Modelling and Analyzing Systems Biology Using Process Algebra
Min Zhang

To cite this version:
Modélisation et analyse de processus biologiques dans des algèbres de processus

présentée et soutenue publiquement
par
Min Zhang

le 27ème avril 2007

devant le jury composé de

M. Pierre-Louis Curien directeur
M. Vincent Danos
M. Yuxi Fu
M. François Fages rapporteur
M. Cosimo Laneve
M. Vincent Schächter rapporteur
Abstract

The focus of this thesis is on modelling and analyzing systems biology using process algebra. We apply three process calculi into systems biology: the $\pi$-calculus and a variant, the $I\pi$-calculus; the $\kappa$-calculus and its finer-grained language, the $m\kappa$-calculus; and the bigraphical reactive systems.

There are three parts of my thesis. In Chapter 3 of the thesis we introduce the signal transduction with aberrance. A new extension of the $\pi$-calculus, the $I\pi$-calculus, is introduced to model signal transduction with aberrance. The calculus is obtained by adding two aberrant actions into the $\pi$-calculus. It is well-defined and biologically visible.

The $I\pi$-calculus shows its expressive capability. However, one may need to record more information about its terms in the process of simulation, especially in the simulation of aberrant biochemical processes. Therefore, two auxiliary systems, a tag system and a typing system, are introduced to help understanding the $I\pi$-calculus model. The tag system is more intuitive. But it may be redundant in the recordings of information of terms. The simple typing system, however, is enough to deal with it. We show that the tag system is equal to the typing system in terms of expressive power.

In Chapter 4 of the thesis we propose a rigorous account of self-assembly in the protein-protein language, $\kappa$-calculus introduced by Vincent Danos and Cosimo Laneve. We make use of reversible rules to embed the $\kappa$-calculus into a finer-grained language, the $m\kappa$-calculus. We prove that this simulation is correct mathematically.

In Chapter 5 of the thesis we use bigraphs to model and analyze systems biology. First we give an example to show how to model the biochemical processes using Bigraphical reactive systems (BRSs for short). We take the normal ras activation as our instance. Then the expressive power of the bigraphical models is discussed. We indicate how the $\kappa$-calculus, the protein-protein language, can be translated into BRSs by one example, which indicates that BRSs can be a suitable model in biological studying as well.

In summary, we give three process calculi to model systems biology. We extend the $\pi$-calculus to model the aberrant signal transduction; prove the correctness of self-assembly in $\kappa$-calculus; and make an attempt of modelling ras activation using BRSs. These results lay out some foundations for future interdisciplinary study of systems biology and process algebra. They also highlight the robustness of process algebra in modelling and analyzing systems biology.
Acknowledgements

I would like to express my gratitude to Pierre-louis Curien, my supervisor in France, for his inspiration, guidance, and encouragement. Without his generosity and unconditional support, this thesis would not have been possible. My hearty gratitude and appreciation are also addressed to Fu Yuxi, my supervisor in China for his encouragement, help and kind support. Both of them were always willing to discuss the problems that I encountered in my research and my life. From them I have received invaluable help and advice.

I am very indebted to Vincent Danos. His intelligence and enthusiasm had a substantial influence on my research interests in the period of my Ph.D. study. I have learned much from him about the ways of doing research and the style of presenting it.

I am also very grateful to Wang Geping, my Masters thesis supervisor, for having introduced me to the field of process algebra.

I owe a lot to Chen Yixiang, Chen Yijia, Dong Xiaoju and Deng Yuxin for their help in correcting some parts of my thesis.

I had much pleasure of staying at BASICS Lab and PPS Lab. Both labs have offered creative and pleasant working atmosphere. I would like to thank all the past and current members for their friendship and interesting discussions. They are: François Maurel, Raphaël Montelatici, Stéphane Lengrand, Sylvain Lebresne, Emmanuel Beffara of PPS Lab, and Li yang, Gu Yonggen, Xu Xian, Zhu Han, Cai Xiaojuan, Long Huan, Cai Xuan, Long Huiyun in BASICS Lab.

I appreciated the stimulating discussions with the group of systems biology of BASICS Lab and Bio-x Lab at Shanghai Jiao Tong Univeristy. I thank particularly my colleagues Li Guoqiang and Li Dan for many discussions and nice collaboration. I thank Samuel Hym, Fabien Tarissan , Li Zheng, Séverine Maingaud for their help in my life in France.

I would like to thank all my friends in China. Their encouragement made me both brave and optimistic.

I would also like to thank all my friends in France. They have made my time in Paris both fruitful and enjoyable.

My special gratitude goes to my family for their unfailing support.
Main Notations

Below are the important notations used in this thesis, with the section number of their first appearance.

**Metavariables**
- \(a, b, \cdots\) actions 2.1.1
- \(fn(\cdot)\) free names 2.2.1

**Process constructions**
- \(0\) inaction 2.1.1
- \(\pi\) prefix 2.1.1
- \(P_1 + P_2\) nondeterministic choice 2.1.1
- \(P_1 | P_2\) parallel composition 2.1.1
- \((\nu x)P\) restriction 2.1.1
- \(A(\rho)\) proteins 2.2.1
- \(S, S'\) solutions 2.2.1
- \(\mathcal{S}\) reaction systems 2.2.1
- \(\Gamma, \Delta, \cdots\) environments 3.5.1
- \(I, J, \cdots\) interfaces 2.3.1
- \(K\) constant 3.3.1
- \(K[\tilde{a}]\) constant application 3.3.1
- \(\tau_a, \tau_b, \cdots\) tags of actions 3.4.1
- \(I_P, I_Q, \cdots\) tags of processes 3.4.1
- \(M, N, \cdots\) terms 3.5.1
- \([R]_g\) graphic-on-sites 4.3.1
- \(K, L, \cdots\) controls 5.1.1
Miscellaneous symbols

\[ \begin{array}{ll}
\& & \text{suicide capability} & 3.3.1 \\
\# & \text{propagation capability} & 3.3.1 \\
\sqcup & \text{disjoint union} & 3.4.1 \\
\kappa & \kappa\text{-reactions} & 4.3.1 \\
[\cdot]_{m} & \text{translation from } \kappa \text{ to } m\kappa & 4.3.1 \\
\square & \text{bigraphs} & 5.1.1 \\
\end{array} \]

Relations

\[ \begin{array}{ll}
\rightarrow & \text{reduction rule} & 2.1.1 \\
\leq & \text{growth relation} & 2.2.1 \\
\rightarrow_{\kappa} & \kappa\text{-transition} & 2.2.1 \\
\models & \text{matching} & 2.2.1 \\
\vdash & \text{type assertions} & 3.5.1 \\
\succ & \text{signal ordering} & 4.3.1 \\
\models & \text{reversible translations} & 4.3.1 \\
\equiv & \text{structural congruence} & 2.1.1 \\
\rightarrow_{c} & \text{translation relation} & 4.4 \\
\rightarrow_{m\kappa} & m\kappa\text{-transition} & 4.4 \\
\end{array} \]
# Contents

Abstract i  
Acknowledgements ii  
Main Notations iii  

1 Introduction 1  
1.1 Background 1  
1.2 Objectives of Thesis 4  
1.3 Some Provisos 6  
1.4 Outline of the Thesis 6  

2 Preliminaries 9  
2.1 The $\pi$-calculus 9  
\hspace*{3em} 2.1.1 The $\pi$-calculus: Definitions 9  
\hspace*{3em} 2.1.2 A Simple Example 11  
2.2 The $\kappa$-calculus 11  
\hspace*{3em} 2.2.1 The $\kappa$-calculus: Definitions 12  
\hspace*{3em} 2.2.2 Simple Examples 15  
2.3 Bigraphical Reactive Systems 17  
\hspace*{3em} 2.3.1 Bigraphs 17  
\hspace*{3em} 2.3.2 A Simple Example 18  
2.4 Systems Biology 19  
\hspace*{3em} 2.4.1 Signal Transduction 19  
\hspace*{3em} 2.4.2 The Graphical Expression of RTK-MAPK 21  

3 A Calculus For Formal Molecular Processes 23  
3.1 Basic Principles for the Model 23  
3.2 Signal Transduction with Aberrance 26  
3.3 The Interference $\pi$-calculus 27  
\hspace*{3em} 3.3.1 The Interference $\pi$-calculus: Definition 27  
\hspace*{3em} 3.3.2 A Model about ras Activation 30  
3.4 The $I\pi$-calculus with a Tag System 33  
\hspace*{3em} 3.4.1 A Tag System 33
### Indices

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4.2</td>
<td>Tag Systems for Quantitative Analysis?</td>
</tr>
<tr>
<td>3.5</td>
<td>The $\pi$-calculus with a Typing System</td>
</tr>
<tr>
<td>3.5.1</td>
<td>A Typing System</td>
</tr>
<tr>
<td>3.5.2</td>
<td>A Simple Example</td>
</tr>
<tr>
<td>3.6</td>
<td>Comparing Tags and Types</td>
</tr>
<tr>
<td>4</td>
<td>A Language for Formal Proteins</td>
</tr>
<tr>
<td>4.1</td>
<td>The $\kappa$-model of ras Activation</td>
</tr>
<tr>
<td>4.2</td>
<td>The $\mu\kappa$-calculus</td>
</tr>
<tr>
<td>4.3</td>
<td>The Self-assembly Protocol for the $\kappa$-calculus</td>
</tr>
<tr>
<td>4.3.1</td>
<td>The Monotonic Protocol</td>
</tr>
<tr>
<td>4.3.2</td>
<td>A Graphical Explanation of the Protocol</td>
</tr>
<tr>
<td>4.4</td>
<td>Mathematical Properties of Self-assembly</td>
</tr>
<tr>
<td>5</td>
<td>A More General System</td>
</tr>
<tr>
<td>5.1</td>
<td>The Dynamics in Bigraphical Reactive Systems</td>
</tr>
<tr>
<td>5.1.1</td>
<td>The Dynamics of Bigraphs</td>
</tr>
<tr>
<td>5.1.2</td>
<td>The Dynamics of Place Graphs and Link Graphs</td>
</tr>
<tr>
<td>5.2</td>
<td>A Bigraphical Model of ras Activation</td>
</tr>
<tr>
<td>5.3</td>
<td>An Example</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions and Future Work</td>
</tr>
<tr>
<td>A</td>
<td>Some Additional Examples</td>
</tr>
<tr>
<td>A.1</td>
<td>The $\pi$-model with the Tag System about the Protein ras</td>
</tr>
<tr>
<td>A.1.1</td>
<td>The Normal State of ras</td>
</tr>
<tr>
<td>A.1.2</td>
<td>The Aberrant State of ras</td>
</tr>
<tr>
<td>A.2</td>
<td>The Link and Place Graphical Model of ras Activation</td>
</tr>
<tr>
<td>B</td>
<td>Some Proofs in the Thesis</td>
</tr>
<tr>
<td>B.1</td>
<td>The Proof of Proposition 3.4</td>
</tr>
<tr>
<td>B.2</td>
<td>The Proof of Proposition 3.5</td>
</tr>
<tr>
<td>B.3</td>
<td>The Proof of Proposition 3.6</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Concurrency theory, as an area of research in computer science, emerged in the early seventies of the last century. It is concerned with the modelling and verification of concurrent systems which can be viewed as a collection of sequential processes, possibly running on different processors, that interact and exchange results with each other and with the external environment [RDN96]. Process algebra (or process calculi) is a subarea in concurrency theory. Origins are traced back to the early eighties of the twentieth century, and developments since that time are surveyed in [Bae04]. Process algebra is an algebraic approach to the study of concurrent processes. Its tools are algebraic languages for the specification of processes and the formulation of statements about them, together with calculi for the verification of these statements.

Systems biology [Kit01] is the study of an organism, viewed as an integrated and interacting network of genes, proteins and biochemical reactions which give rise to life. Instead of analyzing individual components or aspects of the organism, such as sugar metabolism or a cell nucleus, systems biologists focus on all the components and the interactions among them, all as part of one system [Wol]. These interactions are ultimately responsible for an organisms and functions. For example, the immune system is not the result of a single mechanism or gene. Rather the interactions of numerous genes, proteins, mechanisms and the organisms external environment, produce immune responses to fight infections and diseases.

Due to the nature of biological systems and concurrent systems, several authors have argued that process calculi could be the right abstraction to support dynamic bioinformatics and open new scenarios in the computer science and biology research [Car04e, Car05a, Car05b, DL04, RS, RS02].

1.1 Background

Systems biology has become more and more popular in the last few years. Systems biology is systems-level understanding of biological systems that takes into account complex interactions of gene, protein, and cell elements [BB01, MM87]. It aims
to integrate high-throughput biological studies to understand how biological systems function by studying the relationships and interactions between various parts of a biological system \([\text{LBZ}^{a00}, \text{LMF}^{a00}]\) (e.g. metabolic pathways, organelles, cells, physiological systems, organisms etc.)

In short, systems biology is defined as an approach to biology where organisms and biological processes should be analyzed and described in terms of their components and their interactions in a framework of mathematical models.

Systems biology begins with the insight that biological processes must be understood in terms of the components that participate in the processes, and that the complexity of biological systems makes it difficult to understand the workings of the system by simple qualitative arguments. Mathematically strict models must be formulated. This is required both in order to be able to capture the actual behavior of the system with acceptable precision, but also to be able to analyze the fundamental behavior of the system [Pri05].

There exist some models of the whole system so far. The models may be very simple (Boolean on/off) [RV90, PRS98], or very complex (including detailed descriptions of interactions at a molecular level) [Pau01, Pau02]. The important issue is that it should be possible to analyze the model, either by some mathematical approach, or to simulate it, in order to evaluate its correspondence with the observed facts.

As we mentioned, one important goal of systems biology is to understand life processes in sufficient detail to make predictions about their behavior. How to do it? A general biomolecular system is made up of bio-components which are taken as computational devices. The whole system achieves its function by the interactions among these components. All of these characteristics are analogous to the properties of process algebra.

The term \textit{process algebra} was coined in 1984 by Bergstra and Klop [BK84]. A process algebra is a structure in the sense of universal algebra that satisfies a particular set of axioms. It offers description techniques to model a complex computing system, which is involving communicating and concurrently executing components. It mixes the areas of computer science and discrete maths, including system design notations, logic, concurrency theory, specification and verification, operational semantics, algorithms, complexity theory, and, of course, algebra. It develops a behavioral theory of computing processes and allows for description and verification techniques in the same formal system.

\textit{CCS} (Calculus of Communicating Systems) [Mil89], \textit{CSP} (Communicating Sequential Processes) [Hoa85] and \textit{ACP} (Algebra of Communicating Processes) [BK84, BK92], were proposed in the 1980's for describing and analyzing concurrent systems, and became the most successful process algebras. All of them were built around the central idea of interaction or communication between processes. In these formalisms, complex systems are built from simple subcomponents structurally, by a small set of primitive operators.
CCS [Mil80, Mil89] considers the problems caused by non-determinism. CSP [Hoa85] does away completely with global variables, and adopts the message passing paradigm of communication.

The limitation of these traditional process algebras is that they are not able to effectively specify mobile systems, i.e., systems with a dynamically changing configuration of communication links.

Milner, Parrow and Walker developed the $\pi$-calculus [Mil99, MPW92] on the basis of CCS, which achieves mobility by a powerful name-passing mechanism. The $\pi$-calculus [SW01] aims at the challenge of defining an underlying model, with a small number of basic concepts, in terms of which interactional behavior can be rigorously described.

In a word, the $\pi$-calculus includes the syntax which attempts to systemize descriptive grammar and the semantics which offer explanations of systematic relationship. The dynamic is introduced into the $\pi$-calculus by allowing dynamic creation of processes and for names to be passed among different processes, which is also one of the reasons why the $\pi$-calculus is a suitable model for systems biology.

A variant of the $\pi$-calculus, called the stochastic $\pi$-calculus [Pri95] is developed to be applied in the biological domain [Reg01, PC]. It essentially selects the enabled action to be performed according to the Gillespie algorithm [Gil76, Gil77] developed to simulate chemical reactions. Preliminary results in this field have been obtained in modelling a set of interesting biological systems and some analysis and simulation have been carried out [RSS00, RSS01, PRSS01, Reg01].

Applying existing calculi defined with computer systems to systems biology is a common strategy. However, some researchers have adopted the opposite strategy. They have defined calculi which come from biology so that they are better suited to modelling, analyzing and simulating living systems, for instance, see [Car04a, Car04b, Car04c, Car04d, DL04]. Then we can apply the new family of calculi to computer systems to see whether the bio-mimetic approach can further inspire and enhance our comprehension of how computer artificial systems can be modelled, designed and implemented.

The $\kappa$-calculus [DL04, DK03] is one kind of these process calculi. It aims at idealizing protein-protein interactions, essentially as a particular restricted kind of graph rewriting operating on graphs-on-sites. Biological reactions are modelled by two kinds of rewriting rules: one is monotonic and the other is antimonotonic. The former represents complexation, and the latter represents decomplexation.

Bigraphical reactive systems (BRSs for short) is a model for computer systems with mobile placing and linking [JM04, Mil01a, Mil01b, Mil05a, Mil05b, Mil]. It aims to unify calculi such as the $\pi$-calculus, Petri nets [Mil04] and so on. It models spatial activity as well.

BRSs are graphical models of computation which capture the properties of locality and connectivity [JM03, JM04]. They are reconfigurable as well since the nodes in graphs may represent a great variety of computational objects.
1.2 Objectives of Thesis

This thesis focuses on modelling and analyzing of systems biology. It is very important to make abstractions for modelling and analyzing the dynamic evolution of the systems in time and space. That is, we need a behavioral theory of biomolecular systems. As we have seen, computer and biomolecular systems have some resemblance. For example, they both start from a small set of elementary components. Computers are networked to perform larger and larger computations, and cells form multicellular organisms. Therefore, we believe that we can find an appropriate computing model to deal with the biomolecular systems.

In research of systems biology, the basic studying approach is described by the following steps;

- to build a model of biological system;
- to perform experiments to test this model;
- to modify this model according to the results of experiments;
- to use this model.

Since concurrent processes are analogous to biological processes, it is worthy of making an attempt for modelling systems biology using processes calculi.

In this thesis, we apply three process calculi into systems biology.

(1) The $\pi$-calculus is suitable to model various molecular systems, including transcriptional circuits, signal transduction and metabolic pathways etc. [RS]. However, the modelling of cases with aberrance in molecular systems was not considered so far. We extend this calculus by adding two aberrant actions into the $\pi$-calculus. The new calculus, called the $I\tau$-calculus, is applied to model aberrant biochemical processes. Actually, to study the aberrant biochemical processes is important. For example, aberrant signal transduction is the cause of many diseases challenged by modern medicine, including cancers, inflammatory diseases, cardiovascular disease and neuropsychiatric disorders.

We use the following techniques for investigation in this thesis.

(a) To make an abstraction from the real biological system using our language. As the other processes calculi, our model should include a syntax, a semantics and satisfy some algebraic properties.

(b) To introduce auxiliary systems to help understand the model. Besides making the model more clear, we hope that such auxiliary systems will be useful to prove much stronger properties in the future study, for example, the qualitative analysis, as well.

(c) Our model should be dynamic, i.e. it should be evolutive in view of deeper studying.
(2) The $\kappa$-calculus is a protein-protein language introduced by Vincent Danos [DK03]. We focus on its property of self-assembly. Self-Assembly is a very important property in biology. It is a method of integration in which the components spontaneously assemble, typically by bouncing around in a solution or gas phase until a stable structure of minimum energy is reached. The $\kappa$-calculus captures this property by means of a translation into a finer-grained language, the $m\kappa$-calculus [DL04].

In this thesis, we mainly study the correctness of self-assembly.

(a) The graphical explanation of the monotonic protocol of integration: the graphical explanation makes it easier to understand how to divide one biochemical reaction into several basic reactions (or integrate one biochemical reaction from several basic reactions).

(2) The proof of correctness: we have to prove that the translation process implements higher-level reactions correctly by means of the simple, local interactions of the $m\kappa$-calculus. To prove it, we need to know its mathematical structure and properties.

(3) Bigraphical reactive systems (BRSs for short) were first introduced by Robin Milner etc. [JM03, JM04, Mil, Mil05b, Mil05a, Mil04]. The bigraphical model aims at offering further generality both in the treatment of mobility and in behavioral theory. Therefore, BRSs can be applied into systems biology. As a general model, it not only has general properties of models, but also has its particular properties.

We give one biological example to show the expressive power of the bigraphical models in this thesis.

We argue that processes calculi can provide the much-needed abstraction for biomolecular systems. Based on the studying of these three process calculi in this thesis, we know:

(1) process calculi can be used to model biomolecular systems;

(2) process calculi can simulate the behavior of biomolecular systems;

(3) each process calculus can capture special properties of biomolecular systems;

(4) there exists one general model which includes the other models of the other process calculi;

(5) our models are configurable, extensible according to the different needs of biological studying.
1.3 Some Provisos

In this thesis, we make some tacit assumptions about our models. Actually the real biomolecular systems are very complicated. Our goal is to grasp some special properties of biomolecular systems. If the model is too complicated, we cannot get any good properties. So we need to simplify our theoretical models. The level of abstraction in the models are different. For example, the \( \pi \)-calculus is based on the level of functional domains of proteins, the \( \kappa \)-calculus id based on the level of proteins, and BRSs are based on different levels according to our assumptions.

In this thesis, we follow some principles in order to simplify models.

1. Decision of the level of modelling: this is the first step of modelling. For example, if we work at the level of the functional domains of proteins, our biological reactions are reactions of domains; if we work at the level of proteins, our biological reactions are those of proteins; if we work at the level of complexes, our biological reactions are those of complexes.

2. Simplification of components: once we decide some kind of units as the level of modelling, we take them as primitive processes. We ignore or simplify the smaller units which are lower than this level. We also ignore the other parts in larger units except primitive processes. For example, if proteins are taken as primitive processes, we won’t consider the amino acids which constitute proteins; if the domains of proteins are taken as primitive processes, we regard a protein as a group of its domains. If we focus on complexes of proteins, we regard a complex as a group of proteins, regardless of the domains of proteins.

3. Simplification of reactions: the reactions in biomolecular systems are taken as the binding (or interactions) of computing processes. For example, in the \( \kappa \)-calculus, we take monotonic reactions as bindings of solutions; in the \( \mathcal{I} \pi \)-calculus, we take reactions as interactions between processes; in BRSs, we take reactions as the change of bigraphs.

4. We do not consider the factors of environments, including the temperature, consistency, time and so on. We just consider the possibility of reactions between two (or more) of entities.

1.4 Outline of the Thesis

The material presented in Chapter 2 is meant to prepare the technical development in the rest of the thesis. We introduce some basic notions about process calculi, with the \( \pi \)-calculus, the \( \kappa \)-calculus and bigraphs as our templates. We then focus on the biochemical process; we review signal transduction.
In Chapter 3, we introduce a calculus for formal molecular processes. We focus on the signal transduction with aberrance. The model for normal signal transduction is in [RSS00] [RSS01]. An extended calculus, $I\pi$-calculus, is given to model the signal transduction with aberrance. The calculus is obtained by adding two aberrant actions into the $\pi$-calculus. It is well-defined and biologically visible.

The $I\pi$-calculus shows its capability of description. However, one may need to record more information about its terms in the process of simulation, especially in the simulation of aberrant biochemical processes. Therefore, two auxiliary systems, a tag system and a typing system are introduced to help understand the $I\pi$-calculus model. The tag system is easy to understand, more intuitive. But it would be redundant in the recordings of information of terms. The typing system, however, is simple enough to deal with it.

In the end of this chapter, we show that the tag system is equal to the typing system in terms of expressive power.

In Chapter 4, we introduce the language for formal proteins, $\kappa$-calculus. In the $\kappa$-calculus, reactions are modelled at the proteins level. The $\kappa$-model is well-defined and biological visible. A finer-grained language, $m\kappa$-calculus, is introduced as well. In the $m\kappa$-calculus, reactions are restricted to at most binary interactions.

Self-assembly is an important property in biology. In the $\kappa$-calculus, we propose a rigorous account of self-assembly. We construct one monotonic reversible protocol to embed the $k\alpha$-calculus into a finer-grained language, the $m\kappa\alpha$-calculus. Some mathematical properties are discussed. We prove that this simulation is correct mathematically.

In Chapter 5, we recall the basic informal notions of bigraphical reactive systems [JM04] [JM03]. The example of ras activation is given, which shows that the description capability of bigraphical reactive systems is powerful.

We can use this general model it based on our practical needs. On the one hand, we can model different objects as bigraphs. For instance, we can take proteins as bigraphs if we need to study; we also can take the proteins as nodes if we need to know more about proteins. On the other hand, we can simplify our model. For instance, as one goal of experiments, we just want to know the connection among the proteins, we can only consider the link graph, while if we want to know the information of locality of reactants, we can consider the place graph.

As we have mentioned, bigraphical reactive systems are a general model. And some process calculi can be translated into it. In the end of this chapter, we illustrate through an example how the $\kappa$-calculus, the protein-protein language, can be translated into BRSs, which shows that the bigraphical reactive system is suitable model in biological studying as well.

In Chapter 6, we summarize the contributions of this thesis and discuss some directions for potential future work.

