The IT University of Copenhagen

Developing Bigraphical Languages

Troels Christoffer Damgaard

A PhD Dissertation
Presented to the Faculty of the IT University of Copenhagen
in Partial Fulfillment of the Requirements of the PhD Degree

December 2008

Abstract

In this dissertation, we study bigraphical languages—languages based on the theory for bigraphs and bigraphical reactive systems developed by Milner and coworkers.

We begin by examining algebraic theory for binding bigraphs. We give a term language for binding bigraphs and develop a complete axiomatization of structural equivalence. Along the way, we develop a set of normal form theorems, which prove convenient for the analysis of bigraphical structure.

We examine next the central problem of matching, that is, to determine when and where the left-hand side of a bigraphical reaction rule matches a bigraph. We give an inductive characterization in the form of a set of rules for inferring matching for binding bigraphs and show that the characterization is both sound and complete.

We then develop a matching algorithm that works on terms for bigraphs. We start by specializing the characterization above to include application of structural congruence. We isolate a class of *normal* inferences, and prove that normal inferences are sufficient for inferring all matches. The matching algorithm relies on building normal inferences mechanically. An implementation of the algorithm is at the core of the BPL Tool, a prototype tool for experimenting with bigraphical reactive systems.

In a second line of work, we study bigraphical reactive systems as a vehicle for developing a language to model biochemical reactions at the level of cells and proteins. We discuss and isolate $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi, a family of bigraphical reactive systems that we deem sufficient for the language. We develop a self-contained presentation of the syntax and operational semantics for $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi that exploits the restrictions we have made on the family. We also treat how one may extend certain bigraphical reaction rules to include negative contextual side-conditions. As an example, we show that the nondeterministic κ-calculus (due to Danos and Laneve) can be modelled.

Finally, we build on our study above and develop a formal language, the $\mathcal{C}$-calculus, for modelling low-level interaction inside and among cells. At the core of the calculus lies a model of formal proteins and membranes. In addition, formal channels between compartments allow us to model an intermediate state in cell fusion or division, regulated by diffusion. A user models in the $\mathcal{C}$-calculus by refining a set of core rules, each of which encapsulates a core biological reaction. We illustrate the calculus with two examples, one that models the firing of a G-protein coupled receptor protein, and another that models the formation of clathrin-coated cytoplasmic vesicles through budding.

Acknowledgements

First, I would like to thank my supervisor, Lars Birkedal, for his persistent trust and guidance, and for his patience the many times I came to his office rambling about yet another idea that I had gotten.

I would also like to thank Robin Milner, who on numerous occasions provided valuable insight and plenty of inspiration, and who, together with Lucy, hosted Lena and me on an enlightening trip to Cambridge.

I thank Catuscia Palamidessi and the researchers and staff in the Comète group at the Laboratory for Informatics for hosting my stay at École Polytechnique, Paris.

During the stay in Paris, Jean Krivine, co-researcher and friend, along with Soraya Guénifi and their friends and family, took Lena and me under their wing, providing us with an unforgettable experience. In particular, I would like to thank Hélène Raux and Samuel Alizon for their helpfulness and for letting us stay in their lovely furnished apartment. Particular mention also goes to Vincent Danos, for his enthusiasm and for providing inspiration and deep insight—and, for teaching me how a true French researcher dines. Some day, this barbarian shall repay your hospitality by actually learning your language.

This work was developed with the strong support of all the current and former members of the Bigraphical Programming Languages group at the IT University of Copenhagen. Thomas Hildebrandt, Arne Glenstrup, Henning Niss, Martin Elsman, Mikkel Bundgaard, Søren Debois, Ebbe Elsborg, and Espen Højsgaard all deserve special mention. They have all influenced the body of work presented in this dissertation with valuable collaboration, incalculable discussions, and constructive critique.

I would also like to thank the members of the larger Programming, Logic, and Semantics group at the IT University of Copenhagen, for providing an enjoyable working environment. I mention, in particular, Bodil Biering whose sharp wit has been thoroughly missed.

To my family, in particular Eva, Hans Christian, and Stephan, and to my friends—I thank you deeply for your support and patience in my extended periods of absence.

And finally, to my beloved fianceé, Lena, who through the entirety of my Ph.D. has sparred with me, challenged me, trusted me, and cared for me—even going to the length of learning how to use an electric drill—I simply cannot thank you enough.

Troels Damgaard

Copenhagen, December 2008

Table of Contents

Chapter 1

Introduction

This dissertation comprises five papers; together they revolve around testing the thesis that programming languages may be based on the theory for bigraphical reactive systems. We present a body of work that spans contributions to bigraphical theory and implementation, and the development of a concrete bigraphical language.

