A Generic Language for Biological Systems based on Bigraphs

Troels C. Damgaard
Jean Krivine

IT University Technical Report Series **TR-2008-115**

ISSN 1600–6100 **December 2008**

ISSN 1600–6100

ISBN 9788779491892

Copies may be obtained by contacting:

IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
Denmark

Telephone: **+45 72 18 50 00**
Telefax: **+45 72 18 50 01**
Web `www.itu.dk`

A Generic Language for Biological Systems based on Bigraphs

Troels C. Damgaard and Jean Krivine

Abstract

Several efforts have shown that process calculi developed for reasoning about concurrent and mobile systems may be employed for modelling biological systems at the molecular level. In this paper, we initiate investigation of the meta-language framework *bigraphical reactive systems*, due to Milner et al., as a basis for developing rule-based languages for molecular biology.

We describe a family of $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi sharing a small set of familiar operators and operations, and provide them with a simple operational semantics. We show that $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi and their reaction semantics correspond to a version of bigraphical reaction under *non-aliasing* contexts and with reaction rules extended to allow negative side-conditions for the subset of bigraphs corresponding to $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes.

Finally, to illustrate the usage of $\mathcal{B}^{\Sigma,\mathcal{R}}$, we show that with non-aliasing semantics the κ-calculus may be faithfully captured as a $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculus.

1 Introduction

Starting with Regev, Shapiro, and Silverman [RSS01], several efforts have shown that process calculi developed for reasoning about concurrent and mobile systems may be succesfully employed for modelling biological systems at the molecular level. In the κ-calculus [DL04], Danos and Laneve suggested a paradigm, which we may call *rule-based modelling* for capturing protein-protein interaction at the level of protein domains. On top of a flat graph-based static model, a user of the κ-calculus writes her own set of reaction rules modelling in isolation each possible local protein-protein interaction. More recently, the κ-calculus has also been provided with a stochastic semantics and an efficient implementation allowing simulation and various methods of causality analysis [DFF$^+$07, DFFK07].

One way of viewing how we model in the κ-calculus is, that we are allowed to instantiate a domain-specific sub-calculus specialized for the study of a particular problem; the obvious virtue being that we may engineer the reactive system to reflect very directly our setting. This benefit also influences the primitives of the κ-language. Comprising essentially rewrite rules over nodes and named edges, supported by a simple algebraic notation, the κ-calculus is relatively light on language-idiosyncracies. This allows domain-specialists (i.e., molecular biol-

ogists) to more easily perform the abstraction from chemical binding between proteins to edges between nodes.

As it stands, the κ-calculus focuses solely on protein-protein interaction. Nature, however, consists of more than variants of chemical binding among proteins. There are a multitude of different kinds of objects, properties of objects, forces, and environments, which play vital roles at the molecular level. Examples include objects such as variants of biological membranes, viruses, non-protein organic matter, and properties such as the three-dimensional folding structure of proteins. We may also consider how to model fluids (and properties such as their viscosity) or energy (be that, e.g., in the form of radiation or in the form of the energy-currency of cells—ATP/ADP-molecules); or environmental conditions such as pressure, temperature, electricity, salinity, etc. If we wish to model nature in more detail, we need to begin by searching for good abstractions of some of these phenomena. Our overall goal is to develop κ-like languages to encompass also some of these phenomena. In this paper, we shall focus on taking some preparatory steps towards that goal; we shall investigate the so-called *bigraphical framework* as a basis for developing families of calculi for modelling biological systems at the molecular level.

Bigraphs and bigraphical reactive systems (BRSs) have been developed by Milner and coworkers [JM04, Mil06]. Though capable of representing a wide variety of domains, they have been aimed particularly at providing a graphical meta-calculus capable of being instantiated to capture the structure and dynamics of various nominal process calculi concerned with concurrency, mobility, and locality. Loosely, bigraphs provide us with *nodes* for modelling terms, and *links* for modelling names. Nodes are arranged in a *place graph*, a forest, providing a model for nesting and prefixing for terms. A *link graph* allows nodes to connect to links that may be named or unnamed, in turn providing a model of terms using free or bound (fresh) names. A key design parameter for bigraphs has been that graph isomorphism should reflect structural congruence in the modelled calculus. This connection has also been well-researched [Mil05, DB06]. As such, we may think of a bigraph as a model of a structural congruence class of terms. We instantiate a bigraphical calculus by giving a *signature* (for nodes) and a set of reaction rules.

As noted already by Regev et al. [RPS$^+$04], the bigraphical model has a striking resemblance to the (informal) graphical models used for bio-calculi such as BioAmbients. In the meantime, bigraphs have succesfully been used for modelling several calculi—many resembling those that have been developed for studying cellular biology. Variants of bigraphs have been succesfully employed to recapture the semantics of a wide range of process calculi (such as CCS [Mil06], variants of the π-calculus [JM04, BS06], and Homer [BH06]). Several extensions of bigraphs have been investigated (concerned, e.g., with scoping [JM04, DB06], or fusion [GM07]), and bigraphs have also been applied for modelling directly different systems (such as context-aware systems [BDE$^+$06]). Implementation of bigraphs has also been investigated [BDGM07], and a prototype implementation is available [BPL07]. Lately, bigraphs have also been provided with a stochastic semantics [KMT08]. As is evident from several pro-

posals [PRSS01, PQ05, DFFK07], stochastics is important for biology as it allows for more accurate quantitative biological modelling.

In all, the bigraphical framework seem well-poised as a foundation for experimenting with models and languages for biological systems. In particular, it seems that we may employ the bigraphical meta-modelling framework to capture directly the rule-based modelling paradigm as pioneered by the κ-calculus. Further, we expect to be able to employ the notion of nesting for adding to κ-like languages biological compartments á la BioAmbients [RPS$^+$04], Brane calculi [Car04], or beta-binders [PQ05].

In this paper, we adress the following issues, to pave the way for further studies of calculi for biology based on bigraphs.

Bigraphical idiosyncracies Traditionally bigraphs are presented as a categorically based graphical model with a closely corresponding term language with a small set of categorically derived core operators and a wide variety of derivable operators. In turn, the semantics for the reactive systems for bigraphs builds upon this understanding of bigraphical components and rules. This is important and useful for developing the bigraphical meta-calculus, but is less convenient for appreciating a concrete calculus.

Bigraphical rules are non-contextual The bigraphical framework demands strict non-contextuality of rules. Bigraphical rules contains no mechanisms for expressing arbitrary contextual negative side-conditions—for instance, to require some ancestor to be of a certain type or control (although given the versatility of *wide* rules, i.e., rules with more than one region, and with the help of the *binding* variants of bigraphs [JM04, DB06], one may encode certain contextual checks). This has lead to much skillfullness in encoding checks of such contextual condition, typically using small sets of rules performing an iteration over the necessary context (see, e.g., [BDE$^+$06]); or using bigraphical rule-schemas, which range over a denumerable set of bigraphical rules (see, e.g., [BH06]). Such encoding is at best impractical, and at worst may hinder our capture of essential atomicity or transactional properties.

In perspective, we may compare bigraphs to graphical meta-modelling frameworks in the long tradition of graph-transformation systems (GTSs) [EPS73, Roz97, REKE99, REKM99], or to term rewriting systems (TRSs) [TeR03]. For both GTSs and TRSs, variants of negative side-conditions have been studied.

We should note that in modelling biological systems, focusing on local causes for reactions is also of virtue. However, certain reaction patterns may be more conveniently expressed using a modicum of contextual conditions. Specifically, since we model chemical bonds with (named) edges of a graph, we expect connectivity-constraints to become central. The forces governing chemical bonds are inherently severely limited by range; there is no such direct correspondent for named edges.

As an aside, one may note that the noncontextual nature of bigraphical reaction rules is partly due to bigraphical research being rooted in investigations

of automatic derivation of congruential contextual equivalences for process calculi. For languages and models for biology, contextual equivalences have not yet proven very useful. The complexity of nature is such that currently our struggle lies in finding languages with the right abstractions for representing selected key components, events and their causes in biological systems. However, even if we find ways to abstract faithfully certain parts and mechanisms of nature, it is by no means clear that contextual reasoning would be able tell us anything interesting about nature. In any model of nature we focus only on those parts and mechanisms we are able to capture in our language; thus any contextual equivalence is by design heavily dependent on the level of abstraction of our language, in particular also on the amount of factors that we do *not* model. This is, of course, a general point, not particularly pertaining to models of molecular biology; but in a setting, where, for instance, a signalling pathway may be shut off due to a minute change in complex enviromental conditions—such as a minor change in the local acidity conditions or in the flow of the electrical current—the point becomes acutely emphasized.[1]

Contributions of this paper In this paper, we aim at laying the foundation for using bigraphical calculi to experiment with models and languages for biological systems. To address the issues highlighted above, we

- discuss the usage of BRSs for modelling biological interaction and treat bigraphical reaction under *non-aliasing* contexts and extend reaction rules to include testing of negative side-conditions;
- introduce a family of $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi sharing a small set of classical process calculus operators and operations, and provide them with a self-contained operational semantics;
- show formally that $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi and their reaction semantics correspond to bigraphical reaction under non-aliasing contexts for the subset of bigraphs corresponding to $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes; and,
- show that with non-aliasing semantics the (nondeterministic) κ-calculus may be faithfully captured as a $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculus.

In Section 2, we discuss the bigraphical foundation as needed by our domain of interest—our main aim being to strip away some of the generality of the bigraphical model, before we in the second step turn to presentation. We also discuss how to extend bigraphical rules to allow negative side-conditions.

Previous usage of the bigraphical machinery for capturing certain domain-specific models have used variants of lightly sugared syntax for expressing bigraphs
(e.g., [BDE$^+$06]). In Section 3, we make the effort to treat carefully a family

[1] Debois has given a more detailed account and discussion of bigraphical modelling and bisimulation, see [Deb06].

of languages more in line with standard process calculi—$\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi—and develop also a *self-contained* presentation of the dynamic semantics void of most bigraphical idiosyncracies. We stress that the effort to produce a self-contained presentation was a goal in itself. We have expended some effort in settling on a model, which may yield to a short and comprehensible operational semantics, in a structural style, and resembling that given for the κ-calculus.

In Section 4, we formally state the relationship between $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi and BRSs; and we conclude this paper by illustrating in Section 5, that we may recapture the (nondeterministic) κ-calculus, as presented in [DL04], neatly as a $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculus.

Readers' guide This paper deals in part with developing a self-contained presentation of a certain family of bigraphical calculi. Hence, we aim the section concerned with defining $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi (Section 3), and the section on encoding the κ-calculus as a $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculus (Section 5), at readers who know little or next to nothing about bigraphs. In Section 5, we relate the κ-calculus to the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework. We briefly recap the central concepts in the κ-calculus, however, to fully appreciate the discussion, a certain preknowledge on the κ-calculus is probably needed, as can be gotten from, say, [DL04]. The section that deals solely with bigraphical machinery (Section 2) and the section concerned with establishing that $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi are, in fact, a certain kind of bigraphical calculi (Section 4), we adress mainly to readers who are familiar with the basic setup of pure bigraphs (as can be gotten from, say, [Mil06]).

2 Bigraphical Preliminaries

In this section, we start by giving brief and informal recap of necessary concepts of the theory for pure bigraphs [Mil06]. We then continue to discuss and motivate the addition of contextual additions, and finish by supplying a new definition of (linear) reaction rules with side-conditions.

2.1 Pure Bigraphs—a Brief Recap

A *bigraph* consists of a *place graph*; a forest, whose nodes represent a variety of computational objects, and a *link graph*, which is a hyper graph connecting ports of the nodes. Certain leafs of the place graph may contain ordered *sites* (or holes). The expression $\Box_0$ denotes a bigraph with a single site. The link graph may also contain *inner* or *outer* names, and the link graph links inner names and ports to outer names or (unnamed) edges. The *outer face* of a bigraph is a pair $\langle n, X\rangle$ which registers the number of regions n and outer names X; the *inner face* is a pair $\langle m, Y\rangle$, which registers the number of sites m and inner names Y. We may compose the bigraph B with A, if the outer face of B matches the inner face of A. The bigraph AB is computed by plugging the sites of A with the roots of B, and fusing the links to outer names of B with the links from their inner name counterpart in A. With the help of composition we may

model name-hiding; by composing a bigraph A with the bigraph $/x$ we may remove the outer name x, and replace it with an edge.

We may also combine bigraphs with a tensor product, $\otimes$, which is simply juxtapositioning of roots, requiring that both inner and outer names be disjoint. From composition and product, we may derive further combinators, such as $||$ that juxtaposes roots and links up equal names, and $|$ that merges two single-root (prime) bigraphs as well as linking up equal names.