**Provenance of the material** This thesis is partially based on published material (mainly in Chapter 3). The presentation of the $I\pi$-calculus which is for describing
the aberrant signal transduction appeared in [ZLF04]; the simple typing system on this calculus with its properties appeared in [ZLF05]; and an analysis of a biological property for an aberrant signal (the aberrant protein ras) in this calculus appeared in [ZLF06].
Chapter 2

Preliminaries

This chapter introduces some basic notions about process calculi. Process calculi are our language tools to model systems biology. They are used in the following chapters. For more details, see [MPW92, Mil99, DL04, JM04]. Some knowledge about signal transduction in systems biology is introduced as well. Especially, we introduce the well-studied signal transduction, RTK-MAPK. Activation of the protein ras will be our main biological example in this thesis. For more details about signal transduction, see [LBZ00, Pta02, VV95, Wol, Kit01].

2.1 The π-calculus

The π-calculus is a mathematical model of processes whose interconnections change as they interact. We call these processes that change their interconnections structure when they execute mobile processes. A program in this calculus specifies a community of interacting processes.

Intuitively, each computational process is defined by its potential communication activities and may be composed in sequence or in parallel with other processes. Communication occurs via channels, denoted by their names, that represent atomic access capabilities. Computation is modelled as synchronous binary communication between processes over their channels. The only content of messages transmitted in communication is channel name or tuples of channel names, which may be used for further communication.

2.1.1 The π-calculus: Definitions

We presuppose that an infinite set $N$ of names is given. Formally, the π-calculus consists of three components:

- A syntax for writing formal descriptions of a concurrent system;
- An operational semantics consisting of reduction rules, which describe the potential changes of the system induced by communication.
• A set of congruence laws that determine when two syntactic expressions are equivalent.

Processes evolve by performing actions. The capabilities for actions are expressed via Prefixes.

**Definition 2.1 (Syntax)** Prefixes and processes of the π-calculus are given by

\[
\pi \ ::= \ \tau b | a(x) | \tau | a \\
P \ ::= \ 0 | \pi.P | P_1 + P_2 | (P_1 | P_2) | (\nu x)P
\]

In prefixes, \(\tau b\) expresses the action to send the name \(b\) via the name \(a\); and \(a(x)\) expresses the action to receive any name via \(a\); \(\tau\) expresses the communication on channel name \(a\); and \(\tau\) expresses the communication on co-name \(\tau\).

We give a brief interpretation of processes. 0 is inaction; it does nothing. The process \(\pi.P\) has a single capability \(\pi\), moreover, the process \(P\) cannot proceed until the capability \(\pi\) has been exercised. The capabilities of the sum \(P_1 + P_2\) are those of \(P_1\) together with those of \(P_2\). When a sum exercised one of its capabilities, the others are rendered void. In the composition \((P_1 | P_2)\), the components \(P_1\) and \(P_2\) can proceed independently and can interact via shared names. In the restriction \((\nu x)P\), the scope of name \(x\) is restricted to \(P\).

**Definition 2.2 (Structural congruence)** Structural congruence, \(\equiv\), is the smallest congruence on processes that satisfies axioms as follows;

\[
\begin{align*}
P | Q & \equiv Q | P \\
(P | Q) | R & \equiv P | (Q | R) \\
P + Q & \equiv Q + P \\
(P + Q) + R & \equiv P + (Q + R) \\
(\nu a)0 & \equiv 0 \\
(\nu a)(\nu b)P & \equiv (\nu b)(\nu a)P \\
((\nu a)P) | Q & \equiv (\nu a)(P | Q) \quad \text{if} \ a \not\in fn(Q)
\end{align*}
\]

**Definition 2.3 (Semantics)** Reduction rules, \(\rightarrow\), are defined by Table. 2.1.

The rule \(Com-N(communication-names)\) deals with the interaction in which one sends a message with a channel while the other receives a message with the same channel so that they have an interaction. The rule \(Com-SN(communication-single\ name)\) deals with the interaction on the same channel so that they have an interaction though there is nothing to send. The reduction rules are closed under summation (the rule \(Sum\)), composition (the rule \(Comp\)), restriction (the rule \(Res\)) and structural congruence (the rule \(Stc\)).
2.1.2 A Simple Example

We give a simple example to illustrate reduction rules. Consider the following process $Q$

\[
Q = (a(x).b(y).0 + \mathit{dv}.0) \mid \mathit{ct}.0 \mid \mathit{sw}.c(z).\mathit{bu}.0
\]

So, we have $Q \overset{\tau}{\rightarrow} \overset{\tau}{\rightarrow} 0 \mid 0 \mid 0 \equiv 0$. See Table 2.2.

<table>
<thead>
<tr>
<th>Table 2.1: Reduction rules.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Com-N</strong></td>
</tr>
<tr>
<td>$(\pi(b).Q + R_1) \mid (a(x).P + R_2) \overset{\tau}{\rightarrow} Q'[b/x]$</td>
</tr>
<tr>
<td><strong>Com-SN</strong></td>
</tr>
<tr>
<td>$(\sigma.Q + R_1) \mid (a.P + R_2) \overset{\tau}{\rightarrow} Q \mid P$</td>
</tr>
<tr>
<td><strong>Sum</strong></td>
</tr>
<tr>
<td>$P \overset{\tau}{\rightarrow} P'$</td>
</tr>
<tr>
<td>$P + Q \overset{\tau}{\rightarrow} P'$</td>
</tr>
<tr>
<td><strong>Comp</strong></td>
</tr>
<tr>
<td>$P \overset{\tau}{\rightarrow} P'$</td>
</tr>
<tr>
<td>$P \mid Q \overset{\tau}{\rightarrow} P' \mid Q$</td>
</tr>
<tr>
<td><strong>Res</strong></td>
</tr>
<tr>
<td>$P \overset{\tau}{\rightarrow} P'$</td>
</tr>
<tr>
<td>$(\nu a) P \overset{\tau}{\rightarrow} (\nu a) P'$</td>
</tr>
<tr>
<td><strong>Stc</strong></td>
</tr>
<tr>
<td>$Q \equiv P, P \overset{\tau}{\rightarrow} P', P' \equiv Q' \overset{\tau}{\rightarrow} Q'$</td>
</tr>
</tbody>
</table>

Table 2.2: The reduction of $Q$.

2.2 The $\kappa$-calculus

The $\kappa$-calculus was introduced by Vincent Danos and Cosimo Laneve. [DL04]

It is a language of formal proteins. Reactions are modelled at proteins level, bonds
are represented by means of shared names, and reactions are required to satisfy a requirement of monotonicity or antimonotonicity. In this section, we briefly introduce basic notions about the κ-calculus.

### 2.2.1 The κ-calculus: Definitions

We assume an infinite countable set \( \mathcal{P} \) of protein names, an infinite countable set \( \mathcal{E} \) of edge names. We take a signature map \( s \) from \( \mathcal{P} \) to natural numbers \( \mathbb{N} \).

Let \( A, B, \cdots \) range over protein names and \( x, y, \cdots \) range over edge names. For each protein name \( A \), \( s(A) \) is the number of sites of \( A \), and for any \( 1 \leq i \leq s(A) \), the pair \( (A, i) \) will accordingly be called a site of \( A \).

An interface is a partial map from \( \mathbb{N} \) to \( \mathcal{E} \) usually ranged over by \( h, v \) and similar symbols. The domain and range of an interface \( \rho \) will be respectively denoted by \( \text{dom}(\rho) \) and \( \text{ran}(\rho) \), and the set of names free in \( \rho \), written \( \text{fn}(\rho) \), is obtained as \( \text{ran}(\rho) \cap \mathcal{E} \). We will only ever deal with interfaces with finite domain. The empty interface will be denoted \( \emptyset \).

**Definition 2.4 (Solutions)** Solutions of κ-calculus are defined as follows:

\[
S ::= 0 \mid A(\rho) \mid S, S \mid (\text{new } x)(S)
\]

Intuitively, the constructs of the κ-calculus solutions have the following meaning: 0 is the empty solution. The protein \( A(\rho) \) with \( A \in \mathcal{P} \) and \( \rho \) an interface with domain \( s(A) \) is the primitive solution. A group of solutions \( S, S' \) is a complex of simple ones. In the restriction solution \((\text{new } x)(S)\) with \( x \in \mathcal{E} \), the new operator "\text{new}" is a binder and \( S \) is the scope of the binder \((\text{new } x)\).

The set \( \text{fn}(S) \) of free names is defined inductively as follows;

\[
\begin{align*}
\text{fn}(0) & \equiv \emptyset \\
\text{fn}(A(\rho)) & \equiv \text{fn}(\rho) \\
\text{fn}(S, S') & \equiv \text{fn}(S) \cup \text{fn}(S') \\
\text{fn}(x)(S)) & \equiv \text{fn}(S) \setminus \{x\}
\end{align*}
\]

Next we introduce an equivalence relation between solutions, called the structural congruence.

**Definition 2.5** Structural congruence, written \( \equiv \), is the least equivalence closed under syntactic constructions, containing \( \alpha \)-equivalence (injective renaming of bound variables), taking "\( , \)" to be associative (as the choice of symbol suggests) and commutative, with 0 as neutral element, and satisfying the scope laws:

\[
\begin{align*}
(x)(y)(S) & \equiv (y)(x)(S), \\
(x)(S) & \equiv S \quad \text{when } x \notin \text{fn}(S), \\
(x)(S), S' & \equiv (x)(S, S') \quad \text{when } x \notin \text{fn}(S').
\end{align*}
\]
We now construct the growth relation on partial interfaces. This relation is parameterized by a set of names, written $\mathcal{X}$ below, which represent edges grown out by a reaction. It is written $\leq$ and is defined inductively by the clauses given in Table 2.3.

Similarly, we can extend the growth relation to groups of pre-proteins ($A(\rho)$ is a pre-protein if $\rho$ is a partial interface of $A$, namely, $\text{dom}(\rho) \subseteq s(A)$.) as shown in Table 2.4.

<table>
<thead>
<tr>
<th>Table 2.3: The growth relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>create $\frac{x \in \mathcal{X}}{x \leq x'}$</td>
</tr>
<tr>
<td>hv–switch $\frac{\mathcal{X} \├ \mathcal{X} \leq \mathcal{X}}{\mathcal{X} \leq \mathcal{X}}$</td>
</tr>
<tr>
<td>vh–switch $\frac{\mathcal{X} \leq \mathcal{X}}{\mathcal{X} \leq \mathcal{X}}$</td>
</tr>
<tr>
<td>reflex $\frac{\mathcal{X} \cap \text{fn}(\rho) = \emptyset}{\mathcal{X} \leq \rho}$</td>
</tr>
<tr>
<td>sum $\frac{\mathcal{X} \leq \sigma}{\mathcal{X} \leq \sigma'}$</td>
</tr>
<tr>
<td>Table 2.4: Extended growth relation</td>
</tr>
<tr>
<td>--------------------------------</td>
</tr>
<tr>
<td>$\frac{\mathcal{X} \leq T}{\text{nil} \frac{0 \leq 0}{\mathcal{X} \leq 0}}$</td>
</tr>
<tr>
<td>$\frac{\mathcal{X} \leq T}{\text{group } \frac{\mathcal{X} \leq \sigma}{\text{dom}(\sigma) \subseteq s(A)}}$</td>
</tr>
<tr>
<td>$\frac{\mathcal{X} \leq T}{\text{synth } \frac{\text{fn}(\sigma) \subseteq \mathcal{X}}{\text{dom}(\sigma) = s(A)}}$</td>
</tr>
</tbody>
</table>

**Definition 2.6 (graph-likeness)** A solution $S$ is said to be graph-like if:
- free names occur at most twice in $S$;
- binders in $S$ bind either zero or two occurrences.

**Definition 2.7 (Connectedness)** We define inductively when a term is connected:
- if $S$ is connected and $\mathcal{X} = (x)(S)$;
- if $S$ and $S'$ are connected and $\text{fn}(S) \cap \text{fn}(S') \neq \emptyset$ then $S, S'$ is connected;
- if $S$ is connected and $S \equiv T$ then $T$ is connected.

**Definition 2.8** Let $[\cdot]_g$ be the following function from graph-like solutions to graphs with sites:

1. $[A(\rho)]_g$ is the graph with a single node labeled $A$, sites in $\{1, \ldots, s(A)\}$, bound sites $k$ being labeled by $\rho(k)$, and free sites being in the state prescribed by $\rho$;
Given a monotonic reaction $L \rightarrow (\overline{x})R$ is said to be a monotonic reaction if:
- $\overline{x} \models L \leq R$,
- both $L$ and $(\overline{x})R$ are graph-like,
- and $R$ is connected.

$(\overline{x})L \rightarrow R$ is said to be an anti-monotonic reaction if:
- its dual $R \rightarrow (\overline{x})L$ is monotonic.

A reaction which is either monotonic or antimonotonic is called a biological reaction and $L$ and $R$ are referred to respectively as its reactant and product.

Definition 2.10 (matching) Given a monotonic reaction $L \rightarrow (\overline{x})R$, with:
- $L = A_1(\rho_1), \ldots, A_n(\rho_n)$
- and $R = A_1(\sigma_1), \ldots, A_m(\sigma_m)$,
one says that a pair of solutions $S, T$ matches $L \rightarrow (\overline{x})R$, written $S, T \models L \rightarrow (\overline{x})R$, if there exists a renaming $r$ and partial interfaces $\xi_1, \ldots, \xi_m$ such that:

1. for all $i$, $r(\overline{x}) \cap \text{fn}(\xi_i) = \emptyset$,
2. $S = A_1(r \circ \rho_1 + \xi_1), \ldots, A_n(r \circ \rho_n + \xi_n)$
   and $T = (r(\overline{x}))(A_1(r \circ \sigma_1 + \xi_1), \ldots, A_m(r \circ \sigma_m + \xi_m))$.

Matching is defined by symmetry for anti-monotonic rules, that is $S, T \models (\overline{x})L \rightarrow R$ if and only if $T, S \models R \rightarrow (\overline{x})L$.

Definition 2.11 Let $\mathcal{R}$ be a set of biological reactions, the associated $\mathcal{R}$-system is the pair $(\mathcal{S}, \rightarrow)$, where $\mathcal{S}$ is the set of solutions and $\rightarrow_\kappa$, called the transition relation, is the least binary relation over $\mathcal{S}$ such that:

\[
\begin{align*}
\text{mon} & \quad S, T \models L \rightarrow_\kappa (\overline{x})R \in \mathcal{R} & S \rightarrow_\kappa T \\
\text{antimon} & \quad S, T \models (\overline{x})L \rightarrow_\kappa R \in \mathcal{R} & S \rightarrow_\kappa T \\
\text{new} & \quad S \rightarrow_\kappa T & (x)(S) \rightarrow_\kappa (x)(T) \\
\text{group} & \quad S \rightarrow_\kappa T & S, S' \rightarrow_\kappa T, S' \\
\text{struct} & \quad S \equiv S' & S' \rightarrow_\kappa T', T' \equiv T
\end{align*}
\]

In the rest of the thesis, we call solutions in the $\kappa$-calculus $\kappa$-solutions, which are different from solutions in the $m\kappa$-calculus ($m\kappa$-solutions)(see Chapter 4.2).
2.2.2 Simple Examples

We use ovals to represent proteins; the rings on the ovals to represent free sites; dots with links to represent bounded sites. The links with names $x$, $y$ represent edges connecting two sites.

![Figure 2.1: A monotonic $\kappa$-reaction.](image)

**Example 1** Fig. 2.1 shows a monotonic $\kappa$-reaction. The monotonic $\kappa$-reaction in Fig. 2.1 creates a new edge $y$. The formal expression can be written as follows:

$$A(1^x + 2), B(1^x + 2), C(1) \rightarrow (z)(A(1^x + 2), B(1^x + 2^z), C(1^z))$$

![Figure 2.2: An antimonotonic $\kappa$-reaction.](image)

**Example 2** Fig. 2.2 shows an antimonotonic $\kappa$-reaction. The antimonotonic $\kappa$-reaction in Fig. 2.2 dismisses the edge $y$. The formal expression can be written as follows:

$$(xy)A(1^x + 2), B(1^x + 2^y), C(1^y) \rightarrow (x)(A(1^x + 2), B(1^x + 2), C(1))$$

**Example 3** Fig. 2.3 shows a $\kappa$-reaction. The formal expression can be written as follows:

$$A(1^x + 2), B(1^x + 2), C(1) \rightarrow (y)(A(1 + 2), B(1 + 2^y), C(1^y))$$
In fact, this $\kappa$-reaction can be decomposed as a monotonic $\kappa$-reaction followed by an antimonotonic $\kappa$-reaction\cite{2.4}.

\[
A(1^x + 2), B(1^x + 2), C(1) \rightarrow (y)(A(1^x + 2), B(1^x + 2^y), C(1^y))
\]

\[
(x)A(1^x + 2^y), B(1^x + 2^y), C(1) \rightarrow (A(1 + 2), B(1 + 2^y), C(1^y))
\]

Because the intermediate product which this decomposition is making explicit connected, it seems reasonable to consider the sequence as a synchronous composition.

\[
A(1^x + 2^y), B(1^x + 2), C(1^y) \rightarrow (z)(A(1^x + 2), B(1^x + 2^z), C(1^z))
\]

**Example 4** Fig.\cite{2.5} shows an edge-flipping $\kappa$-reaction. The formal expression can be written as follows:

\[
A(1^x + 2^y), B(1^x + 2), C(1^y) \rightarrow (z)(A(1^x + 2), B(1^x + 2^z), C(1^z))
\]
It is different from Example 3. Since there is no free site for $C$ to bind with $B$, this reaction only is decomposed as an antimonotonic $\kappa$-reaction and a monotonic one\textsuperscript{2.6}.

\[
A(1^x + 2^y), B(1^x + 2), C(1^y) \rightarrow (A(1^x + 2), B(1^x + 2), C(1))
\]
\[
A(1^x + 2), B(1^x + 2), C(1) \rightarrow (z)(A(1^x + 2), B(1^x + 2^z), C(1^z))
\]

We notice that there is no intermediate product which is connected.

2.3 Bigraphical Reactive Systems

Bigraphical reactive systems (BRSs) were introduced by Robin Milner and Ole Høgh Jensen [JM03, JM04]. A bigraphical reactive system involves bigraphs; it also allows bigraphs to configure themselves. BRSs are graphical models of computation in which both locality and connectivity are prominent. Actually, BRSs aim to provide a uniform way to model spatially distributed systems that both compute and communicate.

2.3.1 Bigraphs

A bigraph, just as its name implies, involves two graphs. One is the place graph in which the nesting of nodes represents locality. The other is the link graph in which the edges connect nodes.

Bigraphs have the following features. First, nodes may occur inside other nodes in bigraphs, so a bigraph has depth. Second, nodes have ports that may be connected by links, so a bigraph has connectivity. So far, we have two kinds of structure in bigraphs, place graphs and link graphs. A Place graph is the nesting structure of nodes. A Link graph is the linked structure of links which is independent of locality.

Bigraphs have another feature, that is, the notion of holes. The holes in bigraphs denote places at which other bigraphs can be inserted.

Bigraphs are taken as arrows of one kind of precategory. Actually, every bigraph is parametric in general. It has inner faces (written as I) with its parameter(s) and
outer faces (written as J) indicating kinds of hole(s) in which it, in turn, may be replaced.

**Definition 2.12 (Interface)** Interfaces have the form \( m, \overrightarrow{X}, X \), where \( m \) is the depth (the number of sites), \( X \) is the set of names, and \( \overrightarrow{X} = (X_0, X_1, ..., X_{m-1}) \) is a vector of \( m \) disjoint subsets of \( X \) indicating the local names associated with each site. Names in \( X \) but not in \( \overrightarrow{X} \) are global names.

In Fig 2.7, \( l = (3, \{u\}, \{v\}, \{u, v\}) \). Since there are trees in which only three bigraph can be inserted, \( m = 3 \). Since there are two local names \( u \) and \( v \) on two holes respectively. There is no inner name any more, \( \overrightarrow{X} = (\{u\}, \{v\}) \) and \( X = \{u, v\} \). \( J = (1, \{\emptyset\}, \{x, y, z\}) \). Since it can be inserted in the bigger bigraph as a bigraph, \( m = 1 \). Since three names \( x, y \) and \( z \) are free names which can be used to link to another bigraph, \( X = \{x, y, z\} \) and \( \overrightarrow{X} = (\{\emptyset\}) \) because there is no local name.

![Figure 2.7: A simple bigraph.](image)

**2.3.2 A Simple Example**

Fig 2.7 shows a simple bigraph. Each node is assigned a control, such as \( K, L, G \), and \( M \), which tell us what bigraphical reactive system kind of node it is. Each control has an arity, a finite ordinal. For instance, the control \( K \) has arity two. The
names $x$ and $y$ denote links which allow a bigraph to be linked into larger bigraph. The grey box represents a hole where another bigraph may be inserted.

As we have mentioned, this bigraph can be divided into two graphs, the place graph and the link graph; see Fig. 2.8. The place graph and the link graph share a node set, but are otherwise independent structures.

### 2.4 Systems Biology

In this section, we give an informal introduction to signal introduction from the view of biology. Then the well-studied signal transduction, RTK-MAPK, in which the protein ras is activated, is introduced informally. The activation of the protein ras will be taken as our main example in the following chapters.

#### 2.4.1 Signal Transduction

Signal transduction is a biological system at the cellular level. It refers to the movement of signals from outside the cell to inside. The movement of signals can be simple, like that associated with receptor molecules of the acetylcholine class: receptors that constitute channels which, upon ligand interaction, allow signals to be passed in the form of small ion movement, either into or out of the cell. These ion movements result in changes in the electrical potential of the cells that, in turn, propagate the signal along the cell. More complex signal transduction involves the coupling of ligand-receptor interactions to many intracellular events. These events include phosphorylations by tyrosine kinases and (or) serine (threonine) kinases. Protein phosphorylations change enzyme activities and protein conformations. The eventual outcome is an alteration in cellular activity and changes in the program of genes expressed within the responding cells.
Signal transducing receptors are of three general classes:

1. Receptors that penetrate the membrane and have intrinsic enzymatic activity. Receptors that have intrinsic enzymatic activity include those that are tyrosine kinases, tyrosine phosphatases, guanylate cyclases and serine (threonine) kinases. Receptors with intrinsic tyrosine kinase activity are capable of autophosphorylation as well as phosphorylation of other substrates. Additionally, several families of receptors lack intrinsic enzyme activity, yet are coupled to intracellular tyrosine kinases by direct protein-protein interactions.

2. Receptors that are coupled, inside the cell, to GTP-binding and hydrolyzing proteins (termed G-proteins). Receptors of the class that interact with G-proteins all have a structure that is characterized by seven transmembrane spanning domains. These receptors are termed serpentine receptors.

3. Receptors that are found intracellularly and upon ligand binding migrate to the nucleus where the ligand-receptor complex directly affects gene transcription.

In this section, we focus on a well-studied signal transduction, the RTK-MAPK pathway.

We denote Receptor Tyrosine Kinases as RTKs. Proteins encoding RTKs contain four major domains:

- an extracellular ligand binding domain.
- an intracellular tyrosine kinase domain.
- an intracellular regulatory domain.
- a transmembrane domain.

RTK proteins are classified into families based upon structural features in their extracellular portions (as well as the presence or absence of a kinase insert) which include the cysteine rich domains, immunoglobulin-like domains, leucine-rich domains, Kringle domains, cadherin domains, fibronectin type III repeats, discoidin I-like domains, acidic domains, and EGF-like domains. Based upon the presence of these various extracellular domains the RTKs have been sub-divided into at least 14 different families.

Many receptors that have intrinsic tyrosine kinase activity as well as the tyrosine kinases that are associated with cell surface receptors contain tyrosines residues, that upon phosphorylation, interact with other proteins of the signaling cascade. These other proteins contain a domain of amino acid sequences that are homologous to a domain first identified in the c-Src proto-oncogene. These domains are termed SH2 domains. Another conserved protein-protein interaction domain identified in many
signal transduction proteins is related to a third domain in c-Src identified as the SH3 domain.

The interactions of SH2 domain containing proteins with RTKs or receptor associated tyrosine kinases lead to tyrosine phosphorylation of the SH2 containing proteins. The result of the phosphorylation of SH2 containing proteins that have enzymatic activity is an alteration (either positively or negatively) in that activity.

MAPKs were identified by virtue of their activation in response to growth factor stimulation of cells in culture, hence the name mitogen activated protein kinases. MAPKs are also called ERKs for extracellular-signal regulated kinases. Maximal MAPK activity requires that both tyrosine and threonine residues are phosphorylated. This indicates that MAP kinases act as switch kinases that transmit information of increased intracellular tyrosine phosphorylation to that of serine/threonine phosphorylation.

MAPKs are, however, not the direct substrates for RTKs nor receptor associated tyrosine kinases but are in fact activated by an additional class of kinases termed MAP kinase kinases (MAPK kinases) and MAPK kinase kinases (MAPKK kinases). One of the MAPK kinases has been identified as the proto-oncogenic serine/threonine kinase, RAF. Another MAPK kinases which are activated by RAF have been identified as MEK1 and ERK1. Next this cascade culminates in activation of the threonine and tyrosine protein kinase, ERK (MAPK). Activated ERK translocates to the nucleus, where it phosphorylates and activates transcription factors, such as AP-1, leading to the novo gene expression.

2.4.2 The Graphical Expression of RTK-MAPK

In this section, we give a coarse description of signal transduction, RTK-MAPK. That is to say, it is described from protein level, not its domains (e.g. domain SH2, SH3 etc.). Interactions are represented among proteins not domains of proteins.

In brief, the RTK-MAPK pathway is composed of 14 kinds of proteins. These bind and form complexes, modify certain residues on their counterparts (mostly by phosphorylation and dephosphorylation), change their confirmation and activity, and translocate between different cellular compartments (cytosol, nucleus and membrane). A change in gene expression patterns is the end result computed by this network of interactions. Fig. 2.9 [RSS00] is the graphical expression of signal transduction RTK-MAPK.