The theory of bigraphs and bigraphical reactive systems is due to Milner and colleagues [JM04, LM04, Mil05, Mil06b]. Bigraphs provide a graphical model of computation that focuses at the same time on mobile *locality* and *connectivity*. The initial aim was to develop a vehicle for unifying the behavioral theories for process calculi for concurrency and mobility. Milner also suggested as a long-term aim to use bigraphs to model directly and reason about context-aware systems in the domain of mobile ubiquitous computing [Mil06b].

The Bigraphical Programming Languages (BPL) project at the IT University of Copenhagen[1], which financed this Ph.D., set out to investigate how to develop new programming and specification languages for mobile and distributed systems on the theoretical foundation of bigraphs [Bir04, BBD$^+$06]. Up until now, there has been a particular focus on context-aware systems and ubiquitous computing.

In this dissertation, our aim has been to further the investigation of bigraphically based languages (in short, bigraphical languages). We have set two main goals: (1) to develop a sound formal foundation for a tool for experimenting with bigraphical languages; and, (2) to develop a novel bigraphical language for the domain of cellular biology. The fulfillment of the first goal is a vital step in allowing direct experimentation with bigraphical languages. The fulfillment of the second goal gives an exemplifying answer to the central question underlying the BPL project: What is a bigraphical language? Only through the development of concrete new languages can we really test the thesis, that programming languages may be based on bigraphs. And, as we shall illustrate in the dissertation, the bigraphical framework seems well-poised as a foundation for experimenting with models and languages for biological systems.

[1]Funded by the Danish Research Agency grant: 2059-03-0031 (LaCoMoCo).

Contributions in this Dissertation

The trajectory of the work presented here begins with the development of a bigraphical equational theory, brings us across a characterization and subsequent implementation of matching for bigraphs, and ends with the presentation of a domain-specific bigraphical language for expressing the reactions in biological cells. Below, we summarize the contributions contained in this dissertation. The approach we take may be divided into two steps, corresponding to the two goals set above.

Our first step establishes the foundation for a tool for experimenting with bigraphs. Through the study of algebraic theory for bigraphs with binders, including the development of normal forms and an axiomatization of structural equivalence for bigraphs with binders, we arrive at a complete inductive characterization of matching. This characterization forms the basis of a provably sound and complete matching algorithm; this algorithm, in turn, is at the core of the implementation of a prototype tool for working with bigraph matching and reaction—the first version of the BPL Tool was released in December 2007.

In the second step, our work turns to the development of languages. We develop a formal language, the $\mathcal{C}$-calculus, for modelling low-level interaction inside and among biological cells. The language combines a protein-model and rule-based modelling á la the κ-calculus [DL04, DFF$^+$07, DFFK07] with a membrane model inspired by the brane family of calculi [Car04a, DP04]. The extension of a κ-like calculus with dynamic compartments is a novel contribution in itself; in addition, the calculus also includes a novel abstraction for capturing partially fused compartments and associated diffusion of protein-complexes between such compartments.

Our work on developing the $\mathcal{C}$-calculus highlighted that the presentation of certain reaction patterns may be more concisely expressed by allowing contextual negative side-conditions in reaction rules; and, that the presentation of a particular calculus may be hampered by the versatility of the bigraphical model. By catering for a wide variety of languages, the presentation of a *single* language may be overloaded with bigraphical idiosyncracies, such as those inherited from the underlying categorical model. To counter such problems we investigate the use of bigraphs for modelling biological interaction in a separate pre-study; in particular, we treat how to extend reaction rules to include negative side-conditions. We also make the effort to treat carefully a self-contained presentation of the operational semantics for $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi, a subset of bigraphical calculi. We utilize this work to give a presentation of the $\mathcal{C}$-calculus void of most bigraphical idiosyncracies.

Structure of the Dissertation

In the remaining sections of the present chapter, we provide a brief background on the bigraph model and on research in bigraphs. We start by giving a quick introduction to bigraphs with binders and their reactive systems in Section 1.2.1. We hope to provide readers unfamiliar with bigraphs with enough background to follow the following discussions. In Section 1.2.2, we briefly overview some fundamental studies and results for the bigraphical framework.

In Chapter 2, we turn to the contributions in this dissertation. For each of the papers in the dissertation, we give a summary of the goals and the approach taken, and discuss relations to other research. In Section 2.3, we conclude on the obtained results, and in Section 2.4, we outline directions for future work.

The main body of this dissertation is contained in the five Chapters 3 through 7. In Chapter 3, we develop a term language, a set of normal form theorems, and an associated equational theory on terms for binding bigraphs that captures graph isomorphism on the term level. Subsequently, in Chapter 4, we develop a complete inductive characterization of matching, that is, for describing when and where the left-hand side of a reaction rule matches a bigraph. In Chapