Computational Systems Biology:
A Calculus for Biomolecular knowledge

THESIS SUBMITTED FOR THE DEGREE "DOCTOR OF PHILOSOPHY"

BY

Aviv Regev

SUBMITTED TO THE SENATE OF TEL AVIV UNIVERSITY

December, 2002

This work was carried out under the supervision of

Prof. Eva Jablonka
Prof. Ehud Shapiro (Weizmann Institute)

Acknowledgments

I am thoroughly indebted to Ehud Shapiro, who has guided me through this thesis, both scientifically and personally. Udi was an exceptional mentor, supporting my independence while providing guidance, direction, and intellectual stimulation, turning vague ideas into concrete science.

I was also extremely fortunate to benefit from the guidance of Eva Jablonka. I owe my biological motivation and thinking to Eva, who has been my guide and mentor since my undergraduate studies, and taught me the humane side of being a scientist. In a sense, this work is the culmination of a process that started in Eva's undergraduate courses in Genetics and Evolution.

William Silverman, who implemented the computational framework underlying this thesis, is not only an exceptional developer, but was also a wonderful person to work with. I am tremendously grateful for Bill's help.

Naama Barkai has opened up the world of modeling to me, teaching me how to ask biological questions and glean new biological insights from computational models. I am very grateful for Naama's collaboration, devoting time and ideas, as well as her rare understanding and patience.

Sara Lavi, who was my M.Sc. advisor, helped me believe in myself and follow my sometimes unorthodox path, in both the intellectual and administrative sense.

I have learned much from guiding Katya Panina, who came as a summer student and left as a friend. Katya made a significant scientific contribution to this thesis.

Corrado Priami and Luca Cardelli, each in his own field, were essential to some of the computational developments in this work. I am grateful for their openness to a new field, and their help as collaborators. I would also like to thank Vincent Schachter for many stimulating scientific (and other) discussions.

One of the pleasant surprises of this thesis was acquiring a wonderful friend and exceptional colleague, Dana Pe'er. Dana ushered me into the complementary world of computational learning of pathway models. Through the ensuing collaboration, I have had the great opportunity of working with Nir Friedman, learning a whole new way of looking into biology. In addition to Nir, I also enjoyed working in this ever-increasing side project with Amos Tannay, Gal Elidan, Eran Segal and Daphne Koller.

I am particularly grateful to Yehuda Ben-Shaul, my coordinating supervisor, and to the faculty and staff of the Department of Cell Research and Immunology at Tel Aviv University, and especially the department head, Mia Horowitz, for their help and support. I would likewise wish to thank the staff at my "home away from home" at the Department of Computer Science at the Weizmann Institute.

My work was supported by the Colton Family Foundation. I am personally grateful to Judy and Stewart Colton not only for their financial support, but for their interest, concern, and encouragement. I also thank Shoshana Noy for her role as a liaison with the Colton Foundation.

I am thankful to the staff of "Cafe-Cafe" for seeing me through the last mile of the writing process.

Finally, I would like to thank my husband, Rani Nelken, without whom this work would have neither started nor finished.

Note on collaboration

This research was done under the supervision of Prof. Ehud Shapiro, who guided its development at all levels, conceptual, theoretical and practical. Prof. Eva Jablonka guided the biological perspective, especially at the conceptual level. William Silverman developed the BioSpi simulation system, a continuous and ongoing effort. The basic concepts are reported in [111, 110] and in Chapter 1.

Several other people contributed to the work reported here. Naama Barkai has led the research about the circadian clock, modularity, and cell cycle gating (Chapter 2). Corrado Priami has contributed to the development of the stochastic biochemical π-calculus ([109], Chapter 2). Katya Panina helped develop BioAmbients, with the guidance and collaboration of Luca Cardelli ([108], Chapter 3).

The work reported on in the appendix was led by Dana Pe'er, Nir Friedman, and Daphne Koller, with the author providing the biological aspect of the research. Gal Elidan, Amos Tannay, and Eran Segal also participated in this work.

Contents

Abstract

In recent years, molecular biology is experiencing a veritable "Cambrian explosion" of new experimental results, accumulating at a staggering rate. The major challenge is now to make sense of the mound of data and integrate it into a coherent view of the biomolecular world. Considerable success on this front has been achieved in the branches of molecular biology that study the sequence and structure of genes and proteins. This may be largely attributed to the growing reliance on computational approaches for processing data prevalent in these fields. The mainstay of molecular biology, the research of systems' behavior and function, is yet to make a similar leap.

A prerequisite for the application of computational methods is finding a meaningful mathematical abstraction of the data. Even the seemingly simple task of creating a searchable database of experimental results requires an abstraction according to which to encode the data. For instance, we encode DNA sequences as mathematical strings of characters, and proteins as 3-dimensional labeled graphs. This allows research in each of these fields to have a field-specific language in which to describe the objects of study and apply computational methods to them. Unfortunately, systems biology lacks such a unifying abstraction, hindering its amenability to computational processing.

In this thesis, we wish to close this gap, by developing a novel abstraction for biomolecular systems. For our abstraction, we turn to the study of concurrent systems in computer science. A major tenet of this work is that there is an inherent similarity between concurrent computation and systems biology, and that the languages, tools and methods developed for computational systems can be co-opted and adapted for biomolecular ones. In particular, we claim that just as the string is the main building block for abstracting DNA, so the computational process is the main building block for building a "molecule as computation" abstraction of biomolecular systems.

We develop this analogy in detail, by adopting a process algebra, the π-calculus, originally developed with computational systems in mind, for the description of biomolecular systems. We employ our abstraction in the modeling of various representative use-cases as well as the complex signaling pathway from the receptor tyrosine kinase (RTK) through the MAP kinase (MAPK) cascade. To account for quantitative aspects, we extend our abstraction with a stochastic variant, correctly handling the stochastic nature of biomolecular systems. Based on this abstraction, we build a simulation system, BioSpi, for semi-quantitative and stochastic simulations of biomolecular systems.

We further extend our abstraction to handle molecular and membrane bound compartments by incorporating the ambient calculus into our framework. The ambient calculus is another process algebra for the specification of process location and movement through computation domains. We adapt the calculus to represent various aspects of molecular localization and compartmentalization, including the movement of molecules between compartments and dynamic rearrangement of cellular compartments. We incorporate the adapted calculus as part of the BioSpi system, to provide a fuller modular framework for molecular and cellular compartmentalization, and use it to model and study the complex multi-cellular hypothalamic system for weight regulation.

Developing a formal model of biological systems opens up the way for providing answers to higher-level questions. One particularly exciting line of research that is enabled by our framework is the algorithmic comparison of models of biological systems. One of the hallmarks of process algebras in computer science is precisely this ability of comparing the abstract specification of a system with its concrete

implementation. In computer science, this is used as a tool for ensuring system quality. In biology, it can be used both either for comparing an abstract functional view with the concrete details of the process or for evolutionary comparisons. For instance, using the ability of the calculus to capture modular structures, we investigate the circadian machinery at two levels of abstraction. First, we model the molecular interactions explicitly. Second, we replace part of the molecular level model with a a functional one, representing a hysteresis module. By using two BioSpi programs, we show that both levels of description are equally good at capturing the behavior of the system, and establish the function of the hysteresis module within the clock. We extend this modular approach to study inter-pathway interactions, and examine the mutual effects of the circadian clock and the cell division cycle, proposing a new model for the molecular mechanism underlying the gating effect exerted by the circadian clock on cell division in rapidly dividing cells. In future work we will adapt the mathematical machinery of the π-calculus to further formalize the comparison of biological systems both for comparing levels of abstraction and for evolutionary development.

Computer and biomolecular systems both start from a small set of elementary components from which, layer by layer, more complex entities are constructed with ever-more sophisticated functions. Computers are networked to perform larger and larger computations; cells form multicellular organisms. All existing computers have an essentially similar core design and basic functions, but address a broad range of tasks. Similarly, all living cells have a similar core design, yet can survive in radically different environments or serve a wide range of functions in a multicellular organism. Of course, biomolecular systems exist independently of our awareness or understanding of them, whereas computer systems exist because we understand, design and build them. Nevertheless, the abstractions, tools and methods used to specify and study computer systems should illuminate our accumulated knowledge about biomolecular systems.

As a complementary approach to that described above, we also report on a second thread of work undertaken as part of this thesis. We undertook a bottom-up inductive method of inferring pathway structure directly from gene expression profiles. Our work[1] employs graphical probabilistic models to reconstruct models of biomolecular systems. Such models are stochastic descriptions of biological processes that could have generated the observed data. Our approach is holistic: we attempt to reconstruct a model that globally fits our data, by studying the joint probability distribution over the set of all genes. This joint distribution reflects the distribution of cellular and molecular "states" and how these affect transcript levels. Taking various approaches, we attempt to learn biologically relevant models of gene regulation and molecular organization of different granularities.

Direct modeling and inference are complementary approaches that are based on a shared view of biomolecular systems as stochastic processes. Much is to be gained by combining the two. For example, we may attempt to infer extensions or modification of models of (partly) known systems rather than reconstruct whole systems from scratch. As we wish to reconstruct more accurate and detailed models, the number of unknown parameters and the complexity of the computational problem increase significantly. Prior knowledge can constrain and facilitate the learning process, thereby maximizing the quality and relevance of our findings.

[1]This work was done in collaboration with Dr. Nir Friedman's group at the School of Computer Science and Engineering, Hebrew University of Jerusalem and with Prof. Daphne Koller's group at the Computer Science Department, Stanford University. My contribution was towards the biological formulation of the model (prior to model learning), in the development of bioinformatics methods to validate the results, and in carrying out the bioinformatics validations. The work is described in Appendix A and in [100],[101], and [122].

Chapter 1

Biomolecular processes as concurrent computation: The π-calculus

1.1 Introduction

Biochemical processes, carried out by networks of proteins, underly the major functions of living cells ([15],[129]). Although we are successfully consolidating our knowledge of the 'sequence' and 'structure' branches of molecular cell biology in an easily accessible manner, the mountains of knowledge about the function, activity, and interaction of molecular systems in cells remain fragmented. Sequence and structure research use computers and computerized databases to share, compare, criticize and correct scientific knowledge, thus converging to a consensus quickly and effectively. Why can't the study of biomolecular systems make a similar computational leap? The deep reason lies in the broad adoption of good abstractions in sequence and structure research: the 'DNA-as-string' abstraction and the 'protein-as-three-dimensional-labeled-graph' abstraction, respectively. Biomolecular systems research has yet to identify and adopt a similarly successful abstraction.

The hallmark of scientific understanding is the reduction of a natural phenomenon to simpler units. Equally important understanding comes from finding the appropriate abstraction with which to distill an aspect of knowledge. An abstraction - a mapping from a real-world domain to a mathematical domain - highlights some essential properties while ignoring other, complicating, ones. For example, programming languages describe a sequence of operations in an abstract way, independent of the details of operation of the electronic circuits that realize them. Similarly, classical genetic analysis uses the 'gene-as-hereditary-unit' abstraction, ignoring the biochemical properties of genes as DNA sequences.

A good scientific abstraction has four properties: it is relevant, capturing an essential property of the phenomenon; computable, bringing to bear computational knowledge about the mathematical representation; understandable, offering a conceptual framework for thinking about the scientific domain; and extensible, allowing the capture of additional real properties in the same mathematical framework. For example, the sequence-as-string abstraction is relevant in capturing the primary sequence of nucleotides without including higher- and lower-order biochemical properties; it allows the application of a battery of string algorithms, including probabilistic analysis using hidden Markov models, as well as enabling the practical development of databases and common repositories; it is understandable, in that a string over the alphabet {A,T,C,G} is a universal format for discussing and conveying genetic information; and

[0]The work in this chapter was published in [111] and [110]

extensible, enabling, for example, the addition of a fifth symbol denoting methylated cytosine.

Similarly, for a good abstraction for biomolecular processes to be relevant, it should capture two essential properties of these systems in one unifying view: their molecular organization and their dynamic behavior. A computable abstraction will then allow both the simulation of this dynamic behavior, and the qualitative and quantitative reasoning on these systems' properties. An understandable abstraction will correspond well to the informal concepts and ideas of molecular biology, while opening up new computational possibilities for understanding molecular systems, by *e.g.* suggesting formal ways to ascribe biomolecular function to a biological system, or by suggesting objective measures for behavioral similarities between systems. Finally, the desired abstraction should be extensible, scaling to higher levels of organization, in which molecular systems play a key, albeit not exclusive, role.

In this work we aim to adopt the much-needed abstraction for biomolecular systems from computer science. We use advanced concepts from concurrency theory to investigate the "molecule-as-computation" abstraction, in which a system of interacting molecular entities is described and modeled by a system of interacting computational entities. Based on this abstraction, we adapt a family of abstract computer languages, called process algebras, for the representation and simulation of biomolecular systems, including regulatory, metabolic and signaling pathways, as well as multicellular processes.

1.1.1 Previous work

Biochemical systems are traditionally studied using "dynamical systems theory". In recent years, additional approaches from computer science have been adapted for the representation of pathways. The various approaches can be roughly divided into four groups.

Chemical kinetic models

Dynamical systems theory views biomolecular systems based on the "cell-as-collection-of-molecular-species" abstraction. Such models describe different molecular systems from a pure biochemical perspective (*e.g.* [117], [84], [83], [14], [9], [74], [7], [153], [120]). In this approach, a biomolecular system is abstracted in terms of the quantifiable properties of its component entities. Thus, each kind of molecule is represented by one or more variables (measuring *e.g.* concentration and location). Every kind of molecule (including those distinguished by a modification or activity state) is represented by separate variables. The interaction between molecules is abstracted by equations representing the spatio-temporal coupling of these population wide variables. The resulting methods use either continuous, mass-action differential equations or corresponding discrete, stochastic models to simulate and analyze molecular pathways.

While this approach is clear and well understood, it lacks in relevance and understandability, as it fails to deal with the biological objects directly [36]. Biological systems are composed of molecular objects, which maintain their overall identity, while changing in specific attributes, such as their chemical modification, activation state, or location. Chemical models, however, handle the cell as a a monolithic entity and are insusceptible to more structured descriptions that are typical to biological thinking [36]. Thus, each modified state of a molecule, a molecular complex, machine or compartment, must be handled as a distinct entity rather than a single entity with multiple states. Note, that the integration of **graphical user interface** capabilities (*e.g.* E-Cell [142] and STOCKS [71]), that are based on a "molecule-as-object" abstraction (below) to kinetic-model based tools compensates for some of these limitations. However, these capabilities are not built-in, but rather added as a secondary view.

Generalized models of regulation

Boolean network models use a "molecule-as-logical-expression" abstraction to describe and simulate gene regulatory circuits. These models were introduced by Kaufmann and subsequently applied to various biological systems (e.g.[116], [140], [115], [86], [139], [135], [2], [3], [29]). In a Boolean network, each biomolecule is represented by a logical variable which may assume one of two discrete values[1] (typically 0 when inactive and 1 when active). The effect of interaction between biomolecules on their activity state is abstracted by logical rules that determine the state of each biomolecule variable at time $t + 1$ based on its own value and the values of other biomolecule variables at time t.

While these approaches allow us to study general properties of large networks, they are limited in relevance (as they apply only to the regulation aspect of molecular systems), computability (as they have only limited predictive power) and extensibility (as, apparently, a Boolean abstraction of activation and inhibition suffices to handle only specific types of molecular systems).

Functional object-oriented databases

Pathway databases are based on the "molecule-as-object" abstraction.[2] They store information on molecular interactions (*e.g.* the DIP [152], BIND [11], and INTERACT [32] databases) and complete pathways (*e.g.* the MetaCyc [68], WIT [123], KEGG [95], CSNDB [61], aMAZE [143], and TransPath [151] databases). Each such database has a sophisticated object-oriented[3] *schema* that provides a biologically-appealing hierarchical view of molecular entities. In the most sophisticated cases (*e.g.* [143]), two classes of data objects are used to represent biomolecules and interactions, each object having detailed internal information (*e.g.* biomolecule name or chemical composition for the former, reaction name and rate for the latter). Relations serve to link objects to other objects (Figure 1.1A). For example, reaction objects can be linked to one or more biomolecule objects, representing the reactants and/or products of that reaction. Object taxonomies are used to describe the relationships between generic and specific biological entities or interactions as offering means of viewing the information at different levels of resolution, and dealing with incomplete knowledge. Most databases are equipped with querying tools of variable levels of sophistication, from simple queries to pathway reconstruction.

Functional databases provide an excellent solution for organizing, manipulating and (sometimes) visualizing pathway data. However, they provide little if any dynamic capabilities (*e.g.* simulation) and their querying tools are thus seriously limited. Recently developed **exchange languages** (*e.g.* [69], [93], and [35]) attempt to address these limitations by providing means to integrate models and tools from various sources. Most are XML-based markup languages (*e.g.* CellML and SBML). However, the utility of the language depends on its underlying pathway model, which is still primarily a kinetic one.

[1] In some models several (more than two) discrete values are allowed per variable

[2] While databases are primarily static repositories, and are thus distinct from the other approaches listed in this section, they do constitute a computational approach to pathways, in particular when considering formal representation languages, as databases require a formal schema for data representation. Furthermore, the more sophisticated cases discuss below are equipped with various querying tools rendering them closer to more analytic approaches.

[3] Object-orientation is an important design paradigm in computer science. Its basic underlying idea is the view of both information (*data*) and computational processes (*control*) as components of a unified functional entity, referred to as an *object*. The design of large object oriented systems is facilitated by a hierarchical framework, allowing objects to be defined as *instances* of abstract *classes* of growing specificity. A class definition consists of a set of attributes, which contain its data, as well as control operations, termed *methods*, by which this data may be manipulated. Objects are always defined as instances of particular classes, hence endowing them with both the attributes and methods of the original class. Classes may be defined as being more specific than other classes, thereby *inheriting* the more general class' attributes and methods, as well as overriding them or defining additional ones.

Abstract process languages applied to biomolecular systems

Approaches based on abstract process languages use the "molecule-as-computation" abstraction, and have gained increasing importance during the past few years. They were first proposed by Fontana and Buss [36], who employed the λ-calculus and linear logic as alternatives to dynamics systems theory for studying (bio)chemistry. Notable examples use existing formalisms for concurrent computation (often with a strong graphical component) to model real biomolecular systems.

The most comprehensive works used Petri Nets (e.g. [46], [45], [80], [81], [73], and [56]) for representation, simulation and analysis of metabolic pathways. Briefly, in a Petri net [45] there are places and transitions (Figure 1.1B). Each place (circle) represents a distinct molecular species. Places contain tokens, which represent the number of individual molecules. The number of tokens in a place is its marking. Each transition (rectangle) represents an elementary chemical reaction. Input and output functions (arrows) link places and transitions, and determine the stoichiometric coefficients (arrow labels) of the molecular species involved in the reaction. Input places (linked to a transition by an incoming arrow) represent reactants and output places (linked by an outgoing arrow from the transition) represent products. A transition is enabled when the markings of all of its input places are at least as great as the coefficients of their respective input arcs. Enabled transitions can fire, representing a single molecular reaction event where reactant molecules are removed and product molecules added according to the coefficients of the reaction. There are several variants of this basic language for computational or biological purposes (*e.g.* [45], [81]). For example, in stochastic Petri nets enabled transitions fire with an exponentially distributed time delay, with the rate parameter for each transition is given by the weight function. Despite their clarity and graphical convenience, Petri Nets essentially suffer from the same relevance problem as "dynamics systems theory": each molecule (and state or modification) is represented by an individual place, and there is no immediate use of more structured representation of complex molecular entities. Indeed, Petri nets are mostly used for the study of metabolic pathways, handled from a largely chemical perspective. Recent advances [81] try to overcome these limitations by combining Petri nets with object oriented representation.

Recent studies employed Statecharts ([57], [64], [65]) to build qualitative graphical models primarily for cellular systems with a molecular component. Unlike typical graph-based visualization of pathways, Statecharts provide a rich and expressive process language with clear semantics (Figure 1.1C). The model integrates two components. On the one hand, an *Object Model Diagram* represents the biological entities (molecules, cells, etc.) as object classes with various attributes and variables (rectangles), and shows the relations between them (*e.g.* composition of molecules and compartments to form cells). On the other hand, each biomolecular object may be equipped with a *statechart* that defines its behavior in a graphical way, as a change from one state (rectangle, *e.g.* active) to another (*e.g.* inactive), under specific conditions (denoted by labeled arrows and connectors). Interaction between biological entities is represented by communication between the statecharts of the respective objects. While highly expressive, statecharts are a primarily qualitative framework, and do not currently handle quantitative aspects in a direct and accessible manner. Statecharts are, however, extremely useful in describing complex multicellular scenarios, as indeed has been their primary biological application.

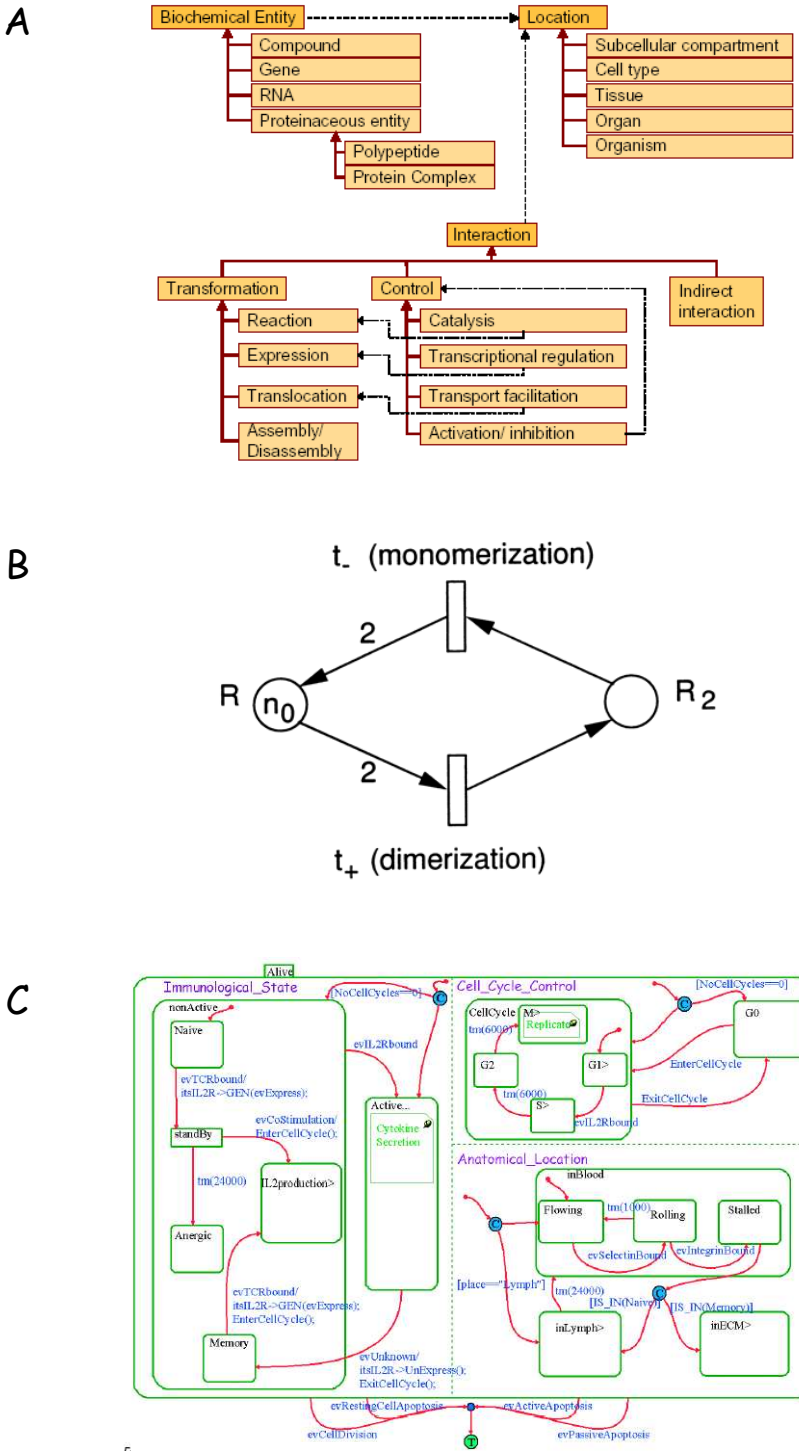

Figure 1.1: **Some models for biomolecular systems.** **A. An entity relation model for a database of biomolecular pathways.** Biological entities and reactions and their interactions are abstracted as database objects and relations. The biochemical entity class represents structural units. The interaction class groups interaction objects, representing transformation and regulatory reactions. The location class stores information on where the entities and interactions take place. Adapted from [143]. **B. A Petri net model of a biomolecular dimerization reaction**. The places R and $R2$ represent the monomer and dimer molecules, respectively. The transition t_+ represents dimerization and t_- represents dissociation. Adapted from [45]. **C. A statecharts simplified model of a T cell.** States are shown as rectangles and conditional state transitions as labeled arrows. Adapted from [64].

These recent attempts highlight the promise in using abstract process languages for pathway informatics. Our approach belongs to this last category, based on the "molecule as computation" abstraction. Starting with the π-calculus [89], an abstract process language, we develop throughout this work ways to represent and simulate biomolecular processes. In each chapter, we describe a particular set of steps taken to adapt, extend and implement this core language to conform to the unique requirements of biochemical systems. First, in this chapter, we present the core π-calculus as a framework for semi-quantitative modeling of molecular systems, emphasizing signal transduction ones. In Chapter 2 we develop a stochastic extension that allows accurate quantitative modeling of biochemical systems and use it to study models of glycogen biosynthesis and the circadian clock, highlighting the capabilities of the language to study compositional and modular systems. Finally, in Chapter 3 we further develop these modular capabilities by extending the language to handle compartmentalization. In each chapter we describe the language both formally and intuitively and provide a multitude of simple and complex examples to which it is applied.

1.1.2 Abstracting biomolecular systems as concurrent computation: An overview

An abstraction is a mapping from a real-world domain to a mathematical one, which preserves some essential properties of the real-world domain while ignoring other properties, considered superfluous. For example, in the "DNA-as-string" abstraction (a mathematical string is a finite sequence of symbols taken from a given alphabet) we capture the identity and the order of nucleotides in a DNA molecule with the symbols and the order of the string, but ignore higher-order structures and the actual chemical structure of each nucleotide.

Building the abstraction has three components (Figure 1.2).

1. **Informal organization of the knowledge on the real world domain.** In this step we chart out our knowledge and understanding of the scientific domain. We identify the essential different entities in this domain, their properties and behavior, and define an informal terminology to refer to them, using existing terminology and concepts whenever possible. For example, for DNA sequences, the notions of four different nucleotides and a primary DNA sequence can be considered as knowledge-entities of the real world. We carry out this organization step in a way believed to capture essential properties of the real world, thus facilitating the subsequent abstraction.

2. **Selection of a mathematical domain.** Concomitantly, we select a mathematical domain which we believe can be useful for the abstraction of the real world one. The selection of a particular mathematical domain is motivated by some informal correspondence between those properties or behaviors we deem essential in the real- world domain and those of the mathematical one. For example, the notion of a linear sequence is shared in both the real DNA world and the mathematical domain of strings.

3. **Designing and performing the abstraction.** This step is external to both the real world and the mathematical domain. First, we lay down the principles for the mapping: which entities, properties, operations and "events" in the mathematical domain represent which real-world ones. We also implicitly determine which entities will be "ignored" by the abstraction and excluded from the mapping. Note, that while the mathematical terms will be formally defined, the real-world ones are informal only. Second, using these pragmatic guidelines (*e.g.* a symbol to represent each

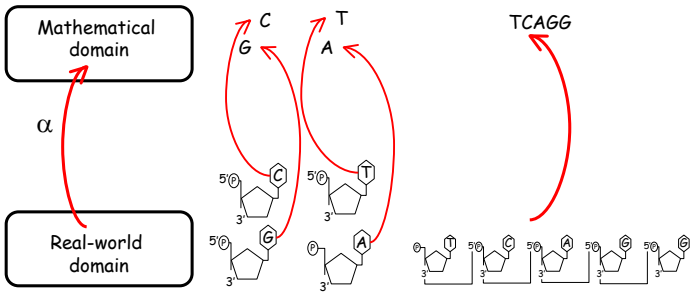

Figure 1.2: **Abstraction.** An abstraction is a mapping between a real-world domain (bottom), such as the world of DNA molecules, to a mathematical one, such as the world of strings over the alphabet {A,T,C,G} (top). The abstraction itself (middle) maps certain entities and their properties in the real-world domain to the mathematical one, while ignoring others (*e.g.* the chemical composition and higher-order structure of DNA sequences.)

nucleotide; a string represents a DNA sequence), we perform actual abstractions building specific representations of actual real world entities (*e.g.* a particular string represents a particular DNA sequence).

How do we judge whether the abstraction we built is a good one? As the abstraction is always partial and ignores certain elements of the real world domain, we must be careful in assessing both its utility and correctness. First, the various practical considerations made above (Section 1.1), such as the abstraction's relevance, computability, understandability, and extensibility, attempt to assess whether the resulting partial representation is useful. Second, we demand that the abstraction be correct [92], such that, informally speaking, whenever something is true in the abstract representation (of the real world), it is also true in the real world (but not necessarily vice versa).

The real-world domain: Essential properties of biomolecular systems

To build a useful and correct abstraction of biomolecular processes we first consider a simple biochemical system and identify its key components (in Sans Serif font). The toy system is composed of three types of protein molecules, protein A, protein B, and protein C (Figure 1.3A), each present in different quantities: We have 3 A, 5 B and 7 C protein molecules (Figure 1.3B). Each of the three proteins is composed of several domains: protein A has three domains (A1, A2, and A3), protein B has two domains (B1 and B2) and protein C has two domains (C1 and C2)(Figure 1.3C). A pair of A and B protein molecules may specifically interact with one another based on the molecular complementarity of specific motifs in the corresponding A1 and B2 domains. Alternatively, domain A1 may interact with domain C1, and domain A2 may interact with domain B1 (Figure 1.3D).

Different molecular events may take place over time in the system. For example, upon interaction between A and B, the A3 domain, which is a protein tyrosine kinase enzyme, modifies protein B, by phosphorylating a tyrosine residue in the B2 domain. Following this modification, the proteins dissociate. Protein A is reconstituted, while the modified B2 domain is now in a new state where it can be specifically recognized and bound by a complementary motif in the C1 domain (Figure 1.4). Other scenarios may evolve as a result of the alternative interactions possible in the system. For example, protein C may compete with protein B on interactions with protein A.

In this sketchy example, we identify several essential aspects of biomolecular systems:

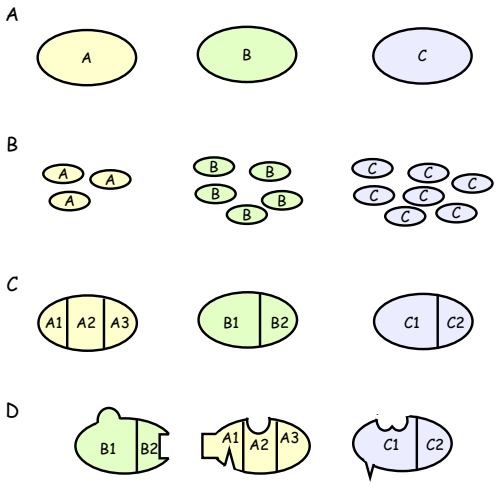

Figure 1.3: **A toy biomolecular system.** Ovals represent molecules, indentations and protrusions represent motifs.

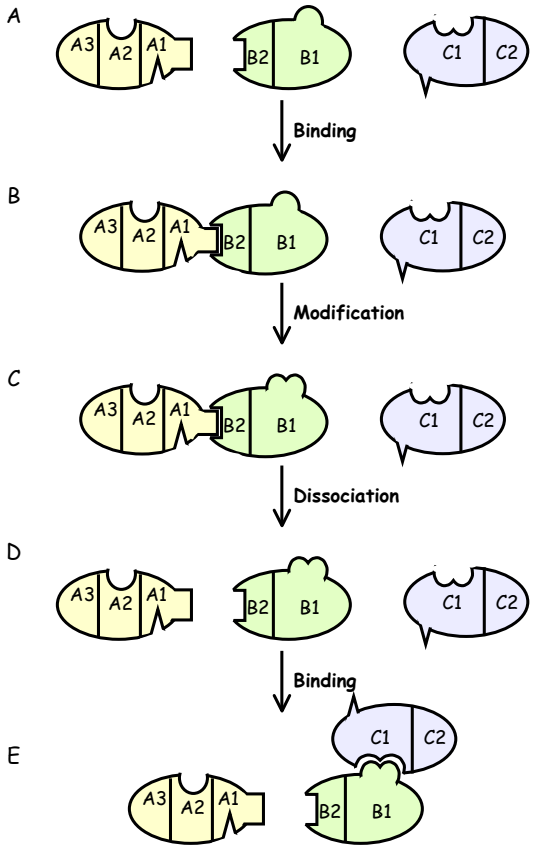

Figure 1.4: **A toy biomolecular system: Modification.** A. Three molecular species: A, B, and C, each with several sub-domains and motifs (protrusions and depressions). B. Domains A1 and B2 interact via complementary motifs. C. A motif on domain B1 is modified. D. The proteins dissociate E. The modified motif on B1 is used for interaction with domain C1.

- **Molecular species and populations.** Biomolecular systems are composed of a population of molecules. The molecular population may include multiple molecular species, with multiple copies of each molecule type.

- **Molecular composition.** Each protein molecule may be composed of several domains – independent structural and functional parts. Each domain may have one or more structural-chemical motifs.

- **Complementarity and specific interaction.** Different domains specifically interact through complementary motifs. Different motifs in domains allow for different specific interactions.

- **Interaction outcomes: Reconstitution, modification and change in molecular state.** One or more things can happen to molecules following interaction. First, the molecule (*e.g.* an enzyme) may be reconstituted without any change. In other cases, the molecule may be changed following interaction. The change may be attributed to a specific chemical modification of one of the motifs in the molecule by its interaction partner, or may be regarded as a more general change in the state of the molecule, such as a change from an inactive to an active state. In each state the molecule may potentially participate in different interactions.

- **Alternative interactions.** In many cases a molecule may be able to participate in more than one interaction. The alternative interactions may be independent. For example, protein A may interact with protein B on one domain and with protein C on another, such that one interaction does not preclude the other. In other cases, the same domain may have several potential interaction partners, which compete with each other.

The mathematical domain: Concurrent computation

In the "molecule-as-computation" abstraction, a system of interacting molecular entities is described and modeled by a system of interacting computational entities. Several abstract computer languages such as Petri nets [112], Statecharts [50], concurrent logic programming [125], and the π-calculus [89], were developed for the specification and study of such systems.

Computational processes are the basic interacting entities of these languages. Each process has an internal state and interaction capabilities. Programs written in the language define the processes' potential behaviors. Computation begins with a system of concurrent processes, each in a well-defined state. General reaction rules determine the conditions under which processes may interact with one another. As the system unfolds, the actual behavior of each process is governed by the program defining it, by the general reaction rules that specify how processes interact with one another, and by the other processes in the system available for interaction. Interactions affect the interacting processes. The effect can include a state change, a change in interaction capabilities, and/or sending messages. Complex entities are described hierarchically - for example, if A and B are two processes, then $A \mid B$ (read: "A parallel B") is a complex process consisting of both. Similarly, $A + B$ (read "A choice B") is a complex process that can behave as either A or B depending on its interaction with other processes.

We focus on the π-calculus language. π-calculus programs specify networks of *communicating* processes. Communication occurs on complementary *channels*, that are identified by specific names (Figure 1.5). The programs describe the potential behavior of processes: what communications the processes can participate in on such channels.

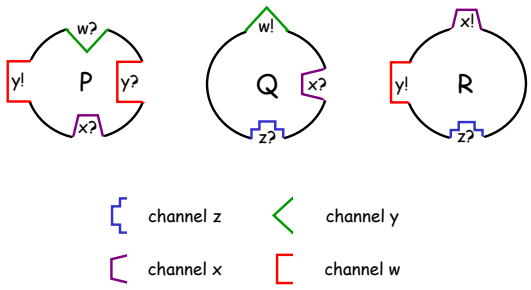

Figure 1.5: **π-calculus processes and channels: An intuitive view.** Three processes, P, Q, R (ovals) with four communication channels (complementary shapes of protrusions and depressions), occuring in parallel.

There are two different kinds of communications in the π-calculus. In the first type (also present in simpler languages, such as CCS [89]), the processes only send *alerts* (or empty *"nil"* messages) to one another. In this case (depicted in Figure 1.6A), a sender process may communicate with a receiver process if they share the same communication channel. Following communication, the action is removed, and each process may either iterate or change its state, becoming another process with different channels and behavior.

The second type of communication is more complex and unique to the π-calculus. In this case (Figure 1.6B), the sender process sends a *message* to the receiving process. The content of this message is one or more channel names. These channels can be used by the receiving process to communicate with other processes. As before, following communication the processes may either iterate or change state. However, passing channels as messages allows the process (and its continuations) to acquire communication capabilities dynamically, that were not specified *a priori* in its explicit program. Such a change in future communication capabilities as a result of passing messages is termed *mobility*.

As shown in Figure 1.6 the two communication capabilities may serve to describe similar systems. However, in the former mechanism the communication capabilities of a process are all strictly pre-defined (and thus may change only as a result of a pre-specified state change), while in the latter messages may change the communication capabilities of a process dynamically. Thus, messages allow us to leave certain components under-specified, to be determined only following interaction with different processes which may send different messages along the same channel.

The molecule-as-computation abstraction

We find an intuitive correspondence between the world of computational processes and of biomolecular systems, based on the representation of molecules as the processes in which they may participate. Note, that this view is essentially a biological one. Biologists typically characterize molecules by *what they can do*. For example, enzymes are named by the reaction that they can catalyze, and binding domains in proteins – by the entities which they bind.

To illustrate this, we return to our toy system and use it to develop guidelines to an abstraction from biomolecular systems to the π-calculus (Table 1.1). In this section we give a general intuition to this abstraction, while in the next one we build the abstraction step-by-step. In each step we map a biomolecular entity (in **bold** type face) to a π-calculus one (in Sans Serif type face).

The first entity we identified in the real-world domain of biomolecular systems was the **molecular**

Biomolecular entity	**Example**	**π-calculus entity**	**Example**
Molecular species	Protein A	Process species	*ProteinA*
Molecular population	5 molecules: 2 protein A and 3 protein B	System of concurrent processes	5 parallel processes: 2 *ProteinA* processes and 3 *ProteinB* processes
Complementary motif types	Complementary binding motif	Complementary input and output occurrences on the same channel name	*bind*? and *bind*! complementary input and output occurrences
Motif occurrence in molecule or domain	Binding motif in protein A	Communication offers on channels in process	Input offer *bind* ? [ ] in process *ProteinA*
Biomolecular event	**Example**	**π-calculus event**	**Example**
Specific interaction on complementary motif	Protein A and B interact on a binding motif	Specific communication on complementary channels	Communication on the *bind* channel between the *ProteinA* process (input offer *bind* ? [ ]) and the *ProteinB* process (output offer *bind* ! [ ])
Outcome following interaction		Communication prefix preceding process or other communication	
Reconstitution following interaction	Protein A reconstituted after interaction with protein B	Communication prefix preceding process recreation	Input communication prefix on the *bind* channel, preceding recreation of the *ProteinA* process
Changed molecular state following interaction	Protein B activated after interaction with protein A	Communication prefix preceding creation of new process	Output communication prefix on the *bind* channel, preceding creation of the *ActiveProteinB* process
Modification of motifs during interaction	Protein A phosphorylates a tyrosine motif on protein B	Non-mobile approach: re-creation of a different type of process with a different channel set	Following communication, the *ProteinB* process is replaced by a newly created *PhosphoB* process, that uses a *phosphotyrosine* channel rather than a *tyrosine* channel
		Mobile approach: Message (a tuple of channel names) sent in communication to replace channels in the receiving process	A *phosphotyrosine* channel sent from the *ProteinA* to the *ProteinB* process, and replaces the *tyrosine* channel in the recreated *ProteinB*

Table 1.1: **Guidelines for the abstraction of biomolecular systems to the π-calculus.**

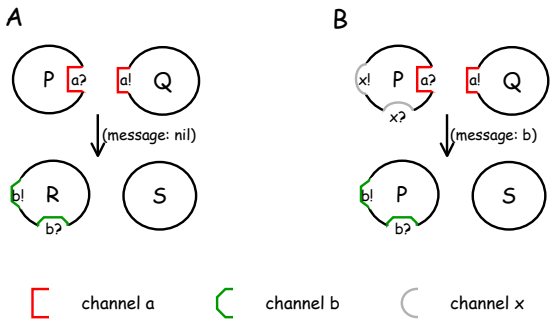

Figure 1.6: **π-calculus communication: A schematic view.** A. **Communication with state change**. Two processes, P and Q (ovals), share a communication channel (a). Q is willing to send on the channel (protrusion), while P is ready to receive (complementary depression). Following communication, the processes undergo a state change, into S and R, with different channels and behavior. B. **Communication with mobility.** Q is willing to send a *message* on channel a, while P is ready to receive a message. The message Q is sending is the channel name b. P is willing to receive this into the channel placeholder x. Following communication, P's channels have changed, even without explicit *a priori* specification, as the place holder x was replaced by the received channel b, which may be used, in principle, as either input, output or message.

species. We will abstract those as types of computational processes in the mathematical domain of the π-calculus, following the notion that the essential aspect of molecular identity we wish to capture is its potential behavior. Similar to molecular species (*e.g.* protein A, molecule B), computational process species are named (*e.g.* $ProteinA$, $MoleculeB$, Figure 1.7A).

Biomolecular systems are composed of a **population of molecules**, in which multiple actual copies of molecules from each molecular species may be present. We will abstract these as a system of parallel processes, each process representing one molecular copy. Thus, a biomolecular system with 3 protein A molecules and 2 molecule B molecules, can be abstracted as a computational system with 3 $ProteinA$ and 2 $MoleculeB$ parallel (or concurrent) processes (Figure 1.7B).

Each **molecular species** may be composed of several independent structural/functional parts, such as domains in protein molecules. We abstract the independent parts as parallel sub-processes of the process corresponding to the molecular species. Thus, if $DomainA$ and $DomainB$ are processes abstracting two domains , then $DomainA \mid DomainB$ (read "$DomainA$ parallel $DomainB$") is the abstraction of the protein C molecule, which is composed of the two domains (Figure 1.7C).

Each molecule (or its sub-parts) may have one or more structural-chemical **motifs**. The motifs we focus on are those that allow the molecule to specifically interact with other molecules. Such interaction is based on the **complementarity** of motif pairs. We distinguish between the **motif type** and it actual **presence** in a molecule. We abstract complementary interaction motif types to complementary (input and output) occurrences on a communication channel in the π-calculus. Thus, the abstraction of a particular **binding motif** is the *bind*! output occurrence on the *bind* channel, while the motif it is complementary to is abstracted as the complementary *bind*? input occurrence on the same channel (Figure 1.7D).[4]

The actual **presence of a motif in a molecule** (allowing it to interact with other molecules, harboring the complementary motif) is abstracted as a communication offer on the channel in the corresponding process[5]. Thus, an input communication offer on the *bind* channel (*bind* ? [ ], read: "input nil on bind")

[4]In the original π-calculus the input occurrence is denoted simply by the channel name *e.g.* x, while the output occurrence is denoted by the co-name *e.g.* $\overline{x}$. Our notation distinguishes both the input (x?) and the output (x!) occurrence from the channel name (x). This is done for purely technical reasons.

[5]In the original π-calculus it is customary to term such capabilities as communication actions. As the term may be

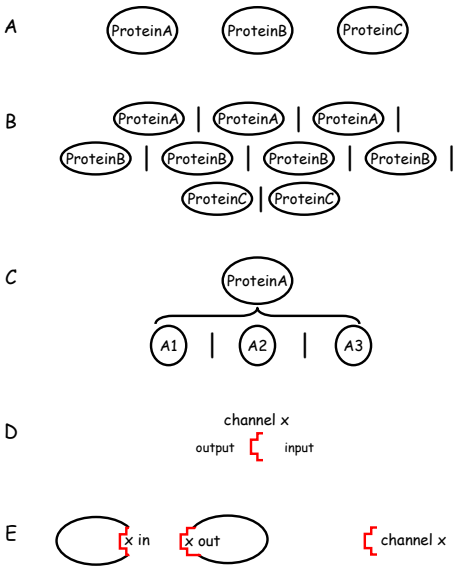

Figure 1.7: **Molecules as computation: Processes and channels.** A. **Molecular species as process species.** Each process is denoted by a circle, with the process name inside. Each molecular species is abstracted by a different process type, with a different name. B. **Molecular population as system of concurrent processes.** Each copy of a particular molecule is abstracted by a concurrent (or parallel) copy (instance) of the corresponding process species. C. **Domains as sub-processes.** Each molecular species may be composed of several domains, abstracted as sub-processes of the process abstracting the molecule. D. **Complementary motif types as complementary occurrences on communication channels.** Each complementary motif pair type is abstracted as a channel name. One of the pair is represented by the input occurrence (depression) and the other by the output occurrence (protrusion). E. **Motif occurrences in molecules as communication offers in processes**. The presence of a particular motif type in a molecule or domain is abstracted as a communication offer (input or output) by the corresponding process (protrusion or depression on the circle's surface).

in the *DomainA* process is the abstraction of the presence of a binding motif in domain A, ready to interact with complementary motifs in other domains or molecules (Figure 1.7E).

Once we abstract the basic entities of the biomolecular domain – molecule, domains and motifs – to the π-calculus domain – as processes, channels and communication actions – we now have to see whether the **behavior of molecules** in biomolecular systems can be correctly abstracted by the behavior of processes in the π-calculus realm.

The basic event in a biomolecular system is a **specific interaction between molecules through complementary motifs**. This event is abstracted by specific communication between processes with complementary communication offers. Thus, if protein A and protein B have complementary motifs they can interact with each other. Similarly, if the *ProteinA* and *ProteinB* processes (abstracting proteins A and B) offer complementary input and output communications on the *bind* channel (abstracting the complementary binding motifs), they can communicate (Figure 1.9A).

Biomolecular interaction may lead to several outcomes: **reconstitution**, **modification** and/or **a change in molecular state**. In each of these cases we assume the interaction **precedes** the outcome. We abstract such an order of events in the π-calculus by a prefix order on communication actions and processes. An interaction **followed by** the **reconstitution** of the interacting molecule is abstracted as a communication prefixing the re-recreation (or re-introduction) of a process of the same type (abstracting the behavior of the same molecule). Similarly, an interaction **followed by** a **change in molecular state** is abstracted as a communication prefixing the creation of a process of another type (abstracting

confused with the actual communication event we will avoid it and replace it with communication offers.

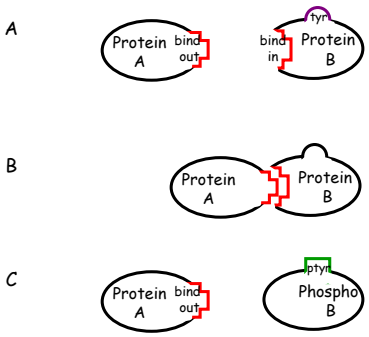

Figure 1.8: **Biochemical interaction as communication and state change.** A,B. **Interaction as communication.** *ProteinA* and *ProteinB* processes (abstracting proteins A and B) have complementary input and output communication offers on the *bind* channel (abstracting the complementary binding motifs), allowing them to specifically communicate with each other. C. **Modification as state change.** Following the communication prefix, the process abstracting the modified molecule is replaced by a created process of a different type, *PhosphoB*, with the *phosphotyrosine* channel. The other process is recreated, abstracting the reconstitution of the tyrosine kinase.

the behavior of the molecule's new state). Finally, an interaction between two molecules leading to the **modification of the motif(s) of one molecule by the other** is handled in one of two ways. First, we may consider it as a special case of molecular state change. In this view the modified molecule is in a new state, and the modified motifs are represented by the different channels used by the process type representing this state. As a simple example (Figure 1.8), consider the interaction between two proteins, A and B, that can interact on a binding motif, resulting in the phosphorylation of a tyrosine motif in B by A, and the release of a modified B protein and a reconstituted A protein. In the abstraction, *ProteinA* and *ProteinB* are the processes representing A's and B's behavior, and complementary communication offers on a *bind* channel abstract their ability to interact on the binding motif. The concomitant phosphorylation of a tyrosine in protein B is abstracted by the creation of a new process of type *Phosphorylated_B* that offers input communication on a *phosphotyrosine* rather than on a *tyrosine* channel. The communication in the *ProteinA* process prefixes the recreation of a copy of the *ProteinA* process, abstracting the reconstitution of the corresponding molecule.

Alternatively, we may maintain the molecular identity of modified proteins by maintaining the corresponding process identity throughout communication. To do this we employ the mobility mechanism of the π-calculus, and abstract **state change and modification** as a message sent from one of the processes to the other one. The message constitutes of a tuple of channel names which the receiving process substitutes for other channel names. These received channel names can be used in the process created (or recreated) after the communication prefix, abstracting its modified state. For example, in the phosphorylation system described above, the phosphorylation of a tyrosine motif in protein B is abstracted as a message, constituting of the *phosphotyrosine* channel name, sent from *ProteinA* and received in *ProteinB* into the *tyrosine* channel. Following communication, rather than recreating a different process, the *ProteinB* process is recreated, but with the newly received *phosphotyrosine* channel replacing the *tyrosine* one. This abstracts the release of a modified protein B molecule, in which the tyrosine motif has been modified by phosphorylation. This example scenario is schematically depicted in Figure 1.9.

Each of the two abstractions for molecular interactions has benefits and drawbacks. In the non-mobile approach, the abstracted system is fully specified, and thus often simpler to write and follow. The mobile

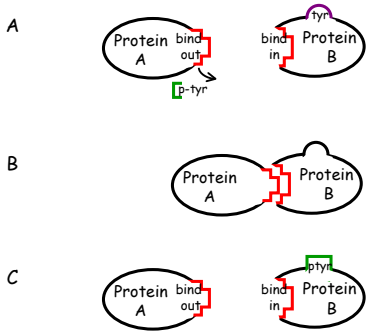

Figure 1.9: **Biochemical interaction as mobile communication between processes.** A. **Interaction as communication.** *ProteinA* and *ProteinB* processes (abstracting proteins A and B) have complementary input and output communication offers on the *bind* channel (abstracting the complementary binding motifs), allowing them to specifically communicate with each other. B,C **Modification as sending messages.** The process abstracting one molecule sends a message (abstracting the modification) to the process abstracting another molecule. The receiving process (abstracting the modified molecule) has a new received channel (abstracting the modified motif) that can be used to communicate with other processes.

approach is more complicated, but also closer to biological reasoning: the same process continues to abstract molecules throughout modification, and message passing corresponds directly and specifically to the modification event.

In some cases a molecule may be able to participate in more than one interaction. When the alternative interactions are independent, we abstract them as communications in several parallel sub-processes of the molecule, as was described above for domains. In other cases, the same domain may have several potential interaction partners, which compete with each other. In such cases we abstract them as mutually exclusive choices between several communication actions within the same process. For example, if a domain may participate in one of two competing binding interactions - one on a agonist binding motif and the other on an antagonist binding motif, we will abstract its behavior as a *Domain* process with a *choice* between two communications: one on the *agonist_bind* channel and the other on the *antagonist_bind* channel.

Following these guidelines (Summarized in Table 1.1) we can now (informally) abstract our toy system (Figures 1.3 and 1.4) into the π-calculus. We abstract the three **molecular species**, protein A, protein B, and protein C (Figure 1.3), as three corresponding process species, $ProteinA$, $ProteinB$, and $ProteinC$. We abstract the 3 A, 5 B and 7 C **protein molecules** as 3 copies of the $ProteinA$ process, 5 copies of the $ProteinB$ process, and 7 copies of the $ProteinC$ process. The protein A species, **composed of three domains** (A1, A2, and A3) is abstracted as the $ProteinA$ process species with three parallel sub-processes: $DomainA1$, $DomainA2$, and $DomainA3$. Similarly, protein B is abstracted as $DomainB1$ in parallel to $DomainB2$, and protein C as $DomainC1$ in parallel to $DomainC2$. The **interaction capabilities** of the different domains are abstracted as communication actions on channels. Domain A1's ability to interact with Domain B2 on one motif or with domain C1 on another is abstracted as a choice between two communication actions in the $DomainA1$ process. Complementary (corresponding) communication actions are present in the $DomainB2$ and $DomainC1$ processes. Similarly, the potential interaction between domain A2 and domain B1 is abstracted as communication actions in the $DomainA2$ and $DomainB1$ processes.

The different molecular **events** in the biomolecular toy system are abstracted by corresponding events in the computational one. For example, **Upon interaction** between Protein A and B, the A1 domain,

which is a protein tyrosine kinase enzyme, **modifies** protein B, by phosphorylating a tyrosine **residue** in the B2 domain. **Following** this modification, the proteins dissociate. Protein A is **reconstituted**, while protein B is released with a modified motif. In the π-calculus abstraction, the two corresponding processes, $DomainA1$ and $DomainB2$, communicate, and a message, containing the $p{-}tyr$ channel, is sent from $DomainA1$ to $DomainB2$. The received channel substitutes the tyr channel. Following the communication prefix, $DomainA1$ and $DomainB2$ are re-created, the latter with the newly received $p{-}tyr$ channel. In the toy system, the **modified motif** (phosphorylated tyrosine) of the B2 domain can now interact with a complementary motif in the C1 domain. In the abstracted representation, the $p{-}tyr$ channel can be used by the $DomainB2$ process to communicate with the $DomainC1$ process. In the next section, we will see how we formally write such abstractions in the π-calculus.

1.2 Biomolecular systems in the π-calculus

To fully describe the abstraction of biomolecular process in the π calculus, we now turn to a series of biological examples, which we will abstract step by step using the various constructs and rules of the π-calculus. A formal presentation of the calculus is given in the next section. Our examples are derived from a model of a canonical signal transduction pathway starting from a receptor tyrosine kinase (RTK) on a cell's membrane, and ending with transcriptional activation of immediate early genes. This pathway is schematically depicted in Figure 1.10, and will be studied in detail in later sections.

1.2.1 Molecules and domains as concurrent processes

We abstract each biomolecular system as a process, denoted by a capitalized name, *e.g.* P. For example, the RTK-MAPK signal transduction pathway is a system, and will be denoted as

$$RTK\_MAPK\_pathway$$

Each constituent molecule in the pathway is a process, too, as are its domains. For example, the growth factor unbound ligand, receptor tyrosine kinase, and Ras molecules will be denoted as $Free\_Ligand$, RTK, and Ras, respectively. Similarly, the extracellular, intra-cellular and transmembranal domains of the receptor molecule will be denoted by $Extra$, $Transmem$, and $Intra$ processes.

We next define a system process as a collection of molecule processes, occuring in parallel. This concurrency is denoted by the PAR operator, |, inserted between the process names. For example,

$$RTK\_MAPK\_pathway ::=$$
$$Free\_ligand \mid RTK \mid RTK \mid RTK \mid Ras \mid Ras \mid Ras \mid Ras\ .$$

denotes that the RTK-MAPK pathway is composed of several ligand, RTK, and Ras molecules.[6] Note, that the number of molecules of each types is specified directly by the number of corresponding processes. In the above example we have one ligand molecule, three RTK molecules and four Ras molecules.

[6]Read: "$RTK\_MAPK\_pathway$ is defined as $Free\_Ligand$ parallel RTK parallel ..."

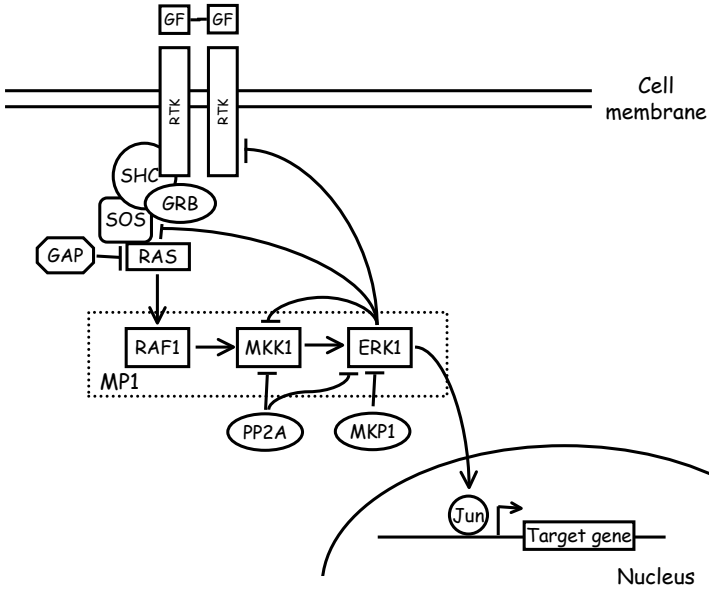

Figure 1.10: **A schematic view of the RTK-MAPK signal transduction pathway.** A protein ligand molecule (GF), with two identical domains, binds two receptor tyrosine kinase (RTK) molecules on their extracellular part. The bound receptors form a dimeric complex, and cross-phosphorylate and activate the protein tyrosine kinase in their intra-cellular part. The activated receptor can phosphorylate various targets, including its own tyrosines. The phosphorylated tyrosine is identified and bound by an adaptor molecule, Shc. A series of protein-protein binding events follows, leading to formation of a protein complex (Shc, Grb2, Sos, and Ras) at the receptor intra-cellular side. Within this complex, the Sos protein activates the Ras protein, which in turn recruits the serine/threonine protein kinase, Raf, to the membrane, where it is subsequently phosphorylated and activated. A cascade of phosphorylations-activations follows, from Raf to Mek1 to Erk1. This cascade culminates in activation of the threonine and tyrosine protein kinase, Erk1. Activated Erk1 translocates to the nucleus, where it phosphorylates and activates transcription factors, such as AP-1, leading to *de novo* gene expression.

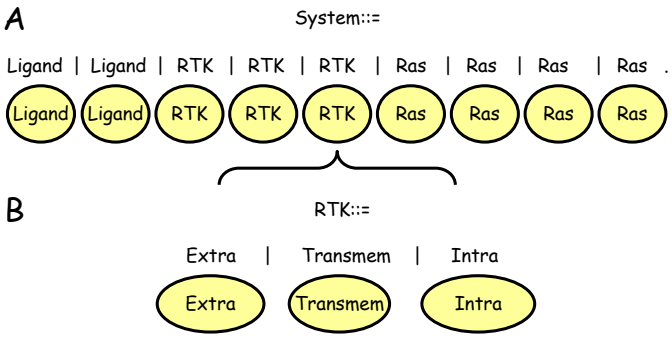

Figure 1.11: **Pathways, molecules and domains as concurrent processes.** A. The $RTK\_MAPK\_pathway$ process is composed of multiple parallel copies of the $Free\_ligand$, RTK, and Raf processes. B. Each RTK process is composed of parallel copies of the $Extra$, $Transmem$, and $Intra$ processes.

Analogously, we define a protein molecule as several domain processes composed in parallel. For example, we represent a homo-dimer ligand composed of two identical binding domains, and a receptor composed of intra-cellular, transmembranal and extracellular domains by the corresponding π-calculus definitions:

$$Free\_ligand ::= Free\_binding\_domain \mid Free\_binding\_domain \ .$$
$$RTK ::= Extra \mid Transmem \mid Intra \ .$$

This representation scheme for pathways, molecules and domains as concurrent processes is graphically depicted in Figure 1.11.

1.2.2 Molecular complementarity as communication channels

Two molecules (or domains) interact with each other based on their structural and chemical complementarity [99]. Interaction is accomplished by the motifs and residues that constitute a domain and serve as "communication ports" for the molecule. We abstract molecular complementarity as complementary offers on communication channels. One side[7] of the interacting pair is represented by the input offer, denoted by *e.g.* $x\ ?\ [\,]$, while the other is the output occurrence, $x\ !\ [\,]$. An interaction event may occur only when such a complementary pair of offers is shared by the interacting processes (see below). For example, if the ligand's free binding domain and the receptor's extracellular domain have complementary binding motifs, we denote this motif pair as a *ligand_binding* channel, which appears as an output $ligand\_binding\ !\ [\,]$ offer in the ligand's $Free\_binding\_domain$ process and as an input $ligand\_binding\ ?\ [\,]$ offer in the receptor's $Extra$ process:[8]

$$Free\_binding\_domain \quad ::= \quad ligand\_binding\ ![\,] \cdots$$

[7]The selection of who is selected as the input partner is a matter of convention in the non-mobile approach, and depends on the direction of modification in the mobile one, as presented below.

[8]For example, read: "$Free\_binding\_domain$ is defined as send nil on the *ligand_binding* channel..."

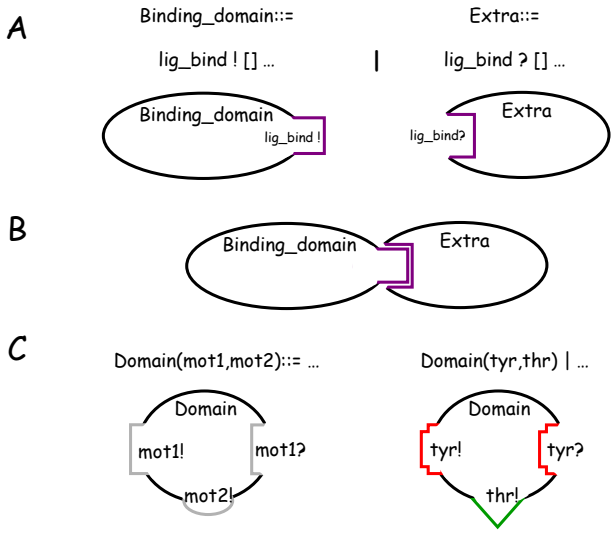

Figure 1.12: **Molecular complementarity as input and output channels.** A. The *Free_binding_domain* and the *Extra* processes (ovals) have complementary output (protrusion) and input (depression) offers on the *ligand_binding* channel. B. Process (molecular) interaction is enabled by channel (motif) complementarity. C. We may define a process with channel parameters (left, grey), to be instantiated with actual channels (right, color) only when the process is created.

$$Free\_extracellular\_domain \quad ::= \quad ligand\_binding\ ?[\,] \cdots$$

Molecules (processes) with complementary motifs (communication offers) are graphically depicted in Figure 1.12, where motifs (channels) are shown as shapes (*e.g.* circles, rectangles), which appear as protrusions (output) and intrusions (input) on the molecule's (process's) surface.