An informal description is as follows; A protein ligand molecule (GF) with two identical domains is a protein (outside signal) which will be sent to signal transduction, RTK-MAPK. It 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 intracellular part. The activated receptor can phosphorylate various targets, including its own tyrosine. The phosphorylated tyrosine is identified and bound by an adaptor molecule, SHC. A series of protein-protein binding events follows, lead-
which in turn recruits the serine threonine protein kinase, RAF, to the membrane, where it is subsequently phosphorylated and activated. A cascade of phosphorylations or activations follows, from RAF to MEK1 to ERK1. This cascade culminates in activation of the threonine and tyrosine protein kinase, ERK. Activated ERK translocates to the nucleus, where it phosphorylates and activates transcription factors, such as AP-1, leading to the novo gene expression.
Chapter 3

A Calculus For Formal Molecular Processes

In this chapter, we study how to model signal transduction by using process algebra. We consider mainly the case of signal transduction with aberrance. We provide a tag system and a simple typing system to mark aberrance. We prove that the tag system is equivalent to the simple typing system in the capability of labelling the existence of aberrance.

This chapter is organized as follows. First we recall some basic principles for modelling signal transduction using process algebra. In Section 3.2, we briefly recall the simple process of signal transduction with aberrance. In Section 3.3, we define the $I\pi$-calculus which is a variant of the $\pi$-calculus. The calculus is obtained by adding two aberrant actions into the $\pi$-calculus. In Section 3.4 and Section 3.5, we introduce a tag system and a simple typing system respectively based on the $I\pi$-calculus. Some properties of these two systems are discussed. In Section 3.6, we compare the tag system and the simple typing system and prove that they are equivalent in the capability of labelling existence of aberrance.

3.1 Basic Principles for the Model

Biomolecular processes are responsible for most of information processing in living cells. They are carried out by networks of interacting protein molecules. Systems biology aims to study systems consisting of these biomolecular processes. However the dynamic nature of these systems, their complexity, high connectivity and modularity further complicate this task. So we need novel approaches to study them. Formal approaches are thought to be feasible in studying of systems biology.

In the opinion of Aviv Regev etc. [RSS01], an appropriate formal approach for studying biomolecular processes should fulfill certain goals:

- It could provide a unifying view of dynamic behaviors it underlies. Preferably, this representation will be biologically visible, i.e. it should correspond well to
informal concepts and ideas of molecular biology.

- The formally represented data should be amenable to computer execution and analysis. Thus, the dynamic behavior of the system could be followed by simulation studies, including mutational analysis and simulated evolution. Alternatively, the system’s behavior may be formally verified.

- The formal approach should facilitate comparative studies of a system’s structure, dynamics and function within and between species.

- The formalism should be scalable and modularized to higher levels of organization.

Process algebras, which allow us to represent and analyze dynamic computational systems, seem appropriate formal approaches which satisfy the goals above.

Comparing to other methods, such as continuous mass-action differential equations, discrete Monte-Carlo simulations or Petri nets, the $\pi$-calculus, one of the process algebras, represents molecular systems as mobile communicating systems which are both highly detailed and biologically visible.

Aviv Regev etc. show that the $\pi$-calculus is suitable for modelling various molecular systems, including transcriptional circuits, metabolic pathways and signal transduction networks [RSS01]. For example, the $\pi$-calculus representations of signal transduction unify dynamic behavior and function of pathways with molecular details that underly their behavior. They provide a comprehensive model of signal transduction. The $\pi$-calculus programs are amenable to computer analysis and execution, analogous to mutational manipulation and experimentation, as well as to formal comparison and verification.

At the modelling level, within the particular framework of the $\pi$-calculus, we set five principles for this correspondence.

First, as our primitive process, we choose the functional signaling domain. We call the set of elements (amino acids for example) which have the similar function in a molecule a functional domain. A molecule can have several functional domains since it has different functions. To capture the functional and structural independence of domains in signaling molecules, a molecule is modelled as the composition of domains, a complex is modelled as the composition of molecules, and a more complicated complex is modelled as the composition of simple complexes. For example, in the well-studied signal transduction, $RTK$-$MAPK$, we view such signal transduction as a process.

$$RTK\_MAPK \ ::= \ Free\_ligand \ | \ \cdot\cdot\cdot | RTKs \ | ras \ | \cdot\cdot\cdot$$ 

(3.1)

where the molecule $Free\_ligand$, the complex of some receptor tyrosine kinases ($RTK$s), the protein molecule $ras$ etc. form the components of $RTK$-$MAPK$. 

24
The complex is composed of several molecules, each of which is modelled as a process as well. For example,

$$RTKs ::= RTK_1 | RTK_2 | \cdots$$ (3.2)

The complex of RTKs is composed of several receptors tyrosine kinases (denoted as $RTK_1, RTK_2, \cdots$).

A protein molecule is composed of several domains, each of which is modelled as a process as well.

$$Free\_ligand ::= Free\_binding\_domain | Free\_extracell\_domain$$ (3.3)

Since $Free\_ligand$ has two domains which have different functions, we can think that the molecule $Free\_ligand$ is composed of these two domains.

Second, we model the motifs and residues of domains as communication channels that construct a process. Motifs and residues are interacting portions of a domain but are not independently functional, just as channel names are communication ports of the process, but are not processes in their own right. For example, we take the residue ligand-binding of the domain $Free\_ligand\_domain$ of the protein $Free\_ligand$ as the channel name,

$$Free\_binding\_domain ::= \\text{ligand-binding} \cdots$$ (3.4)

$$Free\_extracell\_domain ::= \\text{ligand-binding} \cdots$$ (3.5)

In biological reactions, two molecules (or two domains) interact with each other based on their structural and chemical complementarity. Two complementary motifs are denoted by a global name and co-name pair, for example, ligand-binding and ligand-binding in (3.4),(3.5).

Third, molecular interaction and modification are modelled as communication and the subsequent change of channel names. Different types of molecular changes, such as chemical modification, conformation changes, or binding, affect further interaction in an analogous manner to the change in communication capabilities following channel names passing in mobile systems. These residues as channels and interaction as communication, yield a model of the molecular realm which is both highly visible and detailed. For example,

$$Free\_ligand\_domain | Free\_extracell\_domain \xrightarrow{\tau} \cdots$$ (3.6)

Two processes $Free\_ligand\_domain$ and $Free\_extracell\_domain$ make an interaction by the name ligand-binding and co-name ligand-binding (3.6).

Fourth, mutually exclusive interactions are summed together, by $+$. Biochemical interaction events may occur in sequence, in parallel with other independent
occurrences, or in a mutually exclusive, competitive fashion. For instance,

\[
\text{Free\_extracell\_domain} \ ::= \ \text{ligand\_binding\_trk\_binding} \cdots + \text{antagonist\_binding}
\]  

(3.7)

A sequence of interactions in which a molecule may participate is denoted by means of a prefix operator.

Finally, a pathway is not merely a bag of molecules and their domains. It is composed of defined compartments. First, 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. In all three cases molecules which share a common compartment may interact with each other, while molecules excluded from the compartment may not. We represent compartments by restricted communication scopes. For instance, a receptors tyrosine kinase (RTK) of the complex RTKs can be denoted as follows;

\[
\text{RTK} \ ::= \ \nu(\text{backbone})(\text{Extracell\_domain} | \text{Transmem\_domain} | \text{Intracell\_domain})
\]  

(3.8)

In the models presented in this thesis, we will comply with these principles.

### 3.2 Signal Transduction with Aberrance

Signal transduction is the key to uncover the wild growth of cells. When the whole signal transduction works perfectly, decisions of growth and death of cells are also made by rule and line. When some signals mutate aberrantly, the growth of a cell is not controlled anymore by growth factors outside.

Two kinds of mechanisms are changed in signal transduction with aberrance. One is that some aberrant proteins make the cell release growth factors into the environment. These factors can stimulate the cell which sets them free, and make it grow. In fact, one normal protein makes the release growth factors into the environment. These factors will stimulate some other protein, and make it grow. In this way, the growth of one protein depends on growth factors from other proteins, not growth factors from itself.

The other mechanism is aberrance of the ras protein. The normal ras protein in the inactive state is waiting for the signal. It is activated when it receives the signal, and then sends the signal to the other proteins. After that, it could be inactivated to return the initial state. This kind of inactivity ensures that the cell just can send finitely many signals at a time.

An aberrant ras protein has some difference with a normal ras protein. The aberrant ras protein can be activated and send a signal to the others, just as a normal ras protein. The aberrant ras protein however cannot be inactivated any more. That
means, it will be always in the active state and always send the signal to the others, even when there is no real signal coming.

A biochemical process is like a team which works well. They trust each other. A protein will make the decision to send signals if and only if it receives necessary signals. It just checks signals it receives but doesn’t care signals his father receives.

For instance, in Fig. 3.1, proteins $A_1, A_2, B_1, B_2, C$ are in normal state. When the whole process proceeds normally, proteins $B_1$ and $B_2$ receive signals ($S_{A_1}$ and $S_{A_2}$) from proteins $A_1$ and $A_2$ respectively and send signals ($S_{B_1}$ and $S_{B_2}$) to the protein $C$. When $C$ receives all the signals ($S_{B_1}$ and $S_{B_2}$), it sends the signal $S_C$ to the new one. When the protein $B_2$ is in the aberrant state (we denote the aberrant $B_2$ as $B_2'$), it can also send the signal $S_{B_1}$ to $C$ though there is no signal $S_{A_2}$ (in Fig. 3.1, the dashed arrow from $A_2$ to $B_2'$ represents no signal from $A_2$ to $B_2'$). The protein $C$ makes the decision to continue just by checking signals $S_{B_1}$ and $S_{B_2}$. It doesn’t care whether the signal $S_{A_2}$ actually occurred.

Formally, we take the whole biochemical process as a graph (e.g. Fig. 3.1). Suppose that $S$ and $P$ are boolean functions from proteins. $S(C) = T$ means the protein $C$ sends all the signals to the next ones successfully, and $S(C) = F$ means that some signals fail to be sent. $P(C) = T$ means that the protein $C$ receives all the signals from its predecessors successfully, and $P(C) = F$ means that some signals fail to be received.

**Proposition 3.1** If a protein $C$ is normal, then

$P(C) = T \iff S(C) = T$

**Proof:** Assume that $P(C) = T$, that is to say, the protein $C$ receives all necessary signals from his predecessors, then $C$ could send signals to the next ones since $C$ is in the normal state. On the other hand, if $P(C) = F$, because $C$ is in the normal state, it cannot send signals to the next ones, then $S(C) = F$.

For a protein $C$, if it satisfies $P(C) = F$ but $S(C) = T$, then $C$ is in the aberrant state. The corollary is from proposition 3.4.

### 3.3 The Interference $\pi$-calculus

The $\pi$-calculus has been applied to model biochemical networks. In these applications the modelling is done without considerations to exceptions. In order to describe more complex biochemical systems, the $I\pi$-calculus, the *Interference pi calculus*, is introduced for description of signal transduction with aberrance. The calculus is obtained by adding aberrant actions into the $\pi$-calculus.

#### 3.3.1 The Interference $\pi$-calculus: Definition

In process algebra, processes evolve by performing actions. Actions capabilities are introduced by prefix capabilities. In the $I\pi$-calculus, we introduce two capabilities in addition to prefixes defined by the $\pi$-calculus.
Let $a, b, \cdots$ range over names. We also define two symbols $\xi$ and $\eta$ to represent the aberrance capability. Here $\xi$ represents the suicide capability and $\eta$ the propagation capability. When a process has the suicide capability, it terminates its action immediately. And when a process has the propagation capability, it will duplicate its action infinitely.

**Definition 3.1 (Prefix)** The prefixes of $I\pi$-calculus are defined as follows:

$$\pi_0 := \sigma(b) | a(x) | \tau | a \quad \pi_1 := \pi_0 | \xi(\pi) | \eta(\pi)$$

where $\pi := \pi_0 | \pi_0 \cdot \pi$.

The syntactic category $\pi_0$ is like in the $\pi$-calculus [SW01]. The category $\pi$ stands for sequences of $\pi$-calculus capabilities. Finally the category $\pi_1$ allows for aberrant capabilities in addition to the normal ones. The two aberrance are relative to sequences of normal capabilities.

**Definition 3.2 (Process)** The $I\pi$-calculus processes are defined as follows:

$$P ::= 0 | \pi_1.P | \pi_1.P + \pi'_1.P' | P | P' | (\nu a)P$$

Hence the syntax of $I\pi$-calculus is the same as that of $\pi$-calculus except for its richer set of capabilities.

The standard $\pi$-calculus [Mil89] defines replication. In [SW01], it is shown that any process involving recursive definitions is representable using replication, and, conversely, that replication is redundant in the presence of recursion. In our language, to model the biological processes more naturally, we choose recursion instead of replication.
To give recursive definitions of processes, we introduce process constants, ranged by $K$, and add two new forms. First, we have recursive definitions of the form

$$K \triangleq (\check{a}).P$$

where $\text{fn}(P) \subseteq \check{a}$. Secondly, we have a new form of process, the constant application, $K[\check{a}]$

We refer to $K[\check{a}]$ as an instance of the process constant $K$.

The structural congruence $\equiv$ is the least equivalent relation on closed processes that satisfies the following equalities:

$$P \mid Q \equiv Q \mid P$$

$$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$$

$$P + Q \equiv Q + P$$

$$(P + Q) + R \equiv P + (Q + R)$$

$$(\nu a)0 \equiv 0$$

$$(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P$$

$$(\nu a)(P \mid Q) \equiv (\nu a)(P \mid Q) \text{ if } a \notin \text{fn}((()Q)$$

The reaction relation, introduced initially by Robin Milner [Mil89], is a concise account of computation in the $\pi$-calculus. In addition to the well-known interaction rule ($\text{Com-N}$), our reaction relation also includes two new rules about reactions with aberrance ($\text{Pre-\$}$ and $\text{Pre-\#}$).

$\tau$ is used to represent the silent action. $\alpha$ stands for actions, $\tau$, $\|$, and $\$.$

$$\xi(\pi).P \xrightarrow{\$} 0 \quad \text{Pre-\$};$$

$$\| (\pi).P \xrightarrow{\pi.\$} \pi.\$ .P \quad \text{Pre-\#};$$

$$\pi(b).Q + R_1 \mid (\alpha(x).P + R_2) \xrightarrow{\tau} Q \mid P \{b/x\} \quad \text{Com-N};$$

$$\pi.Q + R_1 \mid (\alpha.P + R_2) \xrightarrow{\tau} Q \mid P \quad \text{Com-SN}$$

$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \quad \text{Sum};$$

$$\frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \quad \text{Comp};$$

$$\frac{P \xrightarrow{\alpha} P'}{(\nu a)P \xrightarrow{\alpha} (\nu a)P'} \quad (\alpha \neq a) \quad \text{Res};$$

$$\frac{Q \equiv P}{Q \xrightarrow{\alpha} Q'} \quad \frac{P \xrightarrow{\alpha} P'}{P' \equiv Q'} \quad \text{Stc.}$$

$$K[\check{a}] \longrightarrow P[\check{a}/\check{x}] \quad (K \triangleq (\check{a}).P) \quad \text{R-Const}$$

Table 3.1: Reaction rules of $I\pi$-calculus.

The first two rules deal with reactions with aberrance: the former says that the resulting process is terminated; the latter declares that the resulting process may duplicate its action infinitely. The following rules are like in the $\pi$-calculus. The last rule is for constant applications. For example, assume that the constant $A_0$ is defined
as follows,

\[ A_0 \triangleq (x, y, z). (x(u).A_1[x, u] + y(w).A_0[x, y, w]) \]

The instance \( A_0[a, b, c] \) can be applied the rule \( R-\text{Const} \).

\[ A_0[a, b, c] \rightarrow (a(u).A_1[a, u] + b(w).A_0[a, b, w]) \]

### 3.3.2 A Model about \( ras \) Activation

In order to illustrate the use of our calculus, we consider an example in signal transduction pathway with aberrance. We focus our attention on the well-studied RTK-MAPK pathway. In biology, pathways of molecule interactions provide communication between the cell membrane and intracellular endpoints, leading to some change in the cell. Here we choose a small yet important part, \( ras \) activation, for explanation.

Fig. 3.2 gives an example of \( ras \) activation of the signal transduction pathway, RTK-MAPK (in Fig. 3.2, the protein \( ras \) is denoted as RAS). At the normal state, the protein-to-protein interactions bring the protein SOS close to the membrane, where the protein \( ras \) can be activated. The protein SOS activates the protein \( ras \) by exchanging \( ras \)'s GDP with GTP. More precisely, growth factor and cytokine activation of many tyrosine kinase and kinase-linked receptors recruits many proteins to the plasma membrane including \( ras \)-specific guanine nucleotide releasing proteins GNRP. Under the influence of a GNRP, \( ras \) proteins bind GTP, resulting in activation of the \( ras \) signal. Active \( ras \) interacts the next protein RAF in this signal transduction. After that, the protein GAP inactivates the active protein \( ras \) and makes it in the initial state, the inactive state.

Within the framework of the \( I\pi \)-calculus, we comply with the principles we mentioned in Section 3.1. Aviv Regev etc. have given the representation of normal RTK-MAPK using the \( \pi \)-calculus [RSS00, RSS01].

The interpretation of \( ras \) in the \( I\pi \)-calculus can be done in the following manner: The system defined in (3.9) is a collection of concurrently operating molecules, seen

![Figure 3.2: \( ras \) Activation](image-url)
as processes with potential behavior.

\[ Sys := ras | SOS | GAP | RAF \]  

(3.9)

where \( Sys \) is the abbreviation of our system. It includes four main proteins: \( ras, SOS, GAP, RAF \). This system, in fact, is the subsystem of (3.2).

A protein molecule is composed of several domains, each of which is modelled as a process as well. In (3.10) through (3.13) the detailed \( \pi \)-calculus programs for proteins \( ras, SOS, RAF \) and \( GAP \) are given:

\[ ras := INASWI\_I \cup INASWI\_II \]  

(3.10)

\[ SOS := S\_SH3\_BS \cup S\_GNEF \]  

(3.11)

\[ RAF := R\_Nt \cup R\_ACT\_BS \cup R\_M\_BS \]
\[ \quad \cup INA\_R\_Ct \cup R\_ATP\_BS \]  

(3.12)

\[ GAP := sg(c\_ras).\tau\_ras(gdp).GAP \]  

(3.13)

(3.10) says that \( ras \) is composed of two domains \( INASWI\_I \) and \( INASWI\_II \). We use "IN" to record the fact that two domains are in the inactive state. The protein \( SOS \) has two domains \( S\_SH3 \) and \( S\_GNEF \) where the domain \( S\_GNEF \) participates in interactions of our system \( Sys \), and the other domain \( S\_SH3 \) does not. Similarly, only the domain \( R\_Nt \) of the protein \( RAF \) participates in interactions of our system \( Sys \). The function of the protein \( GAP \) is to send the residue \( GDP \) to the protein \( ras \). So we can take it as one domain. (3.13) says chemical components \( sg, c\_ras, gdp \) compose of the protein \( GAP \) where \( gdp \) (i.e \( GDP \) in Fig. 3.2) is the chemical component which makes the protein \( ras \) stay in the inactive state.

The molecules (or domains) interact with each other based on their structural and chemical complementarity. Interaction is accomplished by motifs and residues that constitute a domain. These are viewed as channels or communication ports of the molecule:

\[ \begin{align*}
INASWI\_I & := \overline{bbone}.ACTSWI\_I \\
INASWI\_II & := sg(rs\_1).rs\_1(x).ACTSWI\_II \\
S\_GNEF & := bbone.S\_GNEF \cup sg(c\_ras).\tau\_ras(gtp).S\_GNEF
\end{align*} \]  

(3.14) \hspace{1cm} (3.15) \hspace{1cm} (3.16)

In (3.16), the motif \( bbone \) is the abbreviation of \( backbone \) which is the interacting portion of the domain \( S\_GNEF \). The residue \( gtp \) (i.e \( GTP \) in Fig. 3.2) is the chemical component which makes the protein \( ras \) stay in the active state. In (3.15), the motif \( sg \) of the domain \( INASWI\_II \) is the complementary of the motif \( sg \) of the protein \( GAP \).

In (3.14), the motif \( bbone \) is the complementary of the motif \( bbone \) of the domain \( S\_GNEF \).

Hence, the following interactions are possible:

\[ \begin{align*}
INASWI\_I \cup S\_GNEF & \rightarrow ACTSWI\_I \cup S\_GNEF \cup ... \\
INASWI\_II \cup S\_GNEF & \rightarrow^* ACTSWI\_II[\text{gtp/x}] \cup S\_GNEF \cup ...
\end{align*} \]  

(3.17) \hspace{1cm} (3.18)
The interaction (3.17) shows that the domain INASWI_I of ras is activated by the
domain of S_GNEF of SOS. The interaction (3.18) shows that the domain INASWI_II
of ras is activated by the domain S_GNEF of SOS by passing the residue gtp. Hence
the protein ras is activated.

The detailed $\pi$ programs for the domains, ACTSWI_I, ACTSWI_II of the pro-
tein ras and the domain R_Nt of RAF are defined in (3.19) through (3.21):

\[
\begin{align*}
ACTSWI_I & := \tau(ras_2).ras_2.bb\text{one}.INASWI_I \quad (3.19) \\
ACTSWI_II & := \overline{sg(r_{swi}_{-1}).r_{swi}_{-1}(x).bb\text{one}.INASWI_II} \quad (3.20) \\
R_Nt & := s(c_ras).c_ras.ACTR_Nt \quad (3.21)
\end{align*}
\]

(3.19) says that the domain ACTSWI_I can interact with another domain which
has the complementary motif s.

The processes so defined have the following interactions:

\[
\begin{align*}
ACTSWI_I & \mid R_Nt \rightarrow^* bb\text{one}.INASWI_I \mid ACTR_Nt \quad (3.22) \\
ACTSWI_II & \mid GAP \rightarrow^* \overline{bb\text{one}.INASWI_II}[gtp/x] \mid GAP \quad (3.23) \\
bb\text{one}.INASWI_I \mid \overline{bb\text{one}.INASWI_II} & \rightarrow INASWI_I \mid INASWI_II \quad (3.24)
\end{align*}
\]

During the process of ras activation, the interaction (3.22) occurs firstly, then in-
teractions (3.23) and (3.24) occur. (3.22) says that the domain ACTSWI_I of ras
interacts with the domain R_Nt of RAF, that is to say, the protein ras sends some
signal to the protein RAF. (3.23) says that GAP inactivates the domain ACTSWI_II
of ras by passing the residue GDP. (3.24) says that the domains of ras interact with
each other and that ras rollbacks to the initial inactivated state. One process of ras
activation is over. Then the protein ras will wait another signal from the protein
SOS, and begin the next activation.

When ras mutates aberrantly, it does not have any effect on the ras’s binding
with GTP but will reduce the activity of the GTP hydrolase of ras and lower its
hydrolysis of GTP greatly; in the meantime ras will be kept in an active state, that is,
the protein ras keeps activating the molecule, inducing the continual effect of signal
transduction, which results in cell proliferation and tumor malignancy.

During the process with aberrance, the protein GAP looses its capability of send-
ing the residue GDP to the protein ras. The structure and chemistry of the motif sg
of GAP are changed, and it cannot be complementary to the motif $\overline{sg}$ of the domain
ACTSWI_II of the protein ras. (3.25) defines the $\pi$ representation of GAP in the
aberrant state.

\[
\begin{align*}
GAP & := \overline{sg(c_ras)}.c_ras(gdp).GAP \quad (3.25) \\
GAP & \rightarrow 0 \quad (3.26) \\
ACTSWI_II \mid GAP & = ACTSWI_II \mid 0 \quad (3.27)
\end{align*}
\]

(3.26) shows that GAP loses its function and does nothing, meaning that it cannot
inactivate the domain ACTSWI_II of ras. Then the interaction (3.24) will not occur,
that is to say, the protein ras will always stay in active state.
During the process with aberrance, the domain \(ACTSWI\_I\) of the protein \(ras\) duplicates its capability of sending signals to the domain \(R\_Nt\) of the protein \(RAF\) by interactions. Now \(ACTSWI\_I\) can be written as follows;

\[
ACTSWI\_I ::= \#(\#(rs_2).\overline{rs_2}).bbone.INASWI\_I
\]

Hence, the following interactions are possible:

\[
ACTSWI\_I | R\_Nt \rightarrow^* \#(\#(rs_2).\overline{rs_2}).bbone.INASWI\_I
\]

The interaction (3.28) says that the domain \(\#(\#(rs_2).\overline{rs_2}).bbone.INASWI\_I\) always has the capability of sending signals to the domain \(R\_Nt\) which has the complemental motif \(s\). The interaction (3.24) never occurs.

In the \(\pi\)-calculus, it is a little difficult to represent the lost of capability and the propagation of some capabilities. Hence, the \(\pi\)-model could not easily describe this aberrant case. \(I\pi\)-calculus, on the contrary, can describe it quite precisely but simply.

\section{3.4 The \(I\pi\)-calculus with a Tag System}

Even in one aberrant biomolecular process, there are some proteins are aberrant while the others are normal. If all the proteins have some additional information about their states (normal or aberrant) in a formal model, the whole model will be clearer. Therefore, the idea of giving a tag to each protein for informing its state (normal or aberrant) is spontaneous.

In this section, we introduce such a tag system to make the model of the \(I\pi\)-calculus clearer.

