, N. Sabadini, and R. F. C. Walters, "A formalization of the IWIM model," in *Coordination Languages and Models, 4th International Conference, COORDINATION 2000, Limassol, Cyprus, September 11-13, 2000, Proceedings*, pp. 267–283, 2000.

[6] P. Katis, N. Sabadini, and R. Walters, "On the algebra of feedback and systems with boundary," *Rendiconti del Circolo Matematico di Palermo Serie II*, pp. 123–156, 2000.

[7] Z. Jiang, M. Pajic, S. Moarref, R. Alur, and R. Mangharam, "Modeling and verification of a dual chamber implantable pacemaker," in *Tools and Algorithms for the Construction and Analysis of Systems - 18th International Conference, TACAS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings* (C. Flanagan and B. König, eds.), vol. 7214 of *Lecture Notes in Computer Science*, pp. 188–203, Springer, 2012.

[8] L. Corolli, C. Maj, F. Marini, D. Besozzi, and G. Mauri, "An excursion in reaction systems: From computer science to biology," *Theor. Comput. Sci.*, vol. 454, pp. 95–108, 2012.

[9] G. Pardini, R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and S. Tini, "Compositional semantics and behavioural equivalences for reaction systems with restriction," *Theor. Comput. Sci.*, vol. 551, pp. 1–21, 2014.

[10] F. J. Romero-Campero and M. J. Pérez-Jiménez, "Modelling gene expression control using P systems: The lac operon, a case study," *Biosyst.*, vol. 91, no. 3, pp. 438–457, 2008.

[11] A. Uri, *An introduction to systems biology: design principles of biological circuits.* Chapman and Hall/CRC, 2006.

[12] K. Zhou and J. C. Doyle, *Essentials of Robust Control.* Prentice-Hall, 1998.

[13] H. Kitano, "Towards a theory of biological robustness," *Molecular Systems Biology*, vol. 3, no. 1, p. 137, 2007.

[14] A. Shahrokni and R. Feldt, "A systematic review of software robustness," *Inf. Softw. Technol.*, vol. 55, no. 1, pp. 1–17, 2013.

[15] H. Kitano, "Biological robustness," *Nat. Rev. Genet.*, vol. 5, pp. 826–837, 2004.

[16] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943.

[17] W. Zielonka, "Notes on finite asynchronous automata," *RAIRO Theor. Informatics Appl.*, vol. 21, no. 2, pp. 99–135, 1987.

[18] C. A. Petri and W. Reisig, "Petri net," *Scholarpedia*, vol. 3, no. 4, p. 6477, 2008.

[19] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *The journal of physical chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.

[20] D. J. Barnes and D. F. Chu, "Introduction to modeling for biosciences," 2010.

[21] J. Bènabou, "Introduction to Bicategories," *Reports of the Midwest Category Seminar*, vol. 47, pp. 1–77, 1967.

[22] P. Katis, N. Sabadini, and R. F. C. Walters, "Span(graph): A categorial algebra of transition systems," in *Algebraic Methodology and Software Technology, 6th International Conference, AMAST '97, Sydney, Australia, December 13-17, 1997, Proceedings* (M. Johnson, ed.), vol. 1349 of *Lecture Notes in Computer Science*, pp. 307–321, Springer, 1997.

[23] P. Katis, N. Sabadini, and R. F. C. Walters, "A formalization of the IWIM model," in *Coordination Languages and Models, 4th International Conference, COORDINATION 2000, Limassol, Cyprus, September 11-13, 2000, Proceedings* (A. Porto and G. Roman, eds.), vol. 1906 of *Lecture Notes in Computer Science*, pp. 267–283, Springer, 2000.

[24] R. Lanotte, D. Manicardi, and S. Tini, "Step-by-step robustness for biochemical networks," in *Proceedings of the 24th Italian Conference on Theoretical Computer Science, Palermo, Italy, September 13-15, 2023* (G. Castiglione and M. Sciortino, eds.), vol. 3587 of *CEUR Workshop Proceedings*, pp. 299–313, CEUR-WS.org, 2023.