A bigraphical *signature*, Σ, determines a set of controls and provides for each control K a finite ordinal, the number of ports, and one of three types, atomic, passive, or, active. Each node of a bigraphs is assigned such a control. Atomic nodes are restricted to be leafs, while active and passive nodes may nest other nodes inside. The expression $K_{\vec{x}}$ denotes the bigraph consisting of the single node K with each port i linked severally to a name x_i.

Bigraphs can be reconfigured by means of *reaction rules*. A rule is essentially a pair of bigraphs (R, R'). Rules are *parametric*—R and R' may contain holes, which are filled with parameters. Parametric rules generate an infinite set of ground rules, which we may then subsequently contextualize. Essentially, we ground and contextualize (R, R') by closing rules under composition from above and below. We may rewrite an agent $a \rightarrow a'$ with (R, R'), when we have $a = C(R \otimes \mathsf{id})d \rightarrow C(R' \otimes \mathsf{id})d' = a'$ for some context C and ground parameter d. Graphically, we may think of this as matching an instance of the pattern R inside a and substituting it with R'. In general, bigraphical rules may both discard and copy parameters; a so-called *instantiation* maps holes in R' to R. The bigraph d' is computed from the parameter d by means of this instantiation.

In essence, a *bigraphical reactive system* consists of a set of bigraphs over a given signature, and a set of reaction rules, which can be used to reconfigure the set of bigraphs.

Linearity We aim to model mainly physical objects and phenomena, and in nature, rarely it happens that matter is copied without matter being expended. Therefore, we shall restrict ourselves to considering only linear rules, that is, when instantiations are bijections. We seek calculi, which serve as vehicles for investigating how nature implements low-level structures and their interaction; not abstract away from it.

Active nesting When contextualizing bigraphical rules, the context C is required to be *active*, determined by requiring that the redex R be nested only inside nodes with active control; in our setting, however, we shall only treat nodes with active control. Passive controls are useful for modelling blocking prefixes and may also be used for modelling passive storage compartments of active code, for instance, modelling envelopes for mobile code. We envisage no important usage of such passive compartments in modelling biological matter. Thus, we require contextual conditions that prevent reaction in certain compartments be modelled explicitly.

2.2 Adding Contextual Conditions

In the domains where the bigraphical framework have been used up until now, links have been applied for modelling entity-relations mostly orthogonal to the locality-structure.

On the contrary, the structures whose essential properties we seek to capture with links—like protein backbones, domain-domain binding, or the intermediate state of two fusing membranes—are highly constrained by distance and locality.[2] For those reasons it is convenient to be able to control *sharing* and *freshness* of names in parameters.

Let us sketch a concrete example: Suppose that we want to express that a node that models protein (matched in the redex) may be diffused from one compartment to another only if the entire complex of proteins of which the protein is part (i.e., its entire connected component) can be transported along with it. For expressing this rule, it is highly inconvenient to require us to match the entire complex in the rule. There is a huge number of possible configurations of complexes (species) in which a particular protein may be a part. Hence, we wish to match *only* the protein in the redex, and match the remainder of the complex in a parameter.

However, in nature such a diffusion reaction may be prevented because of certain local conditions otherwhere in the complex, for instance, if part of the complex is tied to a membrane. In our model, that tied part of the complex may be arbitrarily distant. To test such an inherently contextual condition with vanilla bigraphical rules, we need to write rules for stepping through the complex and perform this test for every part of the complex. This is impractical and fails to capture the atomicity of diffusion—which in turn is problematic (but not unfixable) should we wish to add a stochastic semantics. Essentially, we would need the set of rules implementing diffusion in this manner to uphold a transactional guarentee.

Instead, we aim to extend bigraphical reaction rules to internalize such contextual (negative) side-conditions along with rules; we shall be mostly concerned with capturing connectedness-constraints among the parameters in reactions, such as in the example sketched above. In doing this, it shall be convenient for us to depart slightly from standard tradition in how we derive ground, contextualized rules, as we shall explain in further detail below. Traditionally, bigraphical calculi have been provided with a semantics, which corresponds to pushing name-fusings (i.e., bigraphical substitions) and all binders (i.e., bigraphical closures) to the top. But it need not be so.

2.2.1 Non-aliasing Reaction

We define *non-aliasing reaction* for BRSs, which allows us to extend rules with side-conditions which test connectedness among parameters. We shall change

[2] We remark that such distance-related locality constraints do not match especially well the constraints imposed by bigraphical *binding*, which seeks to capture traditional lexical scoping of bound names.

grounding and contextualization of rules to essentially push name-fusings and binders as far *down* as possible. To explain our choices, we recall in a bit more detail, how grounding and contextualization works.

Recall that in grounding parametric rules it is stipulated that the parameter d be *discrete*, that is, with no bound or shared names. This produces bigraphs where every link is a unique name. In particular, this restriction resolves a possible ambiguity in deriving ground, contextualized rules when the instantation is non-linear. For example, when applying a rule that copies a parameter such as the rule $(\mathsf{K} \otimes \mathsf{id})\Box_0 \to (\mathsf{K} \otimes \mathsf{id})(\Box_0 \,|\, \Box_0)$, to the agent $a = /x\,(\mathsf{K} \otimes \mathsf{id})(\mathsf{M}_x \,|\, \mathsf{M}_x)$ do we copy the binder or not? This corresponds precisely to instantiating the parameter as either $d = /x\,(\mathsf{M}_x | \mathsf{M}_x)$ under the empty context or as $d = (\mathsf{M}_{x_1} | \mathsf{M}_{x_2})$ under the context $C = /x\, x/x_1, x_2$. Both choices constitute valid decompositions of a, that is, for both choices of the parameter d and the context C, do we have $a = C(R \otimes \mathsf{id})d$. But only the second choice, that is, $d = (\mathsf{M}_{x_1} \mid \mathsf{M}_{x_2})$, is discrete. Hence, in the pure variant of bigraphs copying parameters results in name-sharing.

As we are only concerned with linear rules, this ambiguity does not arise in our setting. The choice to take only discrete parameters means, however, that we cannot test in the (grounded or parametric) rule whether two links in a parameter d will be matched as parts of the same link, as any two names in the grounded rule $(R \otimes \mathsf{id})d$ may be fused in the context. For our purposes this is impractical, as we want to express side-conditions, which require certain sets of names to be disjoint. We could express these conditions as requirements to be fullfilled by the context C instead; it turns out, however, out that by instead enforcing the first choice above—matching binders and shared names in the parameter—we may express the conditions directly and succinctly on the grounded rule $(R \otimes \mathsf{id})d$.

In conclusion, we shall choose to allow any bigraph as the parameter d, but restrict contexts C to be link-mono, or *non-aliasing* on names.

Now we define (linear) reaction rules with side-conditions on parameters. We allow arbitrary conditions, but (as we shall exemplify in Section 3) we are mostly interested in side-conditions testing the outer names of parameters.

Definition 2.1 (linear reaction rules with side-conditions)**.** A (concrete) parametric linear reaction rule is a rule on the form $(R : m \to J, R' : m \to J, \varphi)$, where R is the *redex*, R' the *reactum* and φ is a predicate on ground bigraphs. R and R' are required to be lean, i.e., they have no idle edges.

For every ground bigraph $g : \langle m, X \rangle$, where $\varphi(g)$ holds, the parametric rule generates every ground reaction rule of the form (r, r'), where $r \eqsim (\mathsf{id}_X \otimes R)g$ and $r' \eqsim (\mathsf{id}_X \otimes R')g$.

The *non-aliasing* bigraphical reactive system over rules with side-conditions is built as usual, but for the requirement that the context C also be link-mono. We may express when a reaction may occur as below.

Definition 2.2 (non-aliasing reaction)**.** $G \to G'$ with (R, R', φ), if $G = C(\mathsf{id} \otimes R)g$ and $G' = C(\mathsf{id} \otimes R')g$, for active, link-mono C and g such that $\varphi(g)$ holds.

We underline the implication of taking a non-aliasing semantics: That no outer names in the grounded rule (be it from R or d) can be aliased in the context, effectively means that every such name is a unique handle on every link connected to the grounded rule. This is convenient for our purposes, but may, of course, not be so for other applications.

Interestingly, we find that allowing parameters to share and close names, we provide a view on matching bigraphical rules, which corresponds more easily to viewing bigraphical languages as a generic framework for rewriting on terms enriched with names. For instance, it directly allows us to plug the hole in the pattern $(\mathsf{K}_x \otimes \mathsf{id})\Box_0$ with a term using x, e.g., $(J_x \otimes \mathsf{id})0$, to form $(\mathsf{K}_x \otimes \mathsf{id})\mathsf{J}_x.0$ instead of requiring us to think of matching it as $x/x_1, x_2((\mathsf{K}_{x_1} \otimes \mathsf{id})(\mathsf{J}_{x_2} \otimes \mathsf{id})0)$.

We shall expand on this remark in Section 3 to build a self-contained characterization in structural operational semantics-style of how bigraphical calculi evolve under non-aliasing semantics.

3 A Generic Process Calculus

$\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi are a generic family of process calculi equipped with pure names, a new-name operator (sometimes called also hiding), parallel product and nesting, used for modelling both prefixing and nesting (i.e., á la mobile ambients). We instantiate a $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculus by giving a *signature*, Σ, and a set of reaction rules, $\mathcal{R}$. The signature allows us to tell which function symbols we may build processes from. Processes of $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi are quotiented according to a shared structural congruence relation, ensuring that we interpret scope of new names and parallel product as usual. The reaction rules allow us to give the dynamic semantics. Reactions are contextualized using a standardized scheme, which we show in detail later.

We start by formally defining a signature.

Definition 3.1 (signature)**.** A signature, Σ, is a set of *controls*, $\mathcal{K}$; an *arity* map, $\mathrm{ar} : \mathcal{K} \to \mathbb{N}$; and a map determining for every control, $\mathsf{K} \in \mathcal{K}$, whether it is *active* or *atomic*.

The basic computing units in $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi are processes. Processes are built from function symbols with control from Σ, parallel product and new name operators. The arity of a control tells us how many *ports* for names a function symbol has. Furthermore, if a control is active, it may be a prefix. We write prefixing with the help of an infix dot operator "."—in correspondence with the prefixing operator of process calculi in the π-family. For example, suppose that Σ contains L : atomic(2) and M : active(0)—short for telling that Σ contains the controls L and M, that L is atomic and M active, and that $\mathrm{ar}(\mathsf{L}) = 2$ and that $\mathrm{ar}(\mathsf{M}) = 0$. Then $\mathsf{L}_{y,x} \,|\, \mathsf{M}.\mathsf{L}_{y,z}$ is a valid process. For many applications we may intend prefixing to model containment, and then it may be more sensible to choose an ambient-style notation for active controls. We may then define, for instance, $[P] \stackrel{\text{def}}{=} \mathsf{M}.P$; where the M is a representation of the ambient.

x, y, z	pure names
X, Y, Z	variables
K	generic control
M	active control
L	atomic control

Figure 1: Notational conventions—names, variables and controls

In formally defining processes, we presuppose an unbounded supply of pure *names*, $\mathcal{N}$. We use lowercase letters, $x, y, z, \ldots$ for pure names and sanserif letters, $\mathsf{K}, \mathsf{L}, \mathsf{M}, \ldots$, for controls. For further notational convenience, in the following we shall treat M as having active control, and consider L (for leaf) an atomic control.

For writing reaction rules, we shall also need process terms with process-valued *variables*; and process *groups*—ordered sequences of proceses, which may share names. Variables in $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi are numbered, i.e., they are drawn from a countable set of variables $\mathbb{V} = \{V_0, V_1, V_2, \ldots\}$. This allows us to conveniently define substitution of a process group into a process with variables (as we shall see below) variable in an term. We require for well-formed process terms, that all variables in a term are distinct. We use uppercase letters $X, Y, Z, \ldots$ as metavariables ranging over variables.

These notational conventions are summarized in Figure 1. In Definition 3.2, we define formally processes, and in Definition 3.3, we define process groups. A process is a special case of a group—namely, a group with exactly one process. We use this in many of the following definitions and properties and overload the symbols we use for operations such as structural congruence and substitution to work for both processes and groups. Definition 3.4 formally defines the well-formedness criterion.

Definition 3.2 (processes)**.** $\mathcal{B}^{\Sigma,\mathcal{R}}$ processes over the signature Σ are defined inductively as

M, N	$::=$	$\mathsf{M}_{\vec{y}}.M$	active prefix
	$\mid$	$\mathsf{L}_{\vec{x}}$	atom
	$\mid$	$(x)\,M$	new name
	$\mid$	$M \mid N$	parallel product
	$\mid$	0	the empty process
	$\mid$	X	variable

when $|\vec{y}| = \mathrm{ar}(\mathsf{M})$ and $|\vec{x}| = \mathrm{ar}(\mathsf{L})$.