\subsection{3.4.1 A Tag System}

We assume a set \(\mathbb{R}^+ = \{i : i \geq 0\}\) for tags. Let \(i_0, i_1, \ldots\) range over tags.

The syntax of the \(I\pi\)-calculus is modified as follows: we write a pair \((i, \pi)\) instead of the prefix \(\pi_i\), where \(i \in \mathbb{R}^+\) is the tag of \(\pi\). When \(\pi = \pi_0, i = 0\) is the tag of \(\pi_0\); when \(\pi = \#(\pi)\) or \(\pi = \#(\pi), i = 0\). As we have mentioned, \(\pi\) is the collection of some \(\pi_0s\). For example, \(\pi = \pi_1^0, \pi_0\), we define the tag of \(\pi\) as a set of tags of \(\pi_0\) and \(\pi_0\), that is, \(I_\pi = \{i_{\pi_0}, i_{\pi_0}^2\}\).

The expression of a process is also a pair \((I_P, P)\) where \(I_P\) is the tag of the process \(P\). The tag of one process is defined inductively by the following rules (Table 3.2).

We use the symbol \(\uplus\) to denote multiset union. For example, let \(A = \{a, b, c, d\}\) and \(B = \{b, c, e, f\}\), then \(A \uplus B = \{a, b, c, d, b, c, e, f\}\).

We write \(\bigcup_{n=1}^\infty I_P \triangleq I_P \uplus I_P \uplus \cdots\).

Let \(K_i = \vec{x}_i.P_i\) be constants, where \(i = 1, 2, \ldots, n\). We use \(\overline{K}\) to represent \((K_1, K_2, \ldots, K_n)\), and \(\overline{P}\) to represent \((P_1, P_2, \ldots, P_n)\). The expression "Let \(\overline{K} = \overline{P}\) in \(K_i\)" means \(K_i = \vec{x}_i.P_i\). We denote this expression as let.
For each recursive process \( P \), we define an environment \( \rho \) mapping the constants that appear free in \( P \) to sets of tags. Let \( \{K^n\} \) be a set if free constants in \( P \). Define the tag of the process \( P \):

\[
I_{P, \rho} = \rho(\{K^n\}) \cup I_{P \smallsetminus \{K^n\}}
\]

Next we compute the tag of a recursive process, that is, the tag of the expression \( \text{let} \). We define a function \( F \) on sets of tags.

\[
F(J_1, J_2, \ldots J_n) = I_{P, \rho'}
\]

where \( \rho' \) extends \( \rho \) with \( \rho'(K_i) = J_i, \rho'(K_2) = J_2, \ldots, \rho'(K_n) = J_n \).

**Proposition 3.2 (The Least Fixed Point)** The function \( F \) defined above has the least fixed point.

**Proof:** According to the least fixed point theorem, we just need to prove \( F \) is continuous, that is, \( F \) reverses increasing chains. Suppose that \( \mathcal{J} = (J_1, J_2, \ldots, J_n) \), \( \mathcal{J'} = (J'_1, J'_2, \ldots, J'_n) \). \( \mathcal{J} \subseteq \mathcal{J'} \) if and only if \( J_i \subseteq J'_i \), where \( i = 1, 2, \ldots, n \). We have

\[
F(\mathcal{J}) = I_{P, \rho'} = (\rho'(K_1 \cup I_{P_1 \smallsetminus \{K_1\}}), \rho'(K_2 \cup I_{P_2 \smallsetminus \{K_2\}}), \ldots, \rho'(K_n \cup I_{P_n \smallsetminus \{K_n\}})) = (J_1 \cup I_{P_1 \smallsetminus \{K_1\}}, J_2 \cup I_{P_2 \smallsetminus \{K_2\}}, \ldots, J_n \cup I_{P_n \smallsetminus \{K_n\}})
\]

So the least fixed point of the function \( F \) is the sup of \( \{\emptyset, F(\emptyset), F^2(\emptyset), \ldots, F^n(\emptyset), \ldots\} \).

We compute it as follows:

\[
F(\emptyset) = I_{P, \rho'} = (\bigcup I_{P_1 \smallsetminus \{K_1\}}, \ldots, \bigcup I_{P_n \smallsetminus \{K_n\}})
\]

\[
F^2(\emptyset) = I_{P, \rho'} = (\bigcup_{m=1}^2 I_{P_1 \smallsetminus \{K_1\}}, \ldots, \bigcup_{m=1}^2 I_{P_n \smallsetminus \{K_n\}})
\]

\[
\ldots
\]

\[
F^n(\emptyset) = I_{P, \rho^n} = (\bigcup_{m=1}^n I_{P_1 \smallsetminus \{K_1\}}, \ldots, \bigcup_{m=1}^n I_{P_n \smallsetminus \{K_n\}})
\]

\[
\ldots
\]

Therefore the least fixed point of \( F \) is

\[
\text{lfp}(F) = (\bigcup_{m=1}^n I_{P_1 \smallsetminus \{K_1\}}, \ldots, \bigcup_{m=1}^n I_{P_n \smallsetminus \{K_n\}}) = (I_1, I_2, \ldots, I_n)
\]

According to Proposition 3.2, for the expression \( \text{let} \), \( I_{\text{let}, \rho} = I_i = \bigcup_{m=1}^\infty I_{P_m \smallsetminus \{K_3\}} \).

Especially, when a process \( P \) has no free constants \( K, I_P = I_{P, \emptyset} \), where \( \emptyset \) is an empty environment.

**Example 1** Let \( K_0 = (x, y).(x(y).K_0[x, y]) \). The process \( P \) is a constant application, that is, \( P = K_0[a, b] \). According to the rule Const-t, the tag of the process \( P \) is:

\[
I_P = \bigcup_{n=1}^\infty I_{P_0 \smallsetminus \{K_0\}, \rho} = \bigcup_{n=1}^\infty I_{a(b), 0} = \bigcup_{n=1}^\infty \{a\}
\]

The last part of this derivation can be justified according to rules 0-t, N-t.

**Example 2** Let \( K_0 = (x, y).(x(y).K_0[x, y] + K_1[x, u, v]) \). The process \( P \) is a constant application, that is, \( P = K_0[a, b] \). According to the rule Const-t, the tag of the process \( P \) is:

\[
I_P = \bigcup_{n=1}^\infty I_{P_0 \smallsetminus \{K_0\}, \rho} = \bigcup_{n=1}^\infty I_{a(b), 0 + K_1[a, u, v]} = \bigcup_{n=1}^\infty (\{a\} \cup I_{K_1[a, u, v]})
\]

The last part of this derivation can be justified according to rule sum-t.
\[
\dfrac{P = 0}{I_P = \emptyset} \quad 0 \cdot t \quad \dfrac{P = \pi_0.Q}{I_P = \{i_0\} \uplus I_Q} \quad N \cdot t
\]

\[
\dfrac{P = \Sigma(n).Q}{I_P = \{0\}} \quad \Sigma \cdot t \quad \dfrac{P = \Pi(n).Q}{I_P = \biguplus_{n=1}^{\infty} (\{0\} \uplus I_\pi) \uplus I_Q} \quad \Pi \cdot t
\]

\[
\dfrac{P = Q + R}{I_P = I_Q \uplus I_R} \quad \text{Sum-t} \quad \dfrac{P = Q|R}{I_P = I_Q \uplus I_R} \quad \text{Com-t}
\]

\[
\dfrac{P = (\nu x)Q}{I_P = I_Q} \quad \text{Res-t} \quad \dfrac{P = K[\bar{a}]}{I_P = I_{P,\rho}} \quad (K \triangleq \bar{x}.P) \quad \text{Cons-t}
\]

Table 3.2: Tags of processes.

Let \( I_P, I_Q \) be the tags of the processes \( P \) and \( Q \). We define

\[
\langle I_P, P \rangle \equiv \langle I_Q, Q \rangle \iff P \equiv Q \& I_P = I_Q
\]

Since we have reaction rules of \( I\pi \)-calculus (see Table 3.1), we can assign tags to a reaction rule \( P \rightarrow Q \). These "dynamic tags" can be computed by Table 3.3.

We write \( P \rightarrow Q \) if \( P \rightarrow Q \) and \( I \) is the tag of this derivation according to Table 3.3.
\[ \{0\} \setminus \{0\} = \emptyset \quad \text{Pre-\$;} \]

\[ \bigcup_{n=1}^{\infty} \left( \{0\} \uplus \{x_n\} \right) \uplus I_P \setminus \{0\} = \{x_n\} \uplus I_P \uplus \bigcup_{n=1}^{\infty} \left( \{0\} \uplus \{x_n\} \right) \uplus I_P \quad \text{Pre-\$;} \]

\[ \left( \{x_1\} \uplus I_Q \uplus I_{R_1} \right) \uplus \left( \{x_2\} \uplus I_P \uplus I_{R_2} \right) \setminus \left( \{x_1\} \uplus I_{R_1} \uplus I_{R_2} \right) = I_Q \uplus I_P \quad \text{Com-N;} \]

\[ \left( \{x_1\} \uplus I_Q \uplus I_{R_1} \right) \uplus \left( \{x_2\} \uplus I_P \uplus I_{R_2} \right) \setminus \left( \{x_1\} \uplus I_{R_1} \uplus I_{R_2} \right) = I_Q \uplus I_P \quad \text{Com-SN;} \]

\[ \frac{I_P \setminus \{y\} = I_{P'}}{I_P \uplus I_Q} \quad \text{Sum;} \quad \frac{I_P \setminus \{y\} = I_{P'}}{I_P \uplus I_Q} \quad \text{Com;} \]

\[ I_Q = I_P \quad I_P \setminus \{x_1\} = I_{P'} \quad I_{P'} = I_Q \quad \text{Str;} \]

\[ I_P[\vec{a} / \vec{x}] = I_{P,\rho} \quad K \Downarrow \vec{x}.P \quad \text{R-const.} \]

Table 3.3: Tags for reduction rules.
Proposition 3.3 If $P \rightarrow_I Q$, then $I_P \setminus I = I_Q$.

PROOF: According to Table 3.3, we check tags of processes as follows:

(1) Assume that $\delta(\pi).P \xrightarrow{\delta} 0$ by the rule Pre-$\delta$, where $I = \{0\}$. The tag of the process $\delta(\pi).P$ is $\{0\}$ by rule $\delta$-t in Table 3.2. Then we have $\{0\} \setminus \{0\} = \emptyset$, where $\emptyset$ is the tag of the process $0$.

(2) Assume that $\pi(\pi).P \xrightarrow{\pi} \pi(\pi).P$ by the rule Pre-$\pi$, where $I = \{0\}$. The tag of the process $\pi(\pi).P$ is $\biguplus_{n=1}^{\infty} \{(0) \uplus I_{\pi} \} \uplus I_P$. Then we have $\biguplus_{n=1}^{\infty} \{(0) \uplus I_{\pi} \} \uplus I_P \setminus \{0\} = I_{\pi} \uplus \biguplus_{n=2}^{\infty} \{(0) \uplus I_{\pi} \} \uplus I_P$, where $I_{\pi} \uplus \biguplus_{n=2}^{\infty} \{(0) \uplus I_{\pi} \} \uplus I_P$ is the tag of the process $\pi(\pi).P$.

(3) Assume that $(\bar{a}(b).Q + R_1) \mid (a(x).P + R_2) \xrightarrow{\pi} Q|P\{b/x\}$ by rule Con-N, where $I = \{\bar{a}_0\} \uplus \{a_0\} \uplus I_{R_1} \uplus I_{R_2}$. The tag of the process $(\bar{a}(b).Q + R_1) \mid (a(x).P + R_2)$ is $(\{\bar{a}_0\} \uplus I_Q \uplus I_{R_1}) \uplus (\{a_0\} \uplus I_P \uplus I_{R_2})$. We have $(\{\bar{a}_0\} \uplus I_Q \uplus I_{R_1}) \uplus (\{a_0\} \uplus I_P \uplus I_{R_2}) \setminus (\{\bar{a}_0\} \uplus \{a_0\} \uplus I_{R_1} \uplus I_{R_2}) = I_Q \uplus I_P$, where $I_Q \uplus I_P$ is the tag of the process $Q|P\{b/x\}$.

(4) Assume that $(\bar{a}.Q + R_1) \mid (a.P + R_2) \xrightarrow{\pi} Q|P$ by rule Con-SN, where $I = \{\bar{a}\} \uplus \{a\} \uplus I_{R_1} \uplus I_{R_2}$. The tag of the process $(\bar{a}.Q + R_1) \mid (a.P + R_2)$ is $(\{\bar{a}\} \uplus I_Q \uplus I_{R_1}) \uplus (\{a\} \uplus I_P \uplus I_{R_2})$. We have $(\{\bar{a}\} \uplus I_Q \uplus I_{R_1}) \uplus (\{a\} \uplus I_P \uplus I_{R_2}) \setminus (\{\bar{a}\} \uplus \{a\} \uplus I_{R_1} \uplus I_{R_2}) = I_Q \uplus I_P$, where $I_Q \uplus I_P$ is the tag of the process $Q|P$.

(5) In the rule Sum, suppose that for the reaction $P \rightarrow_I P', I_P \setminus I = I_{P'}$, then for the reaction $P + Q \rightarrow_I P'$, where $I' = I \uplus I_Q$, we have $(I_P \uplus I_Q) \setminus (I \uplus I_Q) = I_{P'}$.

(6) In the rule Com, suppose that for the reaction $P \rightarrow_I P', I_P \setminus I = I_{P'}$, then for the reaction $P|Q \rightarrow_I P'$, where $I' = I$, we have $(I_P \uplus I_Q) \setminus I = I_{P'} \uplus I_Q$.

(7) In the rule Res, suppose that for the reaction $P \rightarrow_I P', I_P \setminus I = I_{P'}$, then for the reaction $(\nu a).P \rightarrow_{P'} (\nu a).P'$, where $I' = I$, we have $I_{P'} \setminus I' = I_{P'} \uplus I_Q$.

(8) In the rule Comp, suppose that $I_Q = I_P, I_{Q'} = I_{P'}$ and for the reaction $P \rightarrow_I P'$, $I_P \setminus I = I_{P'}$, then for the reaction $Q \rightarrow_{P'} Q'$, where $I' = I$, we have $I_Q \setminus I' = I_Q$.

(9) Assume that for the recursive process $K \triangleq \tilde{a}.P, K[\tilde{a}] \rightarrow_{P'} P\{\tilde{a}/\tilde{a}\}$ by the rule R-Const. It is trivial to prove.

In Section 3.3.2, we have given an example to show how to model an aberrant signal transduction using the $I\pi$-calculus. We can annotate this example using the $I\pi$-calculus with the tag system. In the new model, we give a tag for each motif and residue, then according to the rules in Table 3.2, we give the tag of each domain.
and process, which makes the model clearer since we have the state of each action and each process. If a prefix has tag 0, then it is in aberrant state; if 0 belongs to the tag of one process, then this process has aberrant actions. For any biomolecular interaction, the change of tags of processes (reactant and products) follows the rules of Table 3.3. Hence, we can check the existing aberrance, using the tag system of the $I\pi$-calculus. See Appendix A.

3.4.2 Tag Systems for Quantitative Analysis?

In Section 3.4.1, tags are just numbers which have not any meaning. We distinguish the aberrant processes and normal ones by checking the number 0 is in tags of processes or not.

In fact, the occurrence of aberrance is affected by temperature, environment, and concentration, etc. Naturally, if our tags come from biological results, that is, tags are gotten by some biological rules, then we want the tags to reflect this information. For example, suppose that the protein $A$ is in the normal state when the temperature is in some interval $[n, m]$ in the environment. When the real environmental temperature is far away from this interval, the protein $A$ will be in aberrant state. Based on this case, we can express it as tags formally:

For each motif or residue $a$, the tag $i_a$ of $a$ can be defined as a function from temperatures to $\mathbb{R}^+ = \{i : i \geq 0\}$:

\[
i_a(l) = \begin{cases} 
0 & n \leq l \leq m \\
(n - l) & l \leq n \\
(l - m) & m \leq l 
\end{cases}
\]

For a slightly more complicated example, we may want to account for the fact that the temperature interval $[n, m]$ at which the protein $A$ is in the normal state can change according to time, that is to say, $n$, $m$, are functions of time instead of constants. At time $t_0$, when the temperature is in the interval $[n(t_0), m(t_0)]$ in the environment, the protein $A$ is in the normal state. Hence, the tag $i_a$ of $a$ can be modified as follows:

\[
i_a(l, t_0) = \begin{cases} 
0 & n(t_0) \leq l \leq m(t_0) \\
(n(t_0) - l) & l \leq n(t_0) \\
(l - m(t_0)) & m(t_0) \leq l 
\end{cases}
\]

In this thesis, we just gave the simple modification of tag systems. We believe that richer tags systems will allow to record more information on the reactants of the biological processes, and hence will allow to account for quantitative aspects of systems biology in the framework of process calculi.
3.5 The $I\pi$-calculus with a Typing System

In this section, we introduce a simple typing system for the $I\pi$-calculus, with the similar aim to make the model of $I\pi$-calculus easier to understand.

3.5.1 A Typing System

As we have mentioned, for a biochemical network with aberrance, we hope to know what is brought on by aberrance. So in the $I\pi$-calculus, we need to control the information flow when modelling an aberrant biochemical network. This section describes rules for controlling information flow in the $I\pi$-calculus. There are several ways of formalizing those ideas, like the tag system introduced in Section 3.4. Here we embody them in a typing system for the $I\pi$-calculus.

In order to represent the aberrance of signal transduction we classify signals into three classes:

- A Normal signal is one that takes part in the normal processes.
- An Aberrant signal is one that takes part in the aberrant processes.
- An Unknown signal could be any signal.

To simplify we define a reflexive order relation $\prec$ among these three classes:

- Normal $\prec$ Unknown;
- Aberrant $\prec$ Unknown.

A name $\Gamma$ is denoted as environment, and $P$ as processes. The typed system has three kinds of assertions:

- "$\vdash \Gamma$ well formed" means that the environment $\Gamma$ is well-formed.
- "$\Gamma \vdash a : T$" means that the name $a$ is of the class $T$ in $\Gamma$.
- "$\Gamma \vdash P : Ok$" means that the process $P$ typechecks in the environment $\Gamma$.

Typing rules are given under an environment $\Gamma$. An environment is a list of distinct names with associated classifications.

**Definition 3.3 (Typed Environment)** Typing environments are given by the following rules:

\[
\begin{align*}
\vdash & \emptyset \text{ well formed} & \text{Environment Empty} \\
\vdash & \Gamma \text{ well formed, } M \notin \Gamma & \text{Environment Name} \\
\vdash & \Gamma, M : T \text{ well formed} & \text{Environment Name}
\end{align*}
\]

Having defined the environments, one can define rules for terms and processes.
Definition 3.4 (Terms) The rules for terms of typing system are as follows:

\[
\begin{align*}
\Gamma \vdash M : T & \quad T \lessdot R \\
\Gamma \vdash M : R
\end{align*}
\]

Level Subsumption

\[
\vdash \Gamma \text{ well formed} \quad M : T \in \Gamma
\]

Level Name

The rule Level Subsumption says that a term of level Normal or Aberrant has level Unknown as well.

Definition 3.5 (Processes) The rules for typing processes are as follows:

\[
\begin{align*}
\Gamma \vdash a : \text{Normal} & \quad \Gamma \vdash b : \text{Normal} & \quad \Gamma \vdash P : \text{Ok} & \quad \Gamma \vdash \alpha(b).P : \text{Ok} & \quad \text{T-out} \\
\Gamma \vdash a : \text{Normal} & \quad \Gamma \vdash x : \text{Unknown} & \quad \Gamma \vdash P : \text{Ok} & \quad \Gamma \vdash P : \text{Ok} & \quad \text{T-in} \\
\Gamma \vdash a : \text{Normal} & \quad \Gamma \vdash P : \text{Ok} & \quad \Gamma \vdash \alpha.P : \text{Ok} & \quad \Gamma \vdash a : \text{Normal} & \quad \Gamma \vdash P : \text{Ok} & \quad \Gamma \vdash \alpha.P : \text{Ok} & \quad \text{T-sout} & \quad \text{T-sin}
\end{align*}
\]

T-null

\[
\vdash \Gamma \text{ well formed} \quad M : T \in \Gamma
\]

T-res

\[
\Gamma, a : \text{Normal} \vdash P : \text{Ok}, \ a \not\in \text{dom}(\Gamma) \quad \Gamma \vdash (v \alpha)P : \text{Ok}
\]

T-ares
The original idea of this simple typing system is from [Aba99].
We give some properties about this simple typing system. The details of proofs are given in Appendix B.

**Proposition 3.4 (Strengthening)** Assume that the name $m$ is not free in the process $P$ and that $n \neq m$. The following properties hold:

- If $\Gamma, m : T \vdash N : S$, then also $\Gamma \vdash n : S$.
- If $\Gamma, m : T \vdash P : Ok$, then also $\Gamma \vdash P : Ok$.

Proposition 3.4 enables us to condense an environment, moving out the declaration of a name that is not used.

**Proposition 3.5 (Weakening)** Assume that $m$ is not defined in the environment $\Gamma$. The following properties hold:

- If $\Gamma \vdash n : S$, then $\Gamma, m : T \vdash n : S$.
- If $\Gamma \vdash P : Ok$, then $\Gamma, m : T \vdash P : Ok$.

Proposition 3.5 declares that anything that can be proved in a given environment can also be proved with more assumptions.

**Proposition 3.6** Assume that $\vdash \Gamma$ well formed and that names in $\text{dom}(\Gamma)$ are all normal. Then the following properties hold:

- If $m$ is a name and $m \in \text{dom}(\Gamma)$, then $\Gamma \vdash m : Normal$.
- if $P$ is a process with $f_a(P) \cup f_o(P) \subseteq \text{dom}(\Gamma)$, then $\Gamma \vdash P : ok$.

Proposition 3.6 says that a name is defined in an environment, then the name is of the class $T$ in the environment; if names of one process are defined in an environment, then the process $P$ typechecks in the environment.
3.5.2 A Simple Example

As we know, the typing system extracts information that is useful for reasoning about the behavior of programs. In this section, we still take ras activation (Section 3.3.2) as our basic example. We add a type for each motif and residue. Our example is type checked according to the typing rules.

