Reaction Constraints for the Pi-Calculus - A Language for the Stochastic and Spatial Modeling of Cell-Biological Processes

Mathias John

Reaction Constraints for the Pi-Calculus

A Language for the Stochastic and Spatial Modeling of Cell-Biological Processes

Dissertation

zur

Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

der Fakultät für Informatik und Elektrotechnik

der Universität Rostock

vorgelegt von

Dipl. Inf. Mathias John, geb. am 1982/02/14 in Teterow

aus Rostock

Rostock, 2010/07/07

Contents

1 Introduction **1**

1.1 Formal Modeling to Study Cell-Biological Processes . 5

1.2 The π-Calculus to Model Cell-Biological Processes . . 10

1.3 Contribution 16

1.3.1 The Attributed π-Calculus 22

1.3.2 The Imperative π-Calculus 25

1.4 Related Work 27

1.5 Outline 33

1.6 Bibliographic Note 33

2 The π-Calculus with Priority **39**

2.1 The π-Calculus with Priority 39

2.1.1 Design Decisions 40

2.1.2 Syntax of Processes 41

2.1.3 Non-deterministic Operational Semantics . . . 45

2.1.4 Uniqueness of Convergence 48

2.1.5 Stochastic Operational Semantics 51

2.1.6 Type System 57
2.2 BioAmbients with Priority 61
2.2.1 Syntax of Processes 62
2.2.2 Non-deterministic Operational Semantics . . . 68

3 The Attributed π-Calculus 73
3.1 Language 73
3.1.1 Idea of Communication Constraints 74
3.1.2 Attribute Languages 75
3.1.3 Syntax of Processes 80
3.1.4 Non-deterministic Operational Semantics . . . 82
3.1.5 Uniqueness of Convergence 86
3.1.6 Stochastic Operational Semantics 88
3.1.7 Type System 89
3.2 Modeling Techniques and Biological Examples 98
3.2.1 Spatial Aspects: Euglena's Phototaxis 98
3.2.2 Cooperative Enhancement: Gene Regulation at the Lambda Switch 106
3.2.3 Population-based Modeling 110
3.2.4 Global Information in Individual-Based Modeling 114
3.2.5 Species-Based Modeling 116
3.3 Expressiveness 116
3.3.1 Encoding of the π-Calculus with Priority . . . 118
3.3.2 Encoding $\pi[@,\neq]$ for Dynamic Compartments 125
3.3.3 Variants of the Stochastic Pi-Calculus 131

3.4 Stochastic Simulator 135
3.5 Implementation and Performance Evaluation 142

4 The Imperative π-Calculus 147
4.1 Language 148
4.1.1 Idea of a Global Imperative Store 149
4.1.2 Design Decisions 151
4.1.3 Attribute Languages 153
4.1.4 Syntax of Processes 158
4.1.5 Non-Deterministic Operational Semantics . . . 162
4.1.6 Uniqueness of Convergence 168
4.1.7 Stochastic Semantics 172
4.2 Modeling Techniques and Biological Examples 177
4.2.1 Osmosis: Variable Volumes and Surfaces . . . 178
4.2.2 Michaelis-Menten kinetics à la sCCP 184
4.3 Expressiveness 194
4.3.1 Conservativeness 194
4.3.2 Encoding BioAmbients 200
4.4 Stochastic Simulator 210

5 Conclusion 215

A Experiment Results 223

B Remaining Proofs 225
B.1 Section 3.1.7 (Type System) 225
B.2 Section 3.3.1 (Encoding the π-Calculus with Priority) . 227

B.3 Section 3.3.2 (Encoding $\pi[@, \neq]$ for Dynamic Compartments) 238

Bibliography **243**

List of Figures

1.1 A realization of a CTMC based on the kinetic law of Mass action. 8

2.1 Syntax of the π-calculus with priority. 42
2.2 Free channel names of the π-calculus with priority. . . 42
2.3 Axioms of the structural congruence of the π-calculus with priority. 43
2.4 Rules of the non-deterministic semantics of π-calculus with priority. 46
2.4 Rules of the non-deterministic semantics of π-calculus with priority. 47
2.5 Rules of the stochastic semantics of the π-calculus with priority. 52
2.6 Example of a CTMC generated by the stochastic semantics of the π-calculus with priorities. 55
2.7 Type system for π-calculus with priority. 59
2.8 Syntax of BioAmbients with channels $x, \tilde{x}, \tilde{y} \in \mathit{Vars}$ and priorities $r \in R$. 63

2.9 Free channel names of BioAmbients. 65
2.10 Rules of the structural congruence of BioAmbients. . . 66
2.11 Communication directions and rearrangement capabilities of BioAmbients. 67
2.12 Rules of operational semantics of BioAmbients with priorities in $(R,<)$. 69
2.12 Rules of the operational semantics of BioAmbients with priorities in $(R,<)$(continued). 70
2.12 Rules of the operational semantics of BioAmbients with priorities in $(R,<)$(continued). 71

3.1 Big-step evaluator of call-by-value λ-calculus with pairs and conditionals. 77
3.2 Additional rules of big-step evaluator of the attribute language $\lambda(\mathbb{N}_0,+,=)_{<_1}$. 77
3.3 Syntax of $\pi(\mathscr{L})$, where $x,\tilde{x} \in Vars$, and $e_1,e_2,\tilde{e} \in Exprs$. 80
3.4 Non-deterministic operational semantics of $\pi(\mathscr{L})$. . . 83
3.5 Rules of the stochastic semantics of $\pi(\mathscr{L})$. 88
3.6 Type system for $\pi(\mathscr{L})$. 93
3.6 Type system for $\pi(\mathscr{L})$ (continued). 94
3.7 A model of Euglena's light-dependent motion with two light sources. 101
3.8 Master Equation to compute the numbers of Euglenas on each depth level in equilibrium. 103
3.9 Euglena model, Experiment A. 104
3.10 Euglena model, Experiment B. 105

3.11 An equivalent model of Euglena in the stochastic π-calculus. 107
3.12 The decay of the rep-OR_2 complex. 108
3.13 A model of cooperative binding between OR_1 and OR_2 at the λ switch. 109
3.14 A population-based model of three species and two reactions in $\pi(\lambda(\mathbb{R},+,-,*,/,pow,\leq)_{<_1})$. 113
3.15 An individual-based variant of the population-based model in Figure 3.14. 115
3.16 A species-based variant of the population-based model in Figure 3.14 in $\pi(\lambda(\mathbb{R},+,-,*,/,pow,\leq)_{<_1})$. 117
3.17 The encoding of the π-calculus with priorities $(R,<)$ into $\pi(\lambda(R)_<)$. 118
3.18 Naive simulator interpreting the stochastic semantics of $\pi(\mathscr{L})$. 137
3.19 Stochastic simulator for $\pi(\mathscr{L})$. 140
3.20 Runtime of different simulators for the Euglena model. 145

4.1 Big-step evaluator of a call-by-value λ-calculus with pairs and conditionals. 155
4.2 Additional rules of the big-step evaluator of the attribute language $\lambda(\mathbb{N}_0,+,=)_{<_1}$. 156
4.3 Abbreviations for expressions of the attribute language $\mathscr{L}$. 157
4.4 Syntax of $\pi^{imp}(\mathscr{L})$ where $x,\tilde{x}\in Vars$, $e_1,e_2,\tilde{e}\in Exprs$, and $f,\tilde{f}\in Exprs^-$. 159

4.5 Free names of $\pi^{imp}(\mathscr{L})$. 160
4.6 Axioms of structural congruence of $\pi^{imp}(\mathscr{L})$. 161
4.7 Non-deterministic operational semantics of $\pi^{imp}(\mathscr{L})$ with priority levels in $(R,<)$. 163
4.7 Non-deterministic operational semantics of $\pi^{imp}(\mathscr{L})$ with priority levels in $(R,<)$ (continued). 164
4.7 Non-deterministic operational semantics of $\pi^{imp}(\mathscr{L})$ with priorities in $(R,<)$ (continued). 165
4.8 Rules of stochastic semantics of $\pi^{imp}(\mathscr{L})$. 173
4.9 Example of a CTMC generated by $\pi^{imp}(\mathscr{L})$. 175
4.10 A model of osmosis with variable compartment volume and surface. 179
4.10 A model of osmosis with variable compartment volume and surface (continued). 180
4.11 Experiment results without (Model A) and with (Model B) variable surfaces. 183
4.12 A population-based model of an enzymatic reaction network with Mass action kinetics in $\pi^{imp}(\lambda(\mathbb{R},\mathtt{K},\mathtt{N}))$. 188
4.12 A population-based model of an enzymatic reaction network with Mass action kinetics in $\pi^{imp}(\lambda(\mathbb{R},\mathtt{K},\mathtt{N}))$ (continued). 189
4.13 A population-based model of an enzymatic reaction network with Michaelis-Menten kinetics in $\pi^{imp}(\lambda(\mathbb{R},\mathtt{K},\mathtt{N}))$. 190
4.14 simulation experiments to compare Mass action with Michaelis-Menten. 192

4.15 Auxiliary functions for the encoding of BioAmbients to $\pi^{imp}(\mathscr{L})$. 203
4.16 Encoding of processes from BioAmbients to $\pi^{imp}(\mathscr{L})$. 203
4.17 Encoding of prefixes from BioAmbients to $\pi^{imp}(\mathscr{L})$. . 204
4.18 Graphical representation of BioAmbients encoding. . . 209
4.19 Stochastic simulator for $\pi(\mathscr{L})$. 211

List of Tables

4.1 Parameters and constants used in osmosis experiments. 181
4.2 Parameters and constants used in experiments to compare models based on Michaelis-Menten and Mass action kinetics. 191

A.1 Runtime of different simulators for the Euglena model. 224

Acknowledgements

When I started my PhD, I had, due to my background in computer graphics, rather little knowledge of formal methods, the π-calculus, modeling and simulation of cell-biological systems, or, for that matter, any other topic that my thesis required me to be an expert in. Therefore, I thank my supervisors Lin Uhrmacher and Joachim Niehren for their endless patience and trust and their great efforts to close the tremendous gaps in my knowledge and to guide me on the way to a successful PhD thesis. I promise that I will keep on learning. I am also grateful to my colleagues of the research training school dIEM oSiRiS and the working group of modeling and simulation who offered me valuable insights into all the topics covered by the research training school, especially into the fields of biology and simulation, and supporting me on the most challenging task of understanding JAMES II. In particular, I thank Roland Ewald for helping me to take my first steps in the field of formal methods. Beyond work, I found in my colleagues and also my supervisors sophisticated and pleasant partners for leisure activities like sports, BBQ, or wine tasting. The moments spent together formed very welcomed interruptions from

long working days and remain as precious memories. I thank the Deutsche Forschungsgemeinschaft (DFG) for funding me in the context of the research training school dIEM oSIRIS. This gave me the chance to work interdisciplinary at a high level and to go for a two-month stay at the INRIA in Lille, France, where I got to know my supervisor Joachim Niehren and my collaborator Cédric Lhoussaine. Also my personal live was impacted by the research training school in a very positive way as one of my colleagues, Orianne, became my girlfriend. I thank her for the unlimited understanding and encouragement whenever needed. I thank my parents and grandparents for permanently being concerned about my mental, physical, and financial state, and in particular my grandpa for always offering me fresh eggs. I thank my brother for being my best friend.

Chapter 1

Introduction

Life scientists exhaustively investigate the processes in living cells not only to progress in curing devastating diseases, like cancer or AIDS, or to push the limits of aging, but also to understand life itself. To this end, the greatest challenge is to deal with the complexity of cell-biological processes, which is a result not just of the large number of players involved but also of their numerous ways of interacting. The traditional approach to understanding complex systems is to follow the reductionist's agenda, i.e. to separately gather all information about their constitutes in order to obtain the overall picture. Yet, rather recently, scientists came to the conclusion that is is the interaction of the players of cell-biological processes in particular that creates the overall picture. Thus, to understand cell-biological systems, they can only be regarded as a whole with an emphasis on molecular interplay (Pollard, 2003).

Dynamic systems form the basic metaphor for cell-biological sys-

tems. Instead of static networks describing the interdependencies between the players, it is the changes in amounts, structures, interaction capabilities, and locations which have to be observed and understood. Therefore, the temporal dimension is of significant importance. On one hand, it allows to simplify the complex overall picture of cell-biological systems as some options can be eliminated. For example, the passive motion of molecules (diffusion) is often considered to be too slow, such that only active forms of transport, i.e. movements under energy consumption, are plausible, see e.g. Krieghoff et al. (2006). On the other hand, the speed of processes has significant impact on their actual outcome. Cases are known where the race condition is a regulatory tool, such as reported by Merino and Yanofsky (2005) for the transcriptional attenuation at the tryptophan operon.

Concurrency strongly influences the dynamics of cell-biological processes. In computational theory, concurrent systems consist of processes that independently run in parallel but compete for common, limited resources, see e.g. Bowman and Gomez (2005) for an introduction. A prominent example in databases is given by two customers of an airline interested in booking the same seat on a single flight. Processes in cells are mostly performed in parallel and many of them compete for common, limited resources in the form of molecules in low abundance. Prominent examples are the simultaneously running transcription and translation processes in bacteria, see e.g. Ralston (2008), or the crosstalk in signaling pathways, see e.g. Nagao et al. (2007).

Non-determinism is inherent to concurrent systems, since processes

running in parallel do not follow a concrete order of events. Thus, limited resources may be distributed in different ways, often with significant impact on the behavior of cell-biological systems. For example, a protein's production involves a reading process of the genetic code (transcription). For transcription to start, a reader (polymerase) has to bind to a specific region of the DNA, called promoter. As a regulatory tool, promoters may be occupied by some molecules different from the polymerase, such that reading cannot start. Thus, the order of binding events at promoters significantly influences a protein's abundance and consequently the intensiveness with which its functions are performed, see e.g. Alberts et al. (2002).

Probabilities assigned to specific orders of concurrent events form a restriction on pure non-determinism. In their essence, cell-biological processes are stochastic since molecules perform Brownian motion, i.e the direction and velocity of a molecule moving in a fluid is mainly determined by its collision with others. For a more abstract view disregarding the fluid, the location of a molecule at a certain time point can be described by a stochastic process, see e.g. Lawler (2006). Stochastic processes distinguish a system's different states and describe the probability of a system being in a certain state at a certain time, where time can be of the domain $\mathbb{N}_0$ or $\mathbb{R}^+$, i.e. discrete or continuous. Thus, the event of two molecules being at a location sufficiently close to interact, e.g. to bind, at a certain time point can be associated with a probability. Gillespie (1977) identified Continuous Time Markov Chains (CTMC's) as the basic type of stochastic processes to describe molecular systems. The essential property of a CTMC is that the prob-

ability for a system to evolve from state S to state S' is entirely determined by S - the predecessors of S do not need to be considered. Whether the stochastic aspects of cell-biological processes have significant impact on their outcome, i.e. lead to observable stochasticity, depends on different factors, see e.g. Wolkenhauer et al. (2004). McAdams and Arkin (1999) point out that a major criterion is that some players of central importance appear in low abundance.

Recently, the location of proteins was identified to have a major impact on cell-biological processes, see e.g. Kholodenko (2006); Chebotareva et al. (2004); Takahashi et al. (2005). Proteins are major players in cell-biological processes; they are large molecules with many, often independent, binding sites, which allow them to interact in a variety of ways. Space in eukaryotic cells is partitioned by intra-cellular structures, such as membranes. Cell parts completely enveloped by membranes form compartments, for example the cytosol or the nucleus. As a basic regulatory concept in cells, membranes distinctively hinder molecules in their movement (semi-permeability). Thus, proteins are not equally distributed in intracellular space. At different locations, proteins perform different functionality. For example, the main player β-Catenin of the Wnt/β-catenin signaling pathway has to move from the cytosol to the nucleus for a cellular response to occur, see e.g. Miller et al. (1999). Spatial aspects like the location and motion of molecules are therefore of increasing interest when studying cell-biological processes.

1.1 Formal Modeling to Study Cell-Biological Processes

Formal modeling is a well-established method to investigate the dynamics of cell-biological systems, see e.g. Kitano (2002a). The basic idea is to transform the system under study into a formal representation, expressed in a modeling formalism, and to analyze the latter often with the help of computers. The transformation process forces the existing knowledge base to be structured, often revealing insufficiencies, contradictions, and ambiguities. The analysis step allows the validation of theories or predictions about the system under study to be made. Often, the investigation of cellular processes is regarded as an iterative process alternating between experimental work in the wet-lab and computational modeling in the dry-lab, see e.g. Kitano (2002b). Although very promising, in practice, projects combining wet-lab work with computational modeling are challenging to realize, often taking several years until scientific contributions can be achieved. Asides from the complexity of cell-biological systems, this is mainly due to the interdisciplinarity of such projects, since wet-lab experiments are usually performed by life scientists, whereas modeling is done by computer scientists. Communication between experts of these two scientific fields turns out to be difficult, not only because of discrepancies in the area-specific knowledge but also since the basic understanding of research is very different: whereas life scientists aim to gain detailed knowledge about the different players and aspects of the system un-

der study, the essential tool for computer scientists to obtain insights is abstraction.

Reaction rules are the basic paradigm to describe cell-biological processes on the molecular level. They textually or graphically represent the chemical reactions occurring between chemical species. In its basic form, a reaction rule specifies the transformation of a set of reactants into a set of products, e.g. $r_1 : Na, Cl \rightarrow NaCl$. As discussed e.g. by (Gillespie, 1977), atomic reactions have at most two reactants, since at a specific point in time at most two molecules may collide. Reaction rules of a higher order, as e.g. $r_2 : H_2, H_2, O_2 \rightarrow H_2O, H_2O$, allow one to abstract from details which are either not of interest or unknown, like the order in which the three molecules bind. However, pure reaction rules are only syntactic constructs, which help to structure knowledge but do not provide information about the dynamics of cell-biological processes.

By providing a formal semantics for reaction rules one takes the step from a purely descriptive formalism that can only be used for structuring knowledge to a full-fledged modeling language which also allows studying the dynamics of cell-biological processes. Nondeterministic transition systems assign state spaces to sets of reactions rules, where solutions form the different states. Thereby, a solution is understood as a multiset of instances of chemical species, e.g. $S = \{Na, Na, Cl, Cl\}$ for a solution of two molecules of sort Na and Cl, respectively. In order to determine the successors and the corresponding transitions of a single state, the given reaction rules are applied to the members of the solution. Those rules that lead to the same solu-

tion, i.e. to the same successor state, are combined to a single transition. For example, applying reaction rule r_1 to the members of solution S yields a single transition to the successor state $S' = \{Na, Cl, NaCl\}$. Approaches like non-deterministic transition systems that only consider the mere order of events but not any temporal aspects are usually referred to as qualitative modeling methods.

Semantics in terms of CTMC's consider stochasticity by assigning a propensity to each transition. Starting from a state S, the ratio between the propensity of a single transition leading to state S' and the propensities of all transitions starting from S captures the probability of S' to be the actual successor state. The absolute value of the propensity reflects the time that is consumed when reaching S'. Since a semantics in terms of CTMC's allows the consideration of time, it qualifies as a basis for quantitative modeling methods. In order to derive the propensities of a CTMC, it is presumed that each reaction rule is annotated with a stochastic rate constant k, e.g. $Na, Cl \xrightarrow{k} C$, which defines the affinity of a reaction as a frequency, i.e. with a unit $1/\text{time}$. Based on some kinetic law, the amounts of molecules in the current solution and the reaction rate constants are combined to a reaction rate providing the propensity. As the basic kinetic law, the law of Mass action, specifies the rate of a reaction as the product of the amounts (or concentrations) of the reactants, representing the number of different molecular interactions captured by the rule, times the rate constant (assuming distinct reactants). For illustration consider two chemical species Na and Cl

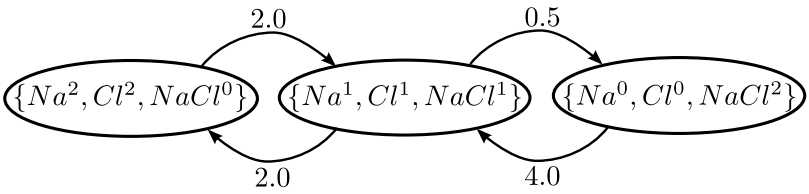


Figure 1.1: A realization of a CTMC based on the kinetic law of Mass action. It describes the possible states of a chemical solution with two molecules of species *Na* and *Cl* and rate constants $k_1 = 0.5$, $k_2 = 2.0$, following the two reactions above.

and the following reactions with rate constants k_1, k_2:

$$\begin{aligned} bind: &\quad Na, Cl \xrightarrow{k_1} NaCl \\ decay: &\quad NaCl \xrightarrow{k_2} Na, Cl \end{aligned}$$

Assuming a solution $S = \{Na^2, Cl^2, NaCl^0\}$, with X^n the multiplicity n of species X is denoted, and rate constants $k_1 = 0.5s^{-1}$, $k_2 = 2.0s^{-1}$, the CTMC in Figure 1.1 is obtained. Other kinetic laws exist, e.g. Michaelis-Menten kinetics or Hill kinetics, which allow to describe more abstract reactions and thus to deal with missing knowledge, e.g. missing rate constants. The decision on which kinetic to apply depends on different factors, see e.g. Millat et al. (2007). Aside from CTMC's, other formalisms can be used to define semantic objects. The most prominent example are ordinary differential equations (ODE's), which so far form the main approach to studying the dynamics of cell-biological processes (for an introduction see e.g. Fall et al., 2002; Klipp et al., 2009).

Gillespie (1977) proposed the stochastic simulation algorithm

(SSA) to analyze models with CTMC semantics by building timed traces through their state space. These can be compared to experimental data from the wet-lab or other references, e.g. existing models, allowing for statements about a model's validity. Since they only produce traces, simulation methods can always be applied, even when faced with very large or infinite state spaces. However, interesting observations may be excluded. Beyond simulation, different forms of model checking can be used as analysis methods, see e.g. Kwiatkowska et al. (2004); Fages and Rizk (2007).

Existing languages for the modeling of cell-biological systems subscribe to different modeling styles, i.e. ways in which the systems under study are described. Rule-based approaches, like the κ-calculus (Danos and Laneve, 2004; Faeder et al., 2005), allow to directly write down reaction rules similar to those shown above. By contrast, object-centered approaches, like the π-calculus (Milner, 1999), rather induce the view of interacting molecules. Furthermore, population-based and individual-based approaches are distinguished, whereas the former rather focus on the amounts of species, e.g. sCCP (Bortolussi and Policriti, 2008b), the latter consider the distinct states of molecules, e.g. the κ-calculus or the π-calculus. Notice, that these two classifications are basically independent dimensions, such that a language may be object-centered and population-based, e.g. Bio-PEPA (Ciocchetta and Hillston, 2009), and vice versa, e.g. the κ-calculus. More specifically, the Bio-PEPA approach can be regarded as species-based, since objects represent species.

1.2 The π-Calculus to Model Cell-Biological Processes

Priami et al. (2001) suggested that the π-calculus is a well-suited language to model cell-biological processes. Milner et al. (1992) introduced the π-calculus to the field of concurrency theory as a formalism to specify the interaction of concurrent processes based on a nondeterministic semantics. A stochastic semantics for the π-calculus in terms of CTMC's was first developed by Priami (1995), introducing the stochastic π-calculus, and later refined by Kuttler (2006). The π-calculus thus naturally covers some of the basic aspects of cell-biological processes. Regev (2003), Phillips and Cardelli (2007), Kuttler et al. (2007), and Leye et al. (2010) proposed stochastic simulators for the stochastic π-calculus building on the SSA. Regev (2003) and Kuttler (2006) provided ways of transforming reaction rules into π-calculus-models. Based on this work, several studies on modeling cell-biological processes in the π-calculus have been realized so far, see e.g. Kuttler and Niehren (2006); Mazemondet et al. (2009); Cardelli et al. (2009); Schaeffer (2008).

The basic idea of mapping reaction rules to π-calculus-models is to abstract molecules as communicating processes and reactions as communications over channels. For example, the reaction rules *bind* and *decay* above can be specified in the π-calculus as follows:

$$\mathsf{Na}() \triangleq \mathit{bind} : k_1 \,!\, () \,.\, \mathsf{NaCl}()$$
$$\mathsf{Cl}() \triangleq \mathit{bind} \,?\, () \,.\, \mathbf{0}$$

$\mathsf{NaCl}() \triangleq \mathit{decay} : k_2 \,!\, () \,.\, (\mathsf{Na}() \mid \mathsf{Cl}())$
$\mathsf{T}() \triangleq \mathit{decay} \,?\, () \,.\, \mathsf{T}()$

Processes Na(),Cl(), and NaCl() are defined to represent the molecules of the different chemical species. In the π-calculus, communication always happens between two processes, one sending ("!") and the other receiving ("?") on the same channel. Here, processes Na() and Cl() may interact on channel *bind* and processes NaCl() and T() on channel *decay*, representing the reaction rules *bind* and *decay*, respectively. Process T() is an artifact to provide a communication partner for the single reactant NaCl() of rule *decay*. Rate constants are included by annotating them to the sender, e.g. $\mathit{bind}{:}k_1$. Reaction products are given by the successors of an interaction provided behind the operator "." by both interaction partners. For example, when processes Na() and Cl() communicate, Na() proceeds with process NaCl() and Cl() with the idle process **0**, denoting its consumption. Thus, on their interaction, processes Na() and Cl(), are replaced by process NaCl(), which reflects the reaction *bind*. When process T() and NaCl() interact, T() proceeds recursively, ready to serve as a partner for the next *decay* reaction, and NaCl() with Na() | Cl(). Thereby, operator "|" defines a parallel composition of processes, here of Na and Cl, representing chemical solutions. Consequently, an initial solution with two molecules of each species (and an interaction partner for NaCl()) is specified by:

$$\mathsf{Na}() \mid \mathsf{Na}() \mid \mathsf{Cl}() \mid \mathsf{Cl}() \mid \mathsf{NaCl}() \mid \mathsf{NaCl}() \mid \mathsf{T}()$$

The parentheses behind process names or sending and receiving ac-

tions may be filled with lists of channels in order to define abstract species which are parametric on the reactions they are involved in or to denote channels that are exchanged on interaction, respectively. Notice that, since communication in the π-calculus always happens between two processes, modeling reactions with more than two reactants is not possible. The decision on which reactants are mapped to senders and receivers or which reactant is consumed or proceeds with a complex is arbitrary.

Although useful for modeling reaction networks, the π-calculus has its limitations when studying the spatial aspect of cell-biological processes. On one hand, this regards the description of molecule locations in compartments. On a basic level, it is necessary to reflect that molecules are only able to interact if they are in the same compartment. More advanced modeling also considers the location dependency of the functionality of proteins and their affinity to interact. To implement such aspects in the π-calculus means to enumerate the different species and communication channels with their rate constants for each location. However, this approach yields models of high complexity in terms of the number of process definitions and channels when studying proteins which are largely location dependent in their functionality or when considering a fine-granular spatial resolution. Moreover, considering space to be a dimension of a continuous domain, it cannot be applied.

On the other hand, modeling dynamic cell structures like merging compartments is also problematic in the π-calculus. Merging compartments induce a change in global information. By a single event,

namely the merging, entire sets of molecules are impacted, since the location of all the molecules in the compartments changes in the sense that their interaction capabilities are increased, e.g. molecules that were separated before may now interact, or decreased, e.g. a motion from one compartment to the other is not possible anymore. For a consistent model state after a compartment merging, it needs to be ensured that the local information in all processes is updated before any other interaction may happen. This is challenging since it contradicts the essential π-calculus idea of locally and independently interacting processes. Besides merging compartments, changes in global information are also caused by dynamics in compartment volumes or temperature.

Expressiveness studies allow making formal statements about the aspects of the systems under study that can be implemented in a certain modeling language. Due to its leanness and strong theory, the π-calculus has been largely used as a basis for expressiveness studies, see e.g. Versari et al. (2009); Palamidessi (2003); Ene and Muntean (1999). Depending on the desired result, expressiveness studies in the π-calculus can be designed in different ways. For example, to show that one π-calculus has at least the same expressiveness as another, an encoding from the latter to the first is provided. Therefore, encodings are required to meet certain criteria. The most important one is that they respect compositionality, i.e. the introduction of a central unit, which implements a protocol to control all interactions should be avoided. More formally, if P is a process in some π-calculus and $[\![P]\!]$ its encoding, then for parallel compositions it should hold that $[\![P_1 \mid P_2]\!] = [\![P_1]\!] \mid [\![P_2]\!]$. An encoding not preserving compositionality

violates the essential π-calculus idea of locally and independently interacting processes. Compositionality is also relevant for the modeling of cell-biological processes, since it supports model refinement, an essential tool in the iterative modeling process. That is, in the absence of a central interaction protocol, changing the interaction of molecules requires only limited modifications. Notice that the actual quest is not to find the most expressive language but the one with optimal expressiveness to model cell-biological processes: too little expressiveness is certainly problematic as significant aspects of the system under study may not be included in the model or only with a reduced level of detail, i.e. at a higher abstraction level. In such cases, it is not the modeler who chooses the abstraction level, as desired, but rather it is dictated by the modeling language. Too much expressiveness, however, may lead to error-prone and inaccessible models. Moreover, due to large state spaces, model analysis is burdened by high computational costs, often up to an impractical level.

Regev et al. (2004) introduce BioAmbients as a spatial extension of the π-calculus. It introduces ad-hoc operators that allow locating processes in possibly nested ambients which represent compartments, and provides ad-hoc operators for communication of processes in ambients, e.g. from an ambient to a contained ambient, and to change ambient structures, e.g. for merging ambients. However, ad-hoc operators are a disadvantage, since each operation that slightly deviates from the provided ones requires an extension of the language or, strictly speaking, a new language. Furthermore, languages with ad-hoc operators are hard to compare, which complicates the task of finding a language

with optimal expressiveness for the modeling of cell-biological processes.

Versari (2009) proposed $\pi@$ for the modeling of dynamic cell structures, a π-calculus which avoids ad-hoc operators but introduces two additional, orthogonal concepts, polyadic synchronization and priority. Polyadic synchronization extends communication by considering tuples of channels, e.g. $x@y@z$. A communication can only happen if the channel tuples of the sender and the receiver match. This allows modeling process interactions that can only happen if the interaction partners are at the same location, e.g. $\mathit{bind}@\mathit{Cytosol}{:}k_1!().\mathsf{NaCl}() \mid \mathit{bind}@\mathit{Cytosol}?().\mathbf{0}$. Discrete motion, i.e. jumps between two locations, can be modeled by making process definitions parametric over locations, e.g. $\mathsf{Na}(l_1) \triangleq \mathit{move}@l_1?(l_2).\mathsf{Na}(l_2)$. Priority levels are assigned to interactions - an interaction with some priority level may only be executed if no interaction of higher priority is enabled. Their idea to model dynamic cell structures is to implement protocols that update the position of each process one by one and assigning them a higher priority than interactions that represent reactions. Priority is thus used to ensure that first the local information of all processes is updated before any interaction happens that may lead to an inconsistent model state. An expressiveness study was performed which indicates that $\pi@$ is sufficiently expressive for the modeling of dynamic cell-structures by providing a compositional encoding of BioAmbients into $\pi@$. Versari and Busi (2009) introduce a stochastic version of $\pi@$, called S$\pi@$. In addition to dynamic cell-structures, it offers ways to model changes in compartment volumes. However, the correspond-

ing operations are hard-wired to the stochastic semantics, which gives them a certain ad-hoc flavor. Furthermore, the stochastic semantics of $\pi@$ is not defined in terms of CTMC's but makes use of a short-cut directly mapping to the SSA.

1.3 Contribution

In this thesis, we introduce reaction constraints for the π-calculus as a language for the stochastic and spatial modeling of cell-biological processes. Reaction constraints allow making the occurrence of reactions dependent on the attribute values of their reactants. Attributes of different types, e.g. Booleans, numbers, or Strings allow the specification of properties, like locations of molecules ($\mathsf{Na}(\mathtt{cyt})$), compartment volumes ($\mathsf{Cytosol}(10.5)$), or the occupation of a protein's binding sites ($\mathsf{Prot}(\mathtt{free},\mathtt{bound})$). For instance, with a reaction constraint one may ensure that the chemical species *Na* and *Cl* only interact if they are in the same compartment:

$$nacl_{pos}\text{: } \forall p_1, p_2 \in \{\mathtt{cyt},\mathtt{nuc}\}.\quad \mathsf{Na}(p_1), \mathsf{Cl}(p_2) \xrightarrow{\textbf{if } p_1{=}p_2 \textbf{ then } k \textbf{ else } 0.0} \mathsf{NaCl}(p_1)$$

Reaction constraints replace the usual rate constant by a function that computes a reaction's affinity dependent of the attribute values of the reactants. By this, they denote the step from reaction rules to rule schemes. Rule schemes include variables (p_1, p_2) and thus compactly represent sets of reaction rules. The example above yields the follow-

ing two reaction rules:

$$nacl_{cyt}: \quad \mathsf{Na}(\mathtt{cyt}), \mathsf{Cl}(\mathtt{cyt}) \xrightarrow{k} \mathsf{NaCl}(\mathtt{cyt})$$
$$nacl_{nuc}: \quad \mathsf{Na}(\mathtt{nuc}), \mathsf{Cl}(\mathtt{nuc}) \xrightarrow{k} \mathsf{NaCl}(\mathtt{nuc})$$

In fact, the number of represented reaction rules can grow up to infinity depending on the domains (types) of variables.

The results of reaction constraints are not restricted to real numbers. We define a set of successful values which specifies those results that allow a reaction to occur. In the example above, if the positions p_1 and p_2 are equal, the rate constant k is associated with the reaction; otherwise the constraint returns the value 0.0. Thus, as usual in a stochastic setting, the set of all reaction rate constants, i.e. the positive real numbers, are considered to be successful. By contrast, a study on the mere non-deterministic order of events may only require a single successful value, e.g. the Boolean `true`.

We introduce communication constraints as the counterpart of reaction constraints in the π-calculus. Communication constraints consist of two parts: the constraint argument at the sender side of the communication and the constraint function at the receiver side. Whether a communication may occur is determined by applying the constraint function to the constraint argument. For example, the following process definitions model the reaction schema $nacl_{pos}$ above, where process NaCl() is left unspecified:

$$\mathsf{Na}(p) \triangleq \mathtt{nacl}_{\mathtt{pos}}[p]!() \,.\, \mathsf{NaCl}()$$
$$\mathsf{Cl}(p) \triangleq \mathtt{nacl}_{\mathtt{pos}}[\lambda q.\textbf{if } p = q \textbf{ then } k \textbf{ else } 0.0]?() \,.\, \mathbf{0}$$
$$\mathsf{NaCl}() \triangleq \ldots$$

Constraint functions and arguments are annotated in square brackets. In a solution $\mathsf{Na}(\mathtt{cyt}) \mid \mathsf{Cl}(\mathtt{cyt})$, applying the function $\lambda q.\mathbf{if}\ \mathtt{cyt} = q\ \mathbf{then}\ k\ \mathbf{else}\ 0.0$ to the argument cyt yields the term $\lambda q.\mathbf{if}\ \mathtt{cyt} = \mathtt{cyt}\ \mathbf{then}\ k\ \mathbf{else}\ 0.0$. Assuming an appropriately defined operator "=", this evaluates to the successful value k enabling the communication.

We extend the π-calculus with communication constraints, considering both the non-deterministic and stochastic semantics, and by this obtain the attributed π-calculus. Based on a modeling study in the attributed π-calculus, we show that communication constraints are useful to model molecule locations, discrete motion, and location dependent reaction affinities. In order to also be able to express changes in global information, we pursue two orthogonal approaches, priority and a global imperative store.

Adapting the idea of Versari (2009), we design the attributed π-calculus, such that its non-deterministic and stochastic semantics consider interactions with different levels of priority. These may be used to implement prioritized update protocols. To ensure the usefulness of the attributed π-calculus for the modeling of dynamic cell-structures, we show that it can express $\pi@$. It became apparent during the preparation of this thesis that polyadic synchronization is not sufficient to express molecule locations as in BioAmbients. We propose a small extension of polyadic synchronization, called $\pi[@,\neq]$, to fix this problem, which allows checking for inequality of channel names. Then we develop a compositional encoding from $\pi[@,\neq]$ to the attributed π-calculus and prove its correctness with respect to the non-deterministic

semantics. This shows that the attributed π-calculus has sufficient expressiveness to model dynamic cell structures and is thus well-suited for the spatial modeling of cell-biological processes. Since there exists no stochastic version of $\pi@$ which includes priority and a semantics in terms of CTMC's, we do not have the basis for a formal expressiveness study with respect to the stochastic semantics of the attributed π-calculus. However, rate constants can be introduced into our encoding in a straightforward way, which lets us conclude that it also works in the stochastic realm. Based on different modeling studies, we show that the attributed π-calculus additionally allows the application of different modeling-styles, population-based and species-based styles in addition to the inherent object-centered, individual-based styles.

Our idea of a global imperative store is to allow communication constraints to access and change the values of global variables. This implies twofold: on one hand, values in the store directly impact the results of communication constraints. On the other hand, communication constraints can change the values in the store. Changes in variable values caused by a communication constraint are committed to the global store when the corresponding communication occurs. Simultaneously, the constraints of all communications are re-evaluated, introducing side effects of process interactions. Variables are integrated in the π-calculus by allowing channels to map to values. Consider, e.g., the chemical species *Na* and *Cl* to be in separate compartments $\texttt{comp}_1$ and $\texttt{comp}_2$, respectively. As soon as $\texttt{comp}_1$ and $\texttt{comp}_2$ merge, *Na* and *Cl* should be able to react as in the examples above. In $\pi^{imp}(\mathscr{L})$, this can be modeled as follows, where again process NaCl() is left unspec-

ified:

Global variables

$p_1 : \mathtt{comp}_1$

$p_2 : \mathtt{comp}_2$

Process definitions

$\mathsf{Na}(p) \triangleq \mathtt{nacl}_{\mathtt{pos}}[p]!() \,.\, \mathsf{NaCl}()$

$\mathsf{Cl}(p) \triangleq \mathtt{nacl}_{\mathtt{pos}}[\lambda q.\textbf{if } (\mathtt{val}\ p) = (\mathtt{val}\ q) \textbf{ then } k$
$\textbf{else } 0.0]?() \,.\, \mathbf{0}$

$\mathsf{Merge}() \triangleq act[\lambda\_.p_1 := (\mathtt{val}\ p_2);\ k']?() \,.\, \mathbf{0}$

$\mathsf{T}() \triangleq act[\_]!() \,.\, \mathbf{0}$

$\mathsf{NaCl}() \triangleq \ldots$

Initial solution

$\mathsf{Na}(p_1) \mid \mathsf{Na}(p_1) \mid \mathsf{Cl}(p_2) \mid \mathsf{Cl}(p_2) \mid \mathsf{Merge}() \mid \mathsf{T}()$

Global variables, i.e. channels, p_1 and p_2 represent the positions of *Na* and *Cl*. If their values equal, p_1 and p_2 refer to the same position. The constraint function of process definition $\mathsf{Cl}(p)$ is slightly modified, such that operator `val` is applied to the global variables in order to access their values. By this, the current positions of processes $\mathsf{Na}(p_1)$ and $\mathsf{Cl}(p_2)$ are obtained. Initially, the values of p_1 and p_2 are set to compartment names $\mathtt{comp}_1$ and $\mathtt{comp}_2$, respectively, denoting the different locations. The constraint function of process $\mathsf{Merge}()$ is defined, such that on interaction with process $\mathsf{T}()$ it executes the compartment merging. The expression $\lambda\_$ denotes the function with no parameter, i.e. the constraint is entirely defined by the constraint function. Expressions $e_1; e_2$ define sequences whose return value is determined by

e_2. In the case of process Merge(), this is considered to be some rate constant for compartment merging k'. The second part of the sequence assigns the value of variable p_2 to variable p_1. Thus, as a side effect, communication of processes T() and Merge() sets the positions of processes $\mathsf{Na}(p_1)$ and $\mathsf{Cl}(p_2)$ to be of equal value, which enables their interaction.

We extend the non-deterministic and stochastic semantics of the attributed π-calculus, such that global variables are considered, by this obtaining the imperative π-calculus. Based on a modeling study, we show the usefulness of the imperative π-calculus for the modeling of changing compartment volumes. For this, our approach does not rely on special operators hard-wired to the stochastic semantics as in Sπ@ and thus promises more flexibility. In fact, we underline this point by showing that in the imperative π-calculus the model can be extended such that it also considers changes in compartment surfaces. Furthermore, based on another modeling study, we show that the imperative π-calculus also allows the implementation of reactions with Michaelis-Menten kinetics in a population-based style. This denotes the first successful attempt to model reactions with Michaelis-Menten kinetics in a π-calculus-based approach.

We perform a formal expressiveness study which follows the approach of Versari (2009) to develop a compositional encoding of BioAmbients. The main idea is to check molecule location with communication constraints. Global variables are used to model changes in global information. We prove that our encoding is correct with respect to the non-deterministic semantics and by this also show that the

imperative π-calculus is well-suited for the spatial modeling of cell-biological processes. Brodo et al. (2007) also proposed a stochastic semantics for BioAmbients which, however, does neither include priority nor is defined in terms of CTMC's, such that we miss a basis for a formal expressive study with respect to the stochastic semantics. However, rate constants can be introduced into our encoding in a straightforward way, which lets us conclude that it also works in the stochastic realm.

Since priority and global variables similarly support the modeling of changes in global information as resulting from dynamic cell-structures, a study directly comparing the expressiveness of both concepts is desirable, not only to find a language with an optimal expressiveness for the spatial modeling of cell-biological processes but also for concurrency theory in general. We take a first step in this direction by developing an encoding of a restricted imperative π-calculus which may only read but not change global values to the attributed π-calculus.

The following reports on the technical contributions regarding the attributed and the imperative π-calculus in detail.

1.3.1 The Attributed π-Calculus

We contribute the attributed π-calculus ($\pi(\mathscr{L})$) that extends the π-calculus with species attributes, communication constraints, and priority. The parameter $\mathscr{L}$ of $\pi(\mathscr{L})$, denotes an attribute language $\mathscr{L}$, in which attributes and constraints are expressed. The attribute language

is chosen to be a parameter in order to allow the modeling language to be adapted to the application at hand. The attribute language $\mathscr{L}$ is a λ-calculus (see e.g. Barendregt, 1985). Thus, it supports modularity, since functional constants can be defined and reused to specify constraints and attributes. For example, relations on locations, like functions to calculate distances, can be introduced. Despite the expressiveness of $\mathscr{L}$, simulation performance is not hampered, as first tests suggest. The attribute language also fixes the set of successful values. The number of priority levels is specified as a partial order on the successful values, i.e. as part of $\mathscr{L}$. Thus, although $\pi(\mathscr{L})$ is capable of handling any number of priority levels, it can also be restricted to a single one, if it turns out to be enough for the application at hand.

We present two operational semantics for $\pi(\mathscr{L})$, a non-deterministic and a stochastic one in terms of CTMC's. In contrast to Phillips and Cardelli (2007), their definitions do not rely on any syntactic restrictions, like biochemical forms. The stochastic semantics is a refinement of the non-deterministic semantics - we show that the latter permits the same reduction steps as the former. The stochastic semantics of $\pi(\mathscr{L})$ basically adopts the kinetics of the stochastic π-calculus semantics, i.e. Mass action, with the only difference that, as shown above, reaction constraints make a reaction's rate dependent on the attribute values of the interaction partners.

Based on the SSA, we develop a stochastic simulator for $\pi(\mathscr{L})$ which is directly derived from the stochastic semantics and independent of the attribute language $\mathscr{L}$. First, we propose a naive version. Then we show that the basic idea of Phillips and Cardelli (2007) to gain

more efficiency, that is, grouping interactions by the channels they are performed on, can also be applied. Performance studies with a first implementation suggest that by this a practical efficiency is achieved, despite the high expressiveness of $\mathscr{L}$.

We perform several small modeling studies to show the usefulness of $\pi(\mathscr{L})$ for the modeling of cell-biological processes on different levels of abstractions. On the cellular level the phototaxis of Euglena serves as an example to detail how location and motion can be modeled in $\pi(\mathscr{L})$. An example on the molecular level regards cooperative enhancement, a common regulatory tool in cell-biological processes (see e.g. Ptashne and Gann, 2001), and depicts how constraints allow simplifying an existing model as proposed by Kuttler and Niehren (2006). Moreover, we study the applicability of $\pi(\mathscr{L})$ to different modeling styles; in addition to the inherent individual-based style, we also examine the population-based and the species-based style. A further example focuses on how changes in global information can be modeled in individual-based $\pi(\mathscr{L})$-models by using prioritized update protocols.

Our expressiveness results formally underline the usefulness of $\pi(\mathscr{L})$. Encodings of the π-calculus with priorities, both the nondeterministic and the stochastic versions, are provided and proved to be correct. Furthermore, encodings of different variants of the π-calculus (SPiCO (Kuttler et al., 2007), BioSPi (Regev, 2003), SPiM (Phillips and Cardelli, 2007)) are developed, by this showing that $\pi(\mathscr{L})$ is a unifying approach. As discussed above, we provide an encoding of an extended version of $\pi@$ and its proof of correctness

to show that the attributed π-calculus is well-suited for the spatial and stochastic modeling of cell-biological processes.

We propose a simple type system for $\pi(\mathscr{L})$. Type systems help to ensure that model implementations behave correctly. They do so by providing and checking additional rules on the syntax for proving the absence of certain erroneous model behaviors (for an introduction see e.g. Pierce, 2002). This is especially important when developing modeling languages for experts from domains different from computer science. In the particular case here, the type system even ensures that the evaluation of expressions in the attribute language always halts, since a simply typed λ-calculus (Church, 1940) is obtained.

1.3.2 The Imperative π-Calculus

We introduce the imperative π-calculus ($\pi^{imp}(\mathscr{L})$) which is a conservative extension of $\pi(\mathscr{L})$ with respect to the non-deterministic and the stochastic semantics. Syntactically it only differs in that values are assigned to channels, representing global variables. Consequently, as in the example above, the attribute language needs to provide additional operators to access and change channel values. Semantically, the main difference is that, due to the global variables, pairs of processes and global stores need to be considered. As before $\pi^{imp}(\mathscr{L})$ provides priority levels for interactions, where the number of priority levels is defined as part of $\mathscr{L}$, such that it can be set to one if desired.

We formally underline the conservativeness of $\pi^{imp}(\mathscr{L})$ by providing two encodings. One shows that every process in $\pi(\mathscr{L})$ can be

transformed into a $\pi^{imp}(\mathscr{L})$-process just by assigning a dummy value to all channels. Conversely, considering an attribute language for the imperative π-calculus without assignments, there exists an encoding from the imperative to the attributed π-calculus. This shows on the one hand that in the imperative π-calculus, $\pi(\mathscr{L})$-processes can be defined almost transparently. On the other hand, the only actual difference between the two calculi is the possibility to change the values of variables. This observation denotes the first step to a direct comparison of prioritized interactions and global variables.

As for $\pi(\mathscr{L})$, we directly derive a stochastic simulator for $\pi^{imp}(\mathscr{L})$ from the stochastic semantics. We show that optimizations based on the grouping of channels can be applied too, in order to obtain a simulator with practical performance.

We highlight the usefulness of the imperative π-calculus for the modeling of dynamic cell-structures using the example of water molecules traveling between two compartments through a membrane. This phenomenon was first brought up by Versari and Busi (2009) to show the usefulness of Sπ@. It turned out that the exact behavior can be only obtained if changes in compartment volumes due to the flow of water are considered. These changes are the global aspects considered in the example model here. The developed model confirms the proposed results. It also takes a step further by considering changes in the compartment surface which turn out to have an impact on the results. This is possible due to the higher flexibility of $\pi^{imp}(\mathscr{L})$ compared to Sπ@, since in the latter changes in compartment volumes are hard-wired in the semantics and cannot be extended accordingly.

Furthermore, we provide an example model of a reaction with Michaelis-Menten kinetics in a population-based style, denoting the first successful attempt to represent a Michaelis-Menten reaction in an approach based on the stochastic π-calculus. Small experiments are performed to show that the implementation behaves as expected.

As a further key result we present an encoding of BioAmbients into $\pi^{imp}(\mathscr{L})$. Therefore, we introduce a first version of BioAmbients with a non-deterministic semantics that provides priority levels for interactions. It is shown that BioAmbients with n priority levels can be expressed in $\pi^{imp}(\mathscr{L})$ with n priority levels, i.e. to encode changes in global information, prioritized update protocols are not necessary, since only global variables are used.

A simple type system for the imperative π-calculus is omitted here since there exists no obvious way to obtain one that also allows to type our encoding of BioAmbients. The reason is that the encoding makes use of lists of varying size requiring recursive types, see e.g. Pierce (2002). In any case, due to the imperative store, a simply typed λ-calculus whose evaluation always halts could not be obtained, see e.g. Pierce (2002).

1.4 Related Work

Many languages for the modeling of cell-biological processes have been proposed. To the realm of object-centered languages, Kwiatkowski and Stark (2008) contribute a continuous version of

the π-calculus with semantics in terms of ODE's. sCCP (stochastic concurrent constraint programming) is an approach closely related to $\pi^{imp}(\mathscr{L})$ in that it introduces a global store to which constraints can be added and removed to model the interaction of concurrent processes, see Bortolussi and Policriti (2008b). Processes can have attributes with impact on the constraints. However, a direct communication between processes is not possible. By this sCCP rather subscribes to a population-based style of modeling. The differences between $\pi^{imp}(\mathscr{L})$ and sCCP are discussed in more detail at the hand of the Michaelis-Menten model in Section 4.2.2. Ciocchetta and Guerriero (2009) introduce an extension of Bio-PEPA that allows the consideration of molecule location in compartments, discrete molecule motion, and compartment volumes. However, dynamic cell-structures are explicitly omitted. Besides BioAmbients, Beta-binders (Dematté et al., 2008a), BlenX (Dematté et al., 2008b), and Brane Calculi (Cardelli, 2004) also extend the π-calculus with ad-hoc operators to model dynamic cell structures. Guerriero et al. (2007) propose an extension of Beta-binders explicitly aiming at describing nested but static compartment structures. Cardelli and Gardner (2009) introduce the π-calculus extension 3π, which fully focuses on describing the geometry of processes in 3D space and its evolution over time by matrix operations. Whereas 3π sticks to the realm of discrete motion, John et al. (2008a) propose with SpacePi an approach that exclusively considers continuous motion. Processes move through space and may only interact if they are sufficiently close. Concepts based on continuous motion may help to reduce the amount of experimental data required to build mod-

els, see Chapter 5. The extension Labeled SpacePi (Schäfer and John, 2009) provides more flexibility on describing the shape of processes. Bartocci et al. (2009) introduce the shape calculus, which takes a step further into this direction.

Due to their closeness to the biological realm, approaches based on reaction rules gain increasing interest. Different qualitative approaches have been proposed to model cell-biological processes, e.g. the Pathway Logic (Eker et al., 2002) or standard Petri-nets (Petri, 1962), where the latter has been used e.g. by Simao et al. (2005) to study the tryptophan biosynthesis in Escheria Coli. Extensions of Petri-nets that provide a stochastic semantics in terms of CTMC's or a continuous semantics in terms of ODE's are widely applied, see e.g. Chaouiya (2007); Hardy and Robillard (2004) for an overview. However, studies that explicitly aim at investigating the usefulness of Petri-nets for the spatial modeling of cell-biological processes have not been performed so far. Pedersen and Plotkin (2010) introduce LBS with a semantics in terms of Petri-nets. LBS allows to define locations for reactions in static compartments with fixed volumes. Similarly, Harris et al. (2009) propose an extension for the κ-calculus that allows placing species into static compartments with fixed volumes. BIOCHAM (Calzone et al., 2006) addresses molecule locations and compartment volumes in the very same way. P systems (Paun, 2000) allow defining reaction rules that consider molecule locations, discrete molecule motion, and dynamic cell-structures. Versari (2007b) propose a π@ encoding of a restricted version of P systems that only allows for reactions with at most two reactants but conserves all features regarding space. The calcu-

lus of looping sequences (CLS) by Barbuti et al. (2008) also provides means of modeling molecule location, discrete molecule motion, and dynamic cell-structures. However, similar to Sπ@, it misses a well-defined stochastic semantics in terms of CTMC's but instead makes use of a short-cut to the SSA. Moreover, the modeling of location-based interaction affinities is not considered. With the spatial calculus of looping sequences (sCLS) Barbuti et al. (2009) transfer CLS to the realm of continuous motion. In Bigraphs (Milner, 2009), models of cell-biological processes are defined in terms of two independent graphs, one describing the bonds between proteins and the other nested compartments structures. Reaction rules describe the replacement of parts of the graphs by other graphs. Thus, Bigraphs inherently provides means to model molecule location, discrete molecule motion, and dynamic cell structures. Krivine et al. (2008) introduce a stochastic semantics for Bigraphs in terms of CTMC's. Therefore, in its motivation Bigraphs is strongly related to the work presented here, such that a full comparison of both approaches is desirable. Initial investigations, which were performed during the preparation of this thesis, revealed as a first differences that Bigraphs does not consider numerical values of nodes, such that it appears difficult to express location dependent affinities or compartment volumes changing their values over time. At this point it shall be emphasized, however, that in contrast to e.g. the κ-calculus, LBS, or Bigraphs, in the π-calculus there exists no obvious way to model reactions with more than two reactants in an individual-based style.

Besides ODE's or CTMC's other formalisms could be used to define

the semantics of languages for the modeling of cell-biological processes. Different textbooks exist that report in detail on how to apply cellular automata to describe the spatial dynamics of biological systems, see e.g. Kier (2005); Deutsch and Dormann (2004). Stochastic and delayed differential equations have been applied e.g. by Saarinen et al. (2008) to model nerve signaling and by Wawra et al. (2007) to study the Wnt/β-catenin signaling pathway, respectively. The VCell project[1] makes use of partial differential equations to provide a semantics for a graphical notation that allows describing cell-biological processes continuously distributed in space. Bortolussi and Policriti (2008b) propose applying hybrid automata (Alur et al., 1992) to study the dynamics of cell-biological processes. A semantics in terms of hybrid automata is given to Labeled SpacePi to describe the continuous motion of processes in combination with discrete interaction events and, by Bortolussi and Policriti (2008a), to the π-calculus to capture the evolution of molecule amounts. Fisher et al. (2005) and Kam et al. (2008) use state charts (Harel, 1987) and live sequence charts (Damm and Harel, 2001) to develop models of the vulval development of C. elegans. Degenring et al. (2004) and Maus (2008) apply dynDEVS (Uhrmacher, 2001) and ML-DEVS (Uhrmacher et al., 2007), i.e. different extensions of DEVS (for an introduction see e.g. Zeigler et al., 2000), to model processes at the tryptophan operon and RNA folding, respectively.

Beyond the application domain of cell-biological processes, Abadi

[1] `http://www.nrcam.uchc.edu/`, 06/22/2010

and Fournet (2001), Baldamus et al. (2005), and Johansson et al. (2008) consider extended π-calculi with values that can not only be channels but general data terms, which is similar to the idea of species attributes. However, apart from synchronization on data terms, ways of making reaction affinities dependent on attribute values have not been considered. John et al. (2008a) and Schäfer and John (2009) allow for species attributes to describe process location, size, and motion. Communication constraints are restricted to distances.