[25] A. Gianola, S. Kasangian, D. Manicardi, N. Sabadini, F. Schiavio, and S. Tini, "Cospanspan(graph): a compositional description of the heart system," *Fundam. Informaticae*, vol. 171, no. 1-4, pp. 221–237, 2020.

[26] A. Gianola, S. Kasangian, D. Manicardi, N. Sabadini, and S. Tini, "Compositional modeling of biological systems in cospanspan(graph)," in *Proceedings of the 21st Italian Conference on Theoretical Computer Science, Ischia, Italy, September 14-16, 2020* (G. Cordasco, L. Gargano, and A. A. Rescigno, eds.), vol. 2756 of *CEUR Workshop Proceedings*, pp. 61–66, CEUR-WS.org, 2020.

[27] A. Gianola, S. Kasangian, D. Manicardi, N. Sabadini, and S. Tini, "Compositional modeling of biological systems in cospanspan (graph)(extended version)," tech. rep., Technical report, https://gianola. people. unibz. it, 2020.

[28] E. Klipp, R. Herwig, A. Kowald, C. Wierling, and H. Lehrach, "Systems biology in practice concepts, implementation and application," *Wiley-VCH, Weinheim*, 05 2005.

[29] l. H. Meinhardt, "Models of biological pattern formation: from elementary steps to the organization of embryonic axes," *Current Topics in Developmental Biology*, vol. 81, pp. 1–63, 2008.

[30] J. Von Neumann, *Theory of self-reproducing automata*. University of Illinois Press, 1966.

[31] G. Ermentrout and L. Edelstein-Keshet, "Cellular automata approaches to biological modeling," *Journal of Theoretical Biology*, vol. 160, no. 1, pp. 97–133, 1993.

[32] P. Prusinkiewicz and A. Lindenmayer, *The algorithmic beauty of plants*. The virtual laboratory, Springer, 1990.

[33] G. Paun, "Computing with membranes," *J. Comput. Syst. Sci.*, vol. 61, no. 1, pp. 108–143, 2000.

[34] G. Paun, *Membrane Computing: An Introduction*. Natural computing series, Springer, 2002.

[35] G. Paun and G. Rozenberg, "A guide to membrane computing," *Theor. Comput. Sci.*, vol. 287, no. 1, pp. 73–100, 2002.

[36] M. J. Pérez-Jiménez and F. J. Romero-Campero, "A study of the robustness of the EGFR signalling cascade using continuous membrane systems," in *Mechanisms, Symbols, and Models Underlying Cognition: First International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC* (J. Mira and J. R. Álvarez, eds.), vol. 3561 of *Lecture Notes in Computer Science*, pp. 268–278, Springer, 2005.

[37] A. Ehrenfeucht and G. Rozenberg, "Reaction systems," *Fundam. Informaticae*, vol. 75, no. 1-4, pp. 263–280, 2007.

[38] S. Azimi, B. Iancu, and I. Petre, "Reaction system models for the heat shock response," *Fundam. Informaticae*, vol. 131, no. 3-4, pp. 299–312, 2014.

[39] S. Azimi, C. Panchal, A. Mizera, and I. Petre, "Multi-stability, limit cycles, and period-doubling bifurcation with reaction systems," *Int. J. Found. Comput. Sci.*, vol. 28, no. 8, pp. 1007–1020, 2017.

[40] H. Kitano, "A graphical notation for biochemical networks," *BIOSILICO*, vol. 1, pp. 169–176, 11 2003.

[41] J. A. Bergstra, A. Ponse, and S. A. Smolka, eds., *Handbook of Process Algebra.* North-Holland / Elsevier, 2001.

[42] A. Regev and E. Shapiro, "Cellular abstractions: Cells as computation," *Nature*, vol. 419, no. 6905, pp. 343–343, 2002.

[43] A. Regev, W. Silverman, and E. Shapiro, "Representation and simulation of biochemical processes using the pi-calculus process algebra," in *Proceedings of the 6th Pacific Symposium on Biocomputing, PSB 2001, Hawaii, USA, January 3-7, 2001* (R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, eds.), pp. 459–470, 2001.

[44] C. Priami, A. Regev, E. Shapiro, and W. Silverman, "Application of a stochastic name-passing calculus to representation and simulation of molecular processes," *Inf. Process. Lett.*, vol. 80, no. 1, pp. 25–31, 2001.