Let $\Gamma$ be an environment, where some names are $\text{Unknown}$ levels:

$$x_{\text{INASWI}_{II}} : \text{Unknown}, \ x_{\text{ACTSWI}_{II}} : \text{Unknown}.$$

Bound names with their $\text{Normal}$ levels:

$$c_{\text{ras}_{\text{GAP}}} : \text{Normal}, \ c_{\text{ras}_{\text{GNEF}}} : \text{Normal}, \ c_{\text{ras}_{\text{RI}}} : \text{Normal}$$
$$gdp_{\text{GAP}} : \text{Normal}, \ \text{bbone}_{\text{INASWI}_{II}} : \text{Normal}, \ s_{\text{INASWI}_{II}} : \text{Normal},$$
$$rs_{1_{\text{INASWI}_{II}}} : \text{Normal}, \ \text{bbones}_{\text{GNEF}} : \text{Normal}, \ s_{\text{GNEF}} : \text{Normal},$$
$$gtp_{\text{GNEF}} : \text{Normal}, \ rs_{2_{\text{ACTSWI}}} : \text{Normal}, \ \text{bbone}_{\text{ACTSWI}_{II}} : \text{Normal},$$
$$sg_{\text{ACTSWI}_{II}} : \text{Normal}, \ \text{bbone}_{\text{ACTSWI}_{II}} : \text{Normal}, \ r_{\text{swi1}_{\text{ACTSWI}_{II}}} : \text{Normal},$$
$$s_{\text{RI}} : \text{Normal}.$$

Bound names with there $\text{Aberrant}$ levels:

$$s_{\text{ACTSWI}} : \text{Aberrant}, \ s_{\text{GAP}} : \text{Aberrant}.$$

The subscript of a name represents the domain to which the name belongs. According to Section 3.3.2, domains $\text{INASWI}_{I}$ and $\text{INASWI}_{II}$ of the protein $\text{ras}$ are actually recursive processes:

$$\text{INASWI}_{I} ::= \text{bbone} . (s_{2_{\text{ACTSWI}}}(x)).\text{bbone}.\text{INASWI}_{I} \quad (3.29)$$
$$\text{INASWI}_{II} ::= (s_{1_{\text{ACTSWI}}}).\text{rs}_{1_{\text{ACTSWI}}}(x).s_{1_{\text{ACTSWI}}}(r_{\text{swi1}_{\text{ACTSWI}}})$$
$$\quad (3.30)$$

Since

$$\Gamma \vdash \text{bbone}_{\text{INASWI}_{II}} : \text{Normal}, \ s_{\text{ACTSWI}} : \text{Aberrant},$$
$$rs_{2_{\text{ACTSWI}}} : \text{Normal}, \ \text{bbone}_{\text{ACTSWI}_{II}} : \text{Normal}$$

for the domain $\text{INASWI}_{II}$ (3.29), applying rules $T\text{-nil}$, $T\text{-sin}$, $T\text{-pout}$, $T\text{-sout}$, and $T\text{-const}$ in turn, we prove it as follows:

$$\Gamma \vdash \text{bbone}_{\text{ACTSWI}_{II}} : \text{Normal} \quad \Gamma \vdash 0 : \text{Ok} \quad \frac{}{\Gamma \vdash \text{bbone}.0 : \text{Ok}} \quad T\text{-nil}$$

$$\Gamma \vdash \text{bbone}_{\text{ACTSWI}_{II}} : \text{Normal} \quad \Gamma \vdash 0 : \text{Ok} \quad \frac{}{\Gamma \vdash \text{bbone}.0 : \text{Ok}} \quad T\text{-sin}$$
\[
\Gamma \vdash s_{ACTSWI,I} : \text{Aberrant} \quad \Gamma \vdash rs_{2,ACTSWI,I} : \text{Normal} \quad \Gamma \vdash \text{bbone.0} : \text{Ok}
\]
\[
\Gamma \vdash (rs_{2,ACTSWI,I}).\text{bbone.0} : \text{Ok}
\]

\[
\Gamma \vdash \text{bbone}_{INASWI,I} : \text{Normal} \quad \Gamma \vdash (rs_{2,INASWI,I}).\text{bbone.0} : \text{Ok}
\]
\[
\Gamma \vdash \text{bbone.}(rs_{2,INASWI,I}).\text{bbone.0} : \text{Ok}
\]
\[
\Gamma \vdash \text{INASWI,I} : \text{Ok}
\]

Since
\[
\Gamma \vdash \text{sg}_{INASWI,I} : \text{Normal}, rs_{1,INASWI,I} : \text{Normal}, x_{INASWI,I} : \text{Unknown}, \text{sg}_{ACTSWI,H} : \text{Normal}, r_{swi.1,ACTSWI,H} : \text{Normal}, x_{ACTSWI,H} : \text{Normal}, \text{bbone}_{ACTSWI,H} : \text{Normal}
\]

for the domain \text{INASWI}_{II} (3.30), applying rules \text{T-nil}, \text{T-sout}, \text{T-in}, \text{T-out}, \text{T-in}, \text{T-out} and \text{T-const} in turn, we prove it as follows:

\[
\vdash \Gamma \text{ well formed}
\]
\[
\Gamma \vdash 0 : \text{Ok}
\]

\[
\Gamma \vdash \text{bbone}_{ACTSWI,H} : \text{Normal} \quad \Gamma \vdash 0 : \text{Ok}
\]
\[
\Gamma \vdash \text{bbone.0} : \text{Ok}
\]

\[
\Gamma \vdash r_{swi.1,ACTSWI,H} : \text{Normal}
\]
\[
\Gamma \vdash x_{ACTSWI,H} : \text{Unknown}
\]
\[
\Gamma \vdash \text{bbone.0} : \text{Ok}
\]
\[
\Gamma \vdash r_{swi.1}(x).\text{bbone.0} : \text{Ok}
\]

\[
\Gamma \vdash \text{sg}_{ACTSWI,H} : \text{Normal}
\]
\[
\Gamma \vdash r_{swi.1,ACTSWI,H} : \text{Normal}
\]
\[
\Gamma \vdash r_{swi.1}(x).\text{bbone.0} : \text{Ok}
\]
\[
\Gamma \vdash \text{sg}(r_{swi.1}).r_{swi.1}(x).\text{bbone.0} : \text{Ok}
\]
\[ \begin{align*}
\Gamma \vdash rs_{1,\text{INASWI}_{II}} & : \text{Normal} \\
\Gamma \vdash x_{\text{INASWI}_{II}} & : \text{Unknown} \\
\Gamma \vdash sg(rs_{1,\text{INASWI}_{II}}).r_{\text{swi}_{1}}(x).bbone.0 : \text{Ok} & \quad \text{T-in} \\
\Gamma \vdash rs_{1}(x).sg(rs_{1,\text{INASWI}_{II}}).r_{\text{swi}_{1}}(x).bbone.0 : \text{Ok} \\
\Gamma \vdash sg(rs_{1}).rs_{1}(x).sg(rs_{1,\text{INASWI}_{II}}).r_{\text{swi}_{1}}(x).bbone.0 : \text{Ok} & \quad \text{T-out} \\
E \vdash \text{INASWI}_{II} : \text{Ok} & \quad \text{T-const}
\end{align*} \]

Hence, \( \Gamma \vdash ras = \text{INASWI}_{I} \mid \text{INASWI}_{II} : \text{Ok} \).

Only the functional domain \( S_{GNEF} \) of the protein \( SOS \) takes part in our system. It is represented as a recursive process in Section \([3.3.2]\). 

\( S_{GNEF} = bbone.S_{GNEF} \mid sg(c\_ras).c\_ras(gtp).S_{GNEF} \)

Since

\( \Gamma \vdash bbone_{S_{GNEF}} : \text{Normal}, sg_{S_{GNEF}} : \text{Normal}, c\_ras_{S_{GNEF}} : \text{Normal}, gtp_{S_{GNEF}} : \text{Normal} \)

applying rules \( T\text{-nil}, T\text{-sin}, T\text{-out} \ T\text{-in}, T\text{-com} \text{ and } T\text{-const} \), we prove it as follows:

\[ \begin{align*}
\vdash \Gamma \text{ well formed} & \quad \text{T-nil} \\
\Gamma \vdash 0 : \text{Ok} \\
\vdash \Gamma \text{ well formed} & \quad \text{T-sin} \\
\Gamma \vdash bbone_{S_{GNEF}} : \text{Normal} \quad \Gamma \vdash 0 : \text{Ok} & \quad \text{T-sin} \\
\Gamma \vdash bbone.0 : \text{Ok} \\
\vdash \Gamma \text{ well formed} & \quad \text{T-out} \\
\Gamma \vdash c\_ras_{S_{GNEF}} : \text{Normal} \quad \Gamma \vdash gtp_{S_{GNEF}} : \text{Normal} \quad \Gamma \vdash 0 : \text{Ok} & \quad \text{T-out} \\
\Gamma \vdash (c\_ras(gtp)).0 : \text{Ok}
\end{align*} \]
Only the functional domain $R.Nt$ of the protein $RAF$ takes part in our system. It is represented in Section 3.3.2 as follows:

\[ R.Nt = s(c.ras).c.ras.ACTR.Nt \]

Since

\[ E \vdash s_{R.Nt} : Normal, c.ras_{R.Nt} : Normal \]

where the domain $ACTR.Nt$ does not take part in our system, and we assume that the environment $E$ has $ACTR.Nt : Ok$.

applying rules $T-sin$, $T-in$, we prove it as follows:

\[ c.ras_{R.Nt} : it.Normal \quad \Gamma \vdash ACTR.Nt : Ok \]

\[ \Gamma \vdash c.ras.ACTR.Nt : Ok \quad T-sin \]

\[ s_{R.Nt} : Normal \quad c.ras_{R.Nt} : Normal \]

\[ \Gamma \vdash ACTR.Nt : Ok \]

\[ \Gamma \vdash s(c.ras).c.ras.ACTR.Nt : Ok \quad T-in \]

Hence, $\Gamma \vdash R.Nt : Ok$

The protein $GAP$ is represented as a recursive process in Section 3.3.2

\[ GAP = \delta(sg(c.ras).c.ras(gdp)).GAP \]

Since

\[ \Gamma \vdash sg_{GAP} : Aberrant, c.ras_{GAP} : Normal, gdp_{GAP} : Normal \]

applying rules $T-nil, T-kin$ and $T-const$, we prove it as follows:

\[ \vdash \Gamma \text{ well formed} \quad \Gamma \vdash 0 : Ok \quad T-nil \]
\[
\Gamma \vdash s_{GAP} : Aberrant \quad \Gamma \vdash c_{rasGAP} : Normal \quad \Gamma \vdash 0 : Ok
\]
\[
\Gamma \vdash \$ (s_{GAP} . c_{rasGAP} . gdp). 0 : Ok
\]
\[
\Gamma \vdash \$ (s_{GAP} . c_{rasGAP} . gdp). 0 : Ok
\]
\[
\Gamma \vdash GAP : Ok
\]

T-kin

T-const

In our system,
\[Sys = ras \mid SOS \mid RAF \mid GAP\]
the domain \( S \_SH3 \_BS \) of the protein \( SOS \), domains \( R \_ACT \_BS \), \( R \_M \_BS \), \( INA \_R \_Ct \), and \( R \_ATP \_BS \) of the protein \( RAF \) do not take part in our system. We assume that the environment \( \Gamma \) has \( S \_SH3 \_BS : Ok \), \( R \_ACT \_BS : Ok \), \( R \_M \_BS : Ok \), \( INA \_R \_Ct : Ok \), and \( R \_ATP \_BS : Ok \).

Therefore, it is easy to find that \( E \vdash Sys : Ok \) by rule \( T\text{-Com} \).

### 3.6 Comparing Tags and Types

In Section 3.4 and Section 3.5, we have given a tag system and a typing system, which help us to understand the \( I\pi \)-calculus. In this section, we compare these two systems and show that the tag system is equal to the typing system in terms of expressive power.

Every term has a tag in the tag system. A tag for a term and a set of tags for a process allow us record states of the whole biochemical process. We can use the language of set theory to record the process. It is natural and simple to understand. For instance,

\[A ::= a . b . 0\]

\(a\) has the tag \(t_a\), \(b\) has the tag \(t_b\), 0 has the tag \(\emptyset\). Then \(A\) has the tag \(\{t_a, t_b\}\).

We find, however, when a biochemical process is very complex, we have to take a large amount of space to record its tag. The tag of the process could be a large set containing all the tags of all the terms. Actually, in the qualitative analysis of biochemical processes, we don’t care what is the tag. We just want to know what happens with the aberrant ones, therefore, in the tag of the process. Of course, we believe that tags are important in the quantitative analysis.

The strong points of the typing system are obvious. As the other type systems, this typing system for \( I\pi \)-calculus is useful for several reasons:

1. to detect programming errors statically;

For example, the following errors in the process \( P \) can be found by our simple typing system:

(a) When \( \Gamma \vdash a : Normal \), \( P \) has the form of \( \$ (\pi) . Q \) or \( \# (\pi) . Q \).
(b) When $\Gamma \vdash a : Aberrant$, $P$ has the form of $\pi_0.Q$.

$$\pi_0 = \sigma(b) \mid a(x) \mid a \mid \sigma, \text{ and } \pi = \pi_0 \mid \pi.$$  

(2) to extract information that is useful for reasoning about the behavior of programs;

(3) to improve the efficiency of code generated by a compiler;

(4) to make programs easier to understand.

Specially, comparing to the tag system, this typing system is much simpler. For instance, taking $a : Aberrant$, again, we have that $a : normal, b : normal, 0 : ok \vdash A : ok$. The process $A$ is checked simply which it is enough to help our simulation. Of course, if we want to analyze the biochemical process quantitatively, this simple typing system is not enough.

Although these two systems have their shortcomings and strong points respectively, they are equal in terms of expressive power.

**Proposition 3.7 (Signal checking)** Let $i_a$ be the tag of the name $a$. Then

- $i_a = 0$ if and only if $a : Aberrant$;
- $i_a > 0$ if and only if $a : Normal$.

The proof of this proposition is trivial by the definition of tags.

Now, we bring out the key theorem of this chapter, presented as follows. We relate the tag system and the typing system by proving that they both capture the presence of an aberrant component in the system.

**Theorem 3.1 (Correspondence)** Let $I_P$ be the tag of $P$. If $\Gamma \vdash P : Ok$, the following statements are equivalent:

- (a) $0 \in I_P$
- (b) $P$ has a subprocess of the form $\xi(\pi).Q$ or $\parallel(\pi).Q$
- (c) There is a typing judgement $\Gamma \vdash P : Ok$ such that in the proof of this typing judgement there is a proof of a judgement of the form $\Gamma' \vdash m : Aberrant$

**Proof:** It is easy to prove that if $P$ has a subprocess of the form $\xi(\pi).Q$ or $\parallel(\pi).Q$, then $0 \in I_P$.

Suppose that $0 \in I_P$. According to Table 3.2, (b) is proved.

Next we prove that (b) is equal to (c). Suppose that $P$ has a subprocess of the form $\xi(\pi).Q$ or $\parallel(\pi).Q$. It suffices to consider the following cases:
(1) Case $\pi. Q$. Since $\Gamma \vdash \pi. Q : Ok$, this assertion must be established through rules $T\cdot k\text{out}$, $T\cdot k\text{in}$, $T\cdot k\text{sout}$, and $T\cdot k\text{sin}$. Then there is a name $m$ in $\pi$ such that $\Gamma \vdash m : Aberrant$.

(2) Case $\pi. Q$. Since $\Gamma \vdash \pi. Q : Ok$, this assertion must be established through rules $T\cdot p\text{out}$, $T\cdot p\text{in}$, $T\cdot p\text{sout}$, or $T\cdot p\text{sin}$. Then there is a name $m$ in $\pi$ such that $\Gamma \vdash m : Aberrant$.

(3) Case $R' + R$. Since $\Gamma \vdash R' + R : Ok$, this assertion must be established through the rule $T\cdot \text{Sum}$. Then we have $\Gamma \vdash R' : Ok$ and $\Gamma \vdash R : Ok$. Because $R' + R$ has a subprocess of the form $\pi. Q$ or $\pi. Q$, we have $R'$ or $R$ has a subprocess of the form $\pi. Q$ or $\pi. Q$. Without loss of generality, we consider $R'$ has a subprocess of the form $\pi. Q$. Then there is a name $m$ in $R'$ such that $\Gamma' \vdash m : Aberrant$ by assumption, of course, $m \in R' + R$ and $\Gamma' \vdash m : Aberrant$. Case $R' \mid R$ is similar to prove.

(4) Case a constant application ($P = K[\tilde{a}]$) for a recursive definition ($K \triangleq \tilde{a}. P'$). Since $\Gamma \vdash P : Ok$, this assertion must be established through the rule $T\cdot \text{Const}$. Then we have $\Gamma \vdash P[\tilde{a}/\tilde{x}, 0/K] : Ok$. Because $P$ has the form of $\pi. Q$ or $\pi. Q$, that is, $P[\tilde{a}/\tilde{x}, 0/K]$ has the form of $\pi. Q$ or $\pi. Q$. By assumption, there is a name $m$ in $P[\tilde{a}/\tilde{x}, 0/K]$, of course in $P$, such that $\Gamma' \vdash m : Aberrant$.

Next we prove the converse. It also suffices to consider the following cases:

(1) The base case is that $P$ is the form of $\pi. Q$ or $\pi. Q$ by the rules $T\cdot k\text{out}$, $T\cdot k\text{in}$, $T\cdot k\text{sout}$, $T\cdot k\text{sin}$, $T\cdot p\text{out}$, $T\cdot p\text{in}$, $T\cdot p\text{sout}$, and $T\cdot p\text{sin}$. Then (b) is proved.

(2) Assume that $P$ is the form of $R' + R$ and processes $R'$ and $R$ satisfy the property (c). Since $\Gamma \vdash P : Ok$, that is $\Gamma \vdash R' + R : Ok$, we have $\Gamma \vdash R' : Ok$, and $\Gamma \vdash R : Ok$. Because there is a term $m \in P$ such that $\Gamma' \vdash m : Aberrant$, we can know that $m \in R'$ or $m \in R$ such that $\Gamma \vdash m : Aberrant$. Without loss of generality, we consider $m \in R'$. From the assumption, we know $R$ has the subprocess of the form of $\pi. Q$ or $\pi. Q$, then of course, $P$ has the subprocess of the form of $\pi. Q$ or $\pi. Q$. The case that $P$ is the form of $Q \mid R$ is similar to discuss.

(3) Assume that $P$ is a constant application ($P = K[\tilde{a}]$) for a recursive definition ($K \triangleq \tilde{a}. P'$) and $P[\tilde{a}/\tilde{x}, 0/K]$ satisfies the property (c). Since $\Gamma \vdash P : Ok$, it must be that $\Gamma \vdash P[\tilde{a}/\tilde{x}, 0/K] : Ok$ by the rule $T\cdot \text{Const}$. If there is a name $m \in P$ such that $\Gamma' \vdash m : Aberrant$, we can know that $m \in P[\tilde{a}/\tilde{x}, 0/K]$. Then $P[\tilde{a}/\tilde{x}, 0/K]$ has the subprocess of the form of $\pi. Q$ or $\pi. Q$ by assumption. That is $P$ has the subprocess of the form of $\pi. Q$ or $\pi. Q$.

With our brief typing system, we can verify the aberrant ST pathways without complex tags.
Chapter 4

A Language for Formal Proteins

In this chapter, we study a language for formal proteins, the $\kappa$-calculus. We consider the problem of self-assembly which we formalize in terms of a translation from the $\kappa$-calculus (playing the role of a coarse-grained language) to a finer-grained (sub)language, the $m\kappa$-calculus. The mathematical properties of self-assembly are discussed. The correctness of this translation is proved using these properties.

The contents of this chapter are organized as follows. In Section 4.1, we give a $\kappa$-model of ras activation. In Section 4.2, we recall some basic notions about the $m\kappa$-calculus. In Section 4.3, we introduce the property self-assembly. The graphic explanation of the reversible and non reversible rules of translation from the $\kappa$-calculus to the $m\kappa$-calculus is given. In Section 4.4, we show some mathematical properties of self-assembly, including confluence, strong normalization. The correctness of this translation is proved using these mathematical properties.

4.1 The $\kappa$-model of ras Activation

In Section 2.2.1, we have recalled some basic notions of the $\kappa$-calculus. In this section, we give an example to show how to model the process of ras activation using the $\kappa$-calculus.

The process of ras activation was introduced in Section 3.3.2. Here we denote proteins as primitive proteins; complexes of proteins as solutions; and residues of functional domains as sites of proteins. Reactions for activation are denoted by monotonic reactions, and reactions for inactivation are denoted by antimonotonic reactions.

First, the protein ras is activated by the protein SOS (reaction (4.1)).

\[
\text{ras} \,(b\text{bone} + s\text{g} + rs_{-1} + \text{b\text{bone}2}), \text{SOS} \,(b\text{bone} + s\text{g} + c_{-ras}) \rightarrow (x_{y_{1}}y_{2})(\text{ras} \,(b\text{bone}^{x} + sg^{y_{1}} + rs_{-1}^{y_{2}} + b\text{bone}2), \text{SOS} \,(b\text{bone}^{x} + sg^{y_{1}} + c_{-ras}^{y_{2}})) \quad (4.1)
\]
Different from the example in Section 3.3.2, we do not denote functional domains in proteins. Actually, in reaction (4.1), $bbone$ is the residue of one functional domain ($INASWI_I$) and $sg, rs_1, bbone_2$ belong to the other domain ($INASWI_{II}$) in the protein $ras$. Reaction (4.1) says that the domain $INASWI_I$ of the protein $ras$ is activated by the binding of $bbone$, and the other domain $INASWI_{II}$ is activated by bindings of $sg, rs_1$. Then the protein $ras$ is activated.

Reaction (4.2) shows how the activated protein $ras$ activates the next protein $RAF$. The protein $RAF$ is activated by binding sites $s, rs_2$ of the protein $ras$ with sites $s, c_{ras}$ of the protein $SOS$.

$$\begin{align*}
ras(bbone^a + sg^{y_1} + rs_1^{y_2} + s + rs_2),& \quad RAF(s + c_{ras}) \rightarrow \\
(z_1 z_2)(ras(bbone^a + sg^{y_1} + rs_1^{y_2} + s^{z_1} + rs_2^{z_2}),& \quad RAF(s^{z_1} + c_{ras}^{z_2}))
\end{align*}$$ (4.2)

Reaction (4.3) says that when the protein $GAP$ takes part in the reactions, the activated protein $ras$ begins to lose its activity. This reaction is taken by unbinding sites $sg, rs_1$ of the protein $ras$ from sites of $sg, c_{ras}$ of the protein $SOS$.

$$\begin{align*}
(y_1 y_2)(ras(bbone^a + sg^{y_1} + rs_1^{y_2} + bbone_2),& \quad \text{SOS}(bbone^a + sg^{y_1} + c_{ras}^{y_2}), GAP(s + c_{ras}) \rightarrow \\
\text{SOS}(bbone^a + sg + c_{ras}),& \quad GAP(s_{\overline{y}} + c_{\overline{ras}}))
\end{align*}$$ (4.3)

Reaction (4.4) says that the protein $ras$ goes back to the initial state, that is, the inactive state. The process of $ras$ activation is over.

$$\begin{align*}
(x)(ras(bbone^a + sg + rs_1 + \overline{bbone}_2),& \quad \text{SOS}(bbone^a)) \rightarrow \\
ras(bbone + sg + rs_1 + \overline{bbone}_2),& \quad \text{SOS}(bbone)
\end{align*}$$ (4.4)

This example is simple yet biologically visible. We simplify complicated biological reactions and only use the binding or unbinding to represent reactions, which capture the information about which proteins take part in the reactions though we omit details of reactions.

### 4.2 The $m\kappa$-calculus

The $m\kappa$-calculus is a finer-gained language which describes a less idealized formal biology. The syntax of the $m\kappa$-calculus is the same as the $\kappa$-calculus discussed
in Section [2.2.1] except for group names, and the semantics of the reaction rules in the \( m\kappa \)-calculus are more restricted.

We call basic components in the \( m\kappa \)-calculus agents. We assume an infinite countable set \( \mathcal{P} \) of agents names, an infinite countable set \( \mathcal{E} \) of edge names and an infinite countable set \( \mathcal{G} \) of group names. Edge and group names are supposed to be disjoint. We take the same signature map \( s \) from \( \mathcal{P} \) to natural numbers \( \mathbb{N} \) as in \( \kappa \)-calculus.

Let \( A, B, \cdots \) range over protein names, \( x, y, \cdots \) range over edge names, and \( r, r', \cdots \) range over group names. For each agent name \( A \), \( s(A) \) is the number of sites of \( A \), and for any \( 1 \leq i \leq s(A) \), each pair \( (A, i) \) will accordingly be called a site of \( A \).

An extended interface, is a finite map from \( \mathbb{N} \) to \( (\mathcal{E} + \mathcal{G} + \{h, \nu\}) \times \mathbb{N} \), ranged over by \( \theta \) and similar symbols. The integer part of \( \theta(i) \) is referred to as a log.

An agent is a pair written \( A(\theta) \) with \( A \in \mathcal{P} \) and \( \theta \) an extended interface defined on \( s(A) \). Solutions are built as in Section 2.2. All concepts about the edges is extended to the edges and group names. For instance, interval extrusion \( S, (x)(S') \equiv (x)(S, S') \) applies both for \( x \) in \( \mathcal{E} \) and \( \mathcal{G} \), with the usual side-condition that \( x \not\in \text{fn}(S) \).

For example, for an agent \( A \), if \( s(A) = 3 \), and \( \theta(1) = x, 1, \theta(2) = r, 2 \) and \( \theta(3) = h, 0 \), then one may simply write \( A(1^{x,1} + 2^{r,2} + 3^0) \). We will also indulge sometimes in not writing a log when it is 0, so that for instance \( C(1^{x,4} + 2^{r,3}) \) will stand for \( C(1^{x,4} + 2^{r,0} + 3^0) \). A convenient consequence of this notational abuse is that \( \kappa \)-proteins become a particular case of \( m\kappa \)-agents.

In the \( m\kappa \)-calculus, group names are used as the means to built transient cooperative structures in our low-level systems. Logs are the additional information on sites. They are used to effect that the agent can log what’s up on this connection. The logs can be forgotten by means of the following projection map:

\[
(z^{x,n})^- = z^x \quad \quad (z^{r,n})^- = (z^{\nu,n})^- = z^\nu \quad \quad (z^{h,n})^- = z^h
\]

This projection extends in the obvious way to interfaces, agents and solutions.

**Definition 4.1 (Interaction)** Let \( L, R \), be two \( m\kappa \)-solutions, \( L \rightarrow R \) is said to be an interaction if:

- both \( L \) and \( R \) consist of at most two agents
- \( \text{fn}(L) \sqsubseteq \text{fn}(R) \)
- \( L \) does not contain any "\( \nu \)" on group names

**Definition 4.2 (Monotonic Interaction)** An \( m\kappa \)-interaction \( L \rightarrow R \) is said to be monotonic (resp. anti-monotonic) if its projection \( (L)^- \rightarrow (R)^- \) is a monotonic (resp. anti-monotonic) \( \kappa \)-reaction.

Thus, in the \( m\kappa \)-calculus we consider only interactions which are simple and close to biological interactions. These interactions are taken as atomic events. And they can encode a given higher level reaction.
4.3 The Self-assembly Protocol for the $\kappa$-calculus

Self-assembly is a very important property in biology. It is a method of integration in which components spontaneously assemble, typically by bouncing around in a solution or gas phase until a stable structure of minimum energy is reached. Self-assembly is crucial to the assembly of bio-molecular nano-technologies, and is thus a promising method for assembling atomically precise devices. Components in self-assembled structures find their appropriate location based solely on their structural properties (or chemical properties in the case of atomic or molecular self-assembly). Self-assembly is by no means limited to molecules or the nano-scale and can be carried out on just about any scale, making it a powerful bottom-up method.

In our specific case, we can synthesize processes given a higher level description in the $\kappa$-calculus. For each of interacting proteins, in a purely decentralized way and with binary synchronization as the only means of communication, proteins are going to behave according to the original higher level description.

4.3.1 The Monotonic Protocol

The purpose of this subsection is to decompose a $\kappa$-reaction in the $m\kappa$-calculus: given a $\kappa$-reaction $\tau$, one wants to define an associated family, $[\tau]_m$, of $m\kappa$-interactions capable of simulating $\tau$ in a sense that will be made precise below.

We follow a protocol that gradually recruits reactants and constructs the products by the only means of binary and unary interactions. Hence, we need some statically predetermined structure in which we can know which agents may or may not be managed in one step in the protocol.