Definition 3.3 (groups)**.** $\mathcal{B}^{\Sigma,\mathcal{R}}$ (process) groups over the signature Σ are defined inductively as

C, D	$::=$	M	single process
	$\mid$	$(x)\,C$	new name
	$\mid$	$C \mid\mid D$	wide parallel product
	$\mid$	ϵ	the empty group

Definition 3.4 (well-formed processes and groups)**.** Processes M and groups G are well-formed iff all variables in M and G are distinct.

In the following, we shall assume that all processes or groups that we treat are well-formed, although, of course, we need to make sure that well-formedness is preserved by the operations we define on processes. We write $\mathrm{var}(M)$ and $\mathrm{var}(C)$ for the (possibly empty) set of variables in processes M and groups C.

Call processes or groups without variables *ground*. Call nonground processes *preprocesses*, and nonground groups *pregroups*—in general, we call nonground processes or groups, *contexts*. We reserve the metavariables P, Q for ground processes, and G, F for ground groups. Finally, we shall write $\mathcal{P}^{\Sigma}$ for ground processes over the signature Σ.

We use parentheses for grouping as usual. To save ink for parenthesis in large terms, we let prefixing . bind tighter than $|$, which binds tighter than $||$, which in turn binds tighter than the operator (x). When $\tilde{x} = \{x_1, \cdots, x_n\}$ we write $(x_1 \cdots x_n)\, M$ or $(\tilde{x})\, M$ to mean $(x_1) \cdots (x_n)\, M$. Finally, as usual, we shall typically elide the trailing 0 under empty prefixes, for instance, writing M instead of $\mathsf{M}.0$.

3.0.2 Ordering of Variables

We are not interested in the particular numbers used for variables in a given term, only in their relative ordering inside that term. For instance, we do not wish to distinguish, the two processes $V_{45} \,|\, V_{56}$ and $V_0 \,|\, V_1$. All variables in a well-formed term are distinct, so we may order the variables in any term uniquely according to their numbers. It is therefore clear what we mean, when we refer to the ith variable in a given term.

We shall now define order-preserving renumbering, and by including order-preserving renumbering in the structural congruence relation, we formalize notion that we consider processes up to such order-preserving renumbering. We write $\mathrm{nv}(C)$ for the set of numbers of variables in C.

Definition 3.5 (order-preserving renumbering)**.** Given a group C and an order-preserving and injective map $r : \mathrm{nv}(C) \to \mathbb{N}$, let $[C]^r$ be the group C with all variables renumbered according to r.

It is convenient to define also the *stratifying* renumbering, the renumbering that maps every variable to the number given by its relative order. To that end, we may order the renumberings themselves, pointwise. Formally, we simply say that for all renumberings r and s, $r \leq s$ iff for all x, $r \downarrow x \iff s \downarrow x$ and $r(x) \leq s(x)$. It is easy to check that this is induces a partial ordering with a least element: the order-preserving renumbering of variables that maps the ith variable in a term to V_i.

Definition 3.6 (stratifying renumbering)**.** The stratifying renumbering of variables in a group C is the least order-preserving renumbering defined on $\mathrm{nv}(C)$.

We write $[C]$ for the group C renumbered according to this renumbering, and call $[C]$ stratified.

The stratifying renumbering shall serve as a help in defining substitution, by mapping, for instance, $V_{45} \mid V_{56}$ to $V_0 \mid V_1$.

3.0.3 Free and Bound Names

The new name operator $(x)\,M$ is a binder—as we shall see below instances of the name x in M are alpha-convertible. We define inductively the set of free or bound names of a term as usual.

Definition 3.7 (free and bound names)**.** For processes M the free names $\mathrm{fn}(M)$ and the bound names $\mathrm{bn}(M)$ are defined inductively as:

$$\begin{array}{rclcrcl}
\mathrm{fn}(\mathsf{M}_{\vec{y}}.M) & = & \vec{y} \cup \mathrm{fn}(M) & \qquad & \mathrm{bn}(\mathsf{M}_{\vec{y}}.M) & = & \mathrm{bn}(M) \\
\mathrm{fn}(\mathsf{L}_{\vec{y}}) & = & \vec{y} & & \mathrm{bn}(\mathsf{L}_{\vec{y}}) & = & \emptyset \\
\mathrm{fn}(M \mid N) & = & \mathrm{fn}(M) \cup \mathrm{fn}(N) & & \mathrm{bn}(M \mid N) & = & \mathrm{bn}(M) \cup \mathrm{bn}(N) \\
\mathrm{fn}((x)\,M) & = & \mathrm{fn}(M) \setminus x & & \mathrm{bn}((x)\,M) & = & x \cup \mathrm{bn}(M) \\
\mathrm{fn}(0) & = & \emptyset & & \mathrm{bn}(0) & = & \emptyset \\
\mathrm{fn}(A) & = & \emptyset & & \mathrm{bn}(A) & = & \emptyset
\end{array}$$

For groups we extend the definition above to include also:

$$\begin{array}{rclcrcl}
\mathrm{fn}(C \mid\mid D) & = & \mathrm{fn}(C) \cup \mathrm{fn}(D) & \qquad & \mathrm{bn}(C \mid\mid D) & = & \mathrm{bn}(C) \cup \mathrm{bn}(D) \\
\mathrm{fn}((x)\,C) & = & \mathrm{fn}(C) \setminus x & & \mathrm{bn}((x)\,C) & = & x \cup \mathrm{bn}(C) \\
\mathrm{fn}(\epsilon) & = & \emptyset & & \mathrm{bn}(\epsilon) & = & \emptyset
\end{array}$$

We call a process or group *closed* if all its names are bound; and *open* if any names are free. We say that two processes M and N are *connected* if they share free names, that is, if $\mathrm{fn}(M) \cap \mathrm{fn}(N) \neq \emptyset$.

3.0.4 Structural Congruence

We quotient processes and groups according to a structural congruence relation enforcing prominently that names in the scope of a binder are alpha-convertible and that (x) floats freely in a term, as long as we do not indaverdently capture free instances of the name x. Furthermore, we make parallel product associative, allow reordering, and stipulate that we may introduce (or delete) empty processes. For groups, we make wide parallel product associative and make ϵ the neutral element.

Definition 3.8 (structural congruence)**.** Structural congruence, $\equiv$, on processes and groups is the least congruence relation containing α-equivalence (i.e., bijective renaming of bound names), order-preserving renumbering of variables, and s.t.:

- parallel product, $\mid$, is associative and commutative with 0 as neutral element;
- wide parallel product, $\mid\mid$, is associative with ϵ as neutral element;

and including the following *scope extrusion* laws

$$\begin{array}{rcll} M \mid (x)\, N & \equiv & (x)\, M \mid N & \text{if } x \notin \mathrm{fn}(M) \quad \text{(extrusion - par)} \\ ((x)\, M) \mid\mid N & \equiv & (x)\, M \mid\mid N & \text{if } x \notin \mathrm{fn}(N) \quad \text{(extrusion - wide par left)} \\ M \mid\mid (x)\, N & \equiv & (x)\, M \mid\mid N & \text{if } x \notin \mathrm{fn}(M) \quad \text{(extrusion - wide par right)} \\ \mathsf{M}_{\vec{y}}.((x)\, M) & \equiv & (x)\, \mathsf{M}_{\vec{y}}.M & \text{if } x \notin \vec{y} \quad \text{(extrusion - prefix)} \\ (x)(y)M & \equiv & (y)(x)M & \quad \text{(reordering)} \\ (x)M & \equiv & M & \text{if } x \notin \mathrm{fn}(M) \quad \text{(elision)} \end{array}$$

As usual it is easy to check that free names are invariant under structural congruence, that is, for processes $M \equiv N$, $\mathrm{fn}(M) = \mathrm{fn}(N)$.

3.0.5 Normal Form

Using structural congruence laws we may push binders to the top (performing α-conversion as needed), remove superfluous binders via elision, and remove empty processes or groups to bring every process and group on a normal form, resembling the standard form for CCS [Mil80].

Proposition 3.9 (normal forms)**.** *Every process M is structurally congruent to a normal form*

$$M \equiv (\tilde{x})(M_0 \mid \cdots \mid M_{n-1})$$

where each $M_0, \cdots, M_{n-1}$ is a variable, an atom, or a prefix (i.e., on the form $\mathsf{K}_{\vec{y}}.N$) containing no binders; and where $\tilde{x} \subseteq \mathrm{fn}(M_0) \cup \cdots \cup \mathrm{fn}(M_{n-1})$. (If $n = 0$, then $M_0 \mid \cdots \mid M_{n-1} \stackrel{\text{def}}{=} 0$, and if $\tilde{x} = \emptyset$ then the binder $(\tilde{x})$ is not there.)

Every group C is structurally congruent to a normal form

$$C \equiv (\tilde{x})(M_0 \mid\mid \cdots \mid\mid M_{n-1})$$

where each $M_0, \cdots, M_{n-1}$ is non-empty, contain no binders, and is otherwise on (process) normal form; and where $\tilde{x} \subseteq \mathrm{fn}(M_0) \cup \cdots \cup \mathrm{fn}(M_{n-1})$. (If $n = 0$, then $M_0 \mid\mid \cdots \mid\mid M_{n-1} \stackrel{\text{def}}{=} \epsilon$, and if $\tilde{x} = \emptyset$ then the binder $(\tilde{x})$ is not there.)

The forms are unique up to α-equivalence, and reordering of binders and parallel processes (i.e., up to the commutative law for $\mid$).

We shall write $C \equiv_{\mathbf{N}} C'$, if $C \equiv C'$ and C' is on normal form.

Having formalized a normal form for groups, we may conveniently define the width of a group as the number of non-empty top-level processes.

Definition 3.10 (width of group)**.** For

$$C \equiv_{\mathbf{N}} (\tilde{x})(M_0 \mid\mid \cdots \mid\mid M_{n-1})$$

let $\mathrm{width}(C) = n$.

3.0.6 Substitution

Nonground processes (and groups) have variables for which we may substitute other processes. For $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi we shall apply substitution mainly to define the substitution of parameters for the variables in parametric reaction rules, that is, rules with variables.

We start by defining basic substitution of the variables in a group C by a process M. (For ease we define substitution only for groups, C. The definition for processes follows as a special case.) Substitution is capture-avoiding, as usual.

Definition 3.11 (raw substitution)**.** Let φ be a bijective map from variables to processes $M_0, M_1, \ldots, M_{n-1}$.

The substitution $C\varphi$ of variables in C by the processes in φ is defined when $|\operatorname{var}(C)| = n$, for all $i \in n$, $\operatorname{bn}(C) \cap \operatorname{fn}(M_i) = \emptyset$. In that case, we define $C\varphi$ inductively over the structure of C,

$$
\begin{aligned}
(\mathsf{M}_{\vec{y}}.M)\,\varphi &= \mathsf{M}_{\vec{y}}.(M\varphi) \\
\mathsf{L}_{\vec{x}}\,\varphi &= \mathsf{L}_{\vec{x}} \\
((x)\,M)\,\varphi &= (x)\,(M\varphi) \\
(M \mid N)\,\varphi &= (M\varphi \mid N\varphi) \\
0\,\varphi &= 0 \\
X\varphi &= \begin{cases} M_i & \text{if } \varphi(X) = M_i \\ X & \text{else} \end{cases} \\
\\
((x)\,C)\,\varphi &= (x)\,(C\varphi) \\
(C \mid\mid D)\,\varphi &= (C\varphi \mid\mid D\varphi) \\
\epsilon\,\varphi &= \epsilon.
\end{aligned}
$$

As each variable is a leaf in C it is easy to see that the substituted term respects the grammar (i.e., in Definition 3.2 and in Definition 3.3). However, in general, raw substitution does *not* preserve well-formedness. Any process M_i may contain variables that C or some other process M_j also contains. Hence, we shall only use raw substitution as a means to define two versions of substitution, where this issue is resolved.

We start by defining total substitution, $C \cdot D$, the substitution of all variables in a group C with the processes in a group D.[3] We shall define total substitution by pushing the binders of D to the top of the created term; hence, we also require that both bound and free names of C be distinct from bound names in D. This is (another) technical requirement, as we can also α-convert bound names of D to avoid any clashes.

Definition 3.12 (total substitution)**.** Substitution $C \cdot D$ of variables in C by processes in D is defined when $|\operatorname{var}(C)| = \operatorname{width}(D)$ and $\operatorname{bn}(C) \cap \operatorname{fn}(D) =$

[3] We choose an infix notation for total (and partial) substitution in analogy with the categorical notation for composition; an analogy, which we shall make formal in the next section, when we relate $\mathcal{B}^{\Sigma,\mathcal{R}}$-substitution to bigraphical composition.