Besides polyadic synchronization, various notions of constraints on π-calculus communications have been proposed so far. SPiCO associates sets of functions with channels as a weak form of polyadic synchronization, see also Section 3.3. In Beta-binders for two processes to communicate they need either to be surrounded by a common box or their surrounding boxes need to have interfaces of compatible types. Cappello and Quaglia (2009) propose an encoding of this constraint to polyadic synchronization which, however, does not respect compositionality. Most spatial extensions of the π-calculus constrain communication, since they need to ensure that the location of two processes permits interactions. For example in BioAmbients, an interaction between two processes is only permitted if they are located according to the declared communication direction. In SpacePi, the distance of two possible interaction partners needs to fall below a certain threshold.

1.5 Outline

The thesis is structured in three main chapters. Chapter 2.1 introduces the π-calculus with priority, including its spatial extension BioAmbients with priority (Section 2.2). In Chapter 3 the attributed π-calculus is presented with modeling and expressiveness studies in Sections 3.2 and 3.3, respectively. Chapter 4 regards the imperative π-calculus including modeling and expressiveness studies in Sections 4.2 and 4.3, respectively. For an improved reading, extensive proofs have been moved to Appendix B.

1.6 Bibliographic Note

The work presented in Chapters 2 and 3 is published: a first version of the attributed π-calculus, including a stochastic semantics in terms of CTMC's, is presented in the following conference article.

John, M., Lhoussaine, C., Niehren, J., and Uhrmacher, A. M. (2008). The attributed pi-calculus. In Heiner, M. and Uhrmacher, A. M., editors, *CMSB*, volume 5307 of *Lecture Notes in Computer Science*, pages 83–102. Springer.

A completely revised version of this article denotes the journal article below. It describes the π-calculus with priority and the attributed π-calculus with their syntax, non-deterministic and stochastic semantics, type systems, stochastic simulators, and modeling examples as presented in this thesis. Most parts of the expressiveness studies

are also provided there. At the time of publication, the problems in the encoding of BioAmbients into $\pi@$ were not known. Thus, the extended version of $\pi@$, $\pi[@,\neq]$, and its encoding had to be developed during the preparation of this thesis. BioAmbients with priority has been derived from the π-calculus with priority exclusively for this thesis, too, in order to provide a basis for the expressiveness studies in the imperative π-calculus.

John, M., Lhoussaine, C., Niehren, J., and Uhrmacher, A. M. (2010). The attributed pi-calculus with priorities. *Transactions on Computaional Systems Biology XII. Special Issue on Modeling Methodologies*, 5945:13–76. LNCS (Lecture Notes in Bioinformatics), Springer Berlin/Heidelberg.

Chapter 4 is based on the following conference article, where a first version of the imperative π-calculus, including a non-deterministic and stochastic semantics, a stochastic simulator, the osmosis example, and an encoding from BioAmbients to the imperative π-calculus is presented. However, the work neither considers priority nor provides a full proof of the BioAmbients encoding. Thus, for this thesis a completely revised version of the imperative π-calculus with an improved semantics has been developed. The encoding of BioAmbients is mostly adopted but its proof is entirely new. The same holds true for the encodings to show the conservativeness of the imperative π-calculus.

John, M., Lhoussaine, C., and Niehren, J. (2009). Dynamic compartments in the imperative pi-calculus. In Degano, P. and Gorrieri, R.,

editors, *CMSB*, volume 5688 of *Lecture Notes in Computer Science*, pages 235–250. Springer.

The concept of modeling Michaelis-Menten kinetics has been published in the conference article below, which provides a modeling study on the cell cycle dependency of the Wnt/β-catenin signaling pathway.

Mazemondet, O., John, M., Maus, C., Uhrmacher, A. M., and Rolfs, A. (2009). Integrating diverse reaction types into stochastic models - a signaling pathway case study in the imperative pi-calculus. In Rossetti, M. D., Hill, R. R., Johansson, B., Dunkin, A., and Ingalls, R. G., editors, *Winter Simulation Conference*, pages 932–943. Institute of Electrical and Electronics Engineers, Inc.

The author of this thesis also contributed to some related work. In the following workshop article, which is usually listed as a journal article as it is published in Electronic Notes in Theoretical Computer Science, SpacePi is presented which denotes a first approach to introduce species attributes and communication constraints. Attributes are restricted to process positions, movement functions, and radii and constraints to check process distances. Processes move continuously through space, communication happens only when they are sufficiently close.

John, M., Ewald, R., and Uhrmacher, A. M. (2008). A spatial extension to the pi-calculus. *Electronic Notes in Theoretical Computer Science*, 194(3):133–148.

A more general approach also sticking to the realm of continuously moving processes denotes Labeled SpacePi as presented in the following conference article. It provides a semantics in terms of hybrid automata and allows for more general shapes.

Schäfer, A. and John, M. (2009). Conceptional modeling and analysis of spatio-temporal processes in biomolecular systems. In Kirchberg, M. and Link, S., editors, *APCCM*, volume 96 of *CRPIT*, pages 39–48. Australian Computer Society.

In the conference article below an extension of DEVS, called ML-DEVS, is presented that aims at supporting the multi-level modeling of cell-biological processes. Multi-level modeling approaches enable the modeler to explicitly reflect the abstraction levels and related hierarchies of cell-biological systems, see also Section 5.

Uhrmacher, A. M., Ewald, R., John, M., Maus, C., Jeschke, M., and Biermann, S. (2007). Combining micro and macro-modeling in devs for computational biology. In Henderson, S. G., Biller, B., Hsieh, M.-H., Shortle, J., Tew, J. D., and Barton, R. R., editors, *Winter Simulation Conference*, pages 871–880. Institute of Electrical and Electronics Engineers, Inc.

The following conference article provides a general concept for implementing simulators for the π-calculus and its extensions. It also includes performance studies but exclusively for the stochastic π-calculus.

Leye, S., John, M., and Uhrmacher, A. M. (2010). A flexible architecture for performance experiments with the pi-calculus and its extensions. In Lawson, B., editor, *SIMUTools 2010*. ICST/IEEE.

Beyond the direct scope of this thesis, work is in progress on the visual analysis of the simulation results of models in the attributed π-calculus. π-calculus models represent reaction networks by process interaction. In order to support domain experts with understanding simulation results, the goal is to define a formal transformation from process interaction as in the attributed π-calculus back to reaction networks, and to develop visualization methods to explore the thus obtained dynamic, bipartite graphs. A first concept for a visualization provides the following workshop article.

Schulz, H.-J., John, M., Unger, A., and Schumann, H. (2008). Visual analysis of bipartite biological networks. In Botha, C., Kindlmann, G., Niessen, W., and Preim, B., editors, *Eurographics Workshop on Visual Computing for Biomedicine, Delft, Netherlands*, pages 135–142.

Chapter 2

The π-Calculus with Priority

In this chapter, the π-calculus with priority and its spatial extension BioAmbients with priority is introduced. From the π-calculus with priority, the attributed π-calculus is derived in Chapter 3. In Section 4.3, BioAmbients with priority serves as a basis to study the expressiveness of the imperative π-calculus.

2.1 The π-Calculus with Priority

In the following, first the π-calculus with priority, including a non-deterministic and a stochastic semantics, is introduced. By providing a stochastic semantics, a new version of the stochastic π-calculus is proposed, which, in contrast to Phillips and Cardelli (2007) or the conference Version of the attributed π-calculus, does not impose any syntactic restrictions (such as biochemical forms).

2.1.1 Design Decisions

Both priority and stochastic rates express global properties that concern sets of processes. For selecting a communication step with highest priority, one has to inspect all potential communication steps. Similarly, the probability of a communication step in a stochastic setting depends globally on all possible communication steps. In both cases, the difficulty is therefore to reason globally about all possible communication steps in a given population. Based on this idea, the goal is to find a unified treatment of the π-calculus with priority and the stochastic π-calculus.

The first design decision is to permit process definitions with recursion and parameters in the syntax (rather than replication). Definitions are convenient for modeling cellular processes, and therefore supported by all current simulators for the stochastic π-calculus, see (Regev, 2003; Phillips and Cardelli, 2007; Kuttler et al., 2007). The difficulty in the presence of priority is to discover all potential communication steps in a given process, since some of them may be hidden by definitions. This problem is solved here by exhaustively applying process definitions before selecting any communication step. Fortunately, the resulting operational semantics remains pleasantly simple, and can be generalized properly to the stochastic setting.

The second design choice is to annotate communication prefixes rather than channels by elements in an ordered set $(R, <)$, which may either contain priority levels or stochastic rates. Notice that stochastic rates were annotated to channels in the conference version of the

attributed π-calculus, and that priority was not considered in the non-deterministic version there. The change toward prefix annotations simplifies the semantics considerably (and some proof obligations tremendously). Notice also that priority levels are only annotated to sender prefixes, rather than to sender and receiver prefixes, since otherwise one has to resolve conflicting priority levels as done by Versari (2007a). See Section 3.3.3 for further discussions.

2.1.2 Syntax of Processes

Let $\mathtt{Bool} = \{\mathtt{true}, \mathtt{false}\}$ be the set of Booleans, $\mathbb{N}$ the set of natural numbers starting from 1, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, $\mathbb{R}_+$ the set of non-negative real numbers, and $\mathbb{R}_+^\infty = \mathbb{R}_+ \cup \{\infty\}$.

The π-calculus with priority is based on a partially ordered set of priority levels $(R, <)$, an infinite set *Vars* of channel names $x, y \in \mathit{Vars}$, and an infinite set of process names $A \in \mathit{Proc}$, that have fixed arities in $\mathbb{N}_0$. At any place, a term $A(x_1, \ldots, x_n)$ presumes that n is the arity of A. Tuple notations are broadly used, as for instance $\tilde{x}$ for tuples of channels. If $\tilde{x} = (x_1, \ldots, x_n)$ then the length of the tuple is given by $|\tilde{x}| = n$. Whenever terms $A(\tilde{x})$ occur, it is assumed that the length of $\tilde{x}$ is equal to the arity of A. Substitutions replacing x by y are denoted by $[y/x]$. Substitutions $[\tilde{y}/\tilde{x}]$ apply to tuples of the same length $|\tilde{y}| = |\tilde{x}|$.

The syntax of the π-calculus with priority is defined in Fig. 2.1. In addition to channel names $x \in \mathit{Vars}$ and priority levels $r \in R$ there are four syntactic categories: prefixes π, processes P, sums M, and definitions D. A prefix is either a receiver $x?(\tilde{y})$ or a sender $x{:}r!(\tilde{z})$. All

Prefixes	π	::=	$x?(\tilde{y})$	receiver
		$\mid$	$x{:}r!(\tilde{z})$	sender
Sums	M	::=	$\pi.P$	prefixed process
		$\mid$	$M_1 + M_2$	choice
Processes	P	::=	M	sums
		$\mid$	$A(\tilde{x})$	defined process
		$\mid$	$P_1 \mid P_2$	parallel composition
		$\mid$	$(\nu x)P$	channel creation
		$\mid$	$\mathbf{0}$	idle process
Definitions	D	::=	$A(\tilde{x}) \triangleq P$	parametric process definition

Figure 2.1: Syntax of the π-calculus with channels $x, \tilde{x}, \tilde{y}, \tilde{z} \in \mathit{Vars}$ and priority levels $r \in R$.

$$
\begin{aligned}
fv(\mathbf{0}) &= \emptyset \\
fv(M_1 + M_2) &= fv(M_1) + fv(M_2) \\
fv(P_1 \mid P_2) &= fv(P_1) \cup fv(P_2) \\
fv(x?(\tilde{y}).P) &= \{x\} \cup (fv(P) \setminus \{\tilde{y}\}) \\
fv(x{:}p!(\tilde{z}).P) &= \{x\} \cup fv(\{\tilde{z}\}) \cup fv(P) \\
fv((\nu x)P) &= fv(P) \setminus \{x\} \\
fv(A(\tilde{x})) &= \{\tilde{x}\} \\
fv(A(\tilde{x}) \triangleq P) &= fv(P) \setminus \{\tilde{x}\}
\end{aligned}
$$

Figure 2.2: Free channel names of the π-calculus with priority.

$$
\begin{array}{rcl}
P \mid \mathbf{0} & \equiv & P \\
P_1 \mid P_2 & \equiv & P_2 \mid P_1 \\
M_1 + M_2 & \equiv & M_2 + M_1 \\
(P_1 \mid P_2) \mid P_3 & \equiv & P_1 \mid (P_2 \mid P_3) \\
(M_1 + M_2) + M_3 & \equiv & M_1 + (M_2 + M_3) \\
(\nu x)(\nu y)P & \equiv & (\nu y)(\nu x)P \\
(\nu x)(P \mid Q) & \equiv & (\nu x)P \mid Q, \text{ if } x \notin fv(Q) \\
P \equiv_\alpha Q & \Rightarrow & P \equiv Q
\end{array}
$$

Figure 2.3: Axioms of the structural congruence of the π-calculus with priority.

channel names in $\tilde{y}$ shall be pairwise distinct (since they are distinct formal parameters). A receiver is supposed to receive a tuple of values for $\tilde{y}$ on channel x, and a sender to send a tuple of values $\tilde{z}$ on channel x. The priority r of an interaction is determined by the sender. A term $\pi_1.P_1 + \ldots + \pi_n.P_n$ is a sum of guarded prefixes, that is denoted by $\sum_{i=1}^{n} \pi_i.P_i$ or by $\sum_{i=1}^{n} M_i$, equivalently. A process P may be either a defined process $A(\tilde{x})$, or a parallel composition $P_1 | \ldots | P_n$, which is also denoted by $\prod_{i=1}^{n} P_i$, or a process $(\nu x)P$ creating a new channel x with scope P. If $\tilde{x} = (x_1, \ldots, x_n)$ then $(\nu \tilde{x})P$ abbreviates $(\nu x_1) \ldots (\nu x_n)P$. Note that the syntax provides empty products but not empty sums, i.e. if $n = 0$ then $\prod_{i=1}^{n} P_i = \mathbf{0}$ is the idle process, while $\sum_{i=1}^{n} P_i$ is undefined.

The free channel names $fv(P)$ are defined as usual in Fig. 2.2. The three variable binders are ν-binders $(\nu x)P$, formal parameters $\tilde{y}$ in in-

put prefixes $x?(\tilde{y}).P$, and formal parameters $\tilde{x}$ in definitions $A(\tilde{x}) \triangleq P$. Bound variables are said to be named distinctly in P, if 1) no variable is bound twice in P, 2) no bound variable of P has a free occurrence in P, and 3) no bound variable of P has a free occurrence in some definition. It is generally assumed, that the variables in all processes P are named distinctly, before applying any interaction step to any subprocess of P.

The structural congruence on processes $\equiv$ is the least congruence satisfying the axioms given in Figure 2.3, i.e. consistent renaming of bound variables, associativity and commutativity of parallel composition and summation, the rule of the neutral element of $\mathbf{0}$ with respect to parallel composition, and scope intrusion and extrusion for ν-binders. The *prenex normal form* of processes is defined as follows:

Definition 1. A process P is said to be in *prenex normal form*, if $P = (\nu\tilde{x})(\prod_{i=1}^{n} M_i \mid \prod_{i=1}^{m} A_i(\tilde{x}))$ and all bound names in P are named distinctly.

Notice that every process P is congruent to some process in prenex form.

For illustration, consider silent actions $\texttt{delay}{:}r.P$. A silent action becomes active with priority r without any communication partner and then behaves like P. In the syntax of the π-calculus with priority, silent actions can be expressed by processes $(\nu\texttt{delay})(\texttt{delay}?().P \mid \texttt{delay}{:}r!().\mathbf{0})$ where a dummy interaction partner sends the empty tuple on local channel `delay` with priority r and then disappears.

2.1.3 Non-deterministic Operational Semantics

The operational semantics of the π-calculus with priority is presented in Figure 2.4. It is defined by a reduction relation $\rightarrow$ that is based on three kinds of binary relations, reductions $\xrightarrow[nd]{r}$ with priority $r \in R$, reductions $\xrightarrow[nd]{err}$ leading to errors, and reductions $\xrightarrow[nd]{app}$ applying process definitions. The label *nd* distinguishes non-deterministic from stochastic reduction steps, *err* stands for error and *app* for application.

A communication step (COM) applies to two parallel sums with matching prefixes, a sum with a receiver $x?(\tilde{y}).P_1 + M_1$ and another with a sender $x!(\tilde{z}).P_2 + M_2$ for the same channel x and with the same number of arguments $|\tilde{y}| = |\tilde{z}|$. The sender hands over its arguments $\tilde{z}$ to the receiver and continues with P_2, while the receiver replaces its formal parameters $\tilde{y}$ by $\tilde{z}$ and continues with $P_1[\tilde{z}/\tilde{y}]$. All alternative choices in M_1 and M_2 are discarded. The communication step may be performed with priority r contributed by the sender. A communication error (E.COM) is raised, if two matching prefixes on the same channel x offer different arities $|\tilde{y}| \neq |\tilde{z}|$. Here, $\perp$ denotes an arbitrary erroneous expression. A single application step (APP) replaces a defined process by its definition. It is assumed that there exists a unique definition for all defined processes. Communication and error steps are closed under structural congruence (STRUC), and permitted under parallel composition (PAR) and new binders (NEW). Rule (PRIOR) states that only communication steps with highest available priority may be selected by final reduction relation $\rightarrow$. The set of all communication prefixes becomes apparent only after having applied definitions ex-

Communication and application steps

$$(\text{COM})\ \frac{|\tilde{y}| = |\tilde{z}|}{x?(\tilde{y}).P_1 + M_1 \mid x{:}r!(\tilde{z}).P_2 + M_2 \xrightarrow[nd]{r} P_1[\tilde{z}/\tilde{y}] \mid P_2}$$

$$(\text{APP})\ \frac{A(\tilde{x}) \triangleq P}{A(\tilde{y}) \xrightarrow[nd]{app} P[\tilde{y}/\tilde{x}]}$$

Program errors

$$(\text{E.COM})\ \frac{|\tilde{y}| \neq |\tilde{z}|}{x?(\tilde{y}).P_1 + M_1 \mid x{:}r!(\tilde{z}).P_2 + M_2 \xrightarrow[nd]{err} \perp}$$

Structural rules where $\beta \in \{err, app\} \cup R$

$$(\text{PAR})\ \frac{P_1 \xrightarrow[nd]{\beta} P_1'}{P_1 \mid P_2 \xrightarrow[nd]{\beta} P_1' \mid P_2} \qquad (\text{NEW})\ \frac{P \xrightarrow[nd]{\beta} P'}{(\nu x)P \xrightarrow[nd]{\beta} (\nu x)P'}$$

$$(\text{STRUC})\ \frac{P \equiv P_1 \quad P_1 \xrightarrow[nd]{\beta} P_2 \quad P_2 \equiv P'}{P \xrightarrow[nd]{\beta} P'}$$

continued...

Figure 2.4: Rules of the non-deterministic semantics of π-calculus with priority levels in $(R, <)$.

Error-free convergence of application

$$(\text{CONV})\ \frac{P \xrightarrow[nd]{app}{}^{*} P' \quad P' = (\nu \tilde{x}) \prod_{i=1}^{n} M_i \quad \neg P' \xrightarrow[nd]{err} \bot}{P \Downarrow P'}$$

Reduction ($r \in R$)

$$(\text{PRIOR})\ \frac{P \Downarrow P' \quad P' \xrightarrow[nd]{r} Q \quad \neg \exists r_1 \in R. \exists Q_1.\ r < r_1 \wedge P' \xrightarrow[nd]{r_1} Q_1}{P \to Q}$$

Figure 2.4: Rules of the non-deterministic semantics of π-calculus with priority levels in $(R, <)$(continued).

haustively (CONV). Application may not terminate such as for $A()$ if defined by $A() \triangleq A()$. Such nonterminating definitions block all potential subsequent communication steps. Similarly, communication errors $P \xrightarrow[nd]{err} \bot$ block all communication steps on P.

Example 1. Consider the example of forwarders $Fwd(x,y)$ which receive some value on channel x and forward it to channel y. Forwarders can be used to let objects flow along lists, such as RNAP polymerases along DNA sequences. Two levels of priorities $low < high$ are assumed, where highest priority is given to forwarding actions.

$$Fwd(x,y) \triangleq x?(z).(y{:}high!(z).\mathbf{0} \mid Fwd(x,y))$$

First, forwarders are used in order to define a list with two elements,

which an object z traverses.

$$List_2() \triangleq x_1{:}low!(z).\mathbf{0} \mid Fwd(x_1,x_2) \mid Fwd(x_2,x_3)$$

Process $List_2()$ can be reduced as follows:

$$\begin{aligned} List_2() &\rightarrow Fwd(x_1,x_2) \mid x_2{:}high!(z).\mathbf{0} \mid Fwd(x_2,x_3) \\ &\rightarrow Fwd(x_1,x_2) \mid Fwd(x_2,x_3) \mid x_3{:}high!(z).\mathbf{0} \end{aligned}$$

Beside lists, rings or other cyclic data structures can be constructed from forwarders:

$$\begin{aligned} Ring_3() \triangleq\ & x_1{:}low!(z).\mathbf{0} \mid x_2{:}low!(z).\mathbf{0} \mid Fwd(x_1,x_2) \mid Fwd(x_2,x_3) \mid \\ & Fwd(x_3,x_1) \end{aligned}$$

One of the two z objects is turning around in the ring forever, while the other can never enter the ring, since entering actions are given lower priority.

$$\begin{aligned} Ring_3() &\rightarrow x_1{:}low!(z).\mathbf{0} \mid Fwd(x_1,x_2) \mid Fwd(x_2,x_3) \mid \\ &\quad\ x_3{:}high!(z).\mathbf{0} \mid Fwd(x_3,x_1) \\ &\rightarrow x_1{:}low!(z).\mathbf{0} \mid x_1{:}high!(z).\mathbf{0} \mid Fwd(x_1,x_2) \mid \\ &\quad\ Fwd(x_2,x_3) \mid Fwd(x_3,x_1) \\ &\rightarrow x_1{:}low!(z).\mathbf{0} \mid Fwd(x_1,x_2) \mid x_2{:}high!(z).\mathbf{0} \mid \\ &\quad\ Fwd(x_2,x_3) \mid Fwd(x_3,x_1) \\ &\rightarrow \ldots \end{aligned}$$

2.1.4 Uniqueness of Convergence

To ensure computational feasibility of the π-calculus with priorities, it needs to be shown that processes converge to a unique result, i.e. for

all P there exists at most one P', such that $P \Downarrow P'$. This is fulfilled if the reduction relation $\xrightarrow[nd]{app}$, as it is used in process convergence, is confluent. Confluence expresses the insignificance of the order of steps of reduction relations. Below, the notion of uniform confluence is defined, which implies strong confluence and thus confluence (Huet, 1980).

Definition 2 (Reformulation of uniform confluence of calculi by Niehren (2000))**.** A rewrite relation σ is confluent modulo structural congruence provided that if $(P,\rho)\sigma(P_1,\rho_1)$ and $(P,\rho)\sigma(P_2,\rho_2)$ then $(P_1,\rho_1) \equiv (P_2,\rho_2)$ or there exists (P',ρ'), such that $(P_1,\rho_1)\sigma(P',\rho')$ and $(P_1,\rho_1)\sigma(P',\rho')$.

The reduction relation $\xrightarrow[nd]{app}$ is uniform confluent and, under the assumption that no cyclic definitions exist, terminates, which allows for the conclusion that in the π-calculus with priorities convergence yields unique results.

Lemma 1. *The rewrite relation $\xrightarrow[nd]{app}$ is uniform confluent modulo structural congruence, irreducible processes are congruent to processes $(\nu\tilde{x})\prod_{i=1}^{n} M_i$.*

Proof. The lemma relies on the assumption that there exists a unique definition for every defined process, and that the order of application of these definitions does not matter. In the following, it is first shown that application terminates on equivalence classes of processes of the form $[(\nu\tilde{x})\prod_{i=1}^{n} M_i]_{\equiv}$ and then that for a single process it always leads to the same result.

Claim. Let $P = (\nu\tilde{x})\prod_{i=1}^{n} P_i$ be a prenex normal form in which all

bound variables are named distinctly, and such that all P_i are sums or defined processes. In this case, $P \xrightarrow[nd]{app} P'$ if and only if the following rule applies:

$$\frac{1 \leq j \leq n \qquad P_j = A_j(\tilde{z}_j) \quad A_j(\tilde{y}_j) \triangleq Q_j \quad P' \equiv (\nu \tilde{x})(\prod_{i=1, i \neq j}^{n} P_i \mid Q_j[\tilde{z}_j/\tilde{y}_j])}{P \xrightarrow[nd]{app} P'}$$

Application defines a relation on equivalence classes of processes modulo structural congruence, such that $[P]_{\equiv} \xrightarrow[nd]{app} [P']_{\equiv}$ if $P \xrightarrow[nd]{app} P'$. The above claim shows that application terminates on equivalence classes of processes of the form $[(\nu \tilde{x}) \prod_{i=1}^{n} M_i]_{\equiv}$, since it is assumed that there exists at least one definition for every defined process.

To see uniform confluence, it is assumed that $P \xrightarrow[nd]{app} P'_1$ and $P \xrightarrow[nd]{app} P'_2$, with j_1 and j_2 being the positions of the respective reduction step (according to the above rule). If $j_1 = j_2$ then $P'_1 \equiv P'_2$, since it is assumed that there exists at most one definition for every defined process. Otherwise, if $j_1 \neq j_2$ then P'' can be set to $(\nu \tilde{x})(\prod_{i=1, i \notin \{j_1, j_2\}}^{n} P_i \mid Q_{j_1}[\tilde{z}_{j_1}/\tilde{y}_{j_1}] \mid Q_{j_2}[\tilde{z}_{j_2}/\tilde{y}_{j_2}])$. □

Proposition 1 (Convergence uniqueness of the π-calculus with priorities)**.** *For all processes P there exists at most one class $[P']_{\equiv}$, such that $P \Downarrow P'$.*

Proof. This follows immediately from the confluence result in Lemma 1. □

There exist processes P that do not converge to any P' since the application of process definitions does not terminate. The semantics of the π-calculus with priorities ensures that such processes cannot be reduced any further, even though they might not contain an immediate error $P \xrightarrow[nd]{err} \bot$. For instance, consider the process $A()$ with the following definition $A() \triangleq A()$ that is not well-founded. An implementation of the π-calculus with priorities may either run into an infinite loop unfolding the definition of A repeatedly, or report the erroneous cycle. Thus, application may yield three possible kinds of results: convergence, arity mismatches, and non-termination of application.

Remark 1. *If* $P \equiv (\nu\tilde{x}) \prod_{i=1}^{n} M_i$ *and* $\neg P \xrightarrow[nd]{err} \bot$ *then* $P \equiv P' \Leftrightarrow P \Downarrow P'$.

Proof. Suppose that $P \equiv (\nu\tilde{x}) \prod_{i=1}^{n} M_i$ and $\neg P \xrightarrow[nd]{err} \bot$. If $P \equiv P'$ then $P \xrightarrow[nd]{app}{}^{*} P'$ by definition of reflexivity, such that $P \Downarrow P'$. Conversely, suppose that $P \Downarrow P'$. By definition of convergence, this implies $P \xrightarrow[nd]{app}{}^{*} P'$, which yields $P \equiv P'$ since $[P]_{\equiv}$ is irreducible with respect to $\xrightarrow[nd]{app}$ by Lemma 1. □

2.1.5 Stochastic Operational Semantics

In this section, a stochastic operational semantics for the π-calculus with priorities is presented, under the assumption that stochastic rates in $\mathbb{R}_{+}^{\infty}$ are used as priorities with two levels, the lower level for numbers in $\mathbb{R}_{+}$ and the higher for ∞.

In contrast to most previous approaches, the syntax of processes

Labeled communication steps ($r \in \mathbb{R}_+^\infty$ and $\ell \in \mathbb{N}^4$)

$$(\text{COM}_\ell)\ \frac{\ell = (i_1, j_1, i_2, j_2) \quad i_1 \neq i_2 \quad \pi_{i_1}^{j_1} = x?(\tilde{y}) \quad \pi_{i_2}^{j_2} = x{:}r!(\tilde{z}) \quad |\tilde{y}| = |\tilde{z}|}{(\nu\tilde{x})\prod_{i=1}^{n}\sum_{j=1}^{m_i}\pi_i^j.P_i^j \xrightarrow[\ell]{r} (\nu\tilde{x})(\prod_{i=1,i\neq i_1,i_2}^{n}\sum_{j=1}^{m_i}\pi_i^j.P_i^j \mid P_{i_1}^{j_1}[\tilde{v}/\tilde{y}] \mid P_{i_2}^{j_2})}$$

Markov chain ($r, r' \in \mathbb{R}^+$)

$$(\text{SUM})\ \frac{P \Downarrow P_1 \quad \sum_{\{(r',\ell)\mid P_1 \xrightarrow[\ell]{r'} P_2 \equiv P'\}} r' = r \neq 0 \quad \neg\exists\ell\exists P''.P_1 \xrightarrow[\ell]{\infty} P''}{P \xrightarrow{r} P'}$$

$$(\text{COUNT})\ \frac{P \Downarrow P_1 \quad n = \sharp\{\ell \mid P_1 \xrightarrow[\ell]{\infty} P_2 \equiv P'\} \neq 0}{P \xrightarrow{\infty(n)} P'}$$

Figure 2.5: Rules of the stochastic semantics of the π-calculus with priorities. Except rules (COM) and (PRIOR) all rules of the non-deterministic semantics are inherited.

remains without change. This means in particular, that stochastic rates are annotated to output prefixes rather than to channel names as done by Regev (2003); Phillips and Cardelli (2007); Kuttler et al. (2007), or, like Priami (1995), to both input and output prefixes.

The stochastic semantics of a process P in the stochastic π-calculus is a continuous time Markov chain (CTMC). The states of a CTMC are classes of processes $[P]_\equiv$. A priori, the state space may be infinite,

even though only finitely many states may be reachable in many cases. State transitions $P \xrightarrow{r} P'$ are labeled by propensities $r \in \mathbb{R}_+^\infty$. If r is finite then the probability of a reduction step from P to P' is r/s, where s is the sum of all propensities r' of transitions starting in P. If s is infinite then the transition is impossible, since a transition exists with infinite rate which has priority.

The probability of a reduction step follows the *Chemical Law of Mass Action*. Given a source process P and a target process P', the rate of $P \to P'$ depends on the number of ways in which P may reduce to P'. For instance, consider $P_1 = x?().\mathbf{0}$ and $P_2 = x{:}r!().\mathbf{0}$ for some rate $r \in \mathbb{R}_+$. By fixing $P = P_1 \mid P_1 \mid P_2$ and $P' = P_1$, two possible interactions of rate r are obtained, yielding transition $P \xrightarrow{2r} P'$ where $2r$ is the reaction rate.

In order to discriminate interactions leading to the same state, rule (COM_ℓ) in Figure 2.5 defines communication steps labeled by positions $\ell \in \mathbb{N}^4$, where the interaction occurs. Given a prenex normal form $P = (\nu\tilde{x})\prod_{i=1}^{n}\sum \pi_i^j.P_i^j$, a tuple $\ell = (i_i, j_1, i_2, j_2)$ defines the pair of communication prefixes $\pi_{i_1}^{j_1}.P_{i_1}^{j_1}$ and $\pi_{i_2}^{j_2}.P_{i_2}^{j_2}$. As before, a communication step can only be applied to senders and receivers on the same channel. Notation $P \xrightarrow[\ell]{r} P'$ denotes that there exists a potential interaction at position ℓ, where r is the rate annotated to the sender.

The definition of the rules to obtain the CTMC transitions of a process are based on the non-deterministic semantics of the π-calculus with priorities, replacing rule (PRIOR) by rules (SUM) and (COUNT) and keeping all others rules. Notice, however, that structural rules do not apply to communication steps anymore, since they are labeled by

nd. With this, the order of processes in a parallel composition and of summands in a summation is fixed, which is necessary to accumulate a process' possible interactions based on the labels ℓ provided by rule (COM_ℓ).

Transitions $[P]_\equiv \xrightarrow[nd]{r} [P']_\equiv$ with finite propensities $r \in \mathbb{R}_+$ are obtained by rule (SUM). First, convergence of P with respect to application is tested. If this test fails then no transition is possible. Otherwise, the unique equivalence class $[P_1]_\equiv$ is computed such that $P \Downarrow P_1$. Second, an arbitrary representative in prenex normal form P_1 of this congruence class is fixed. Third, all pairs (r', ℓ) of P are computed such that there exists $P_2 \equiv P'$ and a communication step $P_1 \xrightarrow[\ell]{r'} P_2$. Finally, all such rates r' are summed up into propensity r. Going back to the previous example, the two communications $P_1 \mid P_1 \mid P_2 \xrightarrow[(1,1,3,1)]{r} P_1$ and $P_1 \mid P_1 \mid P_2 \xrightarrow[(2,1,3,1)]{r} P_1$ are obtained, such that $P_1 \mid P_1 \mid P_2 \xrightarrow{2r} P_1$ as expected.

Communication steps with infinite propensities are treated by rule (COUNT). These are given highest priority as stated already in rule (SUM). The probability of a reduction $P \xrightarrow{\infty(n)} P'$ is n/m where n is the number of interactions with rate ∞ leading from P to a process congruent to P', and m the overall number of interactions with rate ∞ starting from P. Given these probabilities, and provided that no infinite sequence of immediate transitions is reachable, one can build a reduction, without immediate transitions, that defines a proper CTMC and preserves the *probabilities of transitions* and *sojourn times* (see e.g. Kuttler et al. (2007) for details).

Chemical reactions:

$$x : A,B \xrightarrow{0.5} A,C$$
$$y : A,C \xrightarrow{5} A,B$$

π-calculus definitions:

$$A \triangleq x{:}0.5!().A + y{:}5!().A$$
$$B \triangleq x?().C$$
$$C \triangleq y?().B$$

CTMC with states reachable from $A^2 \mid B^2 \mid C^1$:

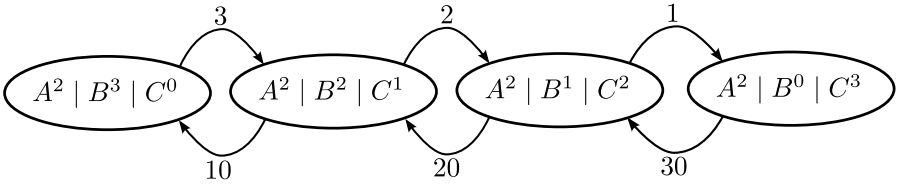


Figure 2.6: Example of a CTMC generated by the stochastic semantics of the π-calculus with priorities.

For illustration, consider a system of two chemical reactions, $x : A,B \xrightarrow{0.5} A,C$ and the inverse $y : A,C \xrightarrow{5} A,B$ whose rate is 10-fold higher. In Figure 2.6, molecules of species A,B,C are defined as processes in the stochastic π-calculus that act according to these chemical reactions and the CTMC of this chemical system is shown in Figure 2.6.

The stochastic semantics of the π-calculus with priorities does indeed properly refine the non-deterministic operational semantics.

Proposition 2. *If the set of priorities $(R,<)$ is equal to $(\mathbb{R}_+^\infty,<_2)$, where $<_2$ defines the usual two levels of priorities (i.e. $r <_2 \infty$ for all $r \in \mathbb{R}_+$), then for all processes P,Q:*

$$P \to Q \text{ iff } (\exists r \in \mathbb{R}_+ : P \xrightarrow{r} Q \vee \exists n \in \mathbb{N} :\ P \xrightarrow{\infty(n)} Q)$$

Proof. The implication from the right to the left is quite obvious, since $P \xrightarrow[\ell]{r} Q$ implies $P \to Q$. The proof of the direction from the left to the right is based on the following claim that relates communication steps to labeled communication steps in this direction:

Claim. If $P_1 \xrightarrow[nd]{r} Q$ and $P_1 = (\nu x)\prod_{j=1}^{n}\sum_{i=1}^{m_j}\pi_i^j.P_i^j$ then there exists a label $\ell = (i_1,j_1,i_2,j_2)$ and a process Q' such that $Q' \equiv Q$ and $P_1 \xrightarrow[\ell]{r} Q'$.

This follows from a standard analysis of the structural congruence. Suppose now, that $P \to Q$ holds. In this case, the following rule must be applicable:

$$(\text{PRIOR})\ \frac{P \Downarrow P_1 \quad P_1 \xrightarrow[nd]{r} Q \quad \neg\exists r_1 \in R.\exists Q_1.\ r < r_1 \wedge P_1 \xrightarrow[nd]{r_1} Q_1}{P \to Q}$$

Without loss of generality, it can be assumed that P_1 is in prenex normal form, since relation $\xrightarrow[nd]{r}$ is closed under structural congruence by rule (STRUC). The second hypothesis and the above claim show that $P_1 \xrightarrow[\ell]{r} Q'$ for some process Q', with $Q' \equiv Q$. The third hypothesis holds if and only if either $r = \infty$ or else $r \in \mathbb{R}_+$ and $\neg\exists Q_1.\ P_1 \xrightarrow[nd]{\infty} Q_1$:

- In the case $r = \infty$, a transition with infinite propensity can be created:

$$(\text{COUNT})\ \frac{P \Downarrow P_1 \quad n = \sharp\{\ell \mid P_1 \xrightarrow[\ell]{\infty} Q' \equiv Q\} \neq 0}{P \xrightarrow{\infty(n)} Q}$$

- In the case $r \in \mathbb{R}_+$, property $P_1 \xrightarrow[\ell]{r} Q'$ shows that $\sum_{\{(r',\ell)|P_1 \xrightarrow[\ell]{r'} Q' \equiv Q\}} r' \neq 0$. Thus a transition of the Markov chain with finite propensity can be created:

$$(\text{SUM})\ \frac{P \Downarrow P_1 \quad \sum_{\{\ell|P_1 \xrightarrow[\ell]{r'} Q' \equiv Q\}} r' = r \neq 0 \quad \neg\exists \ell \exists Q_1 . P_1 \xrightarrow[\ell]{\infty} Q_1}{P \xrightarrow{r} Q}$$

□

2.1.6 Type System

In this section, a type system for the π-calculus with priority is presented, which prevents arity mismatches in communication attempts as defined by rule (E.COM). Non-immediate errors, like nonterminating applications of unguarded process definitions are not captured, though. These can be detected by a simple syntactic cycle check.

Channels are the only values in the π-calculus. In order to exclude arity mismatches on communication, channel types fix the types of all arguments that can be communicated:

$$\text{types} \qquad \tau ::= ch(\tilde{\tau})$$

A channel of type $ch(\tilde{\tau})$ may only be used to receive and send values of types $\tilde{\tau}$.

Example 2. Consider the process $P = x{:}r!(z).z?(y).P_1 \mid x?(y).y{:}r!().P_2$. The arities of the sender and receiver for x coincide in that they

both have one argument. After communication, however, P becomes $z?(y).P_1 \mid z{:}r!().P_2$, which has an arity mismatch on z. These kinds of situations should be excluded for well-typed processes. Indeed, the first subprocess of P is well-typed, considering the following typing $x{:}ch(ch(\tau)), z{:}ch(\tau)$ for some type τ. The second subprocess of P, requires a typing $x{:}ch(ch())$. Both conditions together are unsatisfiable.

In order to capture types, the syntax of the π-calculus is slightly extended:

$$\begin{array}{rrcl} \text{typed processes} & P & ::= & (\nu x){:}\tau P \mid \ldots \\ \text{typed process definition} & D & ::= & A(\tilde{x} : \tilde{\tau}) \triangleq P, \text{with } |\tilde{x}| = |\tilde{\tau}| \end{array}$$

The rules of the type system are given in Figure 2.7. They make use of type environments Γ, which map variables x and process definitions A to types. Environment $\Gamma, x{:}\tau$ denotes the environment with the same mappings as Γ, except x, which it maps to τ. In the same way $\Gamma, A{:}\tilde{\tau}$ denotes Γ changed by mapping $A{:}\tilde{\tau}$. Notation $\Gamma \vdash P$ states that process P is well-typed in typing environment Γ. Rules (T.SEND) and (T.REC) check whether channels are used for communication in a correct way. For processes $\nu x : \tau P$, rule (T.NEW) ensures that P is well-typed in the typing environment extended by the type τ of the new channel x. By rules (T.PAR) and (T.SUM), the components of parallel compositions and summations are inspected separately. Process $\mathbf{0}$ is always well-typed. Rule (T.APP) checks whether the types of the arguments of an application fit the type of the corresponding definition. By rule (T.DEF), a process definition $A(\tilde{x} : \tilde{\tau}) \triangleq P$ is well-typed if P is well-typed in the type environment extended by the types $\tilde{\tau}$ of parameters

$$(\text{T.REC})\ \frac{\Gamma \vdash x{:}ch(\tilde{\tau}) \quad \Gamma, \tilde{y}{:}\tilde{\tau} \vdash P}{\Gamma \vdash x?(\tilde{y}).P} \qquad (\text{T.NIL})\ \frac{}{\Gamma \vdash \mathbf{0}}$$

$$(\text{T.SEND})\ \frac{\Gamma \vdash x{:}ch(\tilde{\tau}) \quad \Gamma \vdash \tilde{z}{:}\tilde{\tau} \quad \Gamma \vdash P}{\Gamma \vdash x{:}r!(\tilde{z}).P}$$

$$(\text{T.SUM})\ \frac{\Gamma \vdash M_1 \quad \Gamma \vdash M_2}{\Gamma \vdash M_1 + M_2} \qquad (\text{T.PAR})\ \frac{\Gamma \vdash P_1 \quad \Gamma \vdash P_2}{\Gamma \vdash P_1 \mid P_2}$$

$$(\text{T.APP})\ \frac{\Gamma \vdash A{:}\tilde{\tau} \quad \Gamma \vdash \tilde{y}{:}\tilde{\tau}}{\Gamma \vdash A(\tilde{y})} \qquad (\text{T.NEW})\ \frac{\Gamma, x{:}\tau \vdash P}{\Gamma \vdash (\nu x{:}\tau)P}$$

$$(\text{T.DEF})\ \frac{\Gamma \vdash A : \tilde{\tau} \quad \Gamma, \tilde{x} : \tilde{\tau} \vdash P}{\Gamma \vdash A(\tilde{x} : \tilde{\tau}) \triangleq P} \qquad (\text{T.DEFS})\ \frac{\forall D \in \mathscr{D}.\Gamma \vdash D}{\Gamma \vdash \mathscr{D}}$$

Figure 2.7: Type system for π-calculus with priority.

$\tilde{x}$. Rule (T.DEFS) states, that a set of process definitions is well-typed if all its elements are.

Example 3. Consider again processes $\mathsf{Fwd}(x,y)$, $\mathsf{List}_2()$, and $\mathsf{Ring}_3()$ from example 1:

$$\mathsf{Fwd}(x,y) \triangleq x?(z).(y:\mathtt{high}!(z).\mathbf{0} \mid \mathsf{Fwd}(x,y))$$
$$\mathsf{List}_2() \triangleq x_1:\mathtt{low}!(z).\mathbf{0} \mid \mathsf{Fwd}(x_1,x_2) \mid \mathsf{Fwd}(x_2,x_3)$$
$$\mathsf{Ring}_3() \triangleq x_1:\mathtt{low}!(z).\mathbf{0} \mid x_2:\mathtt{low}!(z).\mathbf{0} \mid \mathsf{Fwd}(x_1,x_2) \mid \mathsf{Fwd}(x_2,x_3) \mid \mathsf{Fwd}(x_3,x_1)$$

Processes $List_2$ and $Ring_3$ are well-typed in environments, where channel z is given an arbitrary type, say $\tau = ch()$, while process Fwd must be assigned type $(ch(\tau),ch(\tau))$. Furthermore the three channels x_1,x_2,x_3 that connect the forwarders must be of type $ch(\tau)$, too. Valid type environments Γ for $Ring_3$ thus must contain the following assumptions:

$$x_1{:}ch(\tau),\ x_2{:}ch(\tau),\ x_3{:}ch(\tau),\ z{:}\tau,\ Fwd{:}(ch(\tau),ch(\tau)),\ List_2{:}(),\ Ring_3{:}()$$

Proposition 3 (Type safety)**.** *Let P be a process in the π-calculus with priorities and $\mathscr{D}$ its set of definitions. If $\Gamma \vdash P$, $\Gamma \vdash \mathscr{D}$, and $P \to Q$ then $\Gamma \vdash Q$.*

The proof works as usual. See the proof of Theorem 1 for a more general instance.

Corollary 1 (Error freeness)**.** *Let P be a process in the π-calculus with priorities and $\mathscr{D}$ its set of definitions. If $\Gamma \vdash P$, $\Gamma \vdash \mathscr{D}$, and $P \rightarrow^* Q$ then $\neg Q \xrightarrow[nd]{err} \bot$.*

Proof. Assuming $\Gamma \vdash P$ and $P \rightarrow^n Q$, the proof is by induction on n. The inductive step follows from Proposition 3. It thus remains to prove the initial case that is $Q \equiv P$. Assume by contradiction that there exists some process P_0 such that $\Gamma \vdash P_0$ and $P_0 \xrightarrow[nd]{err} \bot$. As for the proof of Lemma 1, a standard analysis of the structural congruence shows the following claim: let $P_0 \equiv (\nu \tilde{x}{:}\tilde{\tau}) \prod_{i=1}^{n} P_i$ be a prenex normal form in which all bound variables are named distinctly, and such that all P_i are sums or defined processes, and $\exists j,k.1 \leq j < k \leq n$, $P_j = x_0?(\tilde{y}).Q_1 + M_1$, $P_k = x_0{:}r!(\tilde{z}).Q_2 + M_2$, and $|\tilde{y}| \neq |\tilde{z}|$. From $\Gamma \vdash P_0$ follows that $\Gamma \vdash (\nu \tilde{x}{:}\tilde{\tau}) \prod_{i=1}^{n} P_i$. This statement results from a series of applications of rules (T.NEW) and (T.PAR) and from statements $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash P_i$ for all $i \in \{1,\ldots,n\}$. In particular, it is true that $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash x_0?(\tilde{y}).Q_1 + M_1$ and $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash x_0{:}r!(\tilde{z}).Q_2 + M_2$. Therefore, by rules (T.REC) and (T.SEND), it holds that $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash x_0{:}ch(\tilde{\tau})$, $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash \tilde{z}{:}\tilde{\tau}$, and $\Gamma, \tilde{x}{:}\tilde{\tau}, \tilde{y}{:}\tilde{\tau} \vdash Q_1$. Thus, it is true that $|\tilde{z}| = |\tilde{\tau}|$ and $|\tilde{y}| = |\tilde{\tau}|$ which contradicts $|\tilde{y}| \neq |\tilde{z}|$. □

2.2 BioAmbients with Priority

BioAmbients allows locating π-calculus processes in *ambients*, e.g. to denote compartments. Ambients can be nested in other ambients which allows building up spatial, hierarchical structures. Additionally, dynamics in ambient structures can be modeled in the sense that

ambients may fuse or enter and leave other ambients. Such operations denote changes in global information, since they can broadly influence the interaction possibilities of entire sets of processes at once. Among process algebras, BioAmbients is also established as one of the main formalisms for the spatial modeling of cell-biological systems. Thus, it forms a proper basis to study the expressiveness of the imperative π-calculus, see Section 4.3.2.

In this section, a version of BioAmbients is presented, which is obtained by extending the π-calculus with priority as it is defined above. It conserves all the features for the spatial modeling of the original language, as presented by Regev et al. (2004). However, as a major difference the version presented here provides priority.

In the following, first, in Section 2.2.1, the syntax of BioAmbients is introduced, followed by the non-deterministic semantics of BioAmbients in Section 2.2.2. A stochastic semantics is omitted, as such seems to be hard to obtain. In fact, Brodo et al. (2007) present a stochastic semantics for BioAmbients, which, however, is neither defined in terms of CTMC's (a direct mapping to the SSA is provided), nor considers priority.

2.2.1 Syntax of Processes

The syntax of BioAmbients is presented in Figure 2.8. The π-calculus is extended in two ways: first, processes can be engulfed by ambients denoting locations, e.g. compartments. Second, prefixes are enriched by a context, which can be either a communication direction or a re-

Communication direction	d	::=	`local`	local
		|	`p2c`	parent to child
		|	`c2p`	child to parent
		|	`s2s`	sibling to sibling
Rearrangement capability	c	::=	`enter`	enter ambient
		|	`exit`	exit ambient
		|	`merge`	merge ambients
Prefixes	π	::=	$d\ x?(\tilde{y})$	receiver
		|	$d\ x{:}r!(\tilde{x})$	sender
		|	$c\ x?$	rearrangement accepter
		|	$c\ x{:}r!$	rearrangement initializer
Sums	M	::=	$\pi.P$	prefixed process
		|	$M_1 + M_2$	choice
Processes	P	::=	$[P]$	ambient
		|	M	sums
		|	$A(\tilde{x})$	defined process
		|	$P_1 \mid P_2$	parallel composition
		|	$(\nu x)P$	channel creation
		|	$\mathbf{0}$	idle process
Definitions	D	::=	$A(\tilde{x}) \triangleq P$	parametric process definition

Figure 2.8: Syntax of BioAmbients with channels $x, \tilde{x}, \tilde{y} \in \mathit{Vars}$ and priorities $r \in R$.

arrangement capability. Communication directions describe the ways in which processes in ambient structures can communicate. They are depicted in Figure 2.11, columns (2) and (3), where boxes represent ambients, which are labeled with names a_1, a_2, p for illustration. The context `local` allows for the communication of two processes in the same ambient. A process in a parent ambient can send to a process in a child ambient in the `p2c` context. The symmetric case is covered by the context `c2p`. Processes in sibling ambients, i.e. in ambients which are contained by the same ambient, can interact in the `s2s` context. Rearrangement capabilities allow for dynamic changes in the ambient structure. They are depicted in Figure 2.11, column (1). By `merge` two sibling ambients merge, such that the processes in the siblings end up in the same ambient and can perform local communication. On `enter` the ambient of the sender enters the ambient of the receiver, such that the former becomes the child ambient of the latter which allows them to perform `c2p` communication. The context `exit` denotes the symmetric case, which results in two sibling ambients.

The set of free names basically remains as before, only considering ambients in addition, whose free names are defined by their content. The context of communication and rearrangement prefixes is ignored, see Figure 2.9.

The rules of the structural congruence of BioAmbients are presented in Figure 2.10. Compared to the π-calculus only two additional rules are needed, one stating that an empty ambient is congruent to the idle process $\mathbf{0}$ and the other one that ν-operators can be freely moved from the outside to the inside of ambients and vice versa.

$$
\begin{aligned}
fv(\mathbf{0}) &= \emptyset \\
fv([P]) &= fv(P) \\
fv(M_1 + M_2) &= fv(M_1) + fv(M_2) \\
fv(P_1 \mid P_2) &= fv(P_1) \cup fv(P_2) \\
fv(c\ x{:}r!.P) &= \{x\} \cup fv(P) \\
fv(c\ x?.P) &= \{x\} \cup fv(P) \\
fv(d\ x?(\tilde{y}).P) &= \{x\} \cup (fv(P) \setminus \{\tilde{y}\}) \\
fv(d\ x{:}r!(\tilde{z}).P) &= \{x\} \cup \{\tilde{z}\} \cup fv(P) \\
fv((\nu x)P) &= fv(P) \setminus \{x\} \\
fv(A(\tilde{x})) &= \{\tilde{x}\} \\
fv(A(\tilde{x}) \triangleq P) &= fv(P) \setminus \{\tilde{x}\}
\end{aligned}
$$

Figure 2.9: Free channel names of BioAmbients.

$$
\begin{array}{rcl}
[\mathbf{0}] & \equiv & \mathbf{0} \\
P \mid \mathbf{0} & \equiv & P \\
P_1 \mid P_2 & \equiv & P_2 \mid P_1 \\
M_1 + M_2 & \equiv & M_2 + M_1 \\
(P_1 \mid P_2) \mid P_3 & \equiv & P_1 \mid (P_2 \mid P_3) \\
(M_1 + M_2) + M_3 & \equiv & M_1 + (M_2 + M_3) \\
(\nu x)(P \mid Q) & \equiv & (\nu x)P \mid Q, \text{ if } x \notin fv(Q) \\
(\nu x)(\nu y)P & \equiv & (\nu y)(\nu x)P \\
[(\nu x)P] & \equiv & (\nu x)[P] \\
P \equiv_\alpha Q & \Rightarrow & P \equiv Q
\end{array}
$$

Figure 2.10: Rules of the structural congruence of BioAmbients.

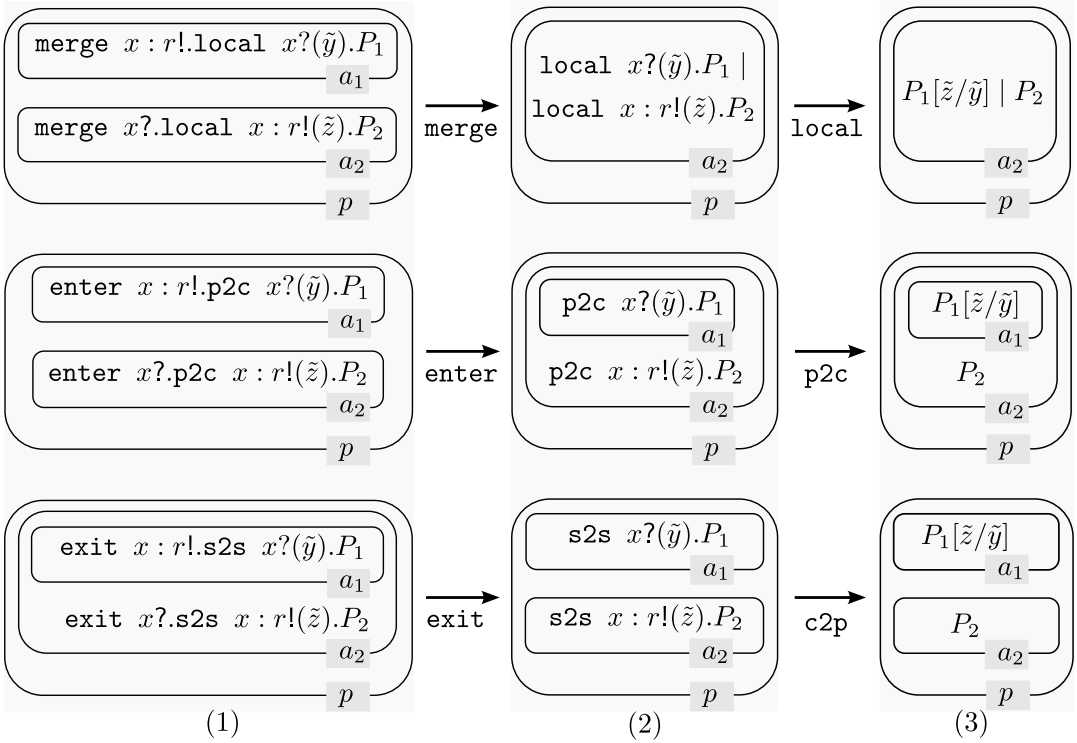


Figure 2.11: Communication directions and rearrangement capabilities of BioAmbients.