**Definition 1 (Micro-scenario)** Let $\tau = L \rightarrow (\vec{\varepsilon})R$ be a monotonic $\kappa$-reaction, a micro-scenario for $\tau$ is a triple $(F_\tau, \tau_\alpha, \text{init})$ such that:

— $F_\tau$ is (isomorphic to) an acyclic orientation of $[R]_g$, called a flow graph;
— $\tau_\alpha$ is a tree spanning $F_\tau$;
— $\text{init}$, also written $\text{init}(F_\tau)$, is the common root of $F_\tau$ and $\tau_\alpha$;
— and $\text{init}$ belongs to $[L]_g$ (up to the isomorphism above).

We have some remarks on micro-scenario.

1. We assume that $F_\tau$ is an oriented graph-with-sites with integers as nodes. For different $\kappa$-reactions, their micro-scenarios are chosen to use disjoint set of nodes. In this way, when agents are recruited and handed a node of $F_\tau$, they can identify which global $\kappa$-reaction they take part in, and what role they have in it.

2. Such micro-scenarios always exist [DL04]. There always is a micro-scenario for any given monotonic $\kappa$-reaction: since $[R]_g$ is connected by monotonicity, any node of $[L]_g$ can be chosen as the root and one could even assume $\tau_\alpha$ to be...
depth-first. Because any connected graph admits an acyclic orientation which can be obtained, for instance, by choosing an arbitrary root, constructing a depth-first tree spanning the graph, and directing all remaining edges according to the tree ordering.

(3) We suppose that $[R]_\sigma$ doesn't have loops. In fact, there could be loops in $[R]_\sigma$, that is edges from a node to itself. The techniques described here adapt very easily to this case, since loops are purely local to a node.

(4) The micro-scenario is not unique. We could do with more than one initiator, without a spanning tree, etc [?]. Even in the restricted kind of micro-scenarios that we are considering, there is room for different choices.

(5) The spanning tree $T$ serves as a way of imposing a "parental priority" between the contacts in our protocol. We will give explanation later.

The graph $F_t$ can be decided as a map over sites, defined as follows:

$$F_t(a, i) = \begin{cases} (b, j) & \text{if } (a, i), (b, j) \text{ is connected in } F_t \\ \perp & \text{if } (a, i) \text{ is free in } F_t \end{cases}$$

We write $F_t^*$ for the inverse of $F_t$.

Intuitively, a site is an output if it belongs to the domain of $F_t$, and an input if it belongs to the range of $F_t$. In other words, a site $(a, i)$ is called an output if $F_t(a, i) \neq \perp$ and an input if $F_t^*(a, i) \neq \perp$.

**Definition 2 (input/output interfaces)** For $n \in \mathbb{N}$, $a \in F_t$ and $\bar{z}$ a tuple indexed over the set of inputs (resp. outputs) of $a$ and with values in $E$, we define the input (resp. output) interfaces:

$$\wedge^a_n : = \sum_{\{i | F_t(a, i) \neq \perp\}} \bar{z}^{i, n}$$

$$\vee^a_n : = \sum_{\{i | F_t^*(a, i) \neq \perp\}} \bar{z}^{i, n}$$

These interfaces will serve us to denote the fact that all input (or output) parts bear a certain log $n$.

A protein with name $A$ is translated as an agent of the same name but with one more auxiliary site, written $*$:

$$[A(\rho)]_m = A(* + \rho).$$

This translation extends to $\kappa$-solutions and likewise, if $S$ is a $\kappa$-solution, we write $[S]_m$ to denote its translation. That special site $*$ is used to log what little additional information one needs, that is the role of $A$ in a given reaction (i.e. to which node it corresponds in $F_t$) and a group name identifying uniquely the current attempted
high-level reaction. To ease reading, we have systematically abbreviated $A(r^a + \theta)$ as $A^r^a(\theta)$ and also made use of the notation for input and output interfaces introduced above.

Thereafter and for the rest of the paper, we suppose that it is given a monotonic reaction $L \rightarrow_\kappa (ux)R$, a choice of micro-scenarios has been made.

The monotonic protocol consists in a first phase of recruitment and a subsequent phase of completion. Recruitment begins with a signal sent by a specific agent called the initiator. Then one sends and propagates two kinds of signals: forward signals to recruit the necessary reactants, and backward signals to report success back to the initiator. At the end of this first phase, the initiator knows that the global $\kappa$-reaction can be completed, and in the completion phase, this information is propagated to the other reactants.

\[
\mathcal{F}_r^a(a, z) = \bot
\]

\[
A^a(\Delta^v + \nabla^2) \Rightarrow (r)(A^r^a(\Delta^v + \nabla^2))^{\text{init}}
\]

\[
\mathcal{T}_r^a(a, z) = (b, j), y \in \text{fn}(r)
\]

\[
A^r^a(\lambda^a_1 + \nabla^0), B(j^v + \nabla^2) \Rightarrow A^r^a(\lambda^a_1 + \nabla^0), B^r^b(j^v + \nabla^2, 0)
\]

\[
\mathcal{F}_c^1
\]

\[
A^r^a(\lambda^a_1 + \nabla^0), B(j^v + \nabla^2) \Rightarrow (y)(A^r^a(\lambda^a_1 + \nabla^0), B^r^b(j^v + \nabla^2, 0))
\]

\[
\mathcal{T}_r^a(a, z) = (b, j), y \not\in \text{fn}(r), b \in L
\]

\[
A^r^a(\lambda^a_1 + \nabla^0), B(j^v + \nabla^2) \Rightarrow (y)(A^r^a(\lambda^a_1 + \nabla^0), B^r^b(j^v + \nabla^2, 0))
\]

\[
\mathcal{F}_c^2
\]

\[
A^r^a(\lambda^a_1 + \nabla^0), B(j^v + \nabla^2) \Rightarrow (y)(A^r^a(\lambda^a_1 + \nabla^0), B^r^b(j^v + \nabla^2, 0))
\]

\[
\mathcal{T}_c^a(a, z) = (b, j), x \in \text{fn}(r)
\]

\[
A^r^a(\lambda^a_1 + \nabla^0), B^r^b(j^v + \nabla^2, 0) \Rightarrow A^r^a(\lambda^a_1 + \nabla^0), B^r^b(j^v + \nabla^2, 0)
\]

\[
\mathcal{F}_c^3
\]

\[
A^r^a(\lambda^a_1 + \nabla^0), B^r^b(j^v + \nabla^2, 0) \Rightarrow (x)(A^r^a(\lambda^a_1 + \nabla^0), B^r^b(j^v + \nabla^2, 0))
\]

\[
\mathcal{T}_c^a(a, z) = (b, j), x \not\in \text{fn}(r)
\]

\[
A^r^a(\lambda^a_1 + \nabla^0), B^r^b(j^v + \nabla^2, 0) \Rightarrow (x)(A^r^a(\lambda^a_1 + \nabla^0), B^r^b(j^v + \nabla^2, 0))
\]

\[
\mathcal{F}_c^4
\]

Table 4.1: $\|_1^0$
\[
\begin{align*}
F_i(a, i) &= (b, j) \\
A^r.a(\tilde{\tilde{x}}^1_a + \tilde{y}^1_b), B^r.b(j\tilde{y}^2_b + \tilde{z}^2_b) &\Rightarrow A^r.a(\tilde{\tilde{x}}^1_a + \tilde{y}^1_b), B^r.b(j\tilde{y}^2_b + \tilde{z}^2_b)
\end{align*}
\]

Table 4.2: $\|_2$

\[
\begin{align*}
\uparrow_{12}; \downarrow_{20} F^*_i(a, i) &= \bot \\
A^r.a(\tilde{\tilde{x}}^3_a + \tilde{y}^3_b) \rightarrow A^r.a(\tilde{\tilde{x}}^2_a + \tilde{y}^2_b)
\end{align*}
\]

phase-shift

\[
A^r.a(\tilde{\tilde{x}}^3_a + \tilde{y}^3_b), B^r.b(j\tilde{y}^2_b + \tilde{z}^2_b) \rightarrow A^r.a(\tilde{\tilde{x}}^3_a + \tilde{y}^3_b), B^r.b(j\tilde{y}^2_b + \tilde{z}^2_b)
\]

propagation

Table 4.3: $\|_3$

\[
A^r.a(\tilde{\tilde{x}}^3_a + \tilde{y}^3_b) \rightarrow A(\tilde{\tilde{x}} + \tilde{y})
\]

exit

Table 4.4: Exit

We give some explanation for the protocol as follows:

(1) Table 4.1 includes the rule *initiation* (init for short), *first contacts* ($FC_1, FC_2, FC_3$), and *late contacts* ($LC_1, LC_2$). Table 4.2 includes the rule *response* (R for short). Table 4.3 includes the rule *phase-shift* (ps for short) and the rule *propagation*. Table 4.4 includes the rule *exit*.

(2) The graphical explanation of the protocol will be given in the next section.

(3) The spanning tree $T_e$ assures that only the parent according to $T_e$ is allowed to wake a child and recruit it, while all the other reactants have to use further interactions. Some self-deadlocks may happen if we do not do this. For instance, given a monotonic $\kappa$-reaction and a solution as follows:

\[
\begin{align*}
t &= A(1^x + 2), B(1^x + 2), C(1 + 2) \rightarrow \\
&\quad (yz)(A(1^x + 2^y), B(1^x + 2^y), C(1^y + 2^y)) \\
T &= (xyz)(A(1^x + 2), B(1^x + 2), C(1 + 2), C(1 + 2))
\end{align*}
\]

if there is no priority between contacts, then $A$ and $B$ might recruit distinct $C$s in $T$, and so, in some sense, the recruitment defeats itself all alone.

Hence, if we have a tree to sort out who is doing the first contact, and who is not, then the problem is solved.
(4) Reversible rules before the rule phase-shift give the group the ability to escape deadlocks. Agents can test whether they can take a step backward without inconsistencies.

4.3.2 A Graphical Explanation of the Protocol

In this section, we give a graphical description of this protocol, which we hope make it easier to understand.

We use ovals to represent proteins; arrows with names to represent the binding from one output of one protein to one input of another protein; the box to represent the group $\tau$. When the arrow is included into the box completely, it says the log of this site is 1, otherwise, the log is 0. The symbol $\uparrow$ represents the feedback respectively, that is, the logs of the correlative input and output are 2. Similarly, the log on site is 3 when there are three arrows $\uparrow\uparrow\uparrow$.

$A$ is the initial protein, so it has not inputs. The rule init (Fig. 4.1) says, as the initial protein, $A$ makes a group ($\tau$) to begin to recruit the next proteins.

The rule FC$_1$ (Fig. 4.2) says that when all inputs of the protein $A$ are recruited into the group $\tau$, $A$ can recruit a new protein $B$ through one of its outputs.

The difference between the rule FC$_2$ (Fig. 4.3) and the rule $FC1$, is that the protein $A$ and $B$ have not connection on sites before the recruitment in the rule FC$_2$. 
The difference between the rule $FC_3$ (Fig. 4.4) and the rules $FC_1$ and $FC_2$, is that the recruited protein $B$ is new in the rule $FC_3$.

The difference between the rules $LC_1$, $LC_2$ (Fig. 4.5 and Fig. 4.6) and the rules $FC_1$, $FC_2$ is the protein $B$ has already been recruited through one of its other input ports in the rules $LC_1$, $LC_2$.

The rule $R$ (Fig. 4.7) says that, when all outputs of the protein $B$ have received the feedback, the $B$ can propagate it to its input sites.

The rule phase-shift (Fig. 4.8) says that, when the initial protein $A$ receives all the feedback from his outputs, it can initiate the succeeding phase of sending a successful message to all the proteins which have been recruited.

The rule propagation (Fig. 4.9) says that, when the protein $B$ receives the successful message from inputs, he will send it to the next proteins from his outputs.

The rule exit (Fig. 4.10) says that, the proteins which take part in the reaction exit when the recruitment is over, which is witnessed by the fact that all input and output sites have the new log $3$ (new $\|$ in the figure).
Figure 4.5: The Rule LC_1..

Figure 4.6: The Rule LC_2..

Figure 4.7: The Rule R.
Figure 4.8: The Rule **phase-shift**.

Figure 4.9: The Rule **propagation**.

Figure 4.10: The Rule **Exists**.
4.4 Mathematical Properties of Self-assembly

In this section, we discuss mathematical properties of self-assembly, which culminate in the proof of the correctness of self-assembly.

In the monotonic protocol, we denote the set of forward rules before the rule phase-shift (ps for short) as \textit{pre-ps} (i.e. all forward rules in Table 4.1 and Table 4.2) and the set of backward rules as \textit{pre-ps}^{-1} (i.e. all backward rules in Table 4.1 and Table 4.2); denote the set of rules after the rule ps as \textit{post-ps} (i.e. all rules in Table 4.3 and Table 4.4 except for the rule ps). The binary relation \(\rightarrow_c\) on \(m\kappa\)-solutions is the union of \textit{pre-ps}^{-1} and \textit{post-ps}.

**Definition 4.3** In the \(m\kappa\)-calculus, let \(\mathcal{R}\) be a set of biological reactions, the associated \(\mathcal{R}\)-system is the pair \((\mathcal{S}, \rightarrow_{m\kappa})\), where \(\mathcal{S}\) is the set of \(m\kappa\)-solutions and \(\rightarrow_{m\kappa}\), called the transition relation, is the least binary relation over \(\mathcal{S}\) such that:

\[
\begin{align*}
\text{mon} & : \quad S, T \models L \rightarrow_{m\kappa} (x) R \in \mathcal{R} \quad \Rightarrow \quad S \rightarrow_{m\kappa} T \quad \text{antimon} \\
\text{new} & : \quad S \rightarrow_{m\kappa} T \quad \Rightarrow \quad (x)(S) \rightarrow_{m\kappa} (x)(T) \\
\text{group} & : \quad S \rightarrow_{m\kappa} T \quad \Rightarrow \quad S, S' \rightarrow_{m\kappa} T, S' \\
\text{struct} & : \quad S \equiv S' \quad S' \rightarrow_{m\kappa} T' \quad T' \equiv T \quad \Rightarrow \quad S \rightarrow_{m\kappa} T
\end{align*}
\]

We shall use the following notation: If \(T\) is a solution, then \(T^*\) denotes the sub-solution consisting of all the proteins of \(T\) of group \(r\).

**Definition 4.4** Consider the following properties of a solution \(T\):

\(INV_p\) Logs 1 and 2 come in pairs, i.e. if one input site \(i\) named \(x\) has log \(n\), then the corresponding output site \(j\) named \(x\) has log \(n\).

\(INV_0\) If log 0 occurs on some protein of \(T\) of group \(r\), then at least one log 1 appears on a protein of that group.

\(INV_1\) If log 1 occurs on some output of a protein of \(T\), then all inputs of that protein have log 1.

\(INV_2\) 1. If log 2 appears on an output of a protein of \(T\), then no input of that protein has log 0 2. If log 2 appears on an input of a protein of \(T\), then all outputs of that protein have log 2.

\(INV_3\) If log 3 occurs on some protein of \(T\) of group \(r\), then no log 0 nor 1 appear on any protein of that group.

We write \(T \models INV\) if \(T\) satisfies all these conditions.
Lemma 4.1 Let $S$ be $\kappa$-solution. $[S]_m \models INV$.

Proof: This property holds vacuously since $[S]_m \models INV$ has no logs.

Lemma 4.2 For any $\kappa$-solution $S$ and $m\kappa$-solution $T$, we have:

$(([S]_m, T) \models INV) \Rightarrow (T \models INV)$

Proof: The statement follows obviously from the observation that the invariants concern only the proteins engaged in a group.

Lemma 4.3

1. If $T_1 \rightharpoonup_{m\kappa} T$, and $T_1 \models INV$, then $T \models INV$;

2. $T_1 \rightharpoonup_{c} T$, and $T_1 \models INV$, then $T \models INV$

Proof: The statement follows easily from the following observations (we consider each invariant in turn):

$INV_p$ All the rules (in either direction for the reversible ones) but $init$, $phase - shift$ and $exit$ are about replacing simultaneously the (identical) logs at the two ends of an edge by a new log. So the invariant is maintained. The three remaining rules do not make any changes to logs 2 or 1 of edges.

$INV_0$ All the redexes have at least one log 1, 2, or 3.

$INV_2$ One checks easily that all the proteins which are appear in the rules and belong to a group either have no output 2 or have no output 0, except possibly protein $B$ in rule $R$ (in either sense) or propagation, but no 0 is introduced on the other side, so the invariant is preserved.

$INV_3$ The set of proteins of group $r$ on the left or on the right side of each rule of groups $|0|$ and $|1|$ has at least one log 1 or 0, hence $INV_3$ says that log 3 does not appear on those proteins (and does not appear either on the left hand site of $init$). Since no 3 is introduced by these rules (in either direction), the invariant is maintained vacuously. The remaining (irreversible) rules do not introduce any logs 0 or 1, hence they preserve the invariant.

$INV_i$ One checks easily that all the proteins which are appear in the rules and belong to a group either have all their input logs at 1 or have not output log at 1 (for rule propagation, we use the fact that $INV_3$ is satisfied (this is the reason why we treat this case last).

Corollary 4.1 If $[S]_m \rightharpoonup^{*}_{m\kappa} T$, then $T \models INV$.

Proof: By lemmas 4.1 and 4.3.

Lemma 4.4 If $T \models INV$, and $T$ is a normal form (NF for short) with respect to $\rightharpoonup_{c}$, then $T$ is the form of $[S]_m$, where $S$ is a $\kappa$-solution.
If \( T \) is not the form of \([S]^m\), then it has at least one log 0, 1, 2, or 3. We show that \( T \) contains a redex.

1. Suppose \( T \) has one log 3 in some group \( r \). Suppose first that there is no log 2 in \( T' \). Then the protein where this log occurs cannot have logs 1, 0, nor 2: hence all its logs are 3 and this protein can exit. If some log 2 appears, take a minimal one, which by \( INV_p \) is on an output of some protein \( A \). By \( INV_3 \) and minimality, all the inputs of \( A \) must be 3. Hence we can apply the propagation rule.

2. Suppose that \( T \) has no log 3 and at least one log 2. Take a minimal such 2, which by \( INV_p \) is on an output of some protein \( A \) while the other end of the edge has also log 2 and is on an input of some protein \( B \). Then \( INV_2 \) guarantees that all input sites of \( A \) have log 1 (no 0 nor 3, nor 2 by minimality), and that all outputs of \( B \) have log 2: hence there is a \( R \) redex.

1. Suppose that \( T \) has no log 3 nor 2, but at least one log 1. Take a maximal such 1. There are two cases.
   - This 1 is on the input of the root. Then the root is an \( init^{-1} \) redex, since all its output logs are 0 by maximality and assumption.
   - Otherwise, by \( INV_p \) this log 1 is on an input of some protein \( B \) while the other end of the edge is on an output site of some protein \( A \). Then \( INV_1 \) guarantees that all input sites of \( A \) have log 1. On the other hand all output sites of \( B \) have log 0 (by maximality and assumption). Hence \( A, B \) form a redex for one of the FC\(^{-1} \) or LC\(^{-1} \) rules.

0. Hence we know that all logs, if any, are 0. But this contradicts \( INV_0 \).

**Lemma 4.5 (Strong Normalization)** The reduction system \( \rightarrow_c \) is strongly normalizing.

**Proof:** Let \( T \) be a solution, with \( n_i \) occurrences of log \( i \) (\( i = 0, 1, 2, 3 \)). We set

\[
\rho(T) = p_0n_0 + p_1n_1 + p_2n_2 + p_3n_3
\]

for some natural numbers \( p_0, p_1, p_2, p_3 \) such that \( 0 < p_0 < p_1 < p_2 > p_3 > 0 \). It is easily checket that if \( T \rightarrow_c T' \) then \( \rho(T') < \rho(T) \), and strong normalization follows.

**Lemma 4.6 (Local Confluence)** The reduction system \( \rightarrow_c \) is locally confluent.

**Proof:** Let \( T \rightarrow_c T'_1 \) and \( T \rightarrow_c T'_2 \), where the respective reductions involve subsolutions \( T_1 \) and \( T_2 \), respectively. The local confluence property is obvious if \( T_1 \) and \( T_2 \) are disjoint subsolutions. Hence we concentrate our intention on the possible overlappings. Then all the proteins of \( T_1 \) and \( T_2 \) bear the same group name: this
follows immediately from the fact that this property holds in $T_1$ and $T_2$ separately, and from the overlapping. Hence we may restrict our attention to the case when the two reductions take place in the simulation of the same macro-reduction step.

**Corollary 4.2 (Confluence)** The reduction system $\rightarrow_c$ is confluent.

**PROOF:** This corollary is from lemma 4.5 and lemma 4.6.

**Lemma 4.7** Let $S$ be a $\kappa$-solution and $T$ be a $m\kappa$-solution. If $[S]_m \rightarrow^*_c T$, then $[S]_m = T$.

**PROOF:** Because all logs of $[S]_m$ are 0, we cannot fire any reduction rule. From $[S]_m \rightarrow^*_c T$, we have $[S]_m = T$.

**Corollary 4.3** Let $S$ and $S'$ be $\kappa$-solutions, if $[S]_m \rightarrow^*_c [S']_m$, then $S = S'$.

**PROOF:** According to lemma 4.7, we have $[S]_m = [S']_m$. Then $S = S'$.

By the corollary 4.1, lemma 4.4 and lemma 4.6.

**Lemma 4.8** For any $m\kappa$-solution $T$ which satisfies $[S]_m \rightarrow^*_c L$, there exists an unique $\kappa$-solution which is denoted as $T^c$ and called the clean-up of $T$ such that $T \rightarrow^*_c [T^c]_m$.

**PROOF:** Assume that there exist two $\kappa$-solutions $S$, $S'$ such that $T \rightarrow^*_c [S]_m$ and $T \rightarrow^*_c [S']_m$. By corollary 4.3, there is a $m\kappa$ solution $T'$ such that $[S]_m \rightarrow^*_c T'$ and $[S']_m \rightarrow^*_c T''$. We have $[S]_m = T' = [S']_m$ by lemma 4.7. And $S = S'$ by corollary 4.3.

**Lemma 4.9** If $T_1 \models INV$ and $T_1 \rightarrow_{ps} T$, then $T_1^c \rightarrow^*_c T^c$.

**PROOF:** The rule of $ps$ is applied on its root protein $A$ in the $\kappa$-reaction $L \rightarrow^*_c (ux)R$ with the group name $r$ and the alias $a$. Since the invariants, we take $T_1$ contains the proteins of $L$ with the group name $r$ such that their logs are 2. Suppose that $T_1^1$ is the form of these proteins. We write $T_1 = T_1^1, T'$. We can apply the rules of $pre-ps^{-1}$, $T_1^1 \rightarrow^*_c [L]_m$. Similarly, we write $T = T_1^1, T'$ and $T_1^1 \rightarrow_{ps} T_1^1$. We have $T_1 \rightarrow^*_c [(ux)R]_m$ by applying the rules of $post-ps$.

Finally, we have;

$$T_1 = T_1^1, T' \rightarrow^*_c [L]_m, T' \rightarrow^*_c [L]_m, [T^c]$$

hence $T_1^c = L, T^c$.

$$T = T_1^1, T' \rightarrow^*_c [(ux)R]_m, T' \rightarrow^*_c [(ux)R]_m, [T^c]$$

hence $T_1^c = (ux)R, T^c$. According to corollary 4.2, $T' \models INV$. Then the last part of these two derivations can be justified. So we have $T_1^c \rightarrow^*_c T^c$.

About the notion clear-up, we have the following proposition;

**Lemma 4.10** If $[S]_m \rightarrow^*_m T$, then $S \rightarrow^*_c T^c$. 

63
PROOF: To proof this proposition, we reduce on the depth of $\llbracket S \rrbracket_m \rightarrow^*_{m\kappa} T$.

If $T = \llbracket S \rrbracket_m$ it it trivial since $\llbracket S \rrbracket_m^c = S$. If $\llbracket S \rrbracket_m \rightarrow^*_{m\kappa} T_1 \rightarrow^*_{m\kappa} T$, and suppose that $S \rightarrow^*_{\kappa} T_1^c$, it is sufficient to consider the following two cases;
(1) $T_1 \rightarrow^*_{m\kappa} T$ by the rule except the rule $ps$. In this case, we have $T_1^c = T^c$ since $\rightarrow^*_{\kappa}$ is confluent and normal form.
(2) $T_1 \rightarrow^*_{ps} T$. Then by the lemma 4.9 $T_1^c \rightarrow^*_{\kappa} T^c$ according to the supposition, we have $S \rightarrow^*_{\kappa} T_1^c \rightarrow^*_{\kappa} T^c$.

Theorem 4.1 (Correctness) Suppose that $T$ is the $m\kappa$-solution and $S$ is the $\kappa$ solution. If $\llbracket S \rrbracket_m \rightarrow^*_{m\kappa} T$, then there exists a $\kappa$ solution $S'$ such that $S \rightarrow^*_{\kappa} S'$ and $T \rightarrow^*_{c}[\llbracket S' \rrbracket_m]$.

PROOF:

Let $S'$ be $T^c$. By lemma 4.10 we prove our theorem. We prove it by induction for the steps of $m\kappa$-reactions.

The base case is that $\llbracket S \rrbracket_m = T$. Let $S' = S$. It is trivial to prove.

The second case is that $\llbracket S \rrbracket \rightarrow^*_{m\kappa} T$. Let $S' = S$. Because all the logs of $\llbracket S \rrbracket$ are 0, we get $T$ only by firing one $m\kappa$-reaction rule before phase shift. Then we can fire the reversible rule from $T$ to $\llbracket S \rrbracket$, that is, $T \rightarrow_{c}[\llbracket S \rrbracket]$.