The motifs on molecules may often vary based on different modifications or states. For example, a binding motif may change its shape and interaction capabilities based on the phosphorylation state of a tyrosine residue. We often abstract such motifs as channel parameters of the process. The parametric definition of the molecule or domain process is general, and becomes specific only when the process is created with actual channels. For example, the intra-cellular domain of the receptor may have three potential binding motifs, which we will represent as parameters when defining the process:

$$Free\_intracellular\_domain(motif1, motif2, motif3) ::= \cdots$$

Then, when we create the $Free\_intracellular\_domain$ process as part of a particular RTK, we replace these parameters with actual channels, representing the specific state for those motifs, *e.g.*[9]

$$RTK ::= \quad Free\_intracellular\_domain(p\_tyr1, tyr2, sh2)\ |\cdots$$

Molecule processes with motif channel parameters are graphically depicted in Figure 1.12C, where the parameter motifs (channels) are shown in grey and their actual instantiation in color.

[9]Read: "RTK is defined as $Free\_intracellular\_domain$ with $p\_tyr1, tyr2$ and $sh2$ parameters ..."

1.2.3 Sequential and mutually exclusive events

Biochemical events may occur in sequence, in parallel with other independent events, or in a mutually exclusive, competitive fashion. We abstract a sequence of interactions as a sequence of input and output offers, separated by a prefix operator: **","** (the comma sign). For example, if the extracellular domain may first bind a ligand, and then bind to another domain of another molecule, we define[10]

$$Extra ::= ligand\_binding\ ?\ [\ ],\ rtk\_binding\ ?\ [\ ],\ \cdots$$

In other cases, there may be several alternative interactions in which a molecule may participate: once one occurs, the other options are no longer possible. Such mutually exclusive offers are summed together, using the **"+"** or **";"** choice operator[11]. For example, if the receptor may be able to bind either the (agonist) ligand or an antagonist but not both, we write[12]

$$\begin{aligned} Extra \quad ::= \quad & ligand\_binding\ ?[\ ],\ rtk\_binding\ ?[\ ],\ \cdots + \\ & antagonist\_binding\ ?[\ ],\ \cdots \end{aligned}$$

Finally, several interactions may occur in parallel, without directly affecting each other. For example, a protein molecule may have several domains, each capable of independent interactions. Such cases will be handled by composing different offers in parallel, as shown above.

1.2.4 Compartmentalization as private channels

A pathway is not merely a bag of molecules and their domains. Rather, it is composed of defined compartments. First, the parallel domains of a single molecule are linked together by a single backbone. Then, distinct multi-molecular complexes form. Finally, molecules are separated into higher-order cellular compartments, often bounded by membranes. In all three cases molecules which share a common compartment may interact with each other, while molecules excluded from the compartment may not [55]. Thus, rather than representing compartments directly[13], we abstract only the limits they impose on communication by introducing "private" channels with restricted communication scopes.

A private channel x is created using the "new x" operator. For example, if we wish to represent the fact that the three domains of a receptor are linked by a common backbone and belong to a single molecule (Figure 1.13), we introduce a new *backbone* channel, that is shared by the three processes, and is distinct from all other (external) channels.[14]

$$RTK ::= (\text{new } backbone)(Extra|Transmem|Intra).$$

The private *backbone* channel can be used for communication only between these three processes, and

[10] Read: "*Extra* is defined as receive nil on the *ligand_binding* channel and then receive nil on *rtk_binding* ..."

[11] We will use both notations interchangeably. For their syntactic distinction, see the implementation section.

[12] Read: "*Extra* is defined as receive nil on the *ligand_binding* channel and then receive nil on *rtk_binding* **choice** receive nil on *antagonist_binding* ..."

[13] The reader is referred to Chapter 3 for an extension that handles compartments directly.

[14] Read: "new *backbone* channel in *Extra*, *Transmem* and *Intra* processes"

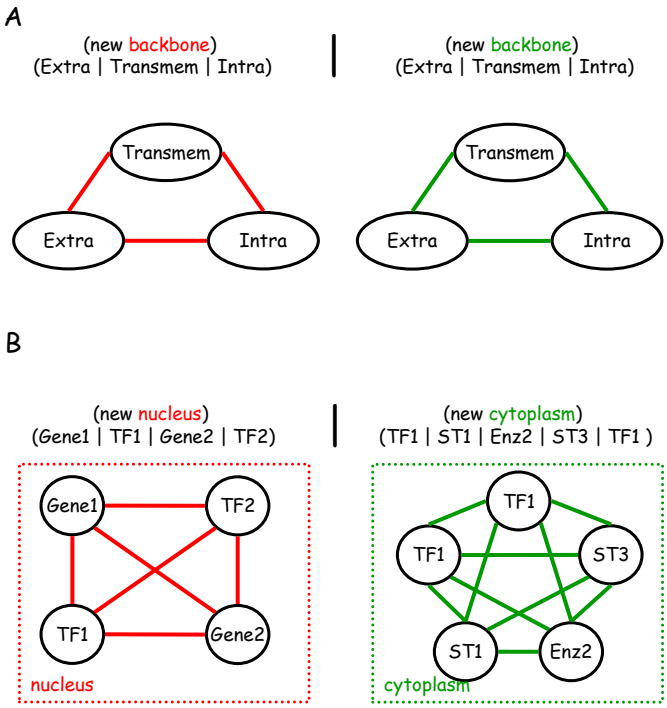

Figure 1.13: **Compartments as private channels.** A. Three domains (processes) linked by a shared private channel. B. A membrane bound compartment represented by a series of private channels.

thus represents the limits that a shared backbone poses on interaction. Whenever a new instance of an RTK process is created, a fresh *backbone* channel is created, too. This channel is distinct from all others and will can be used only by the sub-processes of that particular RTK copy. Note, that the presence of a private channel allows private interactions between the sub-processes, but does not exclude "global" interactions with other processes on the usual "public" channels.

1.2.5 Interaction and biochemical modification as communication

Each biochemical interaction in a system may affect subsequent interactions in several ways. First, interaction is often accompanied by modification of one molecule by the other. For example, when a protein tyrosine kinase interacts with a substrate protein, it phosphorylates a tyrosine residue on the substrate, leaving it in a modified state. Such a modification may change the structural and chemical characteristics of the modified motif, such that it may now be complementary to (and interact with) other motifs, which it was not "suitable for" before (and vice versa, become non-complementary to other motifs, with which it could interact before). Second, a particular interaction may lead to the exclusion of other possibilities, as in the case of an antagonist binding to a receptor and by this precluding the (previously possible) binding of an agonist. Finally, an interaction may lead to a general "state change" in both interacting molecules, enabling events that were disabled before.

We will represent these three results of interaction (and modification) by the communication rules of the π calculus, using either either a non-mobile or a mobile (message passing) abstraction.

Non-mobile communication

In the abstraction of interaction as non-mobile communication we represent modification and state change together by the introduction of a newly created (or re-created) process into the system. The different channel set owned by the created process will represent the modified residues in the corresponding molecule. For example, consider an active protein kinase (*Active_kinase* process) and a target binding domain (*Mod_Bind_domain* process). The binding domain has a motif (*phosph_site* ?) that may be identified and bound by a complementary motif (*phosph_site* !) in the kinase. The kinase may then phosphorylate a tyrosine residue in another motif in the binding site. This modification is represented by the creation of a new instance of a different process type (*Phospho_Bind_domain*), that uses the *phosphotyrosine* channel rather than the *tyrosine* one.

$$\begin{aligned} Active\_kinase &::= phosph\_site\ !\ [\ ]\ ,\ Kinase \\ Mod\_Bind\_domain &::= phosph\_site\ ?\ [\ ]\ , Phospho\_Bind\_domain \\ Phospho\_Bind\_domain &::= phosphotyrosine\ !\ [\ ]\ ,\ \cdots \end{aligned}$$

If an interaction takes place between the *Active_kinase* and the *Mod_Bind_domain* processes, two things will happen simultaneously. First, both the input and output events on the *phosph_site* channel will be removed. Second, the remainder of the two processes will be allowed to continue: creating a *Kinase* process and a *Phospho_Bind_domain* process. The latter one uses a the *phosphotyrosine* channel, representing the modification. Overall, the non-mobile communication process can be summarized in the following reduction:

$$\begin{aligned} phosph\_site\ !\ [\ ]\ ,\ Kinase \quad | \quad phosph\_site\ ?\ [\ ]\ ,\ Phospho\_Bind\_domain \\ \longrightarrow \\ Kinase \quad | \quad Phospho\_Bind\_domain \end{aligned}$$

The newly created *Phospho_Bind_domain* can use its *phosphotyrosine* channel for further communications with other processes which harbor this motif, such as the $SH2$ process:

$$SH2 ::= phosphotyrosine\ ?\ [\ ]\ ,\ \cdots$$

Thus, the creation of different processes with different channel sets represents the actual chemical modification of a motif in the binding site and the effect of this modification on subsequent interaction. The full sequence of events is summarized in Figure 1.14.

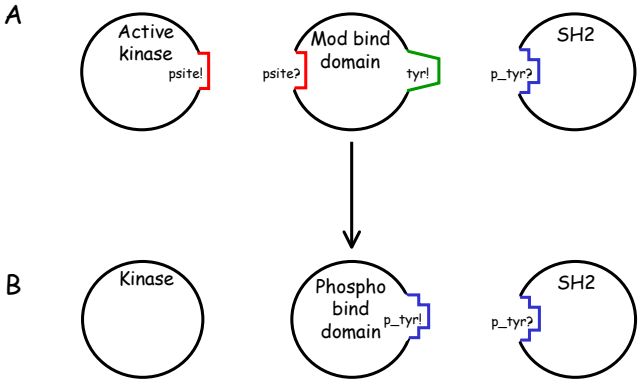

Figure 1.14: **Interaction and modification as communication and state change.** A. Pre-communication. Three processes and their respective channels: An *Active_kinase* process (with a *phosph_site* channel), a *Mod_Bind_domain* process (with *phosph_site* and *tyr* channels), and an *SH2* process (with a *p_tyr* channel). The *Active_kinase* and the *Mod_Bind_domain* may interact on the *phosph_site* channel. B. Post-communication. A *Phsopho_Bind_domain* process is introduced into the system instead of the *Mod_Bind_domain* process. It possesses a *phosphotyrosine* channel which may now be used for communication with the *SH2* process.

Mobile communication

In the mobile abstraction we represent modification as sending and receiving messages. Each message is composed of one or more channel names, representing the modified motif(s). These names are received into "placeholders" in the receiving process which they substitute, representing the modification. Then, they may be used by the receiving process in subsequent communication, representing the effect of modification on subsequent interactions. For example, consider again the active protein kinase (*Active_kinase* process) and a target binding domain (*Mod_Bind_domain* process), where the kinase may phosphorylate a tyrosine residue in another motif in the binding site. In the mobile approach, this modification is represented by the sending of a *p_tyr* message, to be received by the *tyr* "placeholder":[15]

$$\begin{aligned} Active\_kinase &::= phosph\_site\ !\{p\_tyr\}\ ,\ Kinase \\ Mod\_Bind\_domain &::= phosph\_site\ ?\{tyr\}\ ,\ tyr!\{\cdots\}\ ,\ \cdots \end{aligned}$$

If an interaction takes place between *Active_kinase* and *Mod_Bind_domain*, the received *p_tyr* channel will replace the *tyr* channel in the receiving *Mod_Bind_domain*. Overall, the communication-with-message process can be summarized in the following reduction:

$$\begin{aligned} phosph\_site\ !\{p\_tyr\}\ ,\ \cdots \quad | \quad phosph\_site\ ?\{tyr\}\ ,\ tyr!\{\cdots\}\ ,\ \cdots \\ \rightarrow \\ Kinase \quad | \quad p\_tyr!\{\cdots\}\ ,\ \cdots \end{aligned}$$

representing the interaction, modification and release events involved in the chemical reaction in the

[15] For example, read: "send *p_tyr* on *phosph_site* and then iterate as *Kinase*."

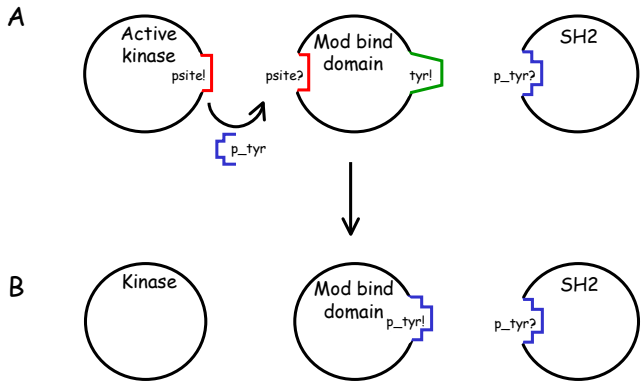

Figure 1.15: **Interaction and modification as communication and mobility.** A. Pre-communication. Three processes and their respective channels: An *Active_kinase* process (with a *phosph_site* channel), a *Mod_Bind_domain* process (with *phosph_site* and *tyr* channels), and an $SH2$ process (with a *p_tyr* channel). The *Active_kinase* and the *Mod_Bind_domain* may interact on the *phosph_site* channel. B. Post-communication. The *tyr* channel in the *Mod_Bind_domain* is replaced with a *p_tyr* channel. This may now be used for communication with the $SH2$ process.

system. Most importantly the *p_tyr* received by the *Mod_Bind_domain* process may now be used for further communications with other processes which harbor this motif, such as the $SH2$ process:

$$SH2 ::= p\_tyr?\{\cdots\} \ , \ \cdots$$

This channel mobility represents the actual chemical modification of a motif in the binding site and the effect of this modification on subsequent interaction. The full sequence of events is summarized in Figure 1.15.

1.2.6 Compartment change as extrusion of a private channel's scope

We use the mobility mechanism to abstract compartment changes, such as complex formation. Recall that we use private channels to represent a compartments' identity. Just like public channels, private channels may be sent as messages and used by the receiving process. We use this extrusion ability to represent compartment changes. For example, consider a three-molecule complex that forms between a dimeric ligand and the extracellular domain of two RTK receptors. As a result of complex formation, the two RTK domains may interact "privately", without the participation of any other RTK molecule. To represent this, the private *backbone* channel is sent from the *Ligand*'s *Binding_domain* sub-processes to the RTKs' *Extra* processes:

$$Ligand ::= (\text{new } backbone)(Binding\_domain \mid Binding\_domain).$$
$$Binding\_domain ::= ligand\_binding \ ! \ \{backbone\}, \ Bound\_domain.$$
$$Extra ::= ligand\_binding \ ?\{cross\_backbone\} \ , \ Bound\_Extra(cross\_backbone).$$

As a result of two communication events (of each of the two *Binding_domains* with a different RTK's

Extra sub-process, we end up with two *Extra* processes which both "own" the same private *backbone* channel, representing their indirect link via a commonly bound ligand molecule.

$$
\begin{array}{c}
(\mathsf{new}\ backbone) \\
(\ ligand\_binding\ !\ \{backbone\},\ Bound\_domain\ \mid \\
ligand\_binding\ !\ \{backbone\},\ Bound\_domain\ )\ \mid \\
ligand\_binding\ ?\ \{cross\_backbone\}\ ,\ Bound\_Extra(cross\_backbone)\ \mid \\
ligand\_binding\ ?\ \{cross\_backbone\}\ ,\ Bound\_Extra(cross\_backbone) \\
\longrightarrow \\
(\mathsf{new}\ backbone) \\
(\ Bound\_domain\ \mid\ Bound\_domain \mid \\
Bound\_Extra(backbone)\ \mid\ Bound\_Extra(backbone)\ \ )
\end{array}
$$

The *Bound_Extra* processes will subsequently use this private *backbone* channel to interact with each other, without external interruptions.

1.2.7 Molecular objects as parametric processes

The use of parametric process definitions allows us to abstract the constant identity of a molecule (process) as its structure (public channels) and compartment (private channels) change with interaction. The latter use is essential, as we have shown above for the *backbone* channel in the $Bound\_Extra(bb)$ process. Parametric public channels are not essential, and can be replaced in principle by various processes, each representing a single molecular state and using different channel sets. However, this representation is cumbersome and inconvenient for many molecules that may be modified in many different combinations, especially signal transduction proteins. Rather, we can model the modification and reconstitution of a molecule by reiterating the same process, but with different parameters. For example, we may define the *Mod_Bind_domain* described above (which can either interact with a modifying kinase, or bind to a SH2 domain) as:

$$
\begin{array}{c}
Mod\_bind\_domain(kinase\_site, sh2\_site)\ ::= \\
kinase\_site\ ?\ \{mod\_sh2\_site\}, Mod\_bind\_domain(kinase\_site, mod\_sh2\_site)\ + \\
sh2\_site\ ?\ [\ ],\ \cdots
\end{array}
$$

1.2.8 Competition as choice

A molecule may often participate in one of several mutually exclusive interactions. We abstract this by the choice operator ("**;**" or "**+**").[16] Once one of these interactions takes place, the others are no

[16]The two symbols are semantically equivalent (have exactly the same meaning). In practice (due to reasons related to the implementation), the former is used in communication clauses (*e.g.* $a\ !\ [\ ],\ A\ ;\ b\ !\ [\ ]\ ,\ B$) while the latter is used for choice between two process definitions each of which consists of such a choice construct (*e.g.* $C\ +\ D$ where $C ::=\ a\ !\ [\ ]\ ,\ A$ and $D ::=\ b!\ [\ ]\ ,\ B$) . We use them interchangeably in the text.

longer possible. The communication mechanism of the π-calculus handles this by discarding the other choices once an alternative is chosen. For example, consider a *System* in which the *Extra* process may interact either with an agonist *Ligand*'s *Binding_domain* (on the *ligand_binding* channel), or with an *Antagonist* (on the *antagonist_binding* channel):

$$\begin{aligned} System ::= &\quad Extra \mid Ligand \mid Antagonist\ . \\ Extra ::= &\quad ligand\_binding\ !\ [\ ]\ ,\ Extra\_Bound\_Agonist\ + \\ &\quad antagonist\_binding\ !\ [\ ]\ ,\ Extra\_Bound\_Atagonist. \\ Ligand ::= &\quad ligand\_binding\ ?\ [\ ]\ ,\ Bound\_ligand. \\ Antagonist ::= &\quad antagonist\_binding\ ?\ [\ ]\ ,\ Bound\_antagonist. \end{aligned}$$

If *ligand_binding* is chosen, then the *antagonist_binding* option is discarded and the resulting system is

$$Extra\_Bound\_Agonist \mid Ligand\_bound \mid Antagonist$$

Vice versa, if *antagonist_binding* is chosen, then the *ligand_binding* option is discarded and the resulting system is

$$Extra\_Bound\_Antagonist \mid Ligand \mid Bound\_antagonist$$

Choice is resolved non-deterministically: All enabled communications (where both input and output are available) are equi-potent. A more biologically realistic model would assign different probabilities to different interactions, based on reaction rates. We extend the language to handle such a model in Chapter 2.

1.3 The π-calculus: A formal presentation

The "molecule-as-computation" abstraction is based on and motivated by the π-calculus abstract process language [89]. In this section, we formally present the calculus, in a way independent of its application to biological systems. As this section is a technical introduction to the calculus, the biologist reader is referred to Section 1.4 for a continuation of the biologically motivated and informal presentation.

In modeling a (computational) system in a calculus we attempt to find and represent the basic entities behind the computation, allowing formal reasoning about them. The π-calculus ([89],[98]) suggests a concise representation for communication between processes in concurrent computational systems. In this calculus, communication is modeled by only two basic entities: *channels* and *processes*. The calculus consists of three components: A simple *syntax* for writing formal descriptions of a concurrent system's state; a set of *congruence laws* that determine when two descriptions are equivalent, and a set of *reduction rules*, defining the change in the state of the system induced by communication. In what follows, we describe this core calculus and some specific extensions we use.

Communication prefixes

$\pi \stackrel{\text{def}}{=}$	$x\,!\,[\,]$		output alert: send nil alert along x
	$x\,!\,\{y_1,\cdots,y_n\}$		output tuple: send $\{y_1,\cdots,y_n\}$ along x
	$x\,?\,[\,]$		input alert: receive nil alert along x
	$x\,?\,\{z_1,\cdots,z_n\}$		input tuple: receive into $\{z_1,\cdots,z_n\}$ along x

Match prefixes

$\mu \stackrel{\text{def}}{=}$	$x_1 =? = y_1 \& \cdots \& x_n =? = y_n$	match
	$x_1 = / = y_1 \& \cdots \& x_n = / = y_n$	mismatch

Processes

$P,Q \stackrel{\text{def}}{=}$	$\mathbf{0}$	Inaction (empty process)
	π,P	Communication prefix
	μ,P	Match prefix
	π_1,$P_1 + \cdots + \pi_n$,P_n	Communication choice
	μ_1,$P_1 + \cdots + \mu_n$,P_n	Match choice
	$P \vert Q$	Composition
	$A(y_1,\cdots,y_n)$	(Parametric) process identifier
	$(\mathsf{new}\ n)P$	Restriction

Figure 1.16: **The π-calculus: Syntax.** The syntactic definitions for prefixes and processes are shown, with the exception of an "otherwise" option in the match choice clause, and arithmetic and logic (non π-calculus) prefixes.

1.3.1 Syntax

The primitive entity in the calculus is a *(channel) name*. Names, infinitely many, are written $x, y, \cdots$, have no additional structure, and refer to specific channels that can be used for communication or be transmitted as messages. The other entity in the calculus is the *process*. Processes are denoted $P, Q, \cdots$, and are built from names according to a specific syntax (Figure 1.16). Figure 1.17 shows a simplified graphical representation of channels and processes.

Communication actions and prefixes

Channels are employed in two complementary types of communication actions. In the *output action*, denoted $x\ !\ \{y, z, w, \cdots\}$, a tuple of channel names $y, z, w, \cdots$ is sent along channel x to a receiving process at the "other end" of x. In the complementary *input action*, denoted $x\ ?\ \{t, u, v, \ldots\}$, a tuple of channel names is received along the channel x. Here, $t, u, v, \cdots$ are but *placeholders* for the received names, which are unknown to the receiving process until the communication actually takes place. Note, that the messages may also be empty. Such output and input *alerts* are noted $x![\,]$ and $x?[\,]$, respectively. Thus, there is a polarity in the *use* of names: $x!$ serves as the *output occurrence* of x (and is used for sending actions), while $x?$ serves as the *input occurrence* (and is used for receiving actions).[17]

Communication actions (denoted collectively as π) can be composed in sequence, using the *prefix operator*, **","** (the comma sign), to construct processes of the form "π, P", termed *normal processes*. In such a process, the prefix action (be it input or output) must take place before the rest of the process, P, is allowed to proceed. The simplest process P is the empty process, denoted $\mathbf{0}$, which cannot perform any actions, now or ever.

[17] In the π- calculus literature the input name is plainly known as a name, while the output name is termed the *co-name*.

Operators on processes and parametric definitions

Processes of increasing complexity can be hierarchically constructed in three ways. First, we can sum together two normal processes (each of the general form π, P) using the *mutually exclusive choice operator*, **";"** (or **"+"**). The resulting process, $\pi_1, P_1 + \pi_2, Q_1$, is a normal process, too, able to participate in one – but only one – of two alternative communications.

Second, we may compose any two processes in parallel using the *parallel composition operator*, PAR (denoted **"|"**). $P|Q$ means that P and Q are concurrently active. They can act independently as well as communicate.

Finally, we may define parametric processes, using *parametric process identifiers*, $A(y_1, \cdots, y_n)$, where n is the arity of A. Every identifier has a definition $A(x_1, \cdots, x_n) ::= P$, where the x_i must be pairwise distinct (*i.e.* if $i \neq j$ then $x_i \neq x_j$). A definition is a process declaration with $x_1, \cdots, x_n$ as formal parameters. The identifier $A(y_1, \cdots, y_n)$ is an invocation of this definition with actual parameters $y_1, \cdots, y_n$. The intuition is that $A(y_1, \cdots, y_n)$ behaves as P with y_i replacing x_i for each i.[18]

Channel restriction

In general, all channel names may be identified by any process which "owns" the name (either when defined or as received in a message). However, we may also restrict the use of a name to a particular process, using new, the restriction operator. new $x.P$ declares a fresh and unique channel name, x, which is distinct from all external names, for use in P.

Extended prefixes: Match, logic and arithmetic

In addition to the basic communication prefixes there are several additional types of prefixes.[19] Unlike the communication actions, which are complementary and require synchronization (below), the additional prefixes are unary: the constructs are evaluated within the process and do not require a counterpart. The *match* (or *mismatch*) prefixes check whether two tuples of channel names match (or mismatch). The process continues if and only if the prefix is satisfied. Several processes with *match* and/or *mismatch* prefixes may be summed together in a choice construct, but may not be "mixed" with normal processes (with communication prefixes). An "otherwise" prefix may be included in this construct, to account for all the alternatives that are not explicitly specified by a dedicated *(mis)match* prefix.

Similar to the use of match and mismatch prefixes, we add a set of prefix types allowing to evaluate a variety of arithmetical and logical expressions on *logic variables* [125]. Logic variables are distinct from channels and processes, and may hold various values (integers, real numbers, strings, etc). Like channels, they may be sent from one process to another. We can also sum together several processes with such prefixes, but we may not mix them with the other prefixes (communication and match). With the exception of the choice construct and message passing, variables' semantics is based on that of Flat

[18] In the original π-calculus [89], there are no parametric process identifiers. Rather, we may replicate a process an unlimited number of times by using the *replication operator*, **"!"**. The process $!P$ means an unlimited resource of concurrently acting Ps: $P|P|P|\cdots$. The replication operator does *not* mean that an infinite number of copies is present in the system at any point. Rather, an unlimited number of Ps is available for interaction, and a particular instance will be created upon a corresponding request. However, the replication operator is hardly useful in a practical implementation, while recursive parametric definitions are extremely useful, and were shown equivalent to replication in [89].

[19] The original π-calculus includes only the communication prefixes. Other prefixes were added later by others (the *match* prefixes, [98]) and us (the other types).

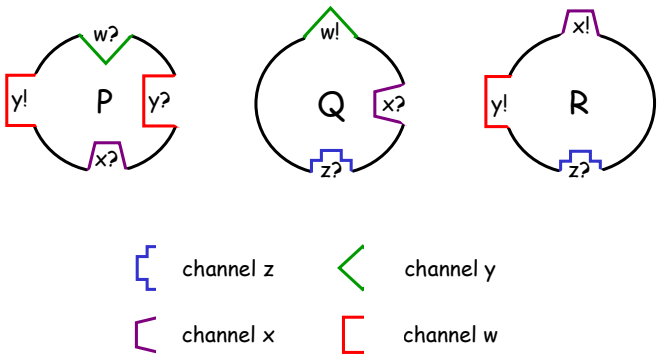

Figure 1.17: **π-calculus processes and channels: An intuitive view.** Three processes, P, Q, R (ovals) with four communication channels (complementary shapes of protrusions and depressions).

Concurrent Prolog [125]. Variables are useful in expressing certain non-molecular aspects of the modeled system as shown in the next chapter.

1.3.2 Congruence laws

Processes that are syntactically distinct may represent the same behavior. For example, the processes $P|Q$ and $Q|P$ both represent the parallel composition of P and Q, albeit in syntactically distinct ways. Parallel composition is inherently unordered and the syntactic distinction is but a limitation of our linear syntax. Several *structural congruence laws* are introduced to identify processes which intuitively "represent the same thing" (Figure 1.18).

- **Associativity and commutativity of parallel composition and choice.** As discussed above, in a concurrent system there is no order between processes. To reflect that, both the choice ("+" or ";") and the parallel composition ("|") operations are commutative (*e.g.* $A + B \equiv B + A$) and associative (*e.g.* $A + (B + C) \equiv (A + B) + C$). Naturally, we may also include or discard the empty process **0**, which has no current or future behavior from a choice or a parallel composition without affecting equivalence.

- **Renaming of bound channels.** We distinguish between free and bound channels. All occurrences of x in P are bound if either $a\ ?\ \{x\}\ .\ P$ (for all a) or if $\mathsf{new}\ x\ .\ P$. Otherwise, x is free in P. Thus, all channels are free except the input *placeholders* and the restricted channels. Bound channel names are "immaterial". Input placeholders will be replaced by actual channels, to be received as input. Thus, the processes $a?\{x\}, b!\{x\}$, and $a?\{y\}, b!\{y\}$ are syntactically different but represent the same behavior: a channel name is received along channel a into a placeholder (x or y) and then sent along channel b. The same is true for restricted channels, whose names are fresh and distinct from any other name outside their scope. As the actual name of a bound channel is "immaterial" we may rename it. Thus, we may obtain an equivalent process Q from a process P by renaming one or more of the bound channels in P. This renaming process is termed α-conversion. In the renaming process the current name of a bound channel may be changed to whatever name we wish, as long as it does not change a free name to a bound one (*i.e.* we may not rename a bound channel

Associativity and commutativity for parallel composition and choice

$P\|Q$	$\equiv$	$Q\|P$	Commutativity of parallel composition
$(P\|Q\|)\|R$	$\equiv$	$P\|(Q\|R)$	Associativity of parallel composition
$P+Q$	$\equiv$	$Q+P$	Commutativity of choice
$(P+Q\|)+R$	$\equiv$	$P+(Q+R)$	Associativity of choice
$P\|\mathbf{0}$	$\equiv$	P	Empty process as unit in parallel composition
$P+\mathbf{0}$	$\equiv$	P	Empty process as unit in choice

Renaming of bound channels

$x?\{y\}, P$	$\equiv$	$x?\{z\}, (\{z/y\}P)$	if $z \notin FN(P)$	Renaming of input channel y
$(\mathsf{new}\ y)P$	$\equiv$	$(\mathsf{new}\ z)(\{z/y\}P)$	if $z \notin FN(P)$	Renaming of restricted channel y

Scope extension

$(\mathsf{new}\ x)\mathbf{0}$	$\equiv$	$\mathbf{0}$	Scope of empty process
$(\mathsf{new}\ x)(\mathsf{new}\ y)P$	$\equiv$	$(\mathsf{new}\ y)(\mathsf{new}\ x)P$	Commutativity of scopes
$(\mathsf{new}\ x)(P\|Q)$	$\equiv$	$P\|(\mathsf{new}\ x))Q$ if $z \notin FN(P)$	Scope extrusion
$(\mathsf{new}\ x)u =? = v, P$	$\equiv$	$u =? = v, (\mathsf{new}\ x)P$ if $x \neq u$ and $x \neq v$	Scope and match
$(\mathsf{new}\ x)u = / = v, P$	$\equiv$	$u = / = v, (\mathsf{new}\ x)P$ if $x \neq u$ and $x \neq v$	Scope and mismatch

Unfolding law

$A(y_1, \cdots, y_n)$	$\equiv$	$P\{y_1/x_1, \cdots, y_n/x_n\}$	if $A(x_1, \cdots, x_n) ::= P$	Parametric definition

Figure 1.18: **The π-calculus: Congruence laws.** FN - free names.

to that of an existing free channel in the same scope, since by this the free channel will become bound).

- **Scope extension.** Intuitively, $(\mathsf{new}\ x)P$ denotes that x is a new unique name in P, and can be considered as marking the occurrences of x in P with a special "color", to identify the local name. Thus, the exact position of $(\mathsf{new}\ x)$ in the process expression matters only in as much as it marks the same occurrences of x. For example, in the empty process, $\mathbf{0}$, there are no occurrences, so the restriction can be removed; in parallel composition, if all occurrences are only in one of the components, then it does not matter if the restriction covers only that component or the whole composition; and if we have two or more consecutive restrictions - their order does not matter.

- **Parametric definition.** To allow parametric definitions of processes we must equate an identifier with its definition, with the appropriate parameter instantiation. This is handled by the unfolding law.

1.3.3 Operational semantics: Reduction rules

We now turn to describe the reduction rules governing inter-process communication. In principle, communication between two processes requires that they share a common channel on which the sender process outputs a message (another channel name), which the receiver receives. As a result, the communication offers (prefixes) in both processes are eliminated, releasing the remainder of each process and discarding alternative choices. The receiver also substitutes all the occurrences of the placeholder(s) with the channel name(s) received in the message. The basic mechanism is schematically depicted in Figure 1.19.

Five reduction rules are used to realize scenario separately and in combinations (Figure 1.20).

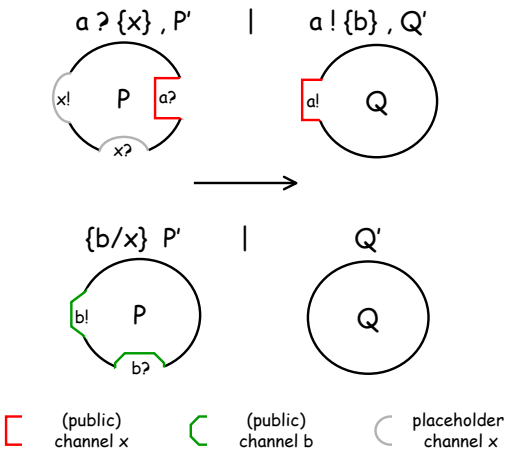

Figure 1.19: **π-calculus communication: An intuitive view.** A. Two processes, P and Q (ovals), share a communication channel (x). Q is willing to send on the channel (protrusion), while Q is ready to receive (complementary depression). The message Q is sending is the channel name y. P is willing to receive this into the channel placeholder z. Alternative choices are not shown. B. Following communication, the processes continue as P' and Q'. The previous communications were removed. In Q, the place holder z was replaced by the received channel y, which may be used, in principle, as either input, output or message.

- **The communication axiom (COMM)** defines the basic premises of communication and its consequences. According to the axiom, two normal processes can communicate if they share a communication channel (a) on which one is willing to send and the other is willing to receive tuples of the same arity. If communication occurs, several things will happen. First, the communication actions ($a?\{x_1, \cdots, x_n\}$ and $a!\{u_1, \cdots, u_n\}$) are eliminated and the processes they guarded (P and Q) proceed. Second, alternative choices (the $\cdots+$) are discarded in both processes. Third, the received channels (u_i) replace all occurrences of the corresponding placeholders (x_i) in the receiving process (denoted as $P\{u_i/x_i\}$). Note, that the COMM rule presented here requires that the tuple of the sent and received messages is identical in size. In addition, we implicitly assume that a channel may be used to send and receive only messages of a single tuple size (*e.g.* only empty messages, or only messages with two channels etc.).[20] The COMM axiom is a schema, and applies to many specific situations (where a, x_i, u_i, P, Q may be replaced by any other channel and process name). Its syntax, however, is rigid. Thus, it cannot be applied directly when we change the order of the sender and the receiver process or of the summed terms or when non-communication prefixes are used. The four additional rules add this flexibility to the COMM axiom.

- **Communication under parallel composition (PAR) and restriction (RES)**. Two processes can communicate with each other even if there is a multitude of other concurrent processes in the system. This situation is handled by the **PAR** rule, stating that a communication (reduction) between two sub-processes of P is independent of the occurrence of an additional concurrent process Q. Similarly, the presence of an external restriction on a channel in a process, should not affect the ability of internal sub-processes to communicate. This is reflected in the **RES** rule. Note, however, that unlike parallel composition and restriction, communication is *not* allowed under

[20] This implicit assumption underlies a flat *sort* scheme for channels. In a *sort scheme* [89], each channel's sort is defined by its message. In a flat scheme, the channel sort is the direct tuple size (arity) of allowed messages. More sophisticated schemes are recursive (*i.e.* evaluating not just the immediate sort of channel but also the sorts of channels sent as its messages).

$\cdots + a?\{x_1,\cdots,x_n\},P) \mid (\cdots + a!\{u_1,\cdots,u_n\},Q)$	$\rightarrow$	$P\{u_1/x_1,\cdots,u_n/x_n\}\mid Q$	**COMM** axiom
if $P \rightarrow P'$	then	$P \mid Q \rightarrow P' \mid Q$	**PAR** rule
if $P \rightarrow P'$	then	$(\mathsf{new}\ x)P \rightarrow (\mathsf{new}\ x)Q$	**RES** rule
if $P \equiv Q,\ P \rightarrow P'$, and $P' \equiv Q'$	then	$Q \rightarrow Q'$	**STRUCT** rule
$\mu_1\ ,\ P_1\ +\ \cdots\ +\ \mu_n\ ,\ P_n\ +\ otherwise\ ,\ Q$	$\rightarrow$	P_i if μ_i =True, Q otherwise	**NON-COMM** axiom

Figure 1.20: **The π-calculus: Reduction rules.** μ_i - a non-communication prefix

prefixes. Rather, the presence of a communication offer prefix precludes any action from the rest of the process. In this way, prefixing induces a sequential order on communication.

- **Inferring communication by structural congruence (STRUCT)**. The **STRUCT** rule states that structurally congruent processes have the same reduction. This rule is critical in order to relax the syntactic rigidity of the previous three rules, extending them to a wide variety of syntactically congruent situations.

- **Reduction of non-communication prefixes.** The above rules handle only binary communication prefixes but not unary ones (*e.g. match*). The last rule handles all such prefixes allowing immediate evaluation of any leftmost non-communication prefix (*i.e.* a non-communication action which is "ready" for immediate evaluation).

Importantly, none of the reduction rules specifies *which* communication *will* happen, but only which communications are *allowed* to happen, and what would happen if this communication is allowed to occur. Since there may be various processes occuring concurrently, each with multiple choices, and since once a particular communication takes place it may influence future events (due to discarding of alternatives, release of guards, and channel mobility), there may be many alternative scenarios that can unfold from a given starting point. The evolution of the system is non-deterministic as we cannot specify which one of all of the various communications events that are enabled in a particular point (according to a combination of the five reduction rules) will occur.

1.4 Essential modeling use-cases

The general principles outlined in Sections 1.1.2 and 1.2 allow us to formally represent detailed information on complex pathways, molecules and biochemical events. In this section we illustrate these capabilities with several small programs representing different aspects of molecular systems.

1.4.1 Molecular complexes

Our first two examples handle the formation and breakage of a bi-molecular complex.

Complex binding and unbinding

To represent the formation and breakage of a complex between two molecules, Molecule1 and Molecule2, we use both public and private channels (Figure 1.21). Each of the molecules is represented by a process ($Molecule1$ and $Molecule2$, Figure 1.22A). The two processes share a public *bind* channel, on which one process ($Molecule1$) is offering to send a message, and the other ($Molecule2$) is offering to receive (Figure 1.22A) . These complementary communication offers represent the molecular complementarity

```
System::= Molecule1 | Molecule1 | Molecule1 |
          Molecule2 | Molecule2 | Molecule2 .

Molecule1::= (new backbone) . bind ! {backbone} , Bound_Molecule1(backbone) .
Bound_Molecule1(bb)::= bb ! [ ] , Molecule1 .

Molecule2::= bind ? {cross_backbone} , Bound_Molecule2(cross_backbone) .
Bound_Molecule2(cbb)::= cbb ? [ ] , Molecule2 .
```

Figure 1.21: π **calculus code for a heterodimer complex**

of the two molecules, and the communication event represents binding. The private *backbone* channel sent from $Molecule1$ to $Molecule2$ represents the formed complex, and the two processes change to a "bound" state ($Bound\_Molecule1$ and $Bound\_Molecule2$, Figure 1.22B). A communication between the two "bound" processes on the shared private *backbone* channel represents complex breakage. As a result, the two processes return to the initial "free" state ($Molecule1$ and $Molecule2$), completing a full cycle. Note, that in a system with many copies of these processes, any two particular copies of $Molecule1$ and $Molecule2$ may communicate (*i.e.* "bind") on the *bind* channel. However, the two resulting "bound" processes share a private channel, which is distinct from all other channels, and may allow only this particular pair to communicate ("unbind") with each other. Three such complexes and their specific private channels (distinguished in red, green, and purple) are shown in Figure 1.22B.

Special case: Homodimerization

The π-calculus program presented above for a heterodimer must be modified to represent homodimerization (Figure 1.23). The reason is that a homodimer forms between two identical molecules that are represented by two copies of the same process. Since it is impossible *a priori* to break down the symmetry between the two components of the homodimer, both input and output must be offered simultaneously by the same process, analogously to the way molecules should have a "zipper" like structure to allow homodimerization (Figure 1.24A). The asymmetry is broken by using *non-deterministic choice*: while both processes offer both a send and a receive communication, one will (non-deterministically) send and the other receive. Since once an option is chosen, the other is automatically withdrawn, a process cannot communicate with itself (in the same sense that a molecule cannot bind onto itself). In the example (Figure 1.23), each $Molecule$ process offers a choice between sending and receiving a message on the *bind* channel. Once one is selected, the other is withdrawn, and the $Molecule$ changes to a "bound" state ($Bound\_Molecule$ process). The message, as in heterodimerization, is a private *backbone* channel, to be used for an "unbinding" communication. This, too, is done by a symmetric choice construct (Figure 1.24B).

1.4.2 Enzymes

Our next use-case tackles enzymatic catalysis. In the two examples in this section we handle "classical" enzymatic reactions typical of metabolic pathways.

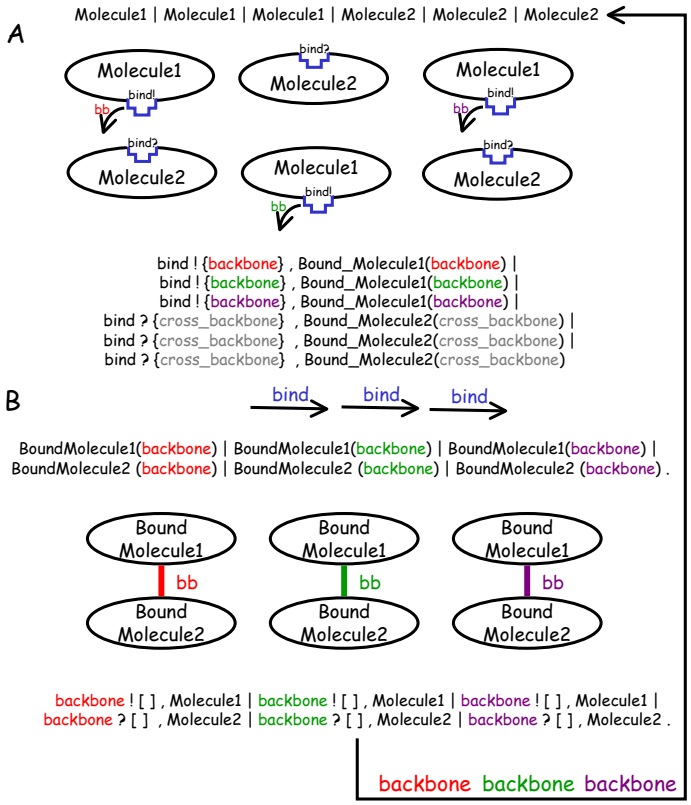

Figure 1.22: **Formation and breakage of a heterodimer complex.** A. Separate molecules. Several copies of *Molecule*1 and *Molecule*2 processes, with complementary communication offers on a shared public *bind* channel (rectangular depression and protrusions). *Molecule*1 is willing to send the private *backbone* channel. B. Three complexes. Each pair of processes shares a private (distinct) *backbone* channel (red, green and purple), which can be use to specifically "unbind".

```
System::= Molecule | Molecule | Molecule | Molecule | Molecule .

Molecule::= (new backbone) . bind ! {backbone} , Bound_Molecule(backbone) ;
                             bind ? {cbb}      , Bound_Molecule(cbb) .

Bound_Molecule(bb)::= bb ! [ ] , Molecule ;
                      bb ? [ ] , Molecule .
```

Figure 1.23: π **calculus code for a homodimer complex**

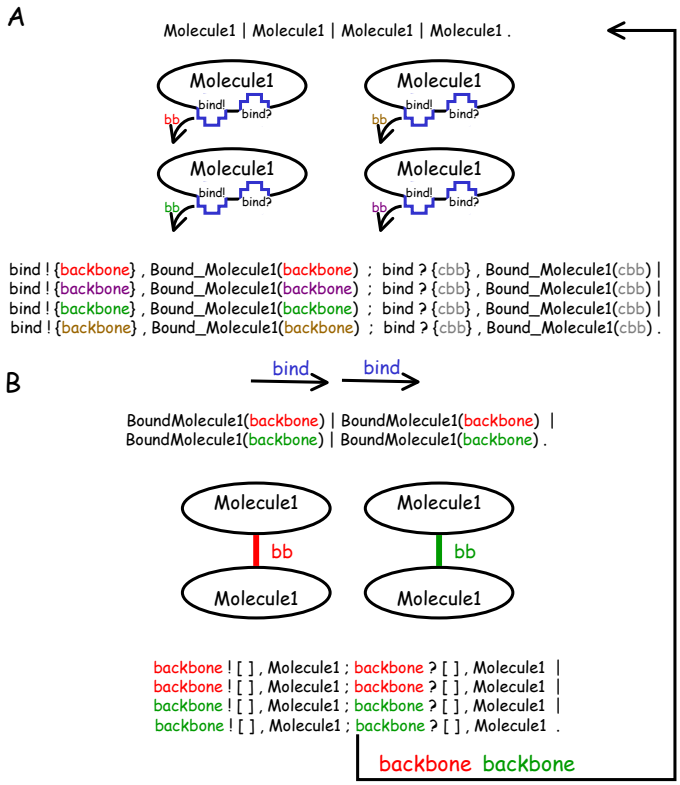

Figure 1.24: **Formation and breakage of a homodimer complex.** A. Separate molecules. Several copies of the *Molecule* process, each offering both to send and to receive on the *bind* channel (both depression and protrusions appear on each *Molecule*). B. Two complexes. Each pair of processes shares a private *backbone* channel (green and red), which can be used to specifically "unbind", in another symmetric interaction.

A.

```
System::= Enzyme | Substrate .
Enzyme::= bind_and_react ! [ ] , Enzyme .
Substrate::= bind_and_react ? [ ] , Product1 | Product2 .
Product1::= 0 .
Product2::= 0 .
```

B.

```
System::= Enzyme | Substrate .
Enzyme::= bind_s_and_react ! [ ] , Enzyme ;
          bind_p_and_react ! [ ] , Enzyme .
Substrate::= bind_s_and_react ? [ ] , Product .
Product::= bind_p_and_react ? [ ] , Substrate .
```

C.

```
System::= Enzyme | Substrate .
Enzyme::= (new rel_s, rel_p) . bind_s ! {rel_s,rel_p} , EX(rel_s,rel_p) ;
                               bind_p ! {rel_s,rel_p} , EX(rel_s,rel_p) .
EX(release_s,release_p)::= release_s ! [ ] , Enzyme ;
                           release_p ! [ ] , Enzyme .

Substrate::= bind_s ? {erel_s , erel_p} , X(erel_s , erel_p) .
Product::= bind_p ? {erel_s , erel_p} , X(erel_s , erel_p) .
X(rel_es,rel_ep)::= rel_es ? [ ] , Substrate ;
                    rel_ep ? [ ] , Product .
```

Figure 1.25: π **calculus code for single-substrate enzymatic reactions.** A. A simple model for a single-substrate irreversible reaction with two products. B. A simple model for a single-substrate reversible reaction with one product. C. An elementary reaction model for a single-substrate reversible reaction with one product.

Single-substrate enzymatic reaction

We first consider a simple view of a single-substrate enzymatic reaction (Figures 1.25A, and 1.26) as an atomic event in which the enzyme interacts with a substrate, catalyzing its transformation to one (or more) other biochemical entities. For simplicity, we first assume that the reaction is irreversible, and that the product(s) have no subsequent role in the system. We represent the enzyme and substrate with two processes, *Enzyme* and *Substrate*, and their interaction as communication on a public *bind_and_react* channel, in which *Enzyme* is the sender, and *Substrate* is the receiver. Following communication, the *Enzyme* is recreated, representing the release of an intact catalyst when the reaction is completed, and the *Substrate* continues as two parallel processes, *Product*1 and *Product*2. We do not assign any further behavior to these two processes, denoting them as empty processes (**0**). This simple scheme can be used to model various scenarios. For example, in Figure 1.25B, we modify the *Enzyme* and *Product* to account for a reversible reaction. *Enzyme* now includes a choice between an interaction with *Substrate* on the *bind_s_and_react* channel and an interaction with *Product* on the *bind_p_and_react* channel. In both cases *Enzyme* is recreated. In the former, a *Product* is released, and in the latter - a *Substrate*.

The Michaelis-Menten mechanism provides a more detailed account of an enzymatic reaction [133], breaking it to its elementary steps. A single-substrate reversible enzymatic reaction includes four elemen-

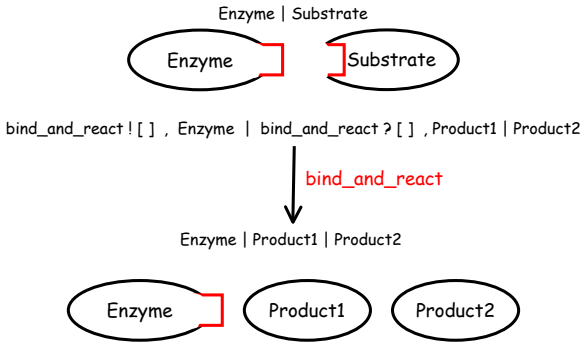

Figure 1.26: **Irreversible, single-substrate enzymatic reaction with two products: a simple model.** *Enzyme* and *Substrate* may interact on a public *bind_and_react* channel. As a result of the interaction *Enzyme* is recreated and *Substrate* becomes two processes: *Product*1 and *Product*2.

tary reactions: binding of an enzyme to the substrate or to the product and formation of an EX complex, and release of either a product or a substrate from this complex. In this case (coded in Figure 1.25C, and schematically depicted in Figure 1.27), we have five process types, representing the free enzyme (*Enzyme*), bound enzyme (EX), substrate (*Substrate*), product (*Product*), and intermediate (X). As in the previous example, *Enzyme* includes a choice between an an interaction with *Substrate* (on *bind_s*) and an interaction with *Product* (on *bind_p*). In both cases, two private channels, *rel_p* and *rel_s*, are sent from *Enzyme* to its counterpart (Figure 1.27A). Following communication, *Enzyme* changes to EX ("bound enzyme"), and its counterpart (be it *Substrate* or *Product*) changes to X (reaction intermediate) (Figure 1.27B). The two private channels shared between EX and X represent the formed complex , and are used to finish the reaction, resulting in *Enzyme* reconstitution and either *Product* or *Substrate* release (Figure 1.27C). Note, that as a result of the non-determinism of the choice construct, *Substrate* binding to *Enzyme* may end up either as a *Product* (by reaction of X and EX on *rel_p*) or be released as an intact *Substrate* (reaction of X and EX on *rel_s*). The same is true for *Product*.

Competitive inhibition

The choice construct is a powerful tool to represent potential molecular interactions and their outcomes. This capability is further illustrated in a model of competitive inhibition of enzymatic reactions (coded in Figure 1.28 and schematically depicted in Figure 1.29). This model extends the previous elementary reaction model of a single substrate reversible enzymatic reaction (above), by adding an *Inhibitor* process, and an additional option in the *Enzyme*'s choice construct, allowing the two to communicate on *bind_i*. Following our standard "binding scheme", *Enzyme* sends a private *rel_i* channel to *Inhibitor*. *rel_i* represents the EI complex and is used for "unbinding", recreating a free *Enzyme* and *Inhibitor*. The additional choice on *bind_i* competes with the already existing ones to interact with *Substrate* and *Product* (on *bind_s* and *bind_p*, respectively), modeling the competitive inhibition scenario. Note, that unlike the other two interactions, we do not explicitly create another process (*e.g.* EI), but rather use a series of communication actions to model inhibitor binding and unbinding (*e.g.* *bind_i* ? *rel* , *rel* ? [ ] , $\cdots$).

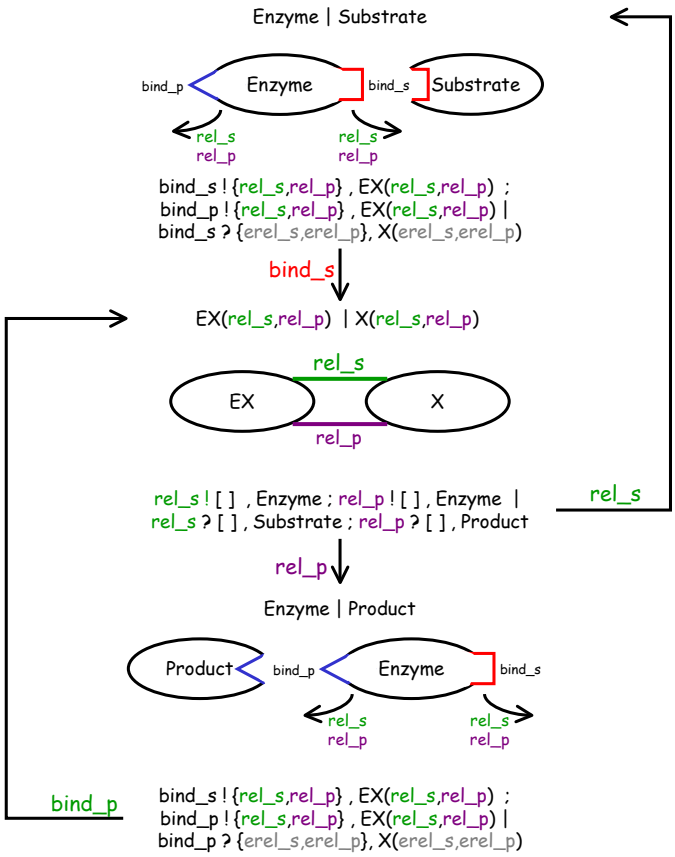

Figure 1.27: **Reversible, single-substrate enzymatic reaction with a single product: Elementary reaction model.** A. *Enzyme* may interact with either *Substrate* (on *bind_s*) or *Product* (on *bind_p*). In both cases, *Enzyme* sends its counterpart two private channels: *rel_s* and *rel_p*. B. Following communication, *Enzyme* becomes EX, while *Substrate* (or *Product*) becomes "intermediate" X. C. X (intermediate) and EX (bound enzyme) may interact on one of two private channels. An interaction on *rel_p* leads to release of *Product*. An interaction on *rel_s* leads to release of *Substrate*. In both cases *Enzyme* is recreated.

```
System::= Enzyme | Substrate | Inhibitor .
Enzyme::= (new rel_s, rel_p, rel_i) . bind_s ! {rel_s,rel_p} , EX(rel_s,rel_p) ;
                                      bind_p ! {rel_s,rel_p} , EX(rel_s,rel_p) ;
                                      bind_i ! {rel_i} , rel_i ! [ ] , Enzyme .

EX(release_s,release_p)::= release_s ! [ ] , Enzyme ;
                           release_p ! [ ] , Enzyme .

Substrate::= bind_s ? {erel_s , erel_p} , X(erel_s , erel_p) .
Product::= bind_p ? {erel_s , erel_p} , X(erel_s , erel_p) .
X(rel_es,rel_ep)::= rel_es ? [ ] , Substrate ;
                    rel_ep ? [ ] , Product .

Inhibitor::= bind_i ? {rel} , rel ? [ ] , Inhibitor .
```

Figure 1.28: π **calculus code for competitive inhibition**

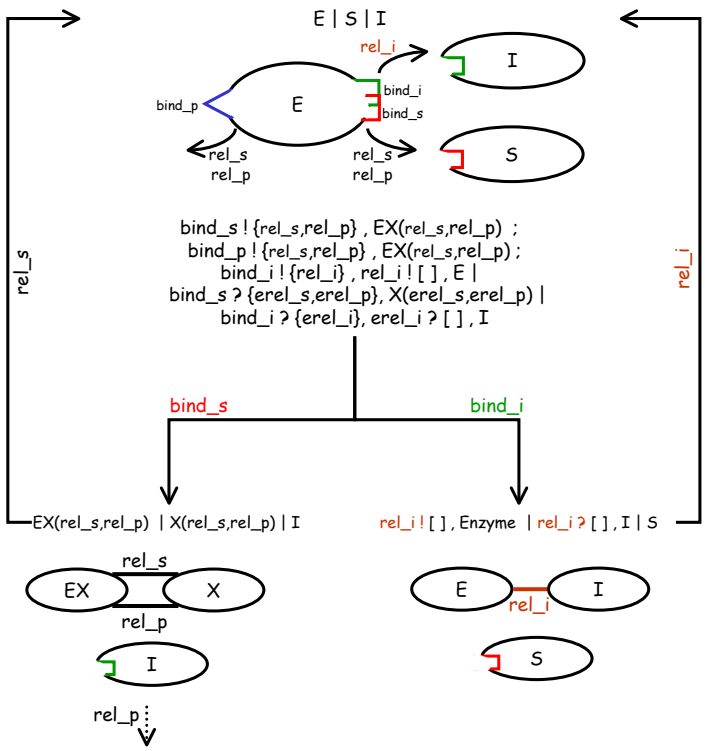

Figure 1.29: **Competitive inhibition.** S (substrate) and I (inhibitor) interact with E (enzyme) in a mutually exclusive way. If E and I interact (on *bind_i*), an "EI complex" forms, which may "unbind" (on the *rel_i* private channel). If E and S interact (on *bind_s*), an "ES complex" forms, which may "unbind" (on the *rel_s* private channel) or react (on the *rel_p* private channel). Product generation and the reverse reaction are not shown, as they are identical to those in Figure 1.27. Note, that I and P will compete in the same way.

A.

```
System::= Binding_Protein(tyr) | Kinase | Phosphatase .

Binding_Protein(res)::= res ? {mod_res} , Binding_Protein(mod_res) .
Kinase::= tyr ! {p_tyr} , Kinase .
Phosphatase::= p_tyr ! {tyr} , Phosphatase .
```

B.

```
System::= Binding_Protein(tyr) | Kinase | Phosphatase .

Binding_Protein(res)::=
    res ? {ereact, erelease} , Sub_Protein(res,ereact,erelease) .
Sub_Protein(res,react,release)::=
    react ? {mod_res} , Binding_Protein(mod_res) ;
    release ! [ ]  , Binding_Protein(res) .

Kinase::=
    (new react, release) .
         tyr ! {react, release} , Bound_Kinase(react,release) .
Bound_Kinase(kreact, krelease)::=  kreact ! {p_tyr} , Kinase ;
                                   krelease ! [ ] , Kinase .

Phosphatase::=
    (new react, release) .
         p_tyr ! {react, release} , Bound_Phosphatase(react,release) .
Bound_Phosphatase(preact, prelease)::= preact ! {tyr} , Phosphatase ;
                                       prelease ! [ ] , Phosphatase .
```

Figure 1.30: π **calculus code for phosphorylation and de-phosphorylation of a binding domain.** A. Simple (single step) enzymatic reaction. B. Model of elementary reaction steps.

1.4.3 Enzymes in signal transduction

Enzymatic reactions play a critical role in signal transduction (ST) pathways. However, unlike metabolic pathways where enzymes (proteins) and substrates (metabolites) are distinct kinds of biochemical entities, in ST pathways most substrates are proteins, serving as binding partners, transcription factors, or enzymes. We represent such "modification of modifiers" by extensively using the mobility mechanism of the π-calculus.[21] Previously, we distinguished the substrate from the product by using distinct process names. In representing the modification of ST molecules, we maintain process identity throughout modification.

Phosphorylation and de-phosphorylation by protein kinases and phosphatases

Consider a toy example involving a binding protein (*Binding_Protein* process), a protein tyrosine kinase (*Kinase* process) and a protein tyrosine phosphatase (*Phosphatase* process). In this system, the protein kinase phosphorylates a tyrosine residue, and the phosphatase de-phosphorylates it. We first model this system handling the simpler scenario of "single-step" enzymatic reactions (Figure 1.30A and Figure 1.31).

[21] An alternative non-mobile representation of such events was discussed above, and will not be presented in further detail.

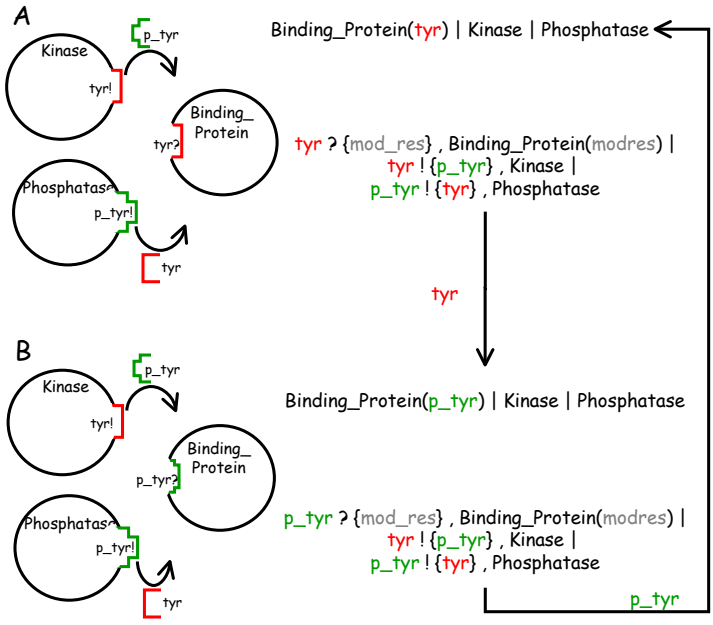

Figure 1.31: **Modification of a protein residue by kinases and phosphatases.** A. A system composed of a *Kinase*, *Phosphatase* and a *Binding_Protein* with a *tyr* channel. B. Following communication between *Kinase* and *Binding_Protein* on *tyr*, *Kinase* is recreated, and *Binding_Protein* now has a *p_tyr* channel, rather than a *tyr* one, allowing interaction with *Phosphatase*. C. Following communication between *Phosphatase* and *Binding_Protein* on *p_tyr*, *Phosphatase* is recreated, and *Binding_Protein* now has a *tyr* channel, rather than a *p_tyr* one, returning to the original state (A).