$\mathrm{bn}(C) \cap \mathrm{bn}(D) = \mathrm{fn}(C) \cap \mathrm{bn}(D) = \emptyset$. In this case, for

$$D \equiv_{\mathbf{N}} (\tilde{x})(M_0 \,||\, \cdots \,||\, M_{n-1})$$

let

$$C \cdot D = (\tilde{x})[C]\{V_0 \mapsto M_0, \cdots, V_{n-1} \mapsto M_{n-1}\},$$

Note, that we use a stratifying renumbering to renumber the variables in C before substituting, to ensure that the variables in C are numbered severally from 0 to $n-1$. Observe also that, as all variables in C have been substituted, the term $C \cdot D$ is well-formed iff D is.

It is easy to check that substitution is associative.

Proposition 3.13 (total substitution is associative)**.** $(C \cdot D) \cdot E = C \cdot (D \cdot E)$.

It shall be convenient to generalize our definition of substitution to also include partial substitution—where only some of the variables of the context C are substituted. Partial substitution is not in general associative; by convention we take it to be left-associative, and distinguish it with a non-symmetric symbol.[4] Definition 3.14 generalizes Definition 3.12 to the cases, where D has fewer processes than C has variables.

Definition 3.14 (partial substitution)**.** Partial substitution $C \triangleleft D$ of variables in C by processes in D is defined when $|\,\mathrm{var}(C)| \geq \mathrm{width}(D)$ and (as for total substitution) when $\mathrm{bn}(C) \cap \mathrm{fn}(D) = \mathrm{bn}(C) \cap \mathrm{bn}(D) = \mathrm{bn}(C) \cap \mathrm{fn}(D) = \emptyset$. In this case, let

$$C \triangleleft D = C \cdot ([D] \,||\, V_k \,||\, \cdots \,||\, V_{k+n}),$$

for $|\,\mathrm{var}(C)| - \mathrm{width}(D) = n$ and $|\,\mathrm{var}(D)| = k$.

It is immediate from the definitions, that in the case where $|\,\mathrm{var}(C)| = \mathrm{width}(D)$, $C \cdot D \equiv C \triangleleft D$. In the case, where C has more variables, than D has processes, we simply extend D with appropriately numbered variables. More generally, we may also consider total and partial substitution as a way to *compose* processes or groups with other processes or groups.[5] As we shall see, this is convenient for expressing succinctly filling parameters in rules with variables; viewing substitution as a way to compose terms will be useful also for seeing that a set of terms are all substitution-instances of a certain kind.

3.1 Operational Semantics

To instantiate a $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculus one gives a signature, Σ and a set of reaction rules $\mathcal{R}$. We start by defining reaction rules.

[4] It is easy to check that partial substition may be undefined if evaluated right to left (thus the non-associativity), but is always defined if evaluated left to right.

[5] In bigraphs, we have substitution residing syntactically in the language—composition is an *term constructor* instead of an *operation* as we define it as here. This is one of the simplifications that we make for $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi, whose repercussions, to the term language, the structural congruence relation (in particular) and the structural operational semantics, help bring the presentation more in line with standard tradition for process calculi.

Definition 3.15 (reaction rule)**.** A $\mathcal{B}^{\Sigma,\mathcal{R}}$-rule (over the signature Σ) $(M \to N, \varphi)$ is a pair of processes, where $\mathrm{fn}(M) = \mathrm{fn}(N)$and $\mathrm{var}(M) = \mathrm{var}(N)$; along with φ, a predicate on ground groups. We call M the left-hand side (or lhs) of the rule, and N the right-hand side (or rhs). We call $\mathrm{fn}(M) = \mathrm{fn}(N)$ the free names of the rule and $\mathrm{var}(M) = \mathrm{var}(N)$ the variables of the rule. We call φ the side-condition of the rule.

Loosely, any ground process P that is a substitution-instance of the left-hand side of a rule may be rewritten with that rule.[6] Given a rule $(M \to N, \varphi)$, if for some (ground) group G, we have $P \equiv M \cdot G$, we may perform a reaction $P \to P' \equiv N \cdot G$, if $\varphi(G)$ is satisfied. We say that P *matches* M with the *parameter* G, since G consists of the group of processes that we will substitute for the variables in N. We allow an arbitrary side-condition, φ, but we shall be concerned mainly with predicates testing free names of the parameters of a reaction.

In rules, variables take on their intended role of placeholders—serving only to carry parameters across a reaction. For rules, our only concern shall be, where variables of the left-hand side are reused on the right-hand side; neither the numbers of variables or ordering internal to the left-hand side or right-hand side matter. More formally, it is easy to verify, that our definition for reaction is closed under order-preserving renumbering (applied to both sides of a rule). Thus, by convention we shall use metavariables $A, B, C, \ldots$ in rules to denote variables (as in the example below), thus eliding which particular (numbered) variables are chosen.

Reactions may occur in any process context. We contextualize reactions according to standard tradition. We close reactions under syntactic constructions, structural congruence, and also under (bijective) renaming of free names. Definition 3.16 records a small set of rules which together characterize reactions for processes.[7]

Definition 3.16 (reactive system)**.** Given a signature Σ and a set of reaction rules $\mathcal{R}$, $\mathcal{T}^{\Sigma,\mathcal{R}}$ the reactive system associated with $\mathcal{R}$ for $\mathcal{P}^{\Sigma}$ is given by the reaction relation $\to$, the least binary relation over $\mathcal{P}^{\Sigma}$, s.t.

$$\textsc{rule}\ \frac{(M \to N, \varphi) \in \mathcal{R} \qquad \exists G \text{ s.t. } P = M \cdot G \text{ and } P' = N \cdot G \qquad \varphi(G) \text{ satisfied}}{P \to P'}$$

$$\textsc{par}\ \frac{P \to P'}{P \mid Q \to P' \mid Q} \qquad \textsc{prefix}\ \frac{P \to P'}{\mathsf{M}_{\vec{y}}.P \to \mathsf{M}_{\vec{y}}.P'}$$

$$\textsc{close}\ \frac{P \to P'}{(x)P \to (x)P'} \qquad \textsc{struct}\ \frac{Q \equiv P \qquad P \to P' \qquad P' \equiv Q'}{Q \to Q'}$$

$$\textsc{subst}\ \frac{P \to P' \qquad \exists \tilde{x}.\alpha : \mathrm{fn}(P) \leftrightarrow \tilde{x}}{\alpha(P) \to \alpha(P')},$$

[6] We could have defined reaction for groups or, for that sake, for pre-processes or -groups, but for our purposes, ground rewriting on processes shall be enough.

[7] Tradition differs on whether to call unlabelled transitions reactions (as in BRSs) or transitions (as in κ). To remove any confusion: our reactions are unlabelled and correspond to transitions in κ.

where α is a bijection between the free names of P and fresh names $\tilde{x}$, and $\alpha(P)$ is the process P with free names substituted by names $\tilde{x}$.

It follows easily from the definition that free names of P are preserved, that is, for $P \to P'$, $\mathrm{fn}(P) = \mathrm{fn}(P')$.

An important property of $\mathcal{B}^{\Sigma,\mathcal{R}}$ is that the semantics is *non-aliasing*. Let us explicate what we mean by this. First of all, we allow bijective renaming of free names, since the names of reaction rules are only intended as placeholders, which may be matched to any name in a process. However, by allowing only bijective renaming, we ensure that *disconnectedness* (immediate) is preserved by contextualization. This in turn ensures us that we may meaningfully give side-conditions, which test the free names of processes.

Consider an example. Take the rule

$$R = (A \mid B \mid \mathsf{M} \to A \mid \mathsf{M}.B, \varphi = \{\mathrm{fn}(A) \cap \mathrm{fn}(B) = \emptyset\}).$$

We intend this rule to mean: "When two processes A and B reside beside each other and an M-container, one of these processes may relocate to the M-container, only if no entities in A and B are connected with each other."[8] For instance, we intend to reject reactions such as $\mathsf{K}_z \mid \mathsf{K}_z \mid \mathsf{M} \to \mathsf{K}_z \mid \mathsf{M}.\mathsf{K}_z$, where the parameter $G = \mathsf{K}_z \mid\mid \mathsf{K}_z$ of the reaction do indeed share names (we have $(A \mid B \mid \mathsf{M}) \cdot (\mathsf{K}_z \mid\mid \mathsf{K}_z) = \mathsf{K}_z \mid \mathsf{K}_z \mid \mathsf{M}$).

Suppose that we had allowed arbitrary substitution, σ, in SUBST instead of only bijective renaming. With this version of the SUBST-rule (call it SUBST_σ), we may build the following derivation

$$\mathrm{RULE}\,\frac{\mathrm{SUBST}_\sigma\,\dfrac{R \in \mathcal{R} \qquad G = \mathsf{K}_x \mid\mid \mathsf{K}_y \qquad \varphi \text{ satisfied}}{\mathsf{K}_x \mid \mathsf{K}_y \mid \mathsf{M} \to \mathsf{K}_x \mid \mathsf{M}.\mathsf{K}_y \qquad \sigma = \{y \mapsto z, x \mapsto z\}}}{\mathsf{K}_z \mid \mathsf{K}_z \mid \mathsf{M} \to \mathsf{K}_z \mid \mathsf{M}.\mathsf{K}_z}$$

which contradicts our intention with the side-condition.

In Section 4, we shall see precisely, how we may consider $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes and groups as corresponding to certain graphs with typed nodes corresponding to function symbols, nesting corresponding to prefixing, and, (named) links corresponding to usage and sharing of names.

4 Bigraphs and $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi

In this section, we verify formally that we may consider $\mathcal{B}^{\Sigma,\mathcal{R}}$ as a sugared and restricted language for expressing certain kinds of bigraphs, and that $\mathcal{B}^{\Sigma,\mathcal{R}}$-reaction correspond to a variant of bigraphical reaction—extended with negative side-conditions and under *non-aliasing* contexts. The verification itself is fairly straightforward—our effort has consisted mainly in choosing suitable restrictions.

[8] In expressing side-conditions for reaction rules concisely, it is convenient to overload the usage of variable-names in the rule, such as A and B, to refer to the processes in the parameter that we are substituting for those variables.

Such restrictions as we adopt in the $\mathcal{B}^{\Sigma,\mathcal{R}}$-language for building processes are key for allowing us to express reactions and contextualization as we have done it in Definition 3.16. As opposed to bigraphs, we do not have explicit name-substitution in the language; instead we take a rule for contextualization that allows renaming of free names. We do not have explicit substitution or composition as a combinator in the language, we take only the restricted prefix combinator and define substitution as an *operation*, instead; and, we take as a primitive the parallel and wide parallel operator, instead of the (bigraphical) tensor product, which requires disjointness of names. These choices simplify both structural congruence, and the operational semantics.[9]

In the following, we look to establish a dynamic correspondence between $\mathcal{B}^{\Sigma,\mathcal{R}}$-reaction and bigraphical reaction. We establish first, in Proposition 4.2, a static correspondence verifying that $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes correspond to certain kinds of bigraphs with a single root (primes). In Lemma 4.3, we state formally that $\mathcal{B}^{\Sigma,\mathcal{R}}$-substitution is engineered to correspond to a relaxed version of bigraphical composition. This shall help us establish the dynamic correspondence; in Lemma 4.4, we characterize $\mathcal{B}^{\Sigma,\mathcal{R}}$-reaction via substitution. By way of this characterization, in Theorem 4.5, we may formally state and verify the dynamic correspondence, we are looking for.

4.1 Statics

We start by verifying that $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes are in one-to-one correspondence with certain bigraphs with one root. Observe first, that for signatures the relation is trivial—$\mathcal{B}^{\Sigma,\mathcal{R}}$-signatures are simply bigraph-signatures with no passive controls.

Recall that *link-epi* are those link graphs with no idle outer names, and that *prime* bigraphs are those with a single root and no inner names. Also recall, that from the bigraphical $\otimes$-product that requires total disjointness of both outer and inner names, we may derive a name-sharing *parallel* product $||$, and *prime* product $|$. Finally, in the following, to distinguish bigraphs from $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes, we write bigraphs with a superscripted $^{\mathbf{B}}$.

We start by stating a normal form for link-epi, prime bigraphs, which we use to make an direct comparison with the normal forms for $\mathcal{B}^{\Sigma,\mathcal{R}}$.