If $T_1 \rightarrow^*_{m\kappa} T$, and $T_1$ satisfies the correctness (that is, assume that if $\llbracket S \rrbracket \rightarrow^*_{m\kappa} T_1$, then there exists a $\kappa$ solution $S_1'$ such that $S \rightarrow^*_{\kappa} S'$ and $T \rightarrow^*_{c}[\llbracket S_1' \rrbracket]$), by induction, we need consider the two cases as follows;
(1) If $T_1$ fires one of rules before phase shift, then $T$ can fires the reversible rule before phase shift to $T_1$ respectively, that is, $T \rightarrow_{c} T_1 \rightarrow^*_{c} S_1'$.
(2) If $T_1$ fires one of rules after phase shift, that is $T_1 \rightarrow_{c} T$, and by induction, $T_1 \rightarrow^*_{c}[\llbracket S_1' \rrbracket]$, then according to corollary 4, there exists a $m\kappa$-solution $T'$ such that $T \rightarrow^*_{c} T'$ and $\llbracket S_1' \rrbracket \rightarrow^*_{c} T'$. According to lemma 5, $T' = \llbracket S_1' \rrbracket$. Therefore $T$ satisfies the correctness.

Corollary 4.4 Let $S$ and $S'$ be $\kappa$-solutions, if $\llbracket S \rrbracket_m \rightarrow^*_{m\kappa} \llbracket S' \rrbracket_m$, then $S \rightarrow^*_{\kappa} S'$.

PROOF: By theorem 4.1, there exists a $\kappa$-solution $S''$ such that $S \rightarrow^*_{\kappa} S''$ and $\llbracket S'' \rrbracket_m \rightarrow^*_{c} \llbracket S' \rrbracket_m$. By corollary, $S' = S''$. 
Chapter 5

A More General System

In this chapter, we study a more general system, bigraphical reactive systems (BRSs). We choose one biological example as our example to show the expressive power of the BRSs. Translating $\kappa$-reactions to bigraphical reactions, we show informally that the BRSs are more general systems.

The contents of this chapter are organized as follows. In Section 5.1, we discuss the dynamics of BRSs. In Section 5.2, we choose the protein ras activation as an example to show the expressive power of the BRSs. In Section 5.3, we translate $\kappa$-reactions to bigraphical reactions to show informally that the BRSs are more general systems.

5.1 The Dynamics in Bigraphical Reactive Systems

In Section 2.3, we have recalled basic notions of bigraphical reactive systems informally. In this section, we continue discussing the dynamics of BRSs.

5.1.1 The Dynamics of Bigraphs

The dynamics of bigraphs is dedicated to reconfigurations of bigraphs [JM04]. They depend upon both structural components; and there are one or more reaction rules to support them. Each such rule has a redex and reactum. The redex is a precondition for a reaction, represented by a pattern of nesting and linkage; the reactum is a postcondition indicating how the reaction will change that pattern. Places at which reactions may occur are determined by controls. A control $K$ has three states as follows;

— A control $K$ may be atomic, meaning that nothing may be nested within a $K$-node;
— A control $K$ may be active, meaning that reactions may occur within a $K$-node;
— A control $K$ may be passive, meaning that a control $K$ must be destroyed before its inhabitant nodes can react.

A control $K$ is called non-atomic if it is active or passive.
A reaction is modelled as communication and the subsequent change of channel names and nodes.

Figure 5.1: The communication rule.

Fig. 5.1 shows a typical communication rule of the \( \pi \)-calculus;

\[
\pi y.P \mid x(z).Q \rightarrow P \mid Q\{y/z\}
\]

The rule above says that the process \( \pi y.P \) can send name \( y \) via the name \( x \) while the other process \( x(z).Q \) receives this name \( y \) via the same name \( x \).

In bigraphical expressions, we take boxes as bigraphs, grey boxes as holes where other bigraphs can be inserted, ovals denoted by controls as nodes, thin lines denoted by names as links, rings as binding ports.

In formal graphical expressions of bigraphs, we have some notions for expression.

- We use \( N_{xy} \) to represent the node with the control \( N \) has two ports linked by \( x \) and \( y \).
- We use the symbol \( \parallel \) to separate the bigraphs in the redex or the reactum, for example, \( R_1 \parallel R_2 \) represents the redex in Fig. 5.1.
- We use the complex of nodes and holes with their ports to represent bigraphs, for example, we use \( \text{Send}_{xy} \square \) to represent the bigraph \( R_1 \).
- We use \( \mid \) to represent the separation of nodes or links in one bigraph. For example the expression \( N_x \mid M_y \) says that in one bigraph, there exist two nodes \( N \) and \( M \) which have ports linked by \( x \) and \( y \) respectively.

According our notions, the formal graphical expression of Fig. 5.1 is as follows;

\[
\text{Send}_{xy} \square \parallel \text{Get}_{x(z)} \square \rightarrow \square \parallel x \{y/z\} \square
\]

(5.1)
In Fig. 5.1, $R_1$ and $R_2$ are the redex before communication, $R'_1$ and $R'_2$ are reactant after communication. We write the redex of communication as a pair $R_1 \mid R_2$ the reactant as a pair $R'_1 \mid R'_2$. The node (the oval in $R_1$), with the control $Send$, has two ports which are linked by $x$ and $y$ respectively. The node (the oval in $R_2$), with the control $Get$, has two ports as well; one is linked by $x$, and the other is the binding port which is linked with other nodes or holes. (In Fig. 5.1, we use a ring to indicate a binding port.) After the reaction, since nodes with controls $Send$ and $Get$ are destroyed, the common name $z$ is unattached in the reactant. The expression $\{y/z\}$ is represented by a curving link.

Hence, the process $x(y).P$ is represented as $R_1$ in Fig. 5.1, and as $Send_{xy}\square$ in the formal graphical expression (5.1). The process $x(y).Q$ is represented as $R_2$ in Fig. 5.1, and as $Get_{xy}\square$ in the formal graphical expression (5.1). After communication, the process $P$ is represented as a bigraph $R'_1$ in Fig. 5.1 and as $\square$ in the formal graphical expression (5.1). The process $Q[y/z]$ is represented as a bigraph $R'_2$ in Fig. 5.1, and as $x \mid \{y/z\}\square$ in the formal graphical expression (5.1).

As Robin Milner etc. mentioned [JM04], rules in the $\pi$-calculus, the ambient calculus etc., for instance, a reaction rule of the asynchronous $\pi$-calculus, a reaction rule for replication and a reaction rule for summation in the $\pi$-calculus, some reaction rules in the ambient calculus, can be translated into BRSs.

### 5.1.2 The Dynamics of Place Graphs and Link Graphs

Since a bigraph is represented as a combination of two independent mathematical structures: a place graph and a link graph, we can discuss the dynamics of place graphs and link graphs respectively. That is to say, a reaction rule of bigraphs is represented as a combination of a reaction rule of link graphs and a reaction rule of place graphs. This kind of separability of bigraphs can simplify our models. According to our practical goals, we can choose place-graphical models or link-graphical models.

We use $\triangle$ to represent the bigraphs (we take bigraphs as roots), $\triangledown$ to represent the hole (e.g. grey boxes in bigraphs), $\circ$ to represent nodes, and edges represent the inclusion relation of bigraphs, nodes and holes.

Fig. 5.2 shows the reaction rule of place graphs taking place in Fig 5.1.

In Fig. 5.2, “$R_1 \triangle$” represents the bigraph $R_1$ in Fig. 5.1, “$R_2 \triangle$” represents the bigraph $R_2$ in Fig. 5.1. “$\circ$” on the left represents the node with the control $Send$ of $R_1$ in Fig. 5.1, “$\circ$” on the right represents the node with the control $Get$ of $R_2$ in Fig. 5.1. The “$\triangledown$” linked with the left “$\circ$” ($Send$) is the hole (the grey box) of the oval in bigraph $R_1$ in Fig. 5.1. The “$\triangledown$” linked with the right “$\circ$” ($Get$) is the hole (the grey box) of the oval in bigraph $R_2$ in Fig. 5.1. The edge from “$R_1 \triangle$” to the “$\circ$” on the left says that the node ($Send$) $\circ$ belongs to the bigraph $R_1$ in Fig. 5.1. Fig. 5.2 declares the inclusion relation of bigraphs, nodes and holes in Fig. 5.1: the bigraph $R'_1$ has a node with control $send$, this node has a hole; the bigraph $R'_2$ has a node with control $Get$, this node has a hole. After communication, the bigraph $R'_1$ hasn’t nodes any more.
and has a hole. The bigraph $R'_1$ also has one hole.

Fig. 5.3 shows the reaction rule of link graphs. It records the track of links in the reaction of Fig. 5.1. The link graph shares the same set of nodes and holes with the place graph. In fig. 5.3, the "○" on the left still represents the node with the control Send in the bigraph $R_1$ in Fig. 5.1. It has two links $x$ and $y$, and $x$ is also connected to the right node (the node with the control Get in the bigraph $R_2$ in Fig. 5.1). The $\nabla$ on the left is the hole of the node with control Send in the bigraph $R_1$ in Fig. 5.1. Since there is no link between the node with the control Send and its hole in the bigraph $R_1$ in Fig. 5.1, there is no link between the left $\nabla$ and the left "○" in Fig. 5.3. Comparing to the left $\nabla$, the right $\nabla$ is linked with the right "○" since there is one link between the node with the control Get and its hole in the bigraph $R_2$ in Fig. 5.1. After communication, the link $y$ is connected to the hole in the bigraph $R_2$ in Fig. 5.1. The link $x$ is independent.

### 5.2 A Bigraphical Model of ras Activation

In this section, we still choose ras activation as our example (see 5.2) to show the expressive power of the BRSs.

In this model, we give some explanations as follows: first, in formal bigraphical expressions, we use the notation introduced in Section 5.1.1.
Second, in bigraphical expressions, we use the notation introduced in Section 5.1.1.

Third, from the angle of modelling, we take proteins and their functional domains as bigraphs, components residues of domains as links. Nodes with controls are suppositional, that is, there is no direct correspondence in biology, they just make it easier to understand the biochemical process.

Finally, molecular interactions and modification are modelled as communication and the subsequent change of names and nodes.

As mentioned in Section 3.3.2, the whole system consists of four proteins (ras, SOS, RAF and GAP).

Fig. 5.4 shows the system as a bigraph in which there are four subbigraphs representing four kinds of proteins. The grey boxes represent their domains (subsubbigraphs) respectively. (Here we only give domains which take part in this system.)

Next we use reaction rules in BRSs to model biological reactions.

Fig. 5.5 and Fig. 5.6 show how the protein SOS activates the protein ras. It is implemented in two steps. One is the reaction between one domain (S.GNEF) of SOS and one domain (INASWI.I) of ras. The other is the reaction between the domain (S.GNEF) of SOS and the other domain (INASWI.II) of ras. N1 denotes nodes in formal graphical expressions.

In Fig. 5.5, the node N1 in the bigraph INASWI.I has the common link bbone with the node N2 in the bigraph S.GNEF. After the reaction, nodes N1 and N2 are destroyed and the link bbone is unattached.

In Fig. 5.6, the node N1 in the bigraph INASWI.II has the common link sg with the node N3 in the bigraph S.GNEF. After one reaction, nodes N1 and N2 are destroyed and the link sg are unattached. The link rs.1 is sent to the node N4 in the bigraph S.GNEF. Since there is the common link rs.1 between nodes N2 and N4, there is one reaction which sends the link gd from the node N4 in the bigraph S.GNEF to the node N3 in the bigraph ACTSWI.II.

Fig. 5.7 shows the reaction between the domain ACTSWI.I of ras and the do-
Figure 5.5: Ras Activation-1.

The diagram shows the activation of Ras by Akt and the subsequent inactivation by GAP. The reactions are described using the notation of place graphs and link graphs. The main reaction is the activation of Ras by Akt, which is followed by inactivation by GAP.

In our model, we separate the whole proceeding of Ras activation into several reaction rules (from Fig. 5.5 to Fig. 5.9). We represent each protein or domain as a bigraph. Actually, according to our requirement, we can choose different biological objects as bigraphs. For example, we can take a complex as a bigraph, which makes the whole model simpler.

On the other hand, we also can separate each reaction rule in BRSs into two reaction rules in place graphs and link graphs respectively. The place-graphical model and the link-graphical model of Ras activation are shown in Appendix A.2.
Figure 5.6: Ras Activation-2.
Figure 5.7: Signal Transfer.
Figure 5.8: \( ras \) inactivation-1.
Figure 5.9: ras inactivation-2.
5.3 An Example

Bigraphs are graphical structures. The $\kappa$-calculus introduced in Chapter 4 idealizes protein-protein reactions. Strictly speaking, it is some kind of graphical rewriting operating on graphs-on-site. It has a graphical structure as well.

It is easy to translate $\kappa$-reactions to bigraphical reactions. We take solutions as bigraphs, proteins as nodes, sites of proteins as ports, and bindings as links. For instance, we consider a monotonic $\kappa$-reaction as follows:

$$A(1^x + 2), B(1^x + 2), C(1) \rightarrow (y)(A(1^x + 2), B(1^x + 2^y), C(1^y)) \tag{5.2}$$

This monotonic $\kappa$-reaction (5.2) says that, when the site 1 of the protein $A$ binds to the site 1 of the protein $B$ via the name $x$, the site 2 of the protein $B$ will bind to the site 1 of the protein $C$ via the name $y$.

![Figure 5.10: A $\kappa$-reaction.](image)

Fig. 5.10 shows how to represent such a reaction in BRSs. We take proteins $A$, $B$, and $C$ as nodes in a bigraph, and sites of proteins as ports of nodes in the bigraph. The edge $x$ connecting sites of proteins is represented as a link in the bigraph. The monotonic $\kappa$-reaction is actually the generation of a new link $y$ in bigraphical reaction. The formal bigraphical expression of Fig. 5.10 is:

$$A_x | B_x | C \rightarrow A_x | B_{xy} | C_y$$

This example gives us an evidence that the $\kappa$-calculus can be translated into the bigraphical reactive system.

Remarkably, this example is very simple. It gives us an evidence that $\kappa$-calculus can be translated in this way into the bigraphical reactive system.
Chapter 6

Conclusions and Future Work

In this thesis, we have studied three process calculi to model systems biology. We extended the $\pi$-calculus to model the aberrant signal transduction; proved the correctness of self-assembly in $\kappa$-calculus; and made an attempt of modelling $ras$ activation using $BRS$s. These results lay out some foundations for future studies of systems biology and process algebra. They also highlight the robustness of process algebra in modelling and analyzing systems biology.

In the rest of this chapter we discuss possible future work, including several problems that have been left open.

**Generalization of Results** In Chapter 3, the $I\pi$-calculus was obtained by adding two aberrant actions (the suicide action $\|$ and the propagation action $\|$) to the $\pi$-calculus. Does there only exist these two kinds of aberrance in biology? Or could other aberrance be represented by these two actions? We don't know yet. Our motivation started from one detailed case, the aberrant protein $ras$ activation in signal transduction. Compared with computer systems, biological systems are so complicated that our model cannot verify all of them.

In Chapter 4, biological reactions are represented by way of binding or unbinding in $\kappa$-calculus. We focused on the monotonic or antimonotonic reactions, which have been proved to have good mathematical properties. We did not find a good way to deal with other more complicated reactions so far.

In a word, we have the following challenges:

- A list for general features of systems biology? The summary of features of systems biology is necessary because our models stem from these properties.

- A general model for biological processes with aberrance? The $I\pi$-calculus is the first step. On the one hand, since systems biology has various parts, for instance, metabolic pathways, organelles, cells, physiological systems, organisms and so on, we should find appropriate models for these parts respectively. Then we shall try to unify them. On the other hand, maybe the $I\pi$ calculus is the one that can unify the others. It needs to be proved.
• A model for general biological systems? Since BRSs aim to unify the existing process algebras, it may be that model. But this needs future study.

In future work, we would like to go beyond case studies, and we are looking for generalization of results.

Models for quantitative analysis All models in this thesis are for qualitative analysis. They capture the relations among components in biological system. Quantitative analysis is another very important direction in systems biology. We can know more accurately relation among the components and the relation with the environment outside by quantitative analysis. Aviv Regev etc. have taken the Stochastic π-calculus as a quantitatively analytic model to model signal transduction RTK-MAPK and have got some useful results.

The models we designed lack of quantitative analysis. In Chapter 3 we introduced a tag system as an auxiliary system to the Iπ-calculus. The tag system is based on set theory. And tags can represent quantitative information. So we believe that the tag system will be useful in the quantitative analysis.

As one direction in future work, we need to get hold of some points as follows:

• Objects. We cannot analyze quantitatively all the elements in one model. But we can focus on one or some of them. For instance, Aviv Regev etc. focus on effect of concentration of proteins in the model of stochastic π-calculus. Temperature, the number of molecules, intensity of pressure, etc. should be considered if necessary.

• Method. We believe that there exist many other models to make quantitative analysis, other than the tag system. Simplicity and feasibility will be an important criterion.

• Results. Results of models should be consistent with results of biology. This is also the criterion to evaluate our models.

The theoretical development of models Deeper studying of models is necessary. The algebraic properties of models enrich the theory of models themselves. However, when focusing on models for systems biology, those properties which can verify properties of systems biology are more important. The use of formal and algorithmic approaches has greatly accelerated progress in the sequence and structure branches of biology. Adopting a common representation language for biochemical processes may similarly accelerate progress in understanding their function and evolution.

So far, we just considered the expressive power of models. We ensured our model can describe the biological process. For models themselves, we know that the π-calculus, the κ-calculus and BRSs have rich good algebraic properties. How
to transfer this in the systems biology? That is to say, how do we ensure some biological properties using these algebraic properties?

Biological properties in systems biology are different from our computing models; for example, (1) complicated concurrence; (2) enormous entities; (3) the indetermination of factors. All of these are our obstacles in future work.

**Implementation of Models**  The implementation of models is another future research direction. Aviv Regev etc. have developed a computer application, called PiFCP. PiFCP is based on the Logix system, which implements Flat Concurrent Prolog. Our research opens up new possibilities in the study of biochemical systems. However, there are no related automatic tools yet. Designing and implementing such a tool will be our next work.
Appendix A

Some Additional Examples

A.1 The $I\pi$-model with the Tag System about the Protein $ras$

In Section 3.3.2, we have used the $I\pi$-calculus to model the protein $ras$ activation. Here we revisit this example with the tag system. Tags of motifs and residues show that motifs and residues are normal or aberrant, and tags of proteins help to justify whether there exit aberrant components in proteins. By the respective change of tags in reduction rules, it is clearer to know the movement of aberrant components during the process of $ras$ activation.

A.1.1 The Normal State of $ras$

First, the system is defined in (A.1)

\[
< I_{sys}, Sys > ::= < I_{ras}, ras >|< I_{SOS}, SOS >|< I_{GAP}, GAP >
|< I_{RAF}, RAF >
\]

where the tag of the system is:

\[
I_{sys} ::= I_{ras} \uplus I_{SOS} \uplus I_{GAP} \uplus I_{RAF}
\]

The four proteins, $ras$, $SOS$, $GAP$, $RAF$, in this system are given through (A.3)
<i_{ras}, ras> ::= <I_{INASWI,I}, INASWI I> <I_{INASWI,I}, INASWI II> (A.3)

<i_{SOS}, SOS> ::= <I_{S.SH3,BS}, S.SH3,BS>|<I_{S.GNEF}, S.GNEF> (A.4)

<i_{RAF}, RAF> ::= <I_{R,NI}, R_Nt>|<I_{R.ACT,BS}, R.ACT,BS>
|<I_{R,MS, R,MS}>|<I_{INA.R.C1}, INA.R.Ct>
|<I_{R,ATP,BS}, R.ATP,BS> (A.5)

<i_{GAP}, GAP> ::= <i_{sg}, sg(c_ras)>. <i_{c_ras}, \gamma ras(gdp)>. <I_{GAP}, GAP> (A.6)

Respectively, the tags are given as follows:

I_{ras} ::= I_{INASWI,I} \sqcup I_{INASWI,II} (A.7)
I_{SOS} ::= I_{S.SH3,BS} \sqcup I_{S.GNEF} (A.8)
I_{RAF} ::= I_{R,NI} \sqcup I_{R.ACT,BS} \sqcup I_{R,MS} \sqcup I_{INA.R,C1} \sqcup I_{R,ATP,BS} (A.9)
I_{GAP} ::= \bigcup_{n=1}^{\infty} \{\{\check{i}_{sg}, \check{i}_{c_ras}\}\} (A.10)

Since GAP is represented by a recursive process in our model, by the rule t-const in Table 3.2, the tag of GAP is (A.10).

(A.3) shows that the protein ras has two domains which are represented as follows:

<i_{INASWI,I}, INASWI I> ::= <i_{bbone}, bbone > . <i_{r,rs-2} > . <i_{bbone}, bbone >.
<i_{INASWI,I}, INASWI II> (A.11)

<i_{INASWI,II}, INASWI I> ::= <i_{sg}, sg(rs-1) > . <i_{rs-1}, rs-1(x) > .
<i_{sg}, sg(r_sw-1) > . <i_{r_sw-1}, r_sw-1(x) > . <i_{bbone}, bbone >.
<i_{INASWI,II}, INACTSWI ,II> (A.12)

where domains INASWI I and INASWI II are recursive processes. Then the tags are given in (A.13) and (A.14).

I_{INASWI,I} ::= \bigcup_{n=1}^{\infty} \{\check{i}_{bbone}, \check{i}_{rs-2}, \check{i}_{bbone}\} (A.13)
I_{INASWI,II} ::= \bigcup_{n=1}^{\infty} \{\check{i}_{sg}, \check{i}_{rs-1}, \check{i}_{sg}, \check{i}_{r_sw-1}, \check{i}_{bbone}\} (A.14)

The domain S_GNEF of the protein SOS is represented in (A.15):

<i_{S.GNEF, S.GNEF} ::= <i_{bbone}, bbone > . <I_{S.GNEF, S.GNEF}>
<i_{sg}, sg(c_ras) > . <I_{c_ras, \gamma ras(gtp)>. <I_{S.GNEF, S.GNEF} (A.15)
where the respective tag is in \[\text{A.16}\].

\[
I_{S\_GENEF} \ := \ \bigcup_{n=1}^{\infty} \left( \{i_b\} \uplus \{i_{sg}, i_{c\_ras}\} \right)
\]

\[
= \ \bigcup_{n=1}^{\infty} \{i_b, i_{sg}, i_{c\_ras}\}
\]  \hspace{1cm} \text{(A.16)}

The domain \(R\_Nt\) of RAF is defined in \(\text{(A.17)}\):

\[
< I_{R\_Nt}, R\_Nt > \ := \ < i_s, s(c\_ras) > . < i_c, c\_ras > . < I_{ACTR\_Nt}, ACTR\_Nt >
\]  \hspace{1cm} \text{(A.17)}

The respective tag is in \(\text{(A.18)}\).

\[
I_{R\_Nt} = \{i_{s}, i_{c\_ras}\} \uplus I_{ACTR\_Nt}
\]  \hspace{1cm} \text{(A.18)}

The protein SOS activates the protein \(ras\). There are two steps: one is that the domain \(bbone.S\_GENEF\) of the protein SOS interacts with the domain \(INASWI\_I\) of the protein \(ras\), the other is that the domain \(sg(c\_ras).c\_ras(gtp).S\_GENEF\) of the protein SOS interacts with the domain \(INASWI\_II\) of the protein \(ras\). Then the following interactions are possible:

\[
< I_{INASWI\_I}, INASWI\_I > \mid < I_{S\_GENEF}, S\_GENEF > \overset{r_n}{\rightarrow}
\]

\[
< i_s, s(rs\_2). i_{rs\_2}, rs\_2 > . i_{bbone}, bbone > .
\]

\[
< I_{INASWI\_I}, INASWI\_I > \mid < I_{S\_GENEF}, S\_GENEF > \mid ... \hspace{1cm} \text{(A.19)}
\]

\[
< I_{INASWI\_II}, INASWI\_II > \mid < I_{S\_GENEF}, S\_GENEF > \overset{r_{1a}}{\rightarrow}, \overset{r_{2a}}{\rightarrow}
\]

\[
< i_{sg}, sg(rs\_swi\_1). i_{rs\_swi\_1}, rs\_swi\_1(x) > . i_{bbone}, bbone > .
\]

\[
< I_{INASWI\_II}, INASWI\_II > \mid < I_{S\_GENEF}, S\_GENEF > \mid ... \hspace{1cm} \text{(A.20)}
\]