[45] V. Danos and C. Laneve, "Formal molecular biology," *Theor. Comput. Sci.*, vol. 325, no. 1, pp. 69–110, 2004.

[46] A. Regev, E. M. Panina, W. Silverman, L. Cardelli, and E. Shapiro, "Bioambients: an abstraction for biological compartments," *Theor. Comput. Sci.*, vol. 325, no. 1, pp. 141–167, 2004.

[47] L. Cardelli, "Brane calculi," in *Computational Methods in Systems Biology, International Conference, CMSB 2004, Paris, France, May 26-28, 2004, Revised Selected Papers* (V. Danos and V. Schächter, eds.), vol. 3082 of *Lecture Notes in Computer Science*, pp. 257–278, Springer, 2004.

[48] C. Priami and P. Quaglia, "Beta binders for biological interactions," in *Computational Methods in Systems Biology, International Conference, CMSB 2004* (V. Danos and V. Schächter, eds.), vol. 3082 of *Lecture Notes in Computer Science*, pp. 20–33, Springer, 2004.

[49] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and A. Troina, "A calculus of looping sequences for modelling microbiological systems," *Fundam. Informaticae*, vol. 72, no. 1-3, pp. 21–35, 2006.

[50] F. Ciocchetta and J. Hillston, "Bio-pepa: A framework for the modelling and analysis of biological systems," *Theor. Comput. Sci.*, vol. 410, no. 33-34, pp. 3065–3084, 2009.

[51] A. Degasperi and M. Calder, "A process algebra framework for multi-scale modelling of biological systems," *Theor. Comput. Sci.*, vol. 488, pp. 15–45, 2013.

[52] B. Liu, S. Kong, S. Gao, P. Zuliani, and E. M. Clarke, "Towards personalized prostate cancer therapy using delta-reachability analysis," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC'15, Seattle, WA, USA, April 14-16, 2015* (A. Girard and S. Sankaranarayanan, eds.), pp. 227–232, ACM, 2015.

[53] H. Matsuno, Y. Tanaka, H. Aoshima, A. Doi, M. Matsui, and S. Miyano, "Biopathways representation and simulation on hybrid functional petri net," *Silico Biol.*, vol. 3, no. 3, pp. 389–404, 2003.

[54] A. Rizk, G. Batt, F. Fages, and S. Soliman, "A general computational method for robustness analysis with applications to synthetic gene networks," *Bioinform.*, vol. 25, no. 12, 2009.

[55] A. Rizk, G. Batt, F. Fages, and S. Soliman, "Continuous valuations of temporal logic specifications with applications to parameter optimization and robustness measures," *Theor. Comput. Sci.*, vol. 412, no. 26, pp. 2827–2839, 2011.

[56] N. Barkai and S. Leibler, "Robustness in simple biochemical networks," *Nature*, vol. 387, pp. 913–917, 1997.

[57] G. Shinar and M. Feinberg, "Structural sources of robustness in biochemical reaction networks," *Science*, vol. 327, no. 5971, pp. 1389–1391, 2010.

[58] G. Shinar and M. Feinberg, "Design principles for robust biochemical reaction networks: what works, what cannot work, and what might almost work," *Mathe. Biosci*, vol. 231, p. 39–48, 2011.

[59] V. Castiglioni, M. Loreti, and S. Tini, "How adaptive and reliable is your program?," in *Formal Techniques for Distributed Objects, Components, and Systems - 41st IFIP WG 6.1 International Conference, FORTE 2021, Held as Part of the 16th International Federated Conference on Distributed Computing Techniques, DisCoTec 2021,*

Valletta, Malta, June 14-18, 2021, Proceedings (K. Peters and T. A. C. Willemse, eds.), vol. 12719 of *Lecture Notes in Computer Science*, pp. 60–79, Springer, 2021.

[60] S. T. Rachev, L. B. Klebanov, S. Stoyanov, and F. J. Fabozzi, "The methods of distances in the theory of probability and statistics," 2013.

[61] C. Villani, "Optimal transport: Old and new," 2008.