2.2.2 Non-deterministic Operational Semantics

The non-deterministic semantics of BioAmbients with priorities is presented in Figure 2.11. It builds upon the non-deterministic semantics of the π-calculus with priorities in that it adopts the rules for process application and convergence. The usual structural rules are extended by rule (AMB), which allows for reduction steps in ambients. Rule (CONV) for process convergence makes use of an additional predicate $C(P)$, which recursively checks that the nested ambient structure of P does not contain any defined processes. As before, it is assumed that all bound variables are named distinctly before applying any reduction step.

Rules to handle interaction prefixes form the major extension to the π-calculus syntax and follow the ideas of communication directions and rearrangement capabilities in nested ambient structures as depicted in Figure 2.11. Rules ($\text{COM}^{\texttt{LOCAL}}$), ($\text{COM}^{\texttt{P2C}}$), ($\text{COM}^{\texttt{C2P}}$), and ($\text{COM}^{\texttt{S2S}}$) allow processes to communicate if they are located in the same ambient, from a surrounding to a child ambient, from a surrounded ambient to the parent ambient, and between two ambients that are located in the same ambient, respectively. As before, communication may only happen if sender and receiver perform on the same channel and provide the same number of arguments. If so, the formal parameters of the receiver are replaced by the actual parameters of the sender in the receiving process. Rules (ENTER), (EXIT), and (MERGE) enable an ambient to enter another ambient, which has to be located in the same ambient, to exit its surrounding ambient, resulting in two ambients

Application and Interaction steps

$$(\text{APP})\ \frac{A(\tilde{x}) \triangleq P}{A(\tilde{y}) \xrightarrow[nd]{app} P[\tilde{y}/\tilde{x}]}$$

$$(\text{COM}^{\texttt{LOCAL}})\ \frac{|\tilde{y}| = |\tilde{z}| \quad P' = P_1[\tilde{z}/\tilde{y}] \mid [P_2 \mid P_2']}{\texttt{c2p}\ x?(\tilde{y}).P_1 + M_1 \mid [\texttt{c2p}\ x{:}r!(\tilde{z}).P_2 + M_2 \mid P_2'] \xrightarrow[nd]{r} P'}$$

$$(\text{COM}^{\texttt{C2P}})\ \frac{|\tilde{y}| = |\tilde{z}| \quad P' = P_1[\tilde{z}/\tilde{y}] \mid [P_2 \mid P_2']}{\texttt{c2p}\ x?(\tilde{y}).P_1 + M_1 \mid [\texttt{c2p}\ x{:}r!(\tilde{z}).P_2 + M_2 \mid P_2'] \xrightarrow[nd]{r} P'}$$

$$(\text{COM}^{\texttt{P2C}})\ \frac{|\tilde{y}| = |\tilde{z}| \quad P' = [P_1[\tilde{z}/\tilde{y}] \mid P_1'] \mid P_2}{[\texttt{p2c}\ x?(\tilde{y}).P_1 + M_1 \mid P_1'] \mid \texttt{p2c}\ x{:}r!(\tilde{z}).P_2 + M_2 \xrightarrow[nd]{r} P'}$$

$$(\text{COM}^{\texttt{S2S}})\ \frac{|\tilde{y}| = |\tilde{z}| \quad P' = [P_1[\tilde{z}/\tilde{y}] \mid P_1'] \mid [P_2 \mid P_2']}{[\texttt{s2s}\ x?(\tilde{y}).P_1 + M_1 \mid P_1'] \mid [\texttt{s2s}\ x{:}r!(\tilde{z}).P_2 + M_2 \mid P_2'] \xrightarrow[nd]{r} P'}$$

$$(\text{ENTER})\ \frac{P' = [P_1 \mid P_1' \mid [P_2 \mid P_2']]}{[\texttt{enter}\ x?.P_1 + M_1 \mid P_1'] \mid [\texttt{enter}\ x{:}r!.P_2 + M_2 \mid P_2'] \xrightarrow[nd]{r} P'}$$

continued...

Figure 2.12: Rules of operational semantics of BioAmbients with priorities in $(R, <)$.

$$(\text{EXIT})\ \frac{P' = [P_1 \mid P_1'] \mid [P_2 \mid P_2']}{[\texttt{exit}\ x?.P_1 + M_1 \mid P_1' \mid [\texttt{exit}\ x{:}r!.P_2 + M_2 \mid P_2']] \xrightarrow[nd]{r} P'}$$

$$(\text{MERGE})\ \frac{P' = [P_1 \mid P_1' \mid P_2 \mid P_2']}{[\texttt{merge}\ x?.P_1 + M_1 \mid P_1'] \mid [\texttt{merge}\ x{:}r!.P_2 + M_2 \mid P_2'] \xrightarrow[nd]{r} P'}$$

Program errors

$$(\text{E.COM})\ \frac{|\tilde{y}| \neq |\tilde{z}|}{[\ldots[[d\ x?(\tilde{y}).P_1 + M_1 \mid P_1'] \mid P_2'] \ldots \mid P_n'] \mid [\ldots[[d\ x{:}r!(\tilde{z}).P_2 + M_2 \mid \hat{P}_1] \mid \hat{P}_2] \ldots \mid \hat{P}_n] \xrightarrow[nd]{err} \perp}$$

Structural rules where $\beta \in \{err, app\} \cup R$

$$(\text{PAR})\ \frac{P_1 \xrightarrow[nd]{\beta} P_1'}{P_1 \mid P_2 \xrightarrow[nd]{\beta} P_1' \mid P_2} \qquad (\text{AMB})\ \frac{P \xrightarrow[nd]{\beta} P'}{[P] \xrightarrow[nd]{\beta} [P']}$$

$$(\text{NEW})\ \frac{P \xrightarrow[nd]{\beta} P'}{(\nu x)P \xrightarrow[nd]{\beta} (\nu x)P'} \qquad (\text{STRUC})\ \frac{P \equiv P_1 \quad P_1 \xrightarrow[nd]{\beta} P_2 \quad P_2 \equiv P'}{P \xrightarrow[nd]{\beta} P'}$$

continued...

Figure 2.12: Rules of the operational semantics of BioAmbients with priorities in $(R, <)$(continued).

Error-free convergence of application

$$C(P) = \begin{cases} \mathit{true}, \text{ if } P = \prod_{i=1}^{n} M_i \\ \wedge_{i=1}^{m} C(P_i), \text{ if } P = \prod_{i=1}^{n} M_i \mid \prod_{i=1}^{m} [P_i] \\ \mathit{false}, \text{ else} \end{cases}$$

$$(\text{CONV}) \frac{P \xrightarrow[nd]{app}{}^{*} P' \quad P' = (\nu \tilde{x})P_1 \quad C(P_1) \quad \neg P' \xrightarrow[nd]{err} \perp}{P \Downarrow P'}$$

Reduction ($r' \in R$)

$$(\text{PRIOR}) \frac{P \Downarrow P_1 \quad P_1 \xrightarrow[nd]{r} P' \quad \neg \exists r' \in R.\, r < r' \wedge P_1 \xrightarrow[nd]{r'} P_2}{P \to P'}$$

Figure 2.12: Rules of the operational semantics of BioAmbients with priorities in $(R, <)$(continued).

that are surrounded by the same ambient, and two ambients, which are located in the same ambient, to become one. Communication and rearrangement reductions are equally annotated with priority r in order to select a reduction with highest priority by rule (PRIOR), as before. A single error rule (E.COM), reduces a process P to erroneous process $\perp$, if it contains unguarded communication attempts on some channel with different numbers of arguments. Notice, that this happens independently of the ambient structure, i.e. any false communication attempt is detected even if not located in neighboring ambients.

Clearly, processes in BioAmbients with priority also converge to a unique result.

Proposition 4 (Convergence uniqueness of BioAmbients with priorities)**.** *For all process P there exists at most one class $[P']_{\equiv}$, such that $P \Downarrow P'$.*

Proof. The proof is the same as for the π-calculus with priorities, building upon a lemma that states that $\xrightarrow[nd]{app}$ is uniform confluent modulo structural congruence. □

Chapter 3

The Attributed π-Calculus

This chapter introduces the attributed π-calculus with its attribute language, syntax, non-deterministic and stochastic semantics, and simple type system (Section 3.1). Modeling and expressiveness studies in Sections 3.2 and 3.3, respectively, show the usefulness of the attributed π-calculus for the spatial and stochastic modeling of cell-biological systems. Section 3.4 presents a stochastic simulator for the attributed π-calculus with a feasible computational complexity as indicated by the results of first performance experiments provided in Section 3.5.

3.1 Language

In this section the attributed π-calculus $\pi(\mathscr{L})$ is introduced, by extending the π-calculus with priority with richer sets of values and expressions in some call-by-value λ-calculus $\mathscr{L}$ that is called the attribute

language $\mathscr{L}$. Generalized senders and receivers feature λ-expressions, which allow defining constraints on communication steps, subsuming priority levels and stochastic rate constants. As before, both a non-deterministic and a stochastic operational semantics are presented (except that the set of successful values must be $\mathbb{R}_+^\infty$ with 2 levels of priority in the stochastic case).

In the following, first the idea of communication constraints is recalled and basic design decisions are explained in Section 3.1.1. Attribute languages, syntax, and non-deterministic and stochastic semantics of $\pi(\mathscr{L})$ are introduced in Sections 3.1.2, 3.1.3, 3.1.4, and 3.1.6, respectively. In Section 3.1.5 it is shown that the convergence of attributed processes produces unique results. A type system for $\pi(\mathscr{L})$ is developed in Section 3.1.7, by refining the type system of the π-calculus, such that it considers the types of constraint functions and arguments.

3.1.1 Idea of Communication Constraints

For illustration, consider proteins $\mathsf{Prot}(x)$, which may bind to operators $\mathsf{Op}(y)$ only if they have equal types $x = y$. Expressing such constraints in an object-centered languages such as the π-calculus is difficult, since it concerns the attribute values of two independent processes. The solution proposed for the attributed π-calculus is to use functions such as $\lambda x.x{=}y$ on the receiver side, and to apply them to the value of x on the sender side:

$\mathsf{Prot}(x) \triangleq \mathit{bind}[x]!() . \mathbf{0}$
$\mathsf{Op}(y) \triangleq \mathit{bind}[\lambda x . \ x{=}\, y]?() . \mathsf{OpBound}(y)$

These definitions allow for the following reduction steps for arbitrary values b, since the application $(\lambda x.x{=}\mathtt{b})\mathtt{b}$ evaluates to Boolean true: $\mathsf{Prot}(\mathtt{b}) \mid \mathsf{Op}(\mathtt{b}) \to \mathsf{OpBound}(\mathtt{b})$ In order to permit richer sets of values and constraints, such as, e.g., arithmetic values and constraints, call-by-value λ-calculi are used as attribute languages $\mathscr{L}$. The choice of constants is kept parametric, in order to avoid reinventing independent calculi for the many useful choices in practice. The semantics of $\pi(\mathscr{L})$ is defined such that they are independent of the concrete choice of the attribute language.

3.1.2 Attribute Languages

An attribute language is a functional programming language that provides expressions by which to compute values. Expressions are built from constants for numbers, functions, relations, or biological entities (such as 0, 1, $+$, $*$, $\geq$, `fst`, `snd`, `repressor`, ...) and from variables $x, y \in \mathit{Vars}$, which may in particular take the role of communication channels. Whenever facing ambiguities, constants are printed in courier, e.g. `first`, and variables in italic font, e.g. *bind*. The set of real numbers is denoted by $\mathbb{R}$, where $\mathbb{R}_+$ is restricted to positive values and $\mathbb{R}^\infty$ includes the value ∞. The set $\mathbb{N}$ contains all natural numbers, and $\mathbb{N}_0$ zero in addition. Boolean values are given by the set $\mathtt{Bool} = \{\mathtt{true}, \mathtt{false}\}$.

As an example, consider the expression $(\mathtt{snd}\ x) + (\mathtt{fst}\ y)$ in which

`fst`, `snd`, and $+$ are constants, while x and y are variables. In the following process definition, this expression defines a stochastic rate constant or a priority level: $\mathsf{A}(x,y,z) \triangleq \mathsf{z}[(\texttt{snd}\ x) + (\texttt{fst}\ y)]!().P$.

An attribute language with variables in *Vars* is a tuple $\mathscr{L} = (\mathit{Consts}, \Downarrow, R, <)$, that contains a set of constants $c \in \mathit{Consts}$, a big-step evaluator $\Downarrow$ for λ-expressions with constants, pairs, and conditionals, and a partially ordered set $(R,<)$ of successful values, that enables communication steps. More precisely, the first component *Consts* is a finite set that fixes the constants of a call-by-value λ-calculus. The set of expressions $e \in \mathit{Exprs}$ of $\mathscr{L}$ is defined as the set of all λ-expressions with constants in $c \in \mathit{Consts}$ and variables in $x \in \mathit{Vars}$. The set of values $v \in \mathit{Vals}$ of $\mathscr{L}$ is the subset of all values of this λ-calculus.

$$\begin{array}{rcl} c \in \mathit{Consts} & ::= & \texttt{false} \mid \texttt{true} \mid \texttt{fst} \mid \texttt{snd} \mid \ldots \\ v \in \mathit{Vals} & ::= & x \mid c \mid \lambda x.e \mid \langle v_1, v_2 \rangle \\ e \in \mathit{Exprs} & ::= & v \mid e_1 e_2 \mid \langle e_1, e_2 \rangle \mid \textsf{if } e \textsf{ then } e_1 \textsf{ else } e_2 \end{array}$$

As usual, λ-expressions provide abstractions $\lambda x.e$ and applications $e_1 e_2$ for specifying function definitions and function application. The set of constants *Consts* is assumed to contain the Booleans `true` and `false`, and pair projections `fst` and `snd`. There also exist expressions for pairs $\langle e_1, e_2 \rangle$ and Boolean conditionals $\textsf{if } e \textsf{ then } e_1 \textsf{ else } e_2$.

The third component is a set $R \subseteq \mathit{Vals}$ of successful values enabling communication steps, such as priorities or stochastic rate constants. The fourth component $<$ is a partial order on successful values R that defines priority levels. The last component of $\mathscr{L}$ is a big-step evaluator for λ-expressions, i.e. a partial function $\Downarrow : \mathit{dom}(\Downarrow) \to \mathit{Vals}$ from

$$(\text{V})\ \frac{v \in \mathit{Vals}}{v \Downarrow v} \qquad (\text{FUN})\ \frac{e_1 \Downarrow \lambda x.e_1' \quad e_2 \Downarrow v' \quad e_1'[v'/x] \Downarrow v}{e_1 e_2 \Downarrow v}$$

$$(\text{PAIR})\ \frac{e_1 \Downarrow v_1 \quad e_2 \Downarrow v_2}{\langle e_1, e_2 \rangle \Downarrow \langle v_1, v_2 \rangle} \qquad (\text{SELECT})\ \frac{e \Downarrow \langle v_1, v_2 \rangle}{\texttt{fst}\ e \Downarrow v_1 \quad \texttt{snd}\ e \Downarrow v_2}$$

$$(\text{COND}_1)\ \frac{e \Downarrow \texttt{true} \quad e_1 \Downarrow v_1}{\textsf{if } e \textsf{ then } e_1 \textsf{ else } e_2 \Downarrow v_1}$$

$$(\text{COND}_2)\ \frac{e \Downarrow \texttt{false} \quad e_2 \Downarrow v_2}{\textsf{if } e \textsf{ then } e_1 \textsf{ else } e_2 \Downarrow v_2}$$

Figure 3.1: Big-step evaluator of call-by-value λ-calculus with pairs and conditionals.

$$(\text{EQ}_1)\ \frac{e_1 \Downarrow v \quad e_2 \Downarrow v \quad v \in \mathit{Vars} \cup \mathit{Consts}}{e_1{=}e_2 \Downarrow \texttt{true}}$$

$$(\text{EQ}_2)\ \frac{e_1 \Downarrow v_1 \quad e_2 \Downarrow v_2 \quad v_1 \neq v_2 \in \mathit{Vars} \cup \mathit{Consts}}{e_1{=}e_2 \Downarrow \texttt{false}}$$

$$(+_{\mathbb{N}})\ \frac{e_1 \Downarrow n_1 \quad e_2 \Downarrow n_2 \quad n_1 +_{\mathbb{N}} n_2 = n}{e_1 + e_2 \Downarrow n}$$

Figure 3.2: Additional rules of big-step evaluator of the attribute language $\lambda(\mathbb{N}_0, +, =)_{<_1}$.

λ-expressions in a domain $dom(\Downarrow) \subseteq Exprs$ to values. It can be understood as a black box algorithm that evaluates all expressions to values or failure, in case of program errors or non-termination. Instead of $\Downarrow(e) = v$ the notation $e \Downarrow v$ is used, where v is called the value of e.

It is assumed that the big-step evaluator satisfies the usual rules of the call-by-value λ-calculus with conditionals in Figure 3.1. Rule (V) states that all values evaluate to themselves. Rule (FUN) defines the standard meaning of call-by-value function application. It says that the value of an application $e_1 e_2$ is obtained by evaluating e_1 to some function $\lambda x.e_1'$ and e_2 to some value v', and then returning the value of $e_1'[v'/x]$. Rule (PAIR) states that the value of a pair is the pair of values of its components (as standard in call-by-value languages). Rule (SELECT) states the usual meaning of pair selectors. Rule (COND_1) and (COND_2) defines the semantics of conditionals such that only the needed branch is evaluated. For richer attribute languages with further constants, such as $+$, $*$, or a call-by-value fixed point operator, further rules need to be added. This is possible, since the big-step evaluator is kept abstract.

As a first example, consider the attribute language $\lambda(R)_<$ for some partially ordered set $(R, <)$ of priority levels. Set R contains the only successful values, which are ordered according to $<$. The big-step evaluator remains unchanged, since no function constants are added.

A second example forms the attribute language $\lambda(\mathbb{N}_0, +, =)_{<_1}$, where the call-by-value λ-calculus is extended by constants for natural numbers with 0 and addition $+$. The successful values are nonzero natural numbers and there is a single level of priority, fixed by the

empty partial order that is denoted by $<_1$. Constant $=$ defines equality on all constants and variables, i.e. on Booleans, natural numbers, channel names, and the function constants. The required extensions of the big-step evaluator are given in Figure 3.2. Notice that addition and equality are treated as curried binary functions. Infix notation is freely used, i.e. $e_1{+}e_2$ instead of $(+\ e_1)\ e_2$, and respectively, $e_1{=}e_2$ instead of $(=\ e_1)\ e_2$.

As a third example, consider the attribute language $\lambda(\mathbb{R}_+^\infty)_{<_2}$, whose successful values are the stochastic rate constants in $\mathbb{R}_+^\infty$. There are two levels of priority, lower priority for all positive real numbers in $\mathbb{R}_+$ and higher priority for ∞. The obvious order that introduces these two levels of priority is denoted with $<_2$.

Further extensions of the attribute language might be useful in various applications, such as n-tuples (beyond pairs), lists, or case statements, but cannot be obtained just by adding new constants, since they require new forms of expressions and values. For the sake of simplicity, these shall be omitted here.

Values of expressions may be undefined, such that for some expressions e there exists no value v with $e \Downarrow v$, due to two possible reasons. The first reason is the occurrence of programming errors, like division by 0, or type errors, like sending or receiving on values, which are no channels. The second reason is non-termination, which may arise in an untyped setting or in rich attribute languages with fixed point operators.

In Section 3.1.7, a type system for the attributed π-calculus is presented, which prevents type errors. If no constants are added to the

Prefixes	π	::=	$e_1[e_2]?(\tilde{x})$	receiver
		$\mid$	$e_1[e_2]!(\tilde{e})$	sender
Sums	M	::=	$\pi.P$	guarded process
		$\mid$	$M_1 + M_2$	choice
Processes	P	::=	M	sums
		$\mid$	$A(\tilde{e})$	defined process
		$\mid$	$P_1 \mid P_2$	parallel composition
		$\mid$	$(\nu x)P$	channel creation
		$\mid$	$\mathbf{0}$	empty solution
Definitions	D	::=	$A(\tilde{x}) \triangleq P$	parametric process definition

Figure 3.3: Syntax of $\pi(\mathscr{L})$, where $x, \tilde{x} \in \mathit{Vars}$, and $e_1, e_2, \tilde{e} \in \mathit{Exprs}$.

attribute language, it even excludes non-termination. For more general attribute languages, however, type systems may neither exclude all program errors (e.g. division by 0) nor ensure termination (e.g. fixed point operators).

3.1.3 Syntax of Processes

Let $\mathscr{L}$ be an attribute language over some infinite set of variables $x \in \mathit{Vars}$, with expressions $e \in \mathit{Exprs}$ and values $v \in \mathit{Vals}$. The syntax of the attributed π-calculus, $\pi(\mathscr{L})$, is defined in Figure 3.3. Compared to the π-calculus, variables x of various types are used instead of just channel names (which correspond to variables of channel type), expressions

e are permitted in all non-binding positions, where previously only channel names were allowed, and priorities "$:r$" in send prefixes and the corresponding receive prefixes are extended to expressions "$[e]$". Receiver prefixes thus have the form $e_1[e'_1]?(\tilde{x})$ and sender prefixes the form $e_2[e'_2]!(\tilde{e})$. Prefixes in which e_1 respectively e_2 do not evaluate to channels are erroneous. The application $e'_1 e'_2$ imposes a constraint on the ability to communicate, in addition to that e_1 and e_2 must evaluate to the same channel. Communication is permitted only if $e'_1 e'_2 \Downarrow v$ for some successful value $v \in R$. This value then fixes the priority or the stochastic rate constant of a communication step.

For illustration, consider three instances of the attributed π-calculus with different attribute languages. For the first example, take the calculus $\pi(\lambda(R)_<)$ whose attribute language is a λ-calculus with priority levels. The second example is $\pi(\lambda(\mathbb{N}_0,+,=)_{<_1})$ which provides natural numbers and defines addition and equality. The third example is $\pi(\lambda(\mathbb{R}^\infty_+)_{<_2})$, where λ-expressions can be used in order to compute priorities on two levels.

For example, consider process definitions in $\pi(\lambda(\mathbb{N}_0,+,=)_{<_1})$, which express the following rule schema that compactly represents an infinite set of chemical reaction rules:

$$\textit{react:}\ \forall x,y \in \mathbb{N}\ \mathsf{A}(x),\ \mathsf{B}(y) \xrightarrow{x+y} \mathsf{A}(x+1),\ \mathsf{B}(y)$$

$$\mathsf{A}(x) \triangleq \textit{react}[x]!() .\mathsf{A}(x+1)$$
$$\mathsf{B}(y) \triangleq \textit{react}[\lambda x.x+y]?() .\mathsf{B}(y)$$

The process $\mathsf{A}(2) \mid \mathsf{B}(5)$ may communicate on channel *react* and become $\mathsf{A}(3) \mid \mathsf{B}(5)$, since the constraint function $\lambda x.x + 5$ applied to

the constraint argument 2 yields the successful value 7. More formally, this number can be computed by evaluating the interaction constraint $(\lambda x.x+5)2$ via (FUN), (V), and $(+_{\mathbb{N}})$ as follows, where $v_f = \lambda x.x+5$:

$$\dfrac{\dfrac{\dfrac{}{v_f \in \mathit{Vals}}}{v_f \Downarrow v_f} \quad \dfrac{}{2 \in \mathit{Vals}}}{(\lambda x.x+5)2 \Downarrow 7} \quad \dfrac{\dfrac{}{2 \in \mathit{Vals}}}{2 \Downarrow 2} \quad \dfrac{\dfrac{}{5 \in \mathit{Vals}}}{5 \Downarrow 5} \quad 2 +_{\mathbb{N}} 5 = 7}{2+5 \Downarrow 7}$$

Free variables $\mathit{fv}(P)$ are defined as before, except that it is now necessary to additionally account for free variables $\mathit{fv}(e)$ in λ-expressions e, i.e. those occurring out of the scope of all λ-binders in e:

$$\begin{aligned} \mathit{fv}(A(\tilde{e})) &= \mathit{fv}(\tilde{e}) \\ \mathit{fv}(e_1[e_2]?(\tilde{y}).P) &= \mathit{fv}(e_1) \cup \mathit{fv}(e_2) \cup (\mathit{fv}(P) \setminus \{\tilde{y}\}) \\ \mathit{fv}(e_1[e_2]!(\tilde{e}).P) &= \mathit{fv}(e_1) \cup \mathit{fv}(e_2) \cup \mathit{fv}(\tilde{e}) \cup \mathit{fv}(P) \end{aligned}$$

Bound variables $\mathit{bv}(P)$ are defined as before, except that λ-binders in expressions $e \in \mathit{Exprs}$ are included too. The structural congruence on processes $\equiv$ remains unchanged, except that α-conversion becomes applicable to bound variables in λ-expressions.

3.1.4 Non-deterministic Operational Semantics

The non-deterministic operational semantics of the attributed π-calculus with priorities is given by the rules in Figure 3.4 and inherits the convergence and reduction rules and the structural rules of the π-calculus with priority in Figure 2.4. The new rules always evaluate

Communication and application steps

$$(\text{TUP})\ \frac{\wedge_{i=1}^{n} e_i \Downarrow v_i}{(e_i)_{i=1}^{n} \Downarrow (v_i)_{i=1}^{n}} \qquad (\text{APP})\ \frac{\tilde{e} \Downarrow \tilde{v} \quad A(\tilde{x}) \triangleq P}{A(\tilde{e}) \xrightarrow[nd]{app} P[\tilde{v}/\tilde{x}]}$$

$$(\text{SEND})\ \frac{e_1 \Downarrow x \quad e_2 \Downarrow v_2 \quad \tilde{e} \Downarrow \tilde{v}}{e_1[e_2]!(\tilde{e}) \Downarrow x[v_2]!(\tilde{v})} \qquad (\text{REC})\ \frac{e_1 \Downarrow x \quad e_2 \Downarrow v_2}{e_1[e_2]?(\tilde{y}) \Downarrow x[v_2]?(\tilde{y})}$$

$$(\text{COM})\ \frac{\pi_1 \Downarrow x[v_1]?(\tilde{y}) \quad \pi_2 \Downarrow x[v_2]!(\tilde{v}) \quad v_1 v_2 \Downarrow r \in R \quad |\tilde{v}| = |\tilde{y}|}{\pi_1.P_1 + M_1 \mid \pi_2.P_2 + M_2 \xrightarrow[nd]{r} P_1[\tilde{v}/\tilde{y}] \mid P_2}$$

Program errors

$$(\text{E.PREF})\ \frac{\neg\exists \pi'.\pi \Downarrow \pi'}{\pi.P + M \xrightarrow[nd]{err} \bot}$$

$$(\text{E.COM})\ \frac{\pi_1 \Downarrow x[v_1]?(\tilde{y}) \quad \pi_2 \Downarrow x[v_2]!(\tilde{v}) \quad |\tilde{y}| \neq |\tilde{v}|}{\pi_1.P_1 + M_1 \mid \pi_2.P_2 + M_2 \xrightarrow[nd]{err} \bot}$$

$$(\text{E.CONSTR})\ \frac{\pi_1 \Downarrow x[v_1]?(\tilde{y}) \quad \pi_2 \Downarrow x[v_2]!(\tilde{v}) \quad \neg\exists v.v_1 v_2 \Downarrow v}{\pi_1.P_1 + M_1 \mid \pi_2.P_2 + M_2 \xrightarrow[nd]{err} \bot}$$

Figure 3.4: Non-deterministic operational semantics of $\pi(\mathscr{L})$. Additionally, the convergence and reduction rules and the structural rules of the π-calculus with priorities in Figure 2.4 are inherited.

expressions to values before applying communication or application steps (COM) and (APP). This is done by using the big-step evaluator of the attribute language according to axioms (SEND), (REC), and (TUP). Notice that evaluation of expressions may get stuck – in contrast to the pure π-calculus with priorities. For instance, an application $A(\tilde{e})$ gets stuck if the evaluation of one of the expressions in $\tilde{e}$ does not succeed. In this case, application does not converge, so that communication gets blocked.

The communication rule (COM) permits receivers $x[v_1]?(\tilde{y}).P_1$ and senders $x[v_2]!(\tilde{v}).P_2$ to interact only if the expression $v_1 v_2$ evaluates to a successful value $v_1 v_2 \Downarrow r \in R$. This value defines the priority level of the communication step. Communication steps perform substitutions $[\tilde{v}/\tilde{y}]$ replacing variables by values. The application of substitutions is well-defined for all processes, since the syntax of $\pi(\mathscr{L})$ permits values in all positions where free variables may be used. Notice, however, that substitution may raise program errors as specified by rule (E.PREF), where non-channel values arise in sender or receiver position. Rule (E.CONSTR) specifies constraint errors, where the evaluation of communication constraints $v_1 v_2$ fails.

The closure rules in Figure 2.4 remain unchanged. As before, all relations are closed under the structural rules, while (CONV) applies definitions exhaustively and continues to require error-freeness. The overall reduction relation $P \to P'$ is defined by rule (PRIOR) without change. All changes are imported from the changes in communication, application, and error steps.

Example 4. Consider a client-server system in the attributed π-calculus with integers and strings and two levels of priority $\pi(\lambda(\mathtt{Int},\mathtt{String})_{<_2})$, such that there are two successful values $R = \{1,2\}$ ordered by the least ordering $<_2$ that satisfies $1 <_2 2$. Furthermore, let $\mathtt{never} =_{df} 0$, $\mathtt{low} =_{df} 1$, $\mathtt{high} =_{df} 2$, and function $\mathtt{price} : \mathtt{String} \to \mathtt{Int}$ be defined by the following expression:

$$\mathtt{price} =_{df} \lambda x.\mathbf{if}\ x = \mathtt{chicken}\ \mathbf{then}\ 10\ \mathbf{else}\ \mathbf{if}\ x = \mathtt{fish}\ \mathbf{then}\ 14\ \mathbf{else}\ 0$$

Servers are accessible on a public channel *connect* to all clients that know a password *key* of type `String`. The server applies function `price` to a string value received from the client and returns the value on a private *ret* channel that was also provided by the client. Servers and clients are defined as follows:

$$\begin{aligned}
&\mathsf{Serv}() \triangleq \\
&\quad \mathit{connect}[\lambda k.\mathbf{if}\ k = \mathtt{key}\ \mathbf{then}\ \mathtt{low} \\
&\qquad\qquad \mathbf{else}\ \mathtt{never}]?(x, \mathit{ret}).\\
&\qquad\qquad\quad (\mathit{ret}[\mathtt{high}]!(\mathtt{price}\ x).\mathbf{0} \mid \mathsf{Serv}()) \\
&\mathsf{Client}(s) \triangleq (\nu\ \mathit{ret})\ \mathit{connect}[\mathit{key}]!(s, \mathit{ret}).\\
&\qquad\qquad\qquad\qquad \mathit{ret}[\lambda z.z]?(y).P
\end{aligned}$$

A process with two clients and one server can be reduced as follows:

$$
\begin{aligned}
&\mathsf{Serv}() \mid \mathsf{Client}(\texttt{chicken}) \mid \mathsf{Client}(\texttt{fish}) \rightarrow \\
&(\nu\ ret)(ret[\texttt{high}]!(\texttt{price chicken}).\mathbf{0} \mid \mathsf{Serv}() \mid \\
&\qquad ret[\lambda x.x]?(y).P) \mid \mathsf{Client}(\texttt{fish}) \equiv \\
&\mathsf{Serv}() \mid \mathsf{Client}(\texttt{fish}) \mid (\nu\ ret)(ret[\texttt{high}]!(\texttt{price chicken}).\mathbf{0} \mid \\
&\qquad ret[\lambda x.x]?(y).P) \rightarrow \\
&\mathsf{Serv}() \mid \mathsf{Client}(\texttt{fish}) \mid (\nu\ ret)\ P[10/y]
\end{aligned}
$$

No unrelated client can access a client-server dialog, since private channels are used for communication. Note however, that the second communication action gets highest priority, so that client $\mathsf{Client}(\texttt{fish})$ cannot act before $\mathsf{Client}(\texttt{chicken})$ obtained the price for `chicken`.

3.1.5 Uniqueness of Convergence

Similar to the π-calculus with priority, it needs to be shown that convergence of attributed processes produces unique results. The next lemma extends on Lemma 1. It states that application is confluent (Definition 2), such that exhaustive application must lead to a unique outcome (including non-termination). This allows for the conclusion that the convergence of attributed processes is unique.

Lemma 2. *The rewrite relation* $\xrightarrow[nd]{app}$ *is confluent modulo structural congruence. Irreducible processes are congruent to processes of the form* $(\nu\tilde{x})\prod_{i=1}^{n} P_i$ *such that all* P_i *are sums or match some defined process* $A_i(\tilde{e}_i)$ *with* $\neg\exists\tilde{v}.\tilde{e}_i \Downarrow \tilde{v}$.

Proof. A standard analysis of the structural congruence yields the following:

Claim. Let $P = (\nu\tilde{x})\prod_{i=1}^{n} P_i$ be a process in prenex normal form in which all bound variables are named distinctly, and such that all P_i are sums or defined processes. In this case, $P \xrightarrow[nd]{app} P'$ if and only if the following rule applies:

$$\frac{1 \leq j \leq n \quad \tilde{e}_j \Downarrow \tilde{v}_j \qquad A_j(\tilde{y}_j) \triangleq Q_j \quad P_j = A_j(\tilde{e}_j) \quad P' \equiv (\nu\tilde{x})(\prod_{i=1, i\neq j}^{n} P_i \mid Q_j[\tilde{v}_j/\tilde{y}_j])}{P \xrightarrow[nd]{app} P'}$$

Let the rewrite system on congruence classes of processes be defined as $[P]_{\equiv} \xrightarrow[nd]{app} [P']_{\equiv}$ if $P \xrightarrow[nd]{app} P'$. The above claim shows that this rewrite system terminates on equivalence classes of processes of the form $(\nu\tilde{x})\prod_{i=1}^{n} P_i$ where all P_i are either sums or irreducible defined processes $A(\tilde{e})$. The uniform confluence of this rewrite system can be proven by minor adaptation of the proof in Lemma 1. □

Proposition 5 (Convergence uniqueness of $\pi(\mathscr{L})$)**.** *For every attributed process P, there exists at most one class* $[P']_{\equiv}$ *such that* $P \Downarrow P'$.

Proof. This follows immediately from the confluence result in Lemma 2. □

Remark 2. *If* $P \equiv (\nu\tilde{x})\prod_{i=1}^{n} M_i$ *and* $\neg P \xrightarrow[nd]{err} \perp$ *then* $P \equiv P' \Leftrightarrow P \Downarrow P'$.

Proof. Analogue to the proof of Remark 1. □

Labeled communication steps ($\ell \in \mathbb{N}^4$, $r \in \mathbb{R}_+^\infty$)

$$(\text{COM}_\ell)\ \frac{\pi_{i_1}^{j_1} \Downarrow x[v_1]?(\tilde{y}) \quad \pi_{i_2}^{j_2} \Downarrow x[v_2]!(\tilde{v}) \quad \ell = (i_1, j_1, i_2, j_2) \quad i_1 \neq i_2 \quad v_1 v_2 \Downarrow r \in \mathbb{R}_+^\infty \quad |\tilde{y}| = |\tilde{v}|}{(\nu\tilde{x}) \prod_{i=1}^n \sum_{j=1}^{m_i} \pi_i^j.P_i^j \xrightarrow[\ell]{r} (\nu\tilde{x})(\prod_{i=1, i\neq i_1, i_2}^n \sum_{j=1}^{m_i} \pi_i^j.P_i^j \mid P_{i_1}^{j_1}[\tilde{v}/\tilde{y}] \mid P_{i_2}^{j_2})}$$

Figure 3.5: Rules of the stochastic semantics of $\pi(\mathscr{L})$. Additionally, the rules (SUM) and (COUNT) of the stochastic semantics of the stochastic π-calculus in Figure 2.5 are inherited. The rules of the non-deterministic semantics of $\pi(\mathscr{L})$ in Figure 3.4, except (COM) and (PRIOR), remain valid.

3.1.6 Stochastic Operational Semantics

In this section, a stochastic semantics for the attributed π-calculus is presented, under the condition that the set of successful values of the attribute language are the stochastic rate constants $R \subseteq \mathbb{R}_+^\infty$. As in the stochastic π-calculus, highest priority is assigned to communication steps with infinite rates, and lowest priority to all others.

The rules of the stochastic semantics of $\pi(\mathscr{L})$ are given in Figure 3.5. Few changes are needed with respect to the stochastic π-calculus. In particular, both calculi have the same closure rules, which were already presented in Figure 2.5.

The main difference concerns the new communication rule (COM_ℓ), where all expressions have to be evaluated in order to compute the

stochastic rate constant. All other differences are hidden in the convergence predicate, as defined in the non-deterministic operational semantics.

The stochastic version remains a proper refinement of the non-deterministic version of the attributed π-calculus with priorities.

Proposition 6. *If the successful values of $\mathscr{L}$ are $\in \mathbb{R}_+^\infty$ with the usual two levels of priority then for all processes P, P':*

$$P \to P' \text{ iff } (\exists r \in \mathbb{R}_+.\ P \xrightarrow{r} P' \vee \exists n \in \mathbb{N}.\ P \xrightarrow{\infty(n)} P')$$

The proof is mostly the same as for Proposition 2, which relates the two operational semantics of the stochastic π-calculus. The only minor difference is in the treatment of basic interaction steps.

3.1.7 Type System

Higher-order attribute languages add much expressive power to the π-calculus, but at the price of introducing many new error situations. The most frequent errors are type errors. In this section, a type system is presented which detects type errors in attributed processes. Notice that well-typedness does not exclude all kinds of errors, such as division by 0.

The type system for $\pi(\mathscr{L})$ is defined, such that it integrates the simple type system for the λ-calculus $\mathscr{L}$ into the type system of the π-calculus from Section 2.1.6. In the following, it is shown that the type system of $\pi(\mathscr{L})$ is safe if the type system of $\mathscr{L}$ is. Whether this holds depends on the precise definition of the big-step evaluator of $\mathscr{L}$

that was left open here. The attribute languages $\lambda(R)_<$ in Figure 3.1 and $\lambda(\mathbb{N}_0,+,=)_{<_1}$ in Figure 3.2 are type safe. These two attribute languages are strongly normalizing, i.e. always terminating, as usual for the simply typed lambda calculus. General termination may fail, however, once new rules are added to the big-step evaluators for new constants with functional type, as e.g. recursion. In this case, the type safety of the attribute language must be checked again.

Based on type constants, such as `Int`, `Bool`, and `String`, *Types* are defined by the following grammar:

type constants	ι	::=	`Int` | `Bool` |`String`| ...	
types	τ, σ	::=	ι	constants
		|	$\tau \to \sigma$	function type
		|	$[\tau] \Rightarrow \tilde{\sigma}$	channel type
		|	$\tau \times \sigma$	pair type

A pair type states that a pair's first and second component are of types τ and σ, respectively. A function's type denotes that the function maps values of type τ to values of σ. An additional type constant `Unit` is assumed in order to type the empty value _ and the function without parameters $\lambda\_.e$, which is of type `Unit` $\to \sigma$ if e is of type σ. Channel types $[\tau] \Rightarrow \tilde{\sigma}$ now type channel constraints by τ and channel arguments by $\tilde{\sigma}$. More precisely, a channel x of type $[\tau_1 \to \tau_2] \Rightarrow \tilde{\sigma}$ can be used as follows:

- in input prefixes $x[e]?(y)$, the type of expressions e must be $\tau_1 \to \tau_2$ and the types of $\tilde{y}$ must be $\tilde{\sigma}$

- in output prefixes $x[e]!(\tilde{e}')$, the type of expression e must by τ_1 and the types of $\tilde{e}'$ must be $\tilde{\sigma}$

Type constants and the types of functions and pairs are standard. As before, *type environments* Γ are sets of type assignments for variables $x{:}\tau$ and process names $A{:}\tilde{\tau}$. Types are captured by slightly extending the syntax, here additionally by annotations to the formal parameters of λ-abstractions and constants:

$$\begin{array}{rrcl} \text{typed } \lambda\text{-expressions} & e & ::= & c_\tau \mid \lambda x:\tau.e \mid \ldots \\ \text{typed processes} & P & ::= & (\nu x){:}\tau P \mid \ldots \\ \text{typed process definition} & D & ::= & A(\tilde{x}:\tilde{\tau}) \triangleq P, \text{with } |\tilde{x}| = |\tilde{\tau}| \end{array}$$

In examples, types annotations are ignored if they are clear from the context. It is particularly useful to annotate functional types to constants, e.g., in order to type pair selectors $\mathtt{fst}_{\tau\times\sigma\to\tau}$ and $\mathtt{snd}_{\tau\times\sigma\to\sigma}$ or fixed point operators. Additionally, note that pairs of different types can also be used, since different annotations for the same constant may be used.

Example 5. In a typed version of the client-server example, constants have to be annotated with their types and a type for the new binder in the definition of process Client has to be specified:

$$\begin{array}{l} \mathsf{Serv}() \triangleq \\ \quad \mathit{connect}[\lambda k:\mathtt{Int}.\mathsf{if}\ k = \mathtt{key}_{\mathtt{Int}}\ \mathsf{then}\ \mathtt{low}_{\mathtt{Int}} \\ \qquad\qquad \mathsf{else}\ \mathtt{never}_{\mathtt{Int}}]?(x, \mathit{ret}). \\ \qquad\quad (\mathit{ret}[\mathtt{high}_{\mathtt{Int}}]!(\mathtt{price}_{\mathtt{Int}\to\mathtt{Int}}\ x).\mathbf{0} \mid \\ \qquad\qquad \mathsf{Serv}()) \end{array}$$

$$\mathtt{Client}\,(\,\mathtt{s}:\mathtt{String}\,) \triangleq$$
$$(\nu\, ret) : [\mathtt{Int} \to \mathtt{Int}] \Rightarrow (\mathtt{Int})$$
$$connect\,[\,key_{\mathtt{Int}}\,]\,!\,(\,s\,,ret\,)\,.\,ret\,[\,\lambda z : \mathtt{Int}.z\,]\,?\,(\,y\,)\,.\,P$$

The definitions are well typed if in the following type environment:

$$connect : [\mathtt{Int} \to \mathtt{Int}] \Rightarrow (\mathtt{String}, [\mathtt{Int} \to \mathtt{Int}] \Rightarrow (\mathtt{Int})),$$
$$Client : (\mathtt{String}),\ Serv : ()$$

Rules for typing expressions and processes are given in Figure 3.6. Typing rules for expressions are standard as in the simply typed λ-calculus. Rule (T.AXIOMS) assigns the type `Bool` to boolean constants. Functional constant `fst` and `snd` operate on pairs and return a value of the type of the first or second component, respectively. Equality and arithmetic functions are curried binary functions, such that e.g. addition (+) has an argument of type `Int` and returns a function with an argument and a return value of type `Int`. Rule (T.CONST) and (T.VAR) check types of constants and variables, respectively. Rule (T.PAIR) ensures that the two expressions that form a pair are of types according to the type of the pair. Rule (T.COND) states that expressions in both branches of a condition have to be of the same type, which is then the return type of the condition. Function definitions are typed by rule (T.FUNDEF), where the return type of a function is given by the type of the function's body in a type environment, which is extended by the type of the function's argument. Rule (T.FUNAPP) ensures that in a function application the first expression is of functional type and the second expression matches the type of the argument of the function. The return type of function application is the return type of the func-

Typing rules for expressions

$$(\text{T.AXIOMS})\ \frac{\tau,\sigma \text{ types}}{\begin{array}{ll} \texttt{fst} : \tau \times \sigma \to \tau & \texttt{true:Bool} \\ \texttt{snd} : \tau \times \sigma \to \sigma & \texttt{false:Bool} \\ = : \tau \to \sigma \to \texttt{Bool} & + : \mathbb{N} \to \mathbb{N} \to \mathbb{N} \end{array}}$$

$$(\text{T.CONST})\ \frac{c \in \mathit{Consts}}{\Gamma \vdash c_\tau{:}\tau} \qquad (\text{T.VAR})\ \frac{x{:}\tau \in \Gamma}{\Gamma \vdash x{:}\tau}$$

$$(\text{T.PAIR})\ \frac{\Gamma \vdash e_1{:}\tau \quad \Gamma \vdash e_2{:}\sigma}{\Gamma \vdash \langle e_1, e_2\rangle{:}\tau \times \sigma}$$

$$(\text{T.COND})\ \frac{\Gamma \vdash e{:}\texttt{Bool} \quad \Gamma \vdash e_1{:}\tau \quad \Gamma \vdash e_2{:}\tau}{\Gamma \vdash \textsf{if } e \textsf{ then } e_1 \textsf{ else } e_2{:}\tau}$$

$$(\text{T.FUNDEF})\ \frac{\Gamma, x : \tau \vdash e : \sigma}{\Gamma \vdash \lambda x : \tau.e : \tau \to \sigma}$$

$$(\text{T.FUNAPP})\ \frac{\Gamma \vdash e_1{:}\tau \to \sigma \quad \Gamma \vdash e_2{:}\tau}{\Gamma \vdash e_1 e_2{:}\sigma}$$

continued...

Figure 3.6: Type system for $\pi(\mathscr{L})$.

Typing rules for processes

$$(\text{T.REC})\ \frac{\Gamma \vdash e_1{:}[\tau] \Rightarrow \tilde{\sigma} \quad \Gamma \vdash e_2{:}\tau \quad \Gamma, \tilde{x}{:}\tilde{\sigma} \vdash P}{\Gamma \vdash e_1[e_2]?(\tilde{x}).P} \qquad (\text{T.NIL})\ \frac{}{\Gamma \vdash \mathbf{0}}$$

$$(\text{T.SEND})\ \frac{\Gamma \vdash e_1{:}[\tau_1 \to \tau_2] \Rightarrow \tilde{\sigma} \quad \Gamma \vdash e_2{:}\tau_1 \quad \Gamma \vdash \tilde{e}_3{:}\tilde{\sigma} \quad \Gamma \vdash P}{\Gamma \vdash e_1[e_2]!(\tilde{e}_3).P}$$

$$(\text{T.PAR})\ \frac{\Gamma \vdash P_1 \quad \Gamma \vdash P_2}{\Gamma \vdash P_1 \mid P_2} \qquad (\text{T.SUM})\ \frac{\Gamma \vdash M_1 \quad \Gamma \vdash M_2}{\Gamma \vdash M_1 + M_2}$$

$$(\text{T.NEW})\ \frac{\Gamma, x{:}[\tau] \Rightarrow \tilde{\sigma} \vdash P}{\Gamma \vdash (\nu x){:}[\tau] \Rightarrow \tilde{\sigma} P} \qquad (\text{T.APP})\ \frac{\Gamma \vdash A{:}\tilde{\tau} \quad \Gamma \vdash \tilde{e}{:}\tilde{\tau}}{\Gamma \vdash A(\tilde{e})}$$

$$(\text{T.DEF})\ \frac{\Gamma \vdash A : \tilde{\tau} \quad \Gamma, \tilde{x} : \tilde{\tau} \vdash P}{\Gamma \vdash A(\tilde{x} : \tilde{\tau}) \triangleq P} \qquad (\text{T.DEFS})\ \frac{\forall D \in \mathscr{D}.\Gamma \vdash D}{\Gamma \vdash \mathscr{D}}$$

Figure 3.6: Type system for $\pi(\mathscr{L})$ (continued).

tion.

Typing rules for communication prefixes (T.REC) and (T.SEND) derive directly from the above explanations of channel types. Rules for process application (T.APP) and definition (T.DEF) are similar to those for the π-calculus with priority in Figure 2.7. Typing rule (T.NEW) now checks explicitly that a new channel name is created and nothing else; previously all values were channel names. Finally, typing rules (T.PAR) and (T.SUM) remain as in Figure 2.7.

Proposition 7 (Type safety for expressions)**.** *The attribute language* $\lambda(\mathbb{N}_0,+,=)$ *in Figure 3.1 is type safe, i.e. if* $\Gamma \vdash e{:}\tau$ *and* $e \Downarrow v$ *then* $\Gamma \vdash v{:}\tau$.

The proof is standard and proceeds by induction on the proof of $\Gamma \vdash e{:}\tau$ and follows from a substitution lemma stating that if $\Gamma, x{:}\tau \vdash e{:}\sigma$ and $\Gamma \vdash v{:}\tau$ then $\Gamma \vdash e[v/x] : \sigma$.

Proposition 8 (Normalization)**.** *In the attribute language* $\lambda(\mathbb{N}_0,+,=)$ *every typable expression evaluates to some value, i.e., if* $\Gamma \vdash e{:}\tau$*, then there exists v such that* $e \Downarrow v$.

Typings of terms in $\lambda(\mathbb{N}_0,+,=)$ that make use of rule (T.CONST) always do so with a constant type (`Int`) for τ. Therefore, $\lambda(\mathbb{N}_0,+,=)$ is a simply-typed λ-calculus (e.g. fixed point operators are not typable) that is known to have the normalization property. A proof of this result by Tait's methods (Tait, 1967) can be found in many text books (see e.g. Mitchell (1996)).

Lemma 3. *The following properties hold for the typing rules of processes:*

1. *(strengthening) if* $\Gamma, x{:}\tau \vdash P$ *and* $x \notin fv(P)$ *then* $\Gamma \vdash P$,
2. *(weakening) if* $\Gamma \vdash P$ *and* $x \notin fv(P)$ *then* $\Gamma, x{:}\tau \vdash P$,
3. *(substitution) if* $\Gamma, x{:}\tau \vdash P$ *and* $\Gamma \vdash v{:}\tau$ *then* $\Gamma \vdash P[v/x]$,
4. *if* $\Gamma \vdash P$ *and* $P \equiv Q$ *then* $\Gamma \vdash Q$.

Strengthening and weakening also hold for the typing of definitions.

Proof. The proofs of the three first properties are straightforward inductions on the derivation of $\Gamma, x{:}\tau \vdash P$ (strengthening and substitution) and of $\Gamma \vdash P$ (weakening). They easily extend to (and depend on) the same properties for expressions. The proof of the last property is by induction on the definition of the structural congruence. The only interesting case is for scope extrusion, i.e., assuming $x \notin fv(Q)$, $\Gamma \vdash (\nu x){:}\tau(P \mid Q) \Leftrightarrow \Gamma \vdash (\nu x){:}\tau P \mid Q$.

($\Rightarrow$) Rules (T.NEW) and (T.PAR) imply that $\Gamma, x{:}\tau \vdash P$ and $\Gamma, x{:}\tau \vdash Q$. Since $x \notin fv(Q)$, by strengthening, it holds that $\Gamma \vdash Q$ and, by rule (T.NEW), it is true that $\Gamma \vdash (\nu x){:}\tau P$. Finally, rule (T.PAR) provide that $\Gamma \vdash (\nu x){:}\tau P \mid Q$.

($\Leftarrow$) Rules (T.PAR) and (T.NEW) imply that $\Gamma, x{:}\tau \vdash P$ and $\Gamma \vdash Q$. By weakening, it holds that $\Gamma, x{:}\tau \vdash Q$ and, by (T.PAR) and (T.NEW) it is also true that $\Gamma \vdash (\nu x){:}\tau(P \mid Q)$.

□

Lemma 4. *Let P be a process with definitions $\mathscr{D}$ in the attributed π-calculus with a type safe attribute language. If $\Gamma \vdash P$, $\Gamma \vdash \mathscr{D}$, and $P \Downarrow Q$ then $\Gamma \vdash Q$.*

Proof. By reduction rule (CONV), there exists $n \geq 0$, such that $P(\xrightarrow[nd]{app})^n Q$. Thus, the proof is by induction on n, i.e. if $\Gamma \vdash P$ and $P(\xrightarrow[nd]{app})^n Q$ then $\Gamma \vdash Q$. The case $n = 0$ is straightforward, so only the case $n = 1$ needs to be considered by induction on the derivation of $P \xrightarrow[nd]{app} Q$. The

induction cases (PAR) and (NEW) are straightforward and (STRUC) follows from Lemma 3(4). The case of rule (APP) yields $A(\tilde{e}) \xrightarrow[nd]{app} P[\tilde{v}/\tilde{e}]$ with $\tilde{e} \Downarrow \tilde{v}$ and $A(\tilde{x}:\tilde{\sigma}) \triangleq P$. Since $\Gamma \vdash A(\tilde{x}:\tilde{\sigma}) \triangleq P$, by rule (T.DEF), it holds that $\Gamma, \tilde{x}{:}\tilde{\sigma} \vdash P$ (†) and $\Gamma \vdash A{:}\tilde{\sigma}$. Moreover, by hypothesis, $\Gamma \vdash A(\tilde{e})$, thus $\Gamma \vdash \tilde{e}{:}\tilde{\sigma}$. Since $\tilde{e} \Downarrow \tilde{v}$, the type safety of attribute language yields $\Gamma \vdash \tilde{v}{:}\tilde{\sigma}$. Property (†) and substitution Lemma 3(3) yield $\Gamma \vdash P[\tilde{v}/\tilde{x}]$. □

Theorem 1 (Type safety for processes)**.** *If $\mathscr{L}$ is a type safe attribute language then $\pi(\mathscr{L})$ is type safe, i.e. for all processes P with definitions $\mathscr{D}$ in the attributed π-calculus, it holds that if $\Gamma \vdash P$, $\Gamma \vdash \mathscr{D}$, and $P \to Q$ then $\Gamma \vdash Q$.*

Proof. By reduction rule (PRIOR), there exists P' such that $P \Downarrow P'$ and $P' \xrightarrow[nd]{r} Q$ where $r \in R$. By Lemma 4, it is thus sufficient to prove the theorem for reduction $\xrightarrow[nd]{r}$ by induction on the derivation. The inductive cases of rules (PAR), (STRUC), and (NEW) are straightforward. The case of rule (COM) yields $P = e_1[e_2]?(\tilde{y}).P_1 + M_1 \mid e_1'[e_2']?(\tilde{e}).P_2 + M_2$, $Q = P_1[\tilde{v}/\tilde{y}] \mid P_2$, such that $e_1 \Downarrow x$, $e_1' \Downarrow x$, $e_2 \Downarrow v_2$, $e_2' \Downarrow v_2'$, $v_2 v_2' \Downarrow r$, and $\tilde{e} \Downarrow \tilde{v}$. Rules (T.PAR), (T.SUM), (T.REC) and (T.SEND) imply that $\Gamma \vdash e_1 : [\tau] \Rightarrow \tilde{\sigma}$ and $\Gamma \vdash e_1 : [\tau_1 \to \tau_2] \Rightarrow \tilde{\sigma'}$. Since $e_1 \Downarrow x$ and $e_1' \Downarrow x$, type safety of expressions ensures that x has the same type as e_1 and e_1'. Thus, it holds that $\tau = \tau_1 \to \tau_2$, $\tilde{\sigma'} = \tilde{\sigma}$, and $\Gamma \vdash x : [\tau_1 \to \tau_2] \Rightarrow \tilde{\sigma}$. Moreover, since $\Gamma \vdash \tilde{e}{:}\tilde{\sigma}$, type safety of expressions yields $\Gamma \vdash \tilde{v}{:}\tilde{\sigma}$. In addition, it holds that $\Gamma, \tilde{y}{:}\tilde{\sigma} \vdash P_1$ and, by the substitution Lemma 3(3), it is true that $\Gamma \vdash P_1[\tilde{v}/\tilde{y}]$. Finally, from $\Gamma \vdash P_2$ and rule (T.PAR), it

follows that $\Gamma \vdash P_1[\tilde{v}/\tilde{y}] \mid P_2$. □

Corollary 2 (Error freeness)**.** *If $\mathscr{L}$ is an attribute language that is both type safe and normalizing (see Propositions 7 and 8) then $\pi(\mathscr{L})$ is error free, i.e. for all processes P with definitions $\mathscr{D}$ in the attributed π-calculus, it holds that if $\Gamma \vdash P$, $\Gamma \vdash \mathscr{D}$, and $P \to^* Q$ then $\neg Q \xrightarrow[nd]{err} \bot$.*

The proof is elaborated in Appendix B.