Where

\[
I_1 = \{i_{bbone}, i_{bbone}\}
\]  \hspace{1cm} \text{(A.21)}

\[
I_2 = \{i_{sg}, i_{sg}\}
\]  \hspace{1cm} \text{(A.22)}

\[
I_3 = \{i_{rs\_swi\_1}, i_{c\_ras}\}
\]  \hspace{1cm} \text{(A.23)}

According to Table 3.3, the change of tags in interactions follows some rules.
\[
\bigcup_{n=1}^{\infty} \{ \text{rbone}, \hat{t}_s, \hat{t}_{rs,2}, \hat{t}_{bbone} \} \cup \bigcup_{n=1}^{\infty} \{ \hat{t}_{bbone}, \hat{t}_{sg}, \hat{t}_{c,ras} \} \setminus \{ \hat{t}_{bbone}, \hat{t}_{bbone} \} = \bigcup_{n=2}^{\infty} \{ \hat{t}_{bbone}, \hat{t}_s, \hat{t}_{rs,2}, \hat{t}_{bbone} \} \cup \bigcup_{n=2}^{\infty} \{ \hat{t}_{bbone}, \hat{t}_{sg}, \hat{t}_{c,ras} \}
\]
\[
\bigcup_{n=1}^{\infty} \{ \hat{t}_{sg}, \hat{t}_{rs,1}, \hat{t}_{bbone}, \hat{t}_{bbone} \} \cup \bigcup_{n=1}^{\infty} \{ \hat{t}_{bbone}, \hat{t}_{sg}, \hat{t}_{c,ras} \} \setminus \{ \hat{t}_{sg}, \hat{t}_{sg}, \hat{t}_{rs,1}, \hat{t}_{c,ras} \} = \bigcup_{n=2}^{\infty} \{ \hat{t}_{sg}, \hat{t}_{rs,1}, \hat{t}_{sg}, \hat{t}_{rs,1}, \hat{t}_{bbone} \} \cup \bigcup_{n=2}^{\infty} \{ \hat{t}_{bbone}, \hat{t}_{sg}, \hat{t}_{c,ras} \}
\]
\[
\bigcup_{n=1}^{\infty} \{ \hat{t}_{sg}, \hat{t}_{rs,1}, \hat{t}_{sg}, \hat{t}_{rs,1}, \hat{t}_{bbone} \} \cup \bigcup_{n=1}^{\infty} \{ \hat{t}_{bbone}, \hat{t}_{sg}, \hat{t}_{c,ras} \} = \bigcup_{n=2}^{\infty} \{ \hat{t}_{bbone}, \hat{t}_{sg}, \hat{t}_{c,ras} \}
\]

(A.24)

After interactions (A.19) and (A.20), the protein \( \text{ras} \) is activated and interact with the new protein \( \text{RAF} \). In fact, it is the interaction between the domain \( \overline{\text{rs}_2} \) and \( \text{bbone}.INASWI_I \) of the protein \( \text{ras} \) and the domain \( \text{R}_Nt \) of the protein \( \text{RAF} \).

\[
< \hat{t}_s, \overline{\text{rs}_2} > \cdot \langle \hat{t}_{rs,2}, \overline{\text{rs}_2} \rangle > < \hat{t}_{bbone}, \text{bbone} > .
< I_{\text{INASWI}, \text{J}}, \text{INASWI}_I > |< I_{\text{R}_Nt}, \text{R}_Nt > \xrightarrow{r} ^{*} \langle \hat{t}_{bbone}, \text{bbone} >.
< I_{\text{INASWI}, \text{J}}, \text{INASWI}_I > |< I_{\text{ACTR}_Nt}, \text{ACTR}_Nt > \quad (A.26)
\]

Respectively, the change of tags in the interaction (A.26) is as follows:

\[
\bigcup_{n=1}^{\infty} \{ \hat{t}_{bbone}, \hat{t}_s, \hat{t}_{rs,2}, \hat{t}_{bbone} \} \cup \{ \hat{t}_{bbone}, \hat{t}_{c,ras} \} \cup I_{\text{ACTR}_Nt} \setminus \{ \hat{t}_s, \hat{t}_{rs,2}, \hat{t}_s, \hat{t}_{c,ras} \} = \bigcup_{n=1}^{\infty} \{ \hat{t}_{bbone}, \hat{t}_s, \hat{t}_{rs,2}, \hat{t}_{bbone} \} \cup I_{\text{ACTR}_Nt} \quad (A.27)
\]

Next, the protein \( \text{GAP} \) inactivates the protein \( \text{ras} \), then the protein \( \text{ras} \) comes back to the initial state, that is, the inactive state. The detailed \( I_\pi \)-programme of this inactivation is as follows:

\[
< \hat{t}_{sg}, \overline{\text{sg}(\text{swi}_1)} > \cdot < \hat{t}_{rs,1}, \text{swi}_1(x) > . < \hat{t}_{bbone}, \text{bbone} > .
< I_{\text{INASWI}, \text{J}}, \text{INASWI}_I > |< I_{\text{GAP}}, \text{GAP} > \xrightarrow{r} ^{*} \langle \hat{t}_{bbone}, \text{bbone} > .
< I_{\text{INASWI}, \text{J}}, \text{INASWI}_I > |< I_{\text{GAP}}, \text{GAP} > \quad (A.28)
\]

The protein \( \text{GAP} \) actually interacts with one domain \( \overline{\text{sg}(\text{swi}_1)}, \text{swi}_1(x), \text{bbone} \). \( \text{INASWI}_II \) of the active protein \( \text{ras} \) (A.28). Then the domain \( \text{bbone}.INASWI_{II} \).
interacts with another domain \( \text{bbone}.INASWI.I \) of the protein \( \text{ras} \) (A.29), which the protein \( \text{ras} \) would be in inactive state.

\[
< \hat{\text{bbone}}, \text{bbone} >. \ < \text{INASWI.I}, \text{INASWI.I} > | < \hat{\text{bbone}}, \text{bbone} > .
\]

\[
< \text{INASWI.II}, \text{INASWI.II} > \quad < < \text{INASWI.I}, \text{INASWI.I} > \quad | < \text{INASWI.II}, \text{INASWI.II} >
\]

(A.29) and (A.31) show the relations of tags with respect to reactions (A.28) and (A.29) respectively.

\[
\begin{align*}
\bigcup_{n=1}^{\infty} \{ \hat{z}_{sg}, \hat{z}_{rs,1}, \hat{z}_{sg}, \hat{z}_{r,sw1,1}, \hat{\text{bbone}} \} & \cup \bigcup_{n=1}^{\infty} \{ \hat{z}_{sg}, \hat{z}_{c,ras} \} \setminus \{ \hat{z}_{sg}, \hat{z}_{r,sw1,1}, \hat{z}_{sg}, \hat{z}_{c,ras} \} = \\
\bigcup_{n=2}^{\infty} \{ \hat{z}_{sg}, \hat{z}_{rs,1}, \hat{z}_{bbone} \} & \cup \bigcup_{n=2}^{\infty} \{ \hat{z}_{sg}, \hat{z}_{c,ras} \} = \\
\bigcup_{n=2}^{\infty} \{ \hat{z}_{sg}, \hat{z}_{rs,1}, \hat{z}_{sg}, \hat{z}_{r,sw1,1}, \hat{bbone} \} & \cup \bigcup_{n=2}^{\infty} \{ \hat{z}_{sg}, \hat{z}_{c,ras} \} (A.30) = \\
\bigcup_{n=1}^{\infty} \{ \hat{z}_{bbone}, \hat{i}_{z}, \hat{z}_{rs,2}, \hat{z}_{bbone} \} & \\
\bigcup_{n=1}^{\infty} \{ \hat{z}_{bbone}, \hat{i}_{z}, \hat{z}_{rs,2}, \hat{z}_{bbone} \} \setminus \{ \hat{z}_{bbone}, \hat{i}_{z} \} & \\
\bigcup_{n=1}^{\infty} \{ \hat{z}_{bbone}, \hat{i}_{z}, \hat{z}_{rs,2}, \hat{z}_{bbone} \} & \bigcup_{n=1}^{\infty} \{ \hat{z}_{sg}, \hat{z}_{rs,1}, \hat{z}_{sg}, \hat{z}_{r,sw1,1}, \hat{bbone} \} (A.31)
\end{align*}
\]

From (A.24), (A.25), (A.27), (A.30), (A.31), we observe that, during the process of activation of the protein \( \text{ras} \), 0 never occurs in the tag of the system (Sys).

### A.1.2 The Aberrant State of ras

When \( \text{ras} \) mutates aberrantly, (A.32) defines the \( I_\pi \) representation of GAP in the aberrant state. (A.36) shows that GAP loses its function and does nothing, meaning that it cannot inactivate the domain \( \text{sg}(r_{sw1}.1).r_{sw1}(x).\text{bbone}.INASWI.II \) of \( \text{ras} \).

\[
< I'_{\text{GAP}}, \text{GAP} > ::= < 0, \emptyset (sg(\text{c.ras})) > . < \hat{c}_{r,ras}, \overline{c_{ras}(gdp)} > .
\]

\[
I'_{\text{GAP}} ::= \{ 0 \}
\]

(A.32)\( \quad \) (A.33)

\[
< I'_{\text{GAP}}, \text{GAP} > \rightarrow < 0, 0 >
\]

(A.34)

\[
I'_{\text{GAP}} \setminus \{ 0 \} = \emptyset
\]

(A.35)

The aberrant protein \( \text{ras} \) has the aberrant domain \( \text{INASWI.I} \):

\[
< I'_{\text{INASWI.I}}, \text{INASWI.I} > ::= < \hat{\text{bbone}}, \text{bbone} >, < 0, \emptyset (s(r_{2}).r_{2}) >, < \hat{\text{bbone}}, \text{bbone} >, < I'_{\text{INASWI.J}}, \text{INASWI.I} >
\]

(A.36)

\[
I'_{\text{INASWI.J}} ::= \{ \hat{\text{bbone}} \} \cup \bigcup_{n=1}^{\infty} \{ 0, \hat{i}_{z}, \hat{z}_{rs,2} \}
\]

(A.37)
Hence, the interactions (A.28) and (A.29) cannot happen. While the interaction (A.26) is changed as follows:

\[
< 0, \#(\mathcal{S}(\mathcal{R}.2), i_{ Ras,2}, \mathcal{R}.2) > .< i_{ bbone}, bbone > .
\]

\[
< I'_{INASWI.L}, INASWI.I >|< I_{R..Nt}, R..Nt > \quad \Rightarrow
\]

\[
< 0, \#(\mathcal{S}(\mathcal{R}.2), i_{ Ras,2}, \mathcal{R}.2) > .< i_{ bbone}, bbone > .
\]

\[
< I'_{INASWI.L}, INASWI.I >|< I_{ACTR..Nt}, ACTR..Nt >
\]

\[
\cup_{n=1} \{0, \hat{i}_s, \hat{i}_{ Ras,2}\} \cup \{\hat{i}_s, \hat{i}_{c..ras}\} \cup I_{ACTR..Nt} \setminus \{0, \hat{i}_s, \hat{i}_{ Ras,2}, \hat{i}_s, \hat{i}_{c..ras}\} =
\]

\[
\cup_{n=1} \{0, \hat{i}_s, \hat{i}_{ Ras,2}\} \cup I_{ACTR..Nt}
\]

(A.38)

\[
\text{We can find, on one hand, 0 occurs in the tag of the system which says that aberrance exits in the system. On the other hand, only one 0 is in the tag of GAP which says that GAP has the suicide capability, and unlimited 0s are in the tag of INASWI.I which says that INASWI.I has the propagation capability. We cannot check whether INASWI.I has the suicide capability or not only by the tag system.}
\]

\section{A.2 The Link and Place Graphical Model of ras Activation}

In this section, we give the link graphical and place graphical model of ras activation, respectively. The reaction rules of BR5s in Section 5.2 are combinations of reaction rules of link graphs and reaction rules of place graphs.

According to Fig. 5.4, the place graph of the system is shown in Fig. A.1.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure_a1.png}
\caption{The place graph of the system.}
\end{figure}

The system in Fig. [A.1] includes four bigraphs (proteins).
Fig. A.2 and Fig. A.3 are the place graphs and the link graphs when the protein SOS activates the protein ras.

In Fig. A.2, the place graph says that the domain INASWI_I has one subdomain ("\(\nabla\)") which is controlled by one node \(N^1\) ("\(\circ\)"), and the domain S_GNEF has two subdomains in which one is controlled by nodes \(N^2\) and the other is controlled by nodes \(N^3\) and \(N^4\). After the reaction, nodes \(N^1\) and \(N^2\) are destroyed.

The link graph says that, before the reaction, nodes \(N^1\) and \(N^2\) are bounded by \(bbone\). After the reaction, since nodes \(N^1\) and \(N^2\) are destroyed, the link \(bbone\) becomes independent.

In Fig. A.3, the place graph says that the domain INASWI_II has one subdomain ("\(\nabla\)") which is controlled by nodes \(N^1\) and \(N^2\), and the domain S_GNEF has two subdomains in which one is controlled by node \(N^3\) and the other is controlled by nodes \(N^4\) and \(N^5\). After two reactions, nodes \(N^1\), \(N^2\), \(N^4\) and \(N^5\) are destroyed.

The link graph says that, before the reaction, nodes \(N^1\) and \(N^4\) are bounded by \(sg\). After one reaction, since nodes \(N^1\) and \(N^2\) are destroyed, the link \(sg\) is independent and the link \(rs_1\) is sent to the node \(N^5\) by the link \(c_ras\). Then nodes \(N^2\) and \(N^5\) have the common link \(c_ras\). After the other reaction, the link \(gtp\) is sent to the subdomain (the left "\(\nabla\)") by the link \(z\). Nodes \(N^2\) and \(N^5\) are destroyed and the link \(rs_1\) becomes independent.

In the bigraphical example of ras activation, reaction rules of place graphs show the occurrences of biological reactions, and link graphs show the direction of biological reactions. In other words, reaction rules of place graphs ensure the occurrence of biological reactions, while reaction rules of link graphs describe how biological reactions occur.

Fig. A.4 are reactions that the activated ras sends the signal to the next protein RAF in place graphs and in the link graphs respectively.

In Fig. A.4, the place graph says that the domain ACTSWI_I has one subdomain controlled by three nodes \(N^1\), \(N^2\) and \(N^3\), and the domain R_Rt has one subdomain controlled by two nodes \(N^4\) and \(N^5\). After two reactions, nodes \(N^1\), \(N^2\), \(N^4\) and \(N^5\) are destroyed.

The link graph says that nodes \(N^1\) and \(N^4\) have the common link \(s\) and there is one reaction such that \(N^1\) and \(N^4\) are destroyed, the link \(s\) is independent, and the link \(rs_2\) is sent to the node \(N^5\) by the link \(c_ras\). So there is another reaction such since nodes \(N^2\) and \(N^5\) have the common link \(rs_2\). After the second reaction, nodes \(N^2\) and \(N^5\) are destroyed and the link \(rs_2\) becomes independent.

Fig. A.5 and Fig. A.6 are the reactions showing how the activated ras returns back to the initial inactive state in place graphs and in link graphs respectively.

In Fig. A.5, the place graph says that, the domain ACTSWI_II has one subdomain controlled by nodes \(N^1\), \(N^2\) and \(N^3\), and the domain GAP has one subdomain controlled by nodes \(N^4\) and \(N^5\). After two reactions, nodes \(N^1\), \(N^2\), \(N^4\) and \(N^5\) are destroyed.

The link graph says that, nodes \(N^1\) and \(N^4\) have the common link \(sg\) and there is a reaction such that the link \(r_sw1_1\) is sent to the node \(N^5\) by the link \(c_ras\). Then
In Fig. A.6, the place graph says that, the domain \( r_{swi.1} \), there is another reaction such that the link \( gdp \) is sent to the node \( N^3 \) by the link \( x \). During the two reactions, nodes \( N^1, N^4, N^2 \) and \( N^5 \) are destroyed and links \( sg \) and \( r_{swi.1} \) are independent.

In Fig. A.6, the place graph says that, the domain \( bbone.INASWI.I \) has one subdomain controlled by one node \( N^1 \), and the domain \( bbone.INASWI.II[gtp/x] \) has one subdomain controlled by the node \( N^2 \). After reaction, the node \( N^1 \) is destroyed while the node \( N^2 \) becomes active. The link graph says that, one reaction occurs since nodes \( N^1 \) and \( N^2 \) have the common link \( bbone \). After the reaction, the node \( N^1 \) is destroyed and the link \( bbone \) becomes independent.
Figure A.3: The place graph and the link graph of ras activation-2.

Figure A.4: The place graph and the link graph of signal transfer.
Figure A.5: The place graph and the link graph of ras inactivation-1.

Figure A.6: The place graph and the link graph of ras inactivation-2.
Appendix B

Some Proofs in the Thesis

B.1 The Proof of Proposition 3.4

Proposition 3.3 [Strengthening] Assume that the term $M$ is not free in the process $P$ and that $N \neq M$. The following properties hold:

1. If $\Gamma, M : T \vdash N : S$, then also $\Gamma \vdash N : S$.
2. If $\Gamma, M : T \vdash P : Ok$, then also $\Gamma \vdash P : Ok$.

Proof: (1) The judgement $\Gamma, M : T \vdash N : S$ must be established through the rule Level Term with $\vdash \Gamma, M : T$ well formed, and $N : S$ in $\Gamma, M : T$. Then the judgement $\vdash \Gamma, M : T$ well formed must be established through the rule Environment Term with $\vdash \Gamma$ well formed. Because $N \neq M$, we have the fact that $N : S$ in $\Gamma$. Hence we have $\Gamma \vdash N : S$ using the rule Level Term.

(2) The second proposition is obtained by induction over the structure of $P$.

(a) Case 0. The judgement $\Gamma, M : T \vdash 0 : Ok$ must be established through the rule $T$-nil with $\vdash \Gamma, M : T$ well formed. The judgement $\vdash \Gamma, M : T$ well formed must be established through the rule Environment Term with $\vdash \Gamma$ well formed. So $\Gamma \vdash 0 : Ok$ by the rule $T$-nil again.

(b) Case $\pi_i.P$. The judgement $\Gamma, M : T \vdash \pi_i.P : Ok$, must be established through one of the rules ($T$-out, $T$-in, $T$-sout, $T$-sin, $T$-kout, $T$-kin, $T$-ksout, $T$-ksin, $T$-pout, $T$-pin, $T$-psout and $T$-psin) with $\Gamma, M : T \vdash a : R$, ($\Gamma, M : T \vdash b : Normal$ or $\Gamma, M : T \vdash x : Unknown$), and $\Gamma, M : T \vdash P : Ok$, where $R \in \{Normal, Aberrant\}$. By induction hypothesis we have $\Gamma \vdash P : Ok$. Hence we have $\Gamma \vdash \pi_i.P : Ok$ also by one of the rules above.

(c) Case $P \mid Q$. The judgement $\Gamma, M : T \vdash P \mid Q : Ok$ must be established through the rule $T$-com with $\Gamma, M : T \vdash P : Ok$ and $\Gamma, M : T \vdash Q : Ok$. By induction hypothesis we have $\Gamma \vdash P : Ok$, and $\Gamma \vdash Q : Ok$, we have $\Gamma \vdash P \mid Q : Ok$ also
by the rule \(T\)-com. Similarly, we can get the same results for \(P + Q\) using the rule, \(T\)-sum.

(d) Case \((\nu a)P\). The judgement \(\Gamma, M : T \vdash (\nu a)P : Ok\) must be established through one of rules \(T\)-res and \(T\)-ares with \(\Gamma, M : T, a : R \vdash P : Ok,\) where \(R \in \{Normal, Aberrant\}\) By induction hypothesis we have \(\Gamma, a : R \vdash P : Ok,\) we have \(\Gamma, M : T \vdash (\nu a)P : Ok\) using the rule \(T\)-res or \(T\)-ares again.

### B.2 The Proof of Proposition 3.5

**Proposition 3.4** [Weakening] Assume that \(M\) is not defined in the environment \(\Gamma,\)

1. If \(\Gamma \vdash N : S,\) then \(\Gamma, M : T \vdash N : S.\)
2. If \(\Gamma \vdash P : Ok,\) then \(\Gamma, M : T \vdash P : Ok.\)

**Proof:** (1) The judgement \(\Gamma \vdash N : S\) must be established through the rule \(Level Terms\) with \(\vdash \Gamma \ well \ formed\), and \(N : S\) is in \(\Gamma.\) Because \(M\) is not defined in the environment \(\Gamma,\) we have \(\vdash \Gamma, M : T \ well \ formed\) by the rule \(Environment Term.\) Since \(N : S\) is in \(\Gamma,\) of course, \(N : S\) is in \(\Gamma, M : T,\) hence we have \(\Gamma, M : T \vdash N : S\) using the rule \(Level Term.\)

(2) It is obtained by induction over the structure of \(P.\)

(a) Case 0. The judgement \(\Gamma \vdash 0 : Ok\) must be established through the rule \(T-nil\) with \(\vdash \Gamma \ well \ formed.\) For \(M\) is not defined in the environment \(\Gamma,\) then we have \(\vdash \Gamma, M : T \ well \ formed\) by the rule \(Environment Term.\) So \(\Gamma, M : T \vdash 0 : Ok\) by the rule \(T-nil\) again.

(b) Case \(\pi_i.P.\) The judgement \(\Gamma \vdash \pi_i.P : Ok\) must be established through one of the rules \((T\text{-out}, T\text{-in}, T\text{-sout}, T\text{-sin}, T\text{-kout}, T\text{-ksout}, T\text{-ksin}, T\text{-pout}, T\text{-pin}, T\text{-psout} \text{ and } T\text{-psin})\) with \(\Gamma \vdash a : R, (\Gamma \vdash b : Normal \text{ or } \Gamma \vdash x : Unknown)\) and \(\Gamma \vdash P : Ok\) where \(R \in \{Normal, Aberrant\}.\) By induction hypothesis we have \(\Gamma, M : T \vdash P : Ok.\) Hence we have \(\Gamma, M : T \vdash \pi_i.P : Ok\) also by the rules above.

(c) Case \(P \mid Q.\) The judgement \(\Gamma \vdash P \mid Q : Ok\) must be established through the rule \(T\text{-com}\) with \(\Gamma \vdash P : Ok\) and \(\Gamma \vdash Q : Ok.\) By induction hypothesis we have \(\Gamma, M : T \vdash P : Ok,\) and \(\Gamma, M : T \vdash Q : Ok,\) then we have \(\Gamma, M : T \vdash P \mid Q : Ok\) also by the rule \(T\text{-com}.\) Similarly, we can get the same results for \(P + Q\) using the rule \(T\text{-sum.}\)

(d) Case \((\nu a)P\). The judgement \(\Gamma \vdash (\nu a)P : Ok\) must be established through one of rules \(T\text{-res} \text{ and } T\text{-ares}\) with \(\Gamma, a : R \vdash P : Ok,\) where \(R \in \{Normal, Aberrant\}.\) By induction hypothesis we have \(\Gamma, a : Normal, M : T \vdash P : Ok,\) we have \(\Gamma, M : T \vdash (\nu a)P : Ok\) also by the rule \(T\text{-res} \text{ or } T\text{-ares.}\)
B.3 The Proof of Proposition 3.6

Proposition 3.5 Assume that \( \vdash \Gamma \) well formed and that terms in \( \text{dom}(\Gamma) \) are all normal. Then the following properties hold:

1. If \( M \) is a term and \( M \in \text{dom}(\Gamma) \), then \( \Gamma \vdash M : \text{Normal} \).
2. If \( P \) is a process with \( f_n(P) \cup f_v(P) \subseteq \text{dom}(\Gamma) \), then \( \Gamma \vdash P : \text{ok} \).

Proof: (1) The former proposition is obtained trivially by the rule Level Term.

(2) The latter proposition is obtained by induction over the structure of \( P \).

(a) The base case is that \( \Gamma \vdash 0 : \text{Ok} \) by the rule T-nil.

(b) Case \( \pi_i.Q \) where \( f_n(\pi_i.Q) \cup f_v(\pi_i.Q) \subseteq \text{dom}(\Gamma) \). Then \( f_n(Q) \cup f_v(Q) \subseteq \text{dom}(\Gamma) \). By induction hypothesis we have \( \Gamma \vdash Q : \text{Ok} \), hence we have \( \Gamma \vdash \pi_i.Q : \text{Ok} \) using one of the rules Level Subsumption, T-out, T-in, T-sout, T-sin, T-kout, T-kin, T-k sout, T-pout, T-pin, T-psout, and T-psin.

(c) Case \( R | Q \) where \( f_n(R | Q) \cup f_v(R | Q) \subseteq \text{dom}(\Gamma) \). Then \( f_n(R) \cup f_v(R) \subseteq \text{dom}(\Gamma) \) and \( f_n(Q) \cup f_v(Q) \subseteq \text{dom}(\Gamma) \). By induction hypothesis we have \( \Gamma \vdash R : \text{Ok} \), and \( \Gamma \vdash Q : \text{Ok} \), and hence we have \( \Gamma \vdash R | Q : \text{Ok} \) using the rule T-com. Similarly, we can get the same result for \( P + Q \) using the rule T-sum.

(d) Case \( (\nu a)Q \) where \( fn((\nu a)Q) \cup fv((\nu a)Q) \subseteq \text{dom}(\Gamma) \). If \( a \in \text{dom}(\Gamma) \), then \( fn(Q) \cup fv(Q) \subseteq \text{dom}(\Gamma) \), where \( \Gamma \) can be written as \( \Gamma', a : \text{Normal} \) and \( a \notin \text{dom}(\Gamma') \). By induction hypothesis we have \( \Gamma \vdash P : \text{Ok} \), that is, \( \Gamma', a : \text{Normal} \vdash Q : \text{Ok} \). We have \( \Gamma' \vdash (\nu a)Q : \text{Ok} \) using the rule T-Res. By Proposition 3.5, we have \( \Gamma \vdash (\nu a)Q : \text{Ok} \). If \( a \notin \text{dom}(\Gamma) \), then \( fn(Q) \cup fv(Q) \subseteq \text{dom}(\Gamma) \cup \{a\} \). From \( \vdash \Gamma \) well formed and \( a \notin \text{dom}(\Gamma) \), we get \( \vdash \Gamma, a : \text{Normal well formed} \). By induction hypothesis we have \( \Gamma, a : \text{Normal} \vdash Q : \text{Ok} \), hence we have \( \Gamma \vdash (\nu a)Q : \text{Ok} \) using the rule T-Res.
Bibliography