We represent the modifiable residue as two channels: *tyr* for the non-phosphorylated residue, and *p_tyr* for the phosphorylated one. *Kinase* sends *p_tyr* as a message on the *tyr* channel, representing the fact that the tyrosine kinase identified non-phosphorylated tyrosines and modifies them to the phosphorylated form. Conversely, *Phosphatase* sends a *tyr* message on the *p_tyr* channel.

The *Binding_Protein* has a channel parameter, representing the phosphorylation state of its tyrosine residue. When the system is initialized we assume that it is non phosphorylated, so we create a *Binding_Protein*(*tyr*). This process offers to receive on *tyr*, and may thus interact with *Kinase*, but not with *Phosphatase*. Upon communication, the *Binding_Protein* receives *p_tyr*, and is re-created, but now with the newly received channel (*i.e.* *Binding_Protein*(*p_tyr*)). This process offers to receive on *p_tyr*, and may thus interact with *Phosphatase*, but not with *Kinase*. As before, upon communication, *Binding_Protein* is re-created with the received *tyr* channel, returning to the initial state.

In this way, the iteration of *Binding_Protein* between *tyr* and *p_tyr* parameters represents the cycling of the protein molecule between a phosphorylated and a non-phosphorylated state. These channel parameters affect the communication capabilities of *Binding_Protein*, reflecting the effect of the residue's modification state on the protein's ability to be identified by kinases and phosphatases. This principle can be extended and incorporated into more detailed models. For example, a model combining the elementary reaction (Michaelis-Menten) scheme with the *tyr*/*p_tyr* channel parameter scheme is shown in Figure 1.30B.

```
System::= Binding_Domain(tyr_mod,tyr_bind) | Kinase | Phosphatase | Adaptor .

Binding_Domain(res_mod,res_bind)::=
   res_mod ? {res_mod1,res_bind1} , Binding_Domain(res_mod1,res_bind1) ;
   res_bind ? {unbind} , Bound_Domain(res_mod,res_bind,unbind) .
Bound_Domain(res_mod,res_bind,ub)::=
   ub ? [ ] , Binding_Domain(res_mod,res_bind) .

Kinase::= tyr_mod ! {p_tyr_mod, p_tyr_bind} , Kinase .
Phosphatase::= p_tyr_mod ! {tyr_mod,tyr_bind} , Phosphatase .
SH2_Adaptor::= (new unbind) . p_tyr_bind ! {unbind} , Bound_Adaptor(unbind) .
Bound_Adaptor(ub)::= ub ! [ ] , SH2_Adaptor .
```

Figure 1.32: π **calculus code for phosphorylation and de-phosphorylation of a binding domain**

Modifiable residues in activity and binding

The channel parameter approach can be extended to handle the effect of modified residues on the binding and activity of the proteins that harbor them. For example, assume that the phosphorylated tyrosine (but not the non-phosphorylated one) of the binding Protein discussed above can be bound by an SH2 domain in an adaptor protein. To model this situation we now extend the previous program (Figure 1.32).

Rather than using a single channel parameter in *Binding_Protein* (either *tyr* or *p_tyr*), we now introduce two channel parameters. The first parameter, *res_mod* (either *tyr_mod* or *p_tyr_mod*), is used for communication with *Kinase* and *Phosphatase*. The second parameter, *res_bind* (either *tyr_bind* or *p_tyr_bind*), represents the ability (or lack thereof) to bind to the adaptor protein. *Kinase* and *Phsophatase* send a tuple of two channels to *Binding_Protein*, representing the fact that residue modification affects both the ability to interact with the modifying enzyme and to bind to the adaptor protein. The resulting scenario is schematically depicted in Figure 1.33.

1.4.4 Gene expression

Different molecular events are involved in the direct regulation of gene expression. In the following toy system a transcription factor binds to the promoter of a gene and induces the transcription of RNA molecules (we currently assume no further behavior of the RNA molecules). We model the transcription factor, the gene and the RNA molecule as TF, $Gene\_A$ and RNA_A processes (Figure 1.35).

Binding of the transcription factor to the promoter of its target gene is modeled by a communication between TF and RNA_A on the cis_motif channel. The transcription event is represented by the creation of an RNA_A process (together with the reconstitution of TF and $Gene\_A$). This model (Figure 1.34), albeit simplified, captures several essential properties of transcription.

1.5 BioSpi 1.0: Simulating and tracing π-calculus programs

Once a detailed π-calculus model of a particular pathway is built, one would like to utilize it in different ways. The most natural use is to treat the model as an executable computer program, and simulate the

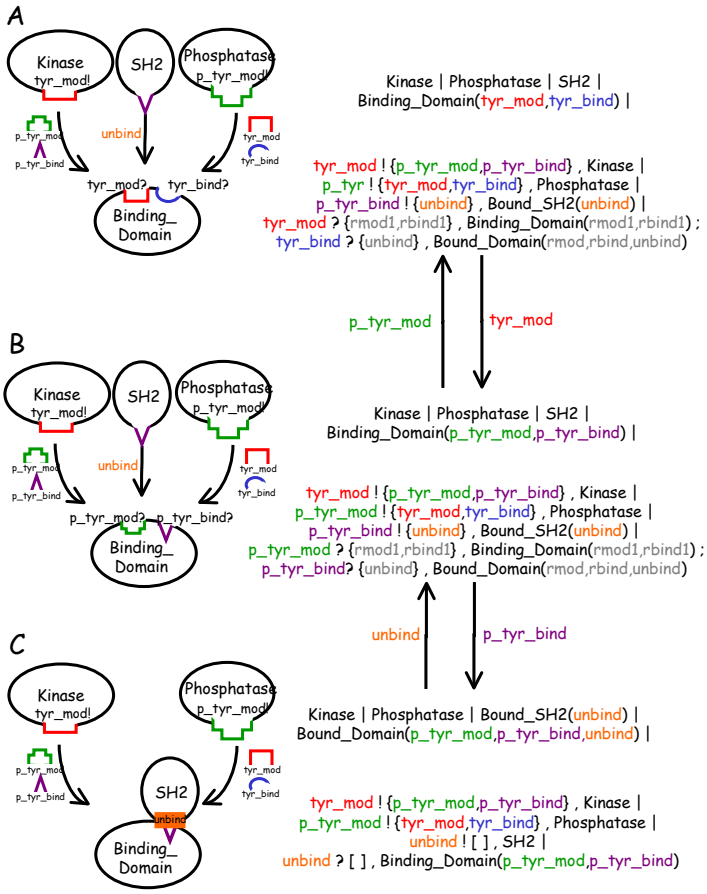

Figure 1.33: **Residue modification and binding.** A. A system composed of *Kinase*, *Phosphatase*, *Binding_Protein* (with *tyr_mod* and *tyr_bind* channels) and *SH2_Adaptor*. B. Following communication between *Kinase* and *Binding_Protein*, *Binding_Protein* now has *p_tyr_mod* and *p_tyr_bind* channels, rather than *"tyr"* ones, allowing interaction with *Phosphatase* (and return to the initial state) or with *SH2_Adaptor*. C. *Adaptor* and *Binding_Protein* "bind" (on *p_tyr_bind*), and "unbind" on a private channel.

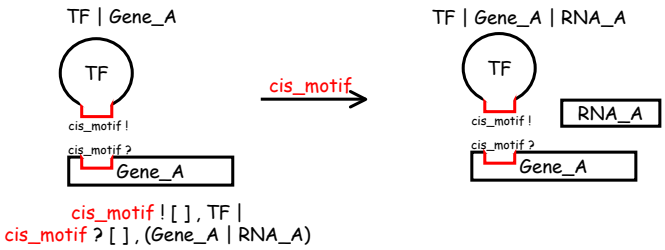

Figure 1.34: **Activated transcription.** A TF process communicates with a $Gene\_A$ process on the cis_motif channel. Following communication TF and $Gene\_A$ are reconstituted and an additional RNA_A process is created.

```
System::= TF | GENE .

Gene_A::= cis_motif ? [ ] , Gene_A | RNA_A .
RNA_A::= 0 .

TF::= cis_motif ! [ ] , TF .
```

Figure 1.35: π **calculus code for simple activated transcription**

behavior of the pathway by running the program.

To this end, we have developed a computer application, called BioSpi. This application (whose implementation is described in Section 1.7.2) receives as input π-calculus code and executes it. In the simulation, each instance of a π-calculus process is realized as a running computational one. Processes run concurrently and interact using channel objects, following the reaction rules of the calculus.

The simulation follows, step by step, the evolution of the system. At each step, a single communication (reaction) step is realized in one atomic operation: a pair of complementary communication actions (input and output on the same channel in two concurrent processes) is chosen, a message is transmitted between the processes, the communication actions and alternative choices are eliminated, the received channel(s) are substituted (instantiated) for the appropriate place holder, and the processes are allowed to continue.

As noted above, communication selection is non-deterministic, and all enabled communication actions are equally likely to be selected. As a result, the simulation process is semi-quantitative: communication (*i.e.* reaction) rates are effectively identical, therefore process (*i.e.* molecule) numbers determine which reactions are more probable than others, and higher-copy-number processes are more likely to participate in communication than lower-copy ones.

Several tools are available for tracing the execution of BioSpi 1.0 programs. Thus, not only the net outcome of a computation can be studied, but also the specific scenario that has led to this outcome. First, the simulation may be executed in a step-by-step fashion, with the ability to set specific break points. This approach is useful to follow small detailed systems. Second, an ordered trace of all the communications in the system and the processes that participated in them is maintained. The trace can be viewed in a form of a tree, ordered chronologically, or according to process evolution. Third, the simulation can be suspended at any desired moment, and the contents (so-called *resolvent*) of the system's current state of computation are examined. The level of detail in which a system is traced (process, channels, messages, senders, etc.) can be determined dynamically throughout a session.

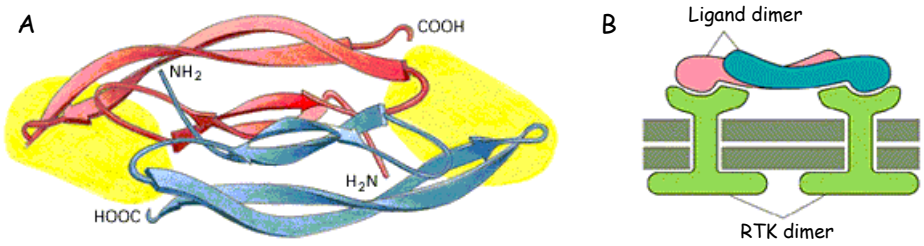

Figure 1.36: **A dimeric ligand and an RTK-ligand complex.** A. A dimeric ligand is formed by di-sulfide bonds between two identical protein monomers. The ligand has two receptor binding sites. Both monomers participate in both binding sites. B. The dimeric ligand can cross-link two adjacent receptors upon their binding, forming a 4 molecule complex. Adapted from [5].

1.6 A π-calculus model for the RTK-MAPK pathway

In this section we dissect the representation of a signaling pathway in detail and illustrate the capabilities of the semi-quantitative simulation system in studying the behavior of a signal transduction pathway.

We have constructed a π-calculus model of a canonical RTK-MAPK pathway (Figure 1.10). The model is composed of 15 molecular processes (including ATP and GTP), with 24 different domains and 15 sub-domains. Four compartments (extracellular, membrane, cytoplasm and nucleus) were defined. A major portion of current knowledge[22], has been incorporated into this concise (270 lines) formal representation. We then studied the model with semi-quantitative BioSpi simulations.

The complete code is available at [106]. Here we highlight a particularly interesting part of the model: the representation of the receptor tyrosine kinase, using the approaches laid out in the previous section for modeling hetero- and homo- dimerization, enzymatic reactions and chemical modification.

1.6.1 The receptor tyrosine kinase model

The receptor tyrosine kinase is composed of extracellular, transmembranal, and intra-cellular parts. We represent these as three parallel processes, sharing a common *backbone* channel:

$$RTK ::= (\mathsf{new}\ backbone)\ .\ Extra \mid Transmem \mid Intra\ .$$

The extracellular part of the receptor binds extracellular ligands (Figure 1.36). The ligand is a dimeric protein molecule, formed by di-sulfide bonds between two identical protein monomers. The two monomers form together two receptor binding domains. We handle the ligand in the same way as a single molecule with two identical binding domains:

$$Ligand ::= (\mathsf{new}\ lbackbone)\ .\ Binding\_Domain \mid Binding\_Domain\ .$$

[22]The model was constructed based on extensive literature on the pathway and related proteins, up to 2000 ([154], [82], [26], [54], [138], [131], [70], [33], [59], [126], [17], [137], [77], [48], [149], [119], [23], [79], [6], [19], [30], [31], [28], [12], [141], [114], [150], [20], [148], [132], [18], [118], [55]).

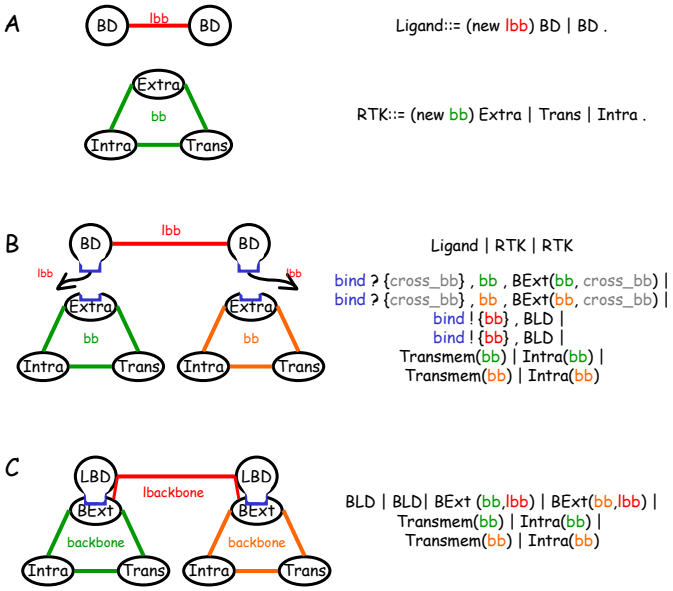

Figure 1.37: **Ligand-induced receptor dimerization.** A. A ligand molecule (the *Ligand* process) is a dimer composed of two binding domains (the *Binding_Domain* processes, sharing the *lbackbone* channel). Each receptor (*RTK* process) is composed of three domains (*Extra*, *Transmem*, and *Intra* processes sharing the *backbone* channel). B. Each of the *Binding_Domains*, may communicate with an *RTK*'s *Extra* sub-process. C. Following two such communications, the two *Bound_LDomains* share the private *lbackbone* channel with two *Extras*, representing a ternary complex between a ligand and the extracellular domains of two receptor molecules.

Ligand-induced receptor dimerization

Each binding domain of the ligand binds to a single receptor molecule. Thus, the dimeric ligand can cross-link two adjacent receptors upon binding. We represent this in the π-calculus code as two separate communications on the *bind* channel: each between a *Ligand*'s *Binding_Domain* process and an *RTK*'s *Extra* process.

$$Binding\_Domain ::= bind\ !\ \{lbackbone\}, Bound\_LDomain\ .$$
$$Extra ::= bind\ ?\ \{cross\_bb\}, backbone\ !\ \{cross\_bb\}, Bound\_Extra(cross\_bb)\ .$$

In each communication (Figure 1.37), a *Ligand*'s *Binding_Domain* sends its private *lbackbone* channel to an *Extra* process. Since the two *Binding_Domains* share the same private channel, following two such communications all four processes (two *Bound_LDomains* and two *Bound_Extra*) share a single private *lbackbone* channel, representing the ternary molecular complex. The private channel is further propagated to the other sub-processes, using each *RTK*'s own *backbone* channel.

The intra-cellular part of the RTK: The kinase domain

The intra-cellular part of the RTK is composed of a protein tyrosine kinase domain and several binding sites that include phosphorylatable tyrosine residues, that can be identified when phosphorylated by SH2 and SH3 domains on other proteins (Figure 1.38). For simplicity, we only model here one of the modifiable sites and the kinase domain, which we represent as two processes, $SH\_BS(tyr)$ and

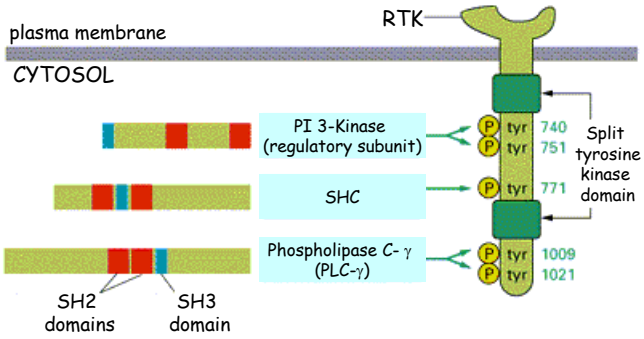

Figure 1.38: **The intra-cellular part of the RTK.** In addition to a split protein tyrosine kinase domain, the intra-cellular part of the RTK contains several phosphorylatable tyrosine residues. When phosphorylated, these residues can be specifically identified and bound by SH2 and SH3 domains in other proteins. Adapted from [5].

$Kinase\_Site$, composed in parallel:

$$Intra ::= SH\_BS(tyr) \mid Kinase\_Site \ .$$

We begin with the kinase domain. The kinase domain of the receptor may exist in one of three activity states: When the receptor is unbound and undimerized, the kinase domain is inactive. Upon binding and dimerization, the kinase becomes partly active, and is able to cross-phosphorylate a specific tyrosine residue on its counterpart's kinase domain, but not its own tyrosines or other targets. Once this tyrosine residue is phosphorylated the kinase is fully active, and can cross phosphorylate its counterpart's kinase domain, auto-phosphorylate its own tyrosine residues, or trans-phosphorylate tyrosine residues on other target proteins that are bound to it. We abstract these states as different processes (Figure 1.39).

- $Tyrosine\_Kinase$ represents an inactive kinase. We model the link between the intra-cellular kinase domain and the extracellular domain as communication offers on the private *backbone* channel. The link between the domains is critical upon ligand binding, and leads to partial activation of the kinase domain. The dependency on ligand binding is represented by the fact that the output offer on *backbone* in $Extra$ is possible only after communication with $Ligand$. Following communication, $Trans\_Active\_Kinase$ is called. The dimerization of the intra-cellular parts is represented by sending the private *ligand* channel to $Tyrosine\_Kinase$. This private channel is passed on as a parameter to $Trans\_Active\_Kinase$, and will be used by it for a communication representing cross-phosphorylation.

- $Trans\_Active\_Kinase$ represents a partially-active tyrosine kinase. To reach full activation it must be cross-phosphorylated on a tyrosine residue by its dimerized counterpart. The cross-phosphorylation event is abstracted as a symmetric interaction on the private *ligand* channel. As a result, both $Trans_Active_Kinase$s are transformed to $Full_Active_Kinase$s, with a *p_tyr* channel as a parameter. The parameter represents the phosphorylated state of the kinase domain's tyrosine residue. Note, that we consider cross-phosphorylation to be a single event, in which each kinase domain simultaneously phosphorylates and is phosphorylated by its counterpart. This

```
Tyrosine_Kinase::=
    backbone ? {ligand_bb} , Trans_Active_Kinase(ligand_bb) .

Trans_Active_Kinase(lig_bb)::=
   lig_bb ? [ ] , Full_Active_Kinase(p_tyr) ;
   lig_bb ! [ ] , Full_Active_Kinase(p_tyr) .

Full_Active_Kinase(res)::=
   res ? {modres} , Tyrosine_Kinase ;
   backbone ! {p_tyr} ,  Full_Active_Kinase .
```

Figure 1.39: π **calculus code for the RTK intra-cellular kinase domain**

assumption is reflected in the model by a single choice construct, which results in switching to $Full\_Active\_Kinase$, regardless of which process sends and which receives. The sequence of events leading from inactive receptor to a fully active one is depicted in Figure 1.40.

- $Full\_Active\_Kinase$ represents the fully active tyrosine kinase domain, that may participate in two types of interactions. First, it may be de-phosphorylated and de-activated by a phosphatase. Second, it may auto-phosphorylate other tyrosine residues on the receptor or trans-phosphorylate tyrosine residues on other molecules that bind to the receptor. We represent the former interaction by communication on the *res* parameter channel, which is initialized as a *tyr* one, leading to a $Tyrosine\_Kinase$ (inactive) process. We represent the second interaction as a communication on the *backbone* channel, in which the $Full\_Active\_Kinase$ sends the p_tyr channel, and is reconstituted. The counterparts for this communication may be either processes representing sites on the same molecule (and thus "own" the same *backbone* channel) or processes which represent sites on other molecules that are already bound to the receptor (and have received the *backbone* channel in the communication representing the binding events). We discuss the two types of counterparts next.

Intra-cellular binding sites and target molecules

There are two types of tyrosine-harboring target sites that can be identified and modified by the RTK. The first are sites located on the same receptor. These are immediately accessible to the fully-active kinase domain. The second are sites located on other molecules. To be phosphorylated, such molecules must first bind to the intra-cellular domain.

The molecules typically bind through SH2 or SH3 domains to specific phosphorylated tyrosines on the receptor. Thus, the intra-molecular phosphorylation sites also serve as modifiable binding sites for target molecules. The SH_BS process represents such an intra-receptor phosphorylation and binding site. It has a *res* channel parameter, which represents the modifiable residue, and is initialized as a *tyr* channel.

$$\begin{aligned} SH\_BS(res) \quad ::= \quad & backbone\ ?\ \{modres\}\ ,\ SH\_BS(modres)\ ; \\ & res\ !\ \{backbone\}\ ,\ Bound\_SH\_BS\ . \end{aligned}$$

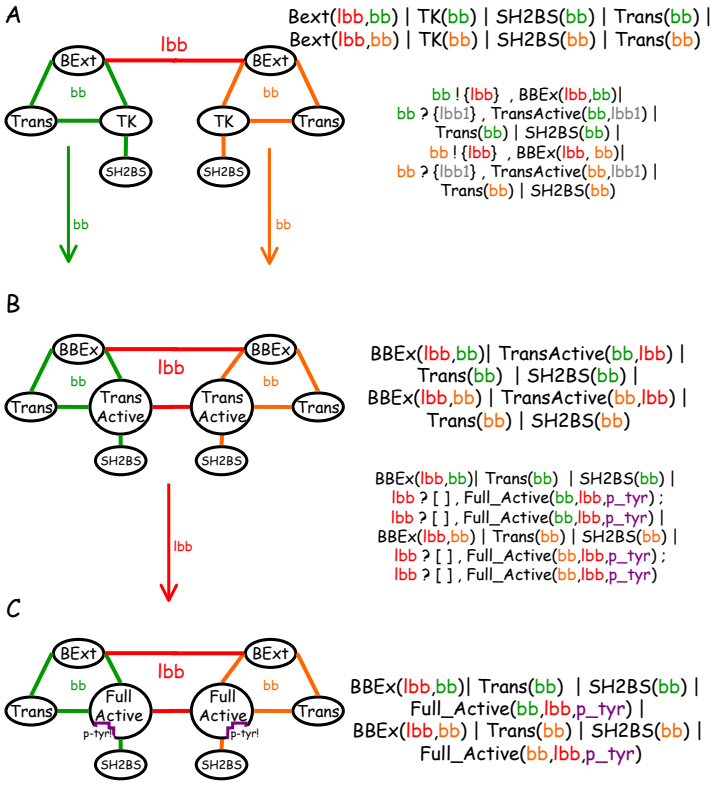

Figure 1.40: **Partial and full activation of the RTK Kinase domain.** A. Dimerization. The two $BExt$ processes share the *lbb* channel, representing the cross linking of the RTKs' extra cellular domains by the ligand. Each $BExt$ passes *lbb* to its cognate $Intra$'s $Tyrosine\_Kinase$ through the corresponding private *bb*, and the process changes state to $Trans\_Active$. B. Cross-phosphorylation. The two $Trans\_Active$ own the same *lbb*, passed from their cognate $BExt$s, and use it to cross activate each other, by communication and state change to $Full\_Active$. C. Each $Full\_Active$ may interact with its cognate $SH2\_BS$, through the corresponding private *bb*, or with a $Phosphatase$ (not shown) through p_tyr, which substitutes the res parameter.

$$Bound\_SH\_BS \;::=\; \mathbf{0}\ .$$

SH_BS may participate in one of two events. The first represents phosphorylation: On *backbone*, it may communicate with $Full\_Active\_Kinase$ and receive p_tyr, to replace the original tyr as the its parameter. The second represents phosphotyrosine specific binding: On the parameter channel it may communicate with other processes, send them the RTK's *backbone* channel, and become a $Bound\_SH\_BS$ (which we leave as an empty process for simplicity).

To understand how this communication represents phosphotyrosine-dependent binding, we must consider our representation of the SHC target molecule, as an SHC process.

$$\begin{aligned}
SHC(res) \;::=\;& (\mathsf{new}\ bb)\ .\\
& p_tyr\ ?\ \{backbone\}, Up_Bound_SHC(res, backbone, bb)\ .\\
Up_Bound_SHC(res, cross_bb, bb) \;::=\;& cross_bb\ ?\ \{modres\}\ ,\\
& \qquad Up_Bound_SHC(modres, cross_bb, bb)\ ;\\
& res\ !\ \{cross_bb\}, Down_Bound_SHC(res, cross_bb, bb)\ .\\
Down_Bound_SHC(res, cross_bb, bb) \;::=\;& \mathbf{0}\ .
\end{aligned}$$

The SHC molecule may interact either with a phosphotyrosine on the receptor (through its SH2 domain) or with an SH2 domain on a "downstream" Grb2 protein, through its own phosphotyrosine. We represent the first potential interaction as an offer to communicate on p_tyr and the second as an offer to communicate on the res parameter channel. We focus on the first interaction, representing phosphotyrosine-dependent binding of the SHC molecule to the phosphotyrosine in the receptor. Initially, tyr is the parameter channel of SH_BS, and it cannot communicate with SHC. Following the "modification" communication between $SH2\_BS$ and $Full\_Active\_Kinase$, a received p_tyr becomes the parameter channel of $SH2\_BS$ and it can communicate with SHC. During this communication, the *backbone* private channel is sent from $SH2\_BS$ to SHC. This channel can now be used for communication between $Kinase\_Active\_Full$ and SHC, representing the trans-phosphorylation of the receptor-bound molecule by the receptor's kinase domain.

The second interaction, a phosphotyrosine-dependent binding to a Grb2 molecule, is represented by exactly the same mechanism (a res parameter channel, to be changed from tyr to p_tyr upon interaction with $Full\_Active\_Kinase$ on the *backbone* channel) as was just described for $SH2\_BS$ process. The auto-phosphorylation, binding, and cross-phosphorylation cycle is shown in Figure 1.41.

The same mechanism — communication on a modifiable channel; propagation of a *backbone* channel to the communication partner; interaction with the $Full\_Active\_Kinase$ process on the *backbone* channel; change of the channel parameter; communicating with a new partner — can be re-used for each additional direct or indirect partner that joins the signaling receptor at the membrane.

Addressing representation problems in the RTK model

The model presented above suffers from several limitations and inaccuracies. First, all interactions with $Full\_Active\_Kinase$ and $Trans\_Active\_Kinase$ occur on private channels (*ligand* and *backbone*). This reflects the fact that binding and complex formation (between the receptors, between a receptor and a

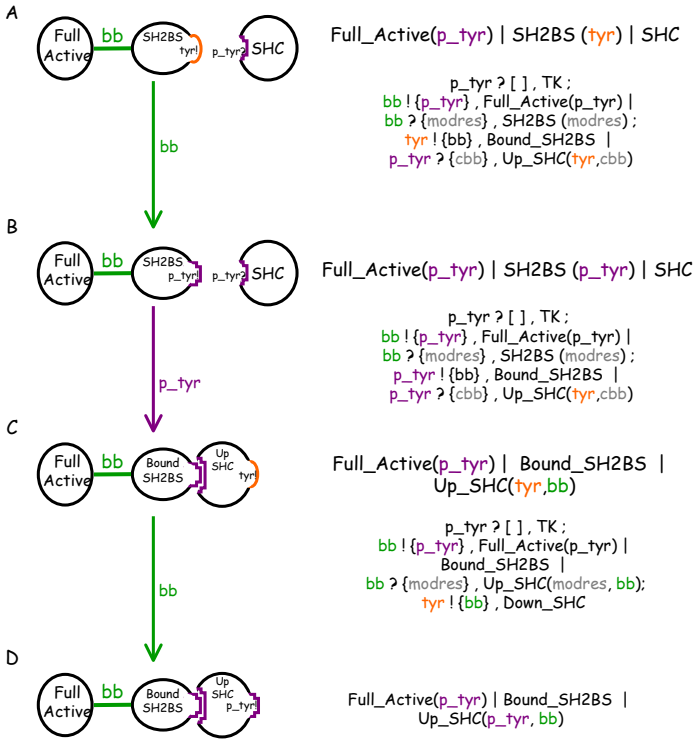

Figure 1.41: **Auto-phosphorylation, binding and trans-phosphorylation at the RTK.** A. Auto-phosphorylation: *Full_Active_Kinase* sends *p_tyr* to *SH2_BS*, through the private *bb* channel they share. B. Modification dependent binding. Following communication, *SH2_BS* is re-created with *p_tyr* instead of the previous *tyr*. The acquired *p_tyr* allows communication with *SHC*, in which the private *bb* is sent from the *RTK*'s *SH2_BS* to *SHC*. C. *Fully_Active_Kinase* utilizes the private *bb* channel to interact with *SHC*, and send it *p_tyr*. D. *SHC* is re-created with *p_tyr* instead of the previous *tyr*. This would allow further downstream interaction. The *p_tyr* motif of the receptor itself is not shown. Note, that this model is partly incorrect, as various phosphorylated motifs are represented by the same channel, allowing incorrect interactions. Furthermore, interaction between *Kinase* and other processes depends only on proximity (*bb* channel) but not on molecular complementarity. For further discussion, see the text.

target molecule) precede enzymatic modification. However, this model ignores the required structural complementarity between the kinase and its target molecule. A more complete model would represent both the physical proximity requirement and the complementarity one. Since we typically represent complementarity with public channels, we need to match two channels (a public and a local channel) simultaneously, but this is impossible in the current calculus. For the time being, we address the problem by combining a communication on the private channel and a match construct on the public one.[23] A principled solution to this problem is presented in Chapter 3, where we introduce ambients as an abstraction of compartmentalization.

Second, the presented model treats all the tyrosines and phosphotyrosine as equivalent (representing all of them as *tyr* and *p_tyr*). In a large model this would inevitably lead to the non-specificity of communication. For example, the *p_tyr* channel of the SH_BS process can serve to communicate with any other $SH2$ process. This inaccuracy is further pronounced when we consider the properties of SH2 domains. In reality, SH2 domains have distinct sites for recognizing a combination of a phosphotyrosine and a particular amino acid side chain. Thus, different SH2 domains recognize phosphotyrosine in the context of different flanking amino acids. As we represent a motif as a public channel, a proper representation of SH2 molecular recognition would require matching of two different public channels simultaneously (impossible in the current language). To address this problem we can use the *res* channel only as an internal parameter, signifying the "state" of the molecule, and an additional parameter channel offered in an interaction communication to other processes, to signify the actual motif. For example, we write the SH_BS and SHC processes as

$$
\begin{aligned}
SH\_BS(res) \;&::=\; res = / = p\_tyr \;,\; backbone\;?\;\{modres\} \;,\; SH\_BS(modres)\;; \\
&\quad\; otherwise \;,\; sh\_ptyr\;!\;\{backbone\} \;,\; Bound\_SH\_BS\;. \\
Bound\_SH\_BS \;&::=\; \mathbf{0}\;.
\end{aligned}
$$

$$
\begin{aligned}
Free\_SHC \;&::=\; (\mathsf{new}\ bb)\;.\; SHC(tyr, bb). \\
SHC(res, bb) \;&::=\; res = / = p\_tyr \;,\; sh\_ptyr\;?\;\{cross\_bb\}, \\
&\qquad\; cross\_bb\;?\;\{modres\} \;,\; SHC(modres, cross\_bb); \\
&\quad\; otherwise \;,\; grb\_ptyr\;!\;\{bb\}, Grb\_Bound\_SHC(res, bb). \\
Grb\_Bound\_SHC(res, bb) \;&::=\; \mathbf{0}\;.
\end{aligned}
$$

Finally, the model does not represent the reversibility of binding and modification (of ligand from receptor, of target molecules from receptor, etc). This, however, is not an inherent limitation, and was rather introduced to simplify the presentation. Additional private channels exchanged in the original communication between processes can be used to represent an "unbinding" communication, as presented in the hetero- and homo-dimerization examples above. Processes representing phosphatase molecules can be added, analogous to the kinase processes, which send *tyr* channels to replace the *p_tyr* ones, as described in previous examples. Such "reverse" communications are part of the full model [106] used for

[23] In a nutshell, the processes first communicate on the private channel, and the public channel name is sent from one process to the other. Then - the receiving process compares the received channel name to its own public channel. If the two match, the reaction proceeds (*e.g.* state change). Otherwise, the processes are reconstituted without any change.

the simulation studies below.

1.6.2 Semi-quantitative simulation studies of the pathway

To study the behavior of our modeled system, we ran semi-quantitative simulations using the BioSpi 1.0 system. In these simulations, we examine the effect of different changes (perturbations) in the system on its "output", as measured by the number of $Target\_RNA$ processes (see Figure 1.10). We distinguish between two types of perturbations: *quantitative perturbations*, in which the quantity of one or more of the system's components is changed, and *qualitative perturbations* in which the behavior of one or more processes is modified. As discussed in Section 1.5, although we measure the actual number of process, the simulation is only semi-quantitative since the system picks the first choice among available alternatives, and does not include a mechanism to abstract different reaction rates.

Quantitative perturbations

The quantitative perturbations focus on the MAPK cascade components of the pathway: Raf, $MP1$, Erk, and Mkp. In each case we varied the initial quantity of one or more of these components, and examined their effect on the pathways "output" (the level of the $Target\_RNA$ level), comparing it to a baseline run. The results are summarized in Table 1.2.

Varied process	**Change details**	**Output mean**	**Output max**	**Comments**
	Baseline	49	95	
$MP1$:$Mek1$ (baseline)	1:2	18	50	Low levels of scaffold increase the output.
	3:1	53	105	
	4:1	44	103	
	7:1	7	22	High levels of scaffold reduce the output.
	10:1	0	0	
Raf	10X increase	56	109	No marked effect on output
$Mek1$ and $MP1$	10X increase	87	188	Moderate increase in output
$Erk1$	10X increase	9	26	Strong reduction in output
$Mkp1$	30X increase	38	89	Moderate reduction in output

Table 1.2: **Quantitative perturbation of the RTK-MAPK model.** Semi-quantitative simulations of the RTK-MAPK model were carried with the BioSpi 1.0 system for 20,000 steps each time. In each simulation, the initial number of one or more of Raf, $MP1$, Erk, and Mkp processes was changed. Process numbers were traced throughout each run. The average and maximum number of the $Target\_RNA$ process is shown in the table as a measure of the system's output. All values are shows relative to the following baseline values: 45 ATP processes, 100 GTP, 15 GDP, 6 $Ligand$, 12 RTK, 6 Ras, 2 GAP, 3 Shc, 3 Grb, 3 Sos, 6 Raf, 10 $MP1$, 10 $Mek1$, 15 $Erk1$, 4 $Mkp1$, 18 Apc.

Some of the results indicate that the pathway model yields the expected behavior under perturbations. For example, an increase in the number of $Mek1$ processes increases the output of the system, while an increase in the number of $Mkp1$ processes (representing MAP kinase phosphatase) reduces the output level. We take these as evidence for the proper working of the model.

At closer inspection, we observe several interesting phenomena in the results. For example, a disproportionate increase in the level of $Erk1$ compared to the other components (10X) leads to a marked reduction in the output. When examining the level of other processes in the simulated system, we observe that the level of $Active\_Mek\_Kinase$ is reduced 5-fold. We can envision two alternative reasons

for this effect. First, $Active\_Erk\_Kinase$ may interact with $Active\_Mek\_Kinase$, resulting in conversion of the latter to an $Inactive\_Mek\_Kinase$. Second, Erk_MP1_BS may compete with Raf_MP1_BS on interactions with $MP1$ (the scaffold). By examining a detailed step-by-step *trace* of the simulation, we noticed that $MP1$ indeed preferentially interacts with Erk_MP1_BS rather that with Raf_MP1_BS. Thus, excess $Erk1$ processes compete with $Raf1$ processes on $MP1$, indirectly reducing the number of interactions between $Mek1$ and $Raf1$, and the resulting conversion of the former to $Active\_Mek\_Kinase$. Correspondingly, we can postulate that in the biological pathway, over expression of Erk1 protein may have an inhibitory effect. Unfortunately, as previous studies of the MP1 scaffold (*e.g.* [118]) employed a constitutively active Mek1 protein, we cannot support this finding at the moment.

The effect of changes in the level of $MP1$ relative to that of $Mek1$ is also interesting. At levels close to those of $Mek1$, an increase in $MP1$ amounts results in an increase in pathway output. Further increases in $MP1$ beyond the level of $Mek1$ (3-fold, 4-fold) have little further effect on the output. However, large additional increases (7-fold, 10-fold) have an inhibitory effect on the signal. Comparing traces of the different runs indicates that at high levels, the $MP1$ process "sequesters" the Mek, Erk, and Raf processes from one another. This observation is consistent with previous experimental (*e.g.* [118]) and theoretical results ([76], [16]) indicating that molecular cross linkers in general (and MP1 in particular) can promote the formation of a multi-molecular complex at low concentrations and inhibit it at high concentrations.

Qualitative perturbations

In qualitative perturbations, we maintain the initial quantities of the component processes, while changing the code describing the behavior of one or more processes. In doing this we extend the abstraction underlying the construction of the model, to perform further manipulations, analogous to mutational analysis in laboratory experiments. Thus, the deletion, insertion or modification of domains or motifs in proteins are represented by removal, addition or channel name changes in processes and communication offers.

For example, a code representing a mutant monomer ligand (Mut_Ligand process) is generated from the dimeric ligand code by removing one of the two $Binding\_Domain$ processes. Similarly, a code for a mutant receptor ($Intra\_Mut\_RTK$ process) is obtained from the wild type RTK code by removing the $Intracellular$ sub-process.

$$
\begin{aligned}
Ligand &::= (\mathsf{new}\ lbackbone)\ .\ Binding\_Domain \mid Binding\_Domain\ .\\
Mut\_Ligand &::= (\mathsf{new}\ lbackbone)\ .\ Binding\_Domain\ .\\
RTK &::= (\mathsf{new}\ backbone)\ .\ Extra \mid Transmem \mid Intra\ .\\
Intra\_Mut\_RTK &::= (\mathsf{new}\ backbone)\ .\ Extra \mid Transmem\ .
\end{aligned}
$$

Note, that as we build complex processes from sub-process in a gradual manner, our manipulations of the code can be similarly gradual. For example, deletion of the modifiable (SH2-bound) tyrosine site in the RTK can be represented by elimination of the SH_BS site.

$$Intra \quad ::= \quad SH\_BS(tyr) \mid Kinase\_Site$$
$$Intra\_Mut \quad ::= \quad Kinase\_Site$$

More specific changes can be represented by manipulating channel parameters and communication offers. For example, in the (wild type) biological pathway, Raf is activated by membrane re-targeting as a result of an interaction with Ras, and a constitutively active Raf can be obtained by directly targeting it to the membrane. In the model, we represent the cytosolic and membranal compartments by the corresponding private *cyt_env* and *mem_env* channels, that are passed to the relevant processes as parameters. To represent a membrane-localized (constitutively active) Raf molecules, it suffices to call the Raf process with the *mem_env* channel ($Raf(mem\_env)$) rather than the *cyt_env* one. In another example, to model a constitutively active receptor, we combine a change in the process (using the $Full\_Active\_Kinase$ process instead of the $Kinase\_Site$ one) with an internal change to the $Full\_Active\_Kinase$ process, removing the communication offer that allowed interaction with *Phosphatase* processes.

Using these approaches we introduced qualitative changes in the *Ligand*, RTK, Raf and $Erk1$ processes, and studied their effect using semi-quantitative BioSpi simulations, as described in Table 1.3.

Modified process	**Change details**	**Output mean**	**Output max**	**Comments**
	Baseline	49	95	
Mut_RTK	Inactive receptor: Removal of $Intra$ sub-process; 1:1 RTK and RTK_Mut processes	9	32	Mutant receptor reduced the output
Mut_Rtk_Const	Constitutively active receptor: $Full\_Active\_Kinase$ (without "state switch" action) as a direct sub-process of $Intra$	35	78	Active even in lack of *Ligand* processes
Mut_Ligand	Monomer ligand: Only one $Binding\_Domain$ sub-process; 1:2 *Ligand* and Mut_Ligand processes	0	0	Mutant monomer ligand reduced the output
Mut_Erk1_Const	Constitutively active Erk1: Removal of $Erk1\_Lip$ sub-process and initialization with $Active\_Erk\_Kinase$ process (without "state switch" action)	146	218	Output even in lack of *Ligand* processes
Mut_Raf	Membrane localized Raf: Raf process initialized with *mem_env* private channel (shared with RTK, Ras, and GAP processes)	56	109	Output even in lack of *Ligand* processes

Table 1.3: **Qualitative perturbation of the RTK-MAPK model.** Semi-quantitative simulations of the RTK-MAPK model were carried with the BioSpi 1.0 system for 20,000 steps each time. In each simulation, the code for one or more of the processes was modified to reflect different mutations. Process numbers were traced throughout each run. The average and maximum number of the $Target\_RNA$ process is shown in the table as a measure of the system's output.

As expected, the presence of Mut_RTK (representing an inactive receptor) and Mut_Ligand (representing a monomeric ligand) processes had a reducing effect on the signal, while adding processes representing constitutively active molecules (Mut_Rtk_Const, Mut_Erk_Const and Mut_Raf) resulted in signal even in the absence of any $Ligand$ processes.

Interestingly, the effect of the different "constitutive" processes was not identical. The effect of Mut_Erk_Const processes was more pronounced than that of Mut_RTK_Const, and the inclusion of $Raf\_Mut(mem\_env)$ had an even weaker and more fluctuating effect. Closer examination of the run traces explains these differences. The RTK signal is insufficient to switch all $Erk1$ processes to an "active" state, as interaction of the active $Erk1$ processes with $Mkp1$ processes counteracts these effects. As the Mut_Erk1_Const process has no "inactive" state, a higher amount of "active" $Erk1$ processes exists, resulting in the generation of increased output signal. Mut_Raf, on the other hand, must interact with Ras, following its membrane localization. Ras itself switches between two activity states, and the balance between them is affected indirectly by RTK's state. Thus, in the absence of $Ligand$, Mut_Raf can elicit only a partial response.

1.7 Perspective: Molecules as computation

The behavior of biomolecular systems is traditionally studied with *Dynamical Systems Theory* (described via differential equations) [36], which abstracts the cell and its molecular constituents to their quantifiable properties (e.g. concentration, position) and their couplings. While this approach is powerful, clear, and well understood, the abstraction it makes is often cumbersome for describing the behavior of complex biomolecular systems. Primarily, it suffers in the "Relevance Test", since it treats the cell as a monolithic unstructured entity, and does not identify molecular *objects* as they are modified. Biological systems are composed of dynamic molecular *objects*, but Dynamical Systems Theory is not compositional. Thus, rather than capturing the identity of a single molecule, molecular complex or machine with several states (and/or sub-components) as a single entity, it represents it by a multitude of entities and follows its fate only indirectly.

1.7.1 The molecule-as-computation abstraction

While Dynamical Systems Theory is limited in its ability to abstract systems composed of dynamically changing objects, computer science offers a good starting point in the search for such good abstractions. Computational behavior is described in terms of the composition of computational objects, commonly called processes, and the notions of construction, modularity, and hierarchy are key to these descriptions. Processes may interact with one another and change as a result of this interaction, a behavior governed by reaction rules. They may also be composed to form complex entities in a hierarchical fashion.

This view of computational systems as composed of interacting processes, renders them a tempting source of novel abstractions for the behavior of systems of interacting biological entities molecules, such as molecules and cells. In this chapter, we began the search for this abstraction with the π-calculus, a process algebra for the representation and study of *concurrent mobile systems*. In such systems processes communicate with each other on specific channels, passing further channel names which can be used for subsequent communication, thereby dynamically changing inter-process "wiring".

As a first step we identified the basic entities of biomolecular systems (populations of molecular species and their component domains and motifs) and the events in which they participate (specific

interactions followed by reconstitution, biochemical modification, and/or a change in molecular state). We then developed general guidelines for the abstraction of these entities to the mathematical domain of the π-calculus, abstracting molecules and their domains as computational processes and sub-processes, complementary motifs as complementary actions on communication channels, and molecular interaction and modification as communication, message passing, and process (re)creation. Compartmentalization (single molecule, complex, and cellular compartment) was uniformly abstracted with private channels.

We employed these guidelines to construct abstract representations of various use-cases, including reversible hetero- and homo-dimerization, enzymatic reactions and competitive inhibition, signaling by modification of protein residues, and transcriptional regulation. We then built an abstract model for the receptor tyrosine kinase (RTK) - MAP kinase (MAPK) signaling pathway.

The use cases and the pathway model demonstrated the expressive power of our abstraction and its ability to map a variety of molecular scenarios to a concise and formal mathematical domain. In general, we were able to limit ourselves to the original calculus and show how a language, developed to represent a purely computational realm, can perform surprisingly well when co-opted as an abstraction of a biological domain. In two notable exceptions we use extensions of the original calculus: we adopted the match/mismatch prefixes of [98] and we allowed basic arithmetic operations on logic variables into the language. The latter is a novel addition, and was done in order to facilitate otherwise elaborate encoding of simple operations. Both changes provide communication-independent means to decide on the "fate" of processes. None of the additions however changes the basic abstraction guidelines we developed. We thus conclude that a computational abstraction of complex biomolecular systems with a concise calculus is feasible.

1.7.2 Simulating the behavior of abstracted biomolecular systems: BioSpi 1.0

The quality of the abstraction we constructed must meet several criteria other than its pure power to express a variety of properties and scenarios in biomolecular systems. One critical test of both the utility of the abstraction and its correctness is our ability to simulate the behavior of biomolecular systems by executing its computational abstraction.

To this end, we developed the BioSpi application for the simulation of π-calculus systems. BioSpi is based on the Logix system [128], which implements Flat Concurrent Prolog (FCP [125]). Two unique features of FCP made it suitable for our purposes. First, the ability to pass logical variables in messages was used to implement the mobility ("sending channels as messages") mechanism of the π-calculus. Second, its support of guarded atomic unification allowed the implementation of synchronized interaction with both input and output guards. Note, that previous implementations of the π-calculus or of related formalisms (*e.g. [89]* and references therein) do not provide such full guarded synchronous communication.

An appropriate surface syntax was devised for the π-calculus syntax [106]. The ASCII-based[24] syntax was devised in such a way that the original π-calculus primitives and operations (including the match/mismatch constructs) are clearly separated from the arithmetic ones. Thus, "pure" and "added" components can be clearly identified. Furthermore, the syntax is insulated from general Logix procedures,

[24]In order to maintain pure ASCII representation and comply with some limitations of the Logix system, some of the original π-calculus notation was replaced in the BioSpi syntax. Only BioSpi notation is used throughout this thesis, except when noted otherwise. We note, that the (new x) construct used in this chapter in not the proper (syntactic) construct used in BioSpi. Rather, the process $P ::=$ (new x) . $P1$. is replaced in BioSpi code with $P + x ::= P1$.

and a pure π-calculus representation is maintained in spite of the use of an FCP-based platform.

We built a compiler from BioSpi to FCP. In BioSpi 1.0, each π-calculus channel is an object (a persistent procedure) and each π-calculus process is transformed to an FCP process. BioSpi processes send communication offers to the channel. There are four kinds of offers: send, receive, send & receive (for symmetric communication), and withdraw. Each time that a new event is required the central BioSpi monitor selects among the various enabled channels (for which both a send and receive offer are available – but not from the same process). The chosen channel completes one transmission (send/receive pair), relaying the sent message to the receiver. The completion of the send and receive requests is synchronized by the channel. In addition, other messages offered on this and other channels by the same two processes whose requests were completed, are withdrawn, implementing a mutually exclusive choice. The withdrawals are not synchronized, but they do precede continuation of their respective processes. The channel set of each BioSpi process – public channels, arguments, newly declared channels and input bound "placeholders" – is identified and associated with the corresponding FCP procedure. This allows full use of channels as in the original calculus. Note, that a reference count is maintained in each channel. The count is incremented when the channel is passed to more than one process (including itself, recursively) and decremented when a process no longer uses the channel. When the reference count is zero, the channel object terminates, ensuring the reclamation of channels that will no longer be used.

BioSpi 1.0 allows semi-quantitative simulations, in which communication (*i.e.* reaction) rates are immaterial, but process (*i.e.* molecule) numbers are. The results of BioSpi 1.0 simulations of the RTK-MAPK pathway show the utility of semi-quantitative simulations in studying molecular systems. For example, some simulations correctly captured the effect of certain perturbations on the pathway's (quantitative) output. Others have indicated interesting avenues for further research. For example, our quantitative perturbation studies suggested an inhibitory effect following a disproportionate increase in the pathway's MAP Kinase, Erk1. They also indicated a dual role for the scaffold molecule, MP1, promoting signaling at lower levels and inhibiting it at higher ones. Some of the qualitative perturbations, affecting the behavior of one or more pathway components, have also yielded interesting results. For example, we observed different effects following the introduction of processes representing different constitutively active pathway components. The simulation studies predict that the effect of a constitutively active Erk1 molecule may be more pronounced than that of a constitutively active receptor, and that the effect of a mutated Raf molecule would be even weaker and more fluctuating. Importantly, the detailed run traces provide explanations for such results, allowing us to identify competition, sequestration and dependent events that account for the final outcome.

While useful, semi-quantitative simulations are only partly correct as they ignore the different rates of biochemical reactions, leading to two related inaccuracies. First, in a semi-quantitative simulation all individual reactions are equally likely to occur, while in the real-world faster reactions would occur more frequently. Second, all time steps in a semi-quantitative simulations are equal and do not represent the time evolution of the real system. Exact quantitative modeling, which can underly accurate quantitative simulations requires us to extend our abstraction to accommodate reaction rates and time. As we show in the next chapter, this can be achieved by a natural extension of the π-calculus and the associated abstraction, which defines a stochastic semantics to interactions, while maintaining the rest of the calculus intact.

Both semi-quantitative and fully quantitative simulations of molecular systems as computational ones can also suffer from performance and scaling problems. The main difficulty lies in the abstraction

of each molecule as a separate *instance* of a computational process. As the number of processes in the computational system is directly proportional to the number of molecules in the biological one, we may have to tackle prohibitively large computational systems. One solution is to handle only *molecular species* explicitly, while the specific molecules are represented by counter variables. While this approach significantly improves performance and scalability, it loses some of the immediate transparency of the "molecule-as-process" abstraction.

1.7.3 Benefits and limitations

How relevant, computable, understandable and extensible is the "molecule-as-computation" abstraction we devised?

Relevance

A relevant abstraction of biomolecular systems should capture two essential properties of these systems in one unifying view: their molecular organization and their dynamic behavior. We believe that the π-calculus abstraction is indeed highly relevant. On the one hand, essential biomolecular entities are directly abstracted to computational ones and the abstraction handles a variety of biomolecular events. On the other hand, the dynamic behavior of the derived computational system closely mimics that of the molecular one.

The two alternative (non-mobile and mobile) approaches we present for abstracting molecular interaction offer a tradeoff between relevance and utility. The first, non-mobile, approach is often simpler, but suffers from a relevance problem, similar to that of Dynamical Systems Theory, as each state and modification are represented as an individual entity. The second approach combines mobility with channel parameters to represent chemical modification as a change in the process' channel set. Thus, processes are under-defined in such a way that allows them to assume different states based on the channels they received. While we believe this approach is more relevant, we also appreciate the greater difficultly to specify and understand mobile systems. In practice, when building individual abstractions we use a mixture of the two approaches, guided by pragmatic considerations.

Computability

A computable abstraction should allow both the simulation of dynamic behavior and qualitative and quantitative reasoning on systems' properties. We have seen that with the aid of BioSpi we can simulate the behavior of biomolecular systems by executing their computational π-calculus abstraction (semi-quantitatively as shown here, or fully quantitatively as presented in the next chapter).

A π-calculus abstraction also supports qualitative and quantitative reasoning on the modeled system's properties. An extensive behavioral theory has been developed for the mathematical domain of the π-calculus (and similar languages), prior to and regardless of its proposed use for biological systems. This provides methods and tools to *formally verify* certain properties of systems described in the π-calculus (*e.g.* [96], [145]). In principle, we can verify certain qualitative assertions on biomolecular systems (*e.g.* "will a signal reach a particular molecule?") by verifying a corresponding assertion on the π-calculus abstraction of the system (*e.g.* "will a particular communication be realized in a particular process?"). Furthermore, methods exist to *compare* two π-calculus program to determine the degree of

mutual similarity of their behavior, termed *bi-simulation* ([89], [96], [145]). Different levels of similarity, of weakening strength, have been defined [89].

While such methods have been generally developed they have not been applied in the past to systems at the scale and complexity of the ones we are building as abstractions of biomolecular systems. Furthermore, the types of queries and analysis methods developed for computational purposes may not be the same necessary to elucidate biological questions. In particular, stochastic and quantitative issues (such as the ones explored in the next chapter) were hardly handled in the purely computational setting.

Thus, while we deem these computable aspects of the "molecule-as-computation" abstraction as critical components of its success, they are beyond the scope of this thesis, and will require further extensive research. Nevertheless, the potential computational possibilities opened by the use of the π-calculus serve as additional support for its importance. Indeed, in very recent work several researchers have started to adapt analysis methods from process algebra to the study of biomolecular systems (*e.g.* [27], [24], [97], [8]).

Understandability

An understandable abstraction offers a conceptual framework for thinking about the scientific domain: it corresponds well to the informal concepts and ideas of science, while opening up new computational possibilities for understanding it. The properties which lend relevance and computability to the "molecule as computation" abstraction also render it understandable. On the one hand, the abstraction attempts to translate our informal knowledge of the biological realm to formal concepts in a transparent and visible way. This is demonstrated by the ability to qualitatively perturb the behavior of process in a manner analogous to mutational analysis in a molecular biology experiments: sub-processes (domains) and channels (motifs) are removed, added, or modified.

On the other hand, the computational theory for the study and analysis of concurrent computational systems described above open up new possibilities for understanding molecular systems. For example, concurrency theory distinguishes between two levels to describe a system's behavior: the implementation (how the system is built, say the wires in a circuit) and the specification (what the system does, say a logical 'AND' gate). Once biological behavior is abstracted as computational behavior, an implementation can be related to a real biological system, for example the MAP kinase cascade, and the corresponding specification to its biological function, for example, an amplifier or a switch. In this context, ascribing a biological function to a biomolecular system is no longer an informal process but an objective measure of the semantic equivalence between a low-level and a high-level computational description. Similarly, two molecular-level abstractions of similar systems in different cells or organisms can be compared for their behavioral similarity, to determine their level of *homology* (if evolutionary related) or *analogy* (if evolutionary distinct). These possibilities (though beyond the scope of this thesis) motivate our study of the circadian clock in Chapter 2 and are further discussed there.

The abstraction we presented in this chapter however is still obscure in certain respects, as we saw for example in the RTK-MAPK model (Section 1.6.1). The use of private channels as an abstraction of biomolecular compartments raises several problems including cumbersome, multi-step encoding of the formation of multi-molecular complexes, and the need to use match constructs to simultaneously check for both proximity and complementarity. Another limitation lies in the need for each communication to involve exactly two processes. While intra-molecular events can still be modeled by using sub-processes, interactions between more than two molecules cannot be abstracted by a single event in the computa-

tional realm. While this limitation is biochemically correct (elementary termolecular interactions are extremely rare), it may raise problems in abstracting systems for which knowledge is limited and cannot be broken down to elementary processes. Another current limitation to the transparency and utility of the abstraction is its purely textual nature and the lack of a graphical component for the specification and visualization of the abstracted system and its unfolding.

Extensibility

Such limitations in the relevance and understandability of the abstraction can often be overcome by appropriate extensions to the abstraction or to the mathematical domain. As we do not expect a single concise computational language to be an immediate all-inclusive solution, the desired abstraction (or the mathematical domain) should be easily extensible to capture additional real-world properties without introducing major changes to the core abstraction. The well-defined and concise domain of the π-calculus is relatively amenable to extensions and has been extended for computational purposes. Indeed, in the following chapters, we extend the calculus and the abstraction to improve its relevance and understandability, by adding a stochastic semantics to handle quantitative aspects (Chapter 2) and *ambients* to handle compartments (Chapter 3).

A graphical extension or translation of the abstraction is beyond the scope of this work. Note, however, that graphical variants of the π-calculus have been previously proposed (*e.g.* [88]) and can serve as a starting point for the development of a graphical extension. Alternatively, the abstraction may be translatable into other well-developed graphical formalisms, such as the unified modeling language (UML, [103]).

Comparison to other representations

The two prominent existing computational approaches to biomolecular systems are "dynamical systems theory", based on the "cell as a collection of molecular species" abstraction, and the functional pathway databases, that employ the "molecule as object" abstraction. In order to render the former abstractions understandable another layer of representation (graphics, object scheme etc.) is typically required; while in order to render the latter computable an additional level of semantics and implementation is required used that adds dynamics and behavioral properties to an essentially static scheme. Karp [67] related the two approaches to two types of formal representations. *Declarative representations*, such as ontologies, break information down into atomic components and define relationships among those atomic components. They allow us to represent and store current knowledge in an exact non-ambiguous form, and then query and process it reliably. *Procedural implementations*, such as those embodied in most simulation programs, typically lack such well-structured representation of knowledge, and information is embedded in a convoluted fashion.

Abstract computer languages, such as the π-calculus, offer a synthesis of both declarative and procedural representations. On the one hand, the molecular world is clearly abstracted into the computational one, in a way which is as relevant and understandable as possible. On the other hand, the generated abstract model is computable allowing both simulation and analysis of the modeled system's behavior.

How does the π-calculus abstraction compare to that offered by other, related languages, such as Petri nets [45], Statecharts [65] and linear logic [36], that are now being used to represent various biomolecular systems? While the abstraction to linear logic has only been rudimentary developed [36], both Petri

Nets and Statecharts are actively employed, and must be considered as alternatives when evaluating our π-calculus based approach.

Petri nets (*e.g.* [45]) are one of the first uses of the "molecule as computation" abstraction, and have been primarily successful as an abstraction of metabolic systems. The computability of this abstraction is well-developed, both in terms of simulation tools (*e.g.* [45]) and verification ones [43]. Furthermore, it is accompanied by a pleasing graphical representation which greatly enhances its utility (and which the π-calculus lacks). However, the Petri Net based abstraction is close to a "cell as collection of molecular species", and suffers from the same relevance problems: it does not allow for the structured representation of molecular objects (which the π-calculus does).

Statecharts [65] are a highly relevant and understandable model of biomolecular and multi-cellular systems, providing a structured and intuitive framework for the abstraction of biomolecular systems. The graphical nature of the language is much more developed than the Petri net one and very useful, and a variety of computational tools exist for qualitative simulation and verification. Despite these major advantages, the abstraction lacks in understandability as, unlike the π-calculus, it is a purely qualitative framework, and does not currently accommodate process quantities or reaction rates.

A full understanding of these differences requires further dedicated research. We believe such research is important as a first step towards distilling and synthesizing an optimal abstraction of molecules as computations.

Chapter 2

The biochemical stochastic π-calculus: A quantitative extension

2.1 Introduction

The abstraction presented in the previous chapter is lacking in certain respects. In particular, the original framework of the π-calculus is semi-quantitative by definition: all *individual* communications are equally likely to occur. Thus, the π-calculus abstraction of the molecular realm reflects the number of molecules, but not the rates of the reactions in which these molecules participate. This results in two inter-dependent inaccuracies. First, reactions do not occur at the correct rate. Second, all time steps are equal and do not represent the time evolution of the real system.

Previous studies with qualitative (*e.g.* [86]) and semi-quantitative (*e.g* [53] and our work) modeling of biomolecular systems show that even such inaccurate abstractions have some merit, for two reasons. First, in certain cases, qualitative aspects, such as a biomolecular network's architecture and connectivity, are the critical aspects in its function [4]. Second, the quantitative parameters of many biomolecular systems are currently under-specified and lacking, and a workable, albeit only partially correct, abstraction is still useful.

Nevertheless, quantitative aspects are key to the function of many biomolecular systems, a fact that is recently gaining growing recognition (*e.g.* [72]), and must be appropriately addressed when devising a correct abstraction of biomolecular systems. Since we model chemical reactions via communication, exact quantitative modeling of reaction rates requires us to extend the mathematical domain to accommodate communication rates and an explicit notion of time, and to adapt our abstraction accordingly to map reaction rates and real-world time to their mathematical counterparts. Fortunately, this extension can be done seamlessly, while maintaining the original calculus and abstraction intact.

In general, we distinguish between two steps in the abstraction of a biomolecular system [41]: (a) creating a complete description of the relevant chemical and biological processes involved (*i.e.* at the relevant level of abstraction); and (b) computing on the abstracted representation to generate predictions and insight. In principle, the first descriptive step can be accomplished in a way dissociated from the second, and is more fundamental. For example, in the first step a chemical reaction can be abstractly represented as $A + B \rightarrow C$. In the second step, the same representation may be interpreted in different ways to compute the behavior of such a system. One may *assume* that there are sufficiently

[0]The work in this chapter was partly published in [109] and [107].

many molecules such that the number of molecules can be approximated as a continuous variable which changes deterministically over time. In such a case, the mathematical tool of differential equations allows us to predict the concentration of each molecule as a function of time. Alternatively, the same abstract representation may be interpreted as a well-mixed system, which is represented by the *number* (rather than concentration) of each molecule, a discrete variable. The *stochastic* behavior of these variables over time can be predicted by a probability function.

Similarly, in the π-calculus abstraction, we have associated our representation so far with a particular non-deterministic and discrete semantics. While the discrete aspect of the calculus is fundamental, the non-deterministic semantics is not, and will now be replaced with a stochastic one, in which different communications have different rates, and communications are selected based on probabilistic conditions.

2.1.1 The need for stochastic modeling

Any molecular system is essentially a discrete one at the molecular level: individual molecules interacting with each other. In many cases, however, biochemical reactions are abstracted as a system of continuous concentration variables, assuming that molecule numbers are large. When the concentrations of the reacting species are low (and some of the reactions rate a significantly slower than others), the continuous deterministic abstraction breaks down and may not correctly describe the behavior of the real-world system [9]. This situation is common in some biomolecular systems and several studies have shown that a stochastic framework is essential to study them (*e.g.* [91], [13], [9], [146]). Signaling pathways commonly operate close to points of instability, frequently employing feedback and oscillatory mechanisms that are sensitive to the operation of small numbers of molecules (*e.g.* [13], [49], [44]). For example, only 200 K^+ and Na^+ channels responsive to changes in intra-cellular Ca^{2+} are responsible for a key step in many neutrophil signaling pathways [49]. Genetic regulatory mechanisms also typically involve small quantities of regulatory molecules, and only one or two copies of most individual genes. The rates of genetic reactions are relatively slow, such that many minutes may separate the successive initiation of individual transcripts from an activated promoter [9]. The stochasticity of gene expression mechanisms may underly numerous unexplained observations of phenotypic variation in isogenic or clonal populations [9]. It also leads to an intrinsic noise in the operation of the networks, and it has been suggested that various functional adaptations in the network design have evolved to buffer this noise and even take advantage of this property (*e.g.* [134], [146]).

2.1.2 Algorithms for stochastic simulation

Any abstraction of biomolecular systems with low quantities and small reaction rate thus requires a discrete stochastic computational framework.[1] Two types of algorithmic approaches were previously developed to simulate the time evolution of a stochastic, well-stirred[2] system of coupled reactions: the Gillespie algorithm [42] and the StochSim algorithm [91].

[1] Obviously, a stochastic abstraction of a system with large molecule quantities and fast rates is still correct. However, in such cases, a continuous abstraction may be sufficiently correct, while having significant advantages in computability and understandability.

[2] The well-stirred assumption states that the solution is well mixed - that non-reactive collisions occur far more often than reactive ones, such that the fast dynamics of motion can be neglected and the system can be described by molecule numbers.

The Gillespie algorithm

The Gillespie approach [42] is based on the "reaction-as-object" abstraction. The system is defined by a set of elementary reactions[3], and their mesoscopic rate constants[4] *e.g.*

$$\begin{aligned} A + B &\xrightarrow{k_1} C \\ B + C &\xrightarrow{k_2} D \\ D + E &\xrightarrow{k_3} E + F \\ F &\xrightarrow{k_4} D + G \end{aligned}$$

The state of the system in the stochastic framework is defined by the number of molecules of each species $(A - G)$ and is changed discretely whenever one of the reactions occurs. The probability that a certain reaction μ will take place in the next instant of time $\mathrm{d}t$ is given by

$$a_\mu \mathrm{d}t \; + \; o(\mathrm{d}t)$$

where a_μ, the propensity function, is defined by k_μ times the number of individual potential copies of reaction μ (that is $k_\mu \times (\#A) \times (\#B)$ for a bimolecular elementary reaction and $k_\mu \times \frac{(\#A)\times(\#A-1)}{2}$ for a unimolecular reaction). Thus, the mesoscopic rate constants serve as the basis for *probability* transition rates between states in the system.

There are two ways to handle the stochastic framework. On the one hand, a Master Equation may be written, which is a set of coupled differential equations representing the system and having probabilities as variables. Solving the Master Equation yields the probability of each possible trajectory in the evolution of the system over time. This, however, is intractable for all but very small systems. For example, assuming each of the $A - G$ species may be present on 0-9 copies, even the 5-equation system above has 10^7 possible states.

The alternative and significantly easier approach is to *simulate* a *single* sample trajectory of a chemical process in the stochastic framework, by generating a sequence of state transitions and the times at which they occur. The Gillespie algorithm allows us to generate a trajectory such that reactions and times are selected *according to the correct probability distributions*. Thus, the probability of generating a given trajectory with the Gillespie simulation algorithm is exactly the probability that would come out of solving the Master Equation. Given the ability to generate a single trajectory with the correct probability, any parameter of interest may be estimated by generating many trajectories, calculating the value of the parameters for each trajectory, and observing the statistics of those calculated values.

The Gillespie algorithm (Gillespie's Direct Method, Figure 2.1) calculates explicitly *which* reaction occurs next and *when* it occurs. Both questions are answered probabilistically by specifying the probability density $P(\mu, \tau)$ that the next reaction is μ and it occurs at time τ

[3] A complicated chemical process can always be decomposed into a set of many elementary bimolecular $(A + B \rightarrow \cdots)$ or unimolecular $(A \rightarrow \cdots)$ reactions. Trimolecular reactions $(A + B + C \rightarrow \cdots)$ are very rate.

[4] Mesoscopic rate constants are related but distinct from the usual macroscopic rate constants. In particular, macroscopic rate constants do not depend on volume but concentrations of molecules do, whereas mesoscopic rate constants do depend on volume, but the number of molecules does not.

$$P(\mu,\tau)d\tau \quad = \quad a_\mu exp(-\tau\Sigma_j a_j)\mathrm{d}\tau$$

From $P(\mu,\tau)$ we obtain the two probability distributions necessary to answer our questions. First, the probability distribution for reactions is obtained by integrating $P(\mu,\tau)$ over all τ

$$\Pr(\text{Reaction} = \mu) \quad = \quad \frac{a_\mu}{\Sigma_j a_j}$$

Second, the probability distribution for times is obtained by summing $P(\mu,\tau)$ over all μ

$$P(\tau)d\tau \quad = \quad (\Sigma_j a_j)exp(-\tau\Sigma_j a_j)\mathrm{d}\tau$$

Using these two distributions the algorithm proceeds iteratively (Figure 2.1) by selecting at each iteration a reaction μ and a time step τ, changing the number of molecules to reflect the execution of the selected reaction μ and advancing the time according to the selected time step τ. The process is repeated until some threshold time or state is reached. While there are several additional variants to the algorithm, which differ in the specific way which they compute the probability and select the states, they are all essentially equivalent (as discussed in [41]), although they may differ in their performance.

The StochSim algorithm

The StochSim approach [91] employs a "molecule-as-object" abstraction. Similar to the Gillespie algorithm, it attempts to generate a single representative trajectory which describes a behavior of a chemical or biomolecular system over time. Both approaches make similar basic assumptions about a well-stirred system defined at the level of elementary reactions. However, while the Gillespie algorithm describes individual reactions (from which it selects the next one to occur) and ignores non-productive interactions, StochSim represents individual molecules, and handles both productive and non-productive interactions. Individual molecules are abstracted as individual software *objects*, interacting according to probabilities derived from pre-determined concentrations and rate constants.

While in the Gillespie framework the system is initialized by defining a set of reactions and molecular species (and their quantities), in the StochSim framework objects representing each type of molecular species are initialized and then large numbers of objects are created, each abstracting an individual molecule. Importantly, it is possible to abstract multi-state molecules that may be covalently modified in different ways, as multi-state objects that interact according to their "state". A number of dummy, or *pseudo-molecule* objects are also created at initialization, and are used to abstract unimolecular reactions: If a molecule object interacts with a pseudo-molecule, it may undergo an intra-object reaction, abstracting the corresponding unimolecular reaction. A lookup table defines all possible ways in which molecules may react in the system, their outcomes and their probabilities, the latter calculated by

$$P_{uni} = \frac{k_1 \times n \times (n + n_0) \times \Delta t}{n_0} \quad \text{unimolecular reaction}$$

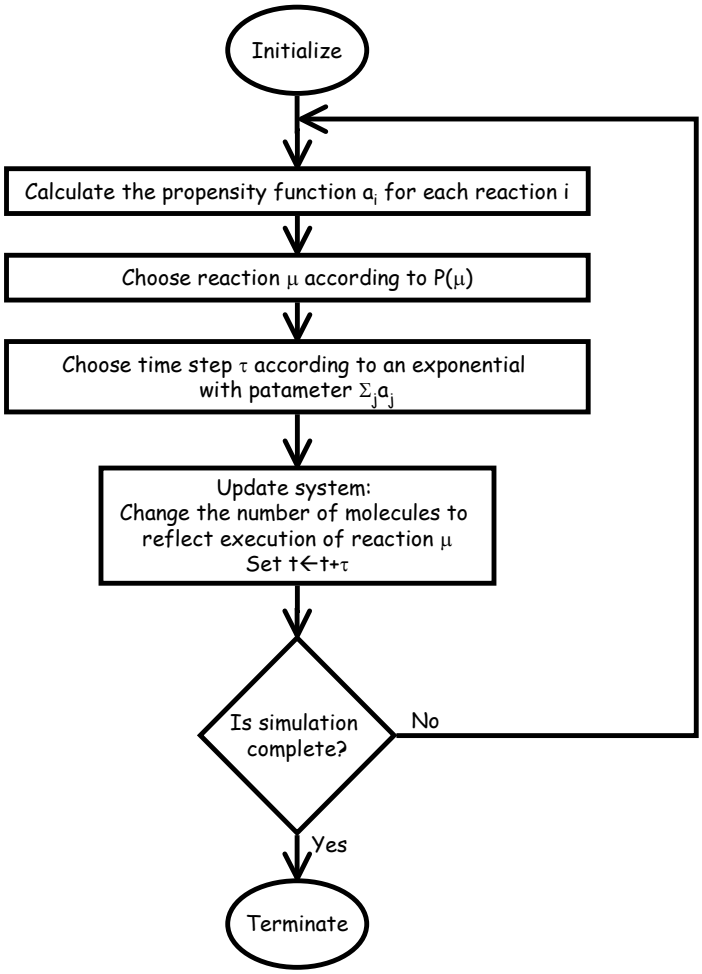

Figure 2.1: **The Gillespie Algorithm.** The algorithm proceeds iteratively by selecting at each iteration a reaction μ and a time step τ, changing the number of molecules to reflect the execution of the selected reaction μ and advancing the time according to the selected time step τ. The process is repeated until some threshold time or state is reached. Adapted from [41].

$$P_{bi} = \frac{k_2 \times n \times (n + n_0) \times \Delta t}{2 \times N_A \times V} \quad \text{bi} - \text{molecular reaction}$$

for unimolecular and bimolecular reactions, respectively; where n is the number of molecules in the system, n_0 is the number of pseudo-molecules in the system, k_1 is the macroscopic unimolecular rate constant, k_2 is the macroscopic bimolecular rate constant, Δt is the time slice duration (below), N_A is the Avogadro constant and V is the volume of the system (typically set to a unit value).

Unlike the variable length time steps determined by the Gillespie approach, StochSim time is quantized into a series of discrete independent time-slices. While the Gillespie approach selects a time step length together with a reaction, the StochSim time steps proceed at even pace, but successful reaction do not necessarily occur at each step. Rather, in each time-slice one molecule object (not a pseudo-molecule one) is selected at random. Then, another object, either a molecule or a pseudo-molecule, is selected at random as well. If two molecule objects are selected, the interaction abstract a bimolecular reaction. If one molecule and one pseudo-molecule object is selected the interaction abstracts a unimolecular reaction. If according to the lookup table the two objects can potentially interact, another random number is then generated to determine if a reaction indeed occurs. If the probability derived from the lookup table is greater than the random number, the objects do not interact, otherwise they react, and the system is updated accordingly. The next time slice then begins (regardless of whether reaction occured or was even possible at all or not!), and the procedure is iterated. The StochSim algorithm is summarized in Figure 2.2.

The stochastic framework and the π-calculus abstraction

Can Gillespie's "reaction-as-object" or StochSim's "molecule-as-object" be adapted into our π-calculus framework to provide the necessary stochastic extension? The answer is essentially yes, in both cases. The π-calculus abstraction maps reactions between molecules with complementary motifs to communication between processes on complementary communication channels. Thus, on the one hand, by handling channel objects as Gillespie handles reaction objects we can implement Gillespie's stochastic framework within the π-calculus. On the other hand, we can implement the StochSim framework by handling process instances as StochSim's molecular objects. Like the Gillespie and StochSim approaches, our representation is limited to *elementary reactions*: either bimolecular or unimolecular.

Each of the two approaches has benefits and drawbacks. The Gillespie algorithm has a rigorously proven relation to the Master Equation, and is well accepted and widely applied [41]. However, it is derived from the "cell as a collection of molecular species" framework, and thus lacks in handling the identity of individual molecules as objects with multiple states, a situation prevalent in biomolecular systems (see Chapter 1). The StochSim algorithm attempts to address exactly this limitation [91], by introducing individual molecule objects, including multi-state ones. However, the relation between the algorithm and the Master Equation is not proven, and the algorithm makes certain assumptions (a fixed total number of molecules) which may be inadequate in certain situations. Furthermore, the abstraction of multi-state objects is done *ad hoc* in a way which is interleaved with the computational component. We therefore decided to base the stochastic extension of our abstraction on the Gillespie algorithm. In doing so we carry many of the benefits of StochSim's "molecule as object" abstraction into Gillespie's "reaction as object" approach.

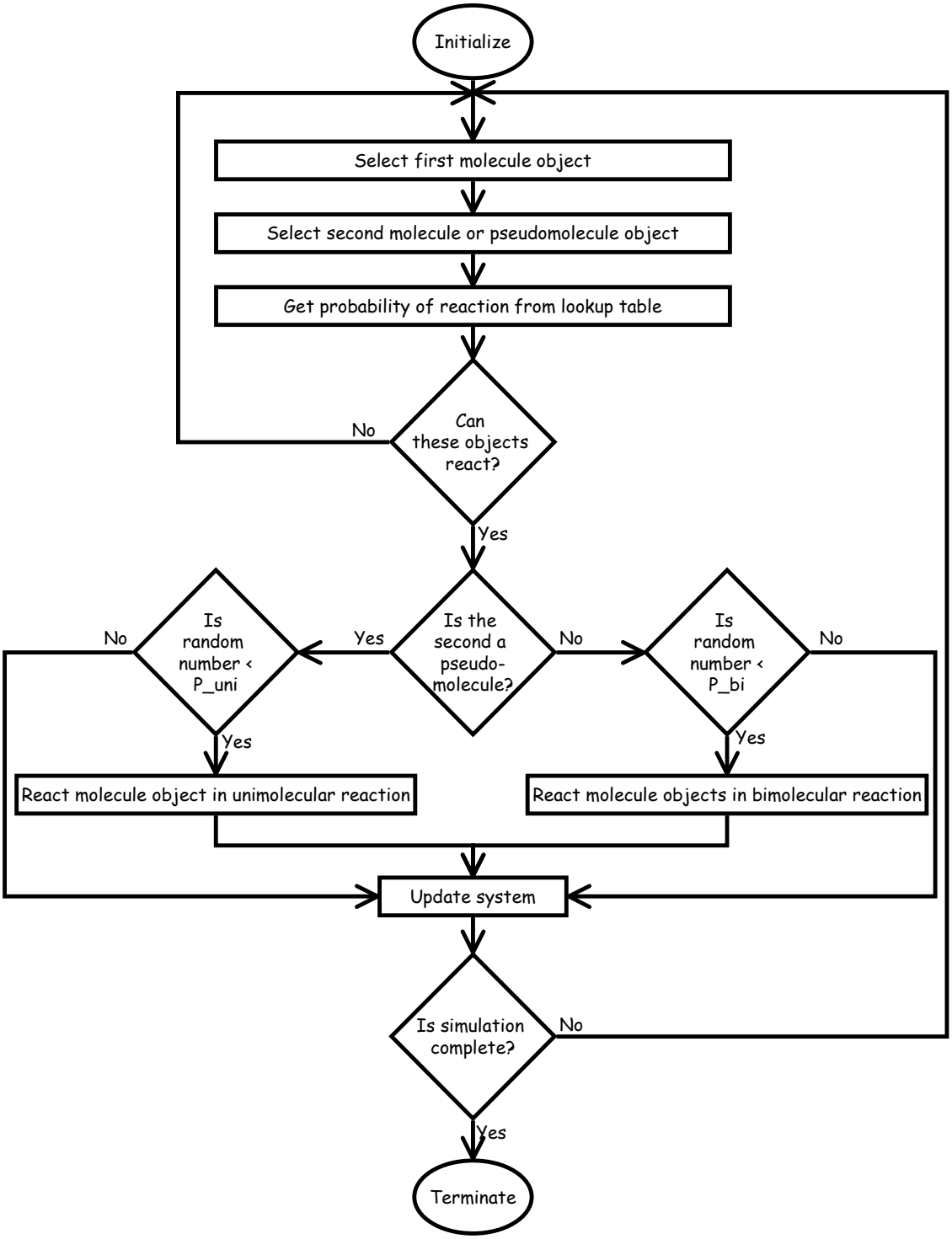

Figure 2.2: **The StochSim Algorithm.** The algorithm proceeds iteratively by randomly selecting a pair of molecular objects at each time slice and checking whether they can and will interact by comparing a random generated number to a pre-specified probability in a lookup table. If interaction is performed the system is updated. In any case, the system progresses to the next time slice. Adapted from [91].

2.2 Quantitative modeling in the biochemical stochastic π-calculus

In the quantitative extension of the π-calculus we handle channels in the same way that Gillespie handles reaction objects. To this end, we introduce several changes into the π-calculus domain and the π-calculus based abstraction (Table 2.1):

- **Channels with base rates as elementary reactions with mesoscopic rate constants.** Each channel is now associated with a *base rate*. The channel's base rate is identical to the mesoscopic rate constant of the corresponding elementary reaction.
- **Channels' actual rates as reactions' actual rates.** At each state in the π-calculus system we determine the actual rate of a channel based on that channel's base rate, and the number of input and output offers on the channel at that state. The actual channel rate is identical in calculation and value to the actual rate of the corresponding reaction.
- **π-calculus time steps as the time evolution of chemical system.** The discrete even time steps implicit to the unfolding of a semi-quantitative BioSpi 1.0 simulation[5] are now replaced by an explicit clock. The clock is advanced in uneven steps, according to the Gillespie algorithm, based on the actual channel rates. The resulting time evolution of the abstract π-calculus system corresponds to the time evolution of a single (representative) trajectory of the chemical system.
- **Stochastic selection of communication according to the probability of a reaction.** The non-deterministic way in which *enabled* communication actions[6] are chosen at each step in the unfolding of a π-calculus system is replaced by a stochastic selection process, based on the actual channel rates at each state of the system. The selection process is identical to the one specified in the Gillespie algorithm. The resulting state evolution of the π-calculus system corresponds to the state evolution of a (statistically representative) trajectory of the chemical system.

The other abstraction principles and syntax remain identical to the non-stochastic variant, and the extension is mostly insulated to the semantic interpretation of the abstract π-calculus.

2.2.1 Bi-molecular reactions

In a bi-molecular reaction two molecules of different types (*e.g.* A and B) interact to yield one or more products. We consider a simple example, in which two protein molecules, protein A and protein B interact and are reconstituted with different states, as active proteins. As we have seen in Chapter 1, we abstract each protein as a process (A, B, $ActiveA$, and $ActiveB$). We represent the ability of proteins A and B to interact as complementary communication actions on a specific channel ($reaction1$):

$$A \ ::= \ reaction1\ ?\ [\,]\ ,\ ActiveA\ .$$
$$B \ ::= \ reaction1\ !\ [\,]\ ,\ ActiveB\ .$$

[5] The semantics of the π-calculus and similar languages is originally purely concurrent, where events can happen in parallel. In practice, however, concurrency is implemented by an *interleaving semantics*, where systems unfold in a step-by-step manner, with a single event chosen at each step. Here we refer to this standard, well-accepted, approach, which was also taken in BioSpi 1.0.

[6] An enabled communication is one for which at least one input and one output action are offered simultaneously on the same channel by parallel processes.

Semi-quantitative, non-deterministic version	**Quantitative, stochastic version**	**Biochemical system**
Channel with no rate	Channel with base rate	(Bi-molecular) reaction with mesoscopic rate constant
	Actual channel rate depends on its base rate and the number of send and receive actions offered at each state of the system	Actual (bi-molecular) reaction rate depends on its mesoscopic rate constant and the numbers of the two reactant molecules at each state of the system
System state: number of processes	System state: number of processes	System state: Number of molecules
Sequential simulation of parallelism with even discrete time steps	Sequential simulation of parallelism with time steps of variable length determined by Gillespie's algorithm according to the actual channel rates at each state of the system	"True" parallelism. The time-state evolution of a single (probabilistically representative) trajectory of the chemical system
Non-deterministic selection of the next communication affected only by process quantities	Stochastic selection of the next communication according to Gillespie's algorithm affected by actual rates (*i.e.* process quantities and channel rates	

Table 2.1: **Quantitative (stochastic) extension of the π-calculus based abstraction.**

$$\begin{aligned} ActiveA &::= \cdots \\ ActiveB &::= \cdots \end{aligned}$$

The actual occurrence of the reaction is abstracted as a communication in the system, *e.g.*:

$$\begin{array}{c} reaction1\ ?\ [\,]\ ,\ ActiveA \quad | \quad reaction1\ !\ [\,]\ ,\ ActiveB \\ \rightarrow \\ ActiveA \quad | \quad ActiveB \end{array}$$

Adapting this to the stochastic framework requires several additions. First, the *reaction*1 channel is associated with a base rate. The base rate is introduced when the channel is defined[7] as *e.g.* $reaction1(k_1)$, where k_1 is a non-negative real number identical to the mesoscopic rate constant of the corresponding chemical reaction.

Second, we calculate the actual rate of the *reaction*1 channel at each step (state) in the evolution of the system. Following the rate calculation law for bimolecular reactions, the actual channel rate is calculated as:

$$\text{base rate} \times (\#\text{send requests}) \times (\#\text{receive requests})$$

where the number of send and receive requests represents the numbers of the two reactants (*e.g.* proteins A and B), available in the system at the given step. For example, if k_1 is 10 and we have 10 A and 15 B processes at a given step in the system, the actual rate of channel *reaction*1 is 1500.

Once we have abstracted reactions, rate constants, and actual reaction rates as channels, base rates, and actual channel rates, the remaining steps are essentially to select the next communication and time step according to the Gillespie algorithm. In Sections 2.3 and 2.4 we present a formal semantics and an implementation that ensure this. For the biologist reader suffice it to know that communications and time steps are selected in this manner.

As discussed in the Chapter 1, the products of biomolecular reactions include multi-molecular complexes, the molecules (modified or not), and *de novo* created biomolecules (as happens in transcription and translation events[8]). Importantly, the quantitative extension requires no change in the way we abstract the generation of products from reactants.

Symmetric bi-molecular reactions

A special case of bimolecular reactions is introduced when two identical molecules interact. Homodimerization is one example of such reactions in biomolecular systems. Remember that such reactions are represented by a choice construct, such that the same process simultaneously offers to send and to receive on the same channel (but only one offer may be selected). For example, we consider the

[7] Remember that all public channels are declared in the beginning of the program, while private ones are declared in the process that owns them. Accordingly, the base rate constant will appear in the public channel declaration (*e.g.* $-global(c1(10), c2(20), \cdots)$) or in the private channel declaration (*e.g.* $P ::= \mathsf{new}\ c1(20) \cdots$).

[8] Such a view of *de novo* generation of molecules is only partly correct, as long as we abstract away the monomer nucleotides and amino acids required for such polymerization events and these are not true elementary reactions.

abstraction of a protein C molecule which is able to interact with an identical C molecule (leading to their activation):

$$\begin{aligned} C &::= reaction2\ ?\ [\,]\ ,\ ActiveC\ ; \\ &\quad\ \ reaction2\ !\ [\,]\ ,\ ActiveC\ . \\ ActiveC &::= \cdots \end{aligned}$$

The occurrence of the reaction is abstracted as a communication in the system, in which one of the processes is arbitrarily picked as a sender and the other as a receiver:

$$\begin{aligned} reaction2\ ?\ [\,]\ ,\ ActiveC \quad &; \quad reaction2\ !\ [\,]\ ,\ ActiveC\ | \\ reaction2\ ?\ [\,]\ ,\ ActiveC \quad &; \quad reaction2\ !\ [\,]\ ,\ ActiveC \\ &\rightarrow \\ ActiveA \quad &| \quad ActiveB \end{aligned}$$

The stochastic extension for symmetric reactions is the same as that of any other bimolecular reaction, with one important exception. While the number of reactant combinations for an asymmetric bimolecular reaction is $\#A \times \#B$, when the two reactant are the same the number of distinct combinations is only $\frac{\#A \times \#(A-1)}{2}$. As a result, the actual channel rate (corresponding to the actual reaction rate) for channels participating in symmetric communication is calculated as

$$\text{base rate} \times \frac{(\#\text{send requests}) \times (\#\text{receive requests } - \ 1)}{2}$$

where the numbers of send and receive requests are by definition identical to each other and represent the number of reactant molecules available in the system at a given point.[9]

2.2.2 Unimolecular reactions

In a unimolecular reaction, a single molecule undergoes a chemical reaction to yield product(s). A typical example for a unimolecular reaction in biomolecular systems is the breaking down of a molecular complex. While bimolecular reactions have an immediate abstraction as bi-process communication, the abstraction of unimolecular ones poses a certain challenge. There are two ways to handle this problem. In the *timer approach*, the reacting molecule is represented by an "atomic entity", and an artificial counterpart (the "timer" process) is added to abstract the unimolecular reaction. In the *private channel approach* we represent the unimolecular reaction between two parts of the reacting molecule (or complex) using a private channel. We consider each of the approaches below and show they give rise to the same behavior.

[9]Such calculation ensures that if only a single process (able to participate in a symmetric communication) is present in the system, the actual reaction rate is zero, reflecting its inability to "interact with itself".

The timer approach

If we abstract the reacting molecule as a single process, we must supply it with an artificial communication counterpart. To this end, we introduce "dummy" timer processes, one copy per each type of unimolecular reaction in the system.[10]

The timer process offers the complementary communication to that offered by a process representing a unimolecular reactant. The timer process is reconstituted following reaction, and thus is present at each step in the evolution of the system.

For example, consider a system in which a complex protein molecule D is spontaneously broken into two proteins, E and F. As always, we abstract the protein molecules as processes (D, E and F). We abstract the unimolecular interaction of the D molecule as a communication on the $reaction3$ channel with a $Timer$ process offering the complementary communication.

$$\begin{aligned} D &::= reaction3\ ?\ [\,]\ ,\ (E \mid F)\ . \\ E &::= \cdots \\ F &::= \cdots \\ Timer(reaction) &::= reaction\ !\ [\,]\ ,\ Timer(reaction)\ . \end{aligned}$$

The parametric definition of the $Timer$ allows us to use it with different channel parameters for different unimolecular reactions. For example, for the unimolecular reaction in which protein D participates, we will create a $Timer(reaction3)$ process, allowing the following communication:

$$\begin{gathered} reaction3\ ?\ [\,]\ ,\ (E \mid F) \mid reaction3\ !\ [\,]\ ,\ Timer(reaction3) \\ \rightarrow \\ E \mid F \mid Timer(reaction3) \end{gathered}$$

Note, that regardless of the number of D processes in the system, there must always be only a *single* copy of the corresponding $Timer(reaction3)$ process. This is critical to derive the correct actual channel rate, as it ensures that the number of send offers *always* equals 1, and the actual channel rate (representing the actual unimolecular reaction rate) is

$$\text{base rate} \times (\#\ receive\ requests) \times 1$$

where the number of receive requests represents the number of reactant molecules available in the system at each step.

[10] A similar situation was encountered by the StochSim "molecule as object" abstraction [91], and the pseudo-molecules introduced there are conceptually similar to our timer processes. However, while under the Gillespie algorithm we can use a single timer process per reaction type, multiple pseudo-molecule objects without a specific identity are used in the StochSim algorithm.

The private channel approach

Alternatively, we may represent the internal specifics of the unimolecular reaction, assuming that they involve two independent, but specifically connected parts of a single molecule or molecular complex. For example, consider a two-molecule protein complex comprised of two bound protein molecules, G and H, which we abstract as two parallel processes ($BoundG$ and $BoundH$) sharing a private channel ($unbind$).

$$\begin{aligned} Complex &::= \mathsf{new}\ unbind\ .\ BoundH(unbind) \mid BoundG(unbind)\ . \\ BoundH(ch) &::= ch\ !\ [\,]\ ,\ H\ . \\ BoundG(ch) &::= ch\ ?\ [\,]\ ,\ G\ . \\ H &::= \cdots \\ G &::= \cdots \end{aligned}$$

The private $unbind$ channel is used for a communication abstracting the unimolecular unbinding reaction, and is specific to the particular pair of processes which own it.

$$\begin{array}{ccc} unbind\ !\ [\,]\ ,\ H & \mid & unbind\ ?\ [\,]\ ,\ G \\ & \rightarrow & \\ H & \mid & G \end{array}$$

In this case we do not need to include additional "dummy" processes. Rather, a unimolecular reaction is treated as a set of individual bi-molecular reactions on private channels: one reaction (private channel) per copy of the reacting molecule. Each of the channels (reactions) has an actual rate equal to the base rate (rate constant):

$$\text{base rate} \times 1 \times 1$$

However, as the Gillespie algorithm selects the time step and next reaction based on a *sum* of all actual rates (Section 2.1.2), the effect of these multiple communication channels with a single communication event is equivalent to that of a single channel on which all processes may communicate with a single timer. Hence, the two approaches are effectively equivalent.

2.2.3 Instantaneous reactions

Sometimes it is useful to treat a collection of computational events as an atomic instantaneous operation with respect to the stochastic simulation. For example, due to limitations in the expressive power of the calculus, a single biomolecular event is sometimes abstracted by more than a single communication. If the extra communications are executed instantaneously they do not change the stochastic time evolution of the system and the trajectory remains correct. Furthermore, certain biomolecular events, such as

Reaction type	**Channel type**	**Explanation**	**Rate calculation**
Asymmetric bimolecular reaction	Regular	One reactant is represented by a process offering to send on the reaction channel and the other by a process offering to receive on the reaction channel.	Base rate $\times$ #senders $\times$ #receivers
Symmetric bimolecular reaction	Symmetric	The reactants are represented by two instances of the same process, each offering a *choice* between a send and a receive on the reaction channel.	Base rate $\times \frac{\#\text{senders}\times(\#\text{receivers}-1)}{2}$ (By definition the number of senders and receivers is equal)
Unimolecular reaction	Regular (public)	The single reactant is represented by a process offering to receive on the public reaction channel. A single $Timer$ process offers the complementary communication.	Base rate $\times$ 1 $\times$ #receivers
	Regular (private)	The internally interacting parts of a single reactant are represented as two parallel processes communicating on a shared private reaction channel. Each potential "copy" of the reaction is represented by a separate private channel.	Base rate $\times$ 1 $\times$ 1 (An actual rate is calculated for each individual private reaction channel)
Instantaneous reaction	Instantaneous	Same as for bimolecular reaction	Infinite: Executed immediately when enabled, prior to any reaction on channels with finite rates, and without advancing the clock

Table 2.2: **Bimolecular, unimolecular and instantaneous reactions in the stochastic extension of the π-calculus**

conformation changes, may be significantly faster than most other events in the system. Including them into the stochastic framework could put an unnecessary burden on the simulation, while representing them as instantaneous communications would maintain the model's expressiveness.

To facilitate this, we introduce *instantaneous* communications, which are formally, communication with an infinite rate (defined as *e.g.* $ch(infinite)$). Instantaneous communications are executed immediately as they are enabled, prior to the execution of any other communication on a channel with a finite rate, and without advancing the clock (For more details see Section 2.4).

The abstraction of bimolecular, unimolecular and "instantaneous" reactions in the π-calculus is summarized in Table 2.2. The different ways in which the actual rate of channels is calculated for different types of reactions, and the different handling of communications on instantaneous channels, mean that a given channel name may only be used in one way throughout the unfolding of a stochastic π-calculus system. We therefore distinguish between three channel types: regular (used for both asymmetric bimolecular

and unimolecular reactions), symmetric (used for symmetric bimolecular reactions) and instantaneous. Once a channel name is used as one type, it may not be used as another.

2.3 The biochemical stochastic π-calculus: A formal presentation

While the syntax of the π-calculus expression remains largely intact in the stochastic extension, our interpretation of their behavior has changed significantly. In this section we formalize this change by presenting an alternative stochastic *semantics* to the π-calculus,[11] which accurately abstracts the time evolution of biomolecular systems. The biologist reader may skip this section and move to Section 2.4.

The only syntactic change is the association of a *base rate*, r, either a non-negative real number or the reserved word infinite, with the declared channel name. This change affects only channel declaration (the global or the new statement). For clarity purposes in the presentation of the semantics below, we will specify this base rate r in each of the communication prefixes, *e.g.* $(\pi, r), P$. We also assume all processes are in *head normal form*.[12]

As for semantics, unfolding of a system is now driven by a *race condition* [102], yielding a probabilistic model of computation, corresponding to the one delineated by Gillespie's algorithm. In each step all the enabled channels (for which both send and receive are offered) compete and only one wins, from which a communication pair is selected at random and executed. The chances to "win" reflect the actual channel rate.

Since actual reaction rates depend on the number of interacting processes, we define two auxiliary functions, $In, Out : 2^{\mathcal{P}} \times \mathcal{N} \to I\!N$, to inductively count the number of receive and send operations on a channel x enabled in a process:

$$
In_x(P) = \begin{cases}
0 & \text{if } P ::= \mathbf{0} \\
|\{(\pi_i, r_i) | i \in I \wedge sbj(\pi_i) = x?\}| & \text{if } P ::= \sum_{i \in I} (\pi_i, r_i), P_i \\
In_x(P_1) + In_x(P_2) & \text{if } P ::= P_1 \mid P_2 \\
In_x(Q) & \text{if } P ::= \mathsf{new}\ z\ .\ Q \text{ and } z \neq x \\
0 & \text{if } P ::= \mathsf{new}\ z\ .\ Q \text{ and } z = x
\end{cases}
$$

Out_x is similarly defined, by replacing any occurrence of In with Out and the condition $sbj(\pi_i) = x?$ with $sbj(\pi_i) = x!$.

Table 2.3 shows the reduction semantics of the biochemical stochastic π-calculus. We distinguish

[11] A stochastic semantics for the π-calculus was originally presented in [102]. However, the actual rate calculation and selection of communication and time steps were inadequate for the representation of molecular systems and were extensively modified as presented here.

[12] A process P is in *head normal form* if it is either the null process or

$$P \equiv \sum_i (\pi_i, r_i), P_i \quad \text{and} \quad \forall i \neq j\ .\ sbj(\pi_i) \neq sbj(\pi_j)$$

where $sbj(\pi)$ denotes the prefix π's output or input link (*e.g.* if π is a ! {b} (a ? {b}) then $sbj(\pi)$ is a! (a?)). Note, that in order to meet this condition we cannot allow more than one communication offer on a given channel in a given choice, except symmetric communication.

between asymmetric and symmetric communication according to the channel name. A subset of all the names, $\mathcal{H} \subseteq \mathcal{N}$, is used to identify channels used in symmetric communication. The first axiom (**StochAsym**) in Table 2.3 corresponds to asymmetric interactions between two different processes. The second axiom (**StochSym**) corresponds to symmetric interactions between two identical instances of one process species. The labels in both represent the basal reaction rates, when only a *single* copy is available of each reactant. The **StochPAR** reaction rule allows us to calculate the actual channel rate, considering multiple send and receive actions in the system. The rate is calculated by the three parameters r_b, r_0 and r_1, where r_b represents the channel's base rate, and r_0 and r_1 denote the quantities of processes offering send and receive actions on the channel, respectively, and are computed compositionally via In_x and Out_x while deducing transitions. The two additional rules (**StochRes** and **StochStruct**) correspond to the original π calculus ones and handle reduction under restriction and of structurally congruent expressions.

Asymmetric communication ($x \notin \mathcal{H}$) (**StochAsym**)
$\cdots + (x\,!\,\{z\}, r)\ ,\ Q \mid (x\,?\,\{y\}, r).P + \cdots \overset{x, r_b \cdot 1 \cdot 1}{\longrightarrow} Q|P\{z/y\}$

Symmetric communication ($x \in \mathcal{H}$) (**StochSym**)
$\cdots + (x\,!\,\{z\}, r)\ ,\ Q\ + (x\,?\,\{y\}, r)\ ,\ P \mid \cdots + (x\,!\,\{z\}, r)\ ,\ Q\ + (x\,?\,\{y\}, r)\ ,\ P \overset{x, 1/2 \cdot r_b \cdot 2 \cdot (2-1)}{\longrightarrow} Q|P\{z/y\}$

Communication under parallel composition (**StochPAR**)
if $P \overset{x, r_b \cdot r_0 \cdot r_1}{\longrightarrow} P'$ then $P|Q \overset{x, r_b \cdot r_0' \cdot r_1'}{\longrightarrow} P'|Q$, where $r_0' = r_0 + In_x(Q)$ and $r_1' = r_1 + Out_x(Q)$

Communication under restriction (**StochRes**)
if $P \overset{x, r_b \cdot r_0 \cdot r_1}{\longrightarrow} P'$ then $\mathsf{new}\ x.P \overset{x, r_b \cdot r_0 \cdot r_1}{\longrightarrow} \mathsf{new}\ x.P'$

Communication and structural congruence (**StochStruct**)
if $Q \equiv P$, $P \overset{x, r_b \cdot r_0 \cdot r_1}{\longrightarrow} P'$, and $P' \equiv Q'$ then $Q \overset{x, r_b \cdot r_0 \cdot r_1}{\longrightarrow} Q'$

Table 2.3: **Reduction semantics of the biochemical stochastic π-calculus.**

2.4 BioSpi 2.0: Stochastic simulation of π-calculus models

We implemented the Gillespie algorithm within the BioSpi system to serve as a stochastic simulation platform. In BioSpi 2.0, each channel object is also associated with a base rate and type (instantaneous, symmetric or asymmetric).[13] The channel base rate is supplied when the channel is defined. An "infinite" base rate serves to define an instantaneous channel type. Asymmetric and symmetric channel types are identified upon their first use.

The central BioSpi monitor maintains the simulation clock and is responsible for the correct execution of the Gillespie algorithm: the selection of the next communication and time step. The monitor operates in the following way (Figure 2.3): Requests to instantaneous channels are satisfied as soon as possible. Requests to channels which have a finite rate (> 0) are queued. Each time that a new event is required the central monitor and all channel objects with a finite, non-zero rate, jointly determine a communication event. In each iteration, each channel object determines its actual rate, according to its type, its basal rate, and the numbers of send and receive offers. The monitor then uses these actual rates to select the next reaction channel and time step, exactly according to the selection procedure of the Gillespie algorithm ([42], Section 2.1.2). The monitor then advances a "clock" counter according to the selected

[13] These are in addition to the send and receive lists and reference counts associated with each channel in both BioSpi 1.0 and 2.0.

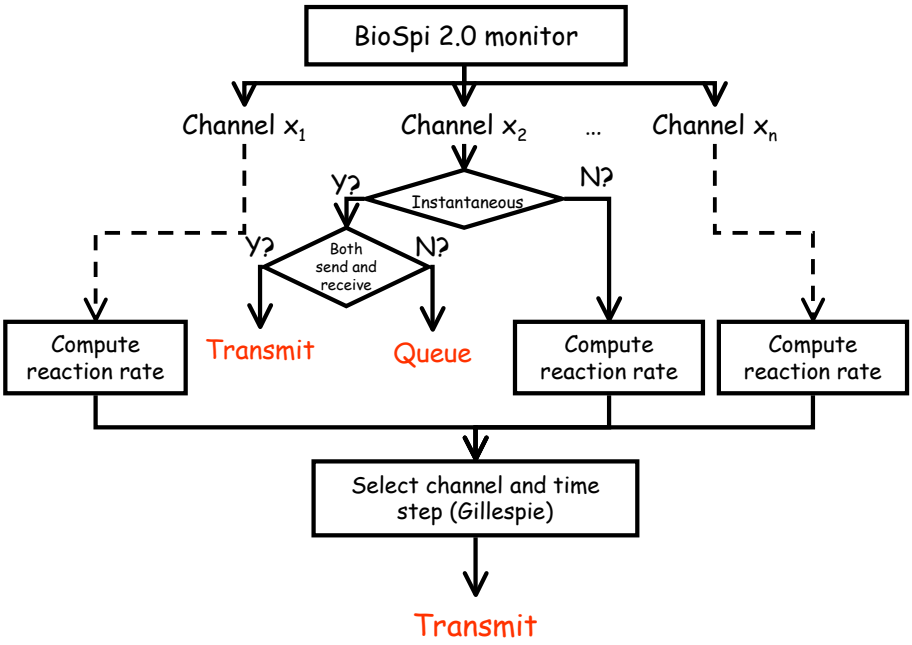

Figure 2.3: **BioSpi 2.0 implementation of the Gillespie algorithm.**

time step and allows the chosen channel to complete one transmission (send/receive pair), relaying the sent message to the receiver.[14] As in BioSpi 1.0, the completion of the send and receive requests is synchronized by the channel. In addition, other messages offered on this and other channels by the same two processes whose requests were completed, are withdrawn. The withdrawals are not synchronized, but they do precede continuation of their respective processes, and the next step of the clock.

In addition to the tracing and debugging tools available for BioSpi 1.0 simulations, BioSpi 2.0 allows us to maintain a full record of the time evolution of the system. The record specifies each communication in the system, the time at which it occured, the communicating processes, the channel on which the communication occured and the resulting processes. A sample portion of a raw record file is shown in Figure 2.4. The record file can be post-processed to yield the time evolution of the systems state (*i.e.* number of each process species at each time point) at any desired level of resolution. In the next section, we will see the utility of the record and trace tools in studying the stochastic behavior of specific systems.

2.5 Studying biomolecular systems with BioSpi 2.0

The changes in the stochastic extension of the π-calculus abstraction are mostly confined to the semantic interpretation of π-calculus programs, and to the pragmatic use of the abstraction (*e.g.* the representation of unimolecular reactions). As a result, the new guidelines for stochastic abstraction can be seamlessly composed on top of the existing semi-quantitative ones. Practically, this means that with the addition of appropriate channel rates, any semi-quantitative π-calculus abstraction can become stochastic.

[14]To be fully accurate, the selection of the send/receive pair should be done in a fully random way. In the current implementation, the pair is selected in a "top-of-the-queue" fashion, which is non-random. In most cases, there are only two process species communicating on a given channel, and this does not raise a problem, as all senders (or receivers) are equivalent. In some cases, however, this may be inaccurate. We are currently developing a fully random selection of a communication pair to be available in the next release.

```
-public(1):FREE_BD
-public(1):EXTRACELLULAR
+public(1):BOUND_BD
+public(1):EXTRACELLULAR.1
0.356028
-public(1):SHC_SH2_BS
-public(1):GRB2_SH2
+public(1):SHC_SH2_BS.2
+public(1):GRB2_SH2.1.2
0.363642
-public(1):INACTIVE_ERK_KINASE
-public(1):ERK_Nt_LOBE
+public(1):INACTIVE_ERK_KINASE
+public(1):ERK_Nt_LOBE.1
0.364756
```

Figure 2.4: **Sample BioSpi 2.0 record of the time evolution of a particular system.** The "reactant" process names (denoted with -) and the "product" process names (denoted with +) are shown, together with the simulation time. An alternative view can also show the channel name on which the communication takes place.

The incorporation of a stochastic semantics allows us to build more accurate abstractions of biomolecular systems. In this section we study two examples of these benefits. In the first, we use the distinction between channels with finite rates and instantaneous channels to build a detailed model of glycogen biosynthesis. We also extensively use the arithmetic extension of the calculus and its ability to abstract the composition of entities of growing complexity from simple building blocks.

In the second example, we study a simple model of the circadian clock using its π-calculus abstraction. In this case a semi-quantitative model is incorrect as it does not capture the oscillatory behavior of the biological system. The stochastic model is not only correct, but also demonstrates how the π-calculus abstraction is both relevant and understandable as it prompts us to study this system from new perspectives.

2.5.1 A compositional abstraction of glycogen biosynthesis

Glycogen biosynthesis

Polymerization is a key event in biomolecular systems. Polymerization reactions are responsible for the formation of DNA from nucleic acids, RNA from ribonucleic acids, proteins from amino acids, and polysaccharides from simple sugars (or monosaccharides).

In contrast to proteins and nucleic acids, polysaccharides can form branched as well as linear polymers, as the glycosidic bonds that link the monomer sugars can be made to any of the hydroxyl groups of a monosaccharide. As a result, a single monomer may be bound to more than two counterparts, and form a branch point. In general, most polysaccharides are linear and those that branch do so in only a few well defined ways.

Glycogen is a branched polysaccharide composed of glucose monomers [147]. Glycogen is biosynthesized in a combination of two alternative enzymatic reactions (Figure 2.5). The first is an *elongation* reaction, in which an $\alpha(1 \rightarrow 4)$ glycosidic bond is formed between the C_4 of the polymerized glucose at

the end of a growing polymer and the C_1 of an "incoming" UDP-glucose monomer.[15] This reaction is catalyzed by the glycogen synthase enzyme. The second is a *branching* reaction, in which an $\alpha(1 \rightarrow 6)$ glycosidic bond is formed between the C_6 of a polymerized glucose within the glycogen polymer and the C_1 of an "incoming" glycogen oligomer. The oligomer is branched "off" an existing polymer chain, as depicted in Figure 2.5C. This reaction is catalyzed by the glycogen branching enzyme ($1,4 \rightarrow 1,6$ transglycosylase). In addition, the glycogen polymer is *initiated* by a third enzyme, called glycogenin. This enzyme forms an initial 7-residue primer, which we term a *seed*.

The biosynthesis of glycogen follows specific "rules". First, the specificity of glycogen synthase ensures that elongation is allowed only from growing 4' ends. After branching, there may be more than one such end, and elongation may proceed independently at each of these ends.

The branching "rules" are more complex. A branch is specifically created by transferring a 7-residue segment from the end of one chain to the $C_6 - OH$ group of a polymerized glucose residue on the same or another glycogen chain. Each transferred segment must come from a chain of at least 11 residues, and the new branch point must be at least 4 residues away from any other branch point. It is believed that the "rules" have been optimized in evolution for efficient storage and mobilization of glucose [85].

The balance between elongation and branching determines the architecture of the glycogen molecule: chain length and extent of branching. This architecture determines the compactness of glucose storage and its availability and is thus critical for the functional role of glycogen as an energy reserve.

Building the abstraction in the π-calculus

The first step in building an abstraction is identifying and defining the basic entities in the real world domain. To this end, we now break down the above description into several components.

First, each glycogen strand is directional (Figure 2.6). We term one end, with a C_1 carbon, that cannot grow, as the *root side*. We term the other end, with a C_4 carbon, that can grow, as the *leaf side*.

Second, we distinguish three potential positions for a polymerized residue (Figure 2.6). First, it may reside on a *straight segment*, where no branch points exist to the leaf side. Second, it may be part of a *branched segment*, that has a branch point to the leaf side (*i.e.* it resides between the branch point and the root or between two branch points). Third, the residue may be the *branch point* itself.

More specifically, the exact position of the residue determines the types of reactions in which it may participate. We distinguish between three types of events in which a residue may participate. A *polymerization enabled* residue may participate in an elongation reaction (as the C_4 donor). A *cleavage enabled* residue may be cleaved in a branching reaction and linked into another glycogen branch. A *branch enabled* residue may serve as a branch point onto which a new glycogen oligomer is added.

According to the above elongation and branching rules, we see that residues on straight segments may be either *polymerization enabled* (the leaf only), *cleavage and branch enabled* (at distance 7 exactly from the leaf and at distance 4 at least from the closest branch point/root), *branch enabled but not cleavage enabled* (at distance other than 7 from the leaf and at distance 4 at least from the closest branch point or root), or *disabled* (not the leaf and at distance less than 4 from the closest branch point or root).

Residues on a branched segment are more limited in their interaction possibilities. As they cannot be elongated (no free end) or cleaved (no straight segment) they are either *branch enabled but not cleavage enabled* (at distance 4 at least from the flanking branch points/root on both sides) or *disabled* (at distance

[15] The glucose $\rightarrow$ UDP-glucose reaction is catalyzed by a separate enzyme which we will not discuss here. We will use the terms glucose and UDP-glucose interchangeably from here on.

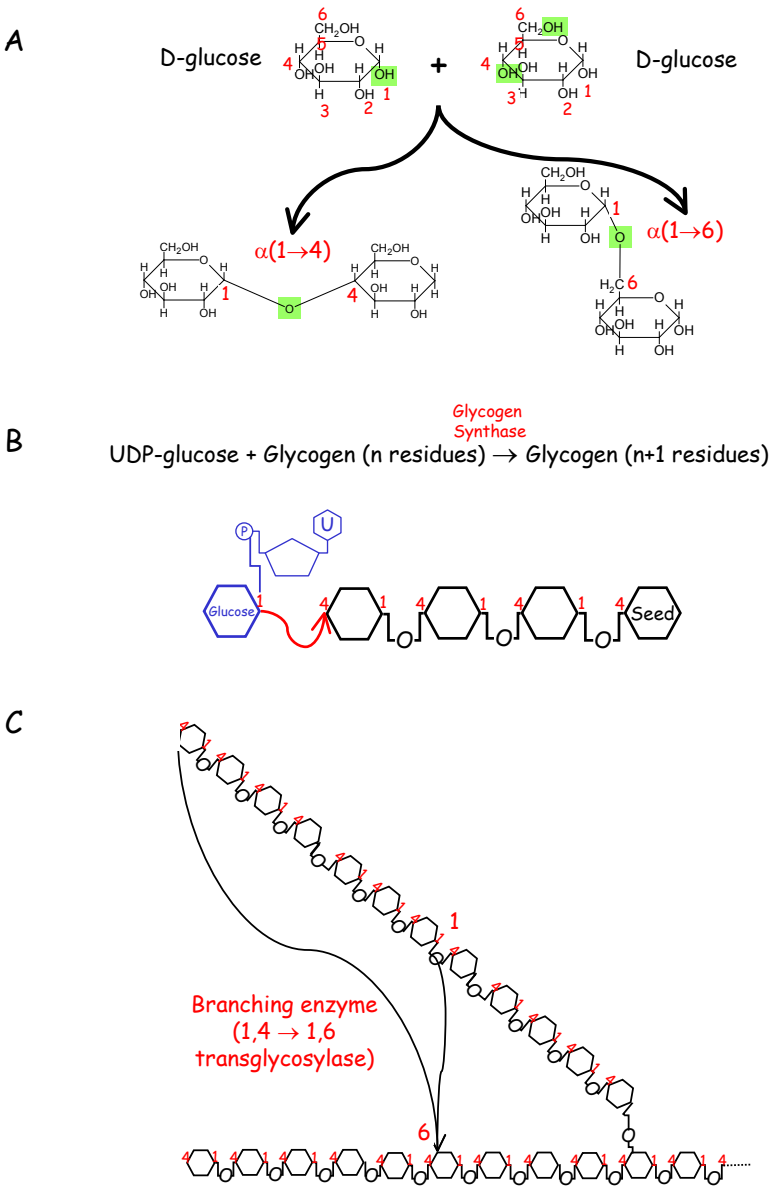

Figure 2.5: **Glycogen biosynthesis reactions.** A. The formation of $1 \rightarrow 4$ (left) and $1 \rightarrow 6$ (right) glycosidic bonds. B. In an elongation reaction, catalyzed by glycogen synthase, the length of a linear glycogen polymer is incremented by one monomer glucose unit. C. In a branching reaction, catalyzed by the branching enzyme, a glycogen oligomer is "chopped off" a liner polymer and added as a new branch.

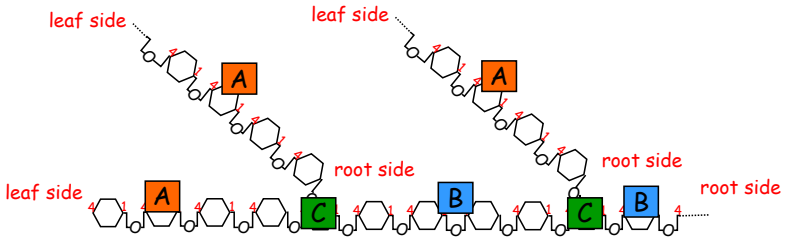

Figure 2.6: **The architecture of a glycogen molecule: definitions.** The C_1 and C_4 end of a glycogen strand defined as "Root" and "Leaf" respectively. A polymerized residue may assume one of three potential positions: on a straight segment (A), on a branched segment (B) or at a branch point (C).

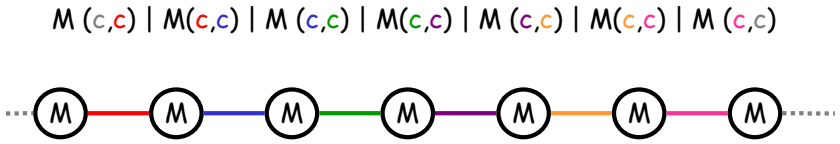

Figure 2.7: **A polymer abstracted as a chain of processes linked by private channels.** Circles represent processes. Connecting colored lines represent individual private channels.

less than 4 from at least one of its flanking branch points/root). Obviously, the branch point itself cannot participate in any further reactions. The different types of residues are listed in Table 2.4.

We are now ready to abstract the system in the stochastic π-calculus. First, each glucose residue is abstracted as a process. Different process states represent the different positions of a glucose residue (Table 2.4, rightmost column). Processes also represent the two types of enzyme molecules (*Glycogen_Synthase* and *Branching_Enzyme*) and the 7-residue seed (*Seed_Glucose* for the "free" seed and *Root_Glucose* for the polymerized seed). The formation of bonds is abstracted as private channels, with one private channel shared between each pair of immediate neighbors. Thus, each process representing a fully polymerized residue has two private channels: one linking it with its root-side neighbor and the other with its leaf-side neighbor. The processes representing the root and leaf processes share only a one private channel with a neighbor (Figure 2.7).

The exact position of a residue is critical to determine its behavior. Similarly, the exact "position" of the corresponding process is critical to determine its state. We abstract a residue's position in the glycogen polymer by the *values* of three *position variables* that are associated with each individual residue process (Figure 2.8):

- **Leaf counter.** The **LC** variable measures the distance of the residue from the leaf. It is initialized to zero, incremented (+1) upon extension, and decreased (-7) upon cleavage in the remaining segment residues.
- **Root counter.** The **RC** variable measures the distance of the residue from root or from the flanking branch point on the root side. It is initialized to the value of the RC variable of the neighbor residue on the root side + 1, updated upon cleavage in the cleaved segment residues to the RC of the new root side neighbor + 1, and updated upon insertion in the segment on the leaf side of the new branch point to RC - (RC of new branch point).

General position	**Residue capabilities**	**Detailed position**	**Process name**
Non-polymerized	Polymerization enabled	Free UDP-glucose (1' end)	*UDP_Glucose*
Straight segment	Polymerization enabled	Leaf (4' end)	*Leaf_Glucose*
	Cleavage and branch enabled	At distance 7 exactly from the leaf and at distance 4 at least from closest branch point or root	*BCE_Glucose*
	Branch but not cleavage enabled	At distance other than 7 from leaf and at distance 4 at least from closest branch point or root	*BNCE_Glucose*
	Disabled	Not the lead and at distance less than 4 from the closest branch point or root	*Disabled_Glucose*
Branched segment	Branch but not cleavage enabled	At distance at least 4 from both flanking branch points/root	*BNCE_Glucose*
	Disabled	At distance less than 4 from at least one of its flanking branch points/ root	*Disabled_Branched_Glucose*
Branch point	Disabled	At the branch point (C_6 donor)	*Branch_Point*

Table 2.4: **Different types of residues defined by their reaction capabilities and abstracted as π-calculus processes.**

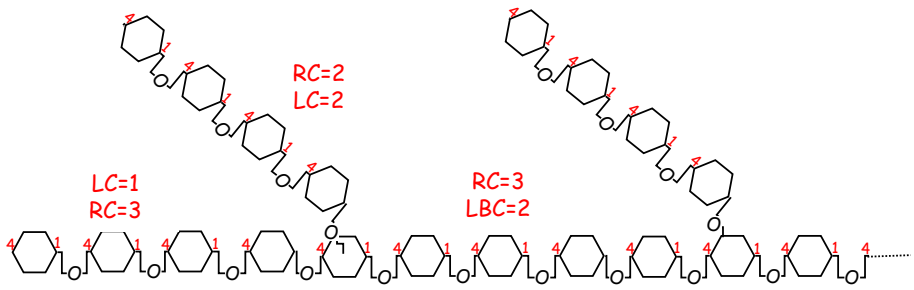

Figure 2.8: **The leaf (LC), root (RC), and leaf-branch (LBC) counters: An example.** The values of the LC, RC, and LBC counters for several residues are shown.

- **Leaf-branch counter.** The **LBC** variable measures the distance of the residue from the flanking branch point on the leaf side. It is initialized to zero and updated upon insertion in the segment on the root-side of the new branch point to the LBC of the leaf side neighbor+1.

As counters abstract the position of each residue in the polymer, and as the residue's position determines its state, we use the counters to define the corresponding process state. We use a choice construct, in which the counter values are checked according to the different rules:

$$
\begin{aligned}
&\{LBC \ = \ 0\} \ , && \%\text{Straight segment} \\
&&& (\ \{LC \ = \ 0\} \ , \ Leaf\_Glucose \ ; \\
&&& \{LC = 7 \ , \ RC >= 4\} \ , \ BCE\_Glucose \ ; \\
&&& \{LC > 0 \ , \ LC = / = 7 \ , \ RC >= 4\} \ , \ BNCE\_Glucose \ ; \\
&&& \{LC > 0 \ , \ RC < 4\} \ , \ Disabled\_Glucose \ ) \ ; \\
&\{LBC > 0\} \ , && \%\text{Branched segment} \\
&&& (\ \{RC >= 4 \ , \ LBC >= 4\}, BNCE_Glucose; \\
&&& \{RC < 4\} \ , \ Disabled\_Branched\_Glucose; \\
&&& \{LBC < 4\} \ , \ Disabled\_Branched\_Glucose \ ) \ .
\end{aligned}
$$

The choice construct is examined each time that the counter value changes.

The position of a residue may change *directly*, when the residue participates in an interaction (polymerization or cleave/branch) or *indirectly*, when the residue is part of a segment which participates in those reactions. For example, upon branching, all residues to the root-side of the branch point have changed their relative position and their capabilities (*e.g.* they can no longer serve as cleavage points). In the abstract representation, in order for the counters in residue processes to correctly represent the position of the corresponding residues in the "real world" polymer, they must be constantly updated. However, as only two processes participate in the immediate interaction, we must incorporate an *update propagate mechanism* into our abstract model.

Counter updating will be performed by communication on the private channels linking the process residues. Importantly, all such private channels will be instantaneous, while all the communications

representing *actual* reactions (elongation, cleavage and branching) will be carried out on channels with a real finite rate. As a result, counter updating will be done in zero simulation time, and will not interfere with the kinetics of biochemical reactions, while allowing us to maintain a detailed view of the glucose monomer's position within the glycogen polymer.

The full abstraction of the glycogen biosynthetic system in the π-calculus is given in Figure 2.9. The enzymatic elongation reaction is represented by the *glycogen* and *udp_glucose* channels, and the cleave and branch reaction by the *branch* and *cleave* channels.[16] Each $UDP\_Glucose$ process is defined with two private channels (*to_root* and *to_leaf*) that will be subsequently used as specific links to residues in the resulting polymer. For simplicity, all the "reaction" channels are given an equal base rate of 1 (we will revisit this decision below), while all private channels are instantaneous with an infinite base rate.

```
global(glycogen(1), udp_glucose(1), dummy(1), branch(1), cleave(1)).
baserate(infinite).

Seed_Glucose(RC,LC,LBC)::=
   glycogen ? {to_leaf} , to_leaf ! {RC,LBC} , Root_Glucose(to_leaf,RC,LC,LBC) .
Root_Glucose(to_leaf,RC,LC,LBC)::=
   to_leaf ? {LC,LBC} , {LC++} , Root_Glucose(to_leaf,RC,LC,LBC) .
UDP_Glucose(LC,LBC)+(to_root,to_leaf)::=
   udp_glucose ! {to_root} , to_root ? {RC,LBC} , {RC++} ,
                          to_root ! {LC,LBC} , Glucose(to_root,to_leaf,RC,LC,LBC) .
Glucose(to_root, to_leaf, RC, LC, LBC)::=
{LC>=0},
<< {LBC = 0} , << {LC = 0} ,  Leaf_Glucose ;
            {LC = 7 , RC >= 4} , BCE_Glucose ;
  {LC > 0 , LC =\= 7 , RC >= 4} , BNCE_Glucose ;
  {LC > 0 , RC < 4} , Disabled_Glucose >> ;
   {LBC > 0} , << {RC >= 4 , LBC >=4} , BNCE_Glucose ;
          {RC < 4} , Disabled_Branched_Glucose ;
  {LBC < 4} , Disabled_Branched_Glucose >> .
Leaf_Glucose::=
   glycogen ? {to_leaf} , to_leaf ! {RC,LBC} , to_leaf ? {LC,LBC} , {LC++} ,
                            to_root ! {LC,LBC} , Glucose(to_root,to_leaf,RC,LC,LBC);
   to_root ? {RC,_} , <<  {RC >=0} , {RC++} , Glucose ;
  {RC < 0} , Disabled_Leaf_Glucose >> .
Disabled_Leaf_Glucose::=
   to_root ? {RC,_} , {RC++} , Glucose .
BNCE_Glucose::=
   to_leaf ? {LC,LBC} , {LC++} , << {LBC = 0} , to_root ! {LC,LBC} , Glucose ;
               {LBC > 0} , {LBC++} ,  to_root ! {LC,LBC} , Glucose >> ;
   to_root ? {RC,_} , << {RC >=0} , {RC++} , to_leaf ! {RC,LBC} , Glucose ;
 {RC < 0} , to_leaf ! {RC, LBC} , Disabled_Glucose >> ;
   branch ? {to_branch} , Branch_Synch1(to_branch,RC,LC,LBC) .
```

Figure 2.9: π **calculus code for the glycogen system.** The entire system (64 lines, 15 processes) is shown with the exception of initialization of position counters. The code is continued on the next page and available in [106].