3.2 Modeling Techniques and Biological Examples

In this section, the usefulness of the attributed π-calculus for the modeling of cell-biological systems is underlined by two example models. The models consider spatial aspects of Euglena's phototaxis and cooperative enhancement for gene regulation at the lambda switch. Furthermore, population- and species-based modeling in the attributed π-calculus is illustrated and it is shown how changes in global information can be broadcasted in individual-based models by using prioritized update protocols.

3.2.1 Spatial Aspects: Euglena's Phototaxis

In the following, a model is presented that considers simple spatial aspects of location dependent motion using the example of Euglena's phototaxis John et al. (2008a). A formal study on the possibility

of modeling systems with dynamic compartment structures in the attributed π-calculus is provided Section 3.3.2.

Engelmann (1882) describes Euglena as a single cell organism that lives in water, performs photosynthesis, and thus exhibits light-dependent motion. In inland water, depending on the brightness, it swims up and down in order to reach a zone with just the right amount of light. In the model presented here, the probability that an Euglena moves upwards is constant, since it always tries to reach regions with more light. However, in order to avoid too intense light, Euglena moves downwards whenever it receives light for photosynthesis, i.e. with a probability proportional to the light intensity of its current position. It is assumed that light photons travel top-down and that light intensity degrades exponentially with respect to the depth (repeated filtering).

Given a light source with initial intensity $I \in \mathbb{R}^+$ at depth 0, and a transparency factor for filtering $\sigma \in]0,1]$, this means that the light intensity at depth $d \in \mathbb{R}^+$ equals $\sigma^d * I$. The model comprises two light sources with initial intensities $\mathtt{I}_1$ and $\mathtt{I}_2$, such that the overall amount of light yields $\mathtt{I} = \mathtt{I}_1 + \mathtt{I}_2$. Furthermore, it assumes a rate constant $\mathtt{u} \in \mathbb{R}$ for upward motion.

Space is considered by discrete depth levels $\{0,\ldots,\mathtt{m}\}$ where level 0 denotes the surface and level $\mathtt{m} \in \mathbb{N}_0$ the ground. Euglena may move up and down in steps of exactly one level. Continuous depth levels and movement steps could be modeled similarly, but would increase simulation costs. The model's initial state provides $\mathtt{n} \in \mathbb{N}_0$ Euglenas on every level, summing up to totally $\mathtt{n} * (\mathtt{m}+1)$ Euglenas in the water.

The two light sources are located at the water surface. The probability of an interaction with a light source is proportional to its intensity. The values $\mathtt{u}, \sigma, \mathtt{m}, \mathtt{n}, \mathsf{l}_1, \mathsf{l}_2$ are model parameters.

The model is defined in $\pi(\lambda(\mathbb{R}, +, -, *, /, pow, \leq)_{<_1})$, the attributed π-calculus with constants for real numbers and the usual arithmetic operations. For convenience, x^y is written instead of $(pow\ x)\ y$. The successful values are the positive real numbers that all have the same level of priority. The big-step evaluator for these operators can be defined as usual (in analogy to natural numbers, see Section 3.1.2).

The model is presented in Figure 3.7. An Euglena at depth level d may interact with a light source of intensity i and go down by one level:

$$down : \mathsf{Euglena}(d), \mathsf{Light}(i) \xrightarrow{\sigma^d * i} \mathsf{Euglena}(d+1), \mathsf{Light}(i)$$

Such interactions happen on channel *down* with rate constant $\sigma^d * i$ under the condition that $d \leq m - 1$. An Euglena can also move up with rate constant $\mathtt{u}$ by interacting with a dummy interaction partner on channel *up*:

$$up : \mathsf{Euglena}(d) \xrightarrow{u} \mathsf{Euglena}(d-1)$$

If Euglena is at the surface, i.e. the constraint $d \geq 1$ is not satisfied, it cannot move upwards any further.

Based on the Master Equation, the numbers of Euglenas on each depth level in equilibrium can be computed. The Master Equation is a set of ODE's that, similar to a CTMC, describes a system's dynamics, see e.g. Reichl (2009). For each species a variable is introduced

Parameters

$\mathtt{n} \in \mathbb{N}_0$ // init number of E. per water level

$\mathtt{m} \in \mathbb{N}_0$ // deepest water level

$\mathsf{l}_1, \mathsf{l}_2 \in \mathbb{R}_+$ // intensity rates of light sources

$\sigma \in [0,1]$ // transparency of water

$\mathtt{u} \in \mathbb{R}_+$ // Euglena's upwards speed

Process definitions

$$\begin{aligned}
\mathsf{Euglena}(d) \triangleq\ & \mathit{up}[\lambda_.\mathbf{if}\ d \geq 1\ \mathbf{then}\ \mathtt{u} \\
& \qquad \mathbf{else}\ 0]?().\mathsf{Euglena}(d-1) \\
& + \mathit{down}[\lambda i.\mathbf{if}\ d \leq \mathtt{m}-1\ \mathbf{then}\ \sigma^d * i \\
& \qquad \mathbf{else}\ 0]?().\mathsf{Euglena}(d+1)
\end{aligned}$$

$$\mathsf{Light}(i) \triangleq \mathit{down}[i]!().\mathsf{Light}(i)$$

$$\mathsf{Dummy}() \triangleq \mathit{up}[\_]!().\mathsf{Dummy}()$$

Initial solution

$$\prod_{d=0}^{\mathtt{m}} \prod_{i=1}^{\mathtt{n}} \mathsf{Euglena}(d) \mid \mathsf{Light}(\mathsf{l}_1) \mid \mathsf{Light}(\mathsf{l}_2) \mid \mathsf{Dummy}()$$

Figure 3.7: A model of Euglena's light-dependent motion with two light sources.

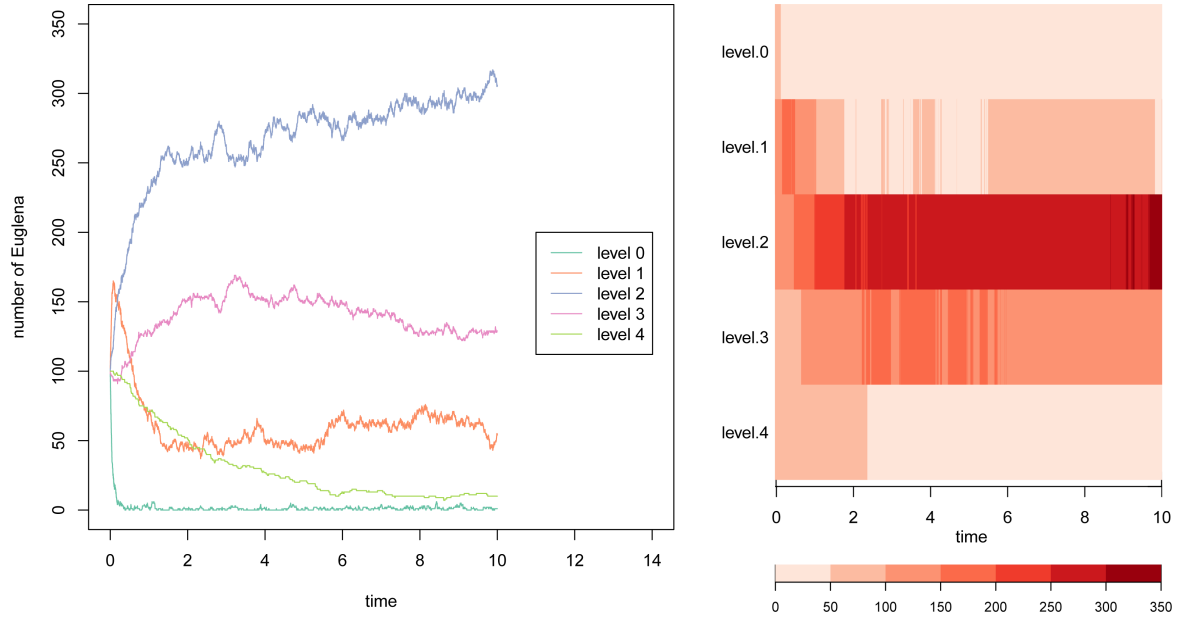
and a differential equation describes the change of the variable's value over time. Equilibrium happens if the system is free of change, i.e. the derivation of every variable is 0. For illustration, consider a model with maximum depth level $m = 4$. Let $l_0, \ldots, l_4$ be the number of Euglenas per level. The master equation provided in Figure 3.8. The first equation $-\sigma^0 * \mathtt{I} * l_0 + u * l_1 = 0 = dl_0'/dt$ states that the change in the number of Euglenas at level 0 is obtained by summing up a loss of $\sigma^0 * \mathtt{I} * l_0$ due to Euglena's downward motion to level 1, and a gain of $u * l_1$ due to Euglena's upward motion from level 1. The last equation $\sum_{i=0}^{4} l_i = 5n$ denotes that the overall number of Euglenas is constant and equals the initial number.

In order to verify the behavior of the model with respect to predictions obtained from the Master Equation, two simulation experiments were performed, named A and B. There are constantly five depth levels ($\mathtt{m} = 4$), 100 Euglenas on each depth level ($\mathtt{n} = 100$), a rate constant of upward motion $\mathtt{u} = 0.4$, intensity rate constants $\mathtt{I}_1 = 5.0, \mathtt{I}_2 = 15.0$ ($\mathtt{I} = 20.0$), and transparency factors $\sigma = 0.1$ in experiment A and $\sigma = 0.2$ in experiment B. Each experiment consists of a single simulation run, both of them performed until simulation time $t = 10.0$.

The simulation results are presented in Figures 3.9, 3.10. Heat maps and line charts show the number of Euglenas on each depth level over time. Below them, the solutions of the Master Equation with the respective model parameters are given. The simulation results confirm the predictions with slight derivations due to stochasticity. The comparison of both experiments shows that with a higher transparency Euglenas accumulate on a deeper level, since more light is available.

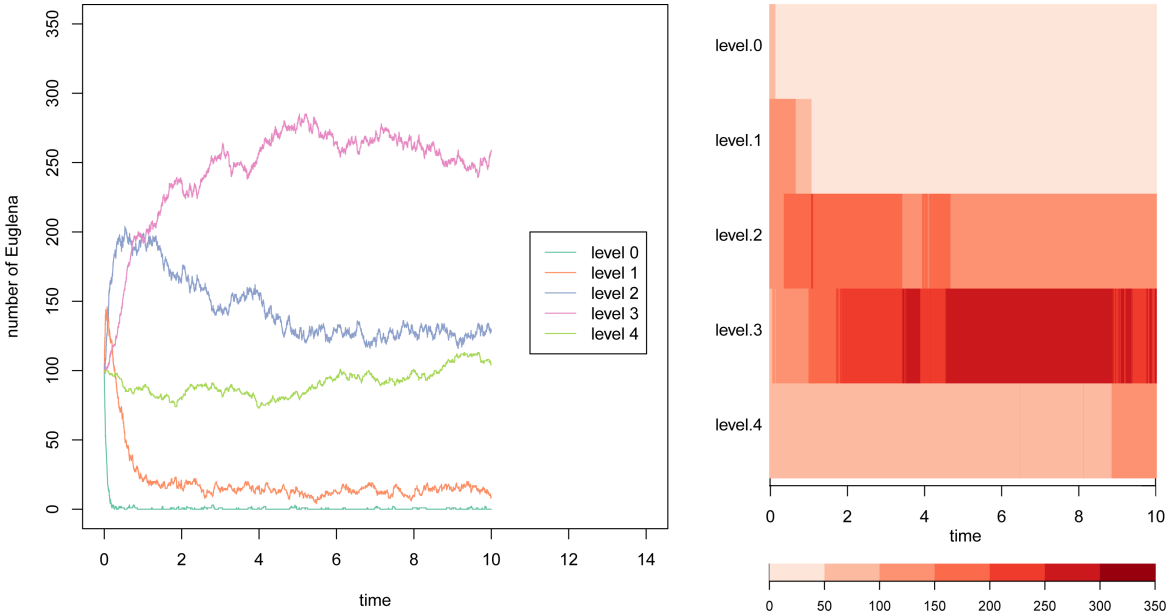
$$\begin{pmatrix} -\sigma^0 * \mathtt{I} & u & 0 & 0 & 0 \\ \sigma^0 * \mathtt{I} & -(\sigma^1 * \mathtt{I} + u) & u & 0 & 0 \\ 0 & \sigma^1 * \mathtt{I} & -(\sigma^2 * \mathtt{I} + u) & u & 0 \\ 0 & 0 & \sigma^2 * \mathtt{I} & -(\sigma^3 * \mathtt{I} + u) & u \\ 0 & 0 & 0 & \sigma^3 * \mathtt{I} & -u \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} l_0 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 5n \end{pmatrix}$$

Figure 3.8: Master Equation to compute the numbers of Euglenas on each depth level in equilibrium.



Experiment A, Predictions: $l_0 = 1.16$, $l_1 = 57.84$, $l_2 = 289.20$, $l_3 = 144.65$, $l_4 = 7.15$

Figure 3.9: Euglena model, Experiment A with $\mathtt{m} = 4$, $\mathtt{n} = 100$, $\mathtt{u} = 0.4$, $\mathtt{l}_1 = 5.0$, $\mathtt{l}_2 = 15.0$ ($\mathtt{I} = 20.0$), and $\sigma = 0.1$. A line chart and a heat map show the Euglena numbers on each depth level versus time for a single simulation run until time $t = 10.0$. The numbers at equilibrium for the different depth levels as obtained by solving the Master Equation are shown below.



Experiment B, Predictions: $l_0 = 0.26$, $l_1 = 12.81$, $l_2 = 128.14$, $l_3 = 256.28$, $l_4 = 102.51$

Figure 3.10: Euglena model, Experiment B that differs from A only in setting $\sigma = 0.2$.

The Euglena model with attributed processes can be translated into the stochastic π-calculus without attributes, since all parameters are finitely valued. The idea is to duplicate the *down*-channels for all depth levels, such that their rate constants are chosen dependent on the depth. This leads to processes $\mathsf{Euglena}_d()$, $\mathsf{Light}_{1,d}()$, and $\mathsf{Light}_{2,d}()$ for all possible depth levels, see Fig. 3.11.

3.2.2 Cooperative Enhancement: Gene Regulation at the Lambda Switch

Cooperative binding is a frequent and often decisive aspect in gene regulatory networks, where proteins stabilize each other's binding to neighboring DNA sites by adhesive contacts. In quantitative terms, the unbinding rate of one DNA-protein complex decreases by the existence of another. This is an instance of cooperative enhancement of reaction rates by third partners. Kuttler et al. (2007) and Kuttler and Niehren (2006) showed that cooperative enhancement can be modeled in the stochastic π-calculus. It however requires nontrivial encodings that can be alleviated within the attributed π-calculus.

A well understood instance of cooperative binding occurs during transcription initiation control at the λ switch. The λ switch is a segment of the DNA of bacteriophage λ. It contains two binding sites OR_1 and OR_2, where rep and cro proteins can bind. An unstable binding of a rep molecule to OR_2 is stabilized by the simultaneous presence of another rep at the neighboring site OR_1. As illustrated in Figure 3.12, the two proteins actually touch each other.

// Euglenas on different depth levels

$\mathsf{Euglena}_0() \triangleq \mathit{down}_0?().\mathsf{Euglena}_1()$

$\mathsf{Euglena}_1() \triangleq \mathit{up}?().\mathsf{Euglena}_0()$

$\quad + \mathit{down}_1?().\mathsf{Euglena}_2()$

...

$\mathsf{Euglena}_{\mathtt{m}}() \triangleq \mathit{up}?().\mathsf{Euglena}_{\mathtt{m\text{-}1}}()$

// light from first source on different levels

$\mathsf{Light}_{1,0}() \triangleq \mathit{down}_0 : \sigma^0 * \mathsf{l}_1!().\mathsf{Light}_{1,0}()$

...

$\mathsf{Light}_{1,\mathtt{m}}() \triangleq \mathit{down}_m : \sigma^{\mathtt{m}} * \mathsf{l}_1!().\mathsf{Light}_{1,\mathtt{m}}()$

// light from second source on different levels

$\mathsf{Light}_{2,0}() \triangleq \mathit{down}_0 : \sigma^0 * \mathsf{l}_2!().\mathsf{Light}_{2,0}()$

...

$\mathsf{Light}_{2,\mathtt{m}}() \triangleq \mathit{down}_m : \sigma^{\mathtt{m}} * \mathsf{l}_2!().\mathsf{Light}_{2,\mathtt{m}}()$

$\mathsf{Dummy}() \triangleq \mathit{up} : \mathtt{u}!().\mathsf{Dummy}()$

Example solution

$\prod_{d=0}^{\mathtt{m}} \left(\prod_{i=0}^{\mathtt{n}} \mathsf{Euglena}_d() \mid \mathsf{Light}_{1,d}() \mid \mathsf{Light}_{2,d}()\right)$

Figure 3.11: An equivalent model of Euglena in the stochastic π-calculus.

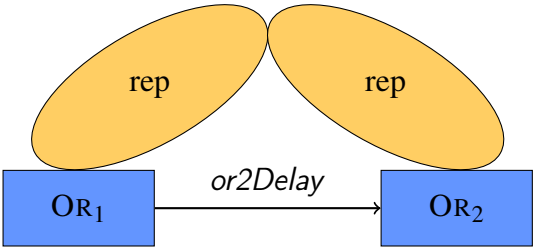


Figure 3.12: The decay of the rep-OR_2 complex: in order to make the decay rate of the rep-OR_2 complex dependent on OR_1's state, the two sites communicate on *or2Delay* before OR_2 unbinds.

A model of cooperative binding at OR_2 in $\pi(\lambda(\mathbb{R}_+^\infty, \mathtt{Prot}, =)_{<_2})$ is presented in Figure 3.13. It contains the parametric definition $\mathsf{Prot}(\mathit{type})$, which emulates the behavior of the proteins. The parameter *type* can be instantiated by constants `rep` or `cro`, introduced by `Prot` to model either protein sort. Proteins can bind to both sites OR_1 and OR_2. Free sites are defined by processes $\mathrm{OR}_1()$ and $\mathrm{OR}_2()$, where proteins can attach via channel *bind*. As this occurs, the channel *release* is created and henceforth connects the protein to the site (complexation). Later communication on *release* breaks the complex. The rate constant of complexation is fixed to `0.098`. For decomplexation the rate constant is determined by the sender, i.e. the binding site, the receiving protein accepts it by applying the identity function.

For illustration, consider the models for the protein bound DNA sites. $\mathrm{OR_iB}(\mathit{type}, \mathit{release})$ describes the unbinding from the occupied site $\mathrm{OR_i}$, where *type* indicates the type of the bound protein. For $i = 1$ the rate of the unbinding reaction merely depends on the protein type.

Process definitions

$$\mathsf{Prot}(\mathit{type}) \triangleq (\nu \mathit{release})\, \mathit{bind}[\_]!(\mathit{type}, \mathit{release}).\ \mathit{release}[\lambda r.\, r]?().\, \mathsf{Prot}(\mathit{type})$$

$$\mathrm{OR}_1() \triangleq \mathit{bind}[\lambda\_.\, 0.098]?(\mathit{type}, \mathit{release}).\ \mathrm{OR}_{1\mathrm{B}}(\mathit{type},\mathit{release}) + \mathit{or2Delay}[\mathit{free}]!.\, \mathrm{OR}_1()$$

$$\mathrm{OR}_{1\mathrm{B}}(\mathit{type},\mathit{release}) \triangleq \mathit{release}[\ \mathbf{if}\ \mathit{type} = \mathtt{rep}\ \mathbf{then}\ 0.155\ \mathbf{else}\ \mathbf{if}\ \mathit{type} = \mathtt{cro}\ \mathbf{then}\ 2.45\ \mathbf{else}\ 0]!().\, \mathrm{OR}_1() + \mathit{or2Delay}[\mathit{type}]!().\, \mathrm{OR}_{1\mathrm{B}}(\mathit{type},\mathit{release})$$

$$\mathrm{OR}_2() \triangleq \mathit{bind}[\lambda\_.\, 0.098]?(\mathit{type}, \mathit{release}).\ \mathrm{OR}_{2\mathrm{B}}(\mathit{type},\mathit{release})$$

$$\mathrm{OR}_{2\mathrm{B}}(\mathit{type},\mathit{release}) \triangleq$$

```
or2delay[λt.
  if type = rep then
    if t = rep then 0.155 // cooperative
    else 3.99 // OR1 bound to cro or free
  else 2.45 // bound to cro
  ]?().release[∞]!().OR2()
```

Example process

$$\mathrm{OR}_1() \mid \mathrm{OR}_2() \mid \prod_{i=1}^{28} \mathsf{Prot}(\mathtt{rep}) \mid \prod_{i=1}^{67} \mathsf{Prot}(\mathtt{cro})$$

Figure 3.13: A model of cooperative binding between OR_1 and OR_2 at the λ switch.

For the second site ($i = 2$) decomplexation is influenced by cooperative binding. To model this, OR_1 and OR_2 are linked via the channel *or2Delay*, as illustrated in Figure 3.12. Additionally, the release operation is decomposed into an interaction on channel *or2Delay*, with a reaction rate defining the actual unbinding delay, and an immediate communication on *release*. As stated in the definition of the global channel *or2Delay* the unbinding delay depends not only on the type of the bound protein, but also on the state of OR_1, which can be either *free*, bound to `rep` or bound to `cro`.

The model of Kuttler and Niehren (2006) in the stochastic π-calculus requires to keep OR_2 constantly informed about state changes of OR_1, which is implemented by immediate communication steps (priority). Keeping state information consistent in this manner is error-prone, it may easily lead to deadlocks. The subsequent model of Kuttler et al. (2007) in SPiCO requires significantly fewer updates. In $\pi(\lambda(\mathbb{R}_+^\infty, \texttt{Prot}, =)_{<_2})$, rate constants directly depend on the attribute values of the interaction partners. State changes are propagated without additional communication steps.

3.2.3 Population-based Modeling

The stochastic π-calculus supports individual-based modeling where molecules are mapped to objects. The attributed π-calculus enables, in addition, a population-based modeling style, where reactions are mapped to objects.

For illustration, consider a chemical system with three species A, B,

and C and the following two reactions with rate constants k_1 and k_2:

$$r_1 : A + B \xrightarrow{k_1} C \qquad r_2 : B + C \xrightarrow{k_2} A$$

Figure 3.14 shows a population-based description of this system in $\pi(\lambda(\mathbb{R}, +, -, *, /, pow, \leq)_{<_1})$. The model parameters $\mathsf{a}_0, \mathsf{b}_0, \mathsf{c}_0 \in \mathbb{N}_0$ represent the initial amounts of the three species. A process $\mathsf{Reac}(f, d_a, d_b, d_c)$ defines a reaction with kinetic function f, while the other parameters d_a, d_b, and d_c reflect how the reaction affects the population, i.e. the differences in the amounts of the species. The latter parameters could also be regarded as stoichiometric factors, except that reactants are always associated with negative numbers. The kinetics of all reactions in this example follow the law of Mass action. For instance, the kinetics of r_1 yields the product of the amounts of species A and B and rate constant k_1. It changes the population by decreasing the amount of A and B by one each and increasing the amount of C by one. Thus, to represent reaction r_1, the initial solution comprises a process $\mathsf{Reac}(\lambda a.\lambda b.\lambda c.a * b * \mathsf{k}_1, -1, -1, 1)$, where the function parameters a, b, c represent the amounts of A, B, C, respectively. Consequently, also one process $\mathsf{Reac}(\lambda a.\lambda b.\lambda c.b * c * \mathsf{k}_2, 1, -1, -1)$ is introduced to model reaction r_2. Notice, that it is also possible to account for different kinetic laws and different stoichiometric factors. In Section 4.2.2, this idea is used to implement a reaction with Michaelis-Menten kinetics. For valid models, however, one needs to ensure that the applied kinetic laws satisfy the Markov property, since the stochastic semantics is defined in terms of CTMC's. The restriction that only reactions with at most two reactants and Mass action kinetics

can be implemented in the attributed π-calculus in an individual-based style remains valid.

In order to represent populations, processes $\mathsf{Pop}(a,b,c)$ are introduced, whose parameters stand for the amounts of the three species. In addition to the two reactions, the initial process thus also comprises $\mathsf{Pop}(\mathsf{a}_0,\mathsf{b}_0,\mathsf{c}_0)$. Interactions on channel r indicate the occurrence of a reaction. For the computation of reaction kinetics $\mathsf{Reac}(f,d_a,d_b,d_c)$ provides its kinetics function f as the constraint argument. $\mathsf{Pop}(a,b,c)$ defines a constraint function, which applies f to the current amounts of the species. When an interaction occurs, $\mathsf{Reac}(f,d_a,d_b,d_c)$ sends its population changes and $\mathsf{Pop}(a,b,c)$ applies them to the current amounts, which is implemented by recursively calling $\mathsf{Pop}(a+d_a,b+d_b,c+d_c)$. Afterward, the next reaction can happen. Function f evaluates to 0 whenever one of the populations becomes 0.

The model can be generalized to systems in which new species can be created dynamically by using property lists as parameters, where each element contains a pair of a species and its amount. Even new reactions could be dynamically introduced. Such a model is close to the way cell-biology is expressed in sCCP, pointing to the possibility of generally encoding sCCP in the attributed π-calculus. Notice, however, that with process $\mathsf{Pop}(a,b,c)$ a central unit is introduced, essentially violating the basic idea of compositional implementations. This can, however, be fixed by distributing the information about species numbers to the different reactions and using more sophisticated prioritized update protocols. In Section 4.2.2, an alternative is presented which avoids priority by making use of the global imperative store of

Parameters

$\mathsf{a}_0, \mathsf{b}_0, \mathsf{c}_0 \in \mathbb{N}_0$ // initial amounts of A, B, and C

$\mathsf{k}_1, \mathsf{k}_2 \in \mathbb{R}$ // reaction rate constants

Process definitions

$$\mathsf{Reac}(f, d_a, d_b, d_c) \triangleq r[f]!(d_a, d_b, d_c).\ \mathsf{Reac}(f, d_a, d_b, d_c)$$

$$\mathsf{Pop}(a, b, c) \triangleq r[\lambda f.(f\ \ a\ \ b\ \ c)]?(d_a, d_b, d_c).\ \mathsf{Pop}(a + d_a, b + d_b, c + d_c)$$

Example solution

$$\mathsf{Reac}(\lambda a.\lambda b.\lambda c.\, a * b * \mathsf{k}_1, -1, -1, 1) \mid \mathsf{Reac}(\lambda a.\lambda b.\lambda c.\, b * c * \mathsf{k}_2, 1, -1, -1) \mid \mathsf{Pop}(\mathsf{a}_0, \mathsf{b}_0, \mathsf{c}_0)$$

Figure 3.14: A population-based model of three species and two reactions in $\pi(\lambda(\mathbb{R}, +, -, *, /, pow, \leq)_{<_1})$. Process $\mathsf{Reac}(f, d_a, d_b, d_c)$ defines reactions, where parameter f is a function reflecting the reaction kinetics and parameters d_a, d_b, d_c account for the way species amounts are changed when the reaction occurs. Process $\mathsf{Pop}(a, b, c)$ reflects species amounts and interacts with process $\mathsf{Reac}(f, d_a, d_b, d_c)$ for reaction execution.

the imperative π-calculus. Since this idea is even closer to sCCP, a discussion between the differences of the imperative π-calculus and sCCP is provided.

3.2.4 Global Information in Individual-Based Modeling

Versari (2009) showed that priority is useful in order to track global information in individual-based models in a consistent manner, e.g. to model changes in compartment structures. The general idea is to propagate changes globally by a sequence of prioritized local interactions, before enabling the next possible reactions, since these are given lower priority.

In the example in Figure 3.15, global information on population sizes is traced, i.e. numbers of individuals. The model rephrases the population-based model in Section 3.2.3 in an individual-based way. Molecules of the three species are represented by processes $\mathsf{A}()$,$\mathsf{B}()$, and $\mathsf{C}()$, a process $\mathsf{Pop}(a,b,c)$ is defined, such that it tracks molecule numbers. Chemical reactions are modeled by interactions on channel r, where $\mathsf{A}()$ or $\mathsf{C}()$ send to $\mathsf{B}()$ with rate constants k_1 and k_2, respectively. Changes in populations are updated by prioritized interactions on channel u once a reaction occurs. Process $\mathsf{Pop}(a,b,c)$ receives such changes with infinite rate, i.e. with priority, such that no reaction can occur before the population information is updated. This ensures that the effect of each reaction is correctly reflected in the species amounts, i.e. that the population information is consistent.

Parameters

$\mathsf{a}_0, \mathsf{b}_0, \mathsf{c}_0 \in \mathbb{N}_0$ // initial amounts of A, B, and C

$\mathsf{k}_1, \mathsf{k}_2 \in \mathbb{R}$ // reaction rate constants

Process definitions

$\mathsf{A}() \triangleq r[\mathsf{k}_1]!() . u[\_]!(-1,-1,1) . \mathsf{C}()$

$\mathsf{B}() \triangleq r[\lambda k.k]?() . \mathbf{0}$

$\mathsf{C}() \triangleq r[\mathsf{k}_2]!() . u[\_]!(1,-1,-1) . \mathsf{A}()$

$\mathsf{Pop}(a,b,c) \triangleq u[\lambda\_.\infty]?(d_a,d_b,d_c) .$
$\mathsf{Pop}(a+d_a, b+d_b, c+d_c)$

Initial solution

$\prod_{i=1}^{\mathsf{a}_0} \mathsf{A}() \mid \prod_{i=1}^{\mathsf{b}_0} \mathsf{B}() \mid \prod_{i=1}^{\mathsf{c}_0} \mathsf{C}() \mid \mathsf{Pop}(\mathsf{a}_0, \mathsf{b}_0, \mathsf{c}_0)$

Figure 3.15: An individual-based variant of the population-based model in Figure 3.14. Processes A(),B(), and C() represent molecules of different species, process Pop(a,b,c) accounts for molecule numbers, which are updated by prioritized interactions on channel u.

3.2.5 Species-Based Modeling

As a last example, it is shown how one can rephrase the model given in Section 3.2.3 in a species-based style as it is introduced by Bio-PEPA, see Figure 3.16. The model makes use of priority (i.e. immediate communications) and the fact that reactions have at most two reactants. A process $\mathsf{A}(a)$ represents the species A which is attributed by the number a of molecules of A. Species B and C are implemented analogously. In contrast to the individual-based model, the solution contains only a single process for each species.

Reaction r_1 is modeled as an interaction between processes $\mathsf{A}(a)$ and $\mathsf{B}(b)$ on channel r_1. The corresponding communication constraint yields a successful value, which follows the definition of reaction rates given by Mass action kinetics. Again, other kinetics could be introduced, as long as they respect the Markov property. After an interaction, $\mathsf{A}(a - 1)$ and $\mathsf{B}(b - 1)$ are called recursively, thus decreasing the number of molecules of species A and B. In parallel, a request is sent with priority on channel u_c in order to increase the number of molecules of species C. Reaction r_1 is implemented analogously.

3.3 Expressiveness

In this section, it is shown that the attributed π-calculus provides a unifying framework that generalizes on various dialects of the π-calculus in the literature. Furthermore, an encoding from $\pi[@, \neq]$ to the attributed π-calculus is presented. The work of Versari (2009) implies

Parameters

$\mathsf{a}_0, \mathsf{b}_0, \mathsf{c}_0 \in \mathbb{N}_0$ // initial amounts of A, B, and C

$\mathsf{k}_1, \mathsf{k}_2 \in \mathbb{R}$ // reaction rate constants

Process definitions

$$\begin{aligned}
\mathsf{A}(a) &\triangleq r_1[a]!().(\mathsf{A}(a-1) \mid u_c[_]!(1)) \\
&\quad + u_a[\lambda_.\infty]?(d).\mathsf{A}(a+d) \\
\mathsf{B}(b) &\triangleq r_1[\lambda a.\mathsf{k}_1 * a * b]?().\mathsf{B}(b-1) \\
&\quad + r_2[\lambda c.\mathsf{k}_2 * c * b]?().\mathsf{B}(b-1) \\
\mathsf{C}(c) &\triangleq r_2[c]!().(\mathsf{C}(c-1) \mid u_a[_]!(1)) \\
&\quad + u_c[\lambda_.\infty]?(d).\mathsf{C}(c+d)
\end{aligned}$$

Initial solution

$\mathsf{A}(\mathsf{a}_0) \mid \mathsf{B}(\mathsf{b}_0) \mid \mathsf{C}(\mathsf{c}_0)$

Figure 3.16: A species-based variant of the population-based model in Figure 3.14 in $\pi(\lambda(\mathbb{R}, +, -, *, /, pow, \leq)_{<_1})$. Processes $\mathsf{A}(a)$, $\mathsf{B}(b)$, and $\mathsf{C}(c)$ represent species parametrized by their multiplicities possibly updated through communication on channel u_a and u_c.

$$
\begin{array}{rclcrcl}
[\![x?(\tilde{y}).P]\!] & = & x[\lambda z.z]?(\tilde{y}).[\![P]\!] & & [\![P_1 \mid P_2]\!] & = & [\![P_1]\!] \mid [\![P_2]\!] \\
[\![x{:}r!(\tilde{y}).P]\!] & = & x[r]!(\tilde{y}).[\![P]\!] & & [\![M_1 + M_2]\!] & = & [\![M_1]\!] + [\![M_2]\!] \\
[\![(\nu x)P]\!] & = & (\nu x)[\![P]\!] & & [\![A(\tilde{x}) \triangleq P]\!] & = & A(\tilde{x}) \triangleq [\![P]\!] \\
[\![\mathbf{0}]\!] & = & \mathbf{0} & & & &
\end{array}
$$

Figure 3.17: The encoding of the π-calculus with priorities $(R,<)$ into $\pi(\lambda(R)_<)$.

that $\pi[@,\neq]$ allows for the modeling of dynamic cell structures. Since the presented encoding is compositional, it shows that the attributed π-calculus is sufficiently expressive for the spatial modeling of cell-biological systems.

3.3.1 Encoding of the π-Calculus with Priority

In the following, an encoding of the π-calculus with priority is presented and proved to be correct with respect to both semantics - the non-deterministic and the stochastic. It is also shown that the encoding can be refined such that it preserves well-typedness.

The translation of the π-calculus with priority levels in $(R,<)$ into $\pi(\lambda(R)_<)$ is given in Figure 3.17. Senders $x{:}r!(\tilde{y}).P$ are mapped to $x[r]!(\tilde{y}).P$ and receivers $x?(\tilde{y}).P$ to $x[\lambda z.z]?(\tilde{y}).P$.

Theorem 2. *The encoding of the π-calculus with priority levels $(R,<)$ into the attributed π-calculus, $\pi(\lambda(R)_<)$, is correct in that for all processes P,P' it holds that:*

1. *if $P \to P'$ then $[\![P]\!] \to [\![P']\!]$*
2. *if $[\![P]\!] \to Q$ then there exists a process $\hat{Q}$ of the π-calculus with priority, such that $[\![\hat{Q}]\!] \equiv Q$ and $P \to \hat{Q}$*

The proof is elaborated in Appendix B. It is mostly straightforward, but covers several pages since all rules of both calculi must be inspected in detail.

The same encoding is also correct with respect to the stochastic operational semantics, under the assumption that the set of stochastic rates $(\mathbb{R}^\infty_+, <_2)$ is chosen as the set of priorities.

Theorem 3. *The encoding of the π-calculus with priority levels in $(\mathbb{R}^\infty_+, <_2)$ into the attributed π-calculus, $\pi(\lambda(\mathbb{R}^\infty_+)_{<_2})$, is correct with respect to the stochastic operational semantics. For all processes $P,P',\hat{P}$, attributed processes Q, and labels $\beta \in \{r, \infty(n) \mid r \in \mathbb{R}_+, n \in \mathbb{N}\}$ it holds that:*

1. *if $P \xrightarrow{\beta} P'$ then $[\![P]\!] \xrightarrow{\beta} [\![P']\!]$*
2. *if $[\![\hat{P}]\!] \xrightarrow{\beta} Q$ then there exists a process $\hat{Q}$, such that $\hat{P} \xrightarrow{\beta} \hat{Q}$ and $[\![\hat{Q}]\!] \equiv Q$.*

Proof. The stochastic semantics of both calculi are built upon their non-deterministic semantics. In the appendix (see the proof of The-

orem 2) it is shown that the translation is invariant under substitution and that it reflects and preserves the structural congruence and errors. Furthermore, it is shown that if $P\sigma Q$ then $[\![P]\!]\sigma[\![Q]\!]$, for $\sigma \in \{\Downarrow, \rightarrow\} \cup \{\xrightarrow[nd]{\alpha} \mid \alpha \in \{app\} \cup R\}$.

Claim. Relation $\xrightarrow[\ell]{r}$ is preserved and reflected by translation (positions ℓ of redexes remain unchanged), i.e.:

1. if $P \xrightarrow[\ell]{r} Q$ then $[\![P]\!] \xrightarrow[\ell]{r} [\![Q]\!]$
2. if $[\![\hat{P}]\!] \xrightarrow[\ell]{r} Q$ then there exists $\hat{Q}$, such that $\hat{P} \xrightarrow[\ell]{r} \hat{Q}$ and $[\![\hat{Q}]\!] \equiv Q$

Proof.

1. If $P \xrightarrow[\ell]{r} Q$ then rule (COM$_\ell$) can be applied as follows, where $P = (\nu\tilde{x}) \prod_{i=1}^{n} \sum_{j=1}^{m_i} \pi_i^j.P_i^j$ and $Q = (\nu\tilde{x})(\prod_{i=1, i\neq i_1, i_2}^{n} \sum_{j=1}^{m_i} \pi_i^j.P_i^j \mid P_{i_1}^{j_1}[\tilde{v}/\tilde{y}] \mid P_{i_2}^{j_2})$:

$$\frac{\ell = (i_1, j_1, i_2, j_2) \quad \pi_{i_1}^{j_1} = x?(\tilde{y}) \quad \pi_{i_2}^{j_2} = x{:}r!(\tilde{z}) \quad |\tilde{y}| = |\tilde{v}|}{P \xrightarrow[\ell]{r} Q}$$

   Thus, it holds that $[\![P]\!] = (\nu\tilde{x}) \prod_{i=1}^{n} \sum_{j=1}^{m_i} [\![\pi_i^j]\!].[\![P_i^j]\!]$, with $[\![\pi_{i_1}^{j_1}]\!] = x[\lambda y.y]?(\tilde{y})$ and $[\![\pi_{i_2}^{j_2}]\!] = x[r]!(\tilde{z})$. Now, rules (VAL) and (FUN) provide that rule (COM$_\ell$) of $\pi(\mathscr{L})$ applies to the translations in the following way, where $[\![P]\!] = (\nu\tilde{x}) \prod_{i=1}^{n} \sum_{j=1}^{m_i} [\![\pi_i^j]\!].[\![P_i^j]\!]$ and $[\![Q]\!] = (\nu\tilde{x})(\prod_{i=1, i\neq i_1, i_2}^{n} \sum_{j=1}^{m_i} [\![\pi_i^j]\!].[\![P_i^j]\!] \mid [\![P_{i_1}^{j_1}[\tilde{v}/\tilde{y}]]\!] \mid [\![P_{i_2}^{j_2}]\!])$:

$$\frac{\begin{array}{c}[\![\pi_{i_1}^{j_1}]\!] \Downarrow x[\lambda y.y]?(\tilde{y}) \quad [\![\pi_{i_2}^{j_2}]\!] \Downarrow x[r]!(\tilde{z}) \\ \ell = (i_1, j_1, i_2, j_2) \quad (\lambda y.y)r \Downarrow r \in \mathbb{R}_+^\infty \quad |\tilde{y}| = |\tilde{z}|\end{array}}{[\![P]\!] \xrightarrow[\ell]{r} [\![Q]\!]}$$

The substitution claim provides that $[\![P_{i_1}^{j_1}[\tilde{v}/\tilde{y}]]\!] = [\![P_{i_1}^{j_1}]\!][\tilde{v}/\tilde{y}]$, such that $[\![Q]\!] = (\nu\tilde{x})(\prod_{i=1,i\neq i_1,i_2}^{n}\sum_{j=1}^{m_i}[\![\pi_i^j]\!].[\![P_i^j]\!] \mid [\![P_{i_1}^{j_1}]\!][\tilde{v}/\tilde{y}] \mid [\![P_{i_2}^{j_2}]\!])$.

2. If $[\![\hat{P}]\!] \xrightarrow[\ell]{r} Q$ then rule (COM$_\ell$) must be applicable as follows, where $[\![\hat{P}]\!] = (\nu\tilde{x})\prod_{i=1}^{n}\sum_{j=1}^{m_i}\pi_i^j.P_i^j$ and $Q = (\nu\tilde{x})(\prod_{i=1,i\neq i_1,i_2}^{n}\sum_{j=1}^{m_i}\pi_i^j.P_i^j \mid P_{i_1}^{j_1}[\tilde{v}/\tilde{y}] \mid P_{i_2}^{j_2})$:

$$\frac{\pi_{i_1}^{j_1} \Downarrow x[v_1]?(\tilde{y}) \quad \pi_{i_2}^{j_2} \Downarrow x[v_2]!(\tilde{v}) \quad \ell = (i_1,j_1,i_2,j_2) \quad v_1v_2 \Downarrow r \in \mathbb{R}_+^\infty \quad |\tilde{y}| = |\tilde{v}|}{[\![\hat{P}]\!] \xrightarrow[\ell]{r} Q}$$

Since the translation is compositional, process $\hat{P}$ must have the form $\hat{P} = (\nu\tilde{x})\prod_{i=1}^{n}\sum_{j=1}^{m_i}\hat{\pi}_i^j.\hat{P}_i^j$, with $[\![\hat{\pi}_i^j]\!] = \pi_i^j$ and $[\![\hat{P}_i^j]\!] = P_i^j$. Furthermore, it is true that $v_1 = \lambda y.y$, $v_2 = r$, such that $\hat{\pi}_{i_1}^{j_1} = x?(\tilde{y})$ and $\hat{\pi}_{i_2}^{j_2} = x{:}r!(\tilde{z})$, with $\tilde{v} = \tilde{z}$. Let $\hat{Q} = (\nu\tilde{x})(\prod_{i=1,i\neq i_1,i_2}^{n}\sum_{j=1}^{m_i}\hat{\pi}_i^j.\hat{P}_i^j \mid \hat{P}_{i_1}^{j_1}[\tilde{v}/\tilde{y}] \mid \hat{P}_{i_2}^{j_2})$. Since the translation is substitution invariant, it holds that $[\![\hat{Q}]\!] = Q$. Thus, rule (COM$_\ell$) applies as follows, where $\hat{P} = (\nu\tilde{x})\prod_{i=1}^{n}\sum_{j=1}^{m_i}\hat{\pi}_i^j.\hat{P}_i^j$ and $\hat{Q} = (\nu\tilde{x})(\prod_{i=1,i\neq i_1,i_2}^{n}\sum_{j=1}^{m_i}\hat{\pi}_i^j.\hat{P}_i^j \mid \hat{P}_{i_1}^{j_1}[\tilde{v}/\tilde{y}] \mid \hat{P}_{i_2}^{j_2})$:

$$\frac{\ell = (i_1,j_1,i_2,j_2) \quad \hat{\pi}_{i_1}^{j_1} = x?(\tilde{y}) \quad \hat{\pi}_{i_2}^{j_2} = x{:}r!(\tilde{z}) \quad |\tilde{y}| = |\tilde{z}|}{\hat{P} \xrightarrow[\ell]{r} \hat{Q}}$$

Given two processes P,Q, a set $I(P,Q) \subseteq \mathbb{R}_+^\infty \times \mathbb{N}^4$ and a number $S(P,Q) \in \mathbb{R}_+^\infty$ as used in rule (SUM) can be defined as follows:

$$I(P,Q) = \{(r,\ell) \mid \exists Q'.\, P \xrightarrow[\ell]{r} Q' \equiv Q\} \quad \text{and} \quad S(P,Q) = \sum_{(r,\ell)\in I(P,Q)} r$$

Claim. $S(P,Q) = S([\![P]\!],[\![Q]\!])$

Proof. It is sufficient to prove that $I(P,Q) = I([\![P]\!],[\![Q]\!])$. There are two inclusions to be shown:

"$\subseteq$" If $(r,\ell) \in I(P,Q)$ then there exists Q', such that $P \xrightarrow[\ell]{r} Q' \equiv Q$. The first part of the previous claim shows that $[\![P]\!] \xrightarrow[\ell]{r} [\![Q']\!]$, and since the translation preserves structural congruence also $[\![Q']\!] \equiv [\![Q]\!]$. Hence, it holds that $(r,\ell) \in I([\![P]\!],[\![Q]\!])$.

"$\supseteq$" If $(r,\ell) \in I([\![P]\!],[\![Q]\!])$ then there exists Q'', such that $[\![P]\!] \xrightarrow[\ell]{r} Q'' \equiv [\![Q]\!]$. The second part of the previous claim shows that there exists Q', such that $P \xrightarrow[\ell]{r} Q'$, with $[\![Q']\!] \equiv Q'' \equiv [\![Q]\!]$. This implies that $Q' \equiv Q$, since translation reflects structural congruence. Thus, it is true that $(r,\ell) \in I(P,Q)$.

Claim. Let Q and $P_1 \equiv P_2$ be processes. If P_1 and P_2 are prenex normal forms in which all bound variables are named distinctly then $S(P_1,Q) = S(P_2,Q)$.

Proof. Suppose that $P_1 = (\nu x_1)\ldots(\nu x_k)\prod_{i=1}^{m}\sum_{j=1}^{n_i} M_i^j$ for guarded processes M_i^j. An analysis of the structural congruence shows that there exists a sequence of variables $(y_1,\ldots,y_k)$ and permutations $\sigma : \{1,\ldots,k\} \to \{1,\ldots,k\}$, $\theta : \{1,\ldots,m\} \to \{1,\ldots,m\}$, and $\theta_i : \{1,\ldots,n_i\} \to \{1,\ldots,n_i\}$, such that:

$$P_2 = (\nu y_{\sigma(1)})\ldots(\nu y_{\sigma(k)})\prod_{i=1}^{m}\sum_{j=1}^{n_i} M'^{\theta_i(j)}_{\theta(i)} \text{ and}$$
$$M_i^j \equiv M'^j_i[y_{\sigma(1)}/x_1,\ldots,y_{\sigma(k)}/x_k]$$

Given this representation of P_2 and since all bound variables are named distinctly, it is easy to check that $(r,(\theta(i_1),\theta_{i_1}(j_1),\theta(i_2),\theta_{i_2}(j_2))) \in I(P_1,Q)$, iff $(r,(i_1,j_1,i_2,j_2)) \in I(P_2,Q)$.

In the following the theorem for reductions with finite rates is proved.

Claim. The translation preserves and reflects relations $\xrightarrow{r}$ for all $r \in \mathbb{R}_+$, i.e.:

1. if $P \xrightarrow{r} P'$ then $[\![P]\!] \xrightarrow{r} [\![P']\!]$
2. if $[\![\hat{P}]\!] \xrightarrow{r} Q$ then there exists $\hat{Q}$, such that $\hat{P} \xrightarrow{r} \hat{Q}$ and $[\![\hat{Q}]\!] \equiv Q$

Proof.

1. Assumption $P \xrightarrow{r} Q$ must be inferred by rule (SUM) as follows:

$$\frac{P \Downarrow P_1 \quad S(P_1,Q) = r \neq 0 \quad \neg\exists\ell\exists Q'.P_1 \xrightarrow[\ell]{\infty} Q'}{P \xrightarrow{r} Q}$$

   In the proof of Theorem 2, it is shown that $P \Downarrow P_1$ implies $[\![P]\!] \Downarrow [\![P_1]\!]$. The second claim above shows that $S(P_1,Q) = S([\![P_1]\!],[\![Q]\!])$. The second part of the first claim above ensures that $\neg\exists\ell\exists Q'.[\![P_1]\!] \xrightarrow[\ell]{\infty} Q'$. Thus, the following rule is applicable:

$$\frac{[\![P]\!] \Downarrow [\![P_1]\!] \quad S([\![P_1]\!],[\![Q]\!]) = r \neq 0 \quad \neg\exists\ell\exists Q'.[\![P_1]\!] \xrightarrow[\ell]{\infty} Q'}{[\![P]\!] \xrightarrow{r} [\![Q]\!]}$$

2. By assumption it holds that $[\![\hat{P}]\!] \xrightarrow{r} Q$ for $r \in \mathbb{R}_+$. Since the stochastic semantics refines the non-deterministic semantics by

Proposition 6, it is true that $[\![\hat{P}]\!] \to Q$. Theorem 2 on the preservation of the non-deterministic semantics shows that there exists a process $\hat{Q}$, such that $[\![\hat{Q}]\!] \equiv Q$ and $\hat{P} \to \hat{Q}$. The following only makes use of $[\![\hat{Q}]\!] \equiv Q$. Assumption $[\![\hat{P}]\!] \xrightarrow{r} Q$ must be inferred by rule (SUM) in the following way:

$$\frac{[\![\hat{P}]\!] \Downarrow P_1 \quad S(P_1, Q) = r \neq 0 \quad \neg\exists\ell\exists Q'.P_1 \xrightarrow[\ell]{\infty} Q'}{[\![\hat{P}]\!] \xrightarrow{r} Q}$$

In particular, P_1 must be in prenex normal form, such that w.l.o.g. all its bound variables are named distinctly. Since $[\![\hat{P}]\!] \Downarrow P_1$, there exists $\hat{P}_1$, such that $\hat{P} \Downarrow \hat{P}_1$ and $[\![\hat{P}_1]\!] \equiv P_1$, as it is was shown in the proof of Theorem 2. Process $\hat{P}_1$ is a prenex normal form, such that w.l.o.g. all its bound variables can be assumed to be named distinctly. The above claims show that:

$$S(P_1, Q) = S([\![\hat{P}_1]\!], [\![\hat{Q}]\!]) = S(\hat{P}_1, \hat{Q})$$

Since the translation reflects $\xrightarrow[\ell]{\infty}$ steps, rule (SUM) can be applied as follows:

$$\frac{\hat{P} \Downarrow \hat{P}_1 \quad S(\hat{P}_1, \hat{Q}) = r \neq 0 \quad \neg\exists\ell\exists Q'.\hat{P}_1 \xrightarrow[\ell]{\infty} Q'}{\hat{P} \xrightarrow{r} \hat{Q}}$$

Claim. The translation preserves and reflects immediate reactions, i.e.:

1. if $P \xrightarrow{\infty(n)} P'$ then $[\![P]\!] \xrightarrow{\infty(n)} [\![P']\!]$
2. if $[\![\hat{P}]\!] \xrightarrow{\infty(n)} Q$ then exists $\hat{Q}$, such that $\hat{P} \xrightarrow{\infty(n)} \hat{Q}$ and $[\![\hat{Q}]\!] \equiv Q$

The proof of this claim shall be omitted here. It concerns rule (COUNT), which can be treated quite similarly to rule (SUM) above:

□

The translation can be refined such that types are preserved. In order to do so, it is assumed that there exists a type constant R by which to type priority levels $r \in R$ during translation. The translation for restriction and output prefixes is refined as follows:

$$[\![(\nu x{:}\tau)P]\!] = (\nu x{:}[\![\tau]\!])[\![P]\!]$$
$$[\![x{:}r!(\tilde{y}).P]\!] = x[r_R]!(\tilde{y}).[\![P]\!]$$

Types of the π-calculus with priority are translated to types of $\pi(\lambda(R)_<)$:

$$[\![ch(\tau_1,\ldots,\tau_n)]\!] = [R \to R] \Rightarrow ([\![\tau_1]\!],\ldots,[\![\tau_n]\!])$$

Proposition 9 (Type preservation)**.** *Let P be a process of the π-calculus with priority and Γ a type environment such that $\Gamma \vdash P$ then $[\![\Gamma]\!] \vdash [\![P]\!]$.*

The proof is straightforward by structural induction over type derivations.

3.3.2 Encoding $\pi[@,\neq]$ for Dynamic Compartments

Versari (2009) proposed an encoding of BioAmbients into $\pi@$ to show that $\pi@$ is sufficient expressive to model dynamic cell structures. The syntax of $\pi@$ is the same as for the π-calculus with priority, except

that communication now acts on non-empty tuples of channels and that priority levels are assigned to both senders and receivers. This means that prefixes now have the following form, where $|\tilde{x}| \geq 1$:

$$\text{polyadic prefixes} \quad \pi ::= \tilde{x}?(r)\tilde{y} \mid \tilde{x}{:}r!(\tilde{z})$$

The communication rule (COM) is adapted, such that tuples of channels and priority levels are tested for equality before communication. Otherwise, the non-deterministic semantics of the π-calculus with priority remains unchanged:

$$(\text{COM}_{@})\ \frac{|\tilde{y}| = |\tilde{z}|}{\tilde{x} : r?(\tilde{y}).P_1 + M_1 \mid \tilde{x}{:}r!(\tilde{z}).P_2 + M_2 \xrightarrow[nd]{r} P_1[\tilde{z}/\tilde{y}] \mid P_2}$$

However, as it was discovered during the preparation of this thesis, there is no obvious way to encode process location as in BioAmbients to polyadic synchronization. The reason is that an encoding of interactions between processes located in two ambients sharing the same parent (siblings), i.e. `s2s` communication and `enter` and `merge` rearrangements, requires not only checking channel equality but also channel inequality. Consider e.g. the following BioAmbients process:

$$P = [[\texttt{s2s}\ x?(\tilde{y}).P_1] \mid [\texttt{s2s}\ x{:}r!(\tilde{z}).P_2]]$$

According to Versari (2009), the encoding to $\pi@$ proceeds by introducing channels for all existing ambients, here e.g. channels a_1, a_2, and p to represent the ambients containing the receiver and the sender and their parent ambient, respectively. Additional channels are needed

to denote communication directions, such as *s2s* for `s2s` communication. The obtained channels are then used to encode location dependent interaction into polyadic synchronization yielding the following encoding of P:

$$[\![P]\!] = s2s@x@p : r?(\tilde{y}).[\![P_1]\!] \mid s2s@x@p{:}r!(\tilde{z}).P_2 \quad \textit{faulty encoding}$$

In words, the encoding allows all processes to perform an `s2s` communication if their surrounding ambients share the same parent. However, this also includes processes of the form that, according to the semantics of BioAmbients, clearly may not communicate:

$$P' = [[\texttt{s2s}\ x?(\tilde{y}).P_1 \mid \texttt{s2s}\ x{:}r!(\tilde{z}).P_2]] \quad \textit{counter example}$$

The same holds true for `enter` and `merge` rearrangements. To encode such BioAmbients prefixes it is thus necessary to check the inequality of the ambients directly surrounding senders and receivers, i.e. in the example that $a_1 \neq a_2$, which is not possible in polyadic synchronization.