[62] L. V. Kantorovich, "On the translocation of masses," in *Dokl. Akad. Nauk. USSR (NS)*, vol. 37, pp. 199–201, 1942.

[63] D. Thorsley and E. Klavins, "Approximating stochastic biochemical processes with Wasserstein pseudometrics," *IET Syst. Biol.*, vol. 4, no. 3, pp. 193–211, 2010.

[64] B. K. Sriperumbudur, K. Fukumizu, A. Gretton, B. Schölkopf, and G. R. G. Lanckriet, "On the empirical estimation of integral probability metrics," *Electronic Journal of Statistics*, vol. 6, pp. 1550–1599, 2021.

[65] P. Linz, *Analytical and numerical methods for Volterra equations*, vol. 7 of *SIAM studies in applied and numerical mathematics*. SIAM, 1985.

[66] A. J. Lotka, "Contribution to the theory of periodic reactions," *The Journal of Physical Chemistry*, vol. 14, no. 3, pp. 271–274, 2002.

[67] E. W. Weisstein, "Logistic equation," *https://mathworld. wolfram. com/*, 2003.

[68] D. Besozzi, G. Mauri, G. Păun, and C. Zandron, "Gemmating P systems: collapsing hierarchies," *Theor. Comput. Sci.*, vol. 296, no. 2, pp. 253–267, 2003.

[69] C. Martín-Vide, G.Mauri, G. Paun, G. Rozenberg, and A. Salomaa, eds., *Membrane Computing, International Workshop, WMC 2003, Tarragona, Spain, July 17-22, 2003, Revised Papers*, vol. 2933 of *Lecture Notes in Computer Science*, Springer, 2004.

[70] C. Ferretti, G. Mauri, and C. Zandron, eds., *DNA Computing, 10th International Workshop on DNA Computing, DNA 10, Milan, Italy, June 7-10, 2004, Revised Selected Papers*, vol. 3384, Springer, 2005.

[71] A. Gianola, S. Kasangian, and N. Sabadini, "Cospan/span(graph): an algebra for open, reconfigurable automata networks," in *7th Conference on Algebra and Coalgebra in Computer Science, CALCO 2017, June 12-16, 2017, Ljubljana, Slovenia* (F. Bonchi and B. König, eds.), vol. 72 of *LIPIcs*, pp. 2:1–2:17, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[72] R. F. C. Walters, *Categories and computer science*, vol. 28 of *Cambridge computer science texts*. Cambridge University Press, 1991.

[73] A. Cherubini, N. Sabadini, and R. F. C. Walters, "Timing in the cospan-span model," vol. 104, pp. 81–97, 2003.

[74] L. de Francesco Albasini, N. Sabadini, and R. F. C. Walters, "The compositional construction of markov processes," *Appl. Categorical Struct.*, vol. 19, no. 1, pp. 425–437, 2011.

[75] L. de Francesco Albasini, N. Sabadini, and R. F. C. Walters, "The compositional construction of markov processes II," *RAIRO Theor. Informatics Appl.*, vol. 45, no. 1, pp. 117–142, 2011.

[76] N. Sabadini, F. Schiavio, and R. Walters, "On the geometry and algebra of networks with state.," *Theor. Comput. Sci.*, vol. 64, pp. 144–163, 2017.

[77] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

[78] V. Castiglioni, M. Loreti, and S. Tini, "Robtl: A temporal logic for the robustness of cyber-physical systems," *CoRR*, vol. abs/2212.11158, 2022.

[79] V. Castiglioni, M. Loreti, and S. Tini, "Stark: A software tool for the analysis of robustness in the unknown environment," in *Coordination Models and Languages - 25th IFIP WG 6.1 International Conference, COORDINATION 2023, Held as Part of the 18th International Federated Conference on Distributed Computing Techniques, DisCoTec 2023, Lisbon, Portugal, June 19-23, 2023, Proceedings* (S. Jongmans and A. Lopes, eds.), vol. 13908 of *Lecture Notes in Computer Science*, pp. 115–132, Springer, 2023.

[80] D. T. Gillespie, "Approximate accelerated stochastic simulation of chemically reacting systems," *The Journal of chemical physics*, vol. 115, no. 4, pp. 1716–1733, 2001.