[16] Remember that in the stochastic framework complex reactions are broken down to elementary steps and their abstraction requires several channels. The enzymatic reaction model here is highly simplified. A more accurate model was presented in the previous chapter.

```
Branch_Synch1(to_branch,RC,LC,LBC)+(RC1,LBC1)::=
  {RC1=0} | {LBC1=1} |
              << to_branch ! {RC1,LBC} ,
                    << to_leaf ! {RC1,LBC} ,
to_root ! {LC,LBC1} , Branch_Point(to_root,to_branch,to_leaf) >> >>  .
Disabled_Glucose::=
   to_leaf ? {LC,LBC} , {LC++} ,
         << {LBC = 0} , to_root ! {LC,LBC} , Glucose ;
    {LBC > 0} , {LBC++} ,  to_root ! {LC,LBC} , Glucose >> ;
   to_root ? {RC,_} , {RC++} , to_leaf ! {RC,LBC} , Glucose .


BCE_Glucose+(new_to_root,RC1,LC1,LBC1)::=
   << to_leaf ? {LC,LBC} , {LC++} , << {LBC = 0} , to_root ! {LC,LBC} , Glucose ;
                {LBC > 0} , {LBC++} ,  to_root ! {LC,LBC} , Glucose >> ;
      to_root ? {RC,_} , << {RC >=0} , {RC++} , to_leaf ! {RC,LBC} , Glucose ;
    {RC < 0} , to_leaf ! {RC,LBC} , Disabled_Glucose >> ;
      branch ? {to_branch} , Branch_Synch(to_branch,RC,LC,LBC) ;
      cleave ! {new_to_root} , {LC1 = -1} | {RC1 = -1} | Cleave_Synch(to_leaf) .

Cleave_Synch(to_leaf)::=
   to_root ! {LC1,LBC} , to_leaf ! {RC1, LBC} , new_to_root ? {RC,_} ,
{RC++} , to_leaf ! {RC,LBC} , Glucose(new_to_root,to_leaf, RC, LC, LBC) >> .

Branch_Synch(to_branch,RC,LC,LBC)+(RC1,LBC1)::=
   {RC1=0} | {LBC1=1} |
               << to_branch ! {RC1,LBC} ,
             << to_leaf ! {RC1,LBC} ,
 to_root ! {LC,LBC1} , Branch_Point(to_root,to_branch,to_leaf) >> >> .

Disabled_Branched_Glucose::=
   to_leaf ? {LC,LBC} , {LC++} ,
        << {LBC = 0} , to_root ! {LC,LBC} , Glucose ;
           {LBC > 0} , {LBC++} ,  to_root ! {LC,LBC} , Glucose >> ;
   to_root ? {RC,_} , {RC++} , to_leaf ! {RC,LBC} , Glucose >> .

Branch_Point(to_root,to_branch,to_leaf)::=
   to_root ? {_,_} , self ;
   to_branch ? {_,_} , self ;
   to_leaf ? {_,_} , self .

Glycogen_Synthase::=
   udp_glucose ? {to_root} , glycogen ! {to_root} , Glycogen_Synthase  .

Branching_Enzyme::=
   cleave ? {to_branch} , branch ! {to_branch} , Branching_Enzyme .
```

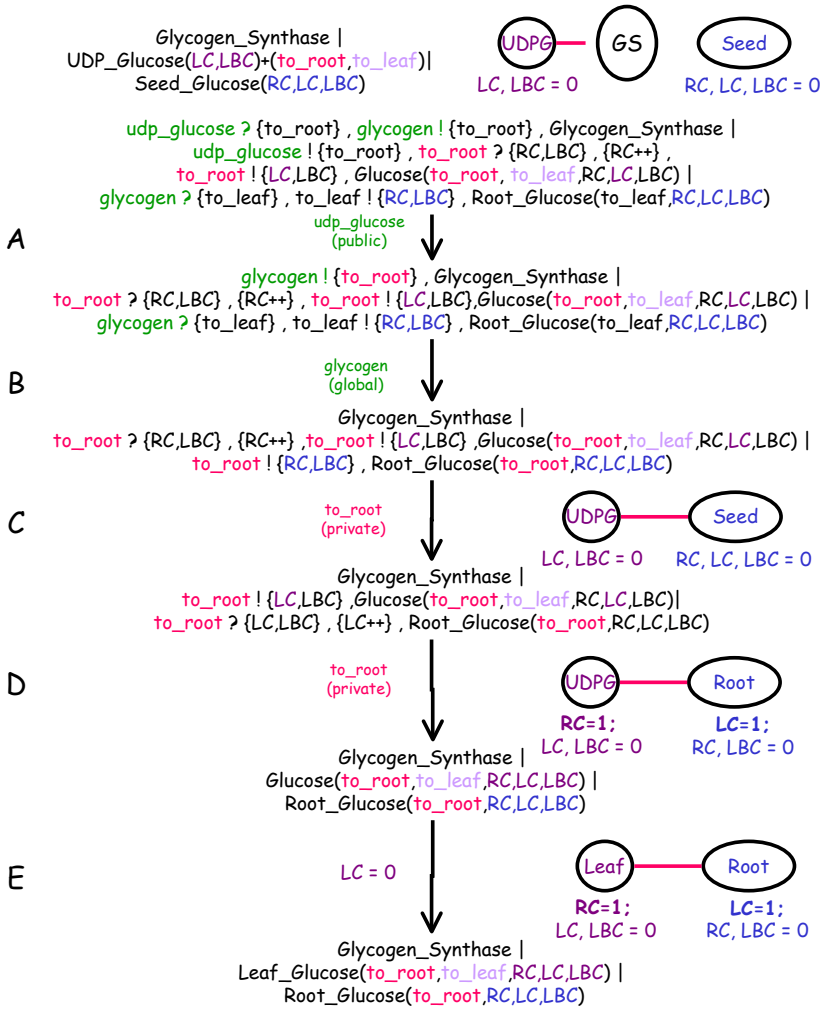

Figure 2.10: **Initiation as a multi-step communication between** *UDP_Glucose*, *Glycogen_Synthase* **and** *Seed_Glucose*. A. Communication between *UDP_Glucose* and *Glycogen_Synthase*. B. Communication between *Glycogen_Synthase* and *Seed_Glucose*. C,D. Counter update using a shared private channel. E. *UDP_Glucose* changed to *Leaf_Glucose* based on counter values.

Abstraction of initiation

In the initiation step (after a seed oligomer is already produced by glycogenin), glycogen synthase catalyzes the formation of a glycosidic bond between a UDP-glucose and the 4' leaf end of the seed oligomer. In the abstract π-calculus model we represent the initiation as two consecutive communication steps. In the first step, the *to_root* private channel of *UDP_Glucose* is sent to *Glycogen_Synthase* on the *udp_glucose* channel. In the second step, this private channel name is further relayed in a communication from *Glycogen_Synthase* to *Seed_Glucose* on *glycogen* (Figure 2.10A and B).

Following the two communications, *UDP_Glucose* and *Seed_Glucose* now share a private (instantaneous) channel. In the next two steps, the position counters of each of the processes are updated to reflect their relative position using this shared channel(Figure 2.10C and D). First, the *RC* counter is sent on the private channel from *Seed_Glucose* to *UDP_Glucose*, and is used to update *UDP_Glucose*'s *RC* value. This represents the change in the position of this now polymerized residue relative to its newly acquired root. Then, the *LC* and *LBC* counters are sent on the private channel from *UDP_Glucose*

to the now *Root_Glucose*, and are used to update the respective counters in *Root_Glucose*. This represents the concomitant change in the position of the root oligomer relative to its newly acquired leaf. Finally ((Figure 2.10E), the new counter values are evaluated in $UDP\_Glucose$, changing it to its new $Leaf\_Glucose$ state.

Abstraction of elongation

Elongation is similar to initiation and involved the same processes and communications, with the exception of *Seed_Glucose* which is "replaced" by $Leaf\_Glucose$. Two consecutive communications (first on *udp_glucose* and then on *glycogen*) serve to relay a private channel from $UDP\_Glucose$ to $Leaf\_Glucose$ via *Glycogen_Synthase* (Figure 2.11 A,B). This private channel is then used to initiate an update of the positional counters of *all* the processes along the growing chain (each time using the relevant private channel, Figure 2.11 C-E). Finally, the updated counters determine the new state of each of the linked processes (Figure 2.11 C-E).

Abstraction of branching

Consider a relatively simple branching scenario in which a 12-residue chain is cleaved at the fifth residue. The 4-residue "rooted" part is then elongated by two more residues, followed by linking of the 8-residue "clipped" part on its fifth residue (Figure 2.12).

A process chain of size 12 is shown in Figure 2.12, where process state is determined according to its relative position. Each pair of processes is linked by a unique private channel, established during the elongation step, as described above. Cleavage is abstracted as a communication on *cleave* between a *Branching_Enzyme* and a $BCE\_Glucose$ (Figure 2.13A). Due to the short chain length in this example, there is only a single $BCE\_Glucose$. In the communication, a private channel (*new_to_root*) is sent from $BCE\_Glucose$ to *Branching_Enzyme*. Following the communication, the LC and RC counters of $BCE\_Glucose$ are set to special values (-1), followed by a series of communications on the chains of private channels to update the relevant positional information. First, the "cleave event" is propagated to the root side of the cleave point (Figure 2.13B). As a result, the immediate root-side neighbor of the cleaved residue process will become a $Leaf\_Glucose$. The other root-side processes will be updated accordingly with the communication propagated up to the *Root_Glucose*.

In parallel, the cleavage event is propagated to the leaf side of the cleavage point (Figure 2.13C). Since cleavage and branching may be separated by additional steps (*e.g.* in the example scenario we have additional elongation of the "rooted branch"), the "clipped" branch is "frozen" until it is be linked to a new position. The "freeze" is based on propagating the special RC counter value (-1) throughout the clipped chain. The special value serves to set each process to a *Disabled_Glucose*, and will change only upon updating the positional counters (following branch linking, see below). On the other hand, the "rooted" chain may participate like any other chain in elongation (*e.g.* the two elongation steps in our example scenario).

We now examine how to abstract the linking of a cleaved ("frozen") branch to form a branch point. Assuming two additional elongation steps, we start with a system with two process chains, one representing a "rooted" branch of size 6 and the other representing a cleaved branch of size 8 (Figure 2.12C). Branching is represented by a communication between *Branching_Enzyme* and $BNCE\_Glucose$ on *branch*, in which the private *new_to_root* channel, originating in the "cleaved" $BCE\_Glucose$ is relayed

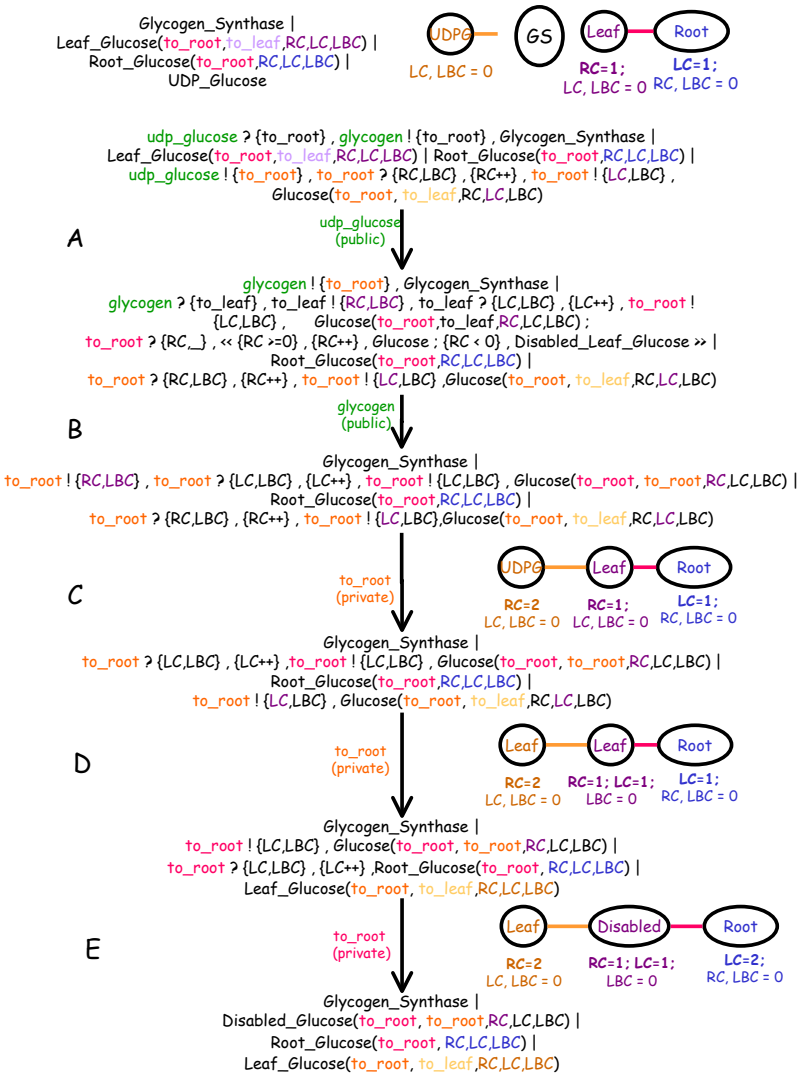

Figure 2.11: **Elongation from a two- to three- residue glycogen chain as a multi-step communication between** *UDP_Glucose*, *Glycogen_Synthase* **and** *Leaf_Glucose*. A. Communication between *UDP_Glucose* and *Glycogen_Synthase*. B. Communication between *Glycogen_Synthase* and *Leaf_Glucose*. C-E. Counter updates using a series of shared private channels, representing the 3-residue chain. E. Process state change based on counter values represents their updated capabilities following the elongation step.

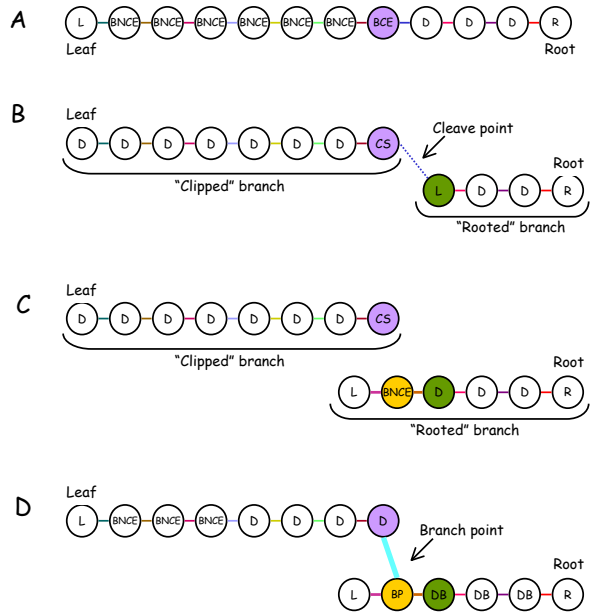

Figure 2.12: **A simple branch and cleave scenario.** A. A chain of length 12. B. Following cleavage, we obtain two chains: a rooted one of length 4 and a clipped one of length 8. C. The rooted chain is elongated by two more residues. D. Linking the clipped chain to residue 5 of the rooted chain. Corresponding process names are shown (R - *Root_Glucose*, L - *Leaf_Glucose*, D - *Disabled_Glucose*, BP - *Branch_Point*, DB - *Disabled_Branched_Glucose*, CS - *Cleave_Synch*)

to *BNCE_Glucose* (Figure 2.14A). This results in a private link between the two *Glucose* processes (Figure 2.14B). A series of communications on private channels propagate the change as an update in positional counters from the cleave point (to the leaf direction, resulting in "unfreezing" of the cleaved branch) and from the branch point (to both the root and leaf direction) (Figure 2.14C-E).

Studying glycogen metabolism with BioSpi 2.0

The detailed glycogen biosynthesis model, based on a "monomer-as-process" abstraction, allows us to monitor the exact architecture of glycogen molecules through the architecture of the process "polymer". The tracing tools of BioSpi 2.0 provide us with the information (process names, counter values and private channel identity) necessary to reconstruct this architecture.

A simple example, based on a system initialized with a single *Seed_Glucose*, 15 *UDP_Glucose* processes, a single *Glycogen_Synthase* and a single *Branching_Enzyme* is shown in Figure 2.15. The processes available in the system after it is run until suspension (no additional enabled communications) are shown in the so-called resolvent in Figure 2.15A, together with the channels they use and the values for the *RC*, *LC* and *LBC* counters. This information is sufficient to specify the molecular architecture of the corresponding glycogen molecule (Figure 2.15B).

BioSpi 2.0 also allows us to follow the time evolution of various quantitative properties in the abstracted population of glycogen molecules, such as the number of branch points, leaves and seeds, giving us an overall measure of molecular architecture. For example, Figure 2.16 shows the number of various processes representing different properties of the polymers (leaves, polymerized residues etc.) under different relative polymerization and branching rates, showing that different degrees of branching can be obtained by changing the balance between polymerization and branching rates.

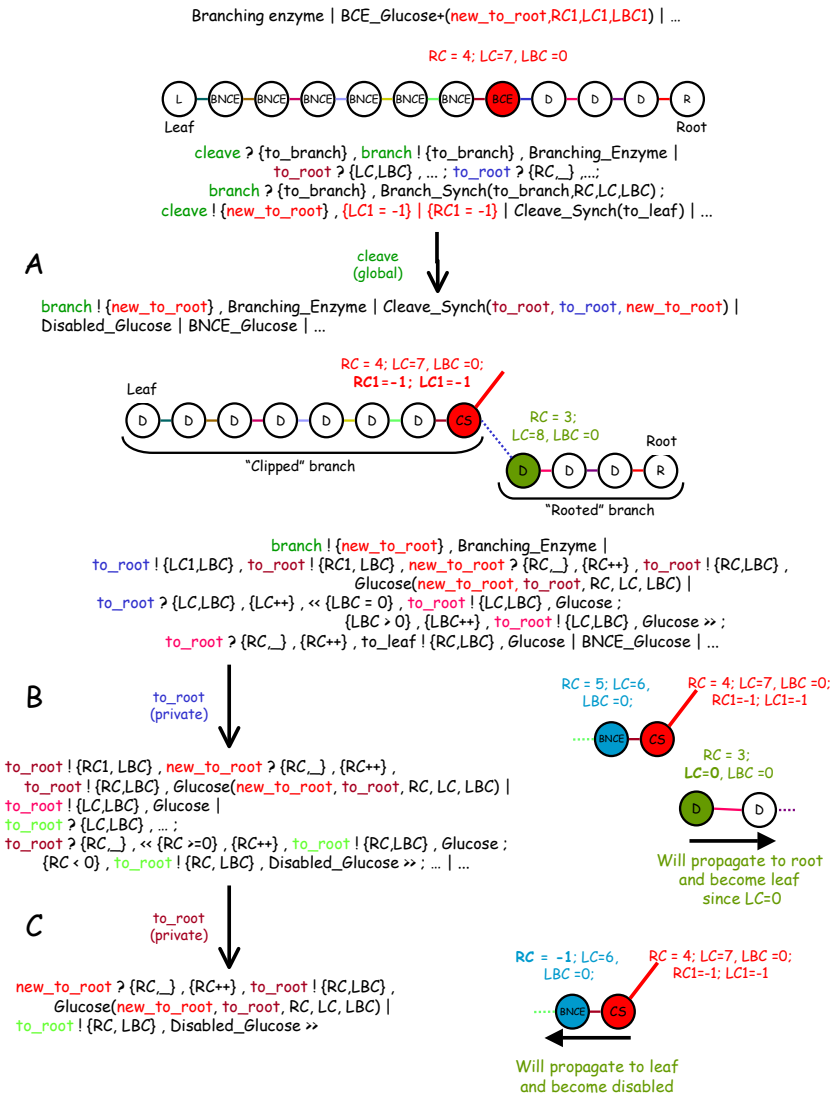

Figure 2.13: **Cleavage as communication between** *Branching_Enzyme* **and** *BCE_Glucose*, **followed by two series of communications on private channels.** A. Communication between *Branching_Enzyme* and *BCE_Glucose* on *cleave*. B. A series of communications in the root direction result in release of a series of processes representing the functional "rooted" branch (only first communications shown). C. A series of communications in the leaf direction, result in release of a series of processes representing a "frozen clipped" branch (only first communications shown).

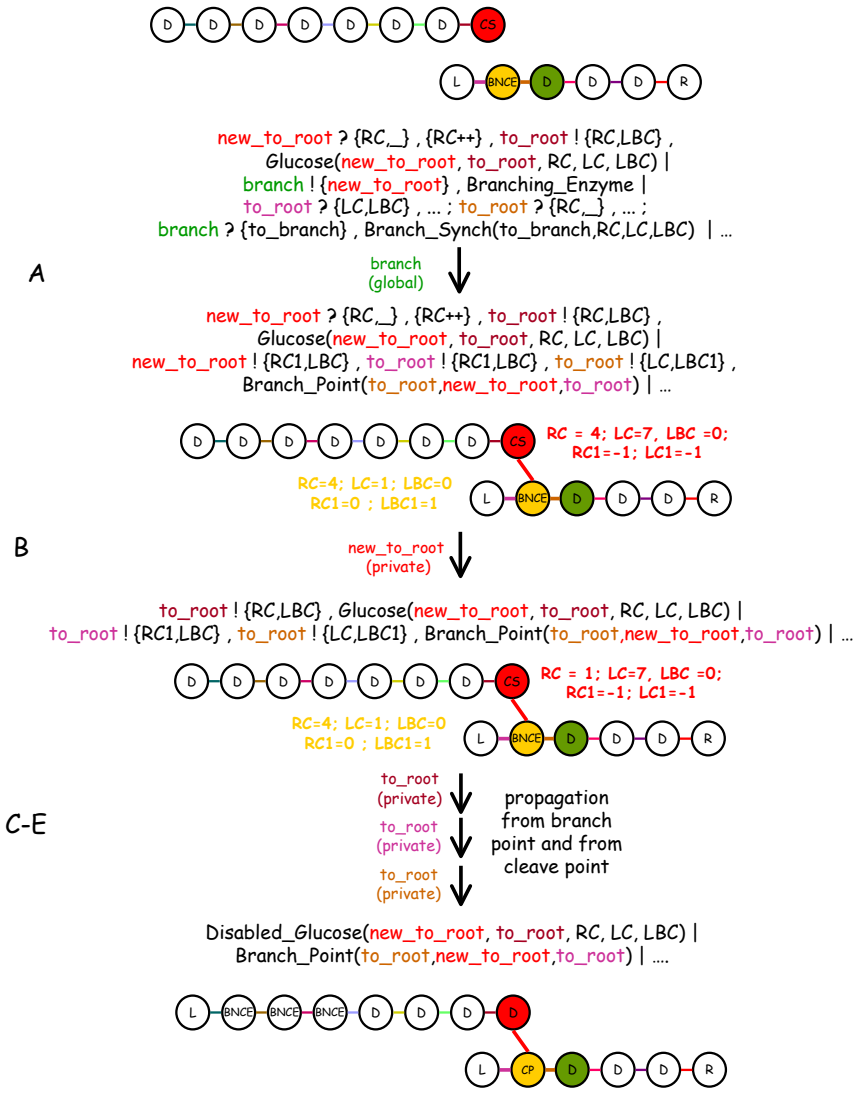

Figure 2.14: **Branch and cleave.** Branch as communication between *Branching_Enzyme*, *BNCE_Glucose*, and *Cleave_Synch*, followed by three series of counter updates using communications on private channels. A. Communication between *Branching_Enzyme* and *BNCE_Glucose* on *branch*. B. *BNCE_Glucose* and *Cleave_Synch* are linked via a private channel. C-E. Update of positional counters and state change by series of communications from the cleave point (to the leaf direction) and from the branch point (to both the root and leaf direction).

A

```
...
.Root_Glucose.comm(.UDP_Glucose.to_root!)
Disabled_Branched_Glucose.comm(.UDP_Glucose.to_root!, .UDP_Glucose.to_root!, 1, 4,
4, global.branch(1)!, global.cleave(1)!, global.glycogen(1)!)
Disabled_Branched_Glucose.comm(.UDP_Glucose.to_root!,.UDP_Glucose.to_root!, 2, 3,
3, global.branch(1)!, global.cleave(1)!, global.glycogen(1)!)
Disabled_Branched_Glucose.comm(.UDP_Glucose.to_root!,.UDP_Glucose.to_root!, 3, 2,
2, global.branch(1)!, global.cleave(1)!, global.glycogen(1)!)
.Branch_Point.comm(.UDP_Glucose.to_root!, BCE_Glucose.new_to_root!,
.UDP_Glucose.to_root!)
Disabled_Glucose.comm(BCE_Glucose.new_to_root!,.UDP_Glucose.to_root!, 1, 8, 0,
global.branch(1)!, global.cleave(1)!, global.glycogen(1)!)
Disabled_Glucose.comm(.UDP_Glucose.to_root!,.UDP_Glucose.to_root!, 2, 7, 0,
global.branch(1)!, global.cleave(1)!, global.glycogen(1)!)
Disabled_Glucose.comm(.UDP_Glucose.to_root!,.UDP_Glucose.to_root!, 3, 6, 0,
global.branch(1)!, global.cleave(1)!, global.glycogen(1)!)
BNCE_Glucose.comm(.UDP_Glucose.to_root!, .UDP_Glucose.to_root!, 4, 5, 0,
global.branch(1)!, global.cleave(1)!, global.glycogen(1)!)
BNCE_Glucose.comm(.UDP_Glucose.to_root!, .UDP_Glucose.to_root!, 5, 4, 0,
global.branch(1)!, global.cleave(1)!, global.glycogen(1)!)
BNCE_Glucose.comm(.UDP_Glucose.to_root!, .UDP_Glucose.to_root!, 6, 3, 0,
global.branch(1)!, global.cleave(1)!, global.glycogen(1)!)
BNCE_Glucose.comm(.UDP_Glucose.to_root!, .UDP_Glucose.to_root!, 7, 2, 0,
global.branch(1)!, global.cleave(1)!, global.glycogen(1)!)
BNCE_Glucose.comm(.UDP_Glucose.to_root!, .UDP_Glucose.to_root!, 8, 1, 0,
global.branch(1)!, global.cleave(1)!, global.glycogen(1)!)
Disabled_Glucose.comm(.UDP_Glucose.to_root!,.UDP_Glucose.to_root!, 1, 1,
0, global.branch(1)!, global.cleave(1)!,global.glycogen(1)!)
Leaf_Glucose.comm(.UDP_Glucose.to_root!, .UDP_Glucose.to_leaf, 2, 0, 0,
global.branch(1)!, global.cleave(1)!, global.glycogen(1)!)
Leaf_Glucose.comm(.UDP_Glucose.to_root!, .UDP_Glucose.to_leaf, 9, 0, 0,
global.branch(1)!, global.cleave(1)!, global.glycogen(1)!)
.Glycogen_Synthase.comm(global.glycogen(1)!, global.udp_glucose(1)!)
.Branching_Enzyme.comm(global.branch(1)!, global.cleave(1)!)
```

B

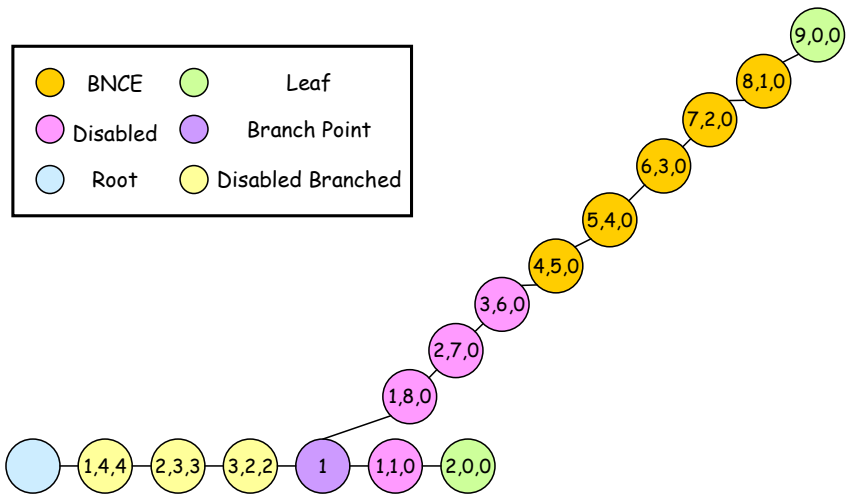

Figure 2.15: **A BioSpi 2.0 resolvent allows reconstruction of the molecular architecture of a glycogen molecule.** A. A BioSpi resolvent following a complete run of a system initialized with a single *Seed_Glucose*, 15 *UDP_Glucose*, a single *Glycogen_Synthase* and a single *Branching_Enzyme*. Process names and the values for the *RC*, *LC* and *LBC* positional counters are highlighted in color. B. The molecular architecture reconstructed from the resolvent. Process shown in circles, with counter values for *RC*, *LC* and *LBC*.

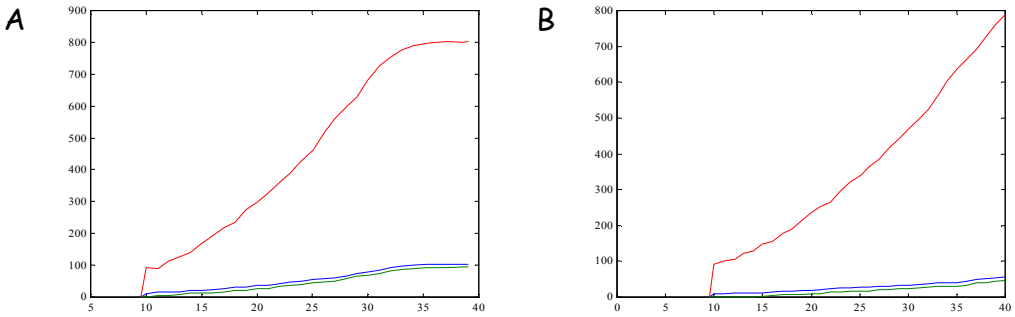

Figure 2.16: **BioSpi 2.0 simulation of glycogen synthesis under different conditions.** The number of *Branch_point* (green), *Leaf_Glucoose* (blue) and "polymerized" (sum of *BCE_Glucose*, *BNCE_Glucose*, *Diabled_Glucose* and *Disabled_Branched_Glucose*) processes (red) is shown as a function of time either when the base rates of the polymerization and branching communications are equal (A) or when the polymerization rate is 10 times faster (B). Note that the number of *Leaf_Glucose* and *Branch_Point* processes is almost identical, as the number of leaves is equal to the summed number of roots and branch points.

2.5.2 A modular abstraction for the circadian clock

The glycogen biosynthesis example emphasized the relevance of the "molecule as computation" abstraction, that allows us to represent the architecture of complex biopolymers in a compositional way. In the next example, a study of the circadian clock, we further use this compositionality to handle the modularity of biomolecular systems. We also highlight the understandability of the abstraction and the framework it sets for the functional study of biomolecular systems.

Circadian rhythms are endogenous biological programs that constitute clocks with a period of about 24 hours. These biological clocks time biochemical and/or physiological events to occur at specific optimal phases of the daily cycle [63]. Circadian rhythms have three major characteristics ([63], [146]): First, under constant conditions, the programs free-run with a period of approximately 24 hours. Second, this endogenous period may be entrained by an environmental cycle (usually a light/dark and/or temperature cycle) and take on the environmental cycle's period. Third, the endogenous period of the free-running rhythm is robust to a wide range of internal and external fluctuations. For example, the circadian clock is temperature-compensated, *i.e.* its endogenous period is nearly the same at different constant ambient temperatures within the physiological range [63]. Another potential source of fluctuations is the presence of internal noise caused by the stochastic nature of chemical reactions and low numbers of molecules ([13],[146]).

A molecular model of the circadian clock

Circadian clocks are found in a large variety of organisms from cyanobacteria to mammals, and have probably evolved more than once ([63], [146]). Recent work [13] suggests that the biomolecular mechanism of clocks shares common features over a wide range of organisms. For example, all networks seem to include an interaction between activator element(s) (*e.g.* KaiA in *Synechococcus*, Wcl-2 in *Neurospora*, Clc and Cyc in *Drosophila*, and Bma1 in mice) and repressor element(s) (*e.g.* KaiB and Kai in *Synechococcus*, Frq in *Neurospora*, Tim and Per in *Drosophila*, and Tim and Per1,2, and 3 in mice). The interaction consists of two interleaved feedback loops. In the positive loop, the activator elements enhances its own expression. In the negative loop, the activator enhances the expression of the negative element, which in

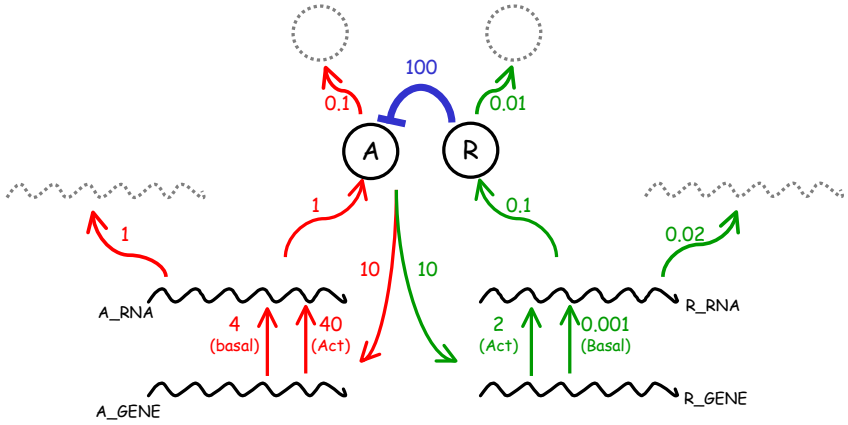

Figure 2.17: **A biomolecular network for a core circadian oscillator.** A and R related reactions are denoted by red and green arrows, respectively. The irreversible R-A binding is shown in blue. The unbinding of the A protein from the A and R promoters is not shown. The rate parameters used for the cognate reactions are shown as numbered labels. These are chosen by [13] as typical ones. For example, the rates for bimolecular reactions are all in the range of diffusion limited reactions. Adapted from [13].

turn sequesters the activator.

Based on these shared principles, Barkai and Leibler [13] have recently proposed an essential model for the core circadian machinery (Figure 2.17). The model involves two genes, an activator, A, and a repressor, R, that are transcribed into mRNA and subsequently translated into protein. The activator A binds to the A and R promoters, and increases their basal transcription rates. Thus, A acts as the positive element in the system, whereas R acts as the negative element by sequestering the activator. This simple model is not intended to abstract any particular biomolecular system, but to capture the basic design principles shared by many systems, and believed to produce its basic functionality.

Barkai and Leibler [13] have also shown that the proper operation of the clock – *i.e.* robust endogenous oscillation – is captured by the model under two conditions. First, to allow oscillations in the model, the reaction rates in the model are clearly distinguished, where the rates of the transcription (both basal and activated), translation and degradation (of both mRNA and protein) of the A species is considerably faster than that of the R species, and that the sequestration reaction is the fastest in the system. Second, extensive analysis and simulation ([13], [146]) has shown that in order for the model to produce oscillations in a wide range of parameter values and to display the known robustness of the biological clock to noise, it should be formulated in a stochastic (rather than continuous) framework.

The importance of quantitative stochastic effects in the operation of the circadian clock render it a good example for our stochastic abstraction framework. We therefore start by building a stochastic π-calculus model (Figure 2.18) based on the network in Figure 2.17.

The π-calculus abstraction of the circadian machinery is straightforward. Each of the relevant molecular species is abstracted as a process: A_Gene, A_RNA, $A\_Protein$, R_Gene, R_RNA, and $R\_Protein$. Specific sub-processes also serve to represent specific molecular states (*e.g.* the activated state of the A gene, is represented by $Activated\_Transcription\_A$). The reactions in the system are abstracted as communications on channels, and reaction rates as channel rates. We distinguish between two types of reactions. Bi-molecular reactions between the A and R molecular species in the system are abstracted as communication between the cognate processes on asymmetric communication channels (*e.g.* the bind-

```
-global(pA(10) ,bA(4),act_rate_A(40),utrA(1),degmA(1),degpA(0.1),pR(10),rbs(100),
bR(0.001),act_rate_R(2),utrR(0.1),degmR(0.002),degpR(0.001)).
```

% A-related processes

```
A_Gene::=  Basal_A | Promoted_A .
     Basal_A::=    bA ? [ ] , Basal_A | A_RNA .
     Promoted_A::= pA ? {unbind} , Activated_Transcription_A(unbind) .
     Activated_Transcription_A(unbindpA)::=
        act_rate_A ? [ ] , A_RNA | Activated_Transcription_A ;
        unbindpA  ? [ ] , Promoted_A .

A_RNA::= utrA ? [ ] , A_RNA | A_Protein  ;
         degmA ? [ ] , 0 .

A_Protein+(unbindC(infinite),unbindpA(10),unbindpR(100))::=
    pA ! {unbindpA} , unbindpA ! [ ] , A_Protein ;
    pR ! {unbindpR} , unbindpR ! [ ] , A_Protein ;
    rbs ! {unbindC} , Bound_A_Protein(unbindC) ;
    degpA ? [ ] , 0 .

    Bound_A_Protein(unbind)::=  degpA ? [ ] , unbind ! [ ] , true ;
                                unbind ! [ ] , A_Protein .
```

% R-related processes

```
R_Gene::=  Basal_R | Promoted_R .
     Basal_R::=    bR ? [ ] , Basal_R | R_RNA .
     Promoted_R::= pR ? {unbind} , Activated_Transcription_R(unbind) .
     Activated_Transcription_R(unbindpR)::=
        act_rate_R ? [ ] , R_RNA | Activated_Transcription_R ;
        unbindpR  ? [ ] , Promoted_R .

R_RNA::= utrR ? [ ] , R_RNA | R_Protein  ;
         degmR ? [ ] , 0 .

R_Protein::= rbs ? {unbindC} , Bound_R_Protein(unbindC) ;
             degpR ? [ ] , 0 .

    Bound_R_Protein(unbind)::= degpR ? [ ] , unbind ? [] , true ;
                               unbind ? [ ] , R_Protein .
```

Figure 2.18: **Stochastic π calculus code for a core molecular model of the circadian clock.** All processes abstracting R and A species (genes, RNA and protein) are shown. *Timer* processes abstracting the transcription, translation and degradation machineries are not shown. The full code consists of 50 lines and 14 processes.

ing of the A protein to the promoter of the R gene is abstracted as a communication between R_Gene and $A\_Protein$ on pA). On the other hand, transcription, translation and degradation reactions in which complex cellular machineries are involved are abstracted as unimolecular reactions, in which the complementary communication is provided by $Machinery$ processes, serving as "timers". For example, degradation of the R mRNA is abstracted as communication between R_RNA (of which multiple copies may exist) and $Degradation\_R$ (of which only a single copy exists) on $degmR$. Note, that in handling these as unimolecular reactions we follow the approach of [13].

The model also employs private channels to abstract the individual identity of A-R, A-pA, and A-pR complexes. In the first case, we assume that the A-R complex cannot break unless one of its two components is degraded. Thus, the private $unbindC$ channel is employed only *after* the communication representing degradation, and is an instantaneous one. In the other cases, we assume that the A protein may unbind from the A and R promoters. The rates of $unbindpA$ and $unbindpR$ represent these unbinding rates, and the communication on these channels represents the corresponding unbinding events. The abstraction of different reactions in the system as communication is shown in Figures 2.19, 2.20, and 2.21.

As shown in Figure 2.22, the resulting model yields the required oscillatory behavior, for each of its components. In this we have reproduced the known result of [13], providing support for the correctness of the stochastic π-calculus abstraction.

A modular model of a hysteretic clock

The π-calculus model of the molecular mechanism of the circadian clock seems similar to that obtained with a traditional kinetic formalism. In fact, such a simple model is just as easy (or even simpler) to express in the traditional scheme, and does not seem to gain much in relevance by employing the alternative "molecule as computation" abstraction.

The potential of employing the "molecule as computation" abstraction is highlighted by the next step of our study. In addition to illustrating the robust oscillations of the core molecular model, Barkai and Leibler [13] suggest that the "A component" of the clock machinery serves as a hysteretic *module*, sensing the level of the R protein in the system (Figure 2.23A,B) and responding by switching between two functional states: An "ON" state, in which the system is induced (both genes are actively transcribed), and an "OFF" state, in which the system is repressed (both genes are transcribed only at the basal level). The switch between the two states is fast and occurs at a narrow threshold, depending on R's level. The circadian machinery can thus be represented at two levels of abstraction. The first is a *molecular* level, detailing all molecules and their interactions. The second is a *modular* level, in which parts of the system are encapsulated and only their *functional behavior* is represented, while abstracting away the underlying molecular details.

The "molecule as computation" abstraction has two important benefits in understanding such systems. First, it provides a semantic framework in which to prove the hypothesis that the "A component" constitutes a hysteresis module. As discussed in Chapter 1, the mathematical domain of concurrent computation distinguishes between two levels of abstraction in which a computational system may be described: the *implementation* and the *specification*. The implementation level describes how a specific system is built, while the specification level describes its function or potential behavior. The underlying theory defines that the two descriptions are equivalent if they can be exchanged within any context, without changing the observable behavior of the system (*e.g.* the system's "output") [89].

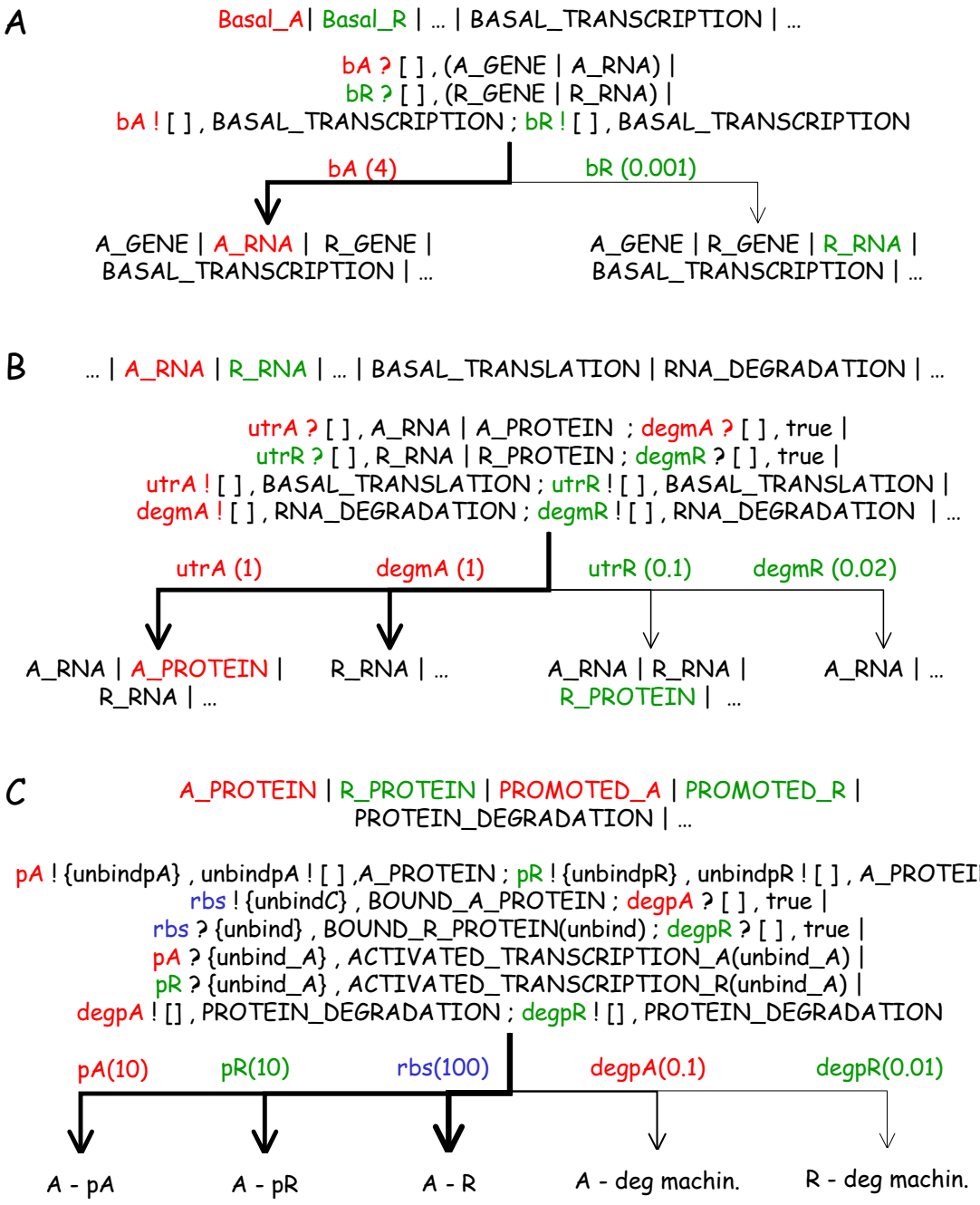

Figure 2.19: **Potential reactions in the clock as communications: Basal transcription, translation, RNA degradation, and various reactions by proteins A and R.** A. Basal translation. As bA has a faster base rate than bR, communication (representing basal transcription) with A_Gene is more likely. B. Translation and RNA degradation. Communications on $utrA$, $degmA$, $utrR$, and $degmR$ represent translation and degradation of the A and R proteins, respectively. The probability of communication depends on each channel's base rate and the number of A_RNA and R_RNA. C. The interactions in which proteins A and R may participate are abstracted as communication on pA (binding of A to Gene A's promoter), pR (binding of A to Gene R's promoter), rbs (formation of A-R complex), $degpA$, and $degpR$ (degradation of A and R respectively).

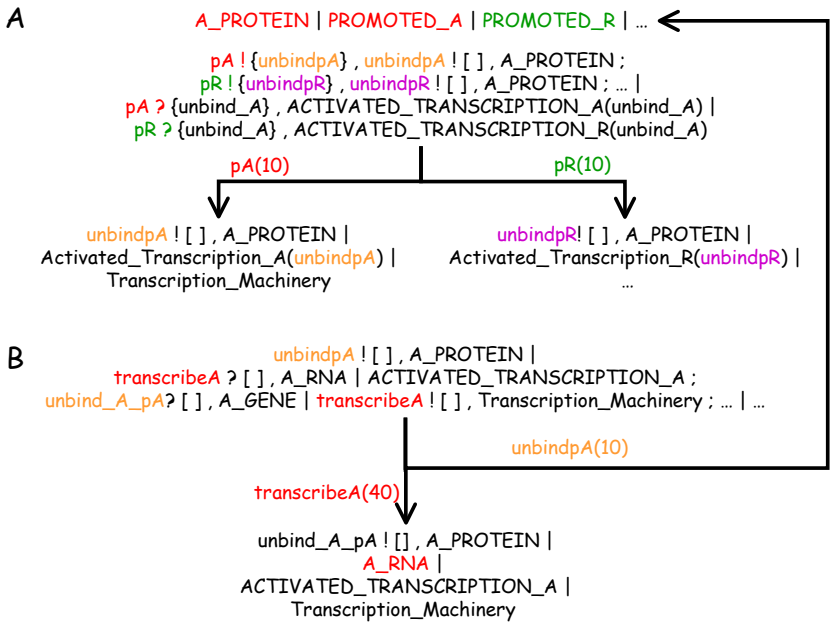

Figure 2.20: **Potential reactions in the clock using communications on private channels: Activated transcription.** Activated transcription is abstracted by two communication steps. In the first step (A) protein A binding to gene A or R's promoter is abstracted as a communication between $A\_Protein$ and $Promoted\_A$ or $Promoted\_R$ (on pA or pR, respectively). In the communication, a private channel is exchanged. B. In the second step (shown only for the "A" side), transcription is represented as a communication between $Activated\_Transcription$ and $Basal\_Machinery$ leading to creation of an RNA process. Alternatively, a communication on the private channel represents the unbinding of the A protein from the promoter.

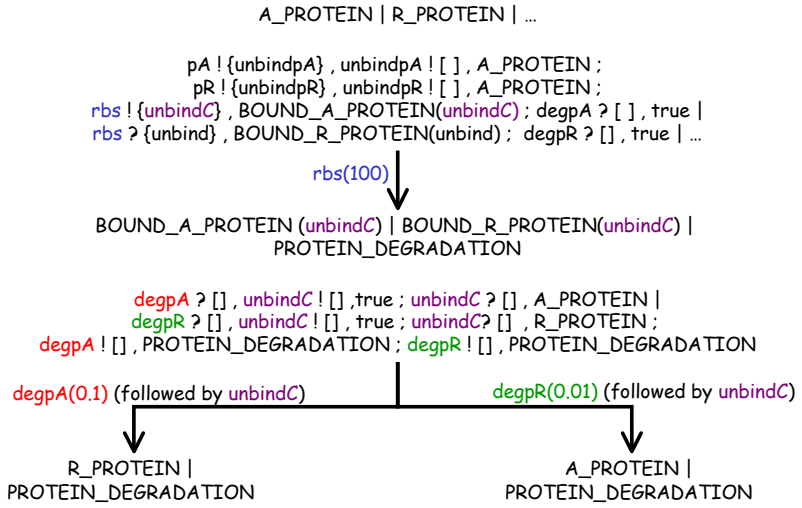

Figure 2.21: **Potential reactions in the clock using communications on private channels: A-R complex.** Complex formation is abstracted as a communication on the rbs channel and exchange of the $unbindC$ channel. As the complex may only break upon degradation of one of its components, the $unbindC$ communication is instantaneous and follows communication on either $degpA$ (in $Bound\_Protein\_A$) or $degpR$ (in $Bound\_Protein\_R$ process).

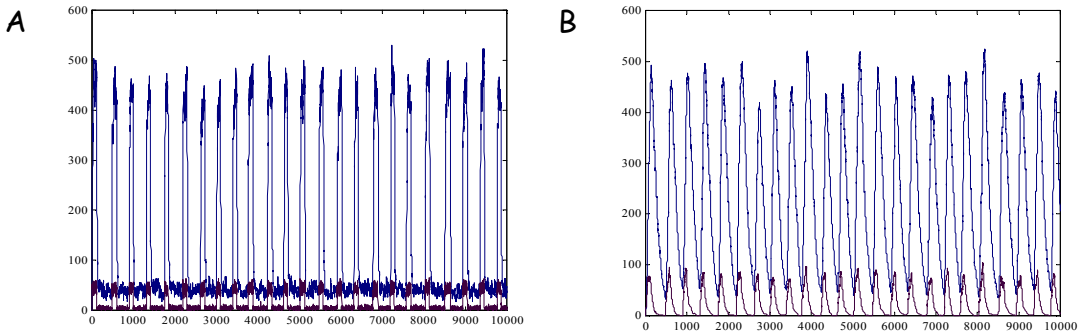

Figure 2.22: **BioSpi 2.0 simulation of the core model of the circadian oscillator.** The π-calculus code (Figure 2.18) was run for 10,000 time units using BioSpi 2.0. Process numbers were plotted as a function of time and illustrate oscillatory behavior. A. "A" related processes. $A\_Protein+Bound\_A\_Protein$ (blue) and RNA_A numbers show sharp "switch-like" oscillations. B. "R" related processes. $R\_Protein+Bound\_R\_Protein$ and RNA_R numbers show gradual oscillations.

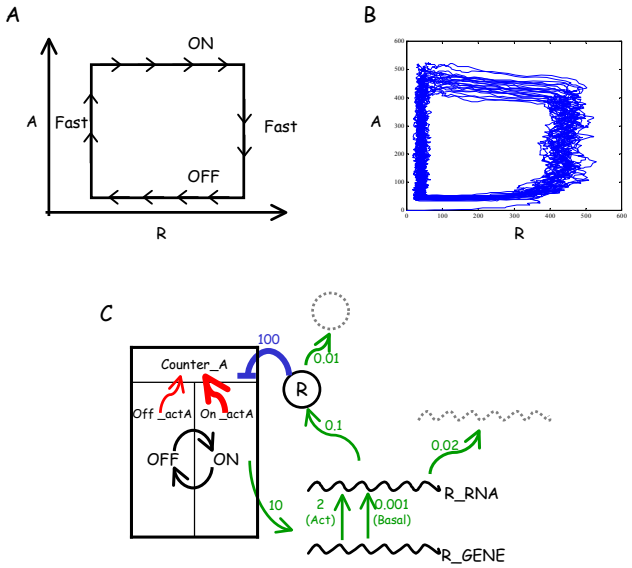

Figure 2.23: **The "A" hysteresis module in the circadian machinery.** A. A schematic view of hysteresis, a bistable dependence of A's concentration (and functional state) on R. The slow kinetics of R lead to oscillations which can be abstracted as successive transitions between "induced" and "repressed" states (adapted from [13]). B. The level of total $Protein\_A$ vs. $Protein\_R$ processes in a BioSpi 2.0 simulation of the molecular model. C. Schematic depiction of the "A component" encapsulated as a hysteresis module.

```
Module_Init+(unbindC(infinite),unbindpR(100))::= ON | A_Increase.

ON::= {CA =< 0} , OFF(unbindC) ;
      {CA >  0} , << rbs! CA*{unbindC} , {CA--} , ON ;
                    on_rate_A ! [ ] , {CA++}, ON ;
                    degpA ? [ ] ,  unbindC ? [] , ON ;
                    pR ! CA*{unbindpR} , {CA--} , ON ;
                    unbindpR ! [ ] , {CA++} ,  ON ;
                    unbindC ? [ ] , {CA++}, ON  >> .

OFF(unbindC)::=
    {CA > 1} , ON ;
    {CA =< 1} , << rbs ! CA*{unbindC} , {CA--} , OFF ;
                   off_rate_A ! [ ] | {CA++} | OFF ;
                   degpA ? [ ] ,  unbindC ? [] , OFF ;
                   unbindC ? [ ] | {CA++} | OFF_H_MODULE >>.

A_increase::= on_rate_A ? [ ] , A_increase ;
              off_rate_A ? [ ] , A_increase .
```

Figure 2.24: **Stochastic π calculus code for a "A" hysteresis module.** Only the module code is shown. The remainder of the "molecular" code remains intact.

In the case of the circadian clock we can thus consider the abstraction of the real molecular system (the "A component") as an implementation-level abstraction, and that of the functional hysteresis module as a specification-level abstraction. We can then show that the two representations are equivalent by either formal verification or computer simulation of their respective behavior. As formal verification is beyond the scope of this thesis, we take the latter route.

The second benefit of the "molecule as computation" approach becomes apparent when we wish to build and test the functional-level abstraction. The compositionality of the π-calculus abstraction allows us to abstract only the "A component" of the model without touching the remainder of the model (as schematically depicted in Figure 2.23C).

The code replacing the "A-related processes" (*A_Gene*, *A_RNA* and *A_Protein* processes and their sub-processes) is shown in Figure 2.24. In this modular representation, a *Module_Init* process switches between two states (*ON* and *OFF*) based on the value of an internal *CA* counter, abstracting the number of free protein A molecules. Both the *ON* and the *OFF* processes can communicate with *Protein_R* and *Bound_Protein_R* (just like *Protein_A* and *Bound_Protein_A* could). These communications are followed by a decrease and increase in *CA*, respectively. Only the *ON* process, however, can communicate with *R_Gene*, representing the induced state of the system, in which R's transcription is on. In both the *ON* and *OFF* states, *CA* is incremented following a communication on *on_rate_A* and *off_rate_A*, respectively, representing a higher and lower level of protein A synthesis in the two states. Importantly, *CA* also affects reaction rates, by serving as a *multiplier* in all the communications that abstract interactions in which the A protein was involved.

This modular "specification" passes the simulation test. Figure 2.25 compares the level of the *R_Protein* process when run together with either an implementation-level or a specification-level abstraction of the "A component". The correspondence between the levels of *R_Protein* in the two runs serves as an indication of the "equivalence" of the molecular "A component" and the hysteresis module.

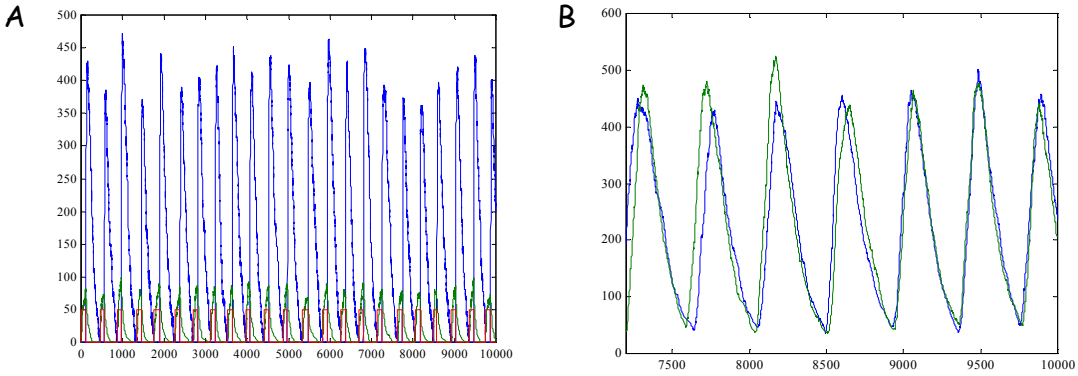

Figure 2.25: **BioSpi 2.0 simulation of the "A" hysteresis module.** A. Levels of R_RNA (green) and free $R\_Protein$ (blue) in a simulation of a π-calculus program in which the "A component" was replaced with a modular model. The ON/OFF state of the $Module$ process in shown in red. B. Comparison of the level of the total number of processes representing free ($R\_Protein$) and bound ($Bound\_R\_Protein$) molecules, between "molecular" (blue) and "modular" (green) runs.

We discuss the steps necessary to further establish this equivalence in Section 2.6.

Studying modular models: Gating of the cell cycle by the circadian clock

A modular approach to abstraction, in which parts of the system are represented at the molecular (implementation) level and others at the functional (specification) level allows us to abstract complex systems incrementally and provides a single framework for molecular and functional information. This is particularly important when our level of knowledge is variable.

As the (compositional) π-calculus easily lends itself to modular abstractions, we pursue this approach to further explore the circadian clock in a wider cellular context, building a larger model in various levels of detail. Barkai and Leibler [13] have shown that a stochastic formulation of the molecular model is robust to random perturbations in transcription or translation rates. This finding was further validated in [146]. However, in a cellular context the circadian clock may be perturbed in a *regular* rather than random fashion due to the presence of an additional oscillatory mechanism - the cell division cycle. In fact, these two major cyclic phenomena are not independent in many organisms (Figure 2.26). Rather, the cell division cycle is *gated* by the circadian rhythm in many unicellular organisms, such that cell division and/or DNA synthesis may only occur in specific phases of the clock. These include the eukaryotes *Chlamidonomas* ([47]) and *Euglena* [22] and the cyanobacteria *Synecchococcus* (*e.g.* [90]) and *Prochlorococcus* ([144], [124]). We also expect a reverse effect of the cell division cycle on the circadian clock. For instance, the transcription rate of the A and R proteins may be affected by the different gene copy number (1 or 2 in haploid organisms) in different phases of the clock. Conversely, transcription rates may be affected by the (interfering) process of DNA synthesis.

In fact, we hypothesize that gating is a *counter-mechanism* to buffer the regular perturbation of the circadian clock by the cell division cycle. Thus, the cell is allowed to be in a state when gene copy number is doubled, only when the clock is in the "Off" state (R is high and A is low and fully bound to R) and there is no active transcription. This hypothesis is supported by the fact that gating has been primarily observed in organisms where the period of the cell division cycle is around the same order as the 24h period of the clock (including 6-10h in cyanobacteria) [90], and may thus interfere with its period.

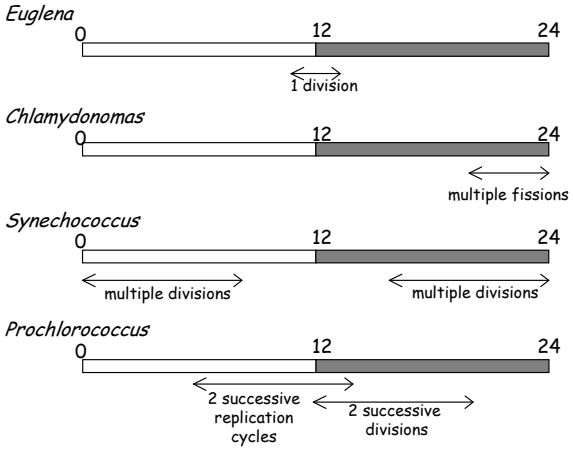

Figure 2.26: **Gating of the cell division cycle by the circadian clock in several unicellular organisms.** The allowed phases for cell division are shown relative to the endogenous "light"(white)-"dark"(grey) cycle.

To investigate this we extended our π-calculus model of the circadian clock with a *Cell_Cycle* process (Figure 2.27). As before, the compositionality of the calculus allowed us to seamlessly perform this extension. Since the eukaryotic cell cycle is highly complex, and the prokaryotic one is not well-understood, we opted for a highly simplified functional (specification) model of the cell division cycle. We assumed that the major functionality in which we are interested is a change in gene copy number. In the resulting model, the cell cycle is abstracted as two alternating processes, *Phase*1 and *Phase*2. Communications on *rep* and *div* initiate the state switch from *Phase*1 to *Phase*2 and vice versa, respectively. The channels' rates corresponds to the average length of the corresponding phases. Thus, the cell cycle is seen as a stochastic process, with a characteristic, but not necessarily fixed, period (and phase length[17]). Upon switching from *Phase*1 to *Phase*2, two additional *A_Gene* and *R_Gene* processes are created in the system. The created processes are equipped with the private *divide* channel. This channel parameter and its subsequent use in the *Gene* processes is the only change in the rest of the molecular model. Upon the reverse switch from *Phase*2 to *Phase*1, alerts are sent on the private *divide* channel, that will "kill" the receiving *Gene* processes, resetting their number to a single copy.

BioSpi simulation of the clock model together with the cell cycle model (Figure 2.28A) verifies our first hypothesis that, at least in this simplified model the cell division oscillator significantly perturbs the circadian one. Furthermore, this disturbance is observed more prominently when the period of the cell division cycle is around the order of that of the circadian clock and even considerably shorter (consistent with the observation of gating in cyanobacteria with a cell cycle period of 6h). When the cell cycle period is significantly longer (Figure 2.28B) or extremely short (Figure 2.28C) we observe only slight perturbation to the period of the circadian clock.

Next, we examined whether this perturbation can be buffered by a gating mechanism controlled by one or more of the components of the circadian clock. We tested several alternative hypotheses regarding the identity of the "gate-keeper" (A, R or both) and the required gating point (entry or exit to *Phase*2 or both). The hypotheses were tested both in a molecular and a modular context. In each case, we modified the above model in two ways. First, we equipped *Phase*1 and/or *Phase*2 with additional

[17] In building this component as a stochastic one, we increase the level of noise that may be introduced by the *Cell_cycle* process. An alternative model without such additional stochasticity yields similar results (not shown).

```
Phase1::= Replication | Checkpoint1 .

    Replication+divide(infinite)::=
         rep ? [ ] , A_GENE(divide)  | R_GENE(divide) | Phase2(divide) .
    Checkpoint1::=
         rep ! [] , true .

Phase2(divide)::= Division(divide) | Checkpoint2 .

    Division::=
         div ? [] , divide ! [] , divide ! [] , divide ! [] , divide ! [] , Phase1 .
    Checkpoint2::=
         div ! [] , true
```

Figure 2.27: **A highly simplified model of the cell cycle in the π-calculus.**

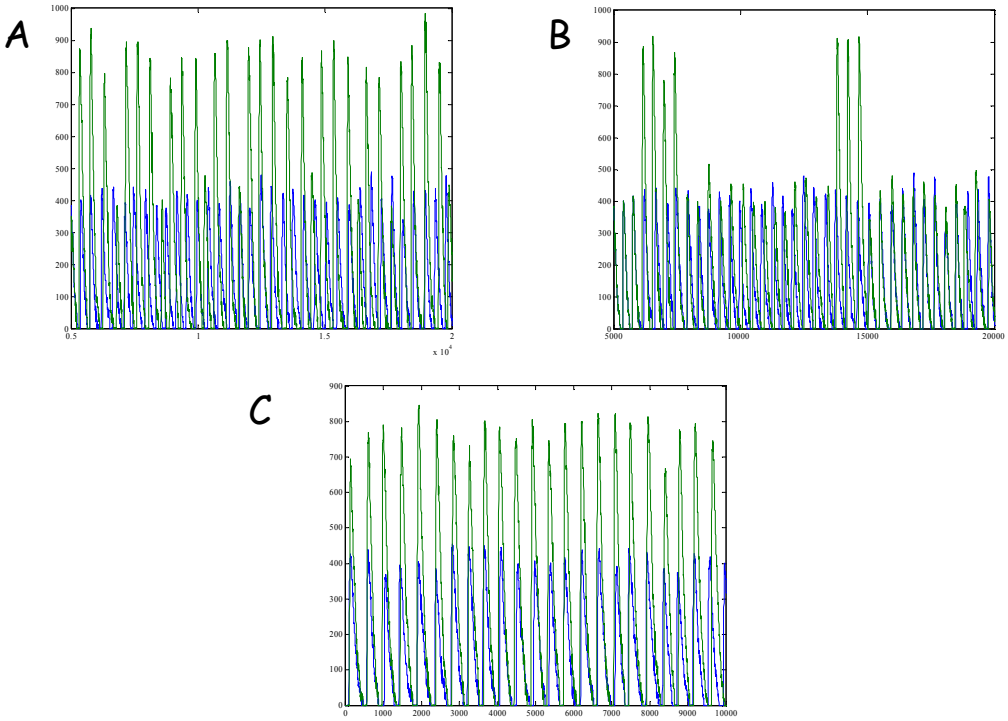

Figure 2.28: **BioSpi 2.0 simulation demonstrates that the cell division cycle may perturb the circadian clock under certain conditions.** The *R_Protein* process levels are compared between a run of the molecular model (blue) and of the same model with the necessary modification together with a *Cell_Cycle* process (green). A. Perturbation of the circadian clock's period when the cell cycle period is 600 units (approximately the clock's period). B,C. Slighter perturbation of the clock's period when the cell cycle's period is significantly longer (B, 10X that of the clock) or extremely short (C, 0.01 that of the clock). The shown segments are representatives of runs for 30,000 units.

% Cell cycle

```
Phase1::= Replication | Checkpoint1 .

  Replication+divide(infinite)::=
     rep ? [ ] , rin ? [ ] , A_Gene(divide)  | R_Gene(divide) | Phase2(divide) .
  Checkpoint1::=
     rep ! [] , true .

Phase2(divide)::= Division(divide) | Checkpoint2 .

  Division::= div ? [] , divide ! [] , divide ! [] , divide ! [] , divide ! [] , Phase1 .

  Checkpoint2::= div ! [] , true .

Checkpoint2::= div ! [ ] , true ;
               rstay ? {unbindR} , unbindR ? [ ] , Checkpoint2 .
```

% R protein

```
R_Protein+unbindR(infinite)::=
  rbs ? {unbindC} , Bound_R_Protein(unbindC) ;
  degpR ? [ ] , 0 ;
  rstay ! {unbindR} , degpR ? [ ] , unbindR ! [ ] , true .

  Bound_R_Protein(unbind)::= degpR ? [ ] , unbind ? [] , true ;
                             unbind ? [ ] , R_Protein .
```

Figure 2.29: **π-calculus code for a gating mechanism.** The communication clauses with *rin* and *rstay* are the only additions in the code over the "non-gated" version.

communications on *in* (representing a *Phase*2 entry gate) and/or *stay* (representing a *Phase*2 exit gate). Second, we added the complementary communications on these channels as additional choices in the code for *A_Protein* and/or *R_Protein*. For example, the code for a double gate mechanism (both entry and exit from *Phase*2) controlled by the *R_Protein* is shown in Figure 2.29.

Each of the models was run in the BioSpi 2.0 system. As can be seen from Table 2.5 only the double gate mechanism by *R_Protein* has the desired effect, both in a molecular and a modular context (Figure 2.30A and B, respectively). This result raises two important conclusions. First, buffering the perturbation in the circadian clock model can be achieved by gating. Second, gating must be exerted in a specific way. We discuss the reasons for this in Section 2.6.

2.6 Perspectives: Stochastic π-calculus models of biomolecular systems

While the function of certain biomolecular systems may be abstracted in a qualitative or semi-quantitative way, the functionality of others, such as the circadian oscillator, critically depends on quantitative aspects.

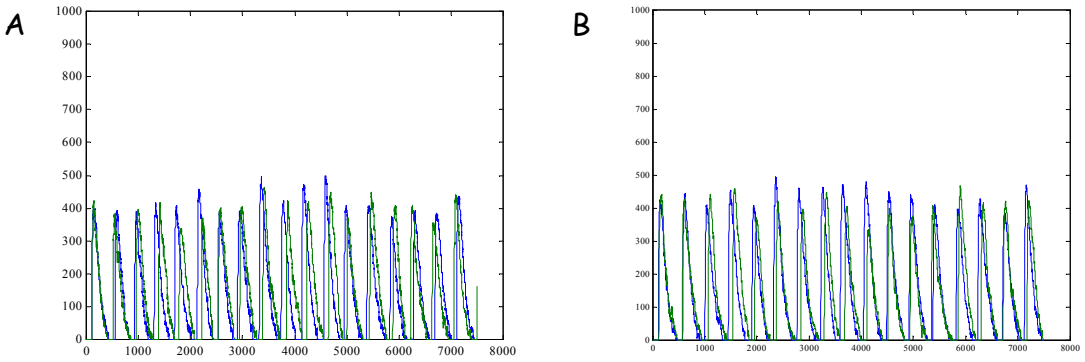

Figure 2.30: **BioSpi 2.0 simulation demonstrates that the perturbation induced by the cell division cycle can be buffered by a gate mechanism controlled by the R protein.** *R_Protein* and *R_RNA* levels are compared between runs without *Cell_Cycle* (blue) and with a "gated" *Cell_Cycle* (green) in the context of a molecular model of A related processes (A) or of a modular model (B).

Gate Keeper	**Gate point**	**Buffering successful?**
A only	Entry only	No
A module only	Entry only	No
	Entry and exit	No
A and R	Entry only	No
	Entry (A and R) and exit (R only)	Yes
R only (with molecular or modular model of A)	Entry and exit	Yes

Table 2.5: **Potential gating mechanisms and their success based on BioSpi 2.0 simulations.**

Thus, a correct abstraction for biomolecular systems must include a quantitative component. In certain systems, due to limited knowledge or their inherent qualitative behavior (*e.g.* a switch) a qualitative approach will suffice or is inevitable. In others, a fully-quantitative abstraction is the only correct one.

There are two alternative quantitative approaches to biomolecular systems: a continuous and a stochastic one. Although the former is more developed analytically, there is a growing recognition of the importance of stochasticity in the function of biomolecular systems (*e.g.* [146], [9], [91]). Since the stochastic formulation is naturally discrete, it is closer to both biomolecular systems and our semi-quantitative abstractions. Thus, we chose this framework as a basis for a quantitative extension of the π-calculus.

2.6.1 The stochastic extension of the π-calculus

There are two previous stochastic frameworks for (bio)chemical systems. The first, known as "The Gillespie Algorithm" [42], is based on the "reaction as object" abstraction. The second, underlying the StochSim Algorithm, is based on the "molecule-as-object" abstraction. Each of the two approaches has benefits and drawbacks. Gillespie's method is well-established and supported by strict proofs and a rigorous theoretical framework. However, the approach does not identify molecules as independent multi-state objects whose identity is maintained through reactions. As such it is more relevant as an abstraction of chemistry than of biology.

The more recent StochSim algorithm [91] addresses exactly this problem by employing the "molecule as object" abstraction. However, its theoretical basis, though derived from the same general assumptions as the Gillespie algorithm, is not fully established, and its exact formulation includes several details which may not be easily generalizable. Furthermore, the strongest part of the StochSim approach, the ability to express and handle multi-state molecules, is done by a mixture of *procedural* and *declarative* approaches, such that the actual solutions are somewhat *ad hoc* and the abstraction is blurred between the description of the object and the computational tool (simulation program) used to execute it.

In principle we could adapt both approaches as a basis for a quantitative stochastic extension of the π-calculus abstraction. Although the StochSim approach is closer to the "molecule as computation" abstraction, we selected the Gillespie one as the basis for our extension. There are two reasons for this choice. First, we wished to enjoy the strong theoretical background, rigorous formulation, and well-accepted status of the approach. Second, by marrying these benefits to the "molecule as computation" abstraction, we bring many of the StochSim benefits as a relevant abstraction, building an overall better solution.

The extension of the π-calculus representation to a stochastic framework is an essentially seamless one, as the necessary components - both theoretically (devising the semantics) and practically (building the implementation) - can be added without major modification to the existing ones. This extensibility of the calculus is an important benefit. In Chapter 3 we extend the calculus further to handle biological compartments, enjoying similar benefits.

The BioSpi 2.0 simulation system, implementing the stochastic π-calculus has the necessary basic functionalities ensuring accurate quantitative simulation according to biochemical principles. However, the current system has several limitations. First, like any stochastic simulation system it is computationally heavy, and is currently limited in the *size* of the system it can simulate.[18] This is a major

[18]The current simulations were limited to approximately 10,000 concurrent processes on a SunOS 4u Ultra-SPARC machine. Typical runs were between 1-30 minutes, with the largest systems run in this work (the circadian clock runs for 10-20

practical limitation to the simulation of complex biological systems. Note, however, that very large scale systems may not justify a stochastic formulation, and would require alternative extensions (including ones based on "molecular species as computation" abstractions, as discussed in Chapter 1). Second, once a channel is selected according to the Gillespie algorithm, the current selection of the send/receive pair is not done in a fully random way (rather, it is done, "off-the-top-of-the-queue"). Usually, this does not raise a problem as all the sender (or receiver) processes on the channel are "equivalent". However, when different processes may send (receive) messages on the same channel, this may cause inaccuracies. To address this problem, we are currently implementing a fully-random selection of communication offers from the channel queue. This will be added as an option to new BioSpi releases. Third, while the BioSpi 2.0 system collects detailed information at the process and channel level throughout the simulation, it currently provides only limited interface to examine this "reactor" during the simulation (in the form of debugging tools) and after it is finished (in the form of record files). Further development is required to improve the accessibility of the user to this rich data.

2.6.2 The stochastic π-calculus as an abstraction of biomolecular systems

The guidelines for the abstraction of quantitative aspects in the π-calculus are few and straightforward: elementary reactions with mesoscopic rate constants are abstracted as channels with base rates, and the actual reaction rate is calculated as an actual channel rate from the base rate and the number of processes offering communications. The selection of a time step and actual communication is based on these actual rates and follows Gillespie's algorithm. Within these general guidelines we distinguish between the abstraction of four types of reactions: asymmetric bimolecular reactions, symmetric elementary reactions, unimolecular elementary reactions and instantaneous reactions. The former three are *bone fide* elementary reaction, while the latter is introduced as a way to either encode "non-biochemical steps" or to handle reactions faster in several orders of magnitude than the rest of the system. The different reactions are abstracted by using three types of channels, each with the corresponding rule for the calculation of actual rate: asymmetric, symmetric and instantaneous. The asymmetric channels are used to abstract both bimolecular asymmetric elementary reactions and unimolecular ones. There are two alternative (but equivalent) guidelines for the abstraction of the latter: use of single-copy $Timer$ processes that supply the complementary communication, or use of private channels between two sub-processes. While the abstraction is essentially limited to elementary reactions, BioSpi allows, in principle, to define alternative rate calculation rule per (asymmetric) channel and extend the channel type scheme accordingly. In this case, however, the correctness of the simulation of the resulting system is no longer ensured.

Compositional modeling: Glycogen biosynthesis

The stochastic extension of the "molecule as computation" abstraction enables its application to a wide variety of biomolecular systems in which quantitative aspects are key. In particular, we used it to model two systems: the glycogen biosynthesis system [147] and the core circadian machinery [13]. Both examples highlight some of the benefits of the abstraction in handling modular systems with multiple components.

Glycogen biosynthesis is an example to the challenge posed by polymers and other biological entities that change dynamically and are difficult to define. In glycogen, the exact configuration of the polymer – the number of constituent monomers, their position and branching pattern – is constantly changing.

cycles) takins approximately an hour. These limits greatly depends on the available computational resources.

Glycogen is an *open-ended* entity, whose boundaries and structure are not pre-defined, but it is composed of (relatively) simple monomers, whose capabilities are well-known. The *compositionality* of the "molecule as computation" abstraction – the ability to compose complex entities from constituent parts according to pre-defined rules – offers a unique way to handle such entities. Indeed, the glycogen biosynthesis model in the π-calculus provides a concise and accurate description of the monomer parts and generates the correct behavior in simulation. Importantly, the approach is generalizable to other polymers and compositional biomolecular systems, and stems from the essence of the abstraction. Note, however, that the compositional approach has drawbacks as well. First, it requires us to maintain each monomer as an independent entity (process) in the abstract model (and simulation). Second, in a compositional representation all properties are "distributed" among the individual components. In the glycogen example this required constant status updates of a series of counters throughout the affected area. This can pose both a computational burden and affect the readability of the abstraction. One solution to this problem is to build a hybrid abstraction in which sections of the polymer are abstracted as a single process.

Modularity: The circadian clock

Our study of the circadian clock highlights some of the novel insights that can be gleaned when employing the "molecule as computation" abstraction. Our starting point was the stochastic molecular model of [13], which we have re-formulated in the stochastic π-calculus, along the usual abstraction guidelines. The resulting model, while generating the expected behavior, did not in itself have an advantage over standard methods. Since the core molecular model of the circadian clock is simplified and involves little molecular detail the usual benefits of our abstractions were not apparent.

In the next two steps we used this model as a stepping stone to introduce new questions on the modular organization of the circadian clock and its integrated function in a cellular context. First, we followed the computational approach that distinguishes between a specification and an implementation of a system. We applied the approach to the circadian clock by handling the "real" molecular system as an implementation and the proposed functionality of the "A" component as a specification. In doing so, we obtain a clear measure of equivalence, determining that an implementation and specification are equivalent if they can be interchanged within any context without any observable change in the system's behavior. The next natural step is to apply formal verification methods (*e.g.* [145], [96]) to prove such an equivalence. Unfortunately, the formal verification of quantitative properties in stochastic settings (*e.g.* [94]) is not as well developed as that of qualitative systems. Furthermore, such a long-term endeavor is clearly beyond the scope of this thesis. Thus, we selected an alternative, simulation-based route and compared the simulated behavior of the two abstractions. Importantly, the simulation route also requires that the equivalence be tested "in all contexts", *e.g* under various types of "R components" (different rates, etc.). This work remains to be completed and is currently underway.

In the second step, we followed the modular approach to study the circadian clock in a wider cellular context and examine some reciprocal effects of the clock and the cell division cycle. The compositionality of our approach facilitates the manipulation of the abstract model, "plugging" pieces in and out without perturbing the remaining components or requiring only minimal changes. Indeed, we easily added components representing a (gated) cell cycle, with changes confined to new events (gating, replication) that were completely absent from the original model. The results of our simulation studies were consistent with a variety of phenomenological observations that were not previously explained at the molecular level. Thus, we observed a perturbation in the oscillation of the processes representing clock components

by a process representing a cell cycle oscillator, but only when the cell cycle period was similar to the circadian one. We also showed that this perturbation can be buffered by the circadian machinery. We pointed at the R component (but not the A component) as the potential gate-keeper, and at two necessary points for gating. These findings can be theoretically explained. First, the need for an entry and an exit gate is clear. Since a duplication of gene-copy-number does not perturb the clock only when it is in the "off" state, the clock must not only ensure that duplication does not occur when it is "on" but also that division is completed before it reaches the next "on" state. Second, the R protein is the only available sensor for events during the "off" state, since the A component constitutes a hysteretic bi-stable switch, and thus changes rapidly and seemingly unexpectedly, together with the state change. Importantly, this is also a novel and focused prediction that can be verified experimentally, for example by exploring gating in cyanobacteria mutated in one or more of the clock components.

2.6.3 The limitations of simulation

While simulation studies of modeled systems have considerable benefits and may provide new biological insight, one should also bear in mind that there are several limitations of the simulation approach. These limitations are mainly related to the modeling of systems that contain components for which full information is lacking. There are several cases to consider. First, we may be completely unaware of some real-world components of the system. In this case, simulation results will be inconsistent with real world observations, indicating this knowledge gap. Second, we may be aware of the existence of certain components but not of their detailed interactions and behavior. In such a case, our environment has a unique advantage, in allowing us to integrate heterogeneous components at different levels of detail into the same model. In fact, this is exactly the method we applied for integrating a coarse model of a cell-cycle component with a detailed molecular model of the circadian clock. Third, we may know all the components and their interactions at the qualitative but not quantitative level. In such a case, we are able to search for the ranges of parameters that satisfy certain properties of the system, *e.g.* robustness. While most of the above limitations are true for modeling in general, the focus of the stochastic variant of the π calculus on elementary reactions poses an extra burden on the level of required knowledge.

A complementary approach to modeling, which tries to reconstruct a system's structure and parameters from raw experimental measurements is described in the appendix.

2.6.4 Future prospects: Homology and analogy of biomolecular processes

An exciting direction of future research emanating from our work is a study of the evolution and function of molecular processes. As in any kind of phylogenetic study, the starting point must be a measure of similarity between the evolving entities. The majority of previous work has focused on measures of similarity between homologous or analogous structures (*e.g.* sequences, 3 dimensional structures or morphologies). Sporadic attempts at comparing biological pathways has taken a similar structural path ([37], [34]). The π-calculus, however, paves the way towards a unique measure of "process similarity".

Indeed, in its original domain of application, concurrency theorists have used the π-calculus in order to define several measures of behavioral correspondence between processes. In CS, such measures are applied, for instance, to verify that an implemented piece of software (one process) is sufficiently similar to the software's original design (a second process), thereby ensuring that it actually does what it was supposed to do. Thus, a general framework for comparing processes exists.

Of course, to apply this methodology to biological processes would require defining a new measure of similarity, based on the ones studied in CS, but more suited to biological processes. For instance, a biological measure must surely be more flexible than the cozy mathematical environments of CS. A biological measure must accommodate both a larger variability between compared processes (*e.g.* both "gaps" and "mismatches" should be allowed, context-independence may be relaxed) and a finer granularity of quantitative similarity scores (in place of the qualitative ones used in CS). The size, complexity and stochasticity of biomolecular systems also far exceed that of previously studied computational ones.

Once such a measure of similarity of processes is defined, we can use it to measure the homology between different systems with a shared evolutionary origin, such as the numerous 3-kinase MAPK modules. Although these systems may be structurally different, we can quantify how closely they behave, with respect to some observable feature. Such a measure can also be defined to be directed, *i.e.* show that one process is significantly similar to a second process, but does some more. Comparing several different processes in this manner can yield a kind of (partial) order between different processes, based on their behavioral properties. This order can illuminate the evolution of complex processes and their associated behavior from simpler ones, can serve as a basis for discovering the essential functional core of a family of pathways (a process "motif" or "signature") and can help to unravel the mechanisms and principles, which underlie the divergence of pathways from this core.

Alternatively, we may opt to study systems that perform similar functions (*e.g.* all are oscillators or all are switches) but have evolved independently. In this case, we measure the degree of analogy, rather than homology. This may lead to a more rigorous study of the analogous *design principles* [51] that underly systems that are functionally similar but are evolutionarily distinct.

Chapter 3

BioAmbients: An abstraction for biological compartments

3.1 Introduction

Compartments play an essential role in the organization and function of biomolecular systems. Many signal transduction pathways regulate transcription by re-localization of transcription factors between the cytoplasm and the nucleus, and shut-off or reconstitute by internalizing and recycling their receptors [78]. Insulation of certain signaling components from one another or tethering them together to a single supra-molecular complex determines which signals may or may not pass under certain conditions (*e.g.* [76]). Cellular metabolism is similarly dependent on compartmentalization and localization of enzymes and metabolites. Many metabolic events are bounded by membranes. Some, like peroxisomal oxidation, cannot be carried out in typical cytoplasmic conditions or may be deleterious for other cellular components. Others, like oxidative phosphorylation, depend on the formation of electro-chemical potential across membranes. In addition, the balance of ions and small molecules depends on their passive and active transport into the cell and between cellular compartments. Finally, in multicellular organisms, the function of many systems depends on the sequestration of different components in separate cells, histological components, and even organs. This is the case in systems as diverse as Notch signaling in boundary formation in *Drosophila*'s large intestine [81], and the hypothalamic system regulating body weight [121].

Compartments introduce modularity and hierarchy to the organization of biomolecular systems. As we have seen in the previous chapter, a modular and hierarchical approach for the representation of biomolecular systems captures essential principles of biological organization [51], and provides a single framework for molecular and functional studies, integrating variable levels of knowledge and detail within one model.

Here, we broaden this approach, by developing an extension of stochastic π-calculus that provides a better abstraction of compartmentalization. The extended abstraction allows us to study various hierarchical compartments of different granularities, movement of molecules between compartments, and dynamic rearrangement of cellular compartments and molecular complexes. We show how the resulting *BioAmbients Calculus* facilitates the modeling of complex molecular systems in a cellular and multicellular context. We implement BioAmbients as part of the BioSpi system to provide a fuller modular

[0]The work in this chapter will be published in [108] (manuscript in preparation).

framework for molecular interaction, localization and compartmentalization.

3.1.1 Previous work

Although compartmentalization is critical to the function of molecular networks, it has not been adequately treated in most existing formal models. Previous work can be roughly divided to three categories.

- **Ontologies and data schemas** based on the "compartment-as-object" abstraction. These are designed for genome and pathway databases and often represent cellular and sub-cellular compartments by a comprehensive hierarchy of objects. For example, the Gene Ontology (GO, [10]), employed by most genome databases, has an elaborate hierarchical vocabulary which encompasses both sub-cellular compartments and molecular complexes and machines. Pathway databases such as BIND [11], TRANSPATH [151], GENIES [38], INTERACT [32] and others all incorporate some compartment hierarchy and localization information. Naturally, however, this information is not dynamic and thus fails to reflect movement of and between compartments.

- ***Ad hoc* kinetic models with a compartment component.** Most kinetic models of specific systems either neglect this aspect entirely or form *ad-hoc* solutions for the problem, with highly specialized models (*e.g.* [52]). These may provide an efficient solution to specific problems, but are not easily generalizable, and do not constitute a rigorous framework.

- **Models based on abstract process languages** use the "compartment-as-process" abstraction. Two notable examples are Matsuno *et al.*'s [81] hybrid Petri net model of Notch signaling in *Drosophila* and Kam *et al.*'s [64] Statecharts models of the immune system and *Caenorhabditis elegans* vulva development. In the former, a stochastic Petri net abstraction is extended with an additional layer, representing cells. In the latter, compartmentalization and localization are an integral part of a qualitative process model. In both cases, however, abstraction of movement of and between compartments is either limited or completely absent.

In the π-calculus based approach presented in Chapter 1 we employ a "compartment as private channel" abstraction. Our underlying assumption is that compartmentalization can be abstracted by communication scope. Thus, all the processes representing molecules in one compartment or molecular complex, share certain exclusive communication capabilities (private channels), that are inaccessible to processes representing molecules outside this compartment. The limited scope of the private channel represents the boundary of the corresponding compartment. Both movement of molecules between compartments and formation of complexes were represented by mobility of private channels.

This approach allowed us to abstract compartments within the same mathematical domain as other entities of biomolecular systems. The "private channel as compartment" was a relevant abstraction and allowed uniform treatment of both sub-cellular compartments and complexes. For example, it allowed us to represent the limited interactions of a single molecule or a molecular complex as a combination of multiple public and private channels.

However, while essentially correct, the private channel based approach is often impractical. Essential events, such as the movement of a molecule in or out of a compartment or the merger of two compartments, require highly elaborate encodings involving the multi-step propagation of large sets of private channels between many processes. This is due to the fact that "compartments" are only derived from the private

channel distribution between different processes all existing at the same level. To rigorously address this problem we need to extend the mathematical domain of the stochastic π-calculus with additional entities, that will correspond better to our biological notions. In this chapter we present this extension, called the BioAmbients calculus.

3.2 Abstracting compartments as ambients: an overview

Building an abstraction consists of three steps: informal organization of the knowledge in the real-world domain, selection (or development) of an appropriate mathematical domain, and designing the abstraction between the two.

3.2.1 The real world domain: Essential properties of biomolecular compartments

We identify two types of compartmentalization in cellular and molecular systems (Table 3.1). **Membrane-bound compartments** (Figure 3.1A) including cells, organelles, and vesicles, have a clearly defined boundary, which insulates the compartment's components from the external environment. Membrane bound-compartments are hierarchically organized, *e.g.* organlles and vesicles residing within cells.

Compartment	Compartment movement	Molecule movement
Cells	Cell movement, phagocytosis	Cross-membrane molecules Molecule entry and exit
Organelles and vesicles	Merging, budding, bursting	Cross-membrane molecules Molecule entry and exit
Complexes, scaffolds, and enzymes	Complex formation and breakage; Movement in and out of membrane-bound compartment	Binding and unbinding to scaffolds and enzymes

Table 3.1: **Compartments in biomolecular systems.** Types of compartmentalization in molecular and cellular systems, their respective movement capabilities, and related movement of molecules.

Molecular compartments (Figure 3.1B) are equally important. They include stable or transient multi-molecular complexes, sometimes involving a mediating scaffold or enzyme. The formed complex insulates the component molecules from the environment to a certain extent, which may vary between different types of complexes. Single multi-domain proteins may sometimes be considered as a compartment in their own right, where the molecular backbone of the protein partly insulates the various domains from their environment. We can therefore refer to a hierarchy of molecular compartments, where molecules (one level) form complexes (a second, higher level). While the boundary of a molecular compartment is more fuzzy than that of a membrane bound one, this type of compartmentalization is very important [76].

Compartments introduce a notion of location (Figure 3.1C). Most entities in the system (*i.e.* molecules or compartments) may be either within or outside a given compartment. In addition, **cross-compartment molecules**, such as cross-membrane channels, receptors and transporters reside across a boundary and belong to two compartments. The two compartments are not symmetrical. Since these molecules are membrane-linked, they primarily belong to one of the compartments, and would move together with it.

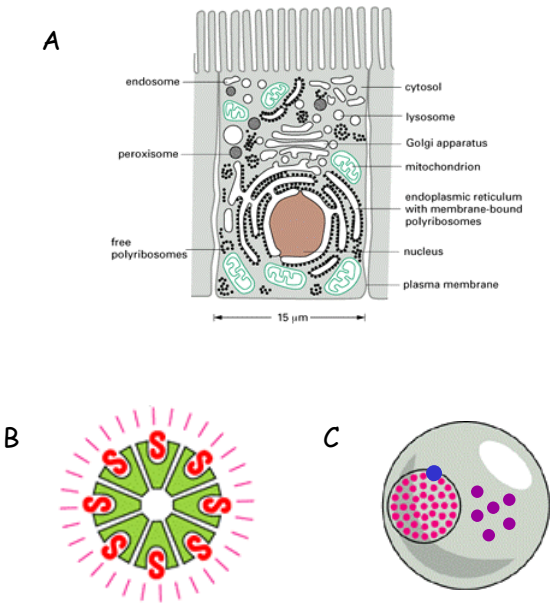

Figure 3.1: **Biological compartments.** A. Membrane bound compartments in a higher eukaryotic cell. B. A protein complex is a molecular compartment. C. Location. A molecule may reside within (pink), outside (purple), or across (blue) the compartment boundary. Panels adopted from [5].

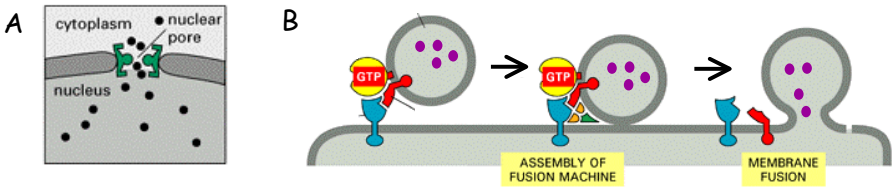

Figure 3.2: **Movement in biomolecular systems.** A. Movement of molecules across the nucleus envelope. B. Merge of a vesicle and a cell membrane. Adapted from [5].

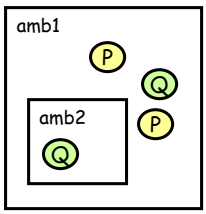

Figure 3.3: **Ambients and processes: An intuitive view.** Ambients (rectangles) are hierarchically organized in a nested structure. Processes (ovals) reside in ambients.

Entities may also change their location by movement between compartments (Figure 3.2). We identify two types of movement. **Movement between compartments** occurs when *e.g.* molecules move across a membrane, thereby entering or existing a membrane-bound compartment (*e.g.* the cell or the Golgi apparatus). Similarly, molecules may join a molecular compartment, for example by binding to a scaffold molecule.

Compartment movement occurs when an entire compartment moves with respect to the other compartments in the system. The most typical event is the merge of two membrane-bound compartments, in which two separate compartments become one, with their contents shared. In other cases, compartments may enter or exit one another, in events such as phagocytosis (cell "enters" cell) or entry of a complex molecule (a molecular compartment) into an organelle (a membrane-bound compartment).

3.2.2 The mathematical domain: Ambients

Compartmentalization and movement across boundaries play a critical role in computational systems, too [21]. In particular, the advent of the World-Wide Web has increased the potential for *mobile computation* that involves mobile devices (*e.g. laptops*), mobile code that moves between devices, and boundaries (*e.g. firewalls*).

To describe such organization Cardelli and Gordon [21] have developed the ambient calculus[1], an abstract process language for mobile computation, by extending the π-calculus "world" of processes with computational compartments, termed *ambients*. Ambients have defined boundaries and are hierarchically organized in nested structures. Computational processes are confined to these ambients (Figure 3.3).

An ambient may move as a whole with its component processes and sub-ambients. The processes inside the ambient control it, by providing movement capabilities, such as entry or exit. Ambient movement is synchronized between processes with complementary capabilities in two ambients using specific movement channels. Thus, for one ambient to *enter* another, a process in the entering ambient must synchronize an *enter* capability with an *accept* capability on the same channel in a process in a sibling (entered) ambient (Figure 3.4A). Analogous synchronization is required between a process offering an *exit* capability in one ambient, and a process in its parent ambient offering an *expel* capability (Figure 3.4B). Ambients may also *merge* by a synchronization between processes in sibling ambients (the complementary $merge+$ and $merge-$ capabilities, Figure 3.4C).

[1]The calculus presented here is an extension of Cardelli and Gordon's ambient calculus, termed *BioAmbients*, developed for the purpose of biomolecular modeling. To simplify the presentation, the original ambient calculus is not presented in detail. The essential differences between the calculi, the modifications we introduced, and their justifications are discussed in Section 3.8.

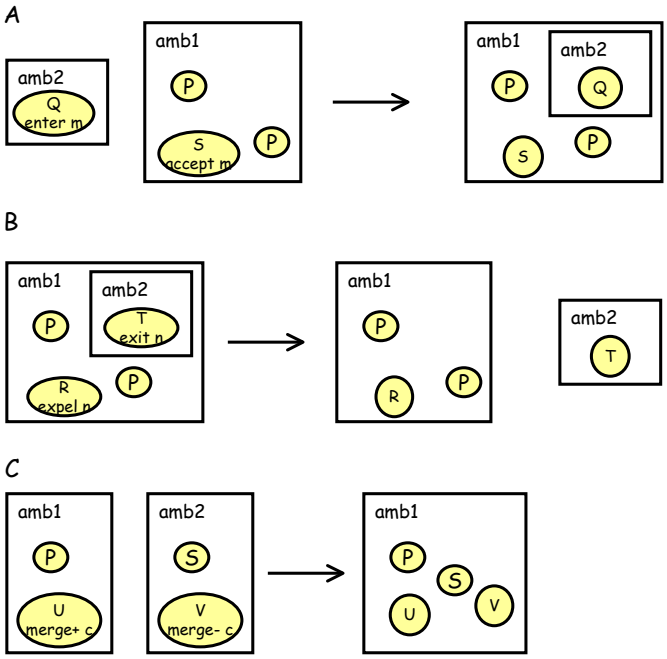

Figure 3.4: **Ambients capabilities: An intuitive view.** A. Ambient entry by complementary *enter* and *accept* capabilities on the same channel in processes in sibling ambients. B. Ambient exit by complementary *exit* and *expel* capabilities on the same channel in processes in a child and a parent ambient. C. Ambient merger by complementary *merge+* and *merge−* capabilities on the same channel in processes in sibling ambients. Ambients are shown as rectangles; processes as ovals.

Communication on channels between processes is restricted by the ambient boundary. There are three types of communication channels in the calculus, which differ in their scope in the ambient hierarchy (Figure 3.5). First, two processes in the same ambient can communicate on *local* communication channels. Second, two processes in sibling ambients can communication on *lateral* (*i.e.* "sibling-to-sibling", or $s2s$) channels. Third, a process in a parent ambient may communicate with a process in its child ambient using *inter* communication channels (a "parent-to-child" ($p2c$) and a "child-to-parent"($c2p$) channel, respectively). While *local* communication is fully restricted by the ambient boundary and cannot cross it, the $s2s$ and $p2c/c2p$ ones must cross it once in order to reach their targets.

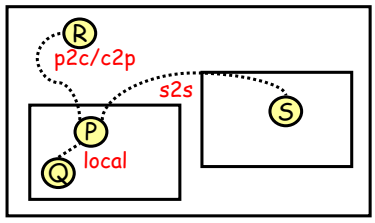

Figure 3.5: **Communication in the ambient calculus: An intuitive view.** Three types of process communication: local, sibling, and parent-child.

3.2.3 The "ambient as biological compartment" abstraction

The computational world represented by the ambient calculus bears an intuitive analogy to the biomolecular realm. In this section we present guidelines for an extension of the stochastic π-calculus, that abstracts biomolecular compartments as ambients (Table 3.2).[2] The abstraction is constructed in Section 3.3.

Biological entity	**Example**	**BioAmbients entity**	**Example**
Membrane bound compartment	Cell, vesicle, organelle	Ambient	The *cell*, *vesicle*, and *organelle* ambients
Molecular compartment	Multi-protein complex	Ambient	The *complex* ambient
Compartment location	Nucleus within the cell	Nested ambient hierarchy	The *nucleus* ambient inside the *cell* ambient
Molecular location inside compartments	Protein A and B in complex; Protein C inside vesicle	Processes reside within ambients	The $ProteinA$ and $ProteinB$ process inside the *complex* ambient; The $ProteinC$ process inside the *vesicle* ambient
Biological event	**Example**	**BioAmbients event**	**Example**
Compartment movement requires interaction between the moving compartment and its environment	Merge of a vesicle and an organelle	Ambient movement synchronized between the moving ambient and an ambient in its immediate environment	*merge* of a *vesicle* and *organelle* ambient on the $fuse$ channel
Molecule movement requires interaction between the moving molecule and a molecule in the relevant compartment	Entry of protein A into the nucleus	Ambient movement synchronized between the moving (molecular) ambient and the relevant (membrane bound or molecular) ambient	Entry of a *protein* ambient to the *nucleus* ambient. Synchronized between a $ProteinA$ process residing in the *protein* ambient and a $Porin$ process residing in the *nucleus* ambient
Intra-compartment interaction	Intra-complex interaction between proteins A and B	*local* communication, restricted within ambient boundaries	*local* communication offers on the *react* channel in processes $ProteinA$ and $ProteinB$
Inter-compartment interaction	Interaction between protein A in one complex and protein B in another	Communication between processes in sibling or parent-child ambients	$s2s$ communication on the *react* channel between the $ProteinA$ process in the $complex1$ ambient and the $ProteinB$ process in the $complex2$ sibling ambient

Table 3.2: **Guidelines for the abstraction of biomolecular systems to BioAmbients.**

We identify two types of compartmentalization in biomolecular systems: **membrane-bounded compartments** and **molecular compartments** (Figure 3.1). Both are abstracted as ambients. For example, a system composed a nucleated cell is abstracted as a *cell* ambient with a nested *nucleus* sub-ambient (Figure 3.6A). A molecular complex composed of two protein molecules, A and B, is abstracted as a

[2]Entities and events in the biomolecular realm are shown in **bold** and entities and events in the mathematical domain are shown in Sans Serif font.

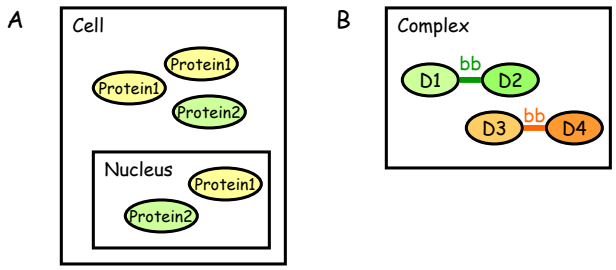

Figure 3.6: **Compartments as ambients.** A. A system of membrane bound compartments as nested ambients. B. A molecular complex as an ambient.

single ambient with two processes inside, *ProteinA* and *ProteinB* (Figure 3.6B).

The **molecular location** introduced by biological compartments (Figure 3.1C) is abstracted as process location in the ambient calculus, and **compartment location** is abstracted as ambient nesting. A notable exception is **cross-compartment molecules** which cannot be directly abstracted in the ambient calculus. These are discussed in Section 3.3.7.

Compartment movement is abstracted as ambient movement. For example (Figure 3.7A), the **merger** of a vesicle into a cell is abstracted as the merge of two ambients, one abstracting the cell, and the other abstracting the vesicle. Similarly, **compartment exit** and **entry** is abstracted by ambient enter and exit moves (Figure 3.7B). Just as compartment movement requires **interaction** between the two participating compartments, ambient movement is synchronized between processes in the two participating ambients. Importantly, **movement of molecules** between compartments is also abstracted as ambient movement (*e.g.* Figure 3.7C), and we do not allow "naked" processes to cross an ambient's boundary.

The **limitations on interaction between molecules** imposed by compartments are abstracted as limitation on communication between processes imposed by ambients. **Intra-compartment interaction** (*e.g.* between proteins A and B that both reside in the nucleus) is abstracted as *local* communication between processes that reside in the same ambient (*e.g.* *ProteinA* and *ProteinB* that reside in the *nucleus* ambient). The limit imposed on interaction between molecules that reside in different membrane bound compartments is thus abstracted by the limit imposed by the ambient boundary of local communication between processes that reside in separate ambients. Inter-ambient communications abstract cases where **inter-compartment interactions** are allowed (see Section 3.3).

3.3 BioAmbients: The cellular ambient calculus

To describe BioAmbients, we now turn to a series of examples, which we will abstract using the formal constructs and rules unique to BioAmbients.

3.3.1 Membrane-bound compartments as ambients

We abstract membrane-bound compartments as ambients. Each ambient is defined by a boundary and may have a name[3], *e.g.* $amb[\cdots]$. The ambient content includes either processes or additional ambients. For example, a system with several cells (Figure 3.8A), each with several molecules inside, is modeled by:

[3]Ambient names are not necessary and are only introduced for readability and convenience

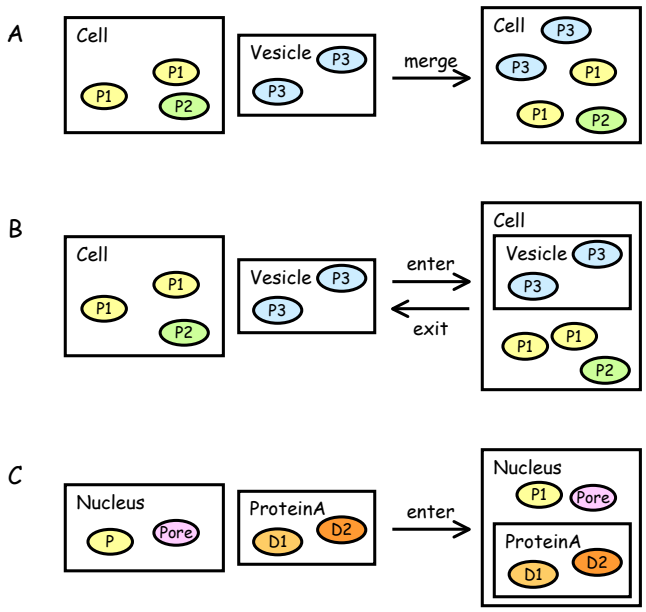

Figure 3.7: **Compartment and molecule movement as ambient movement.** A. Merge of membrane-bound compartments. B. Entry and exit of one membrane bound compartment into and from another. C. Entry of a molecule into a membrane bound compartment.

$$Sytem ::= cell[Molecule|\cdots|Molecule]|\cdots|cell[Molecule|\cdots|Molecule]$$

Compartments hierarchy is abstracted as ambient nesting. For example, a cell, which in addition to several molecules also has a nested nucleus compartment, is represented by:

$$Cell ::= cell[Molecule|\cdots|Molecule|nucleus[Molecule|\cdots|Molecule]]$$

Here, the nesting of ambient brackets abstracts the hierarchy of membrane-bound compartments (Figure 3.8B).

3.3.2 Membrane fusion as ambient merger

The most common change in membrane-bound compartments is their merger by membrane fusion. This is the case when vesicles fuse into the ER and Golgi apparatus, or when a virus enters a cell. Merger requires specific interaction between molecules in the two compartments (*e.g.* receptors). We abstract ambient merger by complementary *merge* capabilities in two processes in the two merging ambients.

$$cell[merge+\ n.Mol_1|\cdots|Mol_n]\ |\ ves[merge-\ n.VMol_1|\cdots|VMol_n]$$

The complementary $merge+\ n$ and $merge-\ n$ capabilities abstract the potential of the two compartments to merge. The specific channel name n represents the specificity of this interaction. Thus, a *cell* ambient

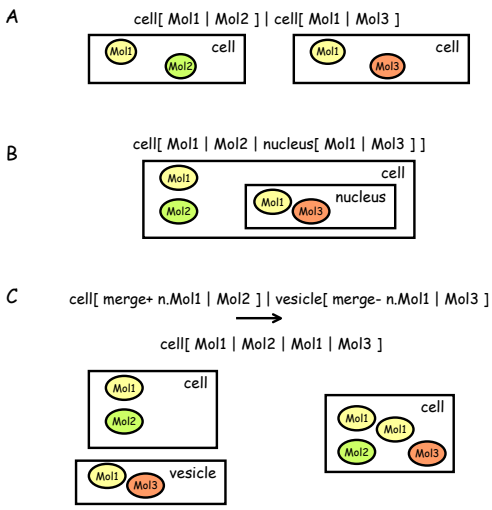

Figure 3.8: **Basic use of BioAmbients: membrane-bound compartments.** A. Membrane-bound compartments as ambients. B. Nested compartments as nested ambients. C. Membrane fusion as ambient merger.

may merge with various *vesicle* ambients by using different channel names in the *merge* operation.

As a result of compartment fusion, the organization of the system changes. The two compartments are unified and their contents are united together, wrapped by a single membrane. This is reflected by the result of the *merge* operation: The two ambients are unified to a single one, holding their entire contents (Figure 3.8C). For example:

$$cell[Mol_1|\cdots|Mol_n|VMol_1|\cdots|VMol_n]$$

Any internal structure (*i.e.* sub-ambients) of the merging ambients would be preserved by this operation, abstracting the preservation of sub-compartments (*e.g.* if a vesicle merges into a cell, the nucleus compartment remains intact in the merged cell-vesicle). Note, that only "sibling" compartments, not separated by a third membrane, may fuse to one another. Correspondingly, the abstracted *merge* operation is allowed only if the two ambients are siblings.

3.3.3 Compartment entry and exit as ambient entry and exit

Ambient entry and exit abstracts the entry and exit of membrane-bound compartments. An example for this relatively-rare biological situation is phagocytosis of a lymphocyte by a phagocyte. This interaction is mediated by a specific interaction between IgG molecules on a lymphocyte and Fc receptors on a phagocyte (Figure 3.9A). We abstract such a system as two ambients: *phagocyte* and *lymphocyte*. The F_cR process in the *phagocyte* ambient offers an *enter* n capability, complemented by an *accept* n capability in the IgG process in the sibling *lymphocyte* ambient:

$$lymphocyte[enter\ n.IgG|\cdots|LM_n|lnuc[LM_1|\cdots|LM_n]]\ |$$
$$phagocyte[accept\ n.F_cR|\cdots|PM_n|pnuc[PM_1|\cdots|PM_n]]$$

The result of an ambient *enter* operation abstracts the result of compartment entry: The entering ambient is now a part of the accepting ambient, and the entire internal content and structure of both ambients is preserved. This is the case with the *pnuc* sub-ambients of both the *lyphocyte* and *phagocyte* in our example (Figure 3.9B):

$$\begin{aligned} phagocyte[\quad & lymphocyte[IgG|\cdots|LM_n|lnuc[LM_1|\cdots|LM_n]] \quad | \\ & F_cR|\cdots|PM_n|pnuc[PM_1|\cdots|PM_n] \qquad\qquad] \end{aligned}$$

Note, that only sibling compartments may enter one another. Correspondingly, the abstracted *enter*/*accept* operation is allowed only if the two ambients are siblings. The exit operation works in an analogous way. However, as membrane-bound compartments rarely exit (they either bud out or fuse and spill their contents), we defer its description.

3.3.4 Molecular compartments as ambients

We abstract molecular compartments, such as multi-domain molecules and molecular complexes, as ambients, too. For example, a protein with three domains is abstracted as a *protein* ambient with three resident processes, each abstracting one domain (Figure 3.10A):

$$protein[(\mathsf{new}\ backbone)\ Domain_1|Domain_2|Domain_3]$$

Note, that as in the π-calculus, we may concomitantly abstract the molecular compartment using a private *backbone* channel. As we will presently see, this abstracts intra-molecular interaction once the molecule is part of a multi-molecular complex.

3.3.5 Molecule movement as ambient movement

The movement of molecules across membrane-bound compartments can now easily be abstracted as entry and exit of ambients (representing molecules) to and from ambients (representing membrane-bound compartments, Figure 3.10B,C). Molecule entry into a membrane-bound compartment typically requires interaction between the entering molecule and a molecule in the entered compartment, such as a receptor or a pore. This is abstracted by the need for an *enter* n (*exit* n) capability in a process in the entering (exiting) ambient and a complementary *accept* n (*expel* n) capability in a process in the entered (exited) ambient.

For example, consider a system where a two-domain protein molecule exits the nucleus by interaction between one of the protein's domains (D1) and a pore protein in the nucleus' envelope. We abstract the protein molecule as a *prot* ambient with two resident processes D_1 and D_2, and the nucleus as a *nucleus* ambient with a *Pore* process. The ambient exit is initiated by synchronization of an *exit nuc* capability in the D_1 process and an *expel nuc* capability in the *Pore* process (Figure 3.10C):

$$cell[M_1 \mid nucleus[\ expel\ nuc.Pore \mid prot[exit\ nuc.D_1 \mid D_2]]]$$

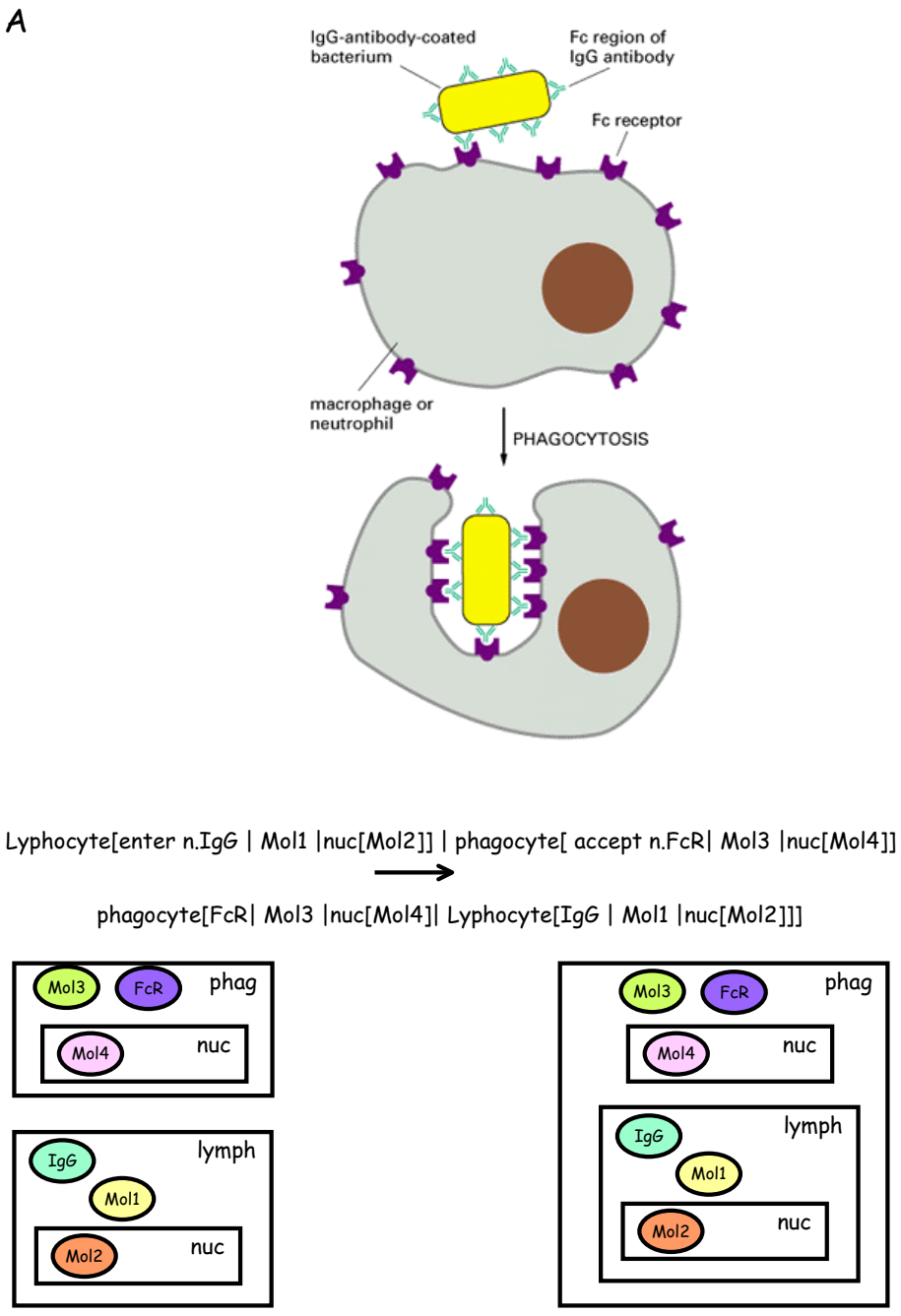

Figure 3.9: **Basic use of BioAmbients: entry of membrane-bound compartments.** A. Antibody activated phagocytosis [5]. B. A BioAmbients model of phagocytosis.

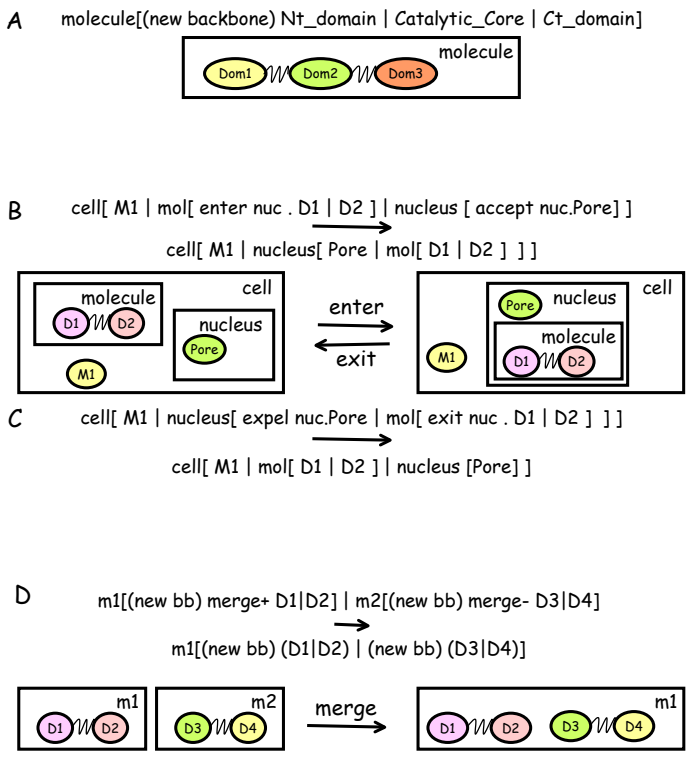

Figure 3.10: **Basic use of BioAmbients: Molecular compartments.** A. Molecular compartments as ambients. B,C. Molecular movement as ambient entry and exit, respectively. D. Complex formation as ambient merger.

The specific channel *nuc* is used to synchronize the exit event and represents the specificity of the interaction between the protein and the pore.

While only one domain of the protein may interact directly with the pore, the entire molecule moves *as a whole*. Similarly, while only one process initiates the ambient's move, the entire ambient moves as a whole, with all its contents (processes and sub-ambients). In our example, both D_1 and D_2 exit the *nucleus* ambient *as part of* the *prot* ambient (Figure 3.10C), resulting in:

$$cell[\ prot[D_1|D_2]\ |\ M_1\ |\ nucleus[Pore]\ ]$$

The same principle applies for the entry of a protein into the nucleus(Figure 3.10B) :

$$cell[\ prot[enter\ nuc.D_1|D_2]\ |\ M_1\ |\ nucleus[accept\ nuc.Pore]\ ]\ \ \rightarrow$$
$$cell[M_1\ |\ nucleus[\ Pore\ |\ prot[D_1|D_2]\ ]]$$

3.3.6 Complex formation as ambient merger

The content of molecular compartments may be pre-defined (*e.g.* several domains of a protein, a stable complex), but may also dynamically change (*e.g.* complex formation). We abstract the latter by the *merge* operation (Figure 3.10D). The specific channel used for *merge* abstracts the specificity of inter-

action. For example, consider the formation of a complex between two proteins, each with two domains. We abstract the two proteins as the $prot1$ and $prot2$ ambients, each with two sub-processes (D_1 and D_2 in the former and D_3 and D_4 in the latter). Their ability to interact is abstracted as specific interaction capabilities (a $merge+\ bind$ in D_1 and $merge-\ bind$ in D_3):

$$prot1[(\mathsf{new}\ bb)merge+\ bind.D_1|D_2]\ |\ prot2[(\mathsf{new}\ bb)merge-\ bind.D_3|D_4]$$

In forming a complex, while only one domain participates directly in the interaction, all the domains end up in the same complex. Similarly, upon ambient *merge* the resulting single ambient contains all the contents of the merging ambients. In our example, ambient merger would result in a single ambient with four processes:

$$prot1[(\mathsf{new}\ bb)(D_1|D_2)|(\mathsf{new}\ bb)(D_3|D_4)]$$

Note, that the two private backbone channels (one *bb* per each protein) are maintained throughout the *merge* operation, and abstract individual protein "identity" in the complex. We subsequently use them to abstract complex breakage, as shown in Section 3.5.2.

3.3.7 Compartment limitation on interaction as ambient restriction of communication

Membrane-bound and molecular compartments restrict molecular interaction. Molecules that reside within membrane-bound compartments may typically interact only with molecules in the same compartment. Molecular compartments also restrict some interactions (*e.g.* intra-molecular or intra-complex interactions) but not others (*e.g.* inter-molecular or inter-complex interactions). In certain cases, cross-membrane molecules may participate in interactions in two compartments.

We abstract the different restrictions on molecular interaction using three types of communication *directions* restricted by ambient boundaries: local, siblings, and parent-child. These are identified by preceding the usual π-calculus input and output prefixes by a direction label, *local*, $s2s$, and $p2c$ (or $c2p$) respectively.

Intra-compartment interaction as communication on local channels

Intra-compartment interaction is abstracted by the *local* communication direction, that allows interaction only between processes within the same ambient. For example, consider a two-domain molecule, in which a tyrosine motif in one domain is identified by a complementary motif in the other domain. If we abstract the molecule as a *mol* ambient with two processes, D_1 and D_2, then the potential intra-molecular interaction is represented by complementary *local* communication actions on the *tyr* channel (Figure 3.11A):

$$mol[local\ tyr\ !\ \{\cdots\}.D_1\ |\ local\ tyr\ ?\ \{\cdots\}.D_2]$$

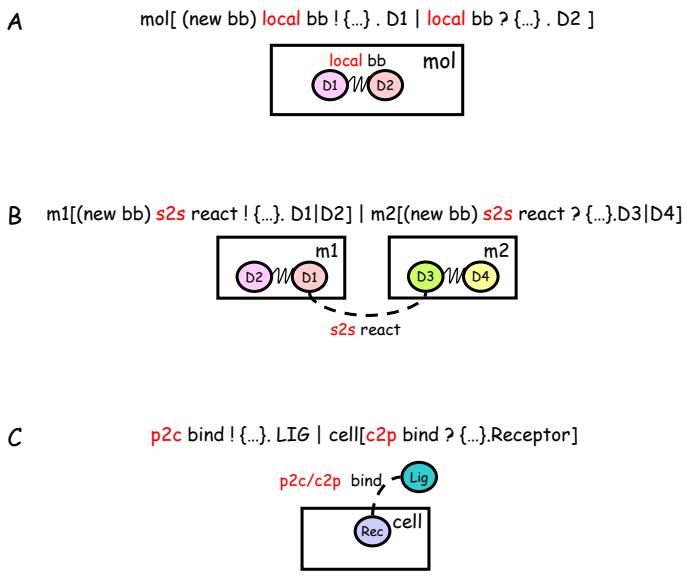

Figure 3.11: **Basic use of BioAmbients: Molecular interactions.** A. Intra-molecular interaction as local communication. B. Inter-molecular interaction as sibling communication. C. Transmembranal interaction as parent-child communication.

The local *tyr* channel allows communication between D_1 and D_2 (as they belong to the same ambient) but cannot be used by D_1 to interact with processes in sibling, parent, or child ambients. Note, that *tyr* is not a private channel. If another ambient merges with the *mol* ambient, then D_1 may use it to communicate with the added processes.

Inter-molecular interaction as communication on s2s channels

Interaction between molecular compartments is abstracted by the $s2s$ communication direction, that allows communication only between processes in sibling ambients. When we abstract complexes and multi-domain molecules by ambients, all inter-molecular interaction is abstracted by $s2s$ communication.[4] For example (Figure 3.11B), in

$$prot1[s2s\ react\ !\ \{\cdots\}.D_1|D_2]\ |\ prot2[s2s\ react\ ?\ \{\cdots\}.D_3|D_4]$$

the $s2s$ communication on the *react* channel abstracts the interaction between domain 1 (of protein 1) and domain 3 of (proteins 2).

Cross-membrane interaction as communication on p2c/c2p channels

Interaction of a membrane-embedded molecule (*e.g.* receptor) with a molecule outside its compartment (*e.g.* ligand) is abstracted by the $p2c$ and $c2p$ complementary directions. These allow interaction between a process in a child ambient and a process in a parent ambient. For example, for the binding of a ligand to a receptor (Figure 3.11C), we write:

[4] In practice, we take a hybrid approach when more complex molecules are abstracted as ambients while others are abstracted as naked processes. The parent-child communication ensures this approach's feasibility, as described below.

$$p2c\ bind\ !\ \{\cdots\}.Lig\ |\ cell[c2p\ bind\ ?\ \{\cdots\}.Receptor]$$

Note, that since we allow communication to cross at most one ambient boundary (no grandchild to grandparent communication), the receptor must be abstracted as a "naked" process and not as a molecular ambient. In general, the child-parent communication allows us to use a hybrid model, where some of the molecules are "naked", while others are wrapped with ambients boundaries. For example, consider a system where Molecule 1 (abstracted as the naked process $Mol1$) can interact with protein 1 (abstracted as the $prot1$ ambient). This interaction is abstracted as a parent→child directed communication on the *react* channel (on $Mol1$'s side) and as a child→parent directed complementary communication (on the $prot1$'s side)[5]:

$$p2c\ react\ !\ \{\cdots\}.Mol1\ |\ prot1[c2p\ react\ ?\ \{\cdots\}.D_1|D_2]$$

3.3.8 Stochastic semantics

To ensure that our abstraction is correct the dynamic behavior of our abstract models should reflect that of biochemical and cellular systems. As with the π-calculus, the original ambient calculus is non-deterministic, and all enabled capabilities and communications are equally likely to be "chosen" to occur next. We adapt the ambient calculus to a stochastic framework by extending our application of the Gillespie algorithm [42] to (movement) capabilities in addition to communications. Thus, at each state change in the system we choose one of the enabled events (either a communication or a capability) to happen next. The basic rate calculation rules for capabilities follow those that are defined for bi-molecular channels, and are based on the channel's base rate and the number of complementary offers. Each type of capability or communication direction is handled as a separate channel, so the same channel name may be used in different ways without violating the correct dynamics. This simple extension does not require any modification of the semantics presented in Chapter 2 and can be incorporated into the simulation system, as discussed below. Note, that since our simulation system allows us to incorporate any type of reaction dynamics by defining and using specific rate calculation rules per channel, we can associate different capabilities with appropriate dynamics as required.

3.4 BioAmbients: A formal presentation

The ambient calculus was originally developed by Cardelli and Gordon [21] as a paradigm for mobile computation. Our modified version of the ambient calculus, termed BioAmbients, facilitates the mapping of biological compartments as ambients. In this section we formally present the BioAmbients variant.

[5] Unlike *local* and *s2s* communication, this interaction is asymmetric, and cannot be replaced by:

$$c2p\ react\ !\ \{\cdots\}.Mol1\ |\ prot1[p2c\ react\ ?\ \{\}.D_1|D_2]$$

in which $Mol1$ directs a communication to a (non-existent) parent ambient, while D_1 directs a communication to a (non-existent) child.

n, m, p		names
$\$ \stackrel{\text{def}}{=}$		Directions
	$local$	Intra-ambient
	$s2s$	Inter-siblings
	$p2c$	Parent to child
	$c2p$	Child to parent
$M, N \stackrel{\text{def}}{=}$		Capabilities
	$enter\ n$	Synch entry
	$accept\ n$	Synch accept
	$exit\ n$	Synch exit
	$expel\ n$	Synch expel
	$merge+\ n$	Synch merge with
	$merge-\ n$	Synch merge into
$\pi \stackrel{\text{def}}{=}$		Actions
	$\$n!\{m\}$	Input action
	$\$n?\{m\}$	Output action
$P, Q \stackrel{\text{def}}{=}$		Processes
	$(\mathsf{new}\ n)P$	Restriction
	$\mathbf{0}$	Inaction (empty)
	$P\|Q$	Composition
	$!P$	Replication
	$[P]$	Ambient (membrane)
	π, P	Communication prefix
	M, P	Capability prefix
	$\pi_1, P\ +\ \pi_2, Q$	Communication choice
	$M_1, P\ +\ M_2, Q$	Capability choice

Figure 3.12: **BioAmbients: Syntax.** All capabilities and communications are synchronous, ambients are (practically) nameless. Communication is allowed within ambients and between sibling and parent-child ambients. Note, that replication is taken as a primitive instead of recursion, for simplicity. The implementation will employ recursive parametric definitions.

The biologist reader is referred to Section 3.5 for a continuation of the informal presentation. Since BioAmbients contains the stochastic π-calculus, we discuss only the additional entities and operations. The full syntax, congruence laws and semantics are given in Figures 3.12 and 3.13. We address the differences between the ambient calculus and BioAmbients and their motivation in Section 3.8.

3.4.1 Ambients

An *ambient* is a bounded place where computation happens. The boundary surrounding the ambient defines what is inside and what is outside it. Ambients may have names, but these are used to improve readability only, and have no functionality. Each ambient harbors a collection of processes, that reside and run directly within it. Ambients can be nested within other ambients, with each ambient having a collection of sub-ambients, with their content. The ambient can move as a whole, with its component processes and sub-ambients. The computations inside the ambient control it, by instructing it to move.

An ambient is written

$$n[P]$$

Structural congruence and process identity

$P \equiv P$	Struct Refl
$P \equiv P \ \Rightarrow \ Q \equiv P$	Struct Symm
$P \equiv Q \ , Q \equiv R \ \Rightarrow P \equiv R$	Struct Trans
$P \equiv Q \ \Rightarrow (\mathsf{new}\ n)P \equiv (\mathsf{new}\ n)Q$	Struct Res
$P \equiv Q \ \Rightarrow P\|R \equiv Q\|R$	Struct Par
$P \equiv Q \ \Rightarrow !P \equiv !Q$	Struct Repl
$P \equiv Q \ \Rightarrow [P] \equiv [Q]$	Struct Amb
$P \equiv Q \ \Rightarrow M.P \equiv M.Q$	Struct Action
$P \equiv Q \ \Rightarrow \$n?\{m\}.P \equiv \$n?\{m\}.Q$	Struct Input
$P \equiv Q \ \Rightarrow \$n!\{m\}.P \equiv \$n!\{m\}.Q$	Struct Output
$P\|Q \equiv Q\|P$	Struct Par Commut
$(P\|Q)\|R \equiv P\|(Q\|R)$	Struct Par Assoc
$P + Q \equiv Q + P$	Struct Choice Commut
$(P + Q) + R \equiv P + (Q + R)$	Struct Choice Assoc
$P\|\mathbf{0} \equiv P$	Struct Par Zero
$(\mathsf{new}\ n)\mathbf{0} \equiv \mathbf{0}$	Struct Res Zero
$(\mathsf{new}\ n)(\mathsf{new}\ m)P \equiv (\mathsf{new}\ m)(\mathsf{new}\ n)P$	Struct Res Res
$(\mathsf{new}\ n)(P\|Q) \equiv P\|(\mathsf{new}\ n)Q$ if $n \notin fn(P)$	Struct Res Par
$(\mathsf{new}\ n)[P] \equiv [(\mathsf{new}\ n)P]$	Struct Res Amb
$(\mathsf{new}\ n)\$m?\{p\}.P \equiv \$m?\{p\}.(\mathsf{new}\ n)P$ if $n \neq m, n \neq p$	Struct Res Input
$(\mathsf{new}\ n)\$m!\{p\}.P \equiv \$m!\{p\}.(\mathsf{new}\ n)P\{p \leftarrow p'\}$ if $n \neq m, p' \notin fn(P)$	Struct Res Output
$\$n?\{m\}.P \equiv \$n?\{p\}.P\{m \leftarrow p\}$ if $p \notin fn(P)$	Renaming bound names
$(\mathsf{new}\ n)P \equiv (\mathsf{new}\ m)P\{n \leftarrow m\}$ if $m \notin fn(P)$	Renaming bound names
$!\mathbf{0} \equiv \mathbf{0}$	Struct Repl Zero
$!P \equiv P\|!P$	Struct Repl Par

Reduction rules

$[enter\ n.P\|Q]\|[accept\ n.R\|S] \rightarrow [[P\|Q]\|R\|S]$	Red In
$[[out\ n.P\|Q]\|expel\ n.R\|S] \rightarrow [P\|Q]\|[R\|S]$	Red Out
$[merge+\ \ n.P\|Q]\|[merge-\ \ n.R\|S] \rightarrow [P\|Q\|R\|S]$	Red Merge
$local\ n!\{m\}.P\|local\ n?\{p\}.Q \rightarrow P\|Q\{p \leftarrow m\}$	Red Comm Local
$p2c\ n!\{m\}.P\|[c2p\ n?\{p\}.Q] \rightarrow P\|[Q\{p \leftarrow m\}]$	Red Comm Parent Output
$[c2p\ n!\{m\}.P]\|p2c\ n?\{p\}.Q \rightarrow [P]\|Q\{p \leftarrow m\}$	Red Comm Parent Input
$[s2s\ n!\{m\}.P]\|[s2s\ n?\{p\}.Q] \rightarrow [P]\|[Q\{p \leftarrow m\}]$	Red Comm Sibling
$P \rightarrow Q \ \Rightarrow \ (\mathsf{new}\ n)P \rightarrow (\mathsf{new}\ n)Q$	Red Res
$P \rightarrow Q \ \Rightarrow \ [P] \rightarrow [Q]$	Red Amb
$P \rightarrow Q \ \Rightarrow \ P\|R \rightarrow Q\|R$	Red Par
$P \equiv P',\ P \rightarrow Q,\ Q \equiv Q' \ \Rightarrow \ P' \rightarrow Q'$	Red $\equiv$ (Struct)

Figure 3.13: **BioAmbients: Structural congruence and operational semantics.** Rules for synchronous movements and ambient-restricted communication replace the communication-only rules. Stochastic semantics follows the same rules as in the biochemical stochastic π-calculus (not shown).

where n is the (optional) name of the ambient, and P is the process running inside the ambient. In $n[P]$, P is actively running and can be the parallel composition of several processes.

The ambient tree hierarchy is represented by the nesting of ambient brackets. Each node of the tree may contain both non-ambient processes running in parallel and sub-ambients. A general description of an ambient with p content processes and q sub-ambients will be written as

$$n[P_1|\cdots|P_p|m_1[\cdots]|\cdots|m_q[\cdots]] \quad (P_i \neq n_j[\cdots])$$

An ambient named n, with two component processes P and Q, and one sub-ambient named m is shown in Figure 3.14A.

3.4.2 Ambient mobility: Capabilities

Capabilities can change the ambient hierarchy by allowing ambient entry, exit, or merger. All capabilities are synchronized. For example, for an ambient to enter into another ambient, the entered ambient must accept it. Synchronization is done on named channels and follows similar reduction rules to those used in the π-calculus. There are three pairs of capabilities, each representing a different type of synchronized movement.

- The *enter/accept* capability pair is required for one ambient to enter a sibling accepting ambient. A process with the *enter* c capability is present in the entering ambient (*e.g.* $enter\ c\ .\ P$), and a process with the complementary *accept* c capability is present in the accepting ambient (*e.g.* $accept\ c\ .\ R$). If several siblings with the complementary capability exist, any one of them may be chosen. Once the capability is exercised, the processes continue as P and R, and alternative *capability choices* are discarded. The reduction rule is

  $$m[\cdots + enter\ ch\ .\ P \mid Q] \mid n[\cdots + accept\ ch\ .\ R] \rightarrow n[\ m[\ P \mid Q] \mid R\ ]$$

  representing the transformation of a sibling m of ambient n into a child of n. This reduction is graphically depicted in Figure 3.14B.

- The *exit/expel* capability pair is required for an ambient to exit its parent (expelling) ambient. A process with the *exit* c capability is present in the exiting ambient (*e.g.* $exit\ c\ .\ P$), and a process with the complementary *expel* c capability is present in the expelling parent ambient (*e.g.* $expel\ c\ .\ R$). Once the capability is exercised, the processes continue as P and R, and alternative *capability choices* are discarded. The reduction rule is

  $$n[\ m[\cdots + exit\ ch\ .\ P \mid Q\ ] \mid \cdots + expel\ ch\ .\ R] \rightarrow n[R] \mid m[\ P \mid Q] \mid R\ ]$$

  representing the transformation of a child ambient m of ambient n into a sibling of n. This reduction is graphically depicted in Figure 3.14C.

- The *merge+ /merge−* capability pair is required for one ambient to merge with another (sibling) ambient. A process with the *merge+* c capability is present in one ambient (*e.g.* $merge+\ c\ .\ P$),

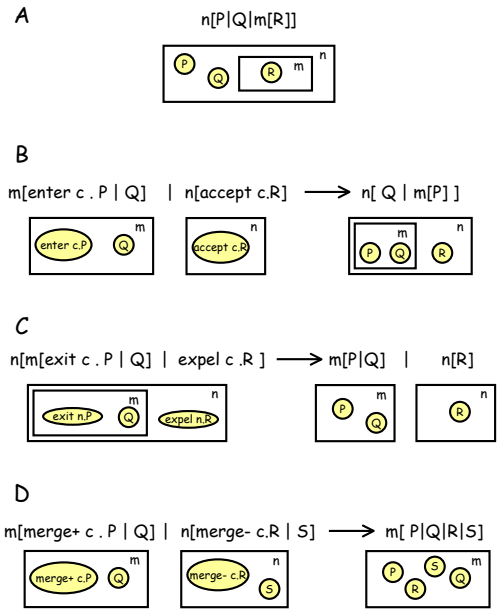

Figure 3.14: **Ambient moves.** A. Ambient n (child) inside ambient m (parent). B. Entry of ambient n into (sibling) ambient m. C. Exit of ambient n out of (parent) ambient m. D. Merge of sibling ambients m and n.

and a process with the complementary $merge-\ c$ capability exists in the sibling ambient (*e.g.* $merge-\ c\ .\ R$). Once the capability is exercised, the processes continue as P and R, and alternative *capability choices* are discarded. The reduction rule is

$$m[\cdots + merge+\ ch\ .\ P \mid Q\ ] \mid n[\ \cdots + merge-\ ch\ .\ R|S] \rightarrow m[\ P \mid Q \mid R \mid S\ ]$$

representing the merger of the two sibling ambients n and m into a single ambient. This reduction is graphically depicted in Figure 3.14D.

3.4.3 Communication in BioAmbients

The semantics of communication from the π-calculus is modified in BioAmbients, to reflect the restriction imposed on communication by ambient boundaries. We distinguish three communication *directions*:

- **Local communication** is denoted by the *local* direction label. For two processes to participate in local communication they must both reside in the same (immediate) ambient. One must offer to *locally* send along the channel and the other must offer to *locally* receive along the same channel. The reduction axiom is thus

$$\cdots + local\ n\ !\ \{m\}\ .\ P \mid local\ n\ ?\ \{p\}\ .\ Q\ +\ \cdots\ \rightarrow\ P \mid Q\{p \leftarrow m\}$$

Note, that no ambient boundaries are crossed in this reduction rule.

- **Sibling communication** is denoted by the $s2s$ direction label. For two processes to participate in sibling communication they must reside in two (immediate) sibling ambients. One must offer to send to a *sibling* along the channel and the other must offer to receive from a *sibling* along the same channel. The reduction axiom is thus

$$[\cdots + s2s\ n\ !\ \{m\}\ .\ P\ ] \mid [\ s2s\ n\ ?\ \{p\}\ .\ Q\ +\ \cdots] \rightarrow [\ P\ ] \mid [\ Q\{p \leftarrow m\}\ ]$$

  allowing the communication to cross the ambient boundary in a specific, pre-defined way.

- **Parent-child communication** is denoted by the $p2c$ label for the parent-to-child direction, and the $c2p$ label for the child-to-parent direction. For two processes to participate in parent-child communication they must reside in (immediate) parent-child ambients. Unlike the previous two cases, communication here is *asymmetric*. In one case a process in the *parent* offers to send to a *child* ambient, while a process in the *child* ambient offers to receive from the *parent*. The reduction axiom for this case is:

$$\cdots + p2c\ n\ !\ \{m\}\ .\ P \mid [\ c2p\ n\ ?\ \{p\}\ .\ Q\ +\ \cdots] \rightarrow P \mid [\ Q\{p \leftarrow m\}\ ]$$

  Alternatively, the sender and receiver roles may be reversed, with the process in the *parent* offering to receive from a *child* ambient, while a process in the *child* ambient offers to send to the *parent*:

$$[\cdots + c2p\ n\ !\ \{m\}\ .\ P\ ] \mid p2c\ n\ ?\ \{p\}\ .\ Q\ + \cdots \rightarrow [\ P\ ] \mid Q\{p \leftarrow m\}$$

These four configurations are summarized in Figure 3.15. Note, that directions are independent of the channel's identity as a public or private channel or from its use for asymmetric, symmetric, and instantaneous communication. These distinctions are kept in combination with the channel's direction. For example, the same private channel *name* may be used for communication between siblings (when both own the same private name) as well as for local communication between processes in the same ambient.

The communication axioms are rigid, and do not apply when we change the order of the sender and the receiving process or of the summed terms. Additional rules (Figure 3.13, Red Res, Red Par, Red Amb, and Red Struct) add this flexibility, by allowing reduction under restriction and parallel composition, inside ambients, and of structurally congruent terms.

Finally, the semantics of communication is *stochastic* following the same rules as in the stochastic π-calculus. Importantly, each combination of channel, direction, and configuration is considered as representing a separate "reaction" in the Gillespie algorithm. For example, if a channel name x is used for local communication in three separate ambients, it represents three separate reactions for which an actual rate is calculated and which "competes" on the next time slot to occur.

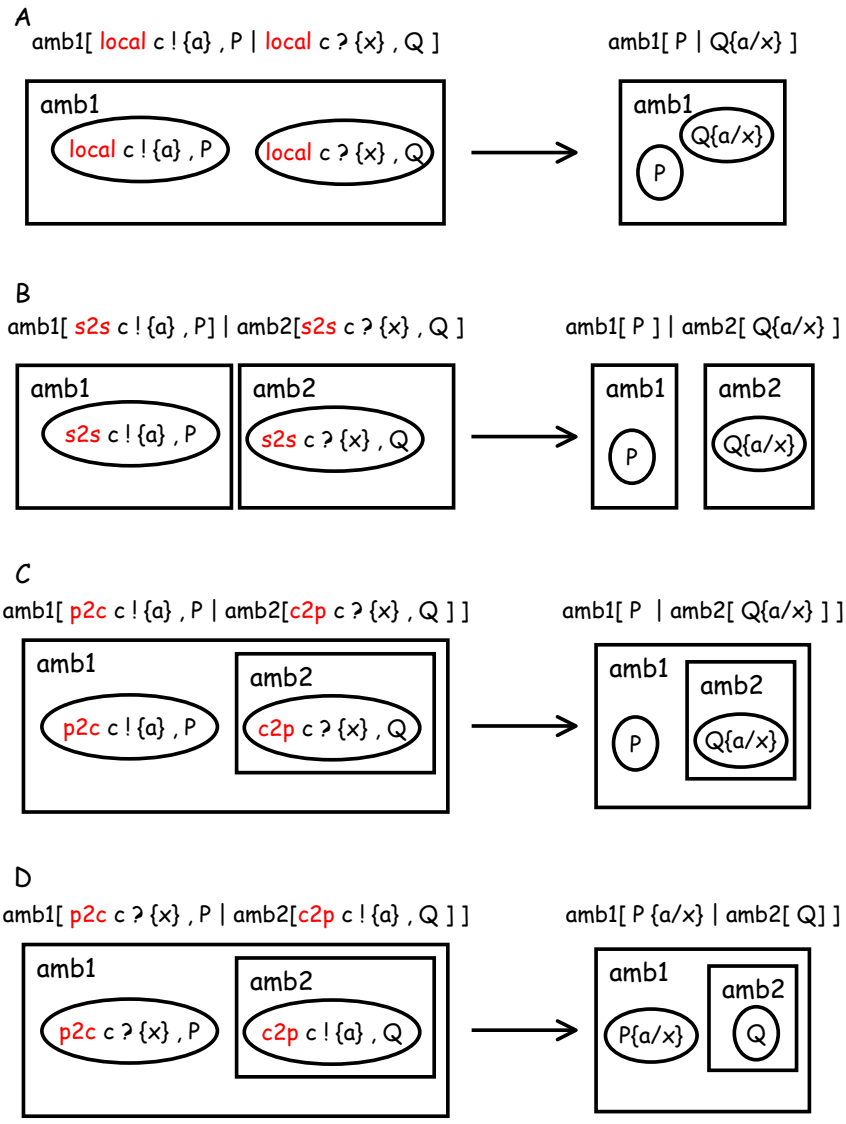

Figure 3.15: **Communication with ambients.** A. *local* communication between two processes in the same ambient. B. *s2s* communication between processes in two sibling ambients. C. *p2c/c2p* communication from a process in a parent ambient to a process in a child ambient. D. *c2p/p2c* communication from a process in a child ambient to a process in its parent ambient.

```
System::= molecule[Mol] | ... | molecule[Mol] | cell[Porin] .

     Mol::= enter cell1, Mol ;
            exit  cell2, Mol .

     Porin::= accept cell1 , Porin ;
              expel  cell2 , Porin .
```

Figure 3.16: **BioAmbients code for the porin example.**

3.5 Simple examples: Transport, enzymes and complexes

To illustrate the "compartment as ambient" abstraction, we now follow a few simple BioAmbients programs representing biological systems.

3.5.1 Transport

Our first example is a membranal pore, which allows bi-directional passage of molecules across a membrane. Passage depends on specific interaction between the molecule and the pore.

We abstract the cell and each of the molecules as ambients ($cell[\cdots]$ and $molecule[\cdots]$, respectively). To represent the molecule's ability to pass through the pore, we equip each *molecule* ambient with a process (Mol), with *choice* between an *enter* and an *exit* capability, synchronized on the $cell1$ and $cell2$ channels respectively. The complementary ability of the pore to let the molecule through, is represented by the complementary choice in a $Porin$ process to *accept* and *expel* on the corresponding channels. Note, that the membrane-embedded porin is abstracted as a naked process residing within the *cell* ambient, while other molecules (that are either properly in or out of the cell) are abstracted as processes encapsulated in ambients. The resulting code is shown in Figure 3.16.

An illustration of the operation of a toy system representing one molecule, one cell and one pore is shown in Figure 3.17. We start when the *molecule* ambient is outside the *cell* ambient. The *molecule* enters the *cell* by a synchronized $enter - accept$ on$cell1$ between the Mol and $Porin$ in the sibling ambients (the $exit - expel$ option is irrelevant at this point, as it requires one ambient to reside within the other). As a result, the system now consists of the *molecule* ambient within the *cell* ambient and the $enter - accept$ option is no longer relevant. Rather, the $exit - expel$ capability on $cell2$ can be used by $Porin$ in the parent *cell* and Mol in the child *molecule*, to pass the *molecule* outside the *cell*. Note, that this process can iterate forever.

A similar example (Figures 3.18 and 3.19) abstracts a symporter, which allows two different types of molecules to pass through a membrane in a unidirectional and sequential manner. As in the porin example, the molecules and the cell are abstracted as ambients, the former with $Mol1$ and $Mol2$ processes and the latter with $SymporterIn$ and $SymporterOut$ processes. The potential of each molecule to enter or exit via the symporter is abstracted by a choice constructed in the Mol processes between *enter* and *exit* capabilities on specific channels ($cell1$, $cell2$, $cell3$, $cell4$). Each of the symporter processes ($SymporterIn$ and $SymporterOut$) allows passage of the *molecule* ambients in one direction only. To represent the required sequential passage of the two molecules through the symporters, we compose the *accept* and *expel* capabilities in sequence. Thus, for example, $SymporterIn$ can first *accept* the first *molecule* on $cell1$ and only then will *accept* the second *molecule* on $cell3$.

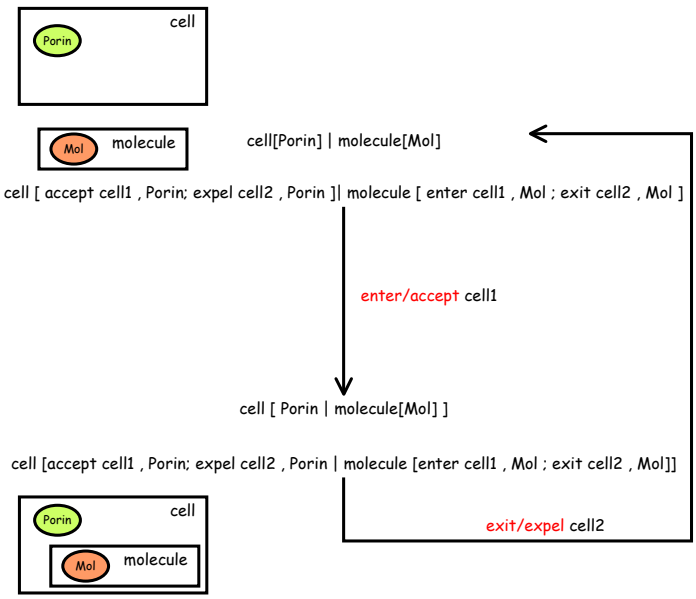

Figure 3.17: **Porin example.** An illustration of the BioAmbients reduction steps representing entry and exit of a molecule to a cell via a pore.

```
System::= molecule[Mol1] | ... | molecule[Mol1] |
          molecule[Mol2] | ... | molecule[Mol2] |
          cell[SymporterIn | SymporterOut] .
Mol1::= enter cell1, Mol1 ;
        exit  cell2, Mol1 .
Mol2::= enter cell3, Mol2 ;
        exit  cell4, Mol2 .
SymporterIn::= accept cell1 , accept cell3, SymporterIn .
SymporterOut::= expel cell2 , expel cell4, SymporterOut .
```

Figure 3.18: **BioAmbients code for the symporter example**

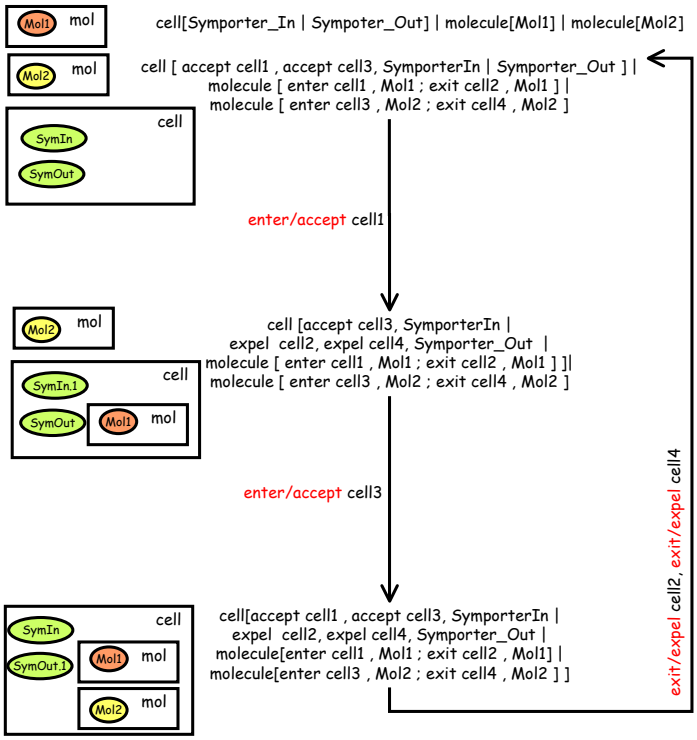

Figure 3.19: **Symporter example: Standard scenario.** An illustration of the BioAmbients reduction steps representing coordinated sequential entry and exit of two molecules via a symporter.

This use of sequential capabilities is similar to the use of sequential communications and ensures that an ambient carrying $Mol2$ can only enter (exit) after an ambient carrying $Mol1$ has already entered (exited) the cell. This is illustrated in a toy example abstracting two molecules, a cell and a symporter (Figure 3.19).

Note, that Figure 3.19 shows only one of several possible scenarios. Since the system is concurrent, the two *molecule* ambients are not directly dependent. Thus, once the ambient with $Mol1$ entered it can exit the cell, by interaction with $SymporterOut$, even if $Mol2$'s ambient has not entered yet. $SymporterIn$ however, does not allow a second consecutive passage of $Mol1's$ ambient, until $Mol2's$ ambient passes through. One such alternative scenario is depicted in Figure 3.20, showing that the system would still operate correctly. If we want to disallow such events, we must introduce more detail into the model to ensure that the entry (exit) events are not only sequential, but also coupled. This can be done with the use of private channels (not shown).

3.5.2 Protein complexes

We use private channels to abstract complex breakage. As shown above, the formation of a multi-protein complex is easily abstracted as ambient merger. As a result, a single ambient forms where the resident processes represent the domains of different proteins. This allows us to abstract intra- and inter-complex interactions as local and sibling communication, respectively. However, it also introduces a difficulty when we wish to abstract the reverse event of complex breakage: how can we identify the individual domains of one protein in the abstracted representation when the processes that represent them are no

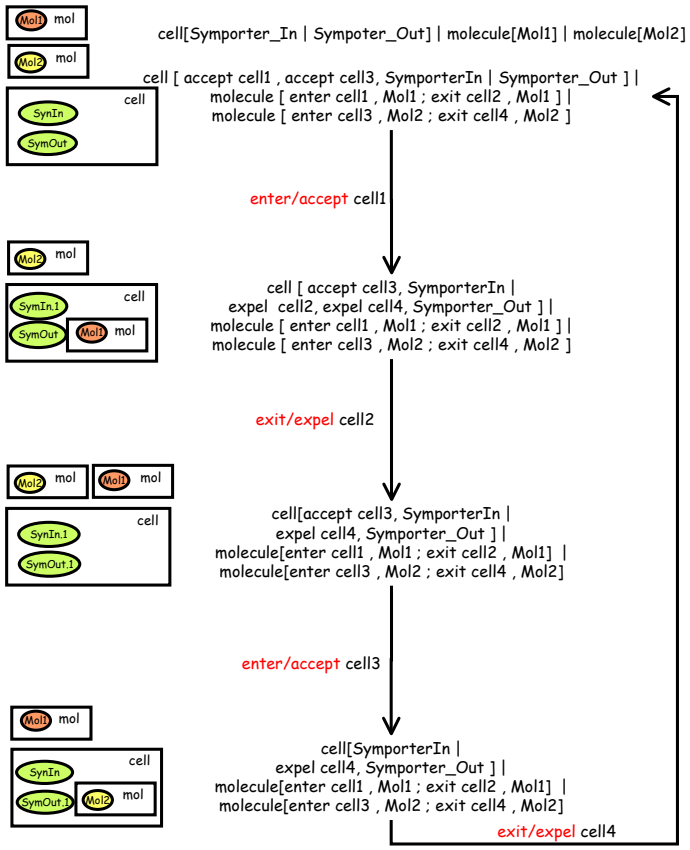

Figure 3.20: **Symporter example: Alternative scenario.** An illustration of the BioAmbients alternative reduction steps in the symporter examples, stemming from the model's concurrent nature.

```
System::= molecule[ProteinA] | ... | molecule[ProteinA] |
          molecule[ProteinB] | ... | molecule[ProteinB] .
ProteinA::= D3.
  D3::= merge- complexAB, BoundD3 .
  BoundD3::= local breakAB ? {...}, expel breakAB1, D3 .
ProteinB::= (new bb) D1 | D2 .
  D1::= merge+ complexAB , BD1 .
  BD1::= local breakAB ! {...} , local bb ! {...} ,
              molecule[merge+ bb , exit breakAB1, D1] .
  D2::= local bb ? {...} , molecule[merge- bb, D2] .
```

Figure 3.21: **BioAmbients code for a two-protein complex**

longer encapsulated by an exclusive ambient boundary?

There are two solutions to this problem. First, we can use private channels to sustain a specific link between the processes representing independent domains of a single protein, similar to the approach we applied before introducing ambients. For example (Figures 3.21 and 3.22) consider a complex formed between one protein molecule ($ProteinA$ process) with a single domain ($D3$ process) and a second molecule ($ProteinB$) comprised of two domains ($D1$ and $D2$). First, we represent complex formation by exercising complementary *merge* capabilities in $D3$ and $D1$ on the $complexAB$ channel (Figure 3.22A). As a result, $D1$, $D2$ and $D3$ all reside within the same ambient, representing the complex. In order to distinguish the processes according to their origin, we employ a private backbone channel *bb*. This private channel is declared when $ProteinB$ is originally created, and is thus known only to its two sub-processes, $D1$ and $D2$, but not to $ProteinA$'s $D3$.

The private channel will be used when abstracting complex breakage. The event is initiated by a local communication on $breakAB$ between $BD3$ and $BD1$ representing the directly bound domains (Figure 3.22B). In the next three steps, $BD1$ uses the private *bb* channel to coordinate the simultaneous exit of itself and $D2$ as a single ambient. First, the event is propagated from $BD1$ to its companion $BD2$, by a local communication on *bb*. This is followed by separate encapsulation of each of the two processes (Figure 3.22C). The two encapsulated processes then use the same private *bb* channel name to merge and form a single ambient, embedded within the original complex ambient (Figure 3.22D). Finally, an $exit - expel$ capability on the $breakAB1$ channel between $BD1$ and $BD3$ completes complex breakage resulting in two sibling ambients, one per protein (Figure 3.22E).

Note, that as we are using a multi-step scenario to model an "atomic" event, we must ensure its biochemical correctness by assigning appropriate channel rates. With the exception of the $breakAB$ channel (used to initiate the break), all the "intermediate" channels (*bb*, $breakAB1$) are instantaneous and do not affect the time evolution of the system. Rather, the rate of the entire scenario is determined by the rate associated with the $breakAB$ channel.

3.5.3 Enzymes

An alternative way to abstract complexes is to replace ambient merger by entry of one ambient into another. This representation, albeit simpler, is inherently asymmetric, and creates a double barrier between the nested ambient and the general environment (its own ambient boundary and that of its including partner). Thus, we use this approach mostly to abstract transient complexes, where one

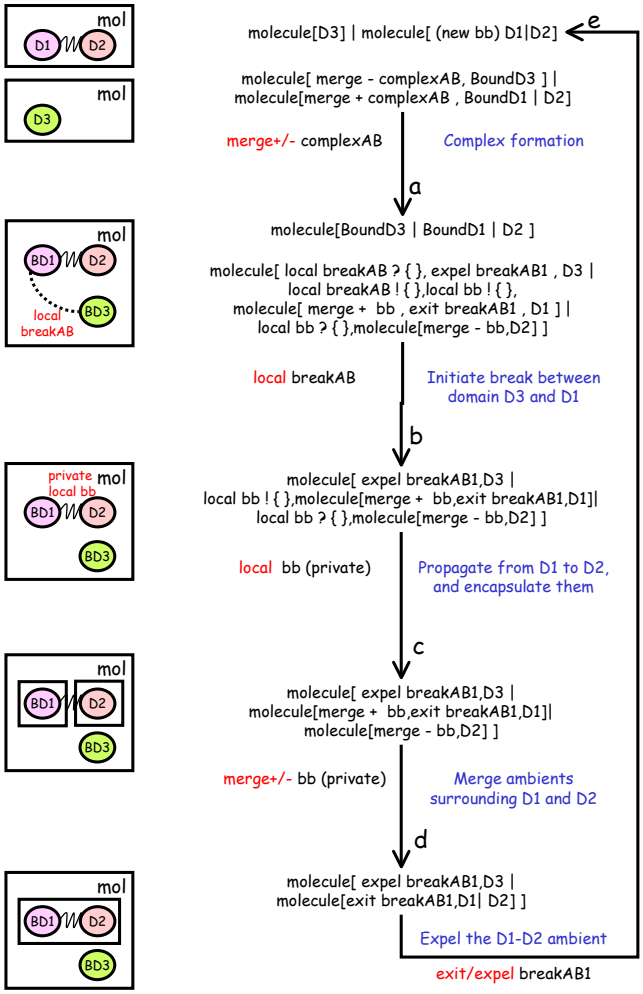

Figure 3.22: **Protein complex example.** An illustration of the BioAmbients reduction steps representing formation and breakage of a two-protein complex. While complex formation is represented by a single step (**a**), breakage requires several steps, from initiation of the break (**b**), through inter-process interactions (**c,d**) and the eventual separation of the two ambients (**e**).

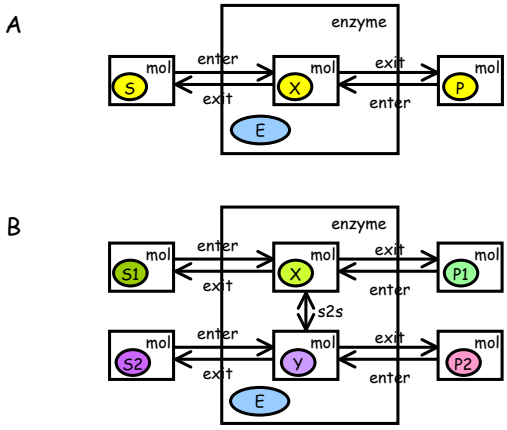

Figure 3.23: **Enzymes as molecular ambients.** Schematic representation of reversible single substrate (**A**) and bi-substrate (**B**) reactions with BioAmbients.

expects only limited interaction between the complexed proteins and the external environment.

A prime example for such a scenario is the enzyme-substrate complex. The general framework for representing enzymatic reactions as the movement of molecular ambients is shown in Figure 3.23. We abstract the enzyme and each of its substrates as a separate ambient (with a process inside) and enzyme-substrate binding is modeled as entry of the *substrate* ambient to the *enzyme* ambient. For a reversible single-substrate reaction (Figure 3.23A), both forward reaction (or product release) and substrate unbinding (or reverse reaction) are abstracted as ambient exit, highlighting the symmetry of both directions. For a bi-substrate reaction (Figure 3.23B), the model is more complex, with a sibling to sibling communication between the two *substrate* ambients inside the *enzyme* ambient representing the reaction (preceded and followed by *enter* and *exit* events).

A toy example of a reversible single substrate reaction is shown in Figures 3.24 and 3.25. The enzyme, substrate, and product are represented by three ambients harboring the E, S, and P processes. $enter - accept$ capabilities on e_s_bind and on e_p_bind represent enzyme-substrate and enzyme-product binding, respectively. The resulting ES complex is abstracted as a nested structure with the *substrate* ambient inside the *enzyme* one. Next, an $exit - expel$ event occurs on either the *unbind* channel (resulting in release of the ambient with an intact S), or the *react* channel (releasing the ambient with a product P). Note that the channel names (*unbind*, *react*) were given with the forward reaction in mind, but in fact represent the same type of molecular event. Thus, both sides of the reaction are seamlessly and symmetrically represented.

3.6 BioSpi 3.0: Extending BioSpi to simulate BioAmbients

We implemented BioAmbients by extending the BioSpi simulation system, developing BioSpi 3.0 for BioAmbients. The adaptation required three major changes: incorporation of a hierarchical ambient tree, handling ambient capabilities (entry, exit, merge), and scoping communication by ambient organization, distinguishing between local, sibling, and parent-child directions. In BioSpi 3.0 a single channel name may be used for six types of synchronization - three for communication and three for capabilities - with capabilities entailing potentially significant changes to the organization of the ambient hierarchy and to

```
System::= enzyme[E] | ... | enzyme[E] |
          molecule[S] | ... | molecule[S] .
E::= accept e_s_bind, ES ;
     accept e_p_bind, ES .
ES::= expel unbind, E ;
      expel react,  E  .
S::= enter e_s_bind , X .
X::= exit unbind, S ;
     exit react,  P .
P::= enter e_p_bind , X .
```

Figure 3.24: **BioAmbients code for a single substrate enzymatic reaction**

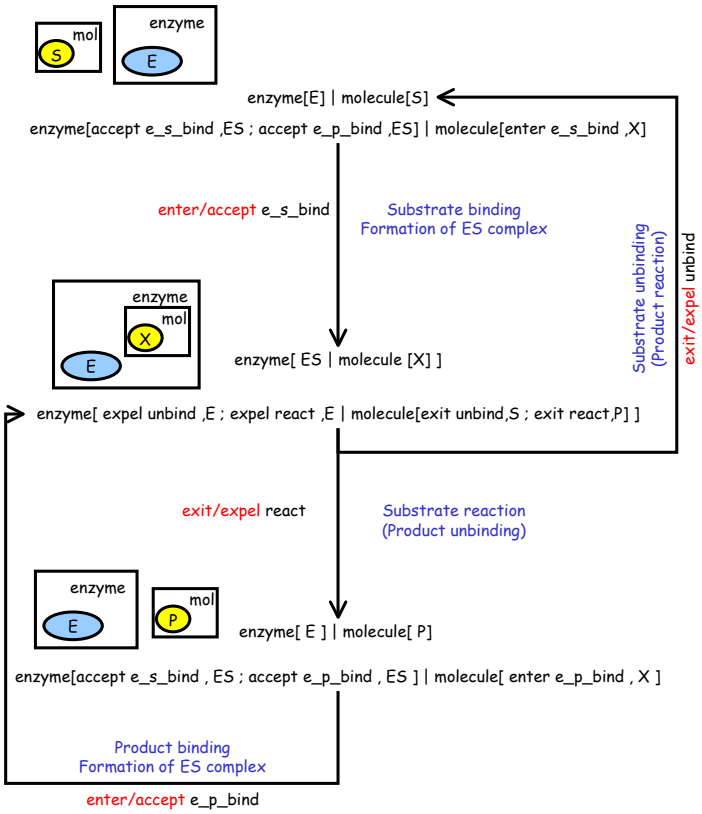

Figure 3.25: **Single substrate enzymatic reaction.** An illustration of the BioAmbients reduction steps representing a reversible single substrate enzymatic reaction.

channel scopes.

3.6.1 The ambient tree hierarchy

Ambients are transformed to FCP [128] procedures, and organized as a tree, permitting inter-ambient communication. Each node in the tree may include active processes and channels. The root of the tree is identified by the name "system" and is always present. The initial tree includes an automatically generated ambient node which is identified by the tuple global(1)[6]. All other nodes are identified by a tuple consisting of a user-specified name (*e.g.* *cell*, *membrane*, *proteinA*, etc.) and a unique positive integer (*e.g.* cell(6)).

Within a given run, a communication or capability offer from a process is first interpreted by the FCP procedure representing the ambient in which the process resides, before being forwarded to the monitor. The stochastic simulation is carried out by the monitor as before, and a single communication or capability out of the entire system is chosen for execution at a time.[7]

The ambient tree structure may change during the course of a given run as a result of exercising capabilities (exit, enter, merge) or due to a declaration of a sub-ambient within a process. In the former case ambients either change their position in the tree (*exit* and *enter*) or are united (*merge*), and the channel sets associated with them and with their new parents may be modified as necessary. In the latter case, once a membrane is declared, a new ambient is created under that node in the tree, with the necessary processes and channels. Both cases are detailed below.

3.6.2 Channels, communication and capabilities

Since channels are used in BioAmbients for six different types of synchronous actions (three for communication and three for capabilities), BioSpi's channel scheme is extended to correctly handle the different kinds of interaction. Importantly, we want a channel name to be declared only once, and allow its use for the different actions in a context-sensitive manner. Thus, we must introduce automatic generation of "derived" channels, by reference. We must also associate the different types of channels with different levels in the ambient tree, based on their particular type, and use them in the correct scope.

To this end, we distinguish between two kinds of channels: internal and external. An internal channel may be used for intra-ambient communication (*local*), while an external channel may be used either for inter-ambient communication ($s2s$, $p2c - c2p$) or to assert a capability (*merge*, *enter*, *exit*). At first, all channels are created as internal channels. Then, an external channel of a particular type is automatically derived from an internal channel when an action of that type is first declared on that channel in a given ambient. For example, a fresh version of a global channel is used for *local* communication in each ambient. Thus, channels which share names but have different types or reside in different ambients are separate channels. In fact, the channels cannot be sent or received. Rather, an internal channel's *name* may be sent to another ambient, but the receiving ambient creates or associates a local internal version of it.

[6]This node may disappear from the ambient tree during the execution of a run under certain conditions, due to elimination of empty ambients. For example: $Process ::= (newa)[exit\ a.a!\{\cdots\}]|expela.0$.

[7]There are several reasons for this two-step implementation. First, channels must be managed separately per ambient (as detailed below), and, therefore, the ambient must record all channel creations. Second, inter-ambient (sibling or parent-child) communications are managed in most cases by the parent of the ambient in which the communications originate, and must be associated with the originating ambient as well. Thus, rather than complicate the monitor by adding these and related management functions per ambient, each ambient is realized as an independent object, with all such objects organized into a tree. In this way, BioSpi 2.0's monitor is preserved in BioSpi 3.0, with only minor extensions.

As before, a channel may be either public or private. Public channel names are globally defined, while private channel names are declared locally by a specific process. Internal and external channels have specific identifiers[8], which distinguish channels of the same name but with different directions. Both public and private channels are internal to the ambient; the ambient's identifier is effectively part of the channel's name.

A special case is introduced when the ambient tree structure changes. When exercising capabilities, the channel sets associated with the moving or merging ambients or their new parents may require modification. When an ambient *exit*s from its parent or *enter*s a sibling ambient, all of its references to non-*local* channels of its parent are transferred to the corresponding channels of the new parent. This may require creation of non-*local* channels in the new parent; references to the channel are transferred from the old parent to the new one. When an ambient *merge*s into a sibling ambient, all of its channels and references to them are transferred to the merged ambient. When a new ambient is declared, we consider all the local channels in the ambient of the declaring process. All such channels which are needed by the transitive closure of the new ambient and its continuations are inherited by the new ambient. A fresh internal version of each channel is created, and resides in the new ambient.

Table 3.3 summarizes the different types of channels, the ways in which they can be generated, the ambient to which they belong in the hierarchy, and the consequences of exercising each action.

3.6.3 Tracing and recording BioAmbients simulations

Following BioAmbients simulations requires some revision of the tracing and recording tools we have used in the previous versions.

The ambient tree of a computation can be displayed at different levels of detail, showing node hierarchy and names as well as processes and channels associated with each node. Specific filters may be used to show only enabled (with active requests in both directions) or communicating (with request in at least one direction) channels. Specific details on the channel name, weight, number of messages, or type are also available where relevant. Either the entire tree may be shown, a sub-tree rooted at a specific ambient, a single specific node, or a class of nodes (e.g. all *cell* ambients). For example, consider the ambient tree resolvent, showing the ambients, processes, and channels after 100 simulation time units of the porin code, with 2 cells (each with one porin molecule) and 10 protein molecules (Figure 3.26A).

The ambient tree provides us with a comprehensive snapshot of system organization. For a dynamic view, a record file can be generated during a run, specifying all communications and capabilities which occured, the time at which they occured, the participating processes, and the employed channel. The BioSpi 3.0 record allows us to either count processes per ambient (as the ambient's unique numeric identifier is part of the process name), across all ambients, or across a specific type of ambient, as specified by its name. For example, by parsing the record of the porin simulation described above in different ways, we can plot the time evolution of the total amount of protein (unchanged, Figure 3.26B) or its amount in each of the two cells separately (Figure 3.26C).

[8] An internal private channel is identified by a tuple whose functor is the name of the process which created it, concatenated with the name specified by that process, and a unique positive integer argument, which discriminates channels created by different instantiations of the same process (*e.g.* $Symporter\_In.0.symI(1)$ where $Symporter\_In.0$ is the declaring process, $symI$ is the specified name, and (1) is the unique positive integer argument; Imbedded numbers, as in $Symporter\_In.0$, indicate the sub-process which declared the channel). A public internal channel is identified by the word "global" concatenated with the declared name of the channel (*e.g.* $global.cell3$). External channel identifiers are derived from the internal ones by a functor specifying the operation, with an argument which is the identifier of the internal channel (*e.g.* $p2c(Symporter\_Out.0.symO(2))$ or $merge(global.complexAB)$).

Channel type	**Syntax**	**Function**	**Generation**	**Location** (relative to declaring or referring ambient)
Local	$local\ a\ ?\ \{\cdots\}$ $local\ a\ !\ \{\cdots\}$	Intra-ambient communication	Private declaration	Same ambient
			Public declaration	Every ambient
			Receipt of message	Receiving ambient
Lateral	$s2s\ a\ ?\ \{\cdots\}$ $s2s\ a\ !\ \{\cdots\}$	Communication between sibling ambients	By $s2s$ reference in a process in a child ambient	Parent ambient
Inter	$p2c\ a\ ?\ \{\cdots\}$ $p2c\ a\ !\ \{\cdots\}$	Communication from parent to child	By $p2c$ reference from a process in the ambient	Same ambient
	$c2p\ a\ ?\ \{\cdots\}$ $c2p\ a\ !\ \{\cdots\}$	Communication from child to parent	By $c2p$ reference from a process in a child ambient	Parent ambient
Capa_Join	$merge+\ a$ $merge-\ a$	Ambient merger	By $merge+$ or $merge-$ reference from a process in a child ambient	Parent ambient
Capa_In	$enter\ a$ $accept\ a$	Ambient entry	By *enter* or *accept* reference from a sibling ambient	Accepting ambient
Capa_Out	$exit\ a$	Ambient exit	By *expel* reference from a parent ambient	Parent ambient
	$expel\ a$	Ambient exit	By *exit* reference from a child ambient	

Table 3.3: **Channel generation.** Types of channels and the ways in which they are generated in BioSpi 3.0.

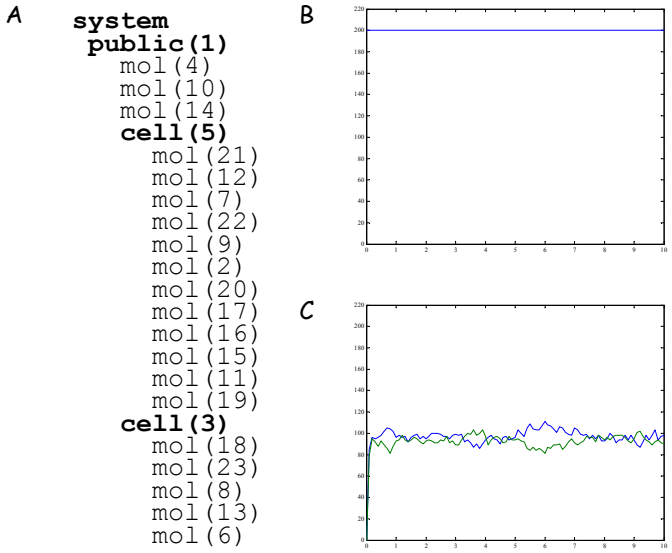

Figure 3.26: **BioSpi ambient tree and record for the porin example.** A. An ambient resolvent tree of the porin example run with 2 *cell* ambients, and 15 *molecule* ambients after 10 units simulation time. B,C. Total *Protein* numbers (B) and *Protein* numbers in each *cell* (C) in a system run for 10 units with 2 *cell* ambients, and 200 *molecule* ambients. In all runs the entry rate is 10 times the exit rate.

3.7 Multi level models

We conclude this chapter with an example of how the ambient biochemical calculus can be used to model and study a multi-cellular, multi-level system. Note, that our focus here is on the abstraction challenge, *i.e.* how compartment-related issues can be handled in a formal language. Further simulations and analysis are required in order to glean novel biological insights from the model.

3.7.1 The hypothalamic weight regulation system

The example we have chosen is the hypothalamic weight regulation system, which involves several levels of biological organization: molecular, cellular and anatomical. First, we briefly review the system.

In general, energy enters the organism as food, and exits as either heat or work. Energy can also be stored, mostly in adipose tissue, and mobilized when necessary. The balance between energy intake, expenditure and storage is determined by a tightly controlled feedback system for body weight regulation that is based on energy homeostasis. This system consists of four components: a central controller in the brain which serves as a receiver and transducer for input signals and integrates output signals, from and to neural and hormonal systems. The output signals modulate the controlled system, which seeks and absorbs food and stores and metabolizes its nutrients. The controlled system reports its state to the controller thereby closing a feedback loop. Complex physiological and molecular mechanisms underly the function and integration of these components (*e.g.* Figure 3.27). Their elucidation is key to the understanding of body weight regulation in normal and obese individuals.

The two major types of input signals are adiposity signals and satiety signals. We focus on the former. Adiposity signals are sent by adipose tissue, processed at the hypothalamus, and result in output signals that reduce energy accumulation. The hormones leptin and insulin are the two adiposity signals identified to date. Both hormones circulate at levels proportional to body fat content, and enter the CNS at levels

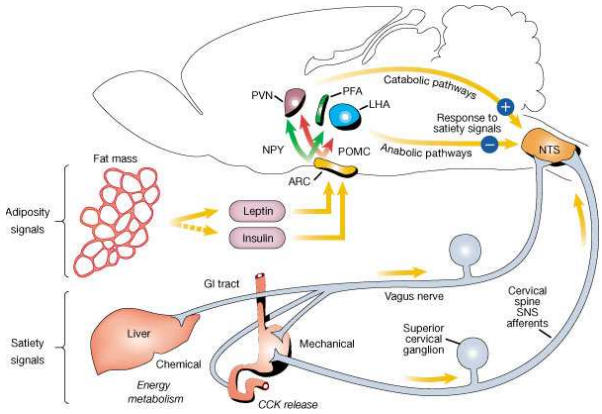

Figure 3.27: **Neuroanatomical model of body weight regulation.** The adiposity hormones, leptin (secreted by adipocytes) and insulin (secreted by the endocrine pancreas in proportion to adiposity), interact with central autonomic circuits regulating meal size. Leptin and insulin are proposed to stimulate a catabolic pathway (POMC/CART neurons) and inhibit an anabolic pathway (NPY/AgRP neurons) that originate in the arcuate nucleus (ARC) in the hypothalamus. These pathways project to the PVN and LHA nuclei, where they make connections with central autonomic pathways that project to hindbrain autonomic centers that process satiety signals. Input signals related to satiety from the liver, gastrointestinal tract and from peptides such as CCK are transmitted through the vagus nerve and sympathetic fibers to the nucleus of the solitary tract (NTS), where they are integrated with descending hypothalamic input. Net neuronal output from the NTS and other hindbrain regions leads to the termination of individual meals, and is potentiated by catabolic projections from the PVN and inhibited by input from the LHA/PFA. Reduced input from adiposity signals (*e.g.* during diet-induced weight loss), therefore, increases meal size. Reproduced from [121]

proportional to their plasma level. Leptin receptors and insulin receptors are expressed by brain neurons involved in energy regulation. Administration of each directly into the brain reduces food intake, whereas deficiency of either does the opposite. However, only leptin deficiency causes obesity.

The central controller in the brain receives input signals, utilizes a complex neuronal and biochemical circuitry to integrate them, and elicits output signals, which will affect the various physiological functions related to energy homeostasis. The arcuate nuclei (ARC), ventromedial nucleus (VMN), and paraventricular nucleus (PVN) of the hypothalamus directly receive adiposity signals through insulin and leptin receptors on the membranes of specific neuronal sub-populations (shown for leptin in Figure 3.28). The insulin receptor, a dimer, is a receptor tyrosine kinase. It is activated upon insulin binding, and leads to phosphorylation of multiple substrates, such as IRS-1 and Tub. Leptin receptors are members of the cytokine receptor superfamily, and have no intrinsic kinase activity. Their effects are mediated both by *de novo* gene expression (through Jak-STAT signaling) and by affecting membrane potential and neuronal firing rate, independently of transcription. Negative feedback, mediated by SOCS3 expression, shuts down leptin signaling.

We focus on the ARC. The activation signal received through leptin and insulin receptors in the ARC is processed by a series of integrated responses, that "compute" the appropriate output (Figure 3.28). Multiple neuronal systems and anatomical sites are involved in this response. The response to leptin is determined by the balance between orexigenic signaling, which induces energy accumulation and anorexigenic signaling, which induces energy expenditure. Different neuron sub-populations in the ARC are responsible for distinct signals. We limit ourselves to the first and second orders of this process.

In the first-order response at the ARC, we distinguish between NPY/AgRP neurons that produce the orexigenic Neuropetide Y (NPY) and AgRP hormones, and POMC/CART neurons that produce

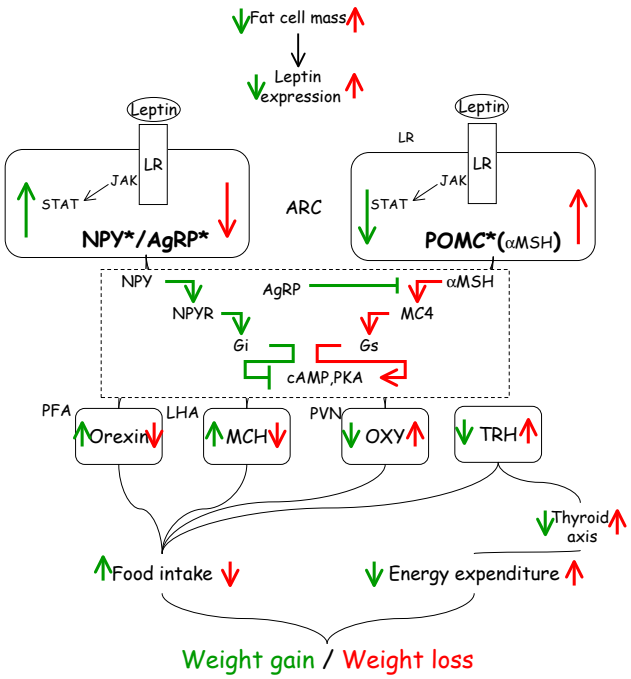

Figure 3.28: **Hypothalamic pathways for weight regulation.** A partial view of molecular pathways, neurons and nuclei involved in weight control Orexigenic (weight gaining) signals are in green, anorexigenic (weight loss) ones are in red. For further details see the main text. Adapted from [121].

the anorexigenic neuropeptides αMSH and CART. Both neuron populations harbor leptin and insulin receptors. A rise in leptin and insulin levels induces expression and secretion of the anorexigenic peptides (αMSH, CART) and reduces expression and secretion of the orexigenic ones (NPY and AgRP). Fall in leptin levels has an opposite effect.

The ARC neurons innervate several additional sites, and the released peptides elicit a second order response and further coordinated synthesis of appropriate signaling neuropeptides (including the anorexigenic TRH, CRH, and oxytocin, and the orexigenic orexin and MCH). This involves signaling through specific receptors, as well as negative interference of orexigenic peptides (AgRP) with anorexigenic signaling (by αMSH). Many of the second-order neuropeptides may have a direct effect on the production of output signals. For example, CRH and TRH affect the hypothalamic-pituitary-adrenal axis and the thyroid axis, respectively. Both axes play a key role in the control of food intake and energy expenditure.

Numerous experiments indicate that various neuropeptides affect body weight by regulating food intake, energy expenditure, or both. However, the understanding of output signals transmitted to the various components of the controlled system is still rudimentary. Two key regulators, however, are well known. These are the thyroid hormone and adrenaline, both of which are anorexigenic signals, which increase energy expenditure.

3.7.2 An ambient model for weight regulation

The fragment of the hypothalamic system for body weight regulation shown in Figure 3.28 offers an interesting abstraction challenge as it requires us to simultaneously handle molecular events (receptors, signaling pathways and gene expression) for which variable degrees of knowledge exist, within a hetero-

geneous cell population (different types of neurons), which are further sequestered to distinct anatomical compartments.

We abstracted the system using BioAmbients (Figure 3.29, [106]). First, we abstracted each hypothalamic nucleus as an ambient (*arc*, *pvn*), harboring ambients representing individual neurons (*arc_neuron*, *pvn_neuron*). For simplicity, we did not represent all individual neuron types. Rather, we assumed that all neuropeptides in a given nucleus can be produced by a single neuron type. Note, however, that full representation of neuron types does not require any conceptual complications.

Second, we represented the molecular components as processes, according to the same principles employed in modeling other molecular systems. Processes abstract each of the extracellular neuropeptides (*Leptin* for the input signal, NPY, $AgRP$, MSH for the first order response and MCH, *Orexin*, and *Oxytocin* for the second order response); their respective receptor molecules (LR, $NPYR$, $MC4$); leptin-regulated, neuropeptide-specific transcription factors (TF_NPY, TF_AgRP, TF_POMC, MCH_TF, $Orex\_TF$, Oxy_TF); and neuropeptide encoding genes ($NPYG$, $AgRPG$, $POMCG$, $MCHG$, $OrexG$, $OxyG$). In addition, several general processes are abstract the machineries for transcription, degradation and hormone export.

The molecular model itself is relatively simplified. The processes representing hormones reside outside the *neuron* ambients, and communicate with their cognate *Recpetor*s using $p2c/c2p$ communication on specific channels. LR and $NYPR$ (representing receptors) can communicate with a single neuropeptide process each (*Leptin* and NPY, respectively). $MC4$ has a choice construct allowing it to communicate with one of two processes representing hormones: $AgRP$ and MSH. The fact that only the latter is an agonist that would result in signaling receptor is represented by the communication offered following the corresponding choice. The fact that the former blocks the receptor until unbound, is reflected by lack of such an offer. In all cases a private *unbind* channel is exchanged and is used for specific $p2c/c2p$ communication, representing unbinding.

We abstract away the partially characterized signal transduction network and allow direct communication between the processes representing receptors and those representing transcription factors via specific $s2s$ channels. The TF-representing processes have two different states (TF and TF_Active) and switch between them according to communication from the processes representing activated (bound) receptors. In the *arc_neuron* ambient, the processes representing orexigenic TFs (TF_NPY, TF_AgRP) are switched to an "inhibited" state by communication from LR (leptin receptor), while the one representing anorexigenic TFs (TF_POMC) is switched to an "activated" state. A counter-active communication (representing a constitutive negative signal) is supplied by ARC_PHOSPH which communicates with the TFs, resetting them in the opposite direction. Note, that this process replaces negative regulators of leptin signaling such as the SOCS3 protein (see above), albeit in a constitutive rather than a leptin-induced manner (for simplicity). In the *pvn_neuron* ambient, there are two receptors. $MC4$ communicates with both the orexigenic TFs (MCH_TF, and $Orex\_TF$) that switch to an inactive state, as well as with the the anorexigenic ones (Oxy_TF) which switch to an active state. $NPYR$ has the opposite effect. No additional negative signal is thus incorporated in this layer.

The processes representing active TFs can communicate with their cognate *Gene* processes and "bind them", switching the *Gene*s to an active state, that can result in release of new *Hormone*s (transcription and translation are abstracted in a single step). TF unbinding is abstracted as before by communication on private *unbind* channels, albeit $s2s$ ones. *Hormone*s are first encapsulated in specific ambients (*npy*, *agrp*, *msh*, *mch*, *orexin*, *oxytocin*). The ambient membrane is essential for the export of the *Hormone*s

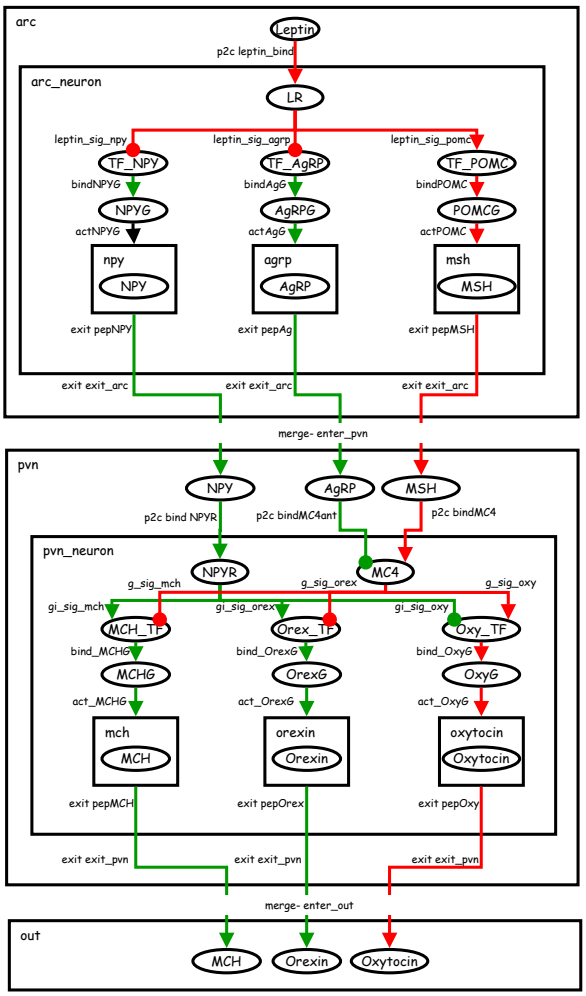

Figure 3.29: **An ambient calculus models for hypothalamic weight regulation.** The ambient model is depicted graphically, with ambients as rectangles and molecular processes as ovals. Channel names used in communications or capabilities are shown as labeled arrows, green and red for orexigenic and anorexigenic signals, respectively. Pointed arrowheads represent activatory events, round heads for inhibitory events. The model is simplified, with a single neuron type per nucleus and only some of the neuropeptides. Generic processes and communications (abstracting transcription, hormone export and clearance, unbinding) are not shown but are included in the model, as is a negative signal in the ARC nucleus. The full code is available at [106] and consists of 150 lines and 38 processes.

to another *nucleus*.

The abstraction of the export process involves three steps: exit of the *hormone* ambient from the *neuron* ambient, exit from the *nucleus*, and merge into another *nucleus*, thereby "spilling" the resident *Hormone* outside the *neuron* ambient. These events are synchronized by an $exit-expel$ capability with a generic *Export* process in the *neuron* (for exiting the neuron), followed by an $exit-expel$ capability with a generic *Boundary* in the original *nuclues*, and finally a $merge+/-$ capability with the *Boundary* in the *nuclues* to which the *hormone* ambient has entered. Generic *Hormone_clearance* processes are constitutively active, and remove the generated *Hormone*s, by sending them an alert, resulting in the nullification of *Hormone* to an empty process. Note, that since we model only two layers of the system, we added an artificial *out* ambient to accumulate the second order *Hormone*s (MCH, $Orexin$, $Oxytocin$).

To obtain a proof of principle of the operation of this model, we executed our BioAmbients program in BioSpi 3.0. As always, when running a simulation we need to select parameters: number of cells,

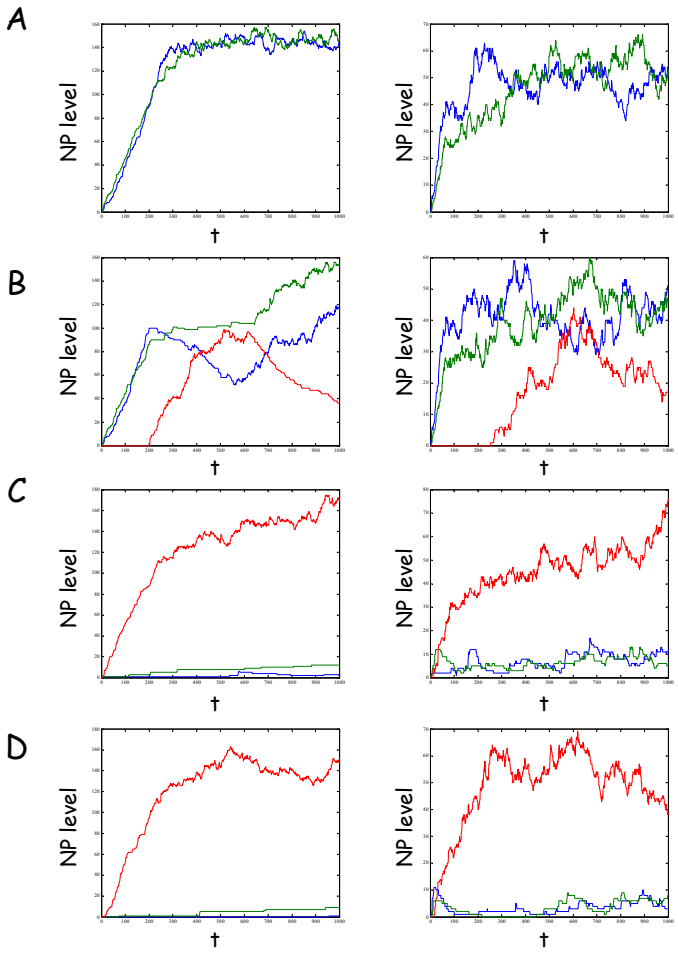

Figure 3.30: **Neuropeptide profiles under different levels of Leptin.** Simulation results of neuropeptide levels under various Leptin creation rates (A) 0.0001 (B) 0.01 (C) 1 (D) 100. In each panel first order hormones, AgRP (blue), NPY (green), and MSH (red) are shown on the right, and second order hormones, MCH (blue), Orexin (green) and Oxytocin (red) are shown on the left. The anorexigenic hormones (red) are high when leptin levels are high, while the orexigenic ones (green,blue) are high when Leptin is low, as expected.

initial number of molecules, and the reaction rates. Here, we only focus on obtaining a crude measure of the model's operation, and we select relatively arbitrary and simplified conditions, where most reactions have a uniform relative rate of 1, with the exception of hormone unbinding and clearance rates which are 100-times slower[9]. We included 10 Leptin receptor processes (LR) and 100 $NPYR$ and $MC4$ receptor processes per cell. For this initial evaluation only a single neuron ambient was included per hypothalamic nucleus. Obviously, these initial parameters must be realistically updated for a thorough study of the system.

Our major test of this preliminary model's operation was to examine the levels of the six output $Hormone$s under four different levels of leptin in the ARC. When operating properly, the modeled system should generate high levels of anorexigenic hormone processes and low levels of orexigenic ones when leptin is high, and vice versa when leptin is low. Indeed, as shown in Figure 3.30, the system has generally behaved as expected.

These results are but a preliminary demonstration of the general operation of the modeled system, with

[9]Faster (1) unbinding rates did not significantly change the reported results (not shown).

the sole purpose of handling the abstraction problem. Further work is required in order to substantiate the model and glean biological insights. First, realistic cell numbers should be used rather than a single representative cell. This can be accommodated within the capabilities of the BioSpi system as cell populations are relatively small. Second, a thorough examination of parameter space should be carried out, to map feasible parameters, and evaluate the robustness of the response. This can be further combined with a "knockout" type study, where different components genes are removed from specific simulations, and the obtained results compared to those of actual knockout experiments. Third, the biological accuracy of the model can be increased by adding other known interactions. These include a feedback loop from the second-order response neurons to the first-order ones; a feed-forward of the leptin signal to both levels of neurons directly (i.e. leptin not only affects the second-order neurons through the first order ones, but also directly through leptin receptors); and lateral interactions between the anorexogenic NPY/AgRP neurons and the POMC ones. Each such addition should be examined for its effect on the behavior of the modeled system in order to evaluate its role.

3.8 Perspective: BioAmbients

3.8.1 Developing BioAmbients from the ambient calculus

The original ambient calculus contains the critical component necessary to abstract biological compartments: the bounded ambient. However, the specific use of ambients and communication was not an optimal mathematical domain for a relevant and understandable abstraction of biological compartments. In developing the BioAmbients variant we attempted to identify such discrepancies and address them by modifying the original ambient calculus.

First, in biology both movement of compartments and of molecules across compartments requires some interaction between the moving entity and the compartment which it attempts to "cross". In the original ambient calculus movement was asynchronous and could be initiated in a one-sided manner by a process in the moving entity. Furthermore, all movement in the original ambient calculus required the moving entity to "know" a single ambient name, which allowed all the exit and entry operations from a given ambient. Biological compartments (both membrane-bound and molecular), however, may be entered, exited or joined by molecules via multiple routes, with different specificities and rates. To handle these two limitations in BioAmbients, we essentially abolished ambient names and replaced the unilateral movement capabilities with bi-lateral ones that are synchronized on specific channels. Thus, movement now requires synchronization between two ambients, and may be done in different ways by using different channel names.

Second, interaction in biology is synchronous, and may occur both within the compartment boundary and across it (*e.g.* through cross-membrane molecules or in complexes), albeit only in specific configurations. In the original ambient calculus, communication events are asynchronous, and all communication is purely local, such that two processes can communicate with one another only if they reside side-by-side in the same (immediate) ambient. To allow cross-boundary communication we equipped BioAmbients with two additional communication directions: sibling and parent-child. The former are essential to handle the abstraction of molecular compartments as ambients (without them we cannot abstract inter-molecule or inter-complex interactions). The latter are required to abstract interactions between cross-membrane molecules and other molecules outside their compartment.

Third, the semantics of both moves and communication in the original ambient calculus is non-

Ambient calculus	BioAmbients
Asynchronous movement	Synchronous movement
Asynchronous communication	Synchronous communication
Non-deterministic semantics	Stochastic semantics
No *merge* primitive	Ambient *merge* primitive
Only local communication	Sibling and parent-child communication is allowed
Named ambients	Nameless ambients (Ambients are named for readability only)

Table 3.4: **The Ambient calculus vs. BioAmbients**

deterministic, while a correct abstraction must be quantitative (as discussed in Chapter 2). To this end, we supplied BioAmbients with stochastic semantics, along the lines of our biochemical stochastic π-calculus.

Fourth, the original ambient calculus did not provide a primitive movement capability for ambient merger. In biology, both the merge of molecular compartments (*i.e.* complex formation) and that of membrane-bound compartments (*i.e.* membrane fusion) is prevalent. We therefore added in BioAmbients the "merge+/merge-" primitive pair (even though a merge operation may be encoded in the original ambient calculus by several steps).

Finally, some of the components of previous variants of the ambient calculus are actually superfluous for the abstraction of biological systems. These included the movement of "naked" processes across boundaries.[10] and ambient names. We removed these from the BioAmbients variant. The distinctions between BioAmbients and the ambient calculus are summarized in Table 3.4.

3.8.2 The compartment as ambient extension

One of the greatest limitations to the understandability of the "molecule as computation" abstraction was the use of private channels as the sole abstraction of biological compartments (see Chapter 1). This raised several problems including the cumbersome encoding of events related to multi-molecular complexes, the need for match constructs to abstract a fit in both structure and location, and the vague representation of membrane-bound compartments.

The "compartment as ambient" abstraction significantly extends the "molecule as computation" abstraction based on the π-calculus. It provides relevant, understandable and simple mechanisms to abstract biological compartments – both membrane-bound and molecular – and their movement. Importantly, this extension is done easily, without any need to change the core mathematical domain and without violating the semantic and pragmatic guidelines of the abstraction. We have shown the utility of this extension to handle a variety of small, but essential, biological examples, as well as a complex realistic one. Thus, a whole spectrum of entities and events related to biological compartmentalization — membranes, stable and transient complexes, location and movement of molecules, vesicles and cells — can be captured in a uniform and straightforward way. The extensibility of the "molecule as computation" abstraction is one

[10] Allowing "naked" processes to move across ambient boundaries is useless for modeling the movement of multi-domain (*i.e.* multi-process) molecules. For small molecules and ions this could prove as a useful solution. However, as small molecules are often present in abundance, we would usually opt to represent them as two (persistent) processes on both sides of the ambient boundary, with internal counters reflecting their quantity at each position. The counter values are updated according to communication with intra-ambient naked process(es) representing cross-compartment molecules (pores, channels, receptors). The counter values also affect interaction rates, serving as "message multipliers" (see Chapter 2).

of its important properties. This is also reflected in the corresponding adaptation of the BioSpi system, to handle BioAmbients. BioSpi 3.0 allows us to study compartmentalized systems by simulation studies, as we show for a complex multi-cellular biological system.

The "compartment as ambient" extension can be further improved in several ways. First, several useful extensions can be added to the current list of primitives to allow the direct modeling of unique biological events. Such extensions include a *kill* capability, to eliminate an ambient and all its contents in one primitive operation; an *acid* capability, to remove an ambient membrane and merge its contents with the parent ambient; a *duplicate* capability, to create two identical sibling ambients from a single one; and a *divide* capability in which the content of the ambient are randomly split into two siblings. While the former three are relatively straightforward to define and implement, the latter one is more challenging. Note, that in all cases these capabilities will not require synchronization, and thus deviate from our current communication and capability model. Other extensions could simplify some of the current multi-step modeling. In particular, complex breakage is now relatively complicated and a $split - break$ capability would be highly useful, but cannot be rigorously defined within the current framework. One possibility is to add a "transparent boundary", which is formed around a protein process during a merge (replacing the old usual ambient boundary), does not create any limitation on communication during the merged state, but serves to identify all of that protein's "pieces" necessary to exercise a $split - break$ capability. Transparent boundaries may also facilitate better representation of multi-domain, cross-compartment molecules, which we currently represent as "naked" processes offering child to parent communication.

Second, the hierarchical organization of ambients calls for a graphical representation. We have made informal use of graphics throughout this work. However, these did not have clear syntax and semantics, and did not include dynamic information. The lack of a graphical component is a major disadvantage of our approach compared to graphical languages such as Statecharts [64]. Bigraphs were recently developed by Milner [104] as a graphical formalism for concurrency. A bigraph (Figure 3.31) has nodes which support two types of structure. The first is the *topograph*: nodes may occur inside one another, so a bigraph has hierarchical organization. The second is the *monograph*, where nodes may be linked by edges. The nesting and linking structures may both change as a result of interaction. A certain combined pattern of nesting and linkage determines the possible communications (or reactions) in the graph, and hence its dynamics. The pre- and post-conditions of such reactions are stated in reaction rules. Clearly, the bigraph notation has striking resemblance to our informal models (e.g. Figure 3.29). In fact, bigraph notation has been shown adequate to represent both the π-calculus and the original ambient calculus. We plan to explore its adaptation to *biopgraphs* suitable for our biological variant of the ambient calculus.

Finally, in handling compartmentalization, ambients provide us with a coarse-grained, albeit flexible, way to handle some aspects of the heterogeneous organization of biomolecular systems. A longer term challenge is to fully handle three dimensional space. A first step in this direction would be to handle space as a lattice of ambients, and diffusion of molecules as movement across them. This would ultimately require a new kinetic semantics as well.

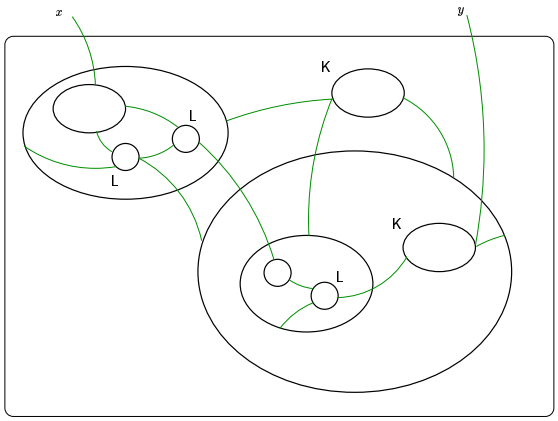

Figure 3.31: **An example of a bigraph.** Ovals and circles represent nodes, which are organized in two structures simultaneously: a hierarchical nesting *topograph* and an edge-based linkage structure *monograph*.

Appendix A

Inference of biological pathways from gene expression data

A.1 Introduction

There are two complementary ways to construct a model. In the first approach, the real world system, *e.g.* a biological pathway, is known, and existing knowledge is abstracted into a predictive model. The abstract representation can then be used to simulate and analyze the modeled system's behavior, ultimately comparing it to that of real-world system. The model may suggest new testable hypotheses or direct us at discrepancies between the operation of the abstract model and real systems. Such discrepancies indicate points at which our understanding of the scientific domain is lacking. The work described in this thesis follows this approach, trying to develop a new framework for direct modeling. This methodology has two limitations. First, due to the lack of quantitative data, it severely limits the number of known systems amenable to investigation by modeling as well as prohibits testing of the derived predictions. Second, the discovery of completely novel pathways and interactions is rarely achieved by model-driven predictions.

The second approach attempts to address these problems by *inferring* models of unknown systems directly from raw experimental data (*e.g.* [39]). For biological pathways the prime sources of data are RNA and protein expression patterns, obtained with high-throughput cDNA microarrays [75] and two-hybrid systems [105]. In on-going collaborative work, I have explored the inference approach on reconstructing molecular and regulatory networks in the yeast *Saccharomyces cerevisiae* from gene expression data ([100], [101], [122]).

Our work employs graphical probabilistic models to reconstruct models of biomolecular systems. Such models are stochastic descriptions of biological biological processes that could have generated the observed data. In particular, we treat gene expression as a probabilistic process, and represent the expression level of each gene as a random variable [39]. Our approach is holistic: we attempt to reconstruct a model that globally fits our data, by studying the joint probability distribution over the set of all genes. This joint distribution reflects the distribution of cellular and molecular 'states' and how these affect

[0]The work in this appendix was carried out in collaboration with the Friedman group at the School of Computer Science and Engineering, Hebrew University of Jerusalem and the Koller group at the Computer Science Department, Stanford University. My contribution was mainly towards the biological formulation of the model (prior to model learning), in the development of bioinformatics methods for the validation of the results, and in carrying out the bioinformatics validations, but not the experimental ones. The work is described in detail in [100], [101], and [122].

transcript levels. Taking various global approaches, we attempt to learn biologically relevant models of gene regulation and molecular organization of different granularities.

A.2 Using Bayesian networks to infer the fine details of molecular interactions

In [100] we used only expression data from *S. cerevisiae* mutants to discover a fine structure of interactions between genes, including causality (as reflected in the statistical dependency of one gene's expression on that of one or more other genes), mediation (where an intermediate set of genes accounts for the dependency between two genes), activation, and inhibition. From these, we identified significant sub-networks of interacting genes, corresponding to metabolic, signaling and regulatory pathways.

To infer such finer relations we use the Bayesian network framework of [39]. In this framework, we treat regulatory interactions between genes as probabilistic dependencies between random variables. We then use gene expression data to learn a Bayesian network model over this set of genes. As the available data are too limited to learn a single high-confidence model, we apply bootstrap analysis to learn an ensemble of such networks, which represent potential models of the interactions between genes. We use this ensemble to extract statistically significant features involving relationships between pairs and triplets of genes. Such features include the Markov and Edge relations (indicating a direct relation between genes, such as regulator-target), the Separator relation (gene or genes which mediate an indirect dependency between two genes), and Activator/Inhibitor annotations. We then identify statistically significant sub-networks, which contain a high density of high-confidence features. These sub-networks capture a strong statistical signal in the expression profile that often reflects a coherent cellular process.

Both the individual features and the sub-networks provide both strong proof-of-principle examples and novel biological hypotheses (For detailed examples see [100], supplied herein). Among the Markov and Edge direct dependencies we found many that represent either a known biochemical or regulatory interaction, a shared common regulator, or a functional link. About a third of these are context-specific, and are not identified by standard correlation-based methods. The Separator relations, in which a single gene directly or indirectly separates two others in a consistent manner, also provide key insights. Often the mediating gene is indeed a known or putative regulator: either a transcriptional or a post-translational one. In some cases, the same Separator appeared in several relations, creating a hub. For example, Slt2, which encodes the MAP kinase of the cell-wall integrity (low-osmolarity) pathway, separates several pairs of functionally related genes encoding cell-membrane, cell-wall, and low-osmolarity signal transduction molecules. The full power of this approach becomes apparent when exploring sub-networks with a high density of higher-confidence features interleaved with lower-confidence ones (which we would not have trusted in isolation). Six structured sub-networks were reconstructed, each representing a coherent molecular response: mating response, low-osmolarity cell-wall integrity pathway, stationary-phase response, iron homeostasis, amino acid metabolism along with mitochondrial function, and citrate metabolism. For example, in the mating response sub-network we discern two distinct branches, one for cell fusion (mediated by the Kar4 gene, a mating transcriptional regulator of karyogamy genes) and the other for outgoing mating signaling, directed from the mating signaling pathway regulator Sst2.

Although impressive, sub-networks are but a 'footprint' of the actual pathways. For example, in the mating sub-network, Ste12, the main mating transcription factor, is completely absent, and Fus3, the pathway's MAP kinase, is in a marginal position. One reason for these problems is that in this

framework, each gene is modeled as an isolated unit, with its own control mechanism. However, from a statistical perspective, there are not enough data to reliably learn such a complex model (hence the need for bootstrap and extraction of individual features). More importantly, from a biological perspective, the detailed model obscures the global organization of genes into modules that act together (in certain conditions) to achieve a common function. In truth, a Bayesian network on a genome-wide scale is an intricate circuit of thousands of genes that is almost impossible to interpret.

A.3 Inferring regulation: *MinReg* and *Module Networks*

To address these two limitations, in subsequent work we are developing alternative probabilistic models that focus on certain essential properties of biological systems while simplifying others.

A.3.1 *MinReg*: An active regulator set

In [101], we focus on regulatory relations between genes, an important component of molecular pathways. Since only a relatively small part of the genome is directly involved in transcriptional regulation, our approach focuses on a small subset of regulators. Accordingly, we define the following task: Given a set of *candidate regulators*, we wish to find a small sub-set of *active regulators*, which control the processes that take place in a given set of experiments, identify the genes that they regulate (their *regulatees*), and characterize both sets.

Our method, *MinReg*, attempts to reconstruct such regulatory relations on a *global* scale, while building on the principles of local modeling of regulatory interactions developed by Friedman *et al.* [39]. In order to detect a small sub-set of active regulators and their respective target *regulatees*, we first model the probabilistic *local* relation between a regulatee and its active regulators and evaluate it based on their mutual information score. We then seek a collection of such relations that optimize an overall score, while adhering to certain *global* constraints.

We constrain our model in two ways. First, We use prior biological information to limit our search to a set of candidate regulators including known and putative transcription factors and signal transduction molecules. While the inclusion of the former as regulators is intuitively reasonable, the relation between signaling molecules and transcriptional regulation is more complex. To justify this choice, consider that in order to capture a regulation event in gene expression data, we must observe changes in the expression of both the regulator and the regulatee. Unfortunately, while transcription factors directly control transcription, their activity may not be transcriptionally regulated and the change in their regulatory activity may be undetectable in gene expression profiles. On the other hand, signal transduction molecules which activate transcription factors are expressed at significantly higher levels. Due to positive and negative feedback loops, their expression may be regulated by the same transcription factors they activate. Thus, we can capture regulation relations indirectly, by the change in the expression levels of the signaling molecule (which in turn regulates a transcription factor) and of its indirect target regulatee genes.

Second, we require that each regulatee may be regulated by only a few regulators, and that the total number of active regulators (over all targets) be small. The first part of this constrain is commonly made when inferring gene networks (*e.g.* [1], [39], [136]). The second part is key to our approach and is both biologically and computationally motivated. It adheres to the assumption that only a small fraction of the genome is directly involved in regulating transcription and that each such "master regulatory gene" may affect the transcription of many other genes. Furthermore, it assists in achieving statistical robustness:

only when a gene consistently scores high as a parent of many genes, we believe it indicates a true signal, while an occasional high score as a parent of a single gene is attributed to spurious chance.

We applied *MinReg* to 358 expression profiles in yeast ([58], [40], [130]). To evaluate our success, we automatically assign biological function to the discovered active regulators according to prominent features of their *entire regulatee set*. We devise a fully automated methodology, based on the Gene Ontology (GO) [10], to map an active regulator r with specific biological processes and molecular functions that are significantly over represented in its set of regulatees. Our derived annotations of active regulators correspond well to their known functions as reported in the literature. The function of 8 of the 10 top regulators were correctly predicted, while of the remaining two, we were able to assign a putative role to one previously uncharacterized gene, but failed to identify the correct role of the other.

Such functional assignment is highly discriminative and comprehensive. For example consider Slt2, the MAP kinase that activates the cell wall integrity pathway. Our GO term derived annotation correctly predicts the biological process which SLT2 regulates (cell wall organization and biogenesis), the molecular function of the effectors (cell wall structural proteins and endopeptidases), the molecular mechanism of regulation (protein kinase cascade) and associated, cross-talking modules (mating, cell-cell fusion). In other cases, *MinReg* provides focused testable hypotheses on the function of previously uncharacterized regulators. For example, it associated the protein kinase Yol128c with the GO terms "Nitrogen starvation response", "response to external stimulus", and "cell cycle control", suggesting that the gene acts as a cell cycle regulator in response to starvation signals (for further details, see [101], supplied herein).

A.3.2 *Module Networks*: Modules and their control programs

In [122], rather than learning regulatory relations between individual genes, we focus on groups of genes with a common expression behavior and shared regulatory mechanism. We developed *Module Networks* - a new framework, which partitions the genes into groups that act together in a coordinated way, and whose behavior, as a group, can be explained as a stochastic function of certain control genes and input signals. As in the *MinReg* approach, the only prior knowledge employed is a set of 600 potential yeast regulators including known and putative transcription factors and signal transduction molecules.

The algorithm automatically learns which of the potential regulators regulate which module, and under what conditions. Each module will thus include two parts: a set of genes (the module's components) and a control program, comprised of regulatory rules (*i.e.* state of regulatory genes and conditions). The regulation program models the behavior of the entire group of genes in the module (*e.g.* if Yap1 is up-regulated (rule), then the detoxifying enzymes module is induced under oxidative stress). The resulting model reveals global patterns over entire groups of genes, and is also characterized by substantially fewer parameters than a Bayesian network, allowing rich and interesting interactions to be learned reliably even from limited amounts of data. The approach is also completely automatic and holistic, providing a global model for all the available data.

The implementation is based on an expectation maximization (*EM*) algorithm that automatically discovers such modules and their regulation programs directly from expression profiles. In a nutshell, this iterative procedure initially groups the genes based on a clustering algorithm. For each 'cluster' of genes, the procedure searches for a set of inputs and a regulation program that provides an explanation for the expression profiles. A regulation program is good if one can hide the expression profiles and then correctly predict them based only on the regulation program and the values of its inputs. After all the regulation programs are defined, the algorithm re-assigns each gene to the regulation program that best predicts

that gene's behavior. The algorithm iterates until convergence, refining both the regulation program and the gene partition at each iteration. By forcing the probabilistic model to explain the variability in the expression data using such rules, we hope to recover groups that correspond to biologically meaningful modules.

We applied the resulting procedure to the "stress" dataset of [40], and recovered fifty (50) modules and their control programs. Evaluation according to functional annotations (from the SGD [25], KEGG [66], and MIPS [87] databases) indicates that most modules are biologically coherent. These include metabolic modules (*e.g.* respiration, galactose metabolism, nitrogen catabolite repression response), stress response modules (*e.g.* oxidative stress, osmotic stress), cellular localization modules (nuclear modules, ER modules), cellular processes (cell cycle, sporulation, mating), and specific molecular functions (*e.g.* protein folding, RNA and DNA processing).

Unlike previous approaches to module reconstruction (*e.g.* [62]), we uncover not only a module's components but also the control program that explains the module's behavior.In most modules (35/50), a module's regulators were consistent with its biological function, and the regulation occurred under the expected conditions. Most modules (30/50) also included genes previously known to be regulated by the module's predicted regulators. Many modules (15/50) also had an exact match between a motif binding site enriched in the module's gene set and the regulator known to bind to that motif.

Reconstructed regulation programs without supporting or refuting evidence specify detailed hypotheses, where each hypothesis suggests a novel regulatory role for a regulator and the conditions under which this regulation occurs. We verified experimentally the computational predictions for four putative regulators with as yet unknown functions (two transcription factors and two signaling molecules). Using microarray analysis, we compared the transcriptional responses of the respective genetically disrupted strains with their congenic wildtypes under the suggested conditions. Mutations in each of the four regulators caused a marked impairment in the expression of their predicted targets, supporting the computational predictions and providing important insight regarding the function of these uncharacterized regulators.

A.4 Perspectives

A.4.1 Why can we infer regulation from expression data?

Our work indicates that correct molecular interactions, in particular regulatory relations, can be can be inferred from expression data allowing partial reconstruction of transcriptional and non-transcriptional regulation. This is an unexpected finding, and a critical challenge for the validity of our approach lies in explaining how regulatory events can be reconstructed.

Our approach relies on the assumption [101] that regulators are themselves transcriptionally regulated, allows our automated procedure to detect statistical associations in expression data between the regulator and its targets. It is widely accepted that this assumption is only partly correct. For example, signaling molecules and transcription factors are often primarily regulated post-translationally. However, large-scale analysis of the regulatory networks of *E. coli* [127] and *S. cerevisiae* [60] reveals the prevalence of feedback and feed-forward cases in which the regulators are indeed themselves transcriptionally regulated. Similar feedback mechanisms is combined signal transduction and regulatory pathways also exist (*e.g.* [40], [113]). For example [101], in the low-osmolarity pathway in yeast, the Slt2 signal transduction molecule activates the Rlm1 transcription factor, which in turn activates both the transcription of various

cell wall proteins and that of Slt2. Indeed, we found that some of the reconstructed regulators and their targets are part of such networks (For details, see [101] and [122], attached herein).

Thus, regulatory events, including post-transcriptional ones, have a detectable signature in the expression of genes encoding transcription factors and signal transduction molecules. Further work is required in order to fully explain why specific regulatory events are inferred in preference to other potential ones. While some of the reasons may have to do with the intricacies of our computational methods, others may be related to critical biological issues such as the strength of feedback regulation and the coordination of action of various regulators.

A.4.2 Integrating direct modeling and inference

Direct modeling and inference are complementary approaches that are based on a shared view of biomolecular systems as stochastic processes. Much is too be gained be combining the two. For example, we may attempt to infer extensions or modification of models of (partly) known systems rather than reconstruct whole systems from scratch. As we wish to reconstruct more accurate and detailed models, the the number of unknown parameters and the complexity of the computational problem increase significantly. Prior knowledge can constrain and facilitate the learning process, thereby maximizing the quality and relevance of our findings.

Bibliography

[1] T. Akutsu, S. Kuhara, O. Maruyama, and S. Miyano. A system for identifying genetic networks from gene expression patterns produced by gene disruptions and overexpressions. In K. Asai, S. Miyano, and T. Takagi, editors, *Genome Informatics*, volume 9, pages 151–160, Tokyo, 1998. Universal Academy Press.

[2] T. Akutsu, S. Miyano, and S. Kuhara. Algorithms for identifying boolean networks and related biological networks based on matrix multiplication and fingerprint function. *Journal of Computational Biology*, 7(3):331–343, 2000.

[3] T. Akutsu, S. Miyano, and S. Kuhara. Algorithms for inferring qualitative models of biological networks. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 5, pages 293–304, Singapore, 2000. World Scientific Press.

[4] R. Albert and H. G. Othmer. The topology of the regulatory interactions predicts the expression pattern of the drosophila segment polarity genes. *Journal of Theoretical Biology. Submitted.*, 2002.

[5] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. D. Watson. *Molecular Biology of the Cell. 3rd edition.* Garland Publishing, 1994.

[6] A. Alessandrini, H. Greulich, W. Huang, and R. L. Erikson. Mek1 phosphorylation site mutants activate Raf-1 in 3T3 cells. *Journal of Biological Chemistry*, 271(49):31612–31618, 1996.

[7] R. Alves and M. A. Savageau. Extending the method of mathematically controlled comparison to include numerical comparisons. *Bioinformatics*, 16(9):786–798, 2000.

[8] M. Antoniatti, B. Mishra, C. Piazza, A. Policriti, and M. Simeoni. Modeling cellular behavior with hybrid automata: bisimulation and collapsing. In *Proceedings of the first international workshop on Computational Methods in Systems Biology*, Lecture Notes in Computer Science. Springer-Verlag, 2003. Accepted for publication.

[9] A. Arkin, J. Ross, and H.H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected escherichia coli cells. *Genetics*, 149:1633–1648, 1998.

[10] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry JM, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nature Genetics*, 25(1):25–29, 2000.

[11] G. D. Bader, I. Donaldson, C. Wolting, B. F. Ouellette, T. Pawson, and C. W. Hogue. Bind-the biomolecular interaction network database. *Nucleic Acids Research*, 29(1):242–245, 2001.

[12] L. Bardwell and J. Thorner. A conserved motif at the amino termini of MEKs might mediate high-affinity interaction with the cognate MAPKs. *Trends in Biochemical Science*, 21(10):373–374, 1996.

[13] N. Barkai and S. Leibler. Circadian clocks limited by noise. *Nature*, 403(6767):267–268, 2000.

[14] U. S. Bhalla and R. Iyengar. Emergent properties of networks of biological signaling pathways. *Science*, 283(5400):381–387, 1999.

[15] D. Bray. Reductionism for biochemists: how to survive the protein jungle. *Trends in Biochemical Sciences*, 22(9):325–326, 1997.

[16] D. Bray and S. Lay. Computer-based analysis of the binding steps in protein complex formation. *Proceedings of the National Academy of Sciences USA*, 94(25):13493–13498, 2000.

[17] A. Brunet and J. Pouyssegur. Mammalian MAP kinase modules: how to transduce specific signals. *Essays in Biochemistry*, 32:1–16, 1997.

[18] E. Buck, J. Li, Y. Chen, G. Weng, S. Scarlata, and R. Iyengar. Resolution of a signal transfer region from a general binding domain in G-beta for stimulation of phospholipase-beta2. *Science*, 283(5406):1332–1335, 1999.

[19] W. R. Burack and T. W. Sturgill. The activating dual phosphorylation of MAPK by MEK is nonprocessive. *Biochemistry*, 36(20):5929–5933, 1997.

[20] B. J. Canagarajah, A. Khokhlatchev, M. H. Cobb, and E. J. Goldsmith. Activation mechanism of the MAP kinase ERK2 by dual phosphorylation. *Cell*, 90(5):859–869, 1997.

[21] L. Cardelli and Andrew D. Gordon. Mobile ambients. In *Foundations of Software Science and Computation Structures: First International Conference, FOSSACS '98*. Springer-Verlag, 1998.

[22] I. A. Carre and L. N. Edmunds. Oscillator control of cell division in euglena: cyclic amp oscillations mediate the phasing of the cell division cycle by the circadian clock. *Journal of Cell Science*, 104(4):1163–1173, 1993.

[23] A. D. Catling, H. J. Schaeffer, C. W. Reuter, G. R. Reddy, and M. J. Weber. A proline-rich sequence unique to MEK1 and MEK2 is required for Raf binding and regulates MEK function. *Molecular and Cellular Biology*, 15(10):5214–5225, 1995.

[24] N. Chabrier and F. Fages. Symbolic model checking of biochemical networks. In *Proceedings of the first international workshop on Computational Methods in Systems Biology*, Lecture Notes in Computer Science. Springer-Verlag, 2003. Accepted for publication.

[25] J. M. Cherry, C. Adler, C. Ball, S. A. Chervitz, S. S. Dwight, E. T. Hester, Y. Jia, G. Juvik, T. Roe, M. Schroeder, S. Weng, and D. Botstein. SGD: Saccharomyces genome database. *Nucleic Acids Research*, 26(1):73–79, 1998.

[26] G. B. Cohen, R. Ren, and D. Baltimore. Modular binding domains in signal transduction proteins. *Cell*, 80(2):237–248, 1995.

[27] M. Curti, P. Degano, and C.T. Baldari. Casual calculus for biochemical modeling. In *Proceedings of the first international workshop on Computational Methods in Systems Biology*, Lecture Notes in Computer Science. Springer-Verlag, 2003. Accepted for publication.

[28] A. Dang, J. A. Frost, and M. H. Cobb. The MEK1 proline-rich insert is required for efficient activation of the mitogen-activated protein kinases ERK1 and ERK2 in mammalian cells. *Journal of Biological Chemistry*, 273(31):19909–19913, 1998.

[29] P. Dhaeseleer, S. Liang, and R. Somogyi. Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707–726, 2000.

[30] N. Dhanasekaran and P. E. Reddy. Signaling by dual specificity kinases. *Oncogene*, 17(11):1447–1455, 1998.

[31] N. S. Duesbery, C. P. Webb, S. H. Leppla, V. M. Gordon, K. R. Klimpel, T. D. Copeland, N. G. Ahn, M. K. Oskarsson, K. Fukasawa, K. D. Paull, and Vande G. F. Woude. Proteolytic inactivation of MAP-kinase-kinase by anthrax lethal factor. *Science*, 280(5364):734–737, 1998.

[32] K. Eilbeck, A. Brass, N. Paton, and C. Hodgman. Interact: an object oriented protein-protein interaction database. In *Intelligent Systems for Molecular Biology*, volume 7, pages 87–94, Palo Alto, 1999. AAAI Press.

[33] J. M. English, G. Pearson, R. Baer, and M. H. Cobb. Identification of substrates and regulators of the mitogen-activated protein kinase ERK5 using chimeric protein kinases. *Journal of Biological Chemistry*, 273(7):3854–3860, 1998.

[34] M. Ettinger. The complexity of comparing reaction systems. *Bioinformatics*, 18(3):465–469, 2002.

[35] A. Finney, H. Sauro, M. Hucka, and H. Bolouri. An xml-based model description language for systems biology simulations. Technical report, California Institute of Technology, September 2000. Technical report.

[36] W. Fontana and L. W. Buss. The barrier of objects: From dynamical systems to bounded organizations. In J. Casti and A. Karlqvist, editors, *Boundaries and Barriers*, pages 56–116. Addison-Wesley, 1996.

[37] C. V. Forst and K. A. Schulten. Evolution of metabolisms: a new method for the comparison of metabolic pathways using genomics information. *Journal of Computational Biology*, 6(3–4):343–360, 1999.

[38] C. Friedman, P. Kra, H. Yu, M. Krauthammer, and A. Rzhetsky. Genies: a natural-language processing system for the extraction of molecular pathways from journal articles. *Bioinformatics*, 17:S74–S82, 2001.

[39] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3–4):601–620, 2000.

[40] A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Mollecular Biology of the Cell*, 11(12):4241–4257, 2000.

[41] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *Journal of Physical Chemistry A*, 104:1876–1889, 2000.

[42] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[43] C. Girault and R. Valk. *Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications*. Springer-Verlag, 2002.

[44] A. Goldbeter. *Biochemical Oscillations and Cellular Rhythms*. Cambridge University Press, Cambridge, 1996.

[45] P. J. E. Goss and J. Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic petri nets. *Proceedings of the National Academy of Sciences USA*, 95(12):6750–6755, 1998.

[46] P. J. E. Goss and J. Peccoud. Analysis of the stabilizing effect of rom on the genetic network controlling cole1 plasmid replication. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 4, pages 65–76, Singapore, 1999. World Scientific Press.

[47] K. Goto and C. H. Johnson. Is the cell division cycle gated by a circadian clock? the case of chlamydomonas reinhardtii. *Journal of Cell Biology*, 129(4):1061–1069, 1995.

[48] M. C. Gustin, J. Albertyn, M. Alexander, and K. Davenport. MAP kinase pathways in the yeast Saccharomyces cerevisiae. *Microbiology and Molelcular Biology Reviewes*, 62(4):1264–1300, 1998.

[49] M. B. Hallett. The unpredictability of cellular behavior: trivial or of fundamental importance to cell biology? *Perspectives in Biological Medicine*, 33(1):110–119, 1989.

[50] D. Harel and E. Gery. Executable object modeling with statecharts. *IEEE Computer*, 30(7):31–42, 1997.

[51] L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray. From molecular to modular cell biology. *Nature*, 402:C47–C52, 1999.

[52] J. M. Haugh and D. A. Lauffenburger. Analysis of receptor internalization as a mechanism for modulating signal transduction. *Journal of Theorerical Biology*, 195(2):187–218, 1998.

[53] K. R. Heidtke and S. Schulze-Kremer. Deriving simulation models from a molecular biology knowledge base. In *Proceeding of the 4th Workshop on Engineering Problems for Qualitative Reasoning of the 16th International Joint Conference on Artificial Intelligence*, 1999.

[54] C. H. Heldin. Dimerization of cell surface receptors in signal transduction. *Cell*, 80(2):213–223, 1995.

[55] C-H. Heldin and M. Purton, editors. *Signal Transduction*. Chapman and Hall, London, 1996. Modular Texts in Molecular and Cell Biology 1.

[56] R. Hofestadt and S. Thelen. Quantitative modeling of biochemical networks. *In Silico Biology*, 1(1):39–53, 1998.

[57] M. Holcombe and A. Bell. Computational models of immunological pathways. In M. Holcombe and R. Paton, editors, *Information Processing in Cells and Tissues: Proceeding of IPCAT '97*, pages 213–226, New York, 1998. Plenum Press.

[58] T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, Y. D. He, M. J. Kidd, A. M. King, M. R. Meyer, D. Slade, P. Y. Lum, S. B. Stepaniants, D. D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S. H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–126, 2000.

[59] T. Hunter. Protein kinases and phosphatases: the yin and yang of protein phosphorylation and signaling. *Cell*, 80(2):225–236, 1995.

[60] Lee I. T, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J. B. Tagne, T. L. Volkert, E. Fraenkel, D. K. Gifford, and R. A. Young. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 298(5594):799–804, 2002.

[61] T. Igarashi and T. Kaminuma. Development of a cell signalling networks database. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Proccedings of the Pacific Symposium of Biocomputing '97*, pages 187–197, Singapore, 1997. World Scientific Press.

[62] J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv, and N. Barkai. Revealing modular organization in the yeast transcriptional network. *Nature Genetics*, 31(4):370–377, 2002.

[63] C. H. Johnson and S. S. Golden. Circadian programs in cyanobacteria: adaptiveness and mechanism. *Annual Review in Microbiology*, 53:389–409, 1999.

[64] N. Kam, I.R. Cohen, and D. Harel. The immune system as a reactive system: Modeling t cell activation with statecharts. *Bulletin of Mathematical Biology*, 2002. to appear. An extended abstract of this paper appeared in the Proceeding of the Symposia on Human-Centric Computing Languages and Environments, pages 15-22, Stresa, Italy, September 2001.

[65] N. Kam, D. Harel, and I.R. Cohen. Modeling biological reactivity: Statecharts vs. boolean logic. In *Proceeding of the Second International Conference on Systems Biology*, Pasadena, CA, 2001.

[66] M. Kanehisa, S. Goto, S. Kawashima, and A. Nakaya. The KEGG databases at genomenet. *Nucleic Acids Research*, 30(1):42–46, 2002.

[67] P. D. Karp. An ontology for biological function based on molecular interactions. *Bioinformatics*, 16(3):269–285, 2000.

[68] P. D. Karp, M. Krummenacker, S. Paley, and J. Wagg. Integrated pathway/genome databases and their role in drug discovery. *Trends in Biotechnology*, 17(7):275–281, 1999.

[69] T. Kazic. Semiotes: a semantics for sharing. *Bioinformatics*, 16(12):1129–1144, 2000.

[70] A. V. Khokhlatchev, B. Canagarajah, J. Wilsbacher, M. Robinson, M. Atkinson, E. Goldsmith, and M. H. Cobb. Phosphorylation of the MAP kinase ERK2 promotes its homodimerization and nuclear translocation. *Cell*, 93(4):605–615, 1998.

[71] A. M. Kierzek. Stocks: Stochastic kinetic simulations of biochemical systems with gillespie algorithm. *Bioinformatics*, 18(3):470–481, 2002.

[72] H. Kitano. Computational systems biology. *Nature*, 420:206–210, 2002.

[73] R. Kuffner, R. Zimmer, and T. Lengauer. Pathway analysis in metabolic databases via differential metabolic display. *Bioinformatics*, 16(9):825–836, 2000.

[74] K.M. Kyoda, M. Muraki, and H. Kitano. Construction of a generalized simulator for multi-cellular organisms and its application to smad signal transduction. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 5, pages 317–328, Singapore, 2000. World Scientific Press.

[75] D. A. Lashkari, J. L. DeRisi, J. H. McCusker, A. F. Namath, C. Gentile, S. Y. Hwang SY, P. O. Brown, and R. W. Davis. Yeast microarrays for genome wide parallel genetic and gene expression analysis. *Proceedings of the National Academy of Sciences USA*, 94(24):13057–13062, 1997.

[76] A. Levchenko, J. Bruck, and P.W. Sternberg. Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties. *Proceedings of the National Academy of Sciences USA*, 97:5818–5823, 2000.

[77] T. S. Lewis, P. S. Shapiro, and N. G. Ahn. Signal transduction through MAP kinase cascades. *Advance in Cancer Research*, 74:49–139, 1998.

[78] H. Lodish, A. Berk, S. L. Zipursky, P. Matsudaira, D. Baltimore, and J. E. Darnell. *Molecular Cell Biology. 4th Edition.* W. H. Freeman, 2000.

[79] S. J. Mansour, J. M. Candia, J. E. Matsuura, M. C. Manning, and N. G. Ahn. Interdependent domains controlling the enzymatic activity of mitogen-activated protein kinase kinase 1. *Biochemistry*, 35(48):15529–15536, 1996.

[80] H. Matsuno, A. Doi, M. Nagasaki, and S. Miyano. Hybrid petri net representation of gene regulatory network. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 5, pages 341–352, Singapore, 2000. World Scientific Press.

[81] H. Matsuno, R. Murakani, R. Yamane, N. Yamasaki, S. Fujita, H. Yoshimori, and S. Miyano. Boundary formation by notch signaling in drosophila multicellular systems: Experimental observations and gene network modeling by genomic object net. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing. In press.*, Singapore, 2003. World Scientific Press.

[82] B. J. Mayer, H. Hirai, and R. Sakai. Evidence that SH2 domains promote processive phosphorylation by protein-tyrosine kinases. *Current Biology*, 5(3):296–305, 1995.

[83] H. H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Sciences USA*, 94(3):814–819, 1997.

[84] H. H. McAdams and L. Shapiro. Circuit simulation of genetic networks. *Science*, 269(5224):650–656, 1995.

[85] R. Melendez, E. Melendez-Hevia, and M. Cascante. How did glycogen structure evolve to satisfy the requirement for rapid mobilization of glucose? a problem of physical constraints in structure building. *Journal of Molecular Evolution*, 45(4):446–455, 1997.

[86] L. Mendoza, D. Thieffry, and E. R. Alvarez-Buylla. Genetic control of flower morphogenesis in arabidopsis thaliana: a logical analysis. *Bioinformatics*, 15(7–8):593–606, 1999.

[87] H. W. Mewes, K. Albermann, K. Heumann, S. Liebl, and F. Pfeiffer. MIPS: a database for protein sequences, homology data and yeast genome information. *Nucleic Acids Research*, 25(1):28–30, 1997.

[88] R. Milner. π-nets: a graphical form of the π-calculus. In D. Sannella, editor, *Proceedings of the Fifth European Symposion on Programming (ESOP '94)*, volume 788 of *Lecture Notes in Computer Science*, pages 26–42, Berlin, 1994. Springer-Verlag.

[89] R. Milner. *Communicating and Mobile Systems: The π-Calculus*. Cambridge University Press, Cambridge, 1999.

[90] T. Mori, B. Binder, and C. H. Johnson. Circadian gating of cell division in cyanobacteria growing with average doubling times of less than 24 hours. *Proceedings of the National Academy of Sciences USA*, 93(19):10183–10188, 1996.

[91] C. J. Morton-Firth and D. Bray. Predicting temporal fluctuations in an intracellular signalling pathway. *Journal of Theoretical Biology*, 192(1):117–128, 1998.

[92] Y. Moscowitz and E. Shapiro. On the structural simplicity of machines and languages. *Annals of Mathematics and Artificial Intelligence*, 15:379–405, 1995.

[93] P. Nielsen, D. Bullivant, C. Lloyd, D. Nickerson, S. Lett, K. Jim, and P. Noble. The CellML modeling language. http://www.cellml.org/public/specification/, 2000.

[94] C. Nottegar, C. Priami, and P. Degano. Semantic-driven performance evaluation. *Lecture Notes in Computer Science*, 1577:204–218, 1999.

[95] H. Ogata, S. Goto, W. Fujibuchi, and M. Kanehisa. Computation with the kegg pathway database. *Biosystems*, 47(1–2):119–128, 1998.

[96] F. Orava and J. Parrow. An algebraic verification of a mobile network. *Formal Aspects of Computing*, 4(6):497–543, 1992.

[97] A. Pagnoni and A. Visconti. Detection and analysis of unexpected state components in biological systems. In *Proceedings of the first international workshop on Computational Methods in Systems Biology*, Lecture Notes in Computer Science. Springer-Verlag, 2003. Accepted for publication.

[98] J. Parrow. An introduction to the π-calculus. In J. A. Bergstra, A. Ponse, and S. A. Smolka, editors, *Handbook of Process Algebra*, chapter 8, pages 479–543. Elsevier Science, 2001.

[99] T. Pawson and P. Nash. Protein-protein interactions define specificity in signal transduction. *Genes and Development*, 14:1027–1047, 2000.

[100] D. Pe'er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17:S215–S224, 2001.

[101] D. Pe'er, A. Regev, and A. Tanay. Minreg: Inferring an active regulator set. *Bioinformatics*, 18:S258–S267, 2002.

[102] C. Priami. Stochastic π-calculus. *The Computer Journal*, 38(6):578–589, 1995.

[103] T. Quatrani. *Visual Modeling with Rational Rose 2000 and UML*. Addison-Wesley, 2000.

[104] Milner R. Bigraphical reactive systems. In *Proceedings of the 12th International Conference on Concurrency Theory (CONCUR 2001)*, volume 2154 of *Lecture Notes in Computer Science*, pages 16–35. Springer-Verlag, 2001.

[105] J. C. Rain, L. Selig, H. De Reuse, V. Battaglia, C. Reverdy, S. Simon, G. Lenzen, F. Petel, J. Wojcik, V. Schachter, Y. Chemama, A. Labigne, and P. Legrain. The protein–protein interaction map of helicobacter pylori. *Nature*, 409:211–215, 2001.

[106] A. Regev. The full code for this model and the BioSpi surface ASCII syntax are available in the attached disk.

[107] A. Regev. Representation and simulation of molecular pathways in the stochastic pi-calculus. In R. Gauges, C. van Gend, and U. Kummer, editors, *Proceedings of the 2nd workshop on computation of biochemical pathways and genetic networks*, pages 109–115, Berlin, 2001. Logos.

[108] A. Regev, K. Panina, W. Silverman, L. Cardelli, and E. Shapiro. Bioambients - a calculus for biological compartments. manuscript in preparation.

[109] A. Regev, C. Priami, E. Shapiro, and W. Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80:25–31, 2001.

[110] A. Regev and E. Shapiro. Cellular abstractions. *Nature*, 419:343, 2002.

[111] A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 6, pages 459–470, Singapore, 2001. World Scientific Press.

[112] W. Reisig and G. Rozenberg. Informal introduction to petri nets. *Lecture Notes in Computer Science: Lectures on Petri Nets I: Basic Models*, 1491, 1998.

[113] C. J. Roberts, B. Nelson, M. J. Marton, R. Stoughton, M. R. Meyer, H. A. Bennett, Y. D. He, H. Dai, W. L. Walker, T. R. Hughes, M. Tyers, C. Boone, and S. H. Friend. Signaling and

circuitry of multiple mapk pathways revealed by a matrix of global gene expression profiles. *Science*, 287(5454):873–880, 2000.

[114] M. J. Robinson, M. Cheng, A. Khokhlatchev, D. Ebert, N. Ahn, K. L. Guan, B. Stein, E. Goldsmith, and M. H. Cobb. Contributions of the mitogen-activated protein (MAP) kinase backbone and phosphorylation loop to specificity. *Journal of Biological Chemistry*, 271(47):29734–29739, 1996.

[115] M. G. Samsonova and V. N. Serov. Network: an interactive interface to the tools for analysis of genetic network structure and dynamics. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 4, pages 102–111, Singapore, 1999. World Scientific Press.

[116] L. Sanchez, J. van Helden, and D. Thieffry. Establishement of the dorso-ventral pattern during embryonic development of drosophila melanogasater: a logical analysis. *Journal of Theoretical Biology*, 189(4):377–389, 1997.

[117] H. M. Sauro. Scamp: a general-purpose simulator and metabolic control analysis program. *Computer Applications in Bioscience*, 9(4):441–450, 1993.

[118] H. J. Schaeffer, A. D. Catling, S. T. Eblen, L. S. Collier, A. Krauss, and M. J. Weber. Mp1: A mek binding partner that enhances enzymatic activitation of the map kinase cascade. *Science*, 281:1668–1671, 1998.

[119] T. K. Schlesinger, G. R. Fanger, T. Yujiri, and G. L. Johnson. The TAO of MEKK. *Frontiers in Bioscience*, 3:D1181–D1186, 1998.

[120] S. Schuster, D. A. Fell, and T. Dandekar. A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks. *Nature Biotechnology*, 18(3):326–332, 2000.

[121] M. W. Schwartz, S. C. Woods, D. Porte, R. J. Seeley, and D. G. Baskin. Central nervous system control of food intake. *Nature*, 404:661–671, 2000.

[122] E. Segal*, M. Shapira, A. Regev*, D. Pe'er*, D. Koller, and N. Friedman. Module networks: Reconstruction of regulatory modules and their control programs from gene expression data. *Nature Genetics*, 34:166–176, 2003. (equal contributors).

[123] E. Selkov, Y. Grechkin, N. Mikhailova, and E. Selkov. Mpw: the metabolic pathways database. *Nucleic Acids Research*, 26(1):43–45, 1998.

[124] A. Shalapyonok, R. J. Olson, and L. S. Shalapyonok. Ultradian growth in prochlorococcus spp. *Applied and environmental microbiology*, 64(3):1066–1069, 1998.

[125] E. Shapiro. Concurrent prolog: a progress report. In E. Shapiro, editor, *Concurrent Prolog (vol. I)*, pages 157–187. MIT Press, Cambridge, Massachusetts, 1987.

[126] P. S. Shapiro and N. G. Ahn. Feedback regulation of Raf-1 and mitogen-activated protein kinase (MAP) kinase kinases 1 and 2 by kinase phosphatase (MKP-1). *Journal of Biological Chemistry*, 273(3):1788–1793, 1998.

[127] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature Genetics*, 31:64–68, 2002.

[128] W. Silverman, M. Hirsch, A. Houri, and E. Shapiro. The Logix system user manual, version 1.21. In E. Shapiro, editor, *Concurrent Prolog (vol. II)*, pages 46–78. MIT Press, Cambridge, Massachusetts, 1987.

[129] T. F. Smith. Functional genomics - bioinformatics is ready for the challenge. *Trends in Genetics*, 14(7):291–293, 1998.

[130] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. C. Anders, M. B. Eisen, , P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mollecular Biology of the Cell*, 9(12):3273–3297, 1998.

[131] P. W. Sternberg and J. Alberola-Ila. Conspiracy theory: and do not act alone. *Cell*, 95(4):447–450, 1998.

[132] S. Stewart, M. Sundaram, Y. Zhang, J. Lee, M. Han, and K. L. Guan. Kinase suppressor of Ras forms a multiprotein signaling complex and modulates localization. *Molecular and Cellular Biology*, 19(8):5523–5534, 1999.

[133] L. Stryer. *Biochemistry*. W.H. Freeman, 1995.

[134] P. S. Swain, M. B. Elowitz, and E. D. Siggia. Intrinsic and extrinsic contributions to stochasticity in gene expression. *Proceedings of the National Academy of Sciences USA*, 99(20):12795–12800, 2002.

[135] Z. Szallasi. Genetic network analysis in light of massively parallel biological data. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 4, pages 5–16, Singapore, 1999. World Scientific Press.

[136] A. Tanay and R. Shamir. Computational expansion of genetic networks. *Bioinformatics*, 17:S270–S278, 2001.

[137] T. Tanoue, M. Adachi, T. Moriguchi, and E. Nishida. A conserved docking motif in MAP kinases common to substrates, activators and regulators. *Nature Cell Biology*, 2(2):110–116, 2000.

[138] C. D. Thanos, K. E. Goodwill, and J. U. Bowie. Oligomeric structure of the human EphB2 receptor domain. *Science*, 283(5403):833–836, 1999.

[139] D. Thieffry and D. Romero. The modularity of biological regulatory networks. *Biosystems*, 50(1):49–59, 1999.

[140] D. Thieffry and R. Thomas. Qualitative analysis of gene networks. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 3, pages 77–88, Singapore, 1998. World Scientific Press.

[141] N. S. Tolwinski, P. S. Shapiro, S. Goueli, and N. G. Ahn. Nuclear localization of mitogen-activated protein kinase kinase 1 (MKK1) is promoted by serum stimulation and G2 progression. requirement for phosphorylation at the activation lip and signaling downstream of MKK. *Journal of Biological Chemistry*, 274(10):6168–6174, 1999.

[142] M. Tomita, K. Hashimoto, K. Takahashi, T. S. Shimizu, Y. Matsuzaki, F. Miyoshi, K. Saito, S. Tanida, K. Yugi, J. C. Venter, and C. A. Hutchison 3rd. E-cell: software environment for whole-cell simulation. *Bioinformatics*, 15(1):72–84, 1999.

[143] J. van Helden, A. Naim, R. Mancuso, M. Eldridge, L. Wernisch, D. Gilbert D, and S. J. Wodak. Representing and analysing molecular and cellular function using the computer. *Biological Chemistry*, 381(9–10):921–935, 2000.

[144] D. Vaulot, D. Marie, R. J. Olson, and S. W. Chisholm. Growth of prochlorococcus, a photosynthetic prokaryote un the equatorial pacific ocean. *Science*, 268:1480–1482, 1995.

[145] B. Victor and F. Moller. The Mobility Workbench — a tool for the π-calculus. In David Dill, editor, *CAV'94: Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 428–440. Springer-Verlag, 1994.

[146] J. M. Vilar, H. Y. Kueh, N. Barkai, and S. Leibler. Mechanisms of noise-resistance in genetic oscillators. *Proceedings of the National Academy of Sciences USA*, 99(9):5988–5992, 2002.

[147] D. Voet and J. Voet. *Biochemistry. 2nd edition.* John Wiley and sons, 1995.

[148] A. J. Whitmarsh, J. Cavanagh, C. Tournier, J. Yasuda, and R. J. Davis. A mammalian scaffold complex that selectively mediates MAP kinase activation. *Science*, 281(5383):1671–1674, 1998.

[149] C. Widmann, S. Gibson, M. B. Jarpe, and G. L. Johnson. Mitogen-activated protein kinase: conservation of a three-kinase module from yeast to human. *Physiological Reviews*, 79(1):143–180, 1999.

[150] J. L. Wilsbacher, E. J. Goldsmith, and M. H. Cobb. Phosphorylation of MAP kinases by MAP/ERK involves multiple regions of MAP kinases. *Journal of Biological Chemistry*, 274(24):16988–16994, 1999.

[151] E. Wingender, X. Chen, E. Fricke, R. Geffers, R. Hehl, I. Liebich, M. Krull M, V. Matys, H. Michael, R. Ohnhauser, M. Pruss, F. Schacherer, S. Thiele, and S. Urbach. The transfac system on gene expression regulation. *Nucleic Acids Research*, 29(1):281–283, 2001.

[152] I. Xenarios, E. Fernandez E, L. Salwinski, X. J. Duan, M. J. Thompson, E. M. Marcotte, and D. Eisenberg. Dip: the database of interacting proteins: 2001 update. *Nucleic Acids Research*, 29(1):239–241, 2001.

[153] T-M. Yi, Y. Huang, M. I. Simon, and J. Doyle. Robust perfect adaptation in bacterial chemotaxis through integral feedback control. *Proceedings of the National Academy of Sciences USA*, 97(9):4649–4653, 2000.

[154] S. Zhou, K. L. Carraway 3rd, M. J. Eck, S. C. Harrison, R. A. Feldman, M. Mohammadi, J. Schlessinger, S. R. Hubbard, D. P. Smith, and C. Eng. Catalytic specificity of protein-tyrosine kinases is critical for selective signalling. *Nature*, 373(6514):536–539, 1995.

תוכן העניינים

עבודה זו נעשתה בהדרכת

פרופ׳ חוה יבלונקה
פרופ׳ אהוד שפירא (מכון וייצמן)

ביולוגיה חישובית מערכתית:
תחשיב לידע ביומולקולרי

חיבור לשם קבלת התואר "דוקטור לפילוסופיה"
מאת

אביב רגב

הוגש לסנאט אוניברסיטת תל אביב

כסלו, תשס"ג