Lemma 4.1 (normal form for link-epi, prime bigraphs)**.** *All link-epi primes $P^{\mathbf{B}}$ may be expressed on the following normal form*

$$\begin{array}{rcl} M^{\mathbf{B}} & = & (\mathsf{K}_{\vec{x}} \mid \mathsf{id}_Y)P^{\mathbf{B}} \\ P^{\mathbf{B}} & = & (/Z \mid \mathsf{id}_1)(\mathsf{id}_n \mid M_0 \mid \cdots \mid M_{k-1})\pi \end{array}$$

In case K is atomic, then the $M^{\mathbf{B}}$ form degenerates to $M^{\mathbf{B}} = \mathsf{K}_{\vec{x}}$. (Also note that, contrary to discrete normal form *(cf. [Mil05]), the names $\vec{x}$ need not be distinct.)*

[9] It should, of course, be said that the categorically derived tensor product and composition prove their worth in establishing many meta-theorems about bigraphs, and, in general, seem to enjoy better algebraic properties [Mil05, DB06]. However, our emphasis in this paper is on presentation; hence our effort to show that bigraphs and BRSs may be presented otherwise.

Proof. Follows easily from the completeness of the DNF and CNF normal forms for bigraphs (see [Mil05]). □

Comparing the normal form for $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes in Proposition 3.9 and the normal form for link-epi primes above, it is easy to check the following property.

Proposition 4.2 ($\mathcal{B}^{\Sigma,\mathcal{R}}$-processes are link-epi primes)**.** *$\mathcal{B}^{\Sigma,\mathcal{R}}$-processes over the signature Σ considered up to $\equiv$ correspond one-to-one to link-epi prime bigraphs (over the corresponding bigraphical signature Σ).*

Proof. As mentioned above, $\mathcal{B}^{\Sigma,\mathcal{R}}$-signatures are simply bigraph-signatures with no passive controls. We need only compare normal forms. They are essentially equal, up to the extra care with identities we need to take for bigraphs, and up to the fact that ordering of variables in the bigraphical term language is captured via a permutation π. □

From Lemma 4.2 and the normal form for groups, it follows also, that $\mathcal{B}^{\Sigma,\mathcal{R}}$-groups are products of link-epi primes. Hence, we may treat $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes or groups as denoting bigraphs. We extend the usage of superscripting bigraphs with a $^{\mathbf{B}}$ to allow applying it as an operator. Given a process M or a group C, in the following we let $M^{\mathbf{B}}$ or $C^{\mathbf{B}}$ denote the corresponding bigraph. Substitution for $\mathcal{B}^{\Sigma,\mathcal{R}}$ has been defined to behave like bigraphical composition allowing extension with a link-identity, as usual. (This is also sometimes known as bigraphical "dotting" [Mil09].)

Lemma 4.3 ($\mathcal{B}^{\Sigma,\mathcal{R}}$-substitution is bigraphical composition)**.** $(C \cdot D)^{\mathbf{B}} = (C^{\mathbf{B}} \,||\, \mathsf{id}_Y)D^{\mathbf{B}}$*, for* $Y = \mathrm{fn}(D^{\mathbf{B}})$.

4.2 Dynamics

We now turn to dynamics. We relate $\mathcal{B}^{\Sigma,\mathcal{R}}$-reaction rules and bigraphical rules (with prime, link-epi redices and reactums) pointwise. Side-conditions for $\mathcal{B}^{\Sigma,\mathcal{R}}$-rules may be translated directly to bigraphical side-conditions (as defined in Section 2.2). We extend the $^{\mathbf{B}}$-notation and write $\varphi^{\mathbf{B}}$ for the bigraphical predicate corresponding to φ.

To pave the way for relating $\mathcal{B}^{\Sigma,\mathcal{R}}$ and bigraphical (non-aliasing) semantics, we start by characterizing in terms of substitution the $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes that Definition 3.16 allows to rewrite.

Lemma 4.4 (characterizing $\mathcal{B}^{\Sigma,\mathcal{R}}$-reaction via substitution)**.** *For any $\mathcal{B}^{\Sigma,\mathcal{R}}$-reaction $P \to P'$ via a rule with left-hand side M and right-hand side N, we have $P \equiv (\tilde{x})(O \cdot \alpha(M \cdot G))$ and $P' \equiv (\tilde{x})(O \cdot \alpha(N \cdot G))$, for some process with one variable O with $\mathrm{fn}(O) = \tilde{z}$, a (ground) group G, and $\alpha : \tilde{x} \cup \tilde{z} \leftrightarrow \tilde{y}$ for some $\tilde{y}$.*

Proof. We may show that for any reaction $P \to P'$, there exists a *normal* derivation, $\mathcal{D}_{\mathcal{N}}$, of a reaction among processes Q and Q' structurally congruent

to P and P' (i.e., $Q \equiv P$ and $Q' \equiv P'$) given by the little grammar below:

$$\mathcal{D}_{\mathcal{N}} ::= \textsc{close}^* \frac{\mathcal{D}_{\textsc{ctxt}}}{\cdots} \qquad \mathcal{D}_{\textsc{ctxt}} ::= \begin{cases} \textsc{rule}\ \frac{}{\cdots} \\ \textsc{subst}\ \frac{}{\cdots} \\ \textsc{struct}\ \frac{\mathcal{D}_{\textsc{ctxt}}}{\cdots} \\ \textsc{par}\ \frac{}{\cdots} \\ \textsc{prefix}\ \frac{\mathcal{D}_{\textsc{ctxt}}}{\cdots} \end{cases}$$

Reading the grammar bottom-up, we stipulate that—modulo structural congruence—any reaction may be derived by first applying RULE, then SUBST to rename free names of the rule; followed by a sequence of PREFIX and PAR to add arbitrary context in the form of prefixes or arbitrary processes in parallel (using STRUCT to shuffle the parallel components of the term, if needed); finally closing names meant to be hidden (using CLOSE* as a shorthand for zero or more applications of CLOSE).

We may prove the completeness of the normal derivation by induction on the structure of the derivation of the reaction $P \to P'$. The only somewhat tedious case is when we consider a derivation concluding with a renaming α using SUBST. The logic behind the positioning of SUBST in the normal derivation grammar is, that the only relevant usage of that rule is to rename free names used in the rule itself. Names used only in context introduced by PAR and PREFIX, we may already choose freely. To conclude this formally, note that by the induction hypothesis we have a normal derivation $\mathcal{D}'_{\mathcal{N}}$ of a reaction without the renaming α. To verify the case, we construct a new normal derivation by essentially applying α across $\mathcal{D}'_{\mathcal{N}}$ up to the RULE/SUBST leaf, and then merge α with the substitution in the SUBST already occuring in $\mathcal{D}'_{\mathcal{N}}$. The remaining cases may be verified straightforwardly.

It remains to remark, that, using contiguous applications of PREFIX and STRUCT/PAR and the final sequence of CLOSE, we may build as context any process $(\tilde{x})\, O$ with one variable; thus reasoning that the substitution of $\alpha(M \cdot G)$ into O characterizes the contextualization of the core reaction. With the help of the normal form for processes (Proposition 3.9) this is straightforward. □

Having thus characterized reactions via substitution, which in turn corresponds to bigraphical composition (Lemma 4.3) we may bridge the gap to bigraphical reaction.

Theorem 4.5 ($\mathcal{B}^{\Sigma,\mathcal{R}}$ reaction is bigraphical reaction under non-aliasing contexts)**.** *$P \to Q$ by $(M \to N, \varphi)$ iff $P^{\mathbf{B}} \to Q^{\mathbf{B}}$ by $(M^{\mathbf{B}} \to N^{\mathbf{B}}, \varphi^{\mathbf{B}})$ as defined in 2.1 and Definition 2.2.*

Proof. ($\Rightarrow$) We are given a reaction $P \to Q$ by $(M \to N, \varphi)$, and we need to construct a link-mono bigraph C and a ground parameter d, s.t. $P^{\mathbf{B}} = C.M^{\mathbf{B}}.g$ and $Q^{\mathbf{B}} = C.N^{\mathbf{B}}.g$.

From Lemma 4.4, we know that $P \equiv (\tilde{x})\, (O \cdot \alpha(M \cdot G))$, and $Q \equiv (\tilde{x})\, (O \cdot \alpha(N \cdot G))$ for some process with one variable O and a (ground) group G. W.l.o.g.,

assume that $\tilde{x} \subseteq \mathrm{fn}(O \cdot \alpha(M \cdot G))$—we may remove excess names from $\tilde{x}$ via elision.

We have

$$((\tilde{x})(O \cdot \alpha(N \cdot G)))^{\mathbf{B}} = (/\tilde{x} \otimes \mathsf{id}_{\langle 1, Y \uplus Z \rangle})O^{\mathbf{B}}.(\alpha \otimes \mathsf{id}_1)(N^{\mathbf{B}}.G^{\mathbf{B}}),$$

where $\alpha : \mathrm{fn}(N \cdot G) \to Y$ and $Z = \mathrm{fn}(O)$. We know from Lemma 4.3 that $\mathcal{B}^{\Sigma,\mathcal{R}}$-substitution is bigraphical dotting; name-hiding and substitution correspond to composition with closure and renaming, respectively.

We choose $C = (/\tilde{x} \otimes \mathsf{id}_{\langle 1, Y \uplus Z \rangle})(O^{\mathbf{B}} || \mathsf{id}_Y)(\alpha \otimes \mathsf{id}_1) = (/\tilde{x} \otimes \mathsf{id}_{\langle 1, Y \uplus Z \rangle})(O^{\mathbf{B}} || \alpha)$, and $g = G^{\mathbf{B}}$. We note that C is link-mono by construction, since $O^{\mathbf{B}}$ (and any other correspondent to a $\mathcal{B}^{\Sigma,\mathcal{R}}$-process) has no inner names and α—the part of C that will create inner names—is, by definition, link-mono (in fact, iso). By construction we have $P^{\mathbf{B}} = C.M^{\mathbf{B}}.g$ and $Q^{\mathbf{B}} = C.N^{\mathbf{B}}.g$, and we are done.

($\Leftarrow$) We are given a reaction among link-epi primes (and thus correspondents of $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes) $P^{\mathbf{B}} \to Q^{\mathbf{B}}$ by a reaction rule $(M^{\mathbf{B}} \to N^{\mathbf{B}}, \varphi^{\mathbf{B}})$; in other words, we know that for some link-mono C and parameter g, $P^{\mathbf{B}} = C.M^{\mathbf{B}}.d$ and $Q^{\mathbf{B}} = C.N^{\mathbf{B}}.g$; and C must also be link-epi as $P^{\mathbf{B}}$ is. We need to construct a $\mathcal{B}^{\Sigma,\mathcal{R}}$-derivation of a reaction, s.t., $P \to P'$ by $(M \to N, \varphi)$.

To verify the case, we may reverse most of the reasoning for $\Rightarrow$-direction. We note that in the construction above, the bigraphical correspondent to $(\tilde{x})\, O$, $(/\tilde{x} \otimes \mathsf{id})O^{\mathbf{B}}$, ranges over all link-epi primes.In turn, it is easy to verify (with the help of the normal forms for bigraphs [Mil05]) that the C derived from O ranges over all link-mono and -epi contexts with both inner and outer width equal to 1 (i.e., of width 1 and with a single hole). Hence, we may reverse the logic of the construction above, to see that $P^B \equiv (\alpha((\tilde{x})\,(O \cdot M \cdot G)))^{\mathbf{B}}$, and $Q^{\mathbf{B}} \equiv (\alpha((\tilde{x})\,(O \cdot N \cdot G)))^{\mathbf{B}}$; and then conclude the case via Lemma 4.4.

(For the side-conditions, we need only remark that for any φ, $\varphi(G)$ iff $\varphi^{\mathbf{B}}(G^{\mathbf{B}})$.) □

5 An Example: The κ-calculus as a $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculus

As an illustration of the $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculus we show how we may capture (modulo a modicum of encoding) the nondeterministic κ-calculus, a language of formal proteins [DL04]. The setup of the κ-calculus has a striking resemblance to the link graph of bigraphs; it is not surprising that we may capture it fairly easily. It is, however, a first step towards our goal of studying extensions of the κ-calculus, as well as serving as an illustration of the $\mathcal{B}^{\Sigma,\mathcal{R}}$-presentation.

In the κ-calculus one instantiates a concrete model by choosing a signature, signifying the proteins one works with, and giving a set of rewrite rules. We describe a family of $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi, $\kappa^{\mathcal{B}}$, which faithfully captures any such κ-model.

5.1 The κ-calculus

Below, we briefly and informally summarize the central concepts in the nondeterministic κ-calculus [DL04].

A κ-calculus description of a system consists of a set of *solutions* over a (κ-calculus) signature, a collection of (formal) proteins, and a set of reaction *rules*. Proteins have a name and a number of ordered *sites*, collectively referred to as the interface of the protein. A site may have an internal state; it can either be *hidden*, *visible*, or, *bound*. Rules provide a description of how agents interact. The elementary interactions consist of binding or unbinding between two sites of proteins, the modification of the state of a site, and the deletion or creation of an agent.

The syntax of the κ-calculus relies on

- a countable set of protein names $\mathbf{P}$, ranged over by uppercase letters $A, B, C, \ldots$;
- a countable set of names $\mathbf{N}$, ranged over by $x, y, z, \ldots$; and,
- $\int$—a (κ-calculus) signature, a map which assigns to each $A \in \mathbf{P}$ an arity signifying the number of sites of the protein A; that is, $\int(A \in \mathbf{P}) = n \in \mathbb{N}$.

For each protein A, $\int(A)$ is the number of sites of A and the pair (A, i) is a site of A.

Interfaces A κ interface is a partial map from $\mathbf{N}$ to $\mathbf{N} \uplus \{\mathrm{h}, \mathrm{v}\}$. We let ρ and σ range over interfaces. A site (A, i) is

- visible, if $\rho(i) = \mathrm{v}$;
- hidden, if $\rho(i) = \mathrm{h}$; and
- bound, if $\rho(i) \in \mathbf{N}$.

A protein A may be assigned an interface ρ, if ρ is defined on a subset of $\int(A)$. For instance, if $\int(A) = 3$, then $\rho = \{1 \mapsto \mathrm{v}, 2 \mapsto \mathrm{h}, 3 \mapsto x\}$ is a well-defined interface for A. It says that site 1 is visible, that site 2 is hidden, and that site 3 is bound to some name x. As interfaces are part of the syntax, it has become tradition to write the interface ρ with syntactic sugar as $\rho = 1 + \overline{2} + 3^x$.

Importantly, interfaces need not give information for all of A's sites; we say that they may be *partial*. This is to allow interfaces to depict partial information on A's sites in reaction rules.

Syntax The syntax for κ-solutions is as follows:

S, T	::=	0	empty solution
	$\mid$	$A(\rho)$	protein
	$\mid$	S, T	group
	$\mid$	$(x)\,(S)$	new name

The $(x)\,S$ operator is a binder of the name x in S, as usual. Definitions for free and bound names may be given inductively, much as in Definition 3.7.

Structural congruence and graph-likeness As $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes, κ-solutions are quotiented by a *structural congruence* relation, $\equiv$, which records that bound names are α-convertible, that $|$ is associative and commutative with 0 as the neutral element; and, that the usual scoping laws for (x) holds; i.e., corresponding to extrusion, reordering, and elision in Definition 3.8.

Graph-like solutions are those solutions, where

- free names occur at most twice, and
- binders bind either zero or two occurences of names.

A graph-like solutions is *strongly* graph-like, if all free names occur exactly twice.

One may define a translation from graph-like solutions to graphs whose nodes have sites—providing a formal graph-based language for the κ-calculus. As expected, structurally congruent solutions translate to the same graph.

At the top of Figure 2, we depict and write terms for two small κ-solutions (ignore for now their translation into $\mathcal{B}^{\Sigma,\mathcal{R}}$-process, at the bottom of the figure). Note that name-sharing between bound sites induces so-called *complexation*-links between sites.

Reaction rules There are two kinds of rules in the κ-calculus, monotonic and anti-monotonic rules. They ensure that rules model biologically well-founded reactions on the chosen level of granularity for the reactions in κ, such that they divide into two clean classes: those that form new complexation links (monotonic), and, those that break complexation links (anti-monotonic); and also that synthesis and degradation (creation and deletion of proteins) is only allowed for proteins with no complexation-links to other proteins. However, as discussed in the paper [DL04], and in later versions of the κ-calculus [DFF$^+$07, DFFK07], the restrictions on monotonicity may also be loosened.

We shall not go into further detail here; suffice to say, that we may also adopt and translate the schema for restriction for monotonic and anti-monotic rules to the $\kappa^{\mathcal{B}}$ setting. The translation and correspondence we prove below is not dependent on monotonicity, however, so in this treatment we shall not be concerned much with monotonicity-restrictions.

For the remainder of this paper, it suffices to say that when $L \to R$ is a monotonic reaction rule, then L and R are graph-like solutions on the following forms:

$$\begin{array}{ll} L = & A_1(\rho_1), \cdots, A_n(\rho_n) \\ R = & (\tilde{x})\, A_1(\sigma_1), \cdots, A_n(\sigma_n), A_{n+1}(\sigma_{n+1}), \cdots, A_m(\sigma_m), \end{array}$$

for (possibly partial) interfaces $\rho_1, \cdots, \rho_n$, and full interfaces $\sigma_1, \cdots, \sigma_m$.

The newly created proteins $A_{n+1}, \cdots, A_m$ must have full interfaces. Formally, this follows from the so-called κ *growth* relation. The growth-relation also ensures, essentially that monotonic rules only add complexation-links and proteins to solutions, while anti-monotonic rules only remove structure.

Anti-monotonic rules are defined by symmetry as reverses of monotonic rules.

Matching The *matching* of a *monotonic* reaction rule to solutions is defined as follows, for S and T, two solutions.

We say that S, T matches $L \to R$ on the form above, written $S, T \vDash L \to R$, iff for some injective renaming r (preserving hidden and unbound sites), and for some partial interfaces $\zeta_1, \cdots, \zeta_n$ we have that:

$$\begin{aligned} S &= A_1(r(\rho_1) + \zeta_1), \cdots, A_n(r(\rho_n) + \zeta_n) \\ T &= (r(\tilde{x}))\, A_1(r(\sigma_1) + \zeta_1), \cdots, A_n(r(\sigma_n) + \zeta_n), A_{n+1}(r(\sigma_{n+1})), \cdots, A_m(r(\sigma_m)), \end{aligned}$$

and, s.t., for all i, $r(\tilde{x}) \cap \mathrm{fn}(\zeta_i) = \emptyset$.

In short, (monotonic) matching allows on L and R injective renaming on names and extension of partial interfaces to full interfaces.

Reaction The reaction relation for the κ-calculus is defined via matching and is closed under syntactic constructions and structural congruence.

The reaction relation is given via a set of rules that resembles Definition 3.16; in fact, the simplicity of contextualization for the κ-reaction relation was a key source of inspiration for the contextualization of $\mathcal{B}^{\Sigma,\mathcal{R}}$-reaction.

We repeat the rules for deriving reactions in the κ-calculus below (eliding only the details of monotonicity):

$$(\mathsf{rule})\frac{S, T \vDash L \to R \in \mathcal{R}}{S \to T} \qquad (\mathsf{new})\frac{S \to T}{(x)\,(S) \to (x)\,(T)}$$

$$(\mathsf{group})\frac{S \to T}{S, U \to T, U} \qquad (\mathsf{struct})\frac{S, T \quad S \equiv S' \quad T \equiv T'}{S' \to T'}$$

5.2 Capturing the κ-calculus as a $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculus

In the following section, we start by defining a $\mathcal{B}^{\Sigma,\mathcal{R}}$ signature for $\kappa^{\mathcal{B}}$ that matches a κ signature. We continue by defining a translation from κ-interfaces, in Definition 5.2, and solutions, in Definition 5.3, to $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes. In Proposition 5.4, we state formally, that this translation respects and reflects structural congruence. Definition 5.5 states that rules are simply translated pointwise, and in Proposition 5.6 we formally state and verify the operational correspondence between reaction in the κ-calculus and reaction in $\kappa^{\mathcal{B}}$. Paving the way for expanding and refining the model of proteins in $\kappa^{\mathcal{B}}$, we finish by giving a little characterization of the images of κ-solutions, in Definition 5.7, and verify it in Lemma 5.8.

We shall model κ-proteins as atomic $\mathcal{B}^{\Sigma,\mathcal{R}}$-function symbols and κ-calculus names directly as $\mathcal{B}^{\Sigma,\mathcal{R}}$-names. However, while the ports of $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi serve only to link names, the sites of proteins hold bits of *state* information. We have to capture also, that κ rules allow us to express only partial interfaces, where some sites of a protein may be left out.

We capture these features of κ-interfaces by "exploding" the protein-model of the κ-calculus—making the protein-node model (only) the backbone of a protein, its ports connecting it to sites modelled as separate nodes.

Consequentially, we inject a translation of κ-interfaces into our $\mathcal{B}^{\Sigma,\mathcal{R}}$-signature and add a few extra controls. We add three kinds of site nodes: $\ominus_p$, a site in unbound state connected to a protein-backbone via p; $\odot_p$, a site in hidden state connected to a protein-backbone via p; and, $\oplus_{p,c}$, a site in bound state connected to a protein-backbone via p and to another site via c. To sum up, the first port of a site link sites to their (backbone) protein, while the second port, if present, are for complexation links.[10]

Definition 5.1 ($\mathcal{B}^{\Sigma,\mathcal{R}}$ signature for $\kappa^{\mathcal{B}}$)**.** We consider the signature

$$\begin{aligned}\Sigma \quad = \quad & \{A : \text{atomic}(n) \mid A \in \mathbf{P}, \int(A) = n\} \uplus \\ & \ominus : \text{atomic}(1) \uplus \odot : \text{atomic}(1) \uplus \oplus : \text{atomic}(2).\end{aligned}$$

5.2.1 Translating κ-solutions

We have already sketched above how we intend to translate κ-solutions to $\kappa^{\mathcal{B}}$-processes. In our model of κ, we push κ-interfaces to the level of first-class citizens. So, in defining translation formally, we define first translation of κ-interfaces to $\kappa^{\mathcal{B}}$-processes. We shall use links to connect also a protein with its (translated) interface, so we parameterize the translation of interfaces by a another partial map, a *backbone* map, from $\mathbb{N}$ to (backbone) names $\mathbf{N}$, which is used to create these links (and is introduced in the translation of solutions, below).

Definition 5.2 (translation of κ-interfaces)**.** Given a backbone map $\beta : \mathbb{N} \to \mathbf{N}$ and a κ-interface ρ of the form $\{(i \mapsto x), (j \mapsto \text{h}), (k \mapsto \text{v}), \ldots\}$[11], we define $[\![\rho]\!]_\beta$, the translation of ρ under β, pointwise, translating

- $(i \mapsto x)$ as $\oplus_{\beta(i),x}$,
- $(j \mapsto \text{h})$ as $\odot_{\beta(j)}$,
- $(k \mapsto \text{v})$ as $\ominus_{\beta(k)}$;

and composing translated parts with parallel bar, $|$.

We continue to give a fully compositional translation from κ-solutions to $\kappa^{\mathcal{B}}$-processes. The only slightly nontrivial part is the translation of proteins. We make a set of fresh names for the backbones and wire up a translated interface using these names. As we see, translation is homomorphic in the new- and parallel-operators (and its unit 0); underlining that the level of encoding is very light.

[10] We have some degree of freedom here, of course. We might have taken just two rather than three kinds of sites, $\odot : \text{atomic}(1)$ and $\oplus : \text{atomic}(2)$ and let $(c)\,\oplus_{p,c}$ model an unbound domain. We choose, however, to stay as close in spirit to the original presentation of the κ-calculus as possible, i.e., to have no link at all, when the site is free.

[11] Strictly speaking, we must also require that β and ρ be defined on an equal subset of $\mathbb{N}$.

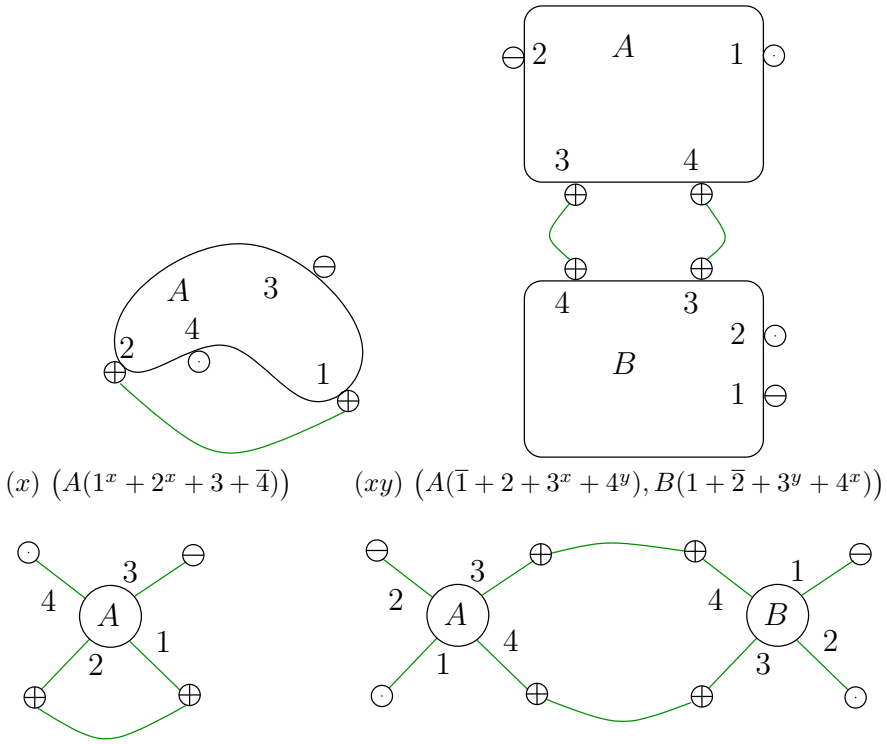

Figure 2: Examples: κ-terms as $\kappa^{\mathcal{B}}$-processes.