The problem can be fixed by introducing $\pi[@,\neq]$, which extends polyadic synchronization in the following way: to each receiver an additional tuple, $\tilde{b}$, of constants `true` and `false` is annotated. Output prefixes remain the same:

$$\text{extended polyadic prefixes} \qquad \pi ::= \tilde{x} : \tilde{b} : r?(\tilde{y}) \mid \tilde{x}{:}r!(\tilde{z})$$

The tuple $\tilde{b}$ denotes the matching result necessary for two processes to interact, i.e. which channels should equal (`true`) and which should

differ (`false`) in order for a receiver and a sender to perform a communication. The channel tuples and the matching result tuple must be of equal length:

$$(\mathrm{COM}_{[@,\neq]}) \quad \frac{|\tilde{y}| = |\tilde{z}| \quad \forall i \in \{1,\ldots,n\}.(x_i = x'_i) \Leftrightarrow (b_i = \texttt{true})}{(x_i)_{i=1}^n : (b_i)_{i=1}^n : r?(\tilde{y}).P_1 + M_1 \mid (x'_i)_{i=1}^n : r!(\tilde{z}).P_2 + M_2 \xrightarrow[nd]{r} P_1[\tilde{z}/\tilde{y}] \mid P_2}$$

Based on the extended version of polyadic synchronization, BioAmbients process P' can be encoded in the following way, where a identifies the ambient directly surrounding the sender and the receiver and no communication steps are possible:

$$[\![P']\!] = s2s@x@a@p : (\texttt{true},\texttt{true},\texttt{false},\texttt{true}) : r?(\tilde{y}).[\![P_1]\!] \mid s2s@x@a@p : r!(\tilde{z}).P_2$$

corrected encoding

In the following, a compositional encoding of $\pi[@,\neq]$ with priority levels in an ordered set $(R,<)$ into the attributed π-calculus, $\pi(\lambda(R,=,\mathtt{EQ})_<)$, is presented and proved to be correct. This result shows that the attributed π-calculus inherits the correct encoding of BioAmbients from $\pi[@,\neq]$ and is thus sufficiently expressive for the modeling of dynamic cell structures.

The encoding of $\pi[@,\neq]$ is decomposed into two parts. The first part is a preprocessing step that rewrites all tuples in sending or receiving positions, such that they are of the same arity. Given a process P of $\pi[@,\neq]$, let n be the maximal arity of tuples in subject position of polyadic prefixes and x a fresh channel name not occurring in P

(which exists since *Vars* is infinite). Sending and receiving tuples in P are completed by x's until they are of arity n. Similarly, the tuples of Booleans of receiver prefix are filled up with value `true`:

$$\begin{array}{rcl}(x_1,\ldots,x_m) & \Rightarrow & (x_1,\ldots,x_m,\underbrace{x,\ldots,x}_{n-m})\\ (b_1,\ldots,b_m) & \Rightarrow & (x_1,\ldots,x_m,\underbrace{\texttt{true},\ldots,\texttt{true}}_{n-m})\end{array}$$

In order to encode processes in $\pi[@,\neq]$ with channel tuples in subject positions of a maximal arity n, the attribute language $\lambda(R,=,\texttt{EQ})_<$ provides functions $\texttt{eq}_n \in \texttt{EQ}$, which check [in-]equality of n-tuples of constants or variables (and thus channel names):

$$\begin{array}{lcl}\texttt{eq}_0 & =_{df} & \texttt{true}\\ \texttt{eq}_n & =_{df} & \lambda x_1\ldots\lambda x_n\lambda y_1\ldots y_n\lambda b_1\ldots b_n.\\ & & \quad \textsf{if}(x_n = y_n) = b_n \textsf{ then}\\ & & \quad\quad \texttt{eq}_{n-1}\ x_1\ldots x_{n-1}y_1\ldots y_{n-1}b_1\ldots b_{n-1}\\ & & \quad \textsf{else } \texttt{false}\end{array}$$

Lemma 5. *For all constants or variables* $v_1,\ldots,v_n$, $v'_1,\ldots,v'_n$, $b_1,\ldots,b_n$ *it is true that:*

1. $eq_n v_1\ldots v_n v'_1\ldots v'_n b_1\ldots b_n \Downarrow true$, *if* $\forall i \in \{1,\ldots,n\}.(v_i = v'_i) \Leftrightarrow (b_i = true)$

2. $eq_n v_1\ldots v_n v'_1\ldots v'_n \Downarrow false$, *if* $\exists i \in \{1,\ldots,n\}.\neg((v_i = v'_i) \Leftrightarrow (b_i = true))$

The proof is straightforward by induction on n. It relies on the definition of conditionals and equality of the big-step evaluator in Figures 3.1 and 3.2. See Appendix B for details.

The main translation $[\![\_]\!] : \pi[@,\neq] \to \pi(\lambda(R,=,\mathtt{EQ})_<)$ maps to the attributed π-calculus with an attribute language that provides additional constants for priority levels and equality. Only priority levels are successful values. Since the encoding $[\![\_]\!]$ is compositional, only the mapping of communication prefixes needs to be specified. Therefore, it is assumed that all subject tuples have the same arity n. A single fresh channel x not occurring in P is introduced at the subject position of all encoded prefixes:

$$\begin{aligned}
[\![(x_i)_{i=1}^n{:}r!(\tilde{z}).P]\!] &= x[\lambda x_1 \ldots \lambda x_n \lambda r \lambda b_1 \ldots \lambda b_n. \\
&\qquad \textsf{if } \mathtt{eq}_{n+1} x_1 \ldots x_n r\, x_1 \ldots x_n r \\
&\qquad\qquad b_1 \ldots b_n \mathtt{true} \textsf{ then } r \\
&\qquad \textsf{else } \mathtt{false}]!\tilde{z}.[\![P]\!] \\
[\![(x_i)_{i=1}^n : (b_i)_{i=1}^n : r?(\tilde{z}).P]\!] &= x[\lambda e.e\, x_1 \ldots x_n r\, b_1 \ldots b_n]?(\tilde{z}).[\![P]\!]
\end{aligned}$$

A sender on channel tuple $(x_1,\ldots,x_n)$ with priority level r is translated to a sender on a single channel x. Its constraint argument is defined to be a function with parameters $x_1,\ldots,x_n$, $b_1,\ldots,b_n$, and r, denoting the channel tuple, the expected matching result, and the priority level of the receiver. Pairwise equality of channels and priority levels are checked with function $\mathtt{eq}_{n+1}$, testing if for all $i \in \{1,\ldots,n\}$ it holds that $(x_i = x_i) = b_i$ and that $(r = r) = \mathtt{true}$. Consequently, a receiver on channel tuple $(x_1,\ldots,x_n)$, with a tuple of Booleans $(b_1,\ldots,b_n)$, and priority level r is translated to a receiver on a single channel x with a

constraint function that applies the functional constraint argument to channels $x_1, \ldots, x_n$, Booleans $b_1, \ldots, b_n$ and priority level r. All process definitions need to be translated in the same way:

$$[\![A(\tilde{x}) \triangleq P]\!] = A(\tilde{x}) \triangleq [\![P]\!]$$

Theorem 4. *The encoding of $\pi[@,\neq]$ with priority levels in $(R,<)$ to the attributed π-calculus, $\pi(\lambda(R,=,\mathtt{EQ})_<)$, is correct, in that all preprocessed processes P in $\pi[@,\neq]$ satisfy:*

1. *if $P \to Q$ then $[\![P]\!] \to [\![Q]\!]$*
2. *if $[\![P]\!] \to Q$ then there exists $\hat{Q}$, such that $Q \equiv [\![\hat{Q}]\!]$ and $P \to \hat{Q}$*

The proof is elaborated in Appendix B. It checks that communication steps correspond in both calculi, i.e. that the polyadic synchronization of $\pi[@,\neq]$ is translated properly to [in-]equality testing in $\pi(\lambda(R,=,\mathtt{EQ})_<)$. This mostly follows from Lemma 5 on the correctness of encoding equality of n-tuples.

Finally, notice that the encoding of $\pi[@,\neq]$ does not preserve types in any obvious sense. Finding a convincing type system for $\pi[@,\neq]$ (and also $\pi@$) is nontrivial, since there the capabilities of tuples and channels are overloaded while usual type systems separate tuples and channel types properly.

3.3.3 Variants of the Stochastic Pi-Calculus

It remains to discuss the relationship to variants of the stochastic π-calculus where rates are annotated to channels.

BioSpi and SPiM

The syntax of BioSpi (Regev, 2003) and SPiM Phillips and Cardelli (2007) differ from that of $\pi(\mathscr{L})$ in that stochastic rates are annotated to channels at creation time, rather than to communication prefixes. The rates of the prefixes can then be deduced from the rate of the communicating channel.

The idea of encoding this variant of the stochastic π-calculus into $\pi(\lambda(\mathbb{R}_+^\infty)_{<_2})$ is to replace channels x with rate r by pairs $\langle x,r\rangle$, that are decomposed at communication time. Here it is relevant that the attributed π-calculus permits pairs and that it allows for expressions in sender and receiver positions.

Below, a formal representation of the encoding is presented, which is claimed to be correct with respect to both semantics, nondeterministic and stochastic (without proof):

assumption: all bound variables in P are named distinctly

$$
\begin{aligned}
[\![P]\!]_1 &= [\![P[\langle x_1,r_1\rangle/x_1]\ldots[\langle x_n,r_n\rangle/x_n]]\!]_1, \\
&\qquad \text{with } fv(P)=\{x_1,\ldots,x_n\}, \\
&\qquad\qquad \forall i\in\{1,\ldots,n\}.r_i \text{ is rate of } x_i \\
[\![A(\tilde{y}) \triangleq P]\!]_1 &= A(\tilde{y}) \triangleq [\![P[\langle x_1,r_1\rangle/x_1]\ldots[\langle x_n,r_n\rangle/x_n]]\!]_1, \\
&\qquad \text{with } fv(A(\tilde{y}) \triangleq P)=\{x_1,\ldots,x_n\}, \\
&\qquad\qquad \forall i\in\{1,\ldots,n\}.r_i \text{ is rate of } x_i \\
[\![(\nu x{:}r)P]\!]_2 &= (\nu x)[\![P[\langle x,r\rangle/x]]\!]_2 \\
[\![x?(\tilde{y}).P]\!]_2 &= (\mathtt{fst}\ x)[\lambda z.z]?(\tilde{y}).[\![P]\!]_2 \\
[\![x!(\tilde{y}).P]\!]_2 &= (\mathtt{fst}\ x)[\mathtt{snd}\ x]!(\tilde{y}).[\![P]\!]_2 \\
&\ldots
\end{aligned}
$$

The encoding defines a two-step approach, first replacing all free names by pairs in P and its definitions in $\mathscr{D}$ as implemented by encoding $[\![\cdot]\!]_1$ and then all names bound by ν-operators ($[\![\cdot]\!]_2$). The two last lines of the definition of $[\![\cdot]\!]_2$ state that the channel is extracted from the pair before communication and that the rate is extracted in the communication constraint. The definitions of the encoding's second step for defined processes, parallel compositions, summations, and idle processes are straightforward and thus omitted here. The encoding can only work if the bound variables are named distinctly, since otherwise names are incorrectly replaced by pairs.

Stochastic Pi-Calculus with Concurrent Objects

The stochastic π-calculus with concurrent objects (SPiCO) supports a static form of polyadic synchronization, called pattern guarded inputs. Patterns are tuples $a(\tilde{y})$ that are built from a finite set of function symbols a in some set Σ and a sequence of channels. Senders transmit tuples $b(\tilde{z})$ to receivers, which match it against a pattern $a(\tilde{y})$. A communication step is allowed only if the function symbol b of the sent tuple matches the function symbol a of the receiving pattern:

$$x?a(\tilde{y}).P \mid x!b(\tilde{z}).P' \rightarrow P[\tilde{z}/\tilde{y}] \mid P', \text{ if } a = b$$

The communication constraint is thus equality $a{=}b$. This is a weak form of polyadic synchronization, additionally checking function symbols. As before, stochastic rates are annotated to channels at creation time.

Kuttler et al. (2007) showed that SPiCO can be encoded in the stochastic π-calculus and by this, following Theorem 3, also in $\pi(\mathscr{L})$. A more direct encoding from SPiCO to $\pi(\lambda(\mathbb{R}_+^\infty, \Sigma, =))_{<_2}$ can be obtained similarly as for SPiM and BioSPi, where $a, b \in \Sigma$:

assumption: all bound variables in P are named distinctly

$$
\begin{array}{rcl}
[\![P]\!]_1 & = & [\![P[\langle x_1, r_1\rangle/x_1]\ldots[\langle x_n, r_n\rangle/x_n]]\!]_1, \\
 & & \quad \text{with } fv(P) = \{x_1, \ldots, x_n\}, \\
 & & \quad\quad \forall i \in \{1, \ldots, n\}.r_i \text{ is rate of } x_i \\
[\![A(\tilde{y}) \triangleq P]\!]_1 & = & A(\tilde{y}) \triangleq [\![P[\langle x_1, r_1\rangle/x_1]\ldots[\langle x_n, r_n\rangle/x_n]]\!]_1, \\
 & & \quad \text{with } fv(A(\tilde{y}) \triangleq P) = \{x_1, \ldots, x_n\}, \\
 & & \quad\quad \forall i \in \{1, \ldots, n\}.r_i \text{ is rate of } x_i \\
[\![(\nu x{:}r)P]\!]_2 & = & (\nu x)[\![P[\langle x, r\rangle/x]]\!]_2 \\
[\![x?a(\tilde{y}).P]\!]_2 & = & (\texttt{fst}\ x)[\lambda z.\textsf{if } z = a \textsf{ then } (\texttt{snd}\ x) \\
 & & \quad\quad \textsf{else } 0.0]?(\tilde{y}).[\![P]\!]_2 \\
[\![x!(\tilde{y}).P]\!]_2 & = & (\texttt{fst}\ x)[\texttt{snd}\ x]!(\tilde{y}).[\![P]\!]_2 \\
 & \ldots &
\end{array}
$$

The only new aspect here is to check the communication constraint $a{=}b$ in addition.

Original Attributed Pi-Calculus

A preliminary version of the attributed π-calculus (John et al., 2008b) annotates stochastic rate constants to channels, and a fixed function `val` is used to obtain the rate constants of channels. This version of

$\pi(\mathscr{L})$ can be encoded into the version of $\pi(\mathscr{L})$ presented here:

assumption: all bound variables in P are named distinctly

$$
\begin{aligned}
[\![P]\!]_1 &= [\![P[\langle x_1, r_1\rangle/x_1]\ldots[\langle x_n, r_n\rangle/x_n]]\!]_1, \\
&\qquad \text{with } fv(P) = \{x_1,\ldots,x_n\}, \\
&\qquad\qquad \forall i \in \{1,\ldots,n\}.r_i \text{ is rate of } x_i \\
[\![A(\tilde{y}) \triangleq P]\!]_1 &= A(\tilde{y}) \triangleq [\![P[\langle x_1, r_1\rangle/x_1]\ldots[\langle x_n, r_n\rangle/x_n]]\!]_1, \\
&\qquad \text{with } fv(A(\tilde{y}) \triangleq P) = \{x_1,\ldots,x_n\}, \\
&\qquad\qquad \forall i \in \{1,\ldots,n\}.r_i \text{ is rate of } x_i \\
[\![v[e]?(\tilde{v}).P]\!]_2 &= (\texttt{fst}\ [\![v]\!])[[\![e]\!]]?([\![\tilde{v}]\!]).[\![P]\!]_2 \\
[\![v[e]!(\tilde{y}).P]\!]_2 &= (\texttt{fst}\ [\![v]\!])[[\![e]\!]]!(\tilde{y}).[\![P]\!]_2 \\
[\![x]\!] &= x \\
[\![\texttt{val}]\!] &= \texttt{snd} \\
[\![\lambda x.e]\!] &= \lambda x.[\![e]\!] \\
[\![e_1 e_2]\!] &= [\![e_1]\!][\![e_2]\!] \\
&\ldots
\end{aligned}
$$

3.4 Stochastic Simulator

In this section, a stochastic simulation algorithm is developed that closely follows the stochastic semantics of the attributed π-calculus in terms of CTMC's. Thereby, it is shown that a simulator for $\pi(\mathscr{L})$ can be obtained independently of the choice of $\mathscr{L}$ by extending previous simulators for the stochastic π-calculus or SPiCO.

The stochastic semantics of $\pi(\mathscr{L})$ induces the naive stochastic simulator given in Figure 3.18. The simulator's input comprises a process

P and a time point $t \in \mathbb{R}$. The next reduction step for process P is chosen in a memoryless stochastic manner. The sojourn time $\Delta \in \mathbb{R}_+$ of P is inferred and the simulator proceeds with the resulting solution at time point $t + \Delta$. This loop continues until no next reduction step can be found; in fact it may run for ever, if not interrupted externally or equipped with some additional termination condition.

The first step of the simulation algorithm is to apply definitions of P exhaustively. This computation may run into an infinite loop or raise errors in case of malformed definitions or if the evaluation of some expressions diverges ($\neg\exists v.e \Downarrow v$). If application raises an immediate error $P_1 \xrightarrow[nd]{err} \bot$ by rules (E.COM), (E.PREF), or (E.CONSTR), then the simulator throws an exception (which kills its continuation). Note that error checking may run into infinite loops or raise an error, too. If P does converge to an error-free process P_1 then P_1 is uniquely determined up to structural congruence (Proposition 5) and must be congruent to some prenex normal form $(\nu\tilde{x})\prod_{i=0}^{n} M_i$. The remainder of the algorithm is independent of the concrete representative of congruence class $[P_1]_{\equiv}$, such that it can be chosen arbitrarily. The next step is to compute the set of all labeled reactions of P_1:

$$Reacts = \{(\ell, r) \in \mathbb{N}^4 \times \mathbb{R}_+^\infty \mid \exists P_2.\ P_1 \xrightarrow[\ell]{r} P_2\}$$

Labeled reactions with rate $r = \infty$ are executed with priority and without time consumption. If no reaction with rate $r = \infty$ exists, the SSA is applied to select a reaction $(\ell, r) \in Reacts$ with probability r/s where $s = \sum_{(\ell,r') \in Reacts} r'$. The sojourn time in P is $\Delta = -ln(1/U)/s$ for some uniformly distributed random number $0 < U \leq 1$.

Simulate−naive(P,t)

// process P, time point $t \in \mathbb{R}$

let P_1 **such that** $P \Downarrow P_1$

// P_1 *obtained from P by exhaustively applying definitions*

// computation may diverge

if $P_1 \xrightarrow[nd]{err} \bot$ **then raise error**

// apply all rules (E.COM), (E.PREF), (E.CONSTR).

// computation may diverge since expressions are evaluated

let $Reacts = \{(\ell, r) \in \mathbb{N}^4 \times \mathbb{R}_+^\infty\} \mid \exists P_2.\ P_1 \xrightarrow[\ell]{r} P_2\}$ *//* (COM$_\ell$)

if $Reacts \cap (\mathbb{N}_4 \times \{\infty\}) = \emptyset$ **then**

  let $((\ell, r), \Delta) = \mathtt{SSA}(Reacts)$ *//* (SUM)

  let P_2 **such that** $P_1 \xrightarrow[\ell]{r} P_2$

  Simulate−naive$(P_2, t+\Delta)$

else

  //(COUNT)

  select $(\ell, \infty) \in Reacts$ *with equal probability*

  let P_2 **such that** $P_1 \xrightarrow[\ell]{\infty} P_2$

  Simulate−naive(P_2, t)

Figure 3.18: Naive simulator interpreting the stochastic semantics of $\pi(\mathscr{L})$.

In order to compute *Reacts*, all possible instances of the communication rule (COM_ℓ) have to be enumerated. This requires evaluating all communication constraints by applying the evaluation algorithm of the attribute language $\mathscr{L}$.

Fortunately, the CTMC itself does not need to be computed by the simulation algorithm. This would be largely unfeasible, since the number of possible outcomes of non-deterministic interactions may grow exponentially. Furthermore, it would require deciding structural congruence (rules (SUM) and (COUNT)), which is a graph isomorphism complete problem, as shown by Khomenko and Meyer (2008), yielding high computational costs.

The efficiency of the naive simulation algorithm can be increased by applying an idea that was basically exploited already in the BioSpi implementation. The objective is to avoid the enumeration of all pairs of alternatives (and thus redexes), since there may be quadratically many in the size of P_1. The strategy is to *group* all reactions on the same channel with the same constraint argument and the same rate constant. The SSA is first applied to such *grouped reactions* and then a specific interaction is chosen with equal distribution.

Group labels allow the identification of grouped reactions. A *group label* of a process P_1 is a triple in $fv(P_1) \times Vals(P_1)^2$. The group of reactions for $P_1 = \prod_{i=1}^{n} \sum_{j=1}^{m} \pi_i^j.P_i^j$ with label $L = (x, v, r)$ is defined as follows:

$$Reacts(L) = \{((i_1, j_1, i_2, j_2), r) \in Reacts \mid \exists v' \tilde{y} \tilde{v}. \pi_{i_1}^{j_1} \Downarrow x[v]!(\tilde{v}),\ \pi_{i_2}^{j_2} \Downarrow x[v']?(\tilde{y}),\ v'v \Downarrow r\}$$

A triple L identifies reaction groups by a communication channel x, a constraint argument v, and a rate constant r yielding the application of the constraint function v' to v. The propensity of a grouping label L, $prop(L) \in \mathbb{R}^+ \uplus \{\infty(n) \mid n \in \mathbb{N}\}$, sums up all rate constants of the labeled reactions that are grouped together or counts the number of labels of infinite rate reactions if there are any:

$$prop(L) = \begin{cases} \infty(n), \text{ if } n = \#\{\ell \mid (\ell, \infty) \in \mathit{Reacts}(L)\} \geq 1 \\ \sum_{(\ell,r) \in \mathit{Reacts}(L)} r, \text{ otherwise} \end{cases}$$

The set of grouped reactions with their propensities, which forms the input of the SSA, is defined as follows:

$$\mathit{GReacts} = \{(L, prop(L)) \mid L \in \mathit{Vars}(P_1) \times \mathit{Vals}(P_1)^2\}$$

The cardinality of *GReacts* is linear in the size of P_1. In many practically relevant cases, only a fixed number of values will ever be used. This is e.g. the case in the example models of Euglena's movement and cooperative enhancement, where none of the processes succeeding the initial solution introduces new channels or new constraint values, see Section 3.2. By contrast, the cardinality of set *Reacts* becomes quadratic in the size of P_1, e.g., if all senders and receivers may interact.

Figure 3.19 provides a simulation algorithm based on grouped reactions. In contrast to the naive simulator, it first selects a grouped reaction based on the SSA and then a label of a reaction within this group with equal distribution.

Simulate (P,t) *// solution P, time point* $t \in \mathbb{R}$

let P_1 be **such that** $P \Downarrow P_1$

// P_1 *is obtained from P by exhaustively applying definitions.*

// computation may diverge

if $P_1 \xrightarrow[nd]{err} \bot$ **then raise error**

// apply all rules (E.COM), (E.PREF), (E.CONSTR).

// computation may diverge since expressions are evaluated

let $GReacts = \{(L, prop(L)) \mid L \in Vars(P_1) \times Vals(P_1)^2\}$

if $\{(L,r) \in GReacts \mid r = \infty(n)\} = \emptyset$ **then**

  let $((L,r),\Delta) = \mathtt{SSA}(GReacts)$

  select $(\ell,r) \in Reacts(L)$ *with equal probability*

  let P_2 **such that** $P_1 \xrightarrow[\ell]{r} P_2$

  Simulate $(P_2, t+\Delta)$

else

  select $(L,\infty(n)) \in GReacts$

    with probability n/m *where* $m = \sum_{(L',\infty(n')) \in GReacts} n'$

  select $(\ell,\infty) \in Reacts(L)$ *with equal probability*

  let P_2 **such that** $P_1 \xrightarrow[\ell]{\infty} P_2$

  Simulate (P_2, t)

Figure 3.19: Stochastic simulator for $\pi(\mathscr{L})$ (to be implemented incrementally).

What remains is to compute the propensities of all labels of grouped reactions in a process P_1. These can be derived from the values below if $P_1 = \prod_{i=1}^{n} \sum_{j=1}^{m} \pi_i^j.P_i^j$:

$$\begin{aligned} out(x,v) &= \#\{(i,j) \mid \exists \tilde{v} :\ \pi_i^j \Downarrow x[v]!(\tilde{v})\} \\ in(x,v,r) &= \#\{(i,j) \mid \exists v' \exists \tilde{y} : \pi_i^j \Downarrow x[v']?(\tilde{y}),\ v'v \Downarrow r\} \\ mixin(x,v,r) &= \#\{(i,j_1,j_2) \mid \\ &\qquad \exists v' \exists \tilde{v} \exists \tilde{y} : \pi_i^j \Downarrow x[v]!(\tilde{v}),\ \pi_i^j \Downarrow x[v']?(\tilde{y}),\ v'v \Downarrow r\} \end{aligned}$$

Lemma 6. *$prop(x,v,r) = (out(x,v) * in(x,v,r) - mixin(x,v,r)) * r$, if the solution does not contain infinite rates.*

Proof. Let $L = (x,v,r)$. It is enough to show that $out(x,v) * in(x,v,r) - mixin(x,v,r) = \#Reacts(L)$. This holds, since all pairs of indices counted by $out(x,v) * in(x,v,r)$ form a redex according to rule (COM), except for those that are counted by $mixin(x,v,r)$. □

The computation of mixins can still produce an output of quadratic size and thus needs quadratic time. The square factor, however, is in the maximal number of alternatives in sums defining molecules of P_1 which will be small in practice. All other needed values can be computed in linear time in the size of P_1, when ignoring the time for evaluating expressions, which is justified in many practical cases.

The final step toward an efficient simulator consists of computing the propensities $prop(x,v,r)$ incrementally, such that they do not need to be recomputed from scratch in every reduction step. This can be based on Lemma 6, since the values of $out(x,v)$, $in(x,v,r)$, and

$mixin(x, v, r)$ can be updated incrementally when adding new solutions or canceling alternative choices by communication.

3.5 Implementation and Performance Evaluation

In this section, the implementation of the stochastic simulator for $\pi(\mathscr{L})$ is outlined and some experimental results are presented in order to give an impression on its performance.

The $\pi(\mathscr{L})$ simulator is implemented on top of the simulator for the stochastic π-calculus by Leye et al. (2010) in the modeling and simulation framework JAMES II (Himmelspach and Uhrmacher, 2007). The implementation is freely available[1]. It relies on a two layer approach: the base layer is the simulator of the stochastic π-calculus along the lines of Phillips and Cardelli (2007), i.e. for each communication channel the propensity is calculated under consideration of the corresponding senders and receivers and the rate constant assigned to the channel. The results are passed to a Stochastic Simulation Algorithm (SSA) that determines the next communication to perform and the sojourn time. There are three alternative versions of the SSA that can be freely chosen, the First Reaction Method by Gillespie (1976), the Direct Reaction Method by Gillespie (1977), and the Next Reaction Method by Gibson and Bruck (2000). The top layer implements the

[1] See the James-Imp-Pi web page at `http://biopi-lille-ros.gforge.inria.fr`.

grouping as explained in Section 3.4, i.e. it groups the communication pairs in a solution by channels, constraint arguments, and rate constants. In order to link the two layers, the idea of implementing the Euglena model in the stochastic π-calculus in Section 3.2.1 is applied. That is, for each group an extra channel is created to which the corresponding senders, receivers and also the rate constant are assigned. The set of the thus obtained communication channels is passed to the base layer, which determines the next model state.

Performance experiments were carried out to compare the simulator of $\pi(\mathscr{L})$ to the stochastic π-calculus simulator in JAMES II and the stochastic Pi Machine (SPiM) by Phillips and Cardelli (2007). Only the Direct Reaction Method is considered, since this is the only version of the SSA supported by SPiM. The experiments were performed on a WindowsXP machine with an Intel Core 2 Duo 2.00 GHz processor, and 2 GB RAM providing a SciMark 2.0 Java benchmark score[2] of 383.9 Mflops. Simulations in JAMES II used the Mersenne Twister random number generator as introduced by Matsumoto and Nishimura (1998). Notice, that there exists a faster version of SPiM for Linux based on native code compilation, an aspect that is, however, irrelevant for this study. For further details on the runtime of SPiM compared to the stochastic π-calculus simulator in JAMES II see Leye et al. (2010).

A well-suited test example is provided by the Euglena model in Section 3.2.1, as it allows gradually raising the number of grouped reactions and process definitions by increasing the number of depth levels.

[2] `http://math.nist.gov/scimark2/`, 06/22/2010

Furthermore, it can be implemented in both the stochastic π-calculus and $\pi(\mathscr{L})$. Implementations in the stochastic π-calculus are obtained by enumerating the Euglena processes for different depth levels in the same way as shown in Section 3.2.1. The Euglena benchmark model comprises two light sources with intensity rates $\mathtt{l}_1 = 5.0$ and $\mathtt{l}_2 = 15.0$ and 100 Euglenas on each depth level ($\mathtt{n} = 100$). The rate of Euglena's upwards motion is set to $\mathtt{u} = 2.0$ and the water opacity to $\sigma = 0.2$. Among the experiments, the number of depth levels was gradually increased from 10 to 100 by steps of 10, i.e. $\mathtt{m} \in \{9, 19, \ldots, 99\}$. To ensure comparability, two model implementations in $\pi(\mathscr{L})$ for each experiment were used, one enumerating the depth levels as in the stochastic π-calculus ("Enum") and one in the more compact form with the depth level as a parameter of Euglena ("Comp"). Measured was the time needed to simulate until time point 100.0, see Appendix A. The results for each experiment are the average of three simulation runs with small deviations due to both the stochastic nature of the simulation and the work load of the machine. The results of the experiment sets are shown in Figure 3.20. The implementations are labeled according to the utilized formalism, "StoPi" or "AttrPi", the tool, "SPiM" or "James", and the implementation, "Enum" or "Comp".

The results show a general increase in simulation time with an increase in the number of depth levels. Presumably due to the choice of operating system, SPiM performs slower. All other implementations need similar amounts of time. The maximal simulation time required is around 160*s*. The results indicate that the computational complexity of the $\pi(\mathscr{L})$ simulator is moderate.

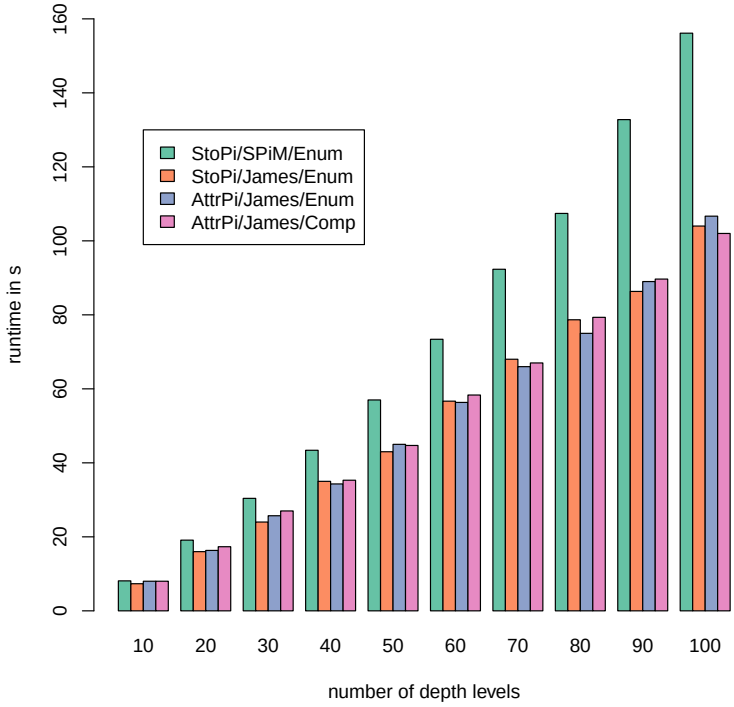


Figure 3.20: Runtime of different simulators in s for the Euglena model with parameters $\mathtt{l}_1 = 5.0$, $\mathtt{l}_2 = 15.0$, $\mathtt{n} = 100$, $\sigma = 0.2$, $\mathtt{u} = 2.0$, and $\mathtt{m} \in \{9, 19, \ldots, 99\}$, i.e. number of depth levels ranging between 0 and 100. Simulation runs were performed until simulation time $t = 100.0$ (average of 3 runs each): "StoPi" = the stochastic π-calculus, "AttrPi" = the attributed π-calculus, "SPiM" = SPiM, "James" = JAMES II, "Enum" = model with enumerated depth levels, "Comp" = model with depth level as species parameter.

Chapter 4

The Imperative π-Calculus

This chapter introduces the imperative π-calculus with its attribute language, syntax, and non-deterministic and stochastic semantics (Section 4.1). As discussed already in Section 1, a simple type system for the imperative π-calculus is omitted since there exists no obvious way to obtain one that covers the encoding of BioAmbients as presented here. Modeling and expressiveness studies in Sections 4.2 and 4.3, respectively, show the usefulness of the imperative π-calculus for the spatial and stochastic modeling of cell-biological systems. Section 4.4 presents a stochastic simulator for the imperative π-calculus including a short discussion on its performance.

4.1 Language

The imperative π-calculus extends $\pi(\mathscr{L})$ by a global imperative store. That is, values are assigned to channels that can be changed by process communication. As before, constraints and attributes are specified in an attribute language with the basic difference that the evaluation of expressions additionally considers a global store and operators for its modification, see Section 4.1.3. Syntactically, attributed processes are only slightly extended by introducing initial channel values in ν-operators, see Section 4.1.4. Semantically, the basic difference is that pairs of processes and stores are considered. However, the impact of assignments in concurrent processes requires special attention. The basic idea here is to restrict assignments to places that are not read concurrently, see Sections 4.1.1, 4.1.4, 4.1.5. Nevertheless, the imperative π-calculus is a mostly conservative extension of the attributed π-calculus. In fact, neglecting the imperative store, the calculi are equal, as reflected by both their non-deterministic and stochastic semantics, see Section 4.3.1.

In the following, first the idea of a global store is recalled and basic design decisions are explained. Attribute languages, syntax, and non-deterministic and stochastic semantics of $\pi^{imp}(\mathscr{L})$ are introduced in Sections 4.1.3, 4.1.4, 4.1.5, and 4.1.7, respectively. In Section 4.1.6 it is shown that, by restricting the use of assignments, the convergence of imperative processes produces unique results. Section 4.3.1 highlights the conservative nature of the imperative extension by comparing the non-deterministic and stochastic semantics of $\pi^{imp}(\mathscr{L})$ and $\pi(\mathscr{L})$.

4.1.1 Idea of a Global Imperative Store

For illustration, consider a compartment C containing molecules of species R_1 and R_2, which may bind to a product P, and a flux constantly adding molecules of species R_1 to C:

$$\begin{array}{lll} \mathit{bind}: & R_1, R_2 & \xrightarrow{k_1} \quad P \\ \mathit{flux}: & & \xrightarrow{k_2} \quad R_1 \end{array}$$

Following Gillespie (1977), the affinity of a two-reactant reaction depends on the volume of the containing compartment. Thus, k_1 is only valid for a constant volume of C. With the flux adding molecules R_1 to C, the volume of C, however, increases. Using a global imperative store, this side effect of the flux can be modeled in the imperative π-calculus in the following way.

Assuming that V_{R_1} denotes the volume of a single R_1 molecule, V_C the initial value of compartment C, N_{R_1}, N_{R_2}, and N_P the initial amounts of the reactants and the product, respectively, and the affinity of the binding reaction in dependency of the compartment volume yields k_1/v, where v denotes the compartment volume, the model may read as follows, where P is left unspecified:

Global variables

v: V_C // compartment volume

Process definitions

$\mathsf{R}_1() \triangleq \mathit{bind}[\lambda\_ . k_1/(\texttt{val } v)]?().\mathsf{P}()$

$\mathsf{R}_2() \triangleq \mathit{bind}[\_]!() . \mathbf{0}$

$$\mathsf{F}() \triangleq \mathit{flux}[\lambda\_.v{:=}(\texttt{val}\ v) + V_{R_1}; k_2]?().\ (\mathsf{F}() \mid \mathsf{R}_1())$$

$$\mathsf{T}() \triangleq \mathit{flux}[\_]!().\mathsf{T}()$$

$$\mathsf{P}() \triangleq \ldots$$

Solution

$$\prod_{i=1}^{N_{R_1}} \mathsf{R}_1() \mid \prod_{i=1}^{N_{R_2}} \mathsf{R}_2() \mid \prod_{i=1}^{N_P} \mathsf{P}() \mid \mathsf{F}() \mid \mathsf{T}()$$

A global variable, i.e. a channel, v denotes the current volume of the compartment which is initially set to the initial compartment volume V_C. Process definitions $\mathsf{R}_1()$ and $\mathsf{R}_2()$ represent the reactants and $\mathsf{P}()$ the product, respectively. Processes $\mathsf{F}()$ and $\mathsf{T}()$ are defined to perform the flux. As usual, the binding reaction is reflected by an interaction of processes $\mathsf{R}_1()$ and $\mathsf{R}_2()$ on a channel *bind*, where $\mathsf{R}_1()$ proceeds with $\mathsf{P}()$ and $\mathsf{R}_2()$ is consumed. Their constraint value yields the affinity of the binding reaction k_1/v. Therefore, the constraint function on the receiver side accesses the value of global variable v by applying functional constant `val` to it. Similarly, processes $\mathsf{F}()$ and $\mathsf{T}()$ communicate on a channel *flux*, where both proceed recursively and $\mathsf{F}()$ additionally adds a process $\mathsf{R}_1()$ to the solution. The constraint function of process $\mathsf{F}()$ is defined as a sequence. The evaluation result of a sequence is considered to be given by its second component, i.e., in this case, constantly the flux's rate constant k_2. The sequence's first component implements the side effect of the flux, i.e. it increases the value of global variable v, representing the compartment's volume, by the volume of a single R_1 molecule V_{R_1}. A change in the value of v is committed to the global store every time the communication

on *flux* occurs. Simultaneously, all other constraint expression are re-evaluated, such that processes $\mathsf{R}_1()$ and $\mathsf{R}_2()$ always interact with the right affinity. A more elaborated example using global variables to model changes in compartment volumes is provided in Section 4.2.1.

4.1.2 Design Decisions

What follows reports on a few important decisions regarding the design of the imperative π-calculus.

Assignments and concurrency. Although processes run in parallel, the order of evaluation must be deterministic. Consider e.g. a solution $A(x{:=}1) \mid B(\texttt{val}\ x)$ with an initial value of $x = 2$. There is no inherent order of evaluation of the expressions in the solution, such that with $B(1)$ and $B(2)$ two different results are possible. In the design of the imperative π-calculus, this problem is met by prohibiting assignments in the context of actual process parameters and initial values of channels. In Section 4.1.6, it is shown that this is, indeed, a proper solution. An alternative is to obtain an order of such evaluations by determining whether processes take the role of senders or receivers in communications. However, this leads to higher computational costs, since parallel processes need to be evaluated more than once in different orders.

Environments as additional binders. As usual in imperative programming languages, the imperative π-calculus reflects the global store in its semantics by introducing pairs of processes and environ-

ments, the latter mapping variables to values. Environments act as binders for processes. In a process-environment pair, all the free names of the process, which the environment maps to values, are bound. In the definition of the imperative π-calculus, this dependency is made explicit by lifting the notion of free names and structural congruence up to the level of process-environments. This also allows extending α-conversion to environments, which, in turn, leads to a slightly weaker form of structural congruence, such that more process-environment pairs can be identified.

Order of evaluation. Since reaction constraints may cause side effects, the order in which senders and receivers are evaluated is of great significance. When evaluating senders first, a receiver can only be evaluated under consideration of the changes in the environment caused by a specific sender, and vice versa. The imperative π-calculus is designed, such that senders and then receivers are evaluated with an argument originating in the context of simulation. The stochastic simulator needs to group senders and receivers not only by their channels but also by the values of their constraints, see Section 4.4. In many cases, senders provide simple constraint values that are easy to compare and thus allow for an effective grouping. By contrast, the constraint values of receivers are functions by definition, which are hard to identify. Thus, first evaluating and grouping senders and then assigning receivers to these groups seems to be more effective than the symmetric approach.

4.1.3 Attribute Languages

The imperative π-calculus inherits the concept of attribute languages from $\pi(\mathscr{L})$, with small extensions in order to handle global variables. Thus, an attribute language $\mathscr{L}$ of the imperative π-calculus is a functional, call-by-value programming language with expressions that base on constants $c \in \mathit{Consts}$ and variables (channel names) $x, y \in \mathit{Vars}$. As before, a specific attribute language is defined by a tuple $\mathscr{L} = (\mathit{Consts}, \Downarrow, R, <)$, with *Consts* representing a set of constants, a big-step evaluator $\Downarrow$, R denoting the set of successful values, and a partial order $<$ on R defining priority levels. With the restricted expressions $f \in \mathit{Exprs}^-$ an additional syntactic category is introduced. The definitions of constants, values $v \in \mathit{Vals}$, and expressions $e \in \mathit{Exprs}$ are selectively extended:

$$
\begin{array}{rcl}
c \in \mathit{Consts} & ::= & \texttt{false} \mid \texttt{true} \mid \texttt{unit} \mid \texttt{fst} \mid \texttt{snd} \mid \texttt{val}\ldots \\
v \in \mathit{Vals} & ::= & x \mid c \mid \lambda x.e \mid \langle v_1, v_2 \rangle \\
e \in \mathit{Exprs} & ::= & v \mid e_1 e_2 \mid \langle e_1, e_2 \rangle \mid \textsf{if } e \textsf{ then } e_1 \textsf{ else } e_2 \mid \texttt{ref}\ x \mid \\
 & & e_1 := e_2 \\
f \in \mathit{Exprs}^- & ::= & v \mid f_1 f_2 \mid \langle f_1, f_2 \rangle \mid \textsf{if } f \textsf{ then } f_1 \textsf{ else } f_2 \mid \texttt{ref}\ x
\end{array}
$$

The basic elements of set *Consts* remain Booleans, numbers and functional constants `fst` and `snd` to access the element of pairs. Additionally, constant `val` and dummy value `unit` are introduced. Operator `val` gives access to the value of a channel. Constant `unit` is used to denote that a channel's value is empty. As before, the set *Vals* contains constants, pairs of values and λ-abstractions, i.e. functions. Possible expressions are function application, pairs of expressions, and condi-

tionals. Expressions `ref` x and assignments $e_1 := e_2$ are specific to $\pi^{imp}(\mathscr{L})$. Since the value of a channel is potentially another channel, reference chains can be built. Expressions `ref` x allow obtaining the last channel in such a chain, i.e. they denote dereferentiation; more precisely, the first channel whose value is not a channel, if one exists. Assignments permit altering the values of variables. As discussed above, for proper computation, assignments are prohibited in certain places. In such places, only restricted expressions f can be used.

In contrast to $\pi(\mathscr{L})$, the big-step evaluator of $\pi^{imp}(\mathscr{L})$ maps pairs of expressions and environments to pairs of values and environments. Circular reference chains introduce a further possibility for the nontermination of evaluation. An environment is a partial function $\rho : \mathit{Vars} \to \mathit{Vals}$, mapping names to values. The set of all possible environments is given by Env. The domain of an environment $\mathit{dom}(\rho)$ yields the set of names for which ρ defines a mapping. Notation $(e,\rho) \Downarrow (v,\rho')$ abbreviates $\Downarrow (e,\rho) = (v,\rho')$. The evaluator follows the rules presented in Figure 4.1. Most of them are defined as before, except that they now consider pairs of values and environments. Only assignments may change environments, but they can be part of other expressions, such as pairs or conditions. Rule (V), (PAIR), and (SELECT) evaluate values, pairs, and select-expressions. Rule (ASS) executes assignments by first evaluating them from the left to the right, where the order is realized by the way the resulting environments are passed around, and then changing the environment, accordingly. Notation $\rho[x \mapsto v]$ describes the store, which maps all names to values as ρ, except name x, which it maps to value v. The evaluation result

$$(\text{V})\ \frac{v \in \mathit{Vals}}{(v,\rho) \Downarrow (v,\rho)} \qquad (\text{PAIR})\ \frac{(e_1,\rho) \Downarrow (v_1,\rho_1) \quad (e_2,\rho_1) \Downarrow (v_2,\rho')}{(\langle e_1,e_2\rangle,\rho) \Downarrow (\langle v_1,v_2\rangle,\rho')}$$

$$(\text{SELECT})\ \frac{(e,\rho) \Downarrow (\langle v_1,v_2\rangle,\rho')}{(\texttt{fst}\ e,\rho) \Downarrow (v_1,\rho') \quad (\texttt{snd}\ e,\rho) \Downarrow (v_2,\rho')}$$

$$(\text{ASS})\ \frac{(e_1,\rho) \Downarrow (x,\rho_1) \quad (e_2,\rho_1) \Downarrow (v,\rho')}{(e_1 := e_2,\rho) \Downarrow (\texttt{unit},\rho'[x \mapsto v])}$$

$$(\text{VAL})\ \frac{(e,\rho) \Downarrow (x,\rho') \quad \rho'(x) = v}{(\texttt{val}\ e,\rho) \Downarrow (v,\rho')} \qquad (\text{REF}_1)\ \frac{\rho(x) \notin \mathit{Vars}}{(\texttt{ref}\ x,\rho) \Downarrow (x,\rho)}$$

$$(\text{REF}_2)\ \frac{\rho(x) \in \mathit{Vars} \quad (\texttt{ref}\ \rho(x),\rho) \Downarrow (y,\rho)}{(\texttt{ref}\ x,\rho) \Downarrow (y,\rho)}$$

$$(\text{COND}_1)\ \frac{(e,\rho) \Downarrow (\texttt{true},\rho_1) \quad (e_1,\rho_1) \Downarrow (v_1,\rho')}{(\textsf{if}\ e\ \textsf{then}\ e_1\ \textsf{else}\ e_2,\rho) \Downarrow (v_1,\rho')}$$

$$(\text{COND}_2)\ \frac{(e,\rho) \Downarrow (\texttt{false},\rho_1) \quad (e_2,\rho_1) \Downarrow (v_2,\rho')}{(\textsf{if}\ e\ \textsf{then}\ e_1\ \textsf{else}\ e_2,\rho) \Downarrow (v_2,\rho')}$$

$$(\text{FUN})\ \frac{\begin{array}{c}(e_1,\rho) \Downarrow (\lambda x.e_1',\rho_1) \\ (e_2,\rho_1) \Downarrow (v',\rho_2) \quad (e_1'[v'/x],\rho_2) \Downarrow (v,\rho')\end{array}}{(e_1e_2,\rho) \Downarrow (v,\rho')}$$

Figure 4.1: Big-step evaluator of a call-by-value λ-calculus with pairs and conditionals.

$$(\mathrm{EQ}_1)\ \frac{(e_1,\rho)\Downarrow(v,\rho_1)\quad(e_2,\rho_1)\Downarrow(v,\rho')\quad v\in \mathit{Vars}\cup\mathit{Consts}}{(e_1{=}e_2,\rho)\Downarrow(\mathtt{true},\rho')}$$

$$(\mathrm{EQ}_2)\ \frac{(e_1,\rho)\Downarrow(v_1,\rho_1)\quad(e_2,\rho_1)\Downarrow(v_2,\rho')\quad v_1\neq v_2\in \mathit{Vars}\cup\mathit{Consts}}{(e_1{=}e_2,\rho)\Downarrow(\mathtt{false},\rho')}$$

$$(\odot_{\mathbb{R}})\ \frac{\begin{array}{c}\odot\in\{+,-,*,/,\mathrm{pow}\}\\ v_1\odot v_2=v\quad(e_1,\rho)\Downarrow(v_1,\rho_1)\quad(e_2,\rho_1)\Downarrow(v_2,\rho')\end{array}}{(e_1\odot e_2,\rho)\Downarrow(v,\rho')}$$

Figure 4.2: Additional rules of the big-step evaluator of the attribute language $\lambda(\mathbb{N}_0,+,=)_{<_1}$.

of assignments is constantly `unit`. By rule (VAL), the value mapped to a name x is determined. dereferentiation is recursively defined by rules (REF_1) and (REF_2), accordingly. Rule (REF_2) implements the recursive step, i.e. it considers those channel values which are channels. Rule (REF_1) returns the value of a channel which is not a channel. dereferentiation may diverge if applied to environments which are not acyclic, see below. Rules (COND_1), (COND_2), and (FUN) follow their counterparts in $\pi(\mathscr{L})$ to evaluate conditions and functions. The evaluator $\Downarrow$ also applies to restricted expressions $f\in \mathit{Exprs}^{-}$.

Since dereferentiation is introduced as a separate operator `ref`, it can be used in different contexts, e.g. in combination with assignments in order to assign a value to the end of a reference chain, or together with the operator `val` in order to obtain the value at the end of the ref-

$$
\begin{array}{rcl}
\lambda x_1 x_2 \ldots x_n.e & =_{df} & \lambda x_1.\lambda x_2.\ldots.\lambda x_n.e \\
e_1;e_2 & =_{df} & (\lambda\_.e_2)e_1 \\
\textsf{let } x = e_1 \textsf{ in } e_2 & =_{df} & (\lambda x.e_2)e_1 \\
\texttt{not } e & =_{df} & e = \texttt{false} \\
e_1 \texttt{ and } e_2 & =_{df} & \textsf{if } e_1 \textsf{ then } e_2 \textsf{ else } \texttt{false}
\end{array}
$$

Figure 4.3: Abbreviations for expressions of the attribute language $\mathscr{L}$. The same abbreviations are valid for restricted expressions $f \in Exprs^{-}$.

erence chain. Providing dereferentiation as a separate operation supports modularity of the evaluator rules, since necessary recursive steps only need to be implemented once - no additional assignment or `val` operators are needed. It is assumed that assignments and value access are only performed in combination with dereferentiation, i.e. corresponding expressions must be of the form $\texttt{val}(\texttt{ref } x)$ and $\texttt{ref } x := e$. In order exclude divergence of dereferentiation it must be applied to acyclic environments, i.e. if there exists no $x \in dom(\rho)$, such that $x = \rho(\rho(x)\ldots)$.

Figure 4.3 presents a set of handy expression abbreviations: let-expressions, sequences of function parameters and expressions, negations, and conjunctions. The same abbreviations are valid for restricted expressions $f \in Exprs^{-}$.

As in the case of $\pi(\mathscr{L})$, the evaluator is kept rather abstract, such that it can be adapted to the application at hand. The encoding of

BioAmbients is based on the attribute language $\mathscr{L} = \lambda(R, \texttt{DIR}, \texttt{CAP}, =, \texttt{and})_{<_n}$, with a set of n priority levels as successful values. It provides an additional operators to check equality (=). It is defined by rules (EQ_1) and (EQ_2) in Figure 4.2, covering the cases of in-/equality, respectively. Sets DIR and CAP provide additional names as constants to represent communication directions and rearrangement capabilities.

The example models in Section 4.2 use the attribute language $\mathscr{L} = \lambda(\mathbb{R}_+, =, +, *, -, /, \texttt{pow})$, where the successful values are set to $R = \mathbb{R}_+$ and which provides constants for equality (=), addition (+), multiplication (*), subtraction (-), division (/), and exponentiation (pow). Evaluation of arithmetic expressions is fixed by a single rule $(\odot_{\mathbb{R}})$ in Figure 4.2, working in the expected way.

4.1.4 Syntax of Processes

Syntactically, the imperative extension of $\pi(\mathscr{L})$ only requires the introduction of ν-operators with initial channel values. As discussed above, restricted expressions $f \in \mathit{Exprs}^-$ excluding assignments are used to define initial channel values and the actual parameters of defined processes. Sequences of ν-operators are denoted by $(\nu\tilde{x}{:}\tilde{f})P$, where it is assumed that $|\tilde{x}| = |\tilde{f}|$. Notation $N_{i=1}^{n}(x_i, f_i)$ is used as an alternative for $(\nu\tilde{x}{:}\tilde{f})P$, with $|\tilde{x}| = n$.

In Figure 4.5 the definition of free names of $\pi^{imp}(\mathscr{L})$ is presented, similar to what is known from $\pi(\mathscr{L})$. Slight changes are introduced due to initial channel values and the two different syntactic categories of the attribute language. An additional equation allows obtaining the

Prefixes	π	::=	$e_1[e_2]?(\tilde{x})$	receiver
		|	$e_1[e_2]!(\tilde{e})$	sender
Sums	M	::=	$\pi.P$	guarded process
		|	$M_1 + M_2$	choice
Processes	P	::=	M	sums
		|	$A(\tilde{f})$	defined process
		|	$P_1 \mid P_2$	parallel composition
		|	$(\nu x{:}f)P$	channel creation with value
		|	$\mathbf{0}$	idle process
Definitions	D	::=	$A(\tilde{x}) \triangleq P$	parametric process definition

Figure 4.4: Syntax of $\pi^{imp}(\mathscr{L})$ where $x, \tilde{x} \in \mathit{Vars}$, $e_1, e_2, \tilde{e} \in \mathit{Exprs}$, and $f, \tilde{f} \in \mathit{Exprs}^{-}$.

$$
\begin{aligned}
fv(\mathbf{0}) &= \emptyset \\
fv(P_1 \mid P_2) &= fv(P_1) \cup fv(P_2) \\
fv(M_1 + M_2) &= fv(M_1) + fv(M_2) \\
fv(e_1[e_2]?(\tilde{y}).P) &= fv(e_1) \cup fv(e_2) \cup (fv(P) \setminus \{\tilde{y}\}) \\
fv(e_1[e_2]!(\tilde{e}).P) &= fv(e_1) \cup fv(e_2) \cup fv(\tilde{e}) \cup fv(P) \\
fv((\nu x{:}f)P) &= fv(P) \cup fv(f) \setminus \{x\} \\
fv(A(\tilde{f})) &= fv(\tilde{f}) \\
fv(A(\tilde{x}) \triangleq P) &= fv(P) \setminus \{\tilde{x}\} \\
fv(P, \rho) &= fv(P) \setminus dom(\rho)
\end{aligned}
$$

Figure 4.5: Free names of $\pi^{imp}(\mathscr{L})$.

set of free names of process-environment pairs $fv(P, \rho)$. Following the idea of ρ being an additional binder, $fv(P, \rho)$ is defined to contain all free names of P, except those which are contained in the domain of ρ. Bound names of processes $bv(P)$ and process-environment pairs $bv(P, \rho)$ are all names which are not free. The concept of naming bound variables distinctly is adopted from $\pi(\mathscr{L})$. In the semantics when applying a reduction step to (P, ρ), it is assumed that all bound variables in (P, ρ) are named distinctly and $fv(P, \rho) = \emptyset$.

The structural congruence of $\pi^{imp}(\mathscr{L})$ is the least relation satisfying the rules given in Figure 4.6. As before, they define associativity and commutativity for summation and parallel composition and set process $\mathbf{0}$ to be the neutral element of parallel composition. Furthermore, rules are introduced that allow for scope intrusion and extru-

$$
\begin{array}{rcl}
P \mid \mathbf{0} & \equiv & P \\
P_1 \mid P_2 & \equiv & P_2 \mid P_1 \\
M_1 + M_2 & \equiv & M_2 + M_1 \\
(P_1 \mid P_2) \mid P_3 & \equiv & P_1 \mid (P_2 \mid P_3) \\
(M_1 + M_2) + M_3 & \equiv & M_1 + (M_2 + M_3) \\
(\nu x{:}f)(P \mid Q) & \equiv & (\nu x{:}f)P \mid Q, \text{ if } x \notin fv(Q) \\
(\nu x_1{:}f_1)(\nu x_2{:}f_2)P & \equiv & (\nu x_2{:}f_2)(\nu x_1{:}f_1)P, \\
 & & \qquad \text{if } x_1 \notin fv(f_2),\ x_2 \notin fv(f_1) \\
P \equiv_\alpha P' & \Rightarrow & P \equiv P' \\
P \equiv P' & \Rightarrow & (P,\rho) \equiv (P',\rho) \\
(P,\rho) \equiv_\alpha (P',\rho') & \Rightarrow & (P,\rho) \equiv (P',\rho')
\end{array}
$$

Figure 4.6: Axioms of structural congruence of $\pi^{imp}(\mathscr{L})$.

sion of ν-binders, for changes in the order of ν-binder sequences, and for α-conversion in processes, i.e. consistent renaming. Apparently, the order of ν-binder sequences may only be changed if the expressions specifying their initial values do not access associated variables. As discussed in Section 4.1.2, structural congruence is also defined for process-environment pairs. Two pairs with the same environment are congruent, if their processes are. α-conversion is extended, such that bound names in $bv(P,\rho)$ may be consistently changed in process-environment pairs. The *prenex normal form* of processes in $\pi^{imp}(\mathscr{L})$ is defined as follows:

Definition 3. A process P is said to be in *prenex normal form*, if $P = (\nu\tilde{x}{:}\tilde{f})(\prod_{i=1}^{n} M_i \mid \prod_{i=1}^{m} A_i(\tilde{f}_i))$ and all bound names in P are named distinctly.

Notice that for all processes P, there exists a process P', such that P' is in prenex normal form and $P \equiv P'$.

4.1.5 Non-Deterministic Operational Semantics

The non-deterministic semantics of the imperative π-calculus operates on process-environments pairs, such that changes in environments can be captured. Thus, rules of the non-deterministic semantics of $\pi(\mathscr{L})$ cannot be directly inherited. An additional difference regards the handling of ν-operators. In $\pi(\mathscr{L})$, ν-operators are used to introduce new names. Processes in the scope of the same ν-binders may interact, as denoted by rule (NEW), and already introduced names are captured by

Communication and application steps and channel initialization

$f \in Exprs^{-}$ (no assignments)

$$(\text{TUP})\ \frac{\wedge_{i=1}^{n}(e_i,\rho_{i-1}) \Downarrow (v_i,\rho_i)}{((e_i)_{i=1}^{n},\rho_0) \Downarrow ((e_i)_{i=1}^{n},\rho_n)}$$

$$(\text{SEND})\ \frac{(e_1,\rho) \Downarrow (x,\rho_1) \quad (e_2,\rho_1) \Downarrow (v,\rho_2) \quad (\tilde{e},\rho_2) \Downarrow (\tilde{v},\rho')}{(e_1[e_2]!(\tilde{e}),\rho) \Downarrow (x[v]!(\tilde{v}),\rho')}$$

$$(\text{REC})\ \frac{(e_1,\rho) \Downarrow (x,\rho_1) \quad (e_2,\rho_1) \Downarrow (v,\rho')}{(e_1[e_2]?(\tilde{x}),\rho) \Downarrow (x[v]?(\tilde{x}),\rho')}$$

$$(\text{APP})\ \frac{(\tilde{f},\rho) \Downarrow (\tilde{v},\rho) \quad A(\tilde{x}) \triangleq P}{(A(\tilde{f}),\rho) \xrightarrow[nd]{app} (P[\tilde{v}/\tilde{x}],\rho)}$$

$$(\text{NEW})\ \frac{(f,\rho) \Downarrow (v,\rho) \quad x \notin dom(\rho)}{((\nu x{:}f)P,\rho) \xrightarrow[nd]{new} (P,\rho[x \mapsto v])}$$

$$(\text{COM})\ \frac{\begin{array}{c}(v_1v_2,\rho_2) \Downarrow (r,\rho') \quad r \in R \quad |\tilde{v}| = |\tilde{y}| \\ (\pi_2,\rho) \Downarrow (x[v_2]!(\tilde{v}),\rho_1) \quad (\pi_1,\rho_1) \Downarrow (x[v_1]?(\tilde{y}),\rho_2)\end{array}}{(\pi_1.P_1 + M_1 \mid \pi_2.P_2 + M_2,\rho) \xrightarrow[nd]{r} (P_1[\tilde{v}/\tilde{y}] \mid P_2,\rho')}$$

continued...

Figure 4.7: Non-deterministic operational semantics of $\pi^{imp}(\mathscr{L})$ with priority levels in $(R,<)$.

Program errors

$$(\text{E.PREF})\ \frac{\neg\exists(\pi',\rho').(\pi,\rho) \Downarrow (\pi',\rho')}{(\pi.P+M,\rho) \xrightarrow[nd]{err} \bot}$$

$$(\text{E.CONSTR})\ \frac{\neg\exists(v,\rho').(v_1v_2,\rho_2) \Downarrow (v,\rho') \quad (\pi_2,\rho) \Downarrow (x[v_2]!(\tilde{v}),\rho_1) \quad (\pi_1,\rho_1) \Downarrow (x[v_1]?(\tilde{y}),\rho_2)}{(\pi_1.P_1+M_1 \mid \pi_2.P_2+M_2,\rho) \xrightarrow[nd]{err} \bot}$$

$$(\text{E.COM})\ \frac{|\tilde{v}| \neq |\tilde{y}| \quad (\pi_2,\rho) \Downarrow (x[v_2]!(\tilde{v}),\rho_1) \quad (\pi_1,\rho_1) \Downarrow (x[v_1]?(\tilde{y}),\rho_2)}{(\pi_1.P_1+M_1 \mid \pi_2.P_2+M_2,\rho) \xrightarrow[nd]{err} \bot}$$

Structural rules where $\beta \in \{err, app, new\} \cup R$

$$(\text{PAR})\ \frac{(P_1,\rho) \xrightarrow[nd]{\beta} (P_1',\rho[\tilde{x}/\tilde{y}] \cup \rho')}{(P_1 \mid P_2,\rho) \xrightarrow[nd]{\beta} (P_1' \mid P_2[\tilde{x}/\tilde{y}],\rho[\tilde{x}/\tilde{y}] \cup \rho')}$$

$$(\text{STRUC})\ \frac{(P,\rho) \equiv (P_1,\rho_1) \quad (P_1,\rho_1) \xrightarrow[nd]{\beta} (P_2,\rho_2) \quad (P_2,\rho_2) \equiv (P',\rho')}{(P,\rho) \xrightarrow[nd]{\beta} (P',\rho')}$$

continued...

Figure 4.7: Non-deterministic operational semantics of $\pi^{imp}(\mathscr{L})$ with priority levels in $(R,<)$ (continued).

Error-free convergence with $\sigma = \xrightarrow[nd]{app} \cup \xrightarrow[nd]{new}$

$$(\text{CONV})\ \frac{(P,\rho)\sigma^*(P',\rho') \quad P' = \prod_{i=1}^{n} M_i \quad \neg(P',\rho') \xrightarrow[nd]{err} \bot}{(P,\rho) \Downarrow (P',\rho')}$$

Reduction ($r \in R$)

$$(\text{PRIOR})\ \frac{(P,\rho) \Downarrow (P_1,\rho_1) \quad \begin{array}{c}(P_1,\rho_1) \xrightarrow[nd]{r} (P',\rho') \\ \neg\exists r' \in R.\ r < r' \wedge (P_1,\rho_1) \xrightarrow[nd]{r'} (P_2,\rho_2)\end{array}}{(P,\rho) \to (P',\rho')}$$

Figure 4.7: Non-deterministic operational semantics of $\pi^{imp}(\mathscr{L})$ with priorities in $(R,<)$ (continued).

keeping all ν-binders. By contrast, in $\pi^{imp}(\mathscr{L})$, ν-operators are intended to extend the global store by a new name with an initial value. Reduction steps must not be applied to processes in the scope of ν-operators, since the access of names, e.g. by applying constant `val` to them, could fail. Already existing names are kept in the store. Thus, rule (NEW) is rephrased to eliminate ν-binders and add their name and initial value to the global store. In case the store already contains the very same name, α-conversion may be applied, see also the example below.

The non-deterministic semantics of $\pi^{imp}(\mathscr{L})$ defines reduction relation $\to$, which itself is based on the four reduction relations $\xrightarrow[nd]{app}$, $\xrightarrow[nd]{new}$, $\xrightarrow[nd]{r}$, and $\xrightarrow[nd]{err}$. Labels *nd*, *app*, *err*, and *r* are used as before,

i.e. to distinguish the steps of the non-deterministic from those of the stochastic semantics, see Section 4.1.7, to denote application steps and the reduction steps of erroneous processes, and to capture the priority level of a communication, respectively. The additional label *new* is annotated to ν-binder elimination steps. Communication steps may only be applied, if all unguarded defined processes are replaced by their definition and all initial channel values are evaluated, such that any non-termination or failure of corresponding expressions blocks reduction.

Figure 4.7 presents the rules of the non-deterministic semantics of $\pi^{imp}(\mathscr{L})$. Rule (TUP) evaluates tuples of expressions. Notice that due to possible changes in the environment, the order of evaluation matters. In all such sequential cases, an order from left to right is assumed. Rule (APP) replaces defined processes by their definitions. Therefore, first the expressions forming the actual parameters are evaluated. Notice that their evaluation cannot lead to changes in the environment, since in expressions $\tilde{f}$ no assignments are allowed. Rule (NEW) adds the initial value of a new channel to the environment and eliminates the corresponding ν-binder. Also here no environmental changes can occur due to the restrictions applied to expressions f. Rules (SEND) and (REC) evaluate the expressions in prefixes. By rule (COM) a send and a receive action in two concurrently running summations can communicate if they perform on the same channel. As before, the constraint function of the receiver applied to the constraint argument of the sender must yield a successful value and the receiver must await for as many values as the sender delivers. Error rules allow detecting

prefixes for which evaluation fails, constraint application which fails, and mismatches in the numbers of send values and receive parameters. Rules (PAR) and (STRUC) enable application, channel creation, and error steps in parallel compositions and under structural congruence. In the case of rule (PAR), α-conversion can be also applied to (P_1, ρ) when reducing it. This is accounted for by replacing the names in P_2 accordingly ($P_2[\tilde{x}/\tilde{y}]$). As discussed above, a structural rule for reduction in the scope of ν-binders is omitted. Rule (CONV) brings processes into prenex normal form by exhaustively applying application and channel creation steps in any order, i.e. the union of relations $\xrightarrow[nd]{app}$ and $\xrightarrow[nd]{new}$. As already mentioned, processes may not converge due to definitions $A() \triangleq A()$, failure in expression evaluation, or errors as they are covered by reduction relation $\xrightarrow[nd]{err}$. Rule (PRIOR) first forces a process to converge and then picks a single interaction with highest priority to reduce.

Example 6. Consider a solution $S = \mathsf{Sn}() \mid \mathsf{Rc}() \mid (\nu y{:}\mathtt{true})(\mathsf{Sn}() \mid \mathsf{Rc}())$, a store $\rho = \{x \mapsto \mathtt{unit}, y \mapsto \mathtt{true}\}$, and the following process definitions:

$$\mathsf{Sn}() \triangleq x[\_]!() . \mathbf{0}$$
$$\mathsf{Rc}() \triangleq x[\lambda\_.\mathbf{let}\ r{:}\ \mathtt{val}\ y = \mathtt{true} \ \mathbf{in}\ (y{:=}\mathtt{false};\ r)]?() . \mathbf{0}$$

When communicating on channel x with sender $\mathsf{Sn}()$, process $\mathsf{Rc}()$ changes the values of variable y to the Boolean `false`. The communication occurs only if variable y is set to `true`. The **let**-expression ensures that the value of variable y is checked before it is changed.

Solution S allows for the following reduction steps in environment ρ:

$$\begin{aligned}(S,\rho) &\rightarrow ((\nu y{:}\texttt{true})(\mathsf{Sn}() \mid \mathsf{Rc}()), \{x \mapsto \texttt{unit}, y \mapsto \texttt{false}\}) \\ &\equiv ((\nu y{:}\texttt{true})(\mathsf{Sn}() \mid \mathsf{Rc}()), \{x \mapsto \texttt{unit}, y' \mapsto \texttt{false}\}) \\ &\rightarrow (\mathbf{0}, \{x \mapsto \texttt{unit}, y' \mapsto \texttt{false}, y \mapsto \texttt{false}\})\end{aligned}$$

First, processes $\mathsf{Sn}()$ and $\mathsf{Rc}()$ outside of the scope of the ν-binder communicate, changing the value of the variable y in the store to Boolean `false`. By α-conversion, the name of variable y in the global store of the resulting process-environment pair can be changed to y'. In the second step, first the ν-binder is eliminated by adding the mapping $y \mapsto \texttt{true}$ to the environment. This allows processes $\mathsf{Sn}()$ and $\mathsf{Rc}()$ to communicate, changing the value of y to Boolean `false`.

4.1.6 Uniqueness of Convergence

In the imperative π-calculus convergence yields unique results if the reduction relation $\xrightarrow[nd]{app} \cup \xrightarrow[nd]{new}$ fulfills uniform confluence (Definition 2) and terminates. In the following, it is first shown that both reduction relations $\xrightarrow[nd]{new}$ and $\xrightarrow[nd]{app}$ are separately uniform confluent and terminate, see Lemmas 7 and 8. Then it is shown that the reduction relation $\xrightarrow[nd]{app} \cup \xrightarrow[nd]{new}$ is also uniform confluent and terminates, see Lemma 9. This allows for the conclusion, that process convergence in $\pi^{imp}(\mathscr{L})$ is unique, see Proposition 10.

Lemma 7. *The rewrite relation $\xrightarrow[nd]{new}$ is confluent modulo structural congruence. Irreducible processes are congruent to processes of the form $N_{i=1}^{n}(x_i, f_i)\prod_{i=1}^{m} P_i$, where for all f_i it is true, that $\neg\exists v. f_i \Downarrow v$.*

Proof. The structural congruence of the imperative π-calculus allows turning parallel compositions of unguarded ν-binders into a sequence under α-conversion. By rule (NEW), the order of steps to reduce a sequence of ν-binders is determined to be from the left to the right. However, structural congruence allows changing the order of ν-binder sequences in the following way: $(\nu x_1{:}f_1)(\nu x_2{:}f_2)P \equiv (\nu x_2{:}f_2)(\nu x_1{:}f_1)P$, if $x_1 \notin fv(f_2), x_2 \notin fv(f_1)$. The lemma relies on the latter condition and on the fact that assignments in channel initializations are not allowed. The proof makes use of the following claim, which is directly implied by the operational semantics of $\pi^{imp}(\mathscr{L})$:
Claim. Let $P = N_{i=1}^{n}(x_i, f_i)\prod_{i=1}^{m} P_i$ be a processes in normal form. Reductions $(P,\rho) \xrightarrow[nd]{new} (P',\rho')$ can be applied if and only if the following rule applies:

$$\frac{1 \leq j \leq n \quad f \Downarrow v \quad x_j \notin dom(\rho) \qquad (P',\rho') \equiv (N_{i=1,i\neq j}^{n}(x_i,f_i)\prod_{i=1}^{m} P_i, \rho[x_j \mapsto v])}{(P,\rho) \xrightarrow[nd]{new} (P',\rho')}$$

Let the rewrite system on congruence classes of processes-environment pairs be defined as $[(P,\rho)]_{\equiv} \xrightarrow[nd]{new} [(P',\rho')]_{\equiv}$, if $(P,\rho) \xrightarrow[nd]{new} (P',\rho')$. Since the non-deterministic semantics of $\pi^{imp}(\mathscr{L})$ is closed under structural congruence, it can be assumed that $x_j \notin dom(\rho)$, w.l.o.g. Thus, the claim above shows that channel initialization terminates on equivalences classes of processes of the form $P = N_{i=1}^{n}(x_i, f_i)\prod_{i=1}^{m} P_i$, where for all f_i it is true that $\neg\exists v. f_i \Downarrow v$.

To see uniform confluence of $\xrightarrow[nd]{new}$, consider reductions $(P,\rho) \xrightarrow[nd]{new} (P_1,\rho_1)$ and $(P,\rho) \xrightarrow[nd]{new} (P_2,\rho_2)$, with j_1 and j_2 chosen according to

the rule in the claim above and $\rho_1 = \rho[x_{j_1} \mapsto v_{j_1}]$ and $\rho_2 = \rho[x_{j_2} \mapsto v_{j_2}]$. If $j_1 = j_2$ then $(P_1, \rho_1) \equiv (P_2, \rho_2)$. Since $(P, \rho) \xrightarrow[nd]{new} (P_1, \rho_1)$ and $(P, \rho) \xrightarrow[nd]{new} (P_2, \rho_2)$, it must be possible to put both $(\nu x_{j_1}{:}f_{j_1})$ and $(\nu x_{j_2}{:}f_{j_2})$ to the left-most position under structural congruence, such that $x_{j_1} \notin fv(f_{j_2})$ and $x_{j_2} \notin fv(f_{j_1})$. Thus, one can choose $(P', \rho') = (N_{i=1, i \neq j_1, j_2}^{n}(x_i, f_i) \prod_{i=1}^{m} P_i, \rho[x_{j_1} \mapsto v_{j_1}][x_{j_2} \mapsto v_{j_2}])$, such that $(P_1, \rho_1) \xrightarrow[nd]{new} (P', \rho')$ and $(P_2, \rho_2) \xrightarrow[nd]{new} (P', \rho')$. □

Lemma 8. *The rewrite relation $\xrightarrow[nd]{app}$ is confluent modulo structural congruence. Irreducible processes are congruent to processes of the form $(\nu \tilde{x}{:}\tilde{f})(\prod_{i=1}^{n_1} M_i \mid \prod_{i=1}^{n_2} A_i(\tilde{f}_i)) \mid \prod_{i=1}^{n_3} A_i(\tilde{f}'_i)$, where for all $i \in \{1, \ldots, n_2\}$ it holds that $\tilde{x} \cap fv(\tilde{f}_i) \neq \emptyset$ and for all $i \in \{1, \ldots, n_3\}$ it holds that $\neg \exists \tilde{v}. \tilde{f}'_i \Downarrow \tilde{v}$.*

Proof. The lemma relies on the facts that for each defined process only one definition exists and that the order of application steps does not matter, since assignments are not allowed. The following claim can be seen by inspecting the non-deterministic semantics of $\pi^{imp}(\mathscr{L})$:
Claim. Let $(\nu \tilde{x}{:}\tilde{f})(\prod_{i=1}^{n} M_i \mid \prod_{i=1}^{m} A_i(\tilde{f}_i))$ be a process in prenex normal form. Reductions $(P, \rho) \xrightarrow[nd]{app} (P', \rho')$ can be applied if and only if the following rule applies:

$$\frac{1 \leq j \leq n \quad fv(\tilde{f}_j) \cap \tilde{x} = \emptyset \quad \tilde{f}_j \Downarrow \tilde{v} \qquad A(\tilde{y}) \triangleq Q \quad (P', \rho') \equiv ((\nu \tilde{x}{:}\tilde{f}) \prod_{i=1, i \neq j}^{m} P_i \mid Q[\tilde{v}/\tilde{y}], \rho)}{(P, \rho) \xrightarrow[nd]{app} (P', \rho')}$$

Condition $fv(\tilde{f}_j) \cap \tilde{x} = \emptyset$ states that for an application step the expression of a defined process must not comprise any name bound by the ν-operators, since otherwise scope intrusion is not possible. Let the rewrite system on congruence classes of process-environment pairs be defined by $[(P,\rho)]_{\equiv} \xrightarrow[nd]{app} [(P',\rho')]_{\equiv}$ if $(P,\rho) \xrightarrow[nd]{app} (P',\rho')$. From the claim above it is apparent that $\xrightarrow[nd]{app}$ terminates on processes congruent to the stated form. Next, it is shown that this rewrite system is uniformly confluent. Consider the reductions $(P,\rho) \xrightarrow[nd]{app} (P_1,\rho_1)$ and $(P,\rho) \xrightarrow[nd]{app} (P_2,\rho_2)$ and let j_1, j_2 be the positions of the application step, respectively, according to the rule in the claim above. If $j_1 = j_2$ then $(P_1,\rho_1) \equiv (P_2,\rho_2)$. If not, one can choose $(P',\rho') = ((\nu\tilde{x}{:}\tilde{f}) \prod_{i=1, i\neq j_1, j_2}^{n} P_i \mid Q_{j_1}[\tilde{v}_{j_1}/\tilde{y}_{j_1}] \mid Q_{j_2}[\tilde{v}_{j_2}/\tilde{y}_{j_2}], \rho')$, such that $(P_1,\rho_1) \xrightarrow[nd]{app} (P',\rho')$ and $(P_2,\rho_2) \xrightarrow[nd]{app} (P',\rho')$. □

Lemma 9. *The reduction relation* $\xrightarrow[nd]{app} \cup \xrightarrow[nd]{new}$ *is confluent modulo structural congruence. Irreducible processes are congruent to processes in the form* $N_{i=1}^{n_1}(x_i, f_i)(\prod_{i=1}^{n_2} M_i \mid \prod_{i=1}^{n_3} A_i(\tilde{f}_i)) \mid \prod_{i=1}^{n_4} A_i(\tilde{f}'_i)$, *where for all* $i \in \{1,\ldots,n_1\}$ *it holds that* $\neg\exists v.f_i \Downarrow v$, *for all* $i \in \{1,\ldots,n_3\}$ *it holds that* $fv(\tilde{f}) \cap fv(\tilde{f}_i) \neq \emptyset$, *and for all* $i \in \{1,\ldots,n_4\}$ *it holds that* $\neg\exists\tilde{v}.\tilde{f}'_i \Downarrow \tilde{v}$.

Proof. By Lemmas 7 and 8 it is clear that $\xrightarrow[nd]{app} \cup \xrightarrow[nd]{new}$ reaches irreducible processes of the form stated above. For the union of two reduction relations to be confluent modulo structural congruence, both relations need to be confluent modulo structural congruence and they need to *commute*, see Niehren (2000). Lemmas 7 and 8 provide that

$\xrightarrow[nd]{new}$ and $\xrightarrow[nd]{app}$ are confluent modulo structural congruence. What is left to show is that they *commute*, i.e. if $(P,\rho) \xrightarrow[nd]{new} (P_1,\rho_1)$ and $(P,\rho) \xrightarrow[nd]{app} (P_2,\rho_2)$ then there exists (P',ρ'), such that $(P_1,\rho_1) \xrightarrow[nd]{app} (P',\rho')$ and $(P_2,\rho_2) \xrightarrow[nd]{new} (P',\rho')$, see (Niehren, 2000). This follows from the claim in the proof of Lemma 8. It states that for a reduction $\xrightarrow[nd]{app}$ a defined process $A(\tilde{f})$ is required such that the free names in $\tilde{f}$ are not bound by ν-operators with initial values. Also, since assignments are not allowed in the expressions of channel initialization and defined processes, it is clear that $\xrightarrow[nd]{new}$ and $\xrightarrow[nd]{app}$ commute. □

Proposition 10 (Convergence uniqueness of $\pi^{imp}(\mathscr{L})$)**.** *For every process-environment pair* (P,ρ) *there exists at most one equivalent class of process-environment pairs* $[(P',\rho')]_{\equiv}$*, such that* $(P,\rho) \Downarrow (P',\rho')$.

Proof. Follows directly from Lemma 9. □

Remark 3. *If* $P \equiv \prod_{i=1}^{n} M_i$ *and* $\neg(P,\rho) \xrightarrow[nd]{err} \perp$ *then* $(P,\rho) \equiv (P',\rho') \Leftrightarrow (P,\rho) \Downarrow (P',\rho')$.

Proof. Analogue to the proof of Remark 1. □

4.1.7 Stochastic Semantics

The stochastic semantics of $\pi^{imp}(\mathscr{L})$ extends on the stochastic semantics of $\pi(\mathscr{L})$ in that, instead of considering processes only, it defines CTMC's for process-environment pairs (P,ρ), whose states are the

Labeled communication steps ($\ell \in \mathbb{N}^4$, $r \in \mathbb{R}_+^\infty$)

$$(\text{COM}_\ell)\ \frac{\begin{array}{c}(\pi_{i_2}^{j_2}, \rho) \Downarrow (x[v_2]!(\tilde{v}), \rho_1) \quad i_1 \neq i_2 \\ (\pi_{i_1}^{j_1}, \rho_1) \Downarrow (x[v_1]?(\tilde{y}), \rho_2) \quad |\tilde{y}| = |\tilde{v}| \\ \ell = (i_1, j_1, i_2, j_2) \quad (v_1 v_2, \rho_2) \Downarrow (r, \rho') \quad r \in \mathbb{R}_+^\infty\end{array}}{\begin{array}{l}(\prod_{i=1}^n \sum_{j=1}^{m_i} \pi_i^j . P_i^j, \rho) \xrightarrow[\ell]{r} \\ \quad (\prod_{i=1, i\neq i_1, i_2}^n \sum_{j=1}^{m_i} \pi_i^j . P_i^j \mid P_{i_1}^{j_1}[\tilde{v}/\tilde{y}] \mid P_{i_2}^{j_2}, \rho')\end{array}}$$

Markov chain ($r, r' \in \mathbb{R}^+$)

$$(\text{SUM})\ \frac{\begin{array}{c}\neg\exists \ell . (P_1, \rho_1) \xrightarrow[\ell]{\infty} (P_2, \rho_2) \\ (P, \rho) \Downarrow (P_1, \rho_1) \quad \sum_{\{(r', \ell) \mid (P_1, \rho_1) \xrightarrow[\ell]{r'} (P_2, \rho_2) \equiv (P', \rho')\}} r' = r \neq 0\end{array}}{(P, \rho) \xrightarrow{r} (P', \rho')}$$

$$(\text{COUNT})\ \frac{\begin{array}{c}(P, \rho) \Downarrow (P_1, \rho_1) \\ n = \sharp\{\ell \mid (P_1, \rho_1) \xrightarrow[\ell]{\infty} (P_2, \rho_2) \equiv (P', \rho')\} \neq 0\end{array}}{(P, \rho) \xrightarrow{\infty(n)} (P', \rho')}$$

Figure 4.8: Rules of stochastic semantics of $\pi^{imp}(\mathscr{L})$. The rules of the non-deterministic semantics of $\pi^{imp}(\mathscr{L})$ in Fig. 4.7, except (COM) and (PRIOR), remain valid.

equivalence classes $[(P', \rho')]_\equiv$ of all process-environment pairs (P', ρ') reachable from (P, ρ). As before, successful values are the stochastic rates, i.e. $R = \mathbb{R}_+^\infty$, with ∞ denoting a higher priority level then real

numbers.

The rules of the stochastic semantics of $\pi^{imp}(\mathscr{L})$ are given in Figure 4.8. Rule (COM$_\ell$) evaluates prefixes and checks communication constraints and the equality of tuple length. As before, redexes are used to distinguish transitions that lead to the same state. Rules (SUM) and (COUNT) define timed and immediate transitions, respectively. Following the law of mass action, rule (SUM) determines the propensity of transition $(P,\rho) \xrightarrow{r} (P',\rho')$ by summing up the rate constants of all interaction pairs in (P,ρ) that lead to state $[(P',\rho')]_{\equiv}$. Similarly, rule (COUNT) obtains the propensity of transition $(P,\rho) \xrightarrow{n(\infty)} (P',\rho')$ by counting the interactions in (P,ρ) with infinite rate that lead to the state $[(P',\rho')]_{\equiv}$. Except rules (COM) and (PRIOR), all rules of the nondeterministic semantics of $\pi^{imp}(\mathscr{L})$ remain valid. Notice, however, that in order to sum up the rate constants of a process' possible interactions based on redexes, structural rules do not apply to communication steps anymore, as they are labeled by *nd*.

Example 7. Figure 4.9 presents a system of two reaction schemes on species A, B, and C with global side effects. The reactions increase or decrease the value of global variable b by one, depending on whether a molecule of species B is produced or consumed. By this, they allow explicitly tracing the amount of B. An according implementation in $\pi^{imp}(\mathscr{L})$ is shown on the right side. Furthermore, the CTMC is given for a process-environment pair $(A^2 \mid B^2 \mid C^1, \{b \mapsto 2\})$ as the initial chemical solution, where we write P^n instead of $\prod_{i=1}^{n} P$. In every state, the value of b and the amount of B coincide. The propensities are

Chemical reactions:

$$x : \mathsf{A},\mathsf{B} \xrightarrow{\texttt{0.5;b:=val b -1}} \mathsf{A},\mathsf{C}$$
$$y : \mathsf{A},\mathsf{C} \xrightarrow{\texttt{5; b:=val b +1}} \mathsf{A},\mathsf{B}$$

$\pi^{imp}(\mathscr{L})$ definitions:

$$\begin{aligned}
\mathsf{A} &\triangleq x[\lambda_.0.5]().\mathsf{A} \\
&\quad + y[\lambda_.5]().\mathsf{A} \\
\mathsf{B} &\triangleq x[b{:=}\ \texttt{val}\ b\ -1]!().\mathsf{C} \\
\mathsf{C} &\triangleq y[b{:=}\ \texttt{val}\ b\ +1]!().\mathsf{B}
\end{aligned}$$

CTMC with states reachable from $(\mathsf{A}^2 \mid \mathsf{B}^2 \mid \mathsf{C}^1, \{b \mapsto 2\})$:

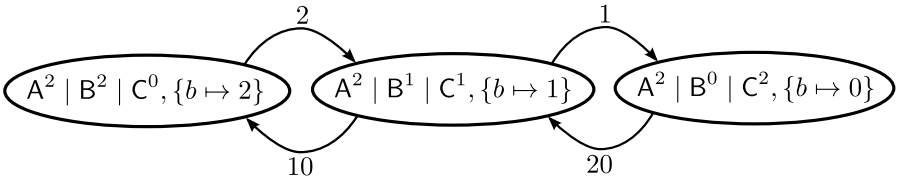


Figure 4.9: Example of a CTMC generated by $\pi^{imp}(\mathscr{L})$.

determined as by rule (SUM).

The stochastic semantics of $\pi^{imp}(\mathscr{L})$ is a proper refinement of the non-deterministic semantics of $\pi^{imp}(\mathscr{L})$.

Proposition 11. *If the set of priorities $(R,<)$ is equal to $(\mathbb{R}_+^\infty, <_2)$, where $<_2$ defines the usual two levels of priorities (i.e. $r <_2 \infty$ for all $r \in \mathbb{R}_+$), then for all process-environment pairs $(P,\rho),(P',\rho')$:*

$$(P,\rho) \to (P',\rho') \text{ iff}$$
$$(\exists r \in \mathbb{R}_+ : (P,\rho) \xrightarrow{r} (P',\rho') \vee \exists n \in \mathbb{N} : \ (P,\rho) \xrightarrow{\infty(n)} (P',\rho'))$$

Proof. The implication from the right to the left is obvious, since $(P,\rho) \xrightarrow[\ell]{r} (P',\rho')$ implies $(P,\rho) \to (P',\rho')$. For the direction from the

left to the right we start with a claim that relates communication steps to labeled communication steps in this direction:
Claim. If $(P_1,\rho_1) \xrightarrow[nd]{r} (P_2,\rho_2)$ and $P_1 = \prod_{j=1}^{n} \sum_{i=1}^{m_j} \pi_i^j.P_i^j$ then there exists a label $\ell = (i_1, j_1, i_2, j_2)$ and a process (P_2',ρ_2'), such that $(P_2,\rho_2) \equiv (P_2',\rho_2')$ and $(P_1,\rho_1) \xrightarrow[\ell]{r} (P_2',\rho_2')$.

This follows from a standard analysis of the structural congruence. Suppose now that $(P,\rho) \to (P',\rho')$ holds. In this case, the following rule must be applicable:

$$(\text{PRIOR})\quad \frac{(P,\rho) \Downarrow (P_1,\rho_1) \quad (P_1,\rho_1) \xrightarrow[nd]{r} (P',\rho') \quad \neg\exists r' \in R.\exists (P_2,\rho_2).\ r < r' \wedge (P_1,\rho_1) \xrightarrow[nd]{r'} (P_2,\rho_2)}{(P,\rho) \to (P',\rho')}$$

Without loss of generality, we can assume that P_1 is in prenex normal form, since relation $\xrightarrow[nd]{r}$ is closed under structural congruence by rule (STRUC). The second hypothesis and the above claim show that $(P_1,\rho_1) \xrightarrow[\ell]{r} (P_2,\rho_2)$ for some process-environment pair (P_2,ρ_2) with $(P_2,\rho_2) \equiv (P',\rho')$. The third hypothesis holds if and only if either $r = \infty$ or else $r \in \mathbb{R}_+$ and $\neg\exists (P_3,\rho_3).\ (P_1,\rho_1) \xrightarrow[nd]{\infty} (P_3,\rho_3)$.

- In the case $r = \infty$, we can create a transition with infinite propensity:

$$(\text{COUNT})\quad \frac{(P,\rho) \Downarrow (P_1,\rho_1) \quad n = \sharp\{\ell \mid (P_1,\rho_1) \xrightarrow[\ell]{\infty} (P_2,\rho_2) \equiv (P',\rho')\} \neq 0}{(P,\rho) \xrightarrow{\infty(n)} (P',\rho')}$$

- In the case $r \in \mathbb{R}_+$, the claim above shows that $\sum_{\{(r',\ell)|(P_1,\rho_1)\xrightarrow{r'}_{\ell}(P_2,\rho_2)\equiv(P',\rho')\}} r' = r \neq 0$. We can thus create a timed transition of the Markov chain:

$$\text{(SUM)}\ \frac{\sum_{\{(r',\ell)|(P_1,\rho_1)\xrightarrow{r'}_{\ell}(P_2,\rho_2)\equiv(P',\rho')\}} r' = r \neq 0 \quad (P,\rho) \Downarrow (P_1,\rho_1) \quad \neg\exists\ell\exists(P_3,\rho_3).(P_1,\rho_1) \xrightarrow[\ell]{\infty} (P_3,\rho_3)}{(P,\rho) \xrightarrow{r} (P',\rho')}$$

□

4.2 Modeling Techniques and Biological Examples

In this section, example models are presented, which on one hand shall show the usefulness of the imperative π-calculus to model changes in global values, in particular in compartment volumes. On the other hand, they serve as a basis to compare the imperative π-calculus to sCCP, which is a modeling formalism very close to $\pi^{imp}(\mathscr{L})$, since it provides attributed processes, constraints, and a global store. Moreover, it is shown that reactions with Michaelis-Menten kinetics can also be implemented in the imperative π-calculus. By introducing prioritized update protocols similar to those used in Section 3.2.4 to reflect changes in global information in an individual-based model, the following models could also be implemented in the attributed π-calculus. For a more detailed comparison of the attributed and the

imperative π-calculus, see Section 4.3.1.

4.2.1 Osmosis: Variable Volumes and Surfaces

Osmosis is a simple example for concurrent systems with compartments of variable volumes. It was modeled already by Versari and Busi (2009) in Sπ@, the stochastic version of π@, which offers a stochastic semantics in terms of a mapping to the SSA. Here it is shown how to simulate osmosis in the imperative π-calculus with an attribute language that provides arithmetics. The solution presented here is more flexible and accurate, in that it accounts for dynamic changes of compartment surfaces, which cannot be expressed in Sπ@.

The model describes a very simple system, which consists of a sphere filled with water (H_2O), sodium (Na^+), and chlorine (Cl^-) and a membrane through which water may diffuse. This membrane separates an inner compartment `Inn` of spherical shape, from an outer compartment `Out`, which has the form of a sphere shell (a ring in 2D). Both compartments have the same center point. The precise values of all parameters are given in Table 4.1.

For simplicity, the model adopts the assumption of Versari and Busi (2009) that the volume of a compartment is determined by summing up the volumes of the contained molecules. However, in general, attribute languages may allow for the definition of complex functions to obtain compartment volumes that e.g. consider atomic forces between particles. The volumes of compartments `Inn` and `Out` change with water moving through the membrane. The radius of compartment `Inn`

Parameters

// copy numbers of species in each compartment

$\mathtt{N}\colon \{\mathtt{H_2O}, \mathtt{Na^+}, \mathtt{Cl^-}\} \times \{\mathtt{Inn}, \mathtt{Out}\} \to \mathbb{N}$

Constants

$\mathtt{V}\colon \{\mathtt{H_2O}, \mathtt{Na^+}, \mathtt{Cl^-}\} \to \mathbb{R}^+$ // molecule volumes

$\mathtt{C} \in \mathbb{R}$ // diffusion coefficient of water

Expressions

$\mathtt{rad} =_{df} \lambda v.((3*v)/(4*\pi))^{\frac{1}{3}}$ // volume to radius

$\mathtt{surf} =_{df} \lambda r.4*\pi*r^2$ // radius to surface

// diffusion distance

$\mathtt{dist} =_{df} \lambda r_1 \lambda r_2 . r_1 + ((r_2 - r_1)/2)$

// outer radius of sphere shell

$\mathtt{rout} =_{df} \mathtt{rad}(\sum_{c \in \{\mathtt{Inn},\mathtt{Out}\}} \sum_{m \in \{\mathtt{H_2O},\mathtt{Na^+},\mathtt{Cl^-}\}} \mathtt{V}(m) * \mathtt{N}(m, c))$

Global variables // init volumes of compartments

$inn\colon \sum_{m \in \{\mathtt{H_2O},\mathtt{Na^+},\mathtt{Cl^-}\}} \mathtt{V}(m) * \mathtt{N}(m, \mathtt{Inn})$ // inner sphere

$out\colon \sum_{m \in \{\mathtt{H_2O},\mathtt{Na^+},\mathtt{Cl^-}\}} \mathtt{V}(m) * \mathtt{N}(m, \mathtt{Out})$ // outer shell

continued...

Figure 4.10: A model of osmosis with variable compartment volume and surface.

Process definitions

```
H2O(ori, des) ≜
  // diffusion from origin to destination
  diffuse[λ_.
      // radius of inner sphere
      let r = rad (val inn) in
      let a = (surf r)/10 in // diffusion area
      let s = dist r rout in // diffusion distance
      // diffusion rate
      let diff = a*C/(s*(val ori)) in (
          // update volume of origin
        ori := val ori − V(H2O);
        // update volume of destination
        des := val des + V(H2O);
        diff) // return diffusion rate
  ]?() . H2O(des, ori)
Membrane() ≜ diffuse[unit]!() . Membrane()
```

Solution

$$\prod_{i=1}^{\mathtt{N(H_2O,Inn)}} \mathsf{H_2O}(\mathit{inn}, \mathit{out}) \mid \prod_{i=1}^{\mathtt{N(H_2O,Out)}} \mathsf{H_2O}(\mathit{inn}, \mathit{out}) \mid \mathsf{Membrane}()$$

Figure 4.10: A model of osmosis with variable compartment volume and surface (continued).

parameter	value	description
$\mathtt{N}(\mathtt{Na}^+,\mathtt{Inn})$	100	number of sodium ions in compartment
$\mathtt{N}(\mathtt{Na}^+,\mathtt{Out})$	10	number of sodium ions in environment
$\mathtt{N}(\mathtt{Cl}^-,\mathtt{Inn})$	100	number of chlorine ions in compartment
$\mathtt{N}(\mathtt{Cl}^-,\mathtt{Out})$	10	number of chlorine ions in environment
$\mathtt{N}(\mathtt{H_2O},\mathtt{Inn})$	1000	initial no. of water molecules in comp.
$\mathtt{N}(\mathtt{H_2O},\mathtt{Out})$	10000	initial no. of water molecules in env.
$\mathtt{V}(\mathtt{H_2O})$	0.01	volume of one water molecule
$\mathtt{V}(\mathtt{Na}^+)$	0.0244	volume of one sodium atom
$\mathtt{V}(\mathtt{Cl}^-)$	0.0042	volume of one chlorine atom
C	2.272	diffusion coefficient of water

Table 4.1: Parameters and constants used in osmosis experiments.

may thus vary with diffusion, while the outer radius rout of compartment Out always remains fixed. Figure 4.10 shows the model of the system in $\pi^{imp}(\lambda(\mathbb{R},\mathtt{V},\mathtt{C},\mathtt{N}))$. The attribute language $\lambda(\mathbb{R},\mathtt{V},\mathtt{C},\mathtt{N})$ provides real number arithmetics with function constants for division /, multiplication *, and subtraction -, and numeric constants such as 2, 10, or π. Furthermore, there are three problem specific constants, the diffusion coefficient C of H_2O, the constant V for the function that maps molecules to their volumes, and the constant N for the function assigning copy numbers to molecules in compartments. The latter two introduce additional constants by their domains, e.g. the name $\mathtt{H_2O}$ in the domain of V. The big-step evaluator for $\lambda(\mathbb{R},\mathtt{V},\mathtt{C})$ is defined as usual. Nonzero positive real numbers are the successful values, i.e.

$R = \mathbb{R}^+$.

The diffusion rate of H_2O is determined by $\frac{a*C}{d*v}$, where a is the diffusion area, d the diffusion distance, and v the volume of the compartment that the molecule leaves, see e.g. Mazemondet et al. (2009). It is assumed that $1/10$ of the surface of compartment `Inn` serves as diffusion area. The radius and surface of compartment `Inn` are computed from its volume by functions `rad` and `surf`. The diffusion distance represents the average path that a molecule travels from one compartment to the other. Following the approach in Mazemondet et al. (2009), the diffusion distance yields the distance from the center point to the middle of the sphere shell. In the model, it is determined by function `dist` applied to the constant outer radius of compartment `Out` and the variable radius of `Inn`.

The compartments `Inn` and `Out` are modeled by public channels `inn` and `out`, respectively, each referring to the variable volume of the corresponding compartment. The public channel *diffuse* with the dummy value `unit` represents diffusion reactions. Three processes exist: $\mathsf{H_2O}(\mathit{inn},\mathit{out})$, which describes a water molecule in compartment `Inn` that may diffuse to compartment `Out`, $\mathsf{H_2O}(\mathit{out},\mathit{inn})$, its symmetric variant, and $\mathsf{Membrane}()$, which enables diffusion on channel *diffuse* at all times.

Process definition $\mathsf{H_2O}(\mathit{ori},\mathit{des})$ may perform diffusion by communication on channel *diffuse* and then continue with $\mathsf{H_2O}(\mathit{des},\mathit{ori})$. The corresponding rate constant varies with volumes and surfaces and is therefore consecutively recomputed. Every application of the constraint function performs volume changes by assignments $\mathit{ori} := \texttt{val}$

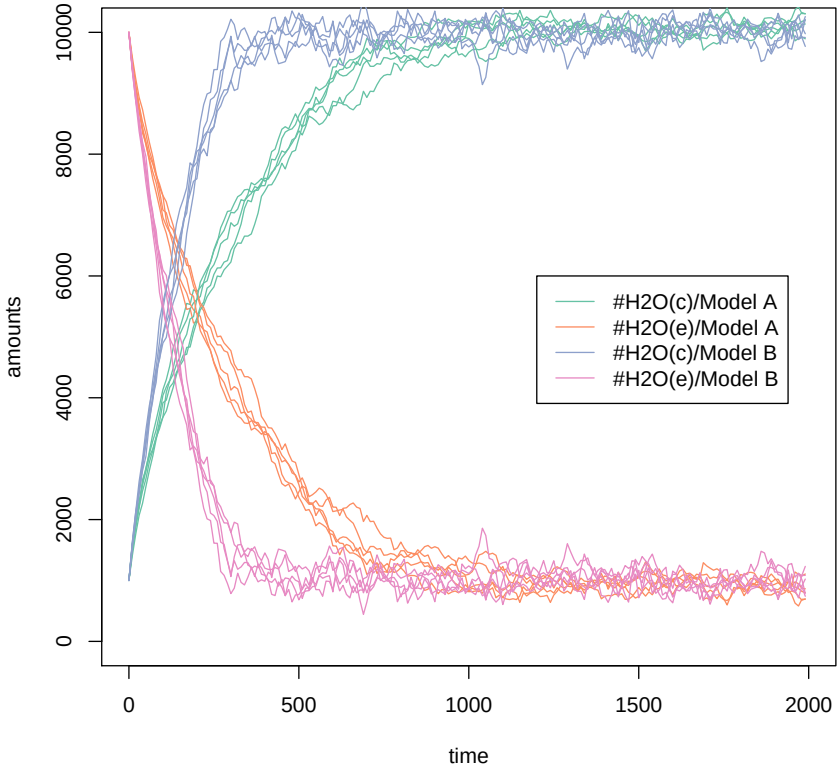


Figure 4.11: Experiment results without (Model A) and with (Model B) variable surfaces. Model parameters are provided in Table 4.1

(*ori*) - V (H_2O) and *des* := val (*ori*) + V (H_2O). Since the simulator needs to compute the diffusion rates for all possible interactions in the system (there are at most two, water moving in or out), it has to reset the environment every time. Only once some interaction is chosen by the SSA, can it commit to the changes required by this interaction.

The model presented by Versari and Busi (2009) is extended here by adapting the diffusion area and distance at each diffusion event, instead of just the compartment volume. Simulation results can be seen in Figure 4.11. Model B, being the one that considers updates of the diffusion area and distance, features a steeper slope. This is due to the fact that with the increasing volume of compartment Inn, the diffusion area grows faster than the distance, which raises the resulting diffusion rates.

4.2.2 Michaelis-Menten kinetics à la sCCP

In this section, two different models of an enzymatic reaction network, one based on Mass action and the other one on Michaelis-Menten kinetics, are presented. The goal is on one hand to provide a basis to discuss the differences between sCCP and the imperative π-calculus. On the other hand, it shall be shown that a model implemented in the sCCP-style may also correctly represent reactions with Michaelis-Menten kinetics. To underline the second point, the results of simulation experiments comparing both models are presented.

Enzymes are proteins that accelerate chemical processes. They are of significant importance, since they allow cells to process chemi-

cal substances that would normally require temperatures beyond that which a cell can survive. A typical enzymatic reaction network converts a substrate into a product in the following way:

$$\begin{aligned} \textit{bind:} \quad & E,S \xrightarrow{k_1} ES \\ \textit{decay:} \quad & ES \xrightarrow{k_{-1}} E,S \\ \textit{convert:} \quad & ES \xrightarrow{k_2} E,P \end{aligned}$$

First, the enzyme E binds to the substrate S. The resulting substrate-enzyme complex ES may either decay to its parts or carry out the conversion to an enzyme and a product P. In both cases, the enzyme is available for further interaction with the substrate.

The Michaelis-Menten theory allows abstracting these three steps to a single reaction with a rate defined by the Michaelis-Menten kinetics. The reaction network above can be reduced in the following way, where $\#S$ and $\#E$ denote the amounts of S and E, respectively, and r the reaction rate:

$$\textit{mm: } S,E \xrightarrow{K_M,k_2} P,E \quad \text{with} \quad K_M = \frac{k_{-1}+k_2}{k_1} \quad \text{and} \quad r = \frac{k_2 * \#S * \#E}{K_M + \#S}$$

K_M denotes the Michaelis-Menten constant. Under the assumption that $k_{-1} >> k_2$, K_M approximately equals the dissociation constant $K_d = k_{-1}/k_1$. The latter is easier to obtain experimentally than k_1 and k_{-1}, separately. Thus, the Michaelis-Menten theory does not just allow reducing the model complexity, since three reactions are replaced by one, but also to cope with difficulties in determining model parameters.

In the following, first a model of the detailed enzymatic reaction network based on Mass action is presented. This allows a discussion

of the basic differences between sCCP and the imperative π-calculus. Then it is shown that the sCCP-style also provides a way to correctly introduce Michaelis-Menten abstractions in the imperative π-calculus. The basic idea is to bypass the Mass action kinetics hard-wired to the stochastic semantics of $\pi^{imp}(\mathscr{L})$.

Models of reaction networks in the sCCP-style can be directly derived from the population-based example in Section 3.2.3: processes represent reactions and change the amounts of species on communication, appropriately. In contrast to the implementation in Section 3.2.3, species numbers are not provided by a central process but are captured by global variables. Figure 4.12 shows a population-based model of the enzymatic reaction network above in $\pi^{imp}(\lambda(\mathbb{R},\mathtt{K},\mathtt{N}))$. The attribute language $\lambda(\mathbb{R},\mathtt{K},\mathtt{N})$ provides arithmetics on real numbers and functional constants K and N that map parameters to values, representing rate constants and initial amounts, respectively. Global variables are introduced to capture the amounts of S (nS), E (nE), ES (nES), and P (nP). Processes Bind(), Decay(), and Convert() are defined to represent the three reactions, respectively. Process T() forms their sending interaction partner. Communication constraints are specified to completely depend on the constraint functions, which follow the same scheme: first a sequence of assignments implements the impact of the reaction, i.e. it decreases and increases species amounts, accordingly, by assigning new values to global variables. Then the rate constant is computed and returned as the constraint value. Local variables are introduced by let-expressions that allow calculating rate constants despite the reverse order of changing species amounts and computing

rate constants. The initial solution contains a single instance of process definitions Bind(), Decay(), and Convert(), since they represent reactions and not species.

The population-based schema to implement reactions in the imperative π-calculus essentially reflects the way of modeling cell-biological systems in sCCP: processes change the amounts of species, which are stored in a global store, by using an operator `tell`. The rate of change is defined based on expressions, which in return access the global store. An operator `ask` is used to check whether the condition for a reaction to happen is fulfilled, e.g. that the amount of all reactants is greater than 0. The major difference to the imperative π-calculus is, however, that processes in sCCP may not communicate. Interaction can only happen between a process and the global store. Thus, in each step, only a single process may be reduced. In an individual-based modeling style, where processes represent molecules, this only allows describing single-reactant reactions. Since, in particular, bindings happen between two reactants, such an approach is not sufficient. Thus, in contrast to the π-calculus, sCCP essentially subscribes to the population-based modeling style. Instead of species, processes describe reactions, which can be attributed with e.g. locations. However, based on priorities, this restriction of sCCP may be circumvented, since atomic reduction sequences of more than one process may be defined. In fact, sCCP provides infinite rate constants. However, such an approach seems to be a work-around. Furthermore, languages in sCCP to express constraints are, in contrast to the λ-calculus, first order and thus do not provide the same possibilities for modular implementations.

Parameters

```
K: {1,−1,2} → ℝ⁺ // rate constants
// initial amounts of substrate and enzyme
N: {S,E} → ℕ₀
```

Global variables

```
nS: N(S) // amount of substrate
nE: N(E) // amount of enzyme
// amount of enzyme−substrate complex,
   initially 0
nES: 0
nP: 0 // amount of product, initially 0
```

Process definitions

```
Bind() ≙ // reaction bind
  perform[λ_.
    let s = val nS in
    let e = val nE in (
      // change amounts according to reaction
      nS := s −1; nE := e −1; nES := val nES +1;
      k₁∗s∗e) // compute rate
  ].Bind()
```

continued...

Figure 4.12: A population-based model of an enzymatic reaction network with Mass action kinetics in $\pi^{imp}(\lambda(\mathbb{R},\mathtt{K},\mathtt{N}))$.

```
Decay() ≜ // reaction decay
  perform[λ_.
    let es = val nES in (
      // change amounts according to reaction
      nES := es −1; nS := val nS +1; nE := val
         nE +1;
      // compute rate
       k_1*es)
  ].Decay()

Convert() ≜ // reaction convert
  perform[λ_.
    let es = val nES in (
      // change amounts according to reaction
      nES := es −1; nP := val nP +1;
      // compute rate
      k_2*es)
  ].Convert()
T() ≜ perform[_]!().T()
Initial solution
// start with one process for each reaction
Bind() | Decay() | Convert() | T()
```

Figure 4.12: A population-based model of an enzymatic reaction network with Mass action kinetics in $\pi^{imp}(\lambda(\mathbb{R},\mathtt{K},\mathtt{N}))$ (continued).

Parameters

$\mathtt{K}\colon \{1,-1,2\} \to \mathbb{R}^+$ // rate constants

$\mathtt{N}\colon \{\mathtt{S},\mathtt{E}\} \to \mathbb{N}_0$ // initial amounts

Expressions

$\mathtt{K_M} =_{df} (\mathtt{K}(-1) + \mathtt{K}(2))/\mathtt{K}(1)$ // dissociation const.

Global variables

$nS\colon \mathtt{N}(\mathtt{S})$ // amount of substrate

$nE\colon \mathtt{N}(\mathtt{E})$ // amount of enzyme

$nP\colon 0$ // amount of product, initially 0

Process definitions

```
// reaction bind
MM() ≙
  perform[λ_.
    let s = val nS in
    let e = val nE in (
    // change amounts according to reaction
    nS := s −1; nE := e −1; nP := val nP +1;
    (K(2)∗s∗e∗)/(K_M∗s)) // compute rate
  ]?().MM()
T() ≙ perform[_]!().T()
```

Initial solution

```
MM() | T() // only one reaction (MM kinetics)
```

Figure 4.13: A population-based model of an enzymatic reaction network with Michaelis-Menten kinetics in $\pi^{imp}(\lambda(\mathbb{R},\mathtt{K},\mathtt{N}))$.

parameter	value	description
K(1)	100	rate constant of enzyme binding to substrate
K(-1)	0.1	rate constant of enzyme releasing substrate
K(2)	0.01	rate constant of enzyme to convert substrate
N(S)	1000	initial amount of substrate
N(E)	100	initial amount of enzyme

Table 4.2: Parameters and constants used in experiments to compare models based on Michaelis-Menten and Mass action kinetics.

Based on the ideas in the example above, a model of the more abstract enzymatic reaction with Michaelis-Menten kinetics can be easily obtained, see Figure 4.13. Process definition MM() changes the amounts of species accordingly when interacting with process T(). The rate (constant) of this interaction is defined following the rate of Michaelis-Menten reactions as presented above. This implementation of Michaelis-Menten kinetics relies on the fact that processes represent reactions and not molecules, since otherwise the way in which molecule numbers are taken into account could not be influenced by the modeler, but would be entirely dictated by the stochastic semantics, see Section 4.1.7.

Figure 4.14 shows the results of simulation experiments that were performed to compare the Mass action implementation to the Michaelis-Menten abstraction, based on the parameters in Table 4.2. In order to account for possible stochastic effects, five simulation runs have been performed for each model. For the Mass action model,

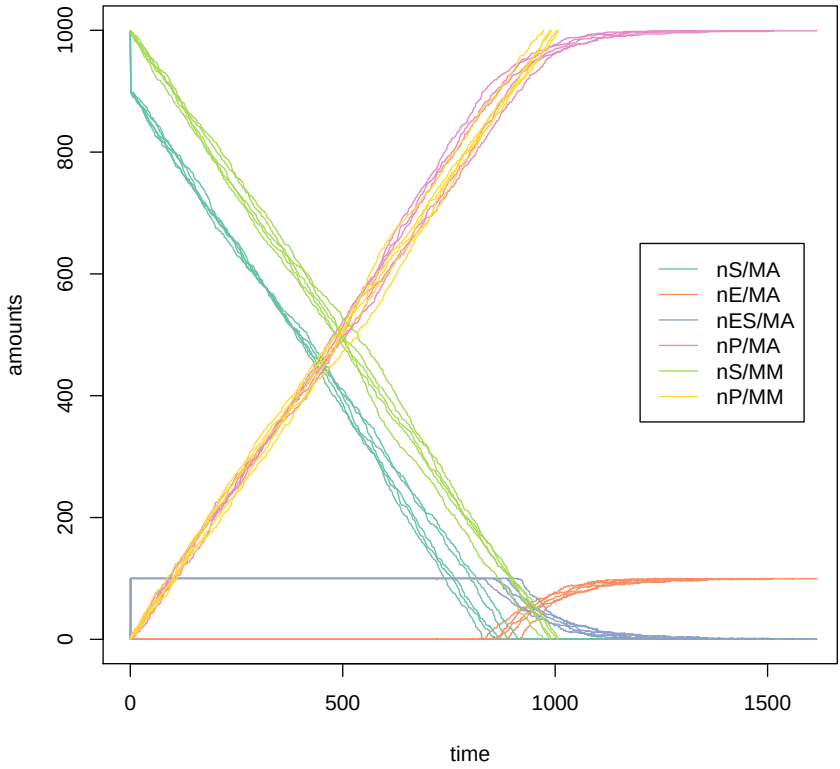


Figure 4.14: Results of simulation experiments to compare the Mass action model in Figure 4.12 with the Michaelis-Menten model in Figure 4.13, based on the model parameters in Table 4.2.

global variables S, E, ES, and P were observed and for the Michaelis-Menten model S and P. In the Figure, variable names are distinguished by annotating them with "MA" and "MM" for Mass action and Michaelis-Menten, respectively.

The behavior of both models is essentially equal. The only difference is that the substrate in the Mass action case is rapidly reduced in the beginning. This is due to the initial binding of the enzyme to the substrate, the step which is omitted by the Michaelis-Menten abstraction. In fact, summing up the amounts of substrate and substrate-enzyme complex for the Mass action model yields the amount of substrate in the Michaelis-Menten case at any time point (ignoring small stochastic variations). Consequently, the Mass action model only asymptotically approaches its final state, since the few remaining substrate-enzyme complexes have to be converted.

In the same way as Michaelis-Menten other kinetics, e.g. Hill-kinetics, may also be included into population-based models in $\pi^{imp}(\mathscr{L})$, even when including more than two reactants. However, for each kinetics it has to be separately checked to determine if it respects the Markov property, since the stochastic semantics is defined in terms of CTMC's. The restriction that only reactions with at most two reactants and Mass action kinetics can be implemented in the imperative π-calculus in an individual-based style remains valid.

4.3 Expressiveness

This section provides expressiveness studies on the imperative π-calculus, showing on one hand its conservativeness (Section 4.3.1) and on the other hand its usefulness for the spatial modeling of cell-biological processes (Section 4.3.2).

4.3.1 Conservativeness

In this section it is shown that the imperative π-calculus is a conservative extension of the attributed π-calculus. That is, there exists a transformation from attributed processes to process-environment pairs, which does nothing else than assigning dummy value `unit` to all channels. Vice versa, considering an attribute language for the imperative π-calculus without assignments, there exists an encoding from process-environment pairs to processes in the attributed π-calculus, which replaces channels by pairs, similar to the encodings in Section 3.3.3. This means, on one hand, that in the imperative π-calculus, attributed processes can be defined almost transparently. On the other hand, the only actual difference between the two calculi is the possibility to change the values of variables. Whether assignments do yield an increased expressiveness is still not investigated and subject to future work, see Section 5.

From the attributed to the imperative π-calculus

Transforming an attributed process into a process-environment pair is simply done by assigning the value `unit` to all channels:

assumption: all bound variables in P are named distinctly

$$\begin{aligned} [\![P]\!]_1 &= ([\![P]\!]_2, \{\tilde{x} \mapsto \texttt{unit}\}), \\ &\qquad \text{with } fv(P) \cup \textstyle\bigcup_{D \in \mathscr{D}} fv(D) = \{\tilde{x}\} \\ [\![(\nu x)P]\!]_2 &= (\nu x{:}\texttt{unit})[\![P]\!]_2 \\ &\ldots \end{aligned}$$

The encoding defines two steps to encode attributed process P. First an environment that maps the free names of P and its definitions to `unit` is created, since the non-deterministic semantics of the imperative $\pi - calculus$ assumes that on reduction of process-environment pairs (Q, ρ) it holds that $fv(Q, \rho) = \emptyset$. Then `unit` is assigned to all ν-operators as their initial value. All the rest remains unchanged. Clearly, this encoding is correct with respect to both the non-deterministic and the stochastic semantics.

From the imperative π-calculus without assignments to the attributed π-calculus

In the following, the encoding from the imperative π-calculus with an attribute language that prohibits assignments to the attributed π-calculus is presented. The overall idea is based on the encoding of the original version of the attributed π-calculus in Section 3.3.3. That is, pairs are used to represent the mapping from channels to their values. As before, a two-step approach is deployed, but here to encode

process-environment pairs (P,ρ), first replacing those names in P by pairs that are mapped in ρ and then those bound by ν-operators. The value of a channel is obtained by accessing the second component of a pair, i.e. operator `val` is replaced by operator `snd`. Special attention has to be drawn to reference chains, which were not explicitly considered before. They are represented by nested pairs forming lists $\langle x_1, \langle x_2, \ldots \langle x_n, v\rangle \ldots\rangle\rangle$. Operator $\texttt{snd}^*$ is introduced to obtain the last pair in a list, by this denoting the counterpart of operator `ref` for dereferentiation in the imperative π-calculus:

$$(\text{SND}_1^*)\ \frac{e \Downarrow \langle v_1, v_2\rangle \quad v_2 \neq \langle v_1', v_2'\rangle}{\texttt{snd}^* e \Downarrow \langle v_1, v_2\rangle}$$

$$(\text{SND}_2^*)\ \frac{e \Downarrow \langle v_1, v_2\rangle \quad v_2 = \langle v_1', v_2'\rangle \quad \texttt{snd}^* v_2 \Downarrow v}{\texttt{snd}^* e \Downarrow v}$$

Rule (SND_1^*) evaluates an argument e to a pair $\langle v_1, v_2\rangle$, where v_2 is not a pair, i.e. $\langle v_1, v_2\rangle$ is the last pair in a list, such that it is returned. Rule (SND_2^*) denotes the recursive step, i.e. it evaluates an argument e to a pair $\langle v_1, v_2\rangle$, where v_2 is a pair itself and recursively applies operator $\texttt{snd}^*$ to v_2.

The value at the end of a reference chain in an initial environment can be obtained a-priori. Thus, using function $\gamma(x,\rho)$ below to recursively traverse reference chains, the introduction of nested pairs in the first step of the encoding can be avoided:

$$\gamma(x,\rho) = \begin{cases} \rho(x), \text{ if } \rho(x) \notin \mathit{Vars} \\ \gamma(\rho(x),\rho), \text{ else} \end{cases}$$

Apparently, an encoding of a process-environment pair can only by obtained for acyclic environments, i.e. environments that do not contain reference chains forming rings.

In contrast to the original version of the attributed π-calculus, ν-operators in the imperative π-calculus allow for expressions to specify initial values. Thus, in the second step, a simple replacement of names bound to ν-operators by pairs of names and values is not possible. It has to be ensured that an expression is first evaluated before it is passed around. This can be achieved by introducing an additional process definition and a defined process for each ν-operator. Consider, e.g., a process $Q = (\nu x{:}f)P$ in the imperative π-calculus, with restricted expression f, containing no assignments. The encoding proposed here introduces an additional process definition $A(x) \triangleq P'$, with a fresh name A, and replaces process Q by attributed process $Q' = (\nu x)A(\langle x, f'\rangle)$, where P' and f' are the encodings of P and f, respectively. Rule (APP) of the non-deterministic semantics of the attributed π-calculus reduces Q' as follows:

$$(\nu x)A(\langle x, f'\rangle) \xrightarrow[nd]{app} (\nu x)P[\langle x, v\rangle/x], \text{ with } f' \Downarrow v$$

Exactly as needed, first expression f' is evaluated to v and then the name x in P is replaced by $\langle x, v\rangle$. Notice that all the names in the scope of ν-operators and also of receivers and process definitions have to be collected and added to the parameter list of newly introduced process definitions, such that e.g. $[\![x?(y).(\nu z{:}f)P]\!]_{\emptyset} = x?(y).(\nu z)A(y, \langle z, f\rangle)$ with new definition $A(y,z) \triangleq [\![P]\!]_{(y,x)}$. Otherwise, scoping is not consistently transferred. Notice that the parameter lists of existing process

definitions and defined processes do not need to be extended, since due to the assumption that all channels in a process to encode are named distinctly the scope of bindings ends with a defined process.

Below, the encoding of the imperative attribute language with a big-step evaluator as defined in Figure 4.1 is presented, only allowing for restricted expressions f without assignments. As discussed above, operators `val` and `ref` are replaced by `snd` and $\texttt{snd}^*$, respectively. Everything else remains the same:

$$
\begin{array}{rclcrcl}
[\![x]\!] & = & x & & [\![\texttt{val}\ f]\!] & = & \texttt{snd}\ [\![f]\!] \\
[\![\texttt{ref}\ f]\!] & = & \texttt{snd}^*\ [\![f]\!] & & [\![\lambda x.f]\!] & = & \lambda x.[\![f]\!] \\
[\![f_1 f_2]\!] & = & [\![f_1]\!][\![f_2]\!] & & [\![\langle f_1, f_2\rangle]\!] & = & \langle [\![f_1]\!], [\![f_2]\!]\rangle \\
[\![\texttt{fst}\ f]\!] & = & \texttt{fst}\ [\![f]\!] & & [\![\texttt{snd}\ f]\!] & = & \texttt{snd}\ [\![f]\!] \\
\multicolumn{7}{c}{[\![\textsf{if}\ f\ \textsf{then}\ f_1\ \textsf{else}\ f_2]\!] \ = \ \textsf{if}\ [\![f]\!]\ \textsf{then}\ [\![f_1]\!]\ \textsf{else}\ [\![f_2]\!]}
\end{array}
$$

The rules of the two-step encoding of process-environment pairs (P,ρ) are defined below. First, each name x in environment ρ is replaced by pair $\langle x, \gamma(x,\rho)\rangle$ in P and all its processes definitions, where $\gamma(x,\rho)$ obtains the value at the end of reference chain x. In the second step, process definitions and defined processes are introduced for ν-operators. The tuple $\tilde{n}$ of encoding $[\![P]\!]_{\tilde{n}}$ contains the names of all binders with scope to P that previously occurred, such that the parameter lists of the introduced process definitions and defined processes are chosen accordingly. As before, the subjects of sending and receiving prefixes are obtained by accessing the first value of a pair and all the

rest remains unchanged:

assumptions: all bound variables in P are named distinctly,
ρ is acyclic, $fv(P,\rho) = \emptyset$

$$
\begin{aligned}
[\![(P,\rho)]\!] &= [\![P[\langle x_1,\gamma(x_1,\rho)\rangle/x_1]\ldots[\langle x_n,\gamma(x_n,\rho)\rangle/x_n]]\!]_\emptyset, \\
&\qquad \textit{with } fv(P) = \{x_1,\ldots,x_n\} \\
[\![(A(\tilde{x}) \triangleq P,\rho)]\!] &= [\![A(\tilde{x}) \triangleq P[\langle x_1,\gamma(x_1,\rho)\rangle/x_1]\ldots \\
&\qquad [\langle x_n,\gamma(x_n,\rho)\rangle/x_n]]\!]_\emptyset, \\
&\qquad \textit{with } fv(A(\tilde{x}) \triangleq P) = \{x_1,\ldots,x_n\} \\
[\![A(\tilde{x}) \triangleq P]\!]_\emptyset &= A(\tilde{x}) \triangleq [\![P]\!]_{\tilde{x}} \\
[\![P_1 \mid P_2]\!]_{\tilde{n}} &= [\![P_1]\!]_{\tilde{n}} \mid [\![P_2]\!]_{\tilde{n}} \\
[\![M_1 + M_2]\!]_{\tilde{n}} &= [\![M_1]\!]_{\tilde{n}} + [\![M_2]\!]_{\tilde{n}} \\
[\![f_1[f_2]!(\tilde{f}).P]\!]_{\tilde{n}} &= (\texttt{fst}\ [\![f_1]\!])[[\![f_2]\!]]!([\![\tilde{f}]\!]).[\![P]\!]_{\tilde{n}} \\
[\![f_1[f_2]?(\tilde{y}).P]\!]_{(\tilde{x})} &= (\texttt{fst}\ [\![f_1]\!])[[\![f_2]\!]]?(\tilde{y}).[\![P]\!]_{(\tilde{x},\tilde{y})} \\
[\![A(f_1,\ldots,f_n)]\!]_{\tilde{n}} &= A([\![f_1]\!],\ldots,[\![f_n]\!]) \\
[\![(\nu x{:}f)P]\!]_{(\tilde{x})} &= (\nu x)A(\tilde{x},\langle x,[\![f]\!]\rangle), \text{ with additional} \\
&\qquad \text{process definition } A(\tilde{x},x) \triangleq [\![P]\!]_{(\tilde{x},x)}
\end{aligned}
$$

Let $\mathscr{L}^-$ be an attribute language of the imperative π-calculus without assignments and let $\mathscr{L}^-[(\texttt{val},\texttt{ref})/(\texttt{snd},\texttt{snd}^*)]$ be the attribute language $\mathscr{L}^-$ where operators `val` and `ref` are replaced by operators `snd` and $\texttt{snd}^*$. The encoding of the imperative π-calculus, $\pi^{imp}(\mathscr{L}^-)$ to the attributed π-calculus, $\pi(\mathscr{L}^-[(\texttt{val},\texttt{ref})/(\texttt{snd},\texttt{snd}^*)])$, is correct with respect to the non-deterministic and the stochastic semantics. The proof is by induction on all the rules of the non-deterministic and the stochastic semantics of the imperative and the attributed π-calculus. It shall be omitted here.

4.3.2 Encoding BioAmbients

In this section, an encoding of BioAmbients with priority into the imperative π-calculus is presented. More precisely, it is shown that BioAmbients with a set of priorities $(R, <_n)$, i.e. n priority levels, can be expressed in $\pi^{imp}(\lambda(R, \mathtt{DIR}, \mathtt{CAP}, =, \mathtt{and})_{<_n})$, the imperative π-calculus with an attribute language that provides n priority levels. That is, the encoding avoids deploying update protocols with higher priority to broadcast changes in global information. The encoding is compositional, i.e. the encoding of a parallel composition yields the encoding of its parts. In BioAmbients, changes in global information occur due to ad-hoc operators for reconfigurations in ambient hierarchies. Thus, the fact that a compositional encoding exists that does not require prioritized update protocols underlines that communication constraints with access to a global imperative store are sufficiently expressive to model dynamic cell structures.

The main idea of the encoding is to represent ambients by channels which refer to pairs that provide information about the ambient. The first component of such a pair is the ambient's name and the second a channel referring to the pair of the parent ambient, see Figure 4.18, row 1, column 1. Thus, ambient pairs represent the hierarchy of nested ambient structures by lists capturing the paths from ambients to their top most parents. The sets `DIR` and `CAP` of the attribute language provide constants to represent communication directions, e.g. `local` or `s2s`, and rearrangement capabilities, e.g. `merge` or `exit`, respectively. Restrictions of process interaction due to their location in ambient struc-

tures are modeled by communication constraints that compare communication directions or rearrangement capabilities and ambient names. Thereby, constraints may access the name of the surrounding ambient or the name of its parent. As Versari (2009) already showed, to judge whether an interaction in BioAmbients may happen, it is not necessary to traverse the ambient hierarchy any further than to the parent of a process' surrounding ambient. Rearrangements in the ambient structure are implemented by changing the values of channels. Since initially all the parallel processes in one ambient access the same channel, a change of the channel's value modifies the ambient information for all these processes in a single step. This idea denotes the essential ingredient in order to avoid prioritized update protocols. Notice that the encoding of `enter` and `exit` rearrangements leads to changes in the length of ambient pair lists since new parent ambients are assigned. Thus, they require a type system that supports recursive types.

The most challenging part of the encoding is ambient merging. After a merging, not only do the channels representing the two compartments have to refer to the same value, but also changes in the information of the fused compartments have to be updated for both channels in a single reduction step. Otherwise, inconsistencies regarding the ability of processes to interact may occur. Instead of introducing additional arrays to store all channels referring to the same ambient and defining loops to update them, the encoding presented here makes use of reference chains: when two ambients merge, the channel representing one of the ambients is assigned as a value to the other. Communication constraints are defined to rely on operator `ref`, such that the pair at the

end of a reference chain is always accessed. Thus, when changing this pair, the ambient information that all the channels in the chain refer to is changed in a single step.

The encoding makes use of auxiliary functions to obtain ambient information from channels, as defined in Figure 4.15. Function `amb` determines the name of an ambient by returning the first component of the pair at the end of a reference chain l. Function `parref` provides the second component of this pair, which is the reference chain referring to the information about the parent of the ambient that l represents. Function `par` combines functions `amb` and `parref` to obtain the name of the parent of the ambient that channel l represents.

Figure 4.16 presents the encoding of BioAmbients processes. It makes use of parameter l which captures the channel representing the ambient that contains the process to encode. For processes on the top level, i.e. those which are not engulfed by an ambient, an additional common ambient is introduced. To each ambient in process P, a fresh channel is assigned. Thereby, it is assumed that channel names are chosen deterministically, such that the number of processes in the result set of $[\![P]\!]_l$ is one, i.e. $\#[\![P]\!]_l = 1$.

Parallel compositions and summations are treated by separately considering their components. Process $\mathbf{0}$ remains unchanged. To encode ν-operators, they are extended by initial value `unit`. Special attention needs to be drawn to the representation of ambients. For each engulfed process $[P]$, two ν-bindings are introduced with scope to the encoding of P. The first introduces a unique name a for the ambient, which is compared to the names of other ambients in com-

$$
\begin{array}{rcl}
\texttt{amb} & =_{df} & \lambda l.\texttt{fst}(\texttt{val}(\texttt{ref}\ l)) \\
\texttt{parref} & =_{df} & \lambda l.\texttt{snd}(\texttt{val}(\texttt{ref}\ l)) \\
\texttt{par} & =_{df} & \lambda l.\texttt{amb}(\texttt{parref}\ l)
\end{array}
$$

Figure 4.15: Auxiliary functions for the encoding of BioAmbients to $\pi^{imp}(\mathscr{L})$.

$$
\begin{array}{rcl}
[\![\mathbf{0}]\!]_l & = & \mathbf{0} \\
[\![P_1 \mid P_2]\!]_l & = & [\![P_1]\!]_l \mid [\![P_2]\!]_l \\
[\![M_1 + M_2]\!]_l & = & [\![M_1]\!]_l + [\![M_2]\!]_l \\
[\![\pi.P]\!]_l & = & [\![\pi]\!]_l.[\![P]\!]_l \\
[\![A(\tilde{x})]\!]_l & = & A(\tilde{x}, l) \\
[\![(\nu x)P]\!]_l & = & (\nu x{:}\texttt{unit})[\![P]\!]_l \\
[\![[P]]\!]_l & = & (\nu a{:}\texttt{unit})(\nu l'{:}\langle a, l\rangle)[\![P]\!]_{l'}, \\
 & & \qquad \text{with fresh } l' \text{ chosen deterministically} \\
[\![A(\tilde{x}) \triangleq P]\!]^0 & = & A(\tilde{x}, l) \triangleq [\![P]\!]_l \\
[\![P]\!]^0 & = & ([\![P]\!]_l, \{l \mapsto \langle a, l'\rangle, l' \mapsto \langle a', \texttt{unit}\rangle, \tilde{x} \mapsto \texttt{unit}\}), \\
 & & \qquad \text{with} \{\tilde{x}\} = fv(P) \cup \bigcup_{D \in \mathscr{D}} fv(D)
\end{array}
$$

Figure 4.16: Encoding of processes from BioAmbients to $\pi^{imp}(\mathscr{L})$.

$$
\begin{array}{rcl}
[\![d\ x?(\tilde{x})]\!]_l & = & x[\lambda e.(e\ d\ l)]?(\tilde{x}) \\
[\![c\ x?]\!]_l & = & x[\lambda e.(e\ c\ l)]?() \\
[\![\mathtt{local}\ x{:}r!(\tilde{x})]\!]_l & = & x[\lambda d\ l.\mathbf{if}(d = \mathtt{local})\ \mathbf{and}\ (\mathtt{amb}\ l = \mathtt{amb}\ l) \\
 & & \quad \mathbf{then}\ r\ \mathbf{else}\ 0]!(\tilde{x}) \\
[\![\mathtt{c2p}\ x{:}r!(\tilde{x})]\!]_l & = & x[\lambda d\ l.\mathbf{if}(d = \mathtt{c2p})\ \mathbf{and}\ (\mathtt{amb}\ l = \mathtt{par}\ l) \\
 & & \quad \mathbf{then}\ r\ \mathbf{else}\ 0]!(\tilde{x}) \\
[\![\mathtt{p2c}\ x{:}r!(\tilde{x})]\!]_l & = & x[\lambda d\ l.\mathbf{if}(d = \mathtt{p2c})\ \mathbf{and}\ (\mathtt{par}\ l = \mathtt{amb}\ l) \\
 & & \quad \mathbf{then}\ r\ \mathbf{else}\ 0]!(\tilde{x}) \\
[\![\mathtt{s2s}\ x{:}r!(\tilde{x})]\!]_l & = & x[\lambda d\ l.\mathbf{if}(d = \mathtt{s2s})\ \mathbf{and}\ (\mathtt{par}\ l = \mathtt{par}\ l) \\
 & & \quad \mathbf{and}\ \mathbf{not}(\mathtt{amb}\ l = \mathtt{amb}\ l) \\
 & & \quad\quad \mathbf{then}\ r\ \mathbf{else}\ 0]!(\tilde{x}) \\
[\![\mathtt{merge}\ x{:}r!]\!]_l & = & x[\lambda c\ l.\mathbf{if}(c = \mathtt{merge})\ \mathbf{and}\ (\mathtt{par}\ l = \mathtt{par}\ l) \\
 & & \quad \mathbf{and}\ \mathbf{not}(\mathtt{amb}\ l = \mathtt{amb}\ l)\ \mathbf{then} \\
 & & \quad\quad \{\mathtt{ref}\ l{:=}\ l; r\}\ \mathbf{else}\ 0]!() \\
[\![\mathtt{enter}\ x{:}r!]\!]_l & = & x[\lambda c\ l.\mathbf{if}(c = \mathtt{enter})\ \mathbf{and}\ (\mathtt{par}\ l = \mathtt{par}\ l) \\
 & & \quad \mathbf{and}\ \mathbf{not}(\mathtt{amb}\ l = \mathtt{amb}\ l)\ \mathbf{then} \\
 & & \quad\quad \{\mathtt{ref}\ l{:=}\ \langle \mathtt{amb}\ l, l\rangle; r\}\ \mathbf{else}\ 0]!() \\
[\![\mathtt{exit}\ x{:}r!]\!]_l & = & x[\lambda c\ l.\mathbf{if}(c = \mathtt{exit})\ \mathbf{and}\ (\mathtt{amb}\ l = \mathtt{par}\ l) \\
 & & \quad \mathbf{then}\ \{\mathtt{ref}\ l{:=}\ \langle \mathtt{amb}\ l, \mathtt{parref}\ l\rangle; r\} \\
 & & \mathbf{else}\ 0]!()
\end{array}
$$

Figure 4.17: Encoding of prefixes from BioAmbients to $\pi^{imp}(\mathscr{L})$.

munication constraints. The second ν-binding creates the channel l' representing the ambient engulfing P. Consequently, instead of channel l, the encoding of P proceeds with channel l'. To capture in which ambient a process definition is instantiated, the parameter lists of defined processes and process definitions are extended with parameter l. The initial step of the encoding $[\![P]\!]_l^0$ defines a process-environment pair for P which introduces an ambient for the top level processes in P and encodes P with the channel l representing that ambient. Since the non-deterministic semantics of the imperative $\pi - calculus$ assumes that on reduction of process-environment pairs (Q,ρ) it holds that $fv(Q,\rho) = \emptyset$, mappings from the free names of P and its definitions $\mathscr{D}$ to the value `unit` are added to the environment.

The basic ideas of encoding BioAmbients prefixes are depicted in Figure 4.18. In columns 2 and 3 the encodings of BioAmbients communications are illustrated. Row 1 depicts the case that two processes in the same ambient try to perform a `local` communication. Initially, the processes were engulfed by separate ambients, such that their encoding is parameterized with different channels l_1 and l_2. However, due to a former merging, the value of l_1 is changed, such that it now refers to l_2, building a reference chain. The encoding defines the constraint of a local communication to check whether (`amb` l_1)=(`amb` l_2). Since function `amb` obtains the first component of a pair at the end of a reference chain, the processes can interact as expected. As for all BioAmbients communications, the values of channels representing ambients remain unchanged.

The second row shows the case of a parent-to-child communication

attempt. In the encoding, the constraint of p2c communication is defined, such that the processes may interact if (par l_1)=(amb l_2). Since l_1 stands for an ambient engulfed by the ambient represented by l_2, the second component of its pair is l_2. Thus, in the same way as amb l_2, par l_1 accesses the first component of the pair of l_2, such that the processes can interact. Possible primer merging is again covered by letting constant amb and par determine the end of reference chains before obtaining values. The constraint of the communication direction c2p is defined symmetrically.

In the third row, two processes attempt to communicate in a s2s context. The constraint for s2s communication is defined, such that processes may interact if their parent ambients equal, i.e. (par l_1)=(par l_2), and, as also discussed in Section 3.3.2 introducing $\pi[@,\neq]$, their surrounding ambients differ, i.e. not (amb l_1 = amb l_2). The second components of the pairs of l_1 and l_2 both equal l, such that in each case function par returns p as the name of the parent. By contrast, their first components differ, such that amb applied to l_1 and l_2 yields a_1 and a_2, respectively. Thus, the processes may communicate.

The encodings of rearrangement attempts are illustrated in column 1. Corresponding communication constraints first check interaction restrictions dependent on the location of processes. In this sense, capabilities merge and enter equal communication direction s2s and capability exit communication direction c2p. Additionally, each constraint consists of a single assignment, which is executed on process communication.

The case of ambient merging is presented in row 1. The channel l_2

representing the ambient engulfing the accepting process is assigned as a value to channel l_1, which stands for the ambient engulfing the process initializing the rearrangement. Notice that by applying function `ref` to l_1 on the left side of the assignment, l_2 is assigned to the end of the reference chain l_1. This ensures that the ambient information, which all channels of the reference chain point to, is changed at once.

In row 2, the case of one ambient (l_1) entering another one (l_2) is depicted. This is implemented by assigning a pair to the end of reference chain l_1, such that the ambient name remains unchanged but the parent is set to channel l_2. Symmetrically, an exit rearrangement is reflected by keeping the first component of the corresponding pair but changing the second component to the channel representing the parent of l_2, see row 3.

Figure 4.17 presents the encoding of prefixes. As a general schema, the encoding leaves the subjects and parameters of prefixes unchanged. Constraint functions are defined, such that the receiver only applies its communication direction c or rearrangement capability d and the channel referring to the information of its ambient l to a function defined by the sender. Consequently, following the ideas described above, the function of the sender side defines the restrictions for interactions and changes in channel values, accordingly. The communication direction, rearrangement capability, and ambient channel of the sender are denoted by c, d, and l, respectively. Those of the receiver are provided by function parameters c, d, and l. As a successful value, the priority level of the encoded sending prefix is always returned.

The proof that this encoding is correct is by structural induc-

tion on all the rules of non-deterministic semantics of BioAmbients and $\pi^{imp}(\lambda(R, \mathtt{DIR}, \mathtt{CAP}, =, \mathtt{and})_{<_n})$. It requires to define a rather weak relation between process-environment pairs: first, processes-environment pairs need to be related that differ only in the length of their reference chains. This can be seen by considering the result of reducing the encoding of a `merge` rearrangement in Figure 4.18, row 1, column 2. It should be possible to identify the obtained reference structure with a process-environment pair, where l_1 directly refers to the pair $\langle a_2, l \rangle$. Second, the structural congruence of BioAmbients allows to remove empty ambients from processes ($[0] \equiv 0$). Thus, processes-environment pairs need to be related if their environments only differ in containing some variables that are neither referred to by their processes nor by the right-hand side of any mapping. Third, the encoding of a BioAmbients process P introduces ν-operators that, according to the non-deterministic semantics, are removed when reducing its encoding. However, these are re-introduced when encoding a process P' that results from reducing P. Thus, process-environment pairs need to be related that only differ by some $\xrightarrow[nd]{new}$ reductions. The definition of this relation and the proof of the correctness of the encoding shall be omitted here.

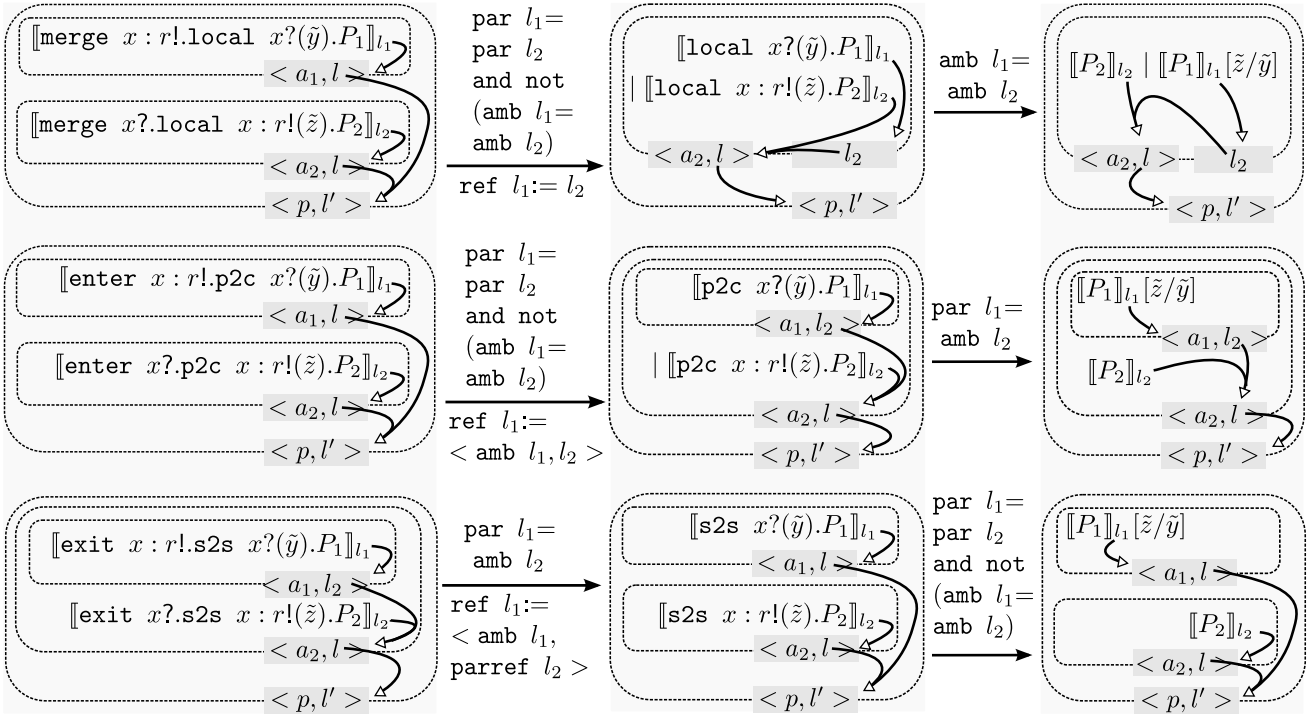


Figure 4.18: Graphical representation of BioAmbients encoding.

4.4 Stochastic Simulator

The stochastic simulator of $\pi^{imp}(\mathscr{L})$ builds on the one of the attributed π-calculus. As usual, the main differences result from the fact that environments for global variables have to be additionally considered. This affects the input of the simulator and all of its steps, see Figure 4.19, in particular the grouping of reactions and the calculation of propensities.

A *group label* $L = (x, v, r, \rho_1, \rho_2)$ of a process-environment pair (P_1, ρ_1) considers two additional environments, the current environment ρ_1 and the environment ρ_2 resulting from evaluating the sender. The latter is needed to ensure that all receivers in group L are evaluated in the same environment, such that they remain fixed in combination with all senders. The group of reactions for $P_1 = \prod_{i=1}^{n} \sum_{j=1}^{m} \pi_i^j.P_i^j$ is defined as follows:

$$\begin{aligned} \mathit{Reacts}(L) \;=\; \{((i_1, j_1, i_2, j_2), r) \in \mathit{Reacts} \mid \\ \exists v' \exists \tilde{y} \exists \tilde{v} \exists \rho_3 \exists \rho_4 . (\pi_{i_1}^{j_1}, \rho_1) \Downarrow (x[v]!(\tilde{y}), \rho_2), \\ (\pi_{i_2}^{j_2}, \rho_2) \Downarrow (x[v']?(\tilde{v}), \rho_3),\ (v'v, \rho_3) \Downarrow (r, \rho_4)\} \end{aligned}$$

The definition of the propensity of L remains unchanged, where $\ell = (i_1, j_1, i_2, j_2)$:

$$\mathit{prop}(L) = \begin{cases} \infty(n), \text{ if } n = \#\{\ell \mid (\ell, \infty) \in \mathit{Reacts}(L)\} \geq 1 \\ \sum_{(\ell,r) \in \mathit{Reacts}(L)} r, \text{ otherwise} \end{cases}$$

The set of grouped reactions with their propensities thus yields:

$$\mathit{GReacts} = \{(L, \mathit{prop}(L)) \mid L \in \mathit{Vars}(P_1) \times \mathit{Vals}(P_1)^2 \times \mathit{Env}^2\}$$

Simulate (P,ρ,t) // *solution P, environment* ρ*, time point* $t \in \mathbb{R}$

let (P_1,ρ_1) be **such that** $(P,\rho) \Downarrow (P_1,\rho_1)$

// (P_1,ρ_1) *is obtained from* (P,ρ) *by exhaustively applying*

// *definitions and eliminating* ν*-operators (may diverge)*

if $(P_1,\rho_1) \xrightarrow[nd]{err} \bot$ **then raise error**

// *apply all rules* (E.COM), (E.PREF), (E.CONSTR)

// *this computation may diverge as expressions are evaluated*

let $GReacts = \{(L, prop(L)) \mid L \in Vars(P_1) \times Vals(P_1)^2 \times Env^2\}$

if $\{(L,r) \in GReacts \mid r = \infty(n)\} = \emptyset$

then

let $((L,r),\Delta) = \mathtt{SSA}(GReacts)$

select $(\ell,r) \in Reacts(L)$ *with equal probability*

let P_2 **such that** $(P_1,\rho_1) \xrightarrow[\ell]{r} (P_2,\rho_2)$

Simulate $((P_2,\rho_2), t+\Delta)$

else

select $(L,\infty(n)) \in GReacts$

with probability n/m *where* $m = \sum_{(L',\infty(n')) \in GReacts} n'$

select $(\ell,\infty) \in Reacts(L)$ *with equal probability*

let (P_2,ρ_2) **such that** $(P_1,\rho_1) \xrightarrow[\ell]{\infty} (P_2,\rho_2)$

Simulate $((P_2,\rho_2), t)$

Figure 4.19: Stochastic simulator for $\pi(\mathscr{L})$ (to be implemented incrementally).

In order to additionally consider environments, the computation of the propensities of L needs to be changed to:

$$\begin{aligned}
out(x,v,\rho_1,\rho_2) &= \#\{(i,j) \mid \exists \tilde{v}.\ (\pi_i^j,\rho_1) \Downarrow (x[v]!(\tilde{v}),\rho_2)\} \\
in(x,v,r,\rho_1) &= \#\{(i,j) \mid \exists v'\tilde{y}\rho_2\rho_3.(\pi_i^j,\rho_1) \Downarrow (x[v']?(\tilde{y}),\rho_2), \\
&\qquad (v'v,\rho_2) \Downarrow (r,\rho_3)\} \\
mixin(x,v,r,\rho_1,\rho_2) &= \#\{(i,j_1,j_2) \mid \\
&\qquad \exists v'\tilde{v}\tilde{y}\rho_3\rho_4.(\pi_i^j,\rho_1) \Downarrow (x[v]!(\tilde{v}),\rho_2), \\
&\qquad (\pi_i^j,\rho_2) \Downarrow (x[v']?(\tilde{y}),\rho_3), \\
&\qquad (v'v,\rho_3) \Downarrow (r,\rho_4)\}
\end{aligned}$$

Lemma 10. *$prop(x,v,r,\rho_1,\rho_2) = (out(x,v,\rho_1,\rho_2) * in(x,v,r,\rho_1) - mixin(x,v,r,\rho_1,\rho_2)) * r$, if the solution does not contain infinite rates.*

Proof. Let $L = (x,v,r,\rho_1,\rho_2)$. It is enough to show that $out(x,v,\rho_1,\rho_2) * in(x,v,r,\rho_1) - mixin(x,v,r,\rho_1,\rho_2) = \#Reacts(L)$. This holds, since all pairs of indices counted by $out(x,v,\rho_1,\rho_2) * in(x,v,r,\rho_1)$ form a redex according to rule (COM), except for those that are counted by $mixin(x,v,r,\rho_1,\rho_2)$. □

The differences in the computational complexity between the simulator of $\pi^{imp}(\mathscr{L})$ and the one of $\pi(\mathscr{L})$ basically depend on the number of assignments in the model. In fact, considering a model not including any assignments, the computational complexity is the same, since senders and receivers can be evaluated separately. A simulator that incrementally computes propensities for $\pi^{imp}(\mathscr{L})$ requires more implementation effort than for $\pi(\mathscr{L})$, since in each step the changes to

the environment need to be traced and prefixes re-evaluated, accordingly. This is also the way the current implementation of the $\pi^{imp}(\mathscr{L})$ simulator works.

Chapter 5

Conclusion

In this thesis, reaction constraints for the π-calculus were proposed as a concept for the spatial and stochastic modeling of cell-biological processes. For the modeling of changes in global information, especially that resulting from dynamic cell structures, two concepts were introduced: priority and global variables. Several formal expressiveness studies and small modeling examples showed the usefulness of both concepts and also of reaction constraints in the π-calculus in general. Simulators of sufficient performance and type systems were developed that ensure the practicability of the concept. A first study that applies the imperative π-calculus to study the Wnt/β-catenin signaling pathway is underway, see Mazemondet et al. (2009). There, the population-based modeling style in particular is deployed, revealing new insights into the applicability of the concept. However, whether reaction constraints in the π-calculus are going to be a well-established approach in the field of modeling cell-biological processes in general

remains an open question. The following concluding remarks of this thesis shall provide an outlook.

The π-calculus subscribes to the individual-based modeling style. Individual-based approaches allow for more sophisticated modeling, since species attributes, such as the occupation of binding sites or molecule positions, can be explicitly considered. However, this leads to a state space explosion, since for each combination of attribute values a new species is introduced. Population-based approaches are especially interesting when trying to keep the state space small. This makes it possible to support a broader range of analysis methods, including those considering entire state spaces. In particular, Bio-PEPA exploits this idea. Novel analysis methods, e.g. statistical model checking, explicitly aim at exploring large stater spaces, thus supporting the applicability of individual-based approaches.

The π-calculus subscribes to the object-centered modeling style. Currently, due to its closeness to the domain, it is rather the rule-based style, as supported by the κ-calculus or LBS, which asserts itself as the dominant paradigm for the modeling of cell-biological processes. However, object-centered modeling seems to be especially useful when studying aspects at the gene level, e.g. as in the co-operative enhancement example in Section 3.2.2. Models of genetic transcription make use of the metaphor of a reader traversing a list of gene sequences, which is very close to what Milner (1999) introduced as mobility in the π-calculus. Object-centered approaches are well-established in computer science, such that programming and thus modeling seems to largely benefit from it. Therefore, it is very well pos-

sible that both modeling styles will find their application domain, just as it is the case e.g. for functional and object-centered programming. Which language then is used for a specific modeling scenario depends not only on the system under study and the question one wishes to answer but also on personal preferences and other "soft" criteria.

A long lasting and still vital discussion is on the question of whether formal languages in general will ever become largely established in the area of modeling cell-biological systems. Most models so far have been implemented in ODE's, since they provide a well-established theory including a broad range of analysis techniques. They also offer almost unlimited possibilities to abstract from the system under study in order to deal with missing knowledge, see e.g. Mirams et al. (2010). Yet, this also comes with a major drawback; models may be over-abstracted and over-fitted to a certain behavior, lacking foundation in the realm of the actual ongoing processes. They are fixed to a narrow scope, which raises questions about the validity of the obtained results. Here, the syntactic layers of modeling languages may help, since they restrict modelers to a certain set of operators that usually find their counterparts in the systems under study, e.g. reaction rules. In the optimal case, the syntax of a modeling language provides an interface for the communication between life and computer scientists. Furthermore, concepts from the field of programming languages, like modularity and abstraction, as e.g. offered by LBS, and typing tailored to the cell-biological realm, see e.g. Fages and Soliman (2008), are of fundamental use in order to avoid error-prone models. Moreover, formal semantics can be defined that allow applying a broader

range of analysis methods, see e.g. Calzone et al. (2006).

Nevertheless, an essential step that has to be taken in order to fully establish the π-calculus in the realm of modeling cell-biological systems is to find ways of defining reactions with more than two reactants in an individual-based style. Although Gillespie (1977) states that reactions with more than two reactants do not occur in reality, abstractions comprising multiple reactants are of great importance. On one hand, they support the modeling in that they allow for more compact models and dealing with missing parameters. On the other hand, especially for stochastic simulation, where multiple runs with many events have to be performed, abstractions help to decrease computational costs. In the π-calculus, multiple reactants are critical, since only pairs of senders and receivers are considered. Versions featuring broadcast communication exist but do not solve the problem. The reason is that it is not all the receivers that listen on some channel that have to be reduced in a single step. It is rather necessary to select a certain number of instances of each species involved in the reaction. In this regard, the question of if CTMC's are the right formalism to define the stochastic semantics for the π-calculus also has to be addressed, since not all abstractions preserve the Markov property. Mura et al. (2009), e.g., consider a stochastic semantics with general probability distributions.

Once one succeeds in extending the π-calculus with n-ary reactions, the question arises if global variables are still needed. It seems also possible to model global side effects by introducing additional reactants and products. Consider e.g. the following two reaction schemes,

where $x \in \mathbb{N}$ is a global variable:

$$\begin{array}{lll} \mathrm{r}_1: & \mathsf{R}_1 & \xrightarrow{\{x:=\,\mathtt{val}\ x\ +1;\ \mathtt{k}_1\}} \quad \mathsf{P}_1 \\ \mathrm{r}_2: & \mathsf{R}_1 & \xrightarrow{\mathtt{k}_2\ *\ \mathtt{val}\ x} \quad \mathsf{P}_2 \end{array} \qquad \textit{with global variable } x \in \mathbb{N}$$

As a side effect, the constraint of reaction r_1 increases x, which directly reflects the constraint of reaction r_2. Introducing a species $\mathsf{X}(x)$, of which only one instance ever exists, this can be modeled without global variables as follows:

$$\begin{array}{l} \textit{assumption: amount of } \mathsf{X}(x) \textit{ is constantly one} \\ \mathrm{r}_1: \quad \forall x \in \mathbb{N}.\mathsf{R}_1, \mathsf{X}(x) \quad \xrightarrow{\mathtt{k}_1} \quad \mathsf{P}_1, \mathsf{X}(x\ +1) \\ \mathrm{r}_2: \quad \forall x \in \mathbb{N}.\mathsf{R}_2, \mathsf{X}(x) \quad \xrightarrow{\mathtt{k}_2\ *\ \mathtt{x}} \quad \mathsf{P}_2, \mathsf{X}(x) \end{array}$$

Kuttler et al. (2010) first applied this idea to model side effects resulting from the interaction of two actors in the context of transcriptional attenuation. However, a formal expressiveness study has not been carried out so far.

In order to obtain a modeling language with optimal expressiveness, a formal study is also missing comparing global variables to priority. In Section 4.3.1, it has been shown that the imperative π-calculus with an attribute language that does not provide assignments can be encoded in the attributed π-calculus. Moreover, it has been proven that both priority and global variables are sufficiently expressive to model changes in global information. However, it is not clear yet in what sense the two concepts differ. It seems to be possible to encode the changes in the values of global variables by using prioritized update protocols, by this showing that priority is at least as expressive as global variables. Vice

versa, based on the idea of Versari (2009) to use the Last Man Standing Problem to separate π-calculi with and without priority, it might be possible to separate priority from the global imperative store. In this case, the latter, less expressive approach would be preferable, since it is sufficient for the spatial modeling of cell-biological processes and avoids error-prone update protocols.

Other limitations to overcome regard not only the π-calculus but formal languages for the spatial and stochastic modeling of cell-biological processes in general.

Abstracting molecular motion as discrete events requires determining rate constants for reactions in many different conditions, e.g. dependent of molecular density or temperature. Takahashi et al. (2005) stated that effects like molecular crowding, where molecular density is very high, are known to have crucial impact on cellular motion and thus on the final outcome of cell biological systems. To this end, Haack et al. (2010) propose to computationally determine rate constants by exhaustive simulation of models that consider deterministic, continuous motion of molecules and their collision. An alternative would be to define a modeling language with a stochastic semantics that supports continuous molecular motion, including molecule size and collision. However, a way of describing continuous motion that fulfills the Markov property has not been found so far, implying again that a semantics supporting general probability distributions might be necessary. Moreover, to ensure practicability regarding the computational complexity of such an approach, an appropriate abstraction level has to be found that avoids accounting for every single molecule as an

individual. First ideas in this regard have already been presented by Jeschke and Uhrmacher (2008).

Different levels of abstraction can underlie the study of cell-biological processes, ranging from molecules over cell organelles to entire cell populations. Multi-level modeling approaches enable the modeler to explicitly reflect these abstraction levels and related hierarchies. By this they allow on one hand describing phenomena more closely to the way they have been observed and on the other hand to abstract from details of low relevance. For example, dependent on their role in the system under study, some proteins are regarded only as single entities, whereas for others different parts need to be identified. Uhrmacher et al. (2007) introduce ML-DEVS as a first approach that explicitly aims at the multi-level modeling of cell-biological systems but does not provide a syntactic layer or a stochastic semantics. Other multi-level approaches rather focus on spatial hierarchies, like BioAmbients, or, as Beta-binders and BlenX, on ways to precisely distinguish between proteins and their environment.

Finally, while developing more and more modeling languages, it also has to be addressed more precisely what aspects of cell-biological processes should be made available by a language. A first step has already been made by Regev et al. (2004), pointing out that dynamic cell structures as in BioAmbients are of interest. Their investigations form the foundation of the expressiveness studies of Versari (2009) and of this thesis. However, for an overall picture further results are necessary. These may be, in particular, obtained through interdisciplinary work between experts from life science and computer science.

Appendix A

Experiment Results

Levels	StoPi/SPiM/Enum	StoPi/James/Enum	AttrPi/James/Enum	AttrPi/James/Comp
10	8.10 (0.09)	7.33 (0.58)	8.00 (0.00)	8.00 (0.00)
20	19.12 (0.12)	16.00 (0.00)	16.33 (0.58)	17.33 (0.58)
30	30.42 (0.12)	24.00 (1.00)	25.67 (0.58)	27.00 (0.00)
40	43.37 (0.13)	35.00 (1.00)	34.33 (0.58)	35.33 (0.58)
50	56.99 (0.18)	43.00 (1.73)	45.00 (2.00)	44.67 (2.00)
60	73.40 (0.20)	56.67 (1.53)	56.33 (0.58)	58.33 (1.16)
70	92.32 (0.34)	68.00 (1.73)	66.00 (0.00)	67.00 (3.00)
80	107.41 (0.16)	78.67 (2.31)	75.00 (2.65)	79.33 (1.53)
90	132.77 (0.26)	86.33 (2.52)	89.00 (1.73)	89.67 (1.53)
100	156.13 (2.16)	104.00 (2.65)	106.67 (1.15)	102.00 (1.00)

Table A.1: Runtime of different simulators in s for the Euglena model with parameters $\mathtt{l}_1 = 5.0$, $\mathtt{l}_2 = 15.0$, $\mathtt{n} = 100$, $\sigma = 0.2$, $\mathtt{u} = 2.0$, and $\mathtt{m} \in \{9, 19, \ldots, 99\}$, i.e. number of depth levels ranging between 0 and 100. Simulation runs were performed until simulation time $t = 100.0$ (average of 3 runs each): "StoPi" = the stochastic π-calculus, "AttrPi" = the attributed π-calculus, "SPiM" = SPiM, "James" = JAMES II, "Enum" = model with enumerated depth levels, "Comp" = model with depth level as species parameter.

Appendix B

Remaining Proofs

B.1 Section 3.1.7 (Type System)

Corollary 2 (Error freeness). *If $\mathscr{L}$ is an attribute language that is both type safe and normalizing (see Propositions 7 and 8) then $\pi(\mathscr{L})$ is error free, i.e. for all processes P with definitions $\mathscr{D}$ in the attributed π-calculus, it holds that if $\Gamma \vdash P$, $\Gamma \vdash \mathscr{D}$, and $P \to^* Q$ then $\neg Q \xrightarrow[nd]{err} \bot$.*

Proof. Assuming that $\Gamma \vdash P$, $\Gamma \vdash \mathscr{D}$, and $P \to^n Q$ the proof proceeds by induction on n. The induction step follows from Theorem 1. It thus remains to prove the initial case that is $P \equiv Q$. Assume by contradiction that there exists some process P_0 such that $\Gamma \vdash P_0$ and $P_0 \xrightarrow[nd]{err} \bot$. A standard analysis of the structural congruence shows the following claim: let $P_0 \equiv (\nu \tilde{x}{:}\tilde{\tau}) \prod_{i=1}^{n} P_i$ be a prenex normal form in which all bound variables are named distinctly, and such that all P_i are sums or defined processes. A derivation of $P_0 \xrightarrow[nd]{err} \bot$ necessarily involves one

of the error axioms given in Figure 3.4. In the following it is shown by case analysis that none of these is applicable:

(E.COM) In that case, it holds that $\exists j,k.1 \leq j < k \leq n$, $P_j = \pi_1.P_1 + M_1$, $P_k = \pi_2.Q_2 + M_2$, $\pi_1 \Downarrow x[v_1]?(\tilde{y})$, $\pi_2 \Downarrow x[v_2]!(\tilde{v})$, and $|\tilde{y}| \neq |\tilde{v}|$. From $\Gamma \vdash P_0$ it follows, by Lemma 3(4), that $\Gamma \vdash (\nu \tilde{x}{:}\tilde{\tau}) \prod_{i=1}^{n} P_i$, which derives from a series of applications of rules (T.NEW) and (T.PAR) and from statements $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash P_i$, for all $i \in \{1,\ldots,n\}$. In particular, it is true that $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash \pi_1.Q_1 + M_1$ and $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash \pi_2.Q_2 + M_2$. By (T.REC), it holds that $\pi_1 = e_1[e_1']?\tilde{y}$, $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash e_1{:}[\tau_1] \Rightarrow \tilde{\sigma_1}$, $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash e_1'{:}\tau_1$, and $|\tilde{y}| = |\tilde{\sigma_1}|$. Similarly it is true that $\pi_2 = e_2[e_2']!\tilde{e_2}''$, $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash e_2{:}[\tau_2 \to \tau_2'] \Rightarrow \tilde{\sigma_2}$, $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash e_2'{:}\tau_2$, and $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash \tilde{e_2}''{:}\tilde{\sigma_2}$. Because it holds that $e_1[e_1']?\tilde{y} \Downarrow x[v_1]?(\tilde{y})$ and $e_2[e_2']!\tilde{e_2}'' \Downarrow x[v_2]!(\tilde{v})$ and, by Proposition 7, e_1 and e_2 have the same type, it follows that $\tau_1 = \tau_2 \to \tau_2'$ and $\tilde{\sigma_1} = \tilde{\sigma_2}$. Because it is true that $\tilde{e_2}'' \Downarrow \tilde{v}$, by Proposition 7, it holds that $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash \tilde{v} : \tilde{\sigma_2}$, such that $|\tilde{v}| = |\tilde{\sigma_2}| = |\tilde{\sigma_1}| = |\tilde{y}|$, which contradicts $|\tilde{y}| \neq |\tilde{v}|$.

(E.PREF) In this case, it holds that $\exists j.1 \leq j \leq n$, $P_j = \pi_1.P_1 + M_1$, and $\neg\exists \pi_1'.\pi_1 \Downarrow \pi_1'$. Similarly to the previous case, one can show that $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash \pi_1.P_1 + M_1$ which is either derived from rule (T.REC) or (T.SEND). Suppose the latter applies (the case (T.REC) is similar), then $\pi_1 = e_1[e_2]!(\tilde{e_3})$, $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash e_1 : [\tau] \Rightarrow \tilde{\sigma}$, $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash e_2 : \tau$, and $\Gamma, \tilde{x}{:}\tilde{\tau} \vdash \tilde{e_3} : \tilde{\sigma}$. By Propositions 7 and 8, each of the expressions e_1, e_2, and $\tilde{e_3}$ evaluate to some typable value and rule (SEND) is applicable which contradicts $\neg\exists \pi_1'.\pi_1 \Downarrow \pi_1'$.

(E.CONSTR) In this case, it holds that $\exists j,k.1 \leq j < k \leq n$, $P_j = \pi_1.P_1 + M_1$, $P_k = \pi_2.Q_2 + M_2$, $\pi_1 \Downarrow x[v_1]?(\tilde{y})$, $\pi_2 \Downarrow x[v_2]!(\tilde{v})$, and $\neg\exists v.v_1 v_2 \Downarrow v$. Similarly to the case (E.COM) one can show that v_1 has some type $\tau_1 \to \tau_2$ and v_2 is of type τ_1. Thus, by rule (T.FUNAPP), $v_1 v_2$ is typable with type τ_2 and, by Proposition 8, evaluates to some value v, which contradicts $\neg\exists v.v_1 v_2 \Downarrow v$.

□

B.2 Section 3.3.1 (Encoding the π-Calculus with Priority)

Theorem 2. *The encoding of the π-calculus with priority levels $(R,<)$ into the attributed π-calculus, $\pi(\lambda(R)_<)$, is correct, in that for all processes P,P' it holds that:*

1. *if $P \to P'$ then $[\![P]\!] \to [\![P']\!]$*
2. *if $[\![P]\!] \to Q$ then there exists a process $\hat{Q}$ in the π-calculus with priority, such that $[\![\hat{Q}]\!] \equiv Q$ and $P \to \hat{Q}$*

Proof. The proof is based on three claims: stating that the encoding is invariant under substitutions, that it preserves and reflects structural congruence and that it preserves and reflects errors, respectively.
Claim. $[\![P[\tilde{v}/\tilde{y}]]\!] = [\![P]\!][\tilde{v}/\tilde{y}]$.
The proof is by induction on the structure of P.
Claim. $P \equiv Q \Leftrightarrow [\![P]\!] \equiv [\![Q]\!]$.

The proof is by structural induction on the derivations of $P \equiv Q$ and $[\![P]\!] \equiv [\![Q]\!]$, respectively. All rules of the structural congruence need to be inspected. Details are omitted here.

Claim. $P \xrightarrow[nd]{err} \bot \Leftrightarrow [\![P]\!] \xrightarrow[nd]{err} \bot$.

Proof.

(E.COM) This case is obvious, since both calculi provide analogous rules. Details are omitted here.

(E.PREF) This rule exists only in the attributed π-calculus. Suppose that $[\![P]\!] \xrightarrow[nd]{err} \bot$ is inferred by (E.PREF):

$$\frac{\neg\exists \pi'.\pi \Downarrow \pi'}{[\![P]\!] = \pi.Q + M \xrightarrow[nd]{err} \bot}$$

Since sums can only be obtained by translating sums, P must match $\hat{\pi}.\hat{Q} + \hat{M}$ for some $\hat{Q}$ and $\hat{M}$. Here, $\hat{\pi}$ must be a prefix of the π-calculus with priority. Thus, it follows that $\pi = [\![\hat{\pi}]\!]$ converges to itself, in contradiction to the hypothesis of the rule.

(E.CONSTR) This rule exists only in the attributed π-calculus. Thus, suppose that (E.CONSTR) infers $[\![P]\!] \xrightarrow[nd]{err} \bot$, such that it is applied as follows:

$$\frac{\pi_1 \Downarrow x[v_1]?(\tilde{y}) \quad \pi_2 \Downarrow x[v_2]!(\tilde{v}) \quad \neg\exists v.v_1 v_2 \Downarrow v}{[\![P]\!] = \pi_1.P_1 + M_1 \mid \pi_2.P_2 + M_2 \xrightarrow[nd]{err} \bot}$$

By inspection of the translation and the first two premises of the rule it holds that P must have the form $x?(\tilde{y}).\hat{P}_1 + \hat{M}_1 \mid$

$x{:}r!(\tilde{z}).\hat{P}_2 + \hat{M}_2$. Thus, $[\![P]\!] = x[\lambda z.z]?(\tilde{y}).P_1 + M_1 \mid x[r]!(\tilde{z}).P_2 + M_2$. This, however, contradicts the third premise, since $v_1 v_2 = (\lambda z.z)\ r \Downarrow r$ by rule (FUN) and $r \in R$.

The treatment of structural rules is omitted, here.

In order to proof the theorem by induction, it needs to be generalized:

Claim. For any relation $\rho \in \{\Downarrow, \xrightarrow[nd]{app}, \xrightarrow[nd]{r}, \rightarrow \mid r \in R\}$, and all processes $P, \hat{P}$ in the π-calculus with priority it holds that:

1. if $P\ \rho\ P'$ then $[\![P]\!]\ \rho\ [\![P']\!]$.
2. if $[\![\hat{P}]\!] \equiv P$ and $P\ \rho\ Q$ then there exists an attributed process $\hat{Q}$, such that $[\![\hat{Q}]\!] \equiv Q$ and $\hat{P}\ \rho\ \hat{Q}$.

Proof. The claim is proved for all the above relations ρ in the order in which they are given. The proof of point 1 is by structural induction on derivations of $P\ \rho\ P'$. All rules of the non-deterministic semantics of the π-calculus with priority need to be considered:

(COM) This rule yields $P \xrightarrow[nd]{r} P'$ as follows:

$$\frac{|\tilde{y}| = |\tilde{z}|}{P = x?(\tilde{y}).P_1 + M_1 \mid x{:}r!(\tilde{z}).P_2 + M_2 \xrightarrow[nd]{r} P_1[\tilde{z}/\tilde{y}] \mid P_2 = P'}$$

Thus, $[\![P]\!] = x[\lambda y.y]?(\tilde{y}).[\![P_1]\!] + [\![M_1]\!] \mid x[r]!(\tilde{z}).[\![P_2]\!] + [\![M_2]\!]$, such that rule (COM) of the non-deterministic semantics of $\pi(\lambda(R)_<)$

applies as follows, while using rules (V) and (FUN) of the big-step evaluator:

$$\frac{\begin{array}{c} x[\lambda y.y]?(\tilde{y}) \Downarrow x[\lambda y.y]?(\tilde{y}) \\ x[r]!(\tilde{z}) \Downarrow x[r]!(\tilde{z}) \quad (\lambda y.y)r \Downarrow r \in R \quad |\tilde{y}| = |\tilde{z}| \end{array}}{[\![P]\!] \xrightarrow[nd]{r} [\![P_1]\!][\tilde{z}/\tilde{y}] \mid [\![P_2]\!]}$$

The claim on substitution invariance provides that $[\![P_1[\tilde{z}/\tilde{y}]]\!] \mid [\![P_2]\!] = [\![P']\!]$.

(APP) Suppose the following rule is applicable:

$$\frac{A(\tilde{x}) \triangleq P}{A(\tilde{v}) \xrightarrow[nd]{app} P[\tilde{v}/\tilde{x}]}$$

The substitution claim provides that $[\![P[\tilde{v}/\tilde{x}]]\!] = [\![P]\!][\tilde{v}/\tilde{x}]$. The translation is defined, such that $[\![A(\tilde{x}) \triangleq P]\!] = A(\tilde{x}) \triangleq [\![P]\!]$. Thus, the following rule applies:

$$\frac{A(\tilde{x}) \triangleq [\![P]\!]}{A(\tilde{v}) \xrightarrow[nd]{app} [\![P]\!][\tilde{v}/\tilde{x}]}$$

(PAR) Suppose that the following rule is applicable:

$$\frac{P_1 \xrightarrow[nd]{\beta} P_1'}{P_1 \mid P_2 \xrightarrow[nd]{\beta} P_1' \mid P_2}$$

The induction hypothesis provides that $[\![P_1]\!] \xrightarrow[nd]{\beta} [\![P_1']\!]$. Since the

translation is compositional, the following rule is applicable:

$$\frac{[\![P_1]\!] \xrightarrow[nd]{\beta} [\![P_1']\!]}{[\![P_1 \mid P_2]\!] \xrightarrow[nd]{\beta} [\![P_1' \mid P_2]\!]}$$

(NEW) Suppose that the following rule is applicable:

$$\frac{P \xrightarrow[nd]{\beta} P'}{(\nu x)P \xrightarrow[nd]{\beta} (\nu x)P'}$$

The induction hypothesis provides that $[\![P]\!] \xrightarrow[nd]{\beta} [\![P']\!]$. By the definition of the translation, the following rule is applicable:

$$\frac{[\![P_1]\!] \xrightarrow[nd]{\beta} [\![P_1']\!]}{[\![(\nu x)P]\!] \xrightarrow[nd]{\beta} [\![(\nu x)P']\!]}$$

(STRUC) Suppose that the following rule is applicable:

$$\frac{P \equiv P_1 \quad P_1 \xrightarrow[nd]{\beta} P_2 \quad P_2 \equiv Q}{P \xrightarrow[nd]{\beta} Q}$$

By the claim on the preservation of structural congruence, it is true that $[\![P]\!] \equiv [\![P_1]\!]$ and $[\![P_2]\!] \equiv [\![Q]\!]$. The induction hypothesis provides that $[\![P_1]\!] \xrightarrow[nd]{\beta} [\![P_2]\!]$. Thus, the following rule is applicable:

$$\frac{[\![P]\!] \equiv [\![P_1]\!] \quad [\![P_1]\!] \xrightarrow[nd]{\beta} [\![P_2]\!] \quad [\![P_2]\!] \equiv [\![Q]\!]}{[\![P]\!] \xrightarrow[nd]{\beta} [\![Q]\!]}$$

(CONV) Suppose that the following rule is applicable:

$$\frac{P \xrightarrow[nd]{app}{}^{*} P' \quad P' \equiv (\nu\tilde{x}) \prod_{i=1}^{n} M_i \quad \neg P' \xrightarrow[nd]{err} \bot}{P \Downarrow P'}$$

By the claim above and the induction hypothesis, the translation preserves structural congruence and application steps, such that $[\![P]\!] \xrightarrow[nd]{app}{}^{*} [\![P']\!]$. The claim on error preservation provides that $\neg[\![P']\!] \xrightarrow[nd]{err} \bot$. Thus, the following rule is applicable:

$$\frac{[\![P]\!] \xrightarrow[nd]{app}{}^{*} [\![P']\!] \quad [\![P']\!] \equiv (\nu\tilde{x}) \prod_{i=1}^{n} [\![M_i]\!] \quad \neg[\![P']\!] \xrightarrow[nd]{err} \bot}{[\![P]\!] \Downarrow [\![P']\!]}$$

(PRIOR) Suppose the following rule is applicable:

$$\frac{P \Downarrow P' \quad P' \xrightarrow[nd]{r} Q \quad \neg\exists r_1 \in R. \exists Q_1.\ r < r_1 \wedge P' \xrightarrow[nd]{r_1} Q_1}{P \to Q}$$

The induction hypothesis provides that $P \Downarrow P'$ yields $[\![P]\!] \Downarrow [\![P']\!]$ and that $P' \xrightarrow[nd]{r} Q$ implies $[\![P']\!] \xrightarrow[nd]{r} [\![Q]\!]$. It can be shown by contradiction that if $\neg\exists r_1 \in R. \exists Q_1. r < r_1 \wedge P \xrightarrow[nd]{r_1} Q_1$ then $\neg\exists r_2. \exists Q_2. r < r_2 \wedge [\![P]\!] \xrightarrow[nd]{r_2} Q_2$. Suppose that $\neg\exists r_1 \in R. \exists Q_1. r < r_1 \wedge P \xrightarrow[nd]{r_1} Q_1$ but $\exists r_2. \exists Q_2. r < r_2 \wedge [\![P]\!] \xrightarrow[nd]{r_2} Q_2$. Reduction $[\![P]\!] \xrightarrow[nd]{r_2} Q'$ is only possible if rule (COM) applies to $[\![P]\!]$, which is true if and only if $[\![P]\!] \equiv (\nu\tilde{x})(\ldots \mid x[r_2]!(\tilde{y}).P_1 + M_1 \mid x[\lambda y.y]?(\tilde{z}).P_2 + M_2 \mid \ldots)$. By the definition of the translation, this is fulfilled only if $P \equiv (\nu\tilde{x})(\ldots \mid x{:}r_2!(\tilde{y}).\hat{P}_1 + \hat{M}_1 \mid x?(\tilde{z}).\hat{P}_2 + \hat{M}_2 \mid \ldots)$. Thus,

$P \xrightarrow[nd]{r_2} Q$ exists, which contradicts the initial assumption. Thus, the following rule is applicable:

$$\frac{[\![P]\!] \Downarrow [\![P']\!] \quad [\![P']\!] \xrightarrow[nd]{r} [\![Q]\!] \quad \neg\exists r_1 \in R.\exists Q_1.\, r < r_1 \wedge [\![P']\!] \xrightarrow[nd]{r_1} Q_1}{[\![P]\!] \to [\![Q]\!]}$$

Proof. The proof of point 2 is by structural induction on derivations of $P \;\rho\; Q$, under the assumption that $[\![\hat{P}]\!] \equiv P$. All rules of the non-deterministic semantics of $\pi(\lambda(R)_<)$ need to be considered:

(COM) By assumption, it holds that $[\![\hat{P}]\!] \equiv P$ and $P \xrightarrow[nd]{r} Q$ by applying the following rule:

$$\frac{\pi_1 \Downarrow x[v_1]?(\tilde{y}) \quad \pi_2 \Downarrow x[v_2]!(\tilde{v}) \quad v_1 v_2 \Downarrow r \in R \quad |\tilde{v}| = |\tilde{y}|}{P = \pi_1.P_1 + M_1 \mid \pi_2.P_2 + M_2 \xrightarrow[nd]{r} P_1[\tilde{v}/\tilde{y}] \mid P_2 = Q}$$

Inspecting the translation reveals that $\hat{P} \equiv \hat{P}'$ for some process $\hat{P}' = x?(\tilde{y}).\hat{P}_1 + \hat{M}_1 \mid x{:}r'!(\tilde{v}).\hat{P}_2 + \hat{M}_2$, where $\pi_1 = x[\lambda z.z]?(\tilde{y})$, $\pi_2 = x[r']!(\tilde{v})$, $[\![\hat{P}_1]\!] \equiv P_1$, and $[\![\hat{P}_2]\!] \equiv P_2$. Prefix evaluation yields $v_1 = \lambda z.z$ and $v_2 = r'$. From $v_1 v_2 = (\lambda z.z)r' \Downarrow r'$ it follows that $r = r'$ by rules (V) and (FUN). Let $\hat{Q} = \hat{P}_1[\tilde{v}/\tilde{y}] \mid \hat{P}_2$, such that $[\![\hat{Q}]\!] = Q$ by the substitution claim. Thus, rules (COM) and (STRUC) apply as follows:

$$\frac{\hat{P} \equiv \hat{P}' \quad \dfrac{|\tilde{y}| = |\tilde{v}|}{\hat{P}' = x?(\tilde{y}).\hat{P}_1 + \hat{M}_1 \mid x{:}r!(\tilde{v}).\hat{P}_2 + \hat{M}_2 \xrightarrow[nd]{r} \hat{P}_1[\tilde{v}/\tilde{y}] \mid \hat{P}_2 = \hat{Q}} \quad \hat{Q} \equiv \hat{Q}}{\hat{P} \xrightarrow[nd]{r} \hat{Q}}$$

(APP) By assumption, it holds that $[\![\hat{P}]\!] \equiv P$ and $P \xrightarrow[nd]{app} Q$ is inferred as follows:

$$\frac{[\![A(\tilde{x}) \triangleq P_1]\!]}{P = A(\tilde{v}) \xrightarrow[nd]{app} [\![P_1]\!][\tilde{v}/\tilde{x}] = Q}$$

Since $[\![\hat{P}]\!] \equiv A(\tilde{v})$, the translation yields that $\hat{P} = A(\tilde{y})$. The substitution claim shows that $[\![P_1]\!][\tilde{y}/\tilde{x}] = [\![P_1[\tilde{y}/\tilde{x}]]\!]$. Let $\hat{Q} = P_1[\tilde{y}/\tilde{x}]$, such that $[\![\hat{Q}]\!] = Q$. Thus, rule (APP) applies as follows:

$$\frac{A(\tilde{x}) \triangleq P_1}{\hat{P} = A(\tilde{y}) \xrightarrow[nd]{app} P_1[\tilde{y}/\tilde{x}] = \hat{Q}}$$

(PAR) By assumption, it holds that $[\![\hat{P}]\!] \equiv P$ and $P \xrightarrow[nd]{\beta} Q$ is obtained as follows:

$$\frac{P_1 \xrightarrow[nd]{\beta} Q_1}{P = P_1 \mid P_2 \xrightarrow[nd]{\beta} Q_1 \mid P_2 = Q}$$

Since the translation is compositional, the assumption $[\![\hat{P}]\!] \equiv P$ implies the existence of two processes $\hat{P}_1$ and $\hat{P}_2$, such that $\hat{P} \equiv \hat{P}_1 \mid \hat{P}_2$, $[\![\hat{P}_1]\!] \equiv P_1$, and $[\![\hat{P}_2]\!] \equiv P_2$. The induction hypothesis applied to $P_1 \xrightarrow[nd]{\beta} Q_1$ provides the existence of a process $\hat{Q}_1$, such that $\hat{P}_1 \xrightarrow[nd]{\beta} \hat{Q}_1$ and $[\![\hat{Q}_1]\!] \equiv Q_1$. Let $\hat{Q} = \hat{Q}_1 \mid \hat{P}_2$, such that $[\![\hat{Q}]\!] = [\![\hat{Q}_1]\!] \mid [\![\hat{P}_2]\!] \equiv Q_1 \mid P_2 = Q$. Reduction $\hat{P} \xrightarrow[nd]{\beta} \hat{Q}$ can be obtained as follows by rules (PAR) and (STRUC):

$$\frac{\hat{P} \equiv \hat{P}_1 \mid \hat{P}_2 \qquad \dfrac{\hat{P}_1 \xrightarrow[nd]{\beta} \hat{Q}_1}{\hat{P}_1 \mid \hat{P}_2 \xrightarrow[nd]{\beta} \hat{Q}_1 \mid \hat{P}_2} \qquad \hat{Q}_1 \mid \hat{P}_2 \equiv \hat{Q}}{\hat{P} \xrightarrow[nd]{\beta} \hat{Q}}$$

(NEW) By assumption, it holds that $[\![\hat{P}]\!] \equiv P$ and $P \xrightarrow[nd]{\beta} Q$ is obtained as follows:

$$\frac{P_1 \xrightarrow[nd]{\beta} Q_1}{P = (\nu x)P_1 \xrightarrow[nd]{\beta} (\nu x)Q_1 = Q}$$

The definition of the translation provides that there exists a process $\hat{P}_1$, such that $\hat{P} \equiv (\nu x)\hat{P}_1$ and $P_1 \equiv [\![\hat{P}_1]\!]$. By induction hypothesis, there exists a process $\hat{Q}_1$, with $\hat{P}_1 \xrightarrow[nd]{\beta} \hat{Q}_1$ and $Q_1 \equiv [\![\hat{Q}_1]\!]$. Let $\hat{Q} = (\nu x)\hat{Q}_1$. Hence, it holds that $[\![\hat{Q}]\!] \equiv Q$ by the definition of the translation. Thus, $\hat{P} \xrightarrow[nd]{\beta} \hat{Q}$ can be inferred as follows:

$$\frac{\hat{P} \equiv (\nu x)\hat{P}_1 \qquad \dfrac{\hat{P}_1 \xrightarrow[nd]{\beta} \hat{Q}_1}{(\nu x)\hat{P}_1 \xrightarrow[nd]{\beta} (\nu x)\hat{Q}_1} \qquad (\nu x)\hat{Q}_1 \equiv \hat{Q}}{\hat{P} \xrightarrow[nd]{\beta} \hat{Q}}$$

(STRUC) By assumption, it holds that $[\![\hat{P}]\!] \equiv P$ and $P \xrightarrow[nd]{\beta} Q$ is inferred as follows:

$$\frac{P \equiv P_1 \quad P_1 \xrightarrow[nd]{\beta} P_2 \quad P_2 \equiv Q}{P \xrightarrow[nd]{\beta} Q}$$

The transitivity of structural congruence provides that $[\![\hat{P}]\!] \equiv P_1$. The induction hypothesis applied to $P_1 \xrightarrow[nd]{\beta} P_2$ thus proves the existence of a process $\hat{P}_2$, such that $\hat{P} \xrightarrow[nd]{\beta} \hat{P}_2$ and $[\![\hat{P}_2]\!] \equiv P_2$. Transitivity of structural congruence yields $[\![\hat{P}_2]\!] \equiv Q$. Thus, $\hat{Q}$ can be defined, such that $\hat{Q} = \hat{P}_2$, $\hat{P} \xrightarrow[nd]{\beta} \hat{Q}$, and $[\![\hat{Q}]\!] \equiv Q$.

(CONV)

Claim. For all $l \in \mathbb{N}_0$ it holds that if $[\![\hat{P}]\!] \equiv P$ and $P(\xrightarrow[nd]{app})^l Q_l$ then there exists $\hat{Q}_l$, such that $\hat{P}(\xrightarrow[nd]{app})^l \hat{Q}_l$ and $Q_l \equiv [\![\hat{Q}_l]\!]$.

Proof. For $l = 0$ the assumption $P(\xrightarrow[nd]{app})^0 Q_0$ is equivalent to $P \equiv Q_0$ by definition. Thus, it is true that $[\![\hat{P}]\!] \equiv Q_0$, such that it is possible to define $\hat{Q}_0 = \hat{P}$ in order to obtain $[\![\hat{Q}_0]\!] \equiv Q_0$. For the induction step, let $[\![\hat{P}]\!] \equiv P$, such that $P(\xrightarrow[nd]{app})^l Q_l \xrightarrow[nd]{app} Q_{l+1}$. By induction hypothesis, there exists $\hat{Q}_l$, such that $\hat{P}(\xrightarrow[nd]{app})^l \hat{Q}_l$ and $Q_l \equiv [\![\hat{Q}_l]\!]$. Since the theorem holds for reduction relation $\xrightarrow[nd]{app}$, there exists $\hat{Q}_{l+1}$, such that $\hat{Q}_l \xrightarrow[nd]{app} \hat{Q}_{l+1}$ and $Q_{l+1} \equiv [\![\hat{Q}_{l+1}]\!]$. Clearly $\hat{P}(\xrightarrow[nd]{app})^{l+1} \hat{Q}_{l+1}$.

By assumption, it holds that $P \equiv [\![\hat{P}]\!]$ and $P \Downarrow Q$ is inferred by rule (CONV) as follows:

$$\frac{P \xrightarrow[nd]{app}{}^{*} Q \quad Q \equiv (\nu \tilde{x}) \prod_{i=1}^{n} M_i \quad \neg Q \xrightarrow[nd]{err} \bot}{P \Downarrow Q}$$

By induction hypothesis, there exists $\hat{Q}$, such that $\hat{P} \xrightarrow[nd]{app}{}^{*} \hat{Q}$ and $[\![\hat{Q}]\!] \equiv Q$. The definition of the translation yields that $\hat{Q} \equiv$

$(\nu\tilde{x})\prod_{i=1}^{n}\hat{M}_i$ for some guarded processes $\hat{M}_i$. Since the translation is error-reflecting, $\neg Q \xrightarrow[nd]{err} \bot$ yields $\neg\hat{Q} \xrightarrow[nd]{err} \bot$. Thus, $\hat{P} \Downarrow \hat{Q}$ can be inferred as follows:

$$\frac{\hat{P} \xrightarrow[nd]{app}{}^{*} \hat{Q} \quad \hat{Q} \equiv (\nu\tilde{x})\prod_{i=1}^{n}\hat{M}_i \quad \neg\hat{Q} \xrightarrow[nd]{err} \bot}{\hat{P} \Downarrow \hat{Q}}$$

(PRIOR) By assumption, it holds that $P \equiv [\![\hat{P}]\!]$ and $P \Downarrow Q$ is inferred as follows:

$$\frac{P \Downarrow P_1 \quad P_1 \xrightarrow[nd]{r} Q \quad \neg\exists r_1 \in R.\exists Q_1.\, r < r_1 \wedge P_1 \xrightarrow[nd]{r_1} Q_1}{[\![P]\!] \rightarrow Q}$$

By induction hypothesis, there exists a process $\hat{P}_1$, such that $\hat{P} \Downarrow \hat{P}_1$ and $P_1 \equiv [\![\hat{P}_1]\!]$. Hence, there exists $\hat{Q}$, such that $\hat{P}_1 \xrightarrow[nd]{r} \hat{Q}$ and $Q \equiv \hat{Q}$. Next, it is shown that $\neg\exists r_1 \in R.\exists\hat{Q}_1.r < r_1 \wedge \hat{P}_1 \xrightarrow[nd]{r_1} \hat{Q}_1$, by contradiction. Suppose that such an r_1 and $\hat{Q}_1$ exist. By the first part of this theorem, this implies that $[\![\hat{P}_1]\!] \xrightarrow[nd]{r_1} [\![\hat{Q}_1]\!]$. Since it is true that $P_1 \equiv [\![\hat{P}_1]\!]$, it is possible to define $Q_1 \equiv [\![\hat{Q}_1]\!]$, such that, by rule (STRUC), it holds that $P_1 \xrightarrow[nd]{r_1} Q_1$, which is in contradiction to the third hypothesis.

□

B.3 Section 3.3.2 (Encoding $\pi[@,\neq]$ for Dynamic Compartments)

Lemma 5. *For all constants or variables* $v_1,\ldots,v_n$, $v'_1,\ldots,v'_n$, $b_1,\ldots,b_n$ *it is true that:*

1. $\mathtt{eq}_n v_1 \ldots v_n v'_1 \ldots v'_n b_1 \ldots b_n \Downarrow \mathtt{true}$, *if* $\forall i \in \{1,\ldots,n\}.(v_i = v'_i) \Leftrightarrow (b_i = \mathtt{true})$

2. $\mathtt{eq}_n v_1 \ldots v_n v'_1 \ldots v'_n \Downarrow \mathtt{false}$, *if* $\exists i \in \{1,\ldots,n\}.\neg((v_i = v'_i) \Leftrightarrow (b_i = \mathtt{true}))$

Proof. Functions $\mathtt{eq}_{n+1}$ are defined as:

$$
\begin{aligned}
&\mathtt{eq}_0 \;=_{df}\; \mathtt{true}\\
&\mathtt{eq}_{n+1} \;=_{df}\; \lambda x_1 \ldots \lambda x_{n+1} \lambda y_1 \ldots y_{n+1} \lambda b_1 \ldots b_{n+1}.\\
&\qquad\qquad \mathtt{if}\ (x_{n+1} = y_{n+1}) = b_{n+1}\ \mathtt{then}\ \mathtt{eq}_n\\
&\qquad\qquad\quad x_1 \ldots x_n y_1 \ldots y_n b_1 \ldots b_n\\
&\qquad\qquad \mathtt{else}\ \mathtt{false}
\end{aligned}
$$

The following claim states that the condition in the if-statement correctly reflects the communication condition of $\pi[@,\neq]$ on the elements of channel tuples.

Claim.(Condition) For all constants or variables v, v', b it is true that $(v = v') = b \Downarrow \mathtt{true}$, iff $(v = v') \Leftrightarrow (b_i = \mathtt{true})$ holds.

Proof.

($\Rightarrow$) two cases have to be considered: if $b = \mathtt{true}$ then it holds that $v = v'$, by rules (EQ_1) and (V) of the big-step evaluator as pre-

sented in Figure 3.1. Similarly, if $b = \texttt{false}$ then it holds that $v \neq v'$.

($\Leftarrow$) two cases have to be considered: if $b = \texttt{true}$ then $v = v'$, such that by rules (EQ_1) and (V) of the big-step evaluator it holds that $(v = v') = b \Downarrow \texttt{true}$. The case $b = \texttt{false}$ implies that $v \neq v'$, such that by rules (EQ_1), (EQ_2), and (V) it holds that $(v = v') = b \Downarrow \texttt{true}$.

Next, the two statements of the lemma are proved separately:

(1) The proof is by induction on n. For $n = 0$ it holds that $\texttt{eq}_0 \Downarrow \texttt{true}$ by rule (V) of the big-step evaluator as presented in Figure 3.1. For the induction step, since it is true that $(v_{n+1} = v'_{n+1}) \Leftrightarrow (b_{n+1} = \texttt{true})$, the condition claim above and rule (COND_1) provide that for all values v it holds that $\texttt{eq}_{n+1} v_1 \ldots v_{n+1} v'_1 \ldots v'_{n+1} b_1 \ldots b_{n+1} \Downarrow v$, iff $\texttt{eq}_n v_1 \ldots v_n v'_1 \ldots v'_n b_1 \ldots b_n \Downarrow v$. The induction hypothesis provides that $\texttt{eq}_n v_1 \ldots v_n v'_1 \ldots v'_n b_1 \ldots b_n \Downarrow \texttt{true}$.

(2) The proof is by induction on n. The case $n = 0$ is trivial, since the hypothesis of the implication is always wrong. For the induction step it is assumed that there exists $i \in \{1, \ldots, n+1\}$, such that $\neg((v_i = v'_i) \Leftrightarrow (b_i = \texttt{true}))$. Suppose it is true that $1 \leq i \leq n$ and $(v_{n+1} = v'_{n+1}) \Leftrightarrow (b_{n+1} = \texttt{true})$. Then the condition claim above and rule (COND_1) provide that for all values v it is true that $\texttt{eq}_{n+1} v_1 \ldots v_{n+1} v'_1 \ldots v'_{n+1} b_1 \ldots b_{n+1} \Downarrow v$, iff $\texttt{eq}_n v_1 \ldots v_n v'_1 \ldots v'_n b_1 \ldots b_n \Downarrow v$. By the induction hypothesis it

holds that $\mathtt{eq}_n v_1 \ldots v_n v'_1 \ldots v'_n b_1 \ldots b_n \Downarrow \mathtt{false}$. If $i = n+1$ then the condition claim above and rule (COND$_2$) of the big-step evaluator provide that $\mathtt{eq}_{n+1} v_1 \ldots v_{n+1} v'_1 \ldots v'_{n+1} b_1 \ldots b_{n+1} \Downarrow \mathtt{false}$.

□

Theorem 4. *The encoding of $\pi[@,\neq]$ with priority levels in $(R,<)$ to the attributed π-calculus, $\pi(\lambda(R,=,\mathtt{EQ})_<)$, is correct, in that all preprocessed processes P in $\pi[@,\neq]$ satisfy:*

1. *if $P \to Q$ then $[\![P]\!] \to [\![Q]\!]$*
2. *if $[\![P]\!] \to Q$ then there exists $\hat{Q}$, such that $Q \equiv [\![\hat{Q}]\!]$ and $P \to \hat{Q}$*

Proof. Theorem 2 provides that the encoding of the π-calculus with priority into $\pi(\lambda(R)_<)$ is correct. The non-deterministic semantics of $\pi[@,\neq]$ and the π-calculus with priority are the same, with one exception: the communication rule. Therefore, in the following, only the communication rule is considered.

1. Let κ_o and κ_i as follows:

$$\kappa_o =_{df} \lambda x_1 \ldots \lambda x_n \lambda b_1 \ldots \lambda b_n \lambda r.$$
$$\quad \textbf{if } \mathtt{eq}_{n+1} x_1 \ldots x_n r\ x_1 \ldots x_n r\ b_1 \ldots \lambda b_n \mathtt{true} \textbf{ then } r$$
$$\quad \textbf{else } \mathtt{false}$$
$$\kappa_i =_{df} \lambda e.e\ x_1 \ldots x_n r\ b_1 \ldots b_n$$

Rule (COM$_{[@,\neq]}$) yields $P \xrightarrow[nd]{r} P'$ as follows:

$$\frac{|\tilde{y}| = |\tilde{z}| \quad \forall i \in \{1,\ldots,n\}.(x_i = x'_i) \Leftrightarrow (b_i = \mathtt{true})}{(x_i)_{i=1}^n : (b_i)_{i=1}^n : r?(\tilde{y}).P_1 + M_1 \mid (x'_i)_{i=1}^n : r!(\tilde{z}).P_2 + M_2 \xrightarrow[nd]{r} P_1[\tilde{z}/\tilde{y}] \mid P_2}$$

Thus, $[\![P]\!] = x[\kappa_i]?(\tilde{y}).[\![P_1]\!] + [\![M_1]\!] \mid x[\kappa_o]!(\tilde{z}).[\![P_2]\!] + [\![M_2]\!]$, with some fresh x not occurring in P. Lemma 5 provides that $\kappa_i \kappa_o \Downarrow r$, such that the (COM) rule of $\pi(R,<)$ applies as follows:

$$\frac{x_1[\kappa_i]?(\tilde{y}) \Downarrow x_1[v_1]?(\tilde{y}) \quad x_1[\kappa_o]!(\tilde{z}) \Downarrow x_1[v_2]!(\tilde{z}) \quad v_1 v_2 \Downarrow r \in R \quad |\tilde{y}| = |\tilde{z}|}{[\![P]\!] \xrightarrow[nd]{r} [\![P_1]\!][\tilde{v}/\tilde{y}] \mid [\![P_2]\!]}$$

The claim on substitutions provides that $[\![P_1]\!][\tilde{v}/\tilde{y}] \mid [\![P_2]\!] = [\![P']\!]$.

2. Let κ_o and κ_i as follows:

$$\begin{aligned}
\kappa_o =_{df}\ & \lambda x_1 \ldots \lambda x_n \lambda b_1 \ldots \lambda b_n \lambda r. \\
& \quad \textsf{if } \mathtt{eq}_{n+1} x_1 \ldots x_n r\, x_1 \ldots x_n \hat{r}\, b_1 \ldots \lambda b_n \mathtt{true} \textsf{ then } \hat{r} \\
& \quad \textsf{else } \mathtt{false} \\
\kappa_i =_{df}\ & \lambda e.e\, x_1 \ldots x_n \hat{r}\, b_1 \ldots b_n
\end{aligned}$$

The assumption provides that $[\![\hat{P}]\!] \equiv P$ and $P \xrightarrow[nd]{r} Q$ by applying the following rule:

$$\frac{\pi_1 \Downarrow x[v_1]?(\tilde{y}) \quad \pi_2 \Downarrow x[v_2]!(\tilde{v}) \quad v_1 v_2 \Downarrow r \in R \quad |\tilde{v}| = |\tilde{y}|}{P = \pi_1.P_1 + M_1 \mid \pi_2.P_2 + M_2 \xrightarrow[nd]{r} P_1[\tilde{v}/\tilde{y}] \mid P_2 = Q}$$

Inspecting the translation reveals that there exists $\hat{P}'$, such that $\hat{P} \equiv \hat{P}'$ and $\hat{P}' = (x_1,\ldots,x_n) : (b_1,\ldots,b_n) : \hat{r}?(\tilde{y}).\hat{P}_1 + \hat{M}_1 \mid (x'_1,\ldots,x'_n):\hat{r}'!(\tilde{z}).\hat{P}_2 + \hat{M}_2$, with $[\![\hat{P}_1]\!] \equiv P_1$, $[\![\hat{P}_2]\!] \equiv P_2$, $\pi_1 = x[\kappa_i]?(\tilde{y})$, and $\pi_2 = x[\kappa_o]!(\tilde{z})$. Prefix evaluation yields $v_1 = \kappa_i$ and $v_2 = \kappa_o$. By Lemma 5 it holds that $\forall i \in \{1,\ldots,n\}.(x_i = x'_i) \Leftrightarrow (b_i = \mathtt{true})$, $\hat{r}' = \hat{r}$, and $r = \hat{r}$, since $P \xrightarrow[nd]{r} Q$ by the rule

above. Let $\hat{Q} = \hat{P}_1[\tilde{v}/\tilde{y}] \mid \hat{P}_2$, such that $[\![\hat{Q}]\!] = Q$ by the substitution claim. Rules ($\text{COM}_{[@,\neq]}$) and (STRUC) apply as follows:

$$\dfrac{\hat{P} \equiv \hat{P}' \qquad \dfrac{\begin{array}{c}|\tilde{y}| = |\tilde{z}| \\ \forall i \in \{1,\ldots,n\}.(x_i = x'_i) \Leftrightarrow (b_i = \texttt{true}) \\ \hline (x_i)_{i=1}^n : (b_i)_{i=1}^n : r?(\tilde{y}).P_1 + M_1 \mid \\ (x'_i)_{i=1}^n : r!(\tilde{z}).P_2 + M_2 \end{array} \xrightarrow[nd]{r} P_1[\tilde{z}/\tilde{y}] \mid P_2 \qquad \hat{Q} \equiv \hat{Q}}{\hat{P} \xrightarrow[nd]{r} \hat{Q}}$$

□

Bibliography

Abadi, M. and Fournet, C. (2001). Mobile values, new names, and secure communication. In *POPL*, pages 104–115. ACM.

Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., and Walter, P. (2002). *Molecular Biology of the Cell*. Garland Science, 4 edition.

Alur, R., Courcoubetis, C., Henzinger, T. A., and Ho, P.-H. (1992). Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Grossman, R. L., Nerode, A., Ravn, A. P., and Rischel, H., editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer.

Anai, H., Horimoto, K., and Kutsia, T., editors (2007). *Algebraic Biology, Second International Conference, AB 2007, Castle of Hagenberg, Austria, July 2-4, 2007, Proceedings*, volume 4545 of *Lecture Notes in Computer Science*. Springer.

Baldamus, M., Parrow, J., and Victor, B. (2005). A fully abstract encoding of the pi-calculus with data terms. In Caires, L., Italiano, G. F., Monteiro, L., Palamidessi, C., and Yung, M., editors, *ICALP*,

volume 3580 of *Lecture Notes in Computer Science*, pages 1202–1213. Springer.

Barbuti, R., Caravagna, G., Maggiolo-Schettini, A., Milazzo, P., and Pardini, G. (2008). The calculus of looping sequences. In Bernardo, M., Degano, P., and Zavattaro, G., editors, *SFM*, volume 5016 of *Lecture Notes in Computer Science*, pages 387–423. Springer.

Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., and Pardini, G. (2009). Spatial calculus of looping sequences. *Electronic Notes in Theoretical Computer Science*, 229(1):21–39.

Barendregt, H. P. (1985). *The Lambda Calculus, Its Syntax and Semantics (Studies in Logic and the Foundations of Mathematics, Volume 103). Revised Edition*. North Holland, revised edition.

Bartocci, E., Corradini, F., Berardini, M. R. D., Merelli, E., and Tesei, L. (2009). Shape calculus: A spatial calculus for 3d colliding shapes. Technical Report 1, University of Camerino.

Bortolussi, L. and Policriti, A. (2008a). Hybrid semantics for stochastic pi-calculus. In Horimoto, K., Regensburger, G., Rosenkranz, M., and Yoshida, H., editors, *AB*, volume 5147 of *Lecture Notes in Computer Science*, pages 40–55. Springer.

Bortolussi, L. and Policriti, A. (2008b). Modeling biological systems in stochastic concurrent constraint programming. *Constraints*, 13(1-2):66–90.

Bowman, H. and Gomez, R. (2005). *Concurrency Theory: Calculi an Automata for Modelling Untimed and Timed Concurrent Systems*. Springer, 1 edition.

Brodo, L., Degano, P., and Priami, C. (2007). A stochastic semantics for bioambients. In Malyshkin, V. E., editor, *PaCT*, volume 4671 of *Lecture Notes in Computer Science*, pages 22–34. Springer.

Calder, M. and Gilmore, S., editors (2007). *Computational Methods in Systems Biology, International Conference, CMSB 2007, Edinburgh, Scotland, September 20-21, 2007, Proceedings*, volume 4695 of *Lecture Notes in Computer Science*. Springer.

Calzone, L., Fages, F., and Soliman, S. (2006). Biocham: an environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics*, 22(14):1805–1807.

Cappello, I. and Quaglia, P. (2009). A translation of beta-binders in a prioritized pi-calculus. *Electronic Notes in Theoretical Computer Science*, 229(1):109–125.

Cardelli, L. (2004). Brane calculi. In Danos, V. and Schächter, V., editors, *CMSB*, volume 3082 of *Lecture Notes in Computer Science*, pages 257–278. Springer.

Cardelli, L., Caron, E., Gardner, P., Kahramanogullari, O., and Phillips, A. (2009). A process model of Rho GTP-binding proteins. *Theoretical Computer Science*, 410(33-34):3166–3185.

Cardelli, L. and Gardner, P. (2009). Processes in space. Technical report, Imperial College London.

Chaouiya, C. (2007). Petri net modelling of biological networks. *Briefings in Bioinformatics*, 8(4):bbm029v1–10.

Chebotareva, N. A., Kurganov, B. I., and Livanova, N. B. (2004). Biochemical effects of molecular crowding. *Biochemistry (Moscow)*, 69(11):1239–1251.

Church, A. (1940). A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5(2):56–68.

Ciocchetta, F. and Guerriero, M. L. (2009). Modelling biological compartments in bio-pepa. *Electronic Notes in Theoretical Computer Science*, 227:77–95.

Ciocchetta, F. and Hillston, J. (2009). Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theorical Computer Science*, 410(33-34):3065–3084.

Damm, W. and Harel, D. (2001). Lscs: Breathing life into message sequence charts. *Formal Methods in System Design*, 19(1):45–80.

Danos, V. and Laneve, C. (2004). Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110.

Degenring, D., Röhl, M., and Uhrmacher, A. M. (2004). Discrete event, multi-level simulation of metabolite channeling. *Biosystems*, 75(1-3):29–41. Computational Systems Biology.

Dematté, L., Priami, C., and Romanel, A. (2008a). The beta workbench: a computational tool to study the dynamics of biological systems. *Briefings in Bioinformatics*, 9(5):437–449.

Dematté, L., Priami, C., and Romanel, A. (2008b). Modelling and simulation of biological processes in blenx. *SIGMETRICS Performance Evaluation Review*, 35(4):32–39.

Deutsch, A. and Dormann, S. (2004). *Cellular Automaton Modeling of Biological Pattern Formation*. Birkhäuser Boston, 1 edition.

Eker, S., Knapp, M., Laderoute, K., Lincoln, P., and Talcott, C. L. (2002). Pathway logic: Executable models of biological networks. *Electr. Notes Theor. Comput. Sci.*, 71:144–161.

Ene, C. and Muntean, T. (1999). Expressiveness of point-to-point versus broadcast communications. In Ciobanu, G. and Paun, G., editors, *FCT*, volume 1684 of *Lecture Notes in Computer Science*, pages 258–268. Springer.

Engelmann, T. (1882). Ueber Licht- und Farbenperception niederster Organismen. *Pflügers Archiv European Journal of Physiology*, 29(1):387–400.

Faeder, J. R., Blinov, M. L., Goldstein, B., and Hlavacek, W. S. (2005). Rule-based modeling of biochemical networks. *Complexity*, 10(4):22–41.

Fages, F. and Rizk, A. (2007). On the analysis of numerical data time series in temporal logic. In Calder and Gilmore (2007), pages 48–63.

Fages, F. and Soliman, S. (2008). Abstract interpretation and types for systems biology. *Theoretical Computer Science*, 403(1):52–70.

Fall, C. P., Marland, E. S., Wagner, J. M., and Tyson, J. J. (2002). *Computational Cell Biology*. Springer.

Fisher, J., Piterman, N., Hubbard, Stern, M. J., and Harel, D. (2005). Computational insights into caenorhabditis elegans vulval development. *Proceedings of the National Academy of Sciences of the United States of America*, 102(6):1951–1956.

Gibson, M. A. and Bruck, J. (2000). Efficient exact stochastic simulation of chemical systems with many species and many channels. *Journal of Physical Chemistry*, 104:1876–1889.

Gillespie, D. T. (1976). A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434.

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81:2340–2361.

Guerriero, M. L., Priami, C., and Romanel, A. (2007). Modeling static biological compartments with beta-binders. In Anai et al. (2007), pages 247–261.

Haack, F., Leye, S., and Uhrmacher, A. M. (2010). A flexible architecture for modeling and simulation of diffusional association. *CoRR*, abs/1002.4064:70–84.

Hardy, S. and Robillard, P. (2004). Modeling and simulation of molecular biology systems using petri nets: modeling goals of various approaches. *Journal of Bioinformatics and Computational Biology*, 2(4):595–613.

Harel, D. (1987). Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274.

Harris, L. A., Hogg, J. S., and Faeder, J. R. (2009). Compartmental rule-based modeling of biochemical systems. In Rossetti et al. (2009).

Heiner, M. and Uhrmacher, A. M., editors (2008). *Computational Methods in Systems Biology, 6th International Conference, CMSB 2008, Rostock, Germany, October 12-15, 2008. Proceedings*, volume 5307 of *Lecture Notes in Computer Science*. Springer.

Himmelspach, J. and Uhrmacher, A. M. (2007). Plug'n simulate. In *Annual Simulation Symposium*, pages 137–143. IEEE Computer Society.

Huet, G. P. (1980). Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821.

Jeschke, M. and Uhrmacher, A. M. (2008). Multi-resolution spatial simulation for molecular crowding. In Mason, S. J., Hill, R. R., Mönch, L., Rose, O., Jefferson, T., and Fowler, J. W., editors, *Winter Simulation Conference*, pages 1384–1392. Institute of Electrical and Electronics Engineers, Inc.

Johansson, M., Parrow, J., Victor, B., and Bengtson, J. (2008). Extended pi-calculi. In Aceto, L., Damgård, I., Goldberg, L. A., Halldórsson, M. M., Ingólfsdóttir, A., and Walukiewicz, I., editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 87–98. Springer.

John, M., Ewald, R., and Uhrmacher, A. M. (2008a). A spatial extension to the pi-calculus. *Electronic Notes in Theoretical Computer Science*, 194(3):133–148.

John, M., Lhoussaine, C., Niehren, J., and Uhrmacher, A. M. (2008b). The attributed pi-calculus. In Heiner and Uhrmacher (2008), pages 83–102.

Kam, N., Kugler, H., Marelly, R., Appleby, L., Fisher, J., Pnueli, A., Harel, D., Stern, M. J., and Hubbard, E. J. A. (2008). A scenario-based approach to modeling development: A prototype model of c. elegans vulval fate specification. *Developmental Biology*, 323(1):1 – 5.

Kholodenko, B. N. (2006). Cell-signalling dynamics in time and space. *Nature Reviews Molecular Cell Biology*, 7:165–176.

Khomenko, V. and Meyer, R. (2008). Checking pi-calculus structural congruence is graph isomorphism complete. Technical Report CS-TR: 1100, School of Computing Science, Newcastle University. 20 pages.

Kier, L. B. (2005). *Modeling Chemical Systems Using Cellular Automata*. Springer.

Kitano, H. (2002a). Computational systems biology. *Nature*, 420(6912):206–210.

Kitano, H. (2002b). Systems biology: a brief overview. *Science*, 295(5560):1662–1664.

Klipp, E., Liebermeister, W., Wierling, C., Kowald, A., Lehrach, H., and Herwig, R. (2009). *Systems Biology: A Textbook*. Wiley-VCH, 1 edition.

Krieghoff, E., Behrens, J., and Mayr, B. (2006). Nucleo-cytoplasmic distribution of β-catenin is regulated by retention. *Journal of Cell Science*, 119(Pt 7):1453–1463.

Krivine, J., Milner, R., and Troina, A. (2008). Stochastic bigraphs. *Electronic Notes in Theoretical Computer Science*, 218:73–96.

Kuttler, C. (2006). *Modeling Bacterial Gene Expression in a Stochastic Pi-Calculus with Concurrent Objects*. PhD thesis, University of Lille 1.

Kuttler, C., Lhoussaine, C., and Nebut, M. (2010). Rule-based modeling of transcriptional attenuation at the tryptophan operon. *Transactions on Computaional Systems Biology XII. Special Issue on Modeling Methodologies*, 5945:199–228. LNCS (Lecture Notes in Bioinformatics), Springer Berlin/Heidelberg.

Kuttler, C., Lhoussaine, C., and Niehren, J. (2007). A stochastic pi calculus for concurrent objects. In Anai et al. (2007), pages 232–246.

Kuttler, C. and Niehren, J. (2006). Gene regulation in the pi calculus: Simulating cooperativity at the lambda switch. *Transactions on Computational Systems Biology*, 4230:24–55.

Kwiatkowska, M. Z., Norman, G., and Parker, D. (2004). Prism 2.0: A tool for probabilistic model checking. In *QEST*, pages 322–323. IEEE Computer Society.

Kwiatkowski, M. and Stark, I. (2008). The continuous pi-calculus: A process algebra for biochemical modelling. In Heiner and Uhrmacher (2008), pages 103–122.

Lawler, G. F. (2006). *Introduction to Stochastic Processes, Second Edition (Chapman & Hall/CRC Probability Series)*. Chapman and Hall/CRC, 2 edition.

Leye, S., John, M., and Uhrmacher, A. M. (2010). A flexible architecture for performance experiments with the pi-calculus and its extensions. In Lawson, B., editor, *SIMUTools 2010*. ICST/IEEE.

Matsumoto, M. and Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transaction on Modeling and Computer Simulation*, 8(1):3–30.

Maus, C. (2008). Component-based modelling of rna structure folding. In Heiner and Uhrmacher (2008), pages 44–62.

Mazemondet, O., John, M., Maus, C., Uhrmacher, A. M., and Rolfs, A. (2009). Integrating diverse reaction types into stochastic models - a signaling pathway case study in the imperative pi-calculus. In Rossetti et al. (2009), pages 932–943.

McAdams, H. H. and Arkin, A. (1999). It's a noisy business! Genetic regulation at the nanomolar scale. *Trends in Genetics*, 15(2):65–69.

Merino, E. and Yanofsky, C. (2005). Transcription attenuation: a highly conserved regulatory strategy used by bacteria. *Trends in Genetics*, 21(5):260–264.

Millat, T., Bullinger, E., Rohwer, J., and Wolkenhauer, O. (2007). Approximations and their consequences for dynamic modelling of signal transduction pathways. *Mathematical Biosciences*, 207(1):40–57.

Miller, J. R., Hocking, A. M., Brown, J. D., and Moon, R. T. (1999). Mechanism and function of signal transduction by the Wnt/β-catenin and Wnt/ca2+ pathways. *Oncogene*, 18(55):7860–7872.

Milner, R. (1999). *Communicating and Mobile Systems: The Pi Calculus*. Cambridge University Press.

Milner, R. (2009). *The Space and Motion of Communicating Agents*. Cambridge University Press, 1 edition.

Milner, R., Parrow, J., and Walker, D. (1992). A calculus of mobile processes, i. *Information and Computation*, 100(1):1–40.

Mirams, G., Byrne, H., and King, J. (2010). A multiple timescale analysis of a mathematical model of the Wnt/β-catenin signalling pathway. *Journal of Mathematical Biology*, 60(1):131–160.

Mitchell, J. C. (1996). *Foundations for Programming Languages*. MIT.

Mura, I., Prandi, D., Priami, C., and Romanel, A. (2009). Exploiting non-markovian bio-processes. *Electron. Notes Theor. Comput. Sci.*, 253(3):83–98.

Nagao, M., Sugimori, M., and Nakafuku, M. (2007). Cross talk between notch and growth factor/cytokine signaling pathways in neural stem cells. *Molecular and Cell Biology*, 27(11):3982–3994.

Niehren, J. (2000). Uniform confluence in concurrent computation. *Journal of Functional Programming*, 10(5):453–499.

Palamidessi, C. (2003). Comparing the expressive power of the synchronous and asynchronous pi-calculi. *Mathematical Structures in Computer Science*, 13(5):685–719.

Paun, G. (2000). Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143.

Pedersen, M. and Plotkin, G. (2010). A language for biochemical systems: Design and formal specification. *Transactions on Computational Systems Biology XII*, 5945:77–145.

Petri, C. A. (1962). *Kommunikatin mit Automaten*. PhD thesis, University of Bonn.

Phillips, A. and Cardelli, L. (2007). Efficient, correct simulation of biological processes in the stochastic pi-calculus. In Calder and Gilmore (2007), pages 184–199.

Pierce, B. C. (2002). *Types and Programming Languages*. The MIT Press, 1 edition.

Pollard, T. D. (2003). The cytoskeleton, cellular motility and the reductionist agenda. *Nature*, 422(6933):741–745.

Priami, C. (1995). Stochastic pi-calculus. *Computer Journal*, 38(7):578–589.

Priami, C., Regev, A., Shapiro, E. Y., and Silverman, W. (2001). Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Inf. Process. Lett.*, 80(1):25–31.

Ptashne, M. and Gann, A. (2001). *Genes and Signals*. Cold Spring Harbor Laboratory Press, 1st edition.

Ralston, A. (2008). Simultaneous gene transcription and translation in bacteria. *Nature Education*, 1(1).

Regev, A. (2003). *Computational Systems Biology: A Calculus for Biomolecular Knowledge*. PhD thesis, Tel Aviv University.

Regev, A., Panina, E. M., Silverman, W., Cardelli, L., and Shapiro, E. Y. (2004). BioAmbients: an abstraction for biological compartments. *Theorical Computer Science*, 325(1):141–167.

Reichl, L. E. (2009). *A Modern Course in Statistical Physics (Physics Textbook)*. Wiley-VCH, 3 edition.

Rossetti, M. D., Hill, R. R., Johansson, B., Dunkin, A., and Ingalls, R. G., editors (2009). *Proceedings of the 2009 Winter Simulation Conference*. Institute of Electrical and Electronics Engineers, Inc.

Saarinen, A., Linne, M.-L., and Yli-Harja, O. (2008). Stochastic differential equation model for cerebellar granule cell excitability. *PLoS Computional Biology*, 4(2):e1000004+.

Schaeffer, O. (2008). *On the Use of Process Algebra Techniques in Computational Modeling of Cancer Initiation and Development*. PhD thesis, University of Birmingham.

Schäfer, A. and John, M. (2009). Conceptional modeling and analysis of spatio-temporal processes in biomolecular systems. In Kirchberg, M. and Link, S., editors, *APCCM*, volume 96 of *CRPIT*, pages 39–48. Australian Computer Society.

Simao, E., Remy, E., Thieffry, D., and Chaouiya, C. (2005). Qualitative modelling of regulated metabolic pathways: application to the tryptophan biosynthesis in e.coli. *Bioinformatics*, 21(suppl_2):ii190–196.

Tait, W. W. (1967). Intensional interpretations of functionals of finite type i. *Journal of Symbolic Logic*, 32(2):198–212.

Takahashi, K., Nanda, S., Arjunan, V., and Tomita, M. (2005). Space in systems biology of signaling pathways : towards intracellular molecular crowding in silico. *FEBS letters*, 579(8):1783–1788.

Uhrmacher, A. M. (2001). Dynamic structures in modeling and simulation: a reflective approach. *ACM Transactions on Modeling and Computer Simulation*, 11(2):206–232.

Uhrmacher, A. M., Ewald, R., John, M., Maus, C., Jeschke, M., and Biermann, S. (2007). Combining micro and macro-modeling in devs for computational biology. In Henderson, S. G., Biller, B., Hsieh, M.-H., Shortle, J., Tew, J. D., and Barton, R. R., editors, *Winter Simulation Conference*, pages 871–880. Institute of Electrical and Electronics Engineers, Inc.

Versari, C. (2007a). A core calculus for a comparative analysis of bio-inspired calculi. In Nicola, R. D., editor, *ESOP*, volume 4421 of *Lecture Notes in Computer Science*, pages 411–425. Springer.

Versari, C. (2007b). Encoding catalytic p systems in pi@. *Electronic Notes in Theoretical Computer Science*, 171(2):171–186.

Versari, C. (2009). *A Core Calculus for the Analysis and Implementation of Biologically Inspired Languages*. PhD thesis, University of Bologna.

Versari, C. and Busi, N. (2009). Stochastic biological modelling in the presence of multiple compartments. *Theoretical Computer Science*, 410(33-34):3039–3064.

Versari, C., Busi, N., and Gorrieri, R. (2009). An expressiveness study of priority in process calculi. *Mathematical Structures in Computer Science*, 19(6):1161–1189.

Wawra, C., Kühl, M., and Kestler, H. A. (2007). Extended analyses of the Wnt/β-catenin pathway: Robustness and oscillatory behaviour. *FEBS Letters*, 581(21):4043 – 4048.

Wolkenhauer, O., Ullah, M., Kolch, W., and Cho, K.-H. H. (2004). Modeling and simulation of intracellular dynamics: choosing an appropriate framework. *IEEE Transactions on Nanobioscience*, 3(3):200–207.

Zeigler, B. P., Praehofer, H., and Kim, T. G. (2000). *Theory of Modeling and Simulation, Second Edition*. Academic Press, 2 edition.