Definition 5.3 (translation of κ-solutions)**.** We define translation of κ-solutions inductively over the structure of κ-solutions.

$$\begin{array}{rcl} [\![0]\!] & = & 0 \\ [\![A(\rho)]\!] & = & (\tilde{b'})A_{b_1,\ldots,b_n} \mid [\![\rho]\!]_\beta \\ [\![S, S]\!] & = & [\![S]\!] \mid [\![S]\!] \\ [\![(x)\,(S)]\!] & = & (x)\,(S), \end{array}$$

for A with arity n, fresh names $b_1, \ldots, b_n$, and $\beta : n \to \tilde{b'} = \{i \mapsto b_i | \rho(i) \text{ defined}\}$.

Why is $\tilde{b'}$ not just $b_1, \ldots, b_n$? Because, we anticipate that we also need to translate proteins in rules with partial interfaces—i.e., where ρ is only defined on a subset of n. We therefore close only those backbone-links which have a site-counterpart in the (translated) interface. (This also explains why we require names $b_1, \ldots, b_n$ to be fresh; in translating rules we might meet other proteins with partial interfaces.)[12]

Figure 2 shows two examples of κ proteins and depictions of their translation into $\kappa^{\mathcal{B}}$-processes. As the encoding is almost homomorphic, and each of the

[12]We assume that κ-solutions are valid, i.e., that interfaces are allowable for proteins. We may adopt conditions stating such restrictions directly, but elide them here for brevity.

laws for structural congruence in the κ-calculus has a direct correspondent in the $\mathcal{B}^{\Sigma,\mathcal{R}}$-setting, it is simple to verify, that structural congruence among $\kappa^{\mathcal{B}}$-processes captures structural congruence in the κ-calculus.

Proposition 5.4 (static correspondence)**.** *$S \equiv T$ (for κ-$\equiv$) if and only if $[\![S]\!] \equiv [\![T]\!]$ (for $\mathcal{B}^{\Sigma,\mathcal{R}}$-$\equiv$).*

5.2.2 Translating Rules

We have already paved the way for translating κ-solutions with partial interfaces, so we may simply translate κ-rules pointwise. The translation and treatment of κ-rules is not dependent on whether the rule is monotonic or anti-monotonic, so (as noted above) we shall disregard monotonicity in this translation.

Definition 5.5 (translation of rules)**.** Given any kind of κ-rule, $L \to R$, the corresponding $\kappa^{\mathcal{B}}$ rule is $[\![L \to R]\!] = ([\![L]\!] \to [\![R]\!], \emptyset)$.

5.2.3 Operational Correspondence between $\kappa^{\mathcal{B}}$-reaction and κ-reaction

Finally, we turn to verifying that $\kappa^{\mathcal{B}}$-reaction recaptures κ-reaction.

Our encoding is light, and the setup of the $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi is inspired in part by the formal semantics of the κ-calculus; hence, the verification of the operational correspondence is not too hard. We sketch the proofs in some detail below.

Proposition 5.6 (operational correspondence)**.**

1. *For all κ-solutions S and T, if $S \to T$ by $L \to R$ then $[\![S]\!] \to [\![T]\!]$ by $[\![L \to R]\!]$.*
2. *If $[\![S]\!] = P \to Q$ by $r = [\![L \to R]\!]$, then there exists a solution T, s.t., $[\![T]\!] = Q$, and $S \to T$ by $L \to R$.*

Proof. In both cases, we shall assume that $L \to R$ is a monotonic rule (the anti-monotonic case is similar); then L and R are on the form

$$\begin{array}{ll} L = & A_1(\rho_1), \cdots, A_n(\rho_n) \\ R = & (\tilde{x})\, A_1(\sigma_1), \cdots, A_n(\sigma_n), A_{n+1}(\sigma_{n+1}), \cdots, A_m(\sigma_m), \end{array}$$

1.: We are given a κ-derivation of $S \to T$ stemming from a match on the rule $L \to R$. The κ-match is on the form

$$(\mathsf{rule})\frac{S', T' \vDash L \to R \in \mathcal{R}}{S' \to T'}$$

for

$$\begin{array}{ll} S' & = A_1(r(\rho_1) + \zeta_1), \cdots, A_n(r(\rho_n) + \zeta_n) \\ T' & = (r(\tilde{x}))\, A_1(r(\sigma_1) + \zeta_1), \cdots, A_n(r(\sigma_n) + \zeta_n), A_{n+1}(r(\sigma_{n+1})), \cdots, A_m(r(\sigma_m)), \end{array}$$

where r is an injective renaming (preserving hidden and unbound sites), s.t., for all i, $r(\tilde{x}) \cap \mathrm{fn}(\zeta_i) = \emptyset$.

We build a $\mathcal{B}^{\Sigma,\mathcal{R}}$-derivation corresponding to this match:

$$\textsc{par}\,\frac{\textsc{struct}\,\dfrac{\textsc{subst}\,\dfrac{\textsc{rule}\,\dfrac{(\llbracket L\rrbracket, \llbracket R\rrbracket, \emptyset) \in \mathcal{R} \qquad P' \equiv \llbracket L\rrbracket \qquad Q' \equiv \llbracket R\rrbracket}{P' \to Q'} \qquad \alpha = r \downarrow \mathrm{fn}(P')}{P'' \equiv \alpha(P') \to \alpha(Q') \equiv Q''}}{P'' \to Q''}}{P'' \mid \llbracket\zeta_1\rrbracket \mid \cdots \mid \mid \llbracket\zeta_n\rrbracket \to Q'' \mid \llbracket\zeta_1\rrbracket \mid \cdots \mid \mid \llbracket\zeta_n\rrbracket}$$

The $\mathcal{B}^{\Sigma,\mathcal{R}}$-rule step is simple, and both renaming and (due to the first-class encoding of interfaces) interface extension is handled through contextual and structural congruence rules. We first rename free names via SUBST and then (for emphasis) bound names via STRUCT; and then extending interfaces by adding their translations via PAR.

The remainder of the contextual steps in the κ-derivation of the reaction, we mimic directly with their immediate $\mathcal{B}^{\Sigma,\mathcal{R}}$-counterparts, noting in particular that, as we have the static correspondence, we can mimic also any structural congruence-steps.

2.: Via the substitution characterization of matches in $\mathcal{B}^{\Sigma,\mathcal{R}}$ (Lemma 4.4), we have that $P = \llbracket S\rrbracket$ and Q, are on the form

$$\begin{aligned} \llbracket S\rrbracket &\equiv (\tilde{y})\ (\alpha\llbracket L\rrbracket \mid C) \\ Q &\equiv (\tilde{y})\ (\alpha\llbracket R\rrbracket \mid C)\,, \end{aligned}$$

where we use that $\llbracket L\rrbracket$ and $\llbracket R\rrbracket$ are ground, and that all controls in $\kappa^{\mathcal{B}}$ are atomic.

We know also, that since S is a solution it has full interfaces; we can conclude that C must be on the form

$$C \equiv \llbracket\zeta_1\rrbracket \mid \cdots \mid \llbracket\zeta_n\rrbracket \mid C',$$

where C' is the image of an open solution with complete interfaces, and s.t. each ζ_i completes the interface of A_i, i.e., for all $i \in n$, $A_i(\rho_i + \zeta_i)$ and $A_i(\sigma_i + \zeta_i)$ are proteins with full interfaces.

Letting $\alpha^+ = \alpha \cup \mathsf{id}_{\mathrm{fn}(C)}$, we conclude with ease—as our compositional translation is particularly simple—that we have

$$\begin{aligned} \llbracket S\rrbracket &\equiv (\tilde{y})\ (\alpha^+\,(\llbracket A_1(\rho_1), \cdots, A_n(\rho_n), B_1(\gamma_1), \cdots, B_k(\gamma_k)\rrbracket)) \\ Q &\equiv (\tilde{y})\ (\alpha^+\,(\llbracket(\tilde{x})\, A_1(\sigma_1), \cdots, A_n(\sigma_n), A_{n+1}(\sigma_{n+1}), \cdots, A_m(\sigma_m), \\ &\qquad\qquad B_1(\gamma_1), \cdots, B_k(\gamma_k)\rrbracket))\,, \end{aligned}$$

introducing $C' = B_1(\gamma_1), \cdots, B_k(\gamma_k)$. The renaming we may equally apply on κ-names; and, noting that the translation is homomorphic on name-closure, we have found the T, s.t., $Q = \llbracket T\rrbracket$.

It is easy to verify, that $S \to T$ by $L \to R$ by building a κ-derivation. □

5.2.4 Characterizing Images of the Translation

In our $\mathcal{B}^{\Sigma,\mathcal{R}}$ model, we have decoupled proteins and interfaces, or, biologically speaking, protein-backbones and their sites. This decoupling is essentially the only level of encoding in our capture of the κ-calculus. We may formally state two well-formedness conditions on the links that close this coupling. We add a third condition to say that bound sites link only to other bound sites. We express the well-formedness conditions as clauses for each type of port in $\kappa^{\mathcal{B}}$.

The characterization is presented essentially as a sorting [Deb08] on the bigraphs underlying $\mathcal{B}^{\Sigma,\mathcal{R}}$ processes. We shall use the model for proteins and the well-formedness conditions presented here, as the basis for further investigation of a language built on bigraphs for modelling biology [DDK08].

Definition 5.7 (well-formedness conditions)**.**

1. All ports of protein are either (i) linked one-to-one via a bound name to either the first port of a site-nodes (i.e., with control $\ominus$, $\odot$, or $\oplus$), or (ii) a distinct name.

2. All sites are linked one-to-one via a bound name by their first port to the port of a protein.

3. The second port of all $\oplus$-nodes are linked (via an open or bound name) to either another $\oplus$-node, or (for open processes) a name.

We may lift further requirements from the κ-calculus, notably graph-likenessfor κ-solutions directly to $\kappa^{\mathcal{B}}$, copying definitions essentially verbatim.

It is fairly easy to verify that well-formed $\kappa^{\mathcal{B}}$-processes characterize exactly the images of κ-solutions (with partial or complete interfaces).

Lemma 5.8 (well-formed processes correspond to solutions)**.**

1. *for all S, $[\![S]\!]$ is well-formed and unique up to structural congruence, and*

2. *if a $\kappa^{\mathcal{B}}$-process P is well-formed then there exists a unique (up to structural congruence) S (over the corresponding κ-signature), s.t., $P = [\![S]\!]$.*

Proof. We sketch the reasoning.

1.: Easy to verify by induction on the structure of S from the translation in Definition 5.3.

2. By the normal form (Proposition 3.9) for $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes, we know that we may consider P as a sequence of protein-nodes and site nodes, under a set of binders.

Well-formedness condition (2.) tells us that names used on the first port of a site have exactly two occurences—they match up pairwise to a unique name used by a port on a protein; and that name is bound. well-formedness condition (1.) tells us in addition, that any name used on a port of a protein, which does *not* have such a correspondent, is free and has only that one occurence.

In other words, grouping proteins, sites and binders according to that pairing, we see that P consists a topmost binder (allowing the binding of complexation links), and groupings of closed processes of the following form

$$(\tilde{x})\left(A_{\vec{x}} \mid \ominus_{y_1} \mid \cdots \mid \ominus_{y_j} \mid \odot_{u_1} \mid \cdots \mid \odot_{u_k} \mid \oplus_{v_1,w_1} \mid \cdots \mid \oplus_{v_l,w_l}\right),$$

where each y_i, u_i, and, v_i are equal to exactly one name in $\tilde{x}$ (the set corresponding to the distinct names $\vec{x}$).

Such a closed grouping correspond to exactly one κ-protein with a (partial or full) interface.

Well-formedness condition (3.) tells us simply that the names w_i used at the second port of $\oplus$-nodes are distinct from all names used on other types of ports, but may have one (or more, if it is not graph-like) other occurences among names used on the second port of $\oplus$-nodes.

Well-formedness condition (3.) ensures us that links among the second port of $\oplus$-ports correspond one-to-one to κ-complexation links, as do their bound or free state. □

5.2.5 Concluding Remarks

In all, comparing the semantics of κ with that of $\kappa^{\mathcal{B}}$, we may remark that to allow for partial interfaces, we needed to lift sites to the level of first-class citizens and disjoin them from protein backbones.

In return for this bit of added complexity, we get a more uniform treatment of interface extension and renaming, and a simpler rule for matching. In $\mathcal{B}^{\Sigma,\mathcal{R}}$, the matching rule, RULE, is comparatively simpler (in particular, for the atomic controls of $\kappa^{\mathcal{B}}$) than its κ-counterpart. This is because interface extension and renaming is incorporated as part of the κ-matching rule; in $\kappa^{\mathcal{B}}$, this is handled as just another part of contextualization.

We may also note, that non-aliasing $\mathcal{B}^{\Sigma,\mathcal{R}}$-reaction is actually necessary for recapturing directly the injectivity of the renaming of names as produced by the κ-matching rule. As noted in the introduction, when modelling biology, controlling and testing connectedness is valuable.

6 Conclusion

In this paper, we have treated and motivated the extension of bigraphical reaction rules to include testing of negative side-conditions, and, for defining these sensibly, defined reaction under *non-aliasing* contexts.

We have introduced the family of $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi, an independent presentation of a subset of bigraphical calculi, and provided a simple operational semantics in a structural style, which we have shown corresponds to a non-aliasing bigraphical semantics. This contribution in itself, provides bigraphical calculi with a novel self-contained syntactically founded semantics (the standard presentation being firmly based on the categorical foundations of bigraphs [Mil06]).

Finally, to exemplify our contributions, we have shown that we may model the nondeterministic κ-calculus as a $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculus. In doing so, we have paved the way for further experimentation on languages based on bigraphs for studying biological systems.

Related and Future Work In a sequel paper [DDK08], we shall build upon the preliminary work presented here, and present a language encompassing both domain-level protein-protein interaction, compartments and transport among these. In that paper, we shall reap the benefit of having built a simple, self-contained presentation of the subset of bigraphs that we need.

In the Introduction, we have already discussed the background and several sources of inspiration for $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi. We should also, however, comment on other novel languages or models based on bigraphs.

In [BDE$^+$06], Birkedal et al. evaluated the use of bigraphs for building, as BRSs, so-called plato-graphical models of context-aware systems in the domain of mobile ubiquitous systems. In [Els06], Elsborg continues this investigation. The authors encode and analyse a MiniML-like calculus with references and use this language to interact with direct representations of sensor networks. This work focuses on context-aware systems, in particular the location aspect of context, and the goal is to represent and analyze a minimalistic location-aware model as a plato-graphical (BRS) model.

The CosmoBiz research project (Computer Supported Mobile Adaptive Business Processes) at the IT University of Copenhagen has as aim to provide formalisations and implementations of business process languages for mobile and adaptive business processes [HNB$^+$07].

In [BGH$^+$08a], Bundgaard et al. present a higher-order variant of WS-BPEL [TC07], and shows how this language may be formalized in binding bigraphs. In the companion tech report [BGH$^+$08b] the language is also implemented and simulated with the help of the BPL tool.

In this paper, we have focused on reaction and dynamic correspondences, not contextual equivalences. As discussed in the introduction, we wish to start by focusing on finding good abstractions for modelling biological entitities and events. In future work, we may wish, however, to investigate contextual equivalences. In doing so, it is an obvious step to try to use the bigraphical framework for bisimulations that are congruences via the derivation of minimal labels [Mil06]. (More recently, Bonchi et al. have studied an alternative approach to minimal labels called a *saturated* semantics [BKM06].) However, as we have modified the definition of the reaction relation to incorporate (negative) side-conditions and require the context to be link-mono, it needs to be studied how to update the framework, accordingly. In doing so, one may look to the experiences in deriving labels for graph transformation systems that have rules with negative application conditions (using the similar, so-called *borrowed context* approach [EK06]) by Rangel et al. [RKE08].

Such studies may also be a first step towards relating the bigraphical theory for bisimulation congruences to the studies of meta-theoretical theorems con-

cerned with establishing congruential behavioral equivalences for syntactic rule formats for structural operational semantics [MRG07].

Acknowledgements The authors thank Søren Debois, Mikkel Bundgaard, Lars Birkedal, Vincent Danos, and Robin Milner, for many useful discussions and suggestions during the development of this work. Some of this work was developed while the first author was visiting Catuscia Palamidessi's group at Laboratory for Informatics at École Polytechnique, Paris, and the second author was a research fellow in the same group.

References

[BDE+06] Lars Birkedal, Søren Debois, Ebbe Elsborg, Thomas Hildebrandt, and Henning Niss. Bigraphical models of context-aware systems. In Luca Aceto and Anna Ingólfsdóttir, editors, *FOSSACS '06: Proceedings of 9th International Conference on Foundations of Software Science and Computation Structures*, volume 3921 of *LNCS*, pages 187–201. Springer-Verlag, March 2006.

[BDGM07] Lars Birkedal, Troels Christoffer Damgaard, Arne J. Glenstrup, and Robin Milner. Matching of bigraphs. *Electronic Notes in Theoretical Computer Science*, 175(4):3–19, 2007.

[BGH+08a] Mikkel Bundgaard, Arne John Glenstrup, Thomas Hildebrandt, Espen Højsgaard, and Henning Niss. Formalizing higher-order mobile embedded business processes with binding bigraphs. In Doug Lea and Gianluigi Zavattaro, editors, *Proceedings of the 10th international conference on Coordination Models and Languages (Coordination'08)*, volume 5052 of *Lecture Notes in Computer Science*, pages 83–99. Springer Verlag, 2008.

[BGH+08b] Mikkel Bundgaard, Arne John Glenstrup, Thomas Hildebrandt, Espen Højsgaard, and Henning Niss. Formalizing WS-BPEL and higher order mobile embedded business processes in the bigraphical programming languages (BPL) Tool. Technical Report TR-2008-103, IT University of Copenhagen, 2008.

[BH06] Mikkel Bundgaard and Thomas Hildebrandt. Bigraphical semantics of higher-order mobile embedded resources with local names. In Arend Rensink, Reiko Heckel, and Barbara König, editors, *Proceedings of Graph Transformation for Verification and Concurrency Workshop 2005*, volume 154 of *Electronic Notes in Theoretical Computer Science*, pages 7–29. Elsevier, 2006.

[BKM06] Filippo Bonchi, Barbara König, and Ugo Montanari. Saturated semantics for reactive systems. In *Proceedings of 21st IEEE Sympo-*

sium on Logic in Computer Science (LICS'06), pages 69–80. IEEE Computer Society, 2006.

[BPL07] The BPL Group. BPLweb—the BPL tool web demo, 2007. IT University of Copenhagen, Denmark. Prototype.

[BS06] Mikkel Bundgaard and Vladimiro Sassone. Typed polyadic pi-calculus in bigraphs. In *Proceedings of the 8th ACM SIGPLAN international conference on Principles and Practice of Declarative Programming 2006*, pages 1–12, 2006. Invited talk.

[Car04] Luca Cardelli. Brane calculi - interactions of biological membranes. In *Computational Methods in Systems Biology*, pages 257–278. Springer, 2004.

[DB06] Troels C. Damgaard and Lars Birkedal. Axiomatizing binding bigraphs. *Nordic Journal of Computing*, 13(1-2):58–77, 2006.

[DDK08] Troels C. Damgaard, Vincent Danos, and Jean Krivine. A language for the cell. Technical Report TR-2008-116, IT University of Copenhagen, Rued Langgaards Vej 7, DK-2300 Copenhagen V, December 2008.

[Deb06] Søren Debois. Stenning's protocol in bigraphs. Unpublished manuscript, September 2006.

[Deb08] Søren Debois. *Sortings and Bigraphs*. PhD thesis, IT University of Copenhagen, 2008.

[DFF+07] Vincent Danos, Jérôme Feret, Walter Fontana, Russell Harmer, and Jean Krivine. Rule-based modelling of cellular signalling. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *CONCUR*, volume 4703 of *LNCS*, pages 17–41, 2007. Tutorial paper.

[DFFK07] Vincent Danos, Jérôme Feret, Walter Fontana, and Jean Krivine. Scalable modelling of biological pathways. In Z. Shao, editor, *Proceedings of APLAS 2007*, volume 4807, pages 139–157, 2007.

[DL04] Vincent Danos and Cosimo Laneve. Formal molecular biology. *Theor. Comput. Sci.*, 325(1):69–110, 2004.

[EK06] Hartmut Ehrig and Barbara König. Deriving bisimulation congruences in the dpo approach to graph rewriting with borrowed contexts. *Mathematical Structures in Computer Science*, 16(6):1133–1163, 2006.

[Els06] Ebbe Elsborg. Bigraphical location models. Technical Report 94, IT University of Copenhagen, Rued Langgaards Vej 7, DK-2300 Copenhagen V, September 2006.

[EPS73] Hartmut Ehrig, M. Pfender, and H. J. Schneider. Graph Grammars: An Algebraic Approach. In *IEEE Conf. on Automata and Switching Theory*, pages 167–180, Iowa City, 1973.

[GM07] Davide Grohmann and Marino Miculan. Reactive systems over directed bigraphs. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR'07)*, volume 4703 of *Lecture Notes in Computer Science*, pages 380–394. Springer-Verlag, 2007.

[HNB+07] Thomas Hildebrandt, Henning Niss, Mikkel Bundgaard, Kjeld Schmidt, and Thomas Jensen. Computer supported mobile adaptive business processes: Position paper on the cosmobiz research project (2007-2010), 2007. Project position paper.

[JM04] Ole H. Jensen and Robin Milner. Bigraphs and mobile processes (revised). Technical Report UCAM-CL-TR-580, University of Cambridge, February 2004.

[KMT08] Jean Krivine, Robin Milner, and Angelo Troina. Stochastic bigraphs. In *Porc. of MFPS'08, 24th Conference on the Mathematical Foundations of Programming Semantics*, volume to appear of *ENTCS*. Elsevier, 2008.

[Mil80] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[Mil05] Robin Milner. Axioms for bigraphical structure. *Mathematical Structures in Computer Science*, 15(6):1005–1032, 2005.

[Mil06] Robin Milner. Pure bigraphs: structure and dynamics. *Information and Computation*, 204(1):60–122, 2006.

[Mil09] Robin Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009. Forthcoming.

[MRG07] Mohammad Reza Mousavi, Michel A. Reniers, and Jan Friso Groote. Sos formats and meta-theory: 20 years after. *Theor. Comput. Sci.*, 373(3):238–272, 2007.

[PQ05] Corrado Priami and Paola Quaglia. Beta binders for biological interactions. In *Computational Methods in Systems Biology*, volume 3082, pages 20–33, 2005.

[PRSS01] Corrado Priami, Aviv Regev, William Silverman, and Ehud Shapiro. Application of stochastic name-passing calculus to representation and simulation of molecular processes. *Infomation Processing Letters*, 80(1):25–31, 2001.

[REKE99] G. Rozenberg, H. Ehrig, H.-J. Kreowski, and G. Engels, editors. *Handbook on Graph Grammars and Computing by Graph Transformation Vol. 2 (Specifications and Programming)*. World Scientific, Singapore, 1999.

[REKM99] G. Rozenberg, H. Ehrig, H.-J. Kreowski, and U. Montanari, editors. *Handbook on Graph Grammars and Computing by Graph Transformation Vol. 3 (Concurrency, Parallelism and Distribution)*. World Scientific, Singapore, 1999.

[RKE08] Guilherme Rangel, Barbara König, and Hartmut Ehrig. Deriving bisimulation congruences in the presence of negative application conditions. In Roberto M. Amadio, editor, *FoSSaCS*, volume 4962 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2008.

[Roz97] Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. 1: Foundations*. World Scientific, Singapore, 1997.

[RPS+04] Aviv Regev, Ekaterina M. Panina, William Silverman, Luca Cardelli, and Ehud Shapiro. Bioambients: An abstraction for biological compartments. *Theoretical Computer Science*, 325:141–167, 2004.

[RSS01] Aviv Regev, William Silverman, and Ehud Shapiro. Representation and simulation of biochemical processes using the π-calculus process algebra. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 6, pages 459–470. World Scientific Press, 2001.

[TC07] OASIS WSBPEL Technical Committee. Web Services Business Process Execution Language, version 2.0. `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf`, 2007.

[TeR03] TeReSe. *Term Rewriting Systems*. Number 55 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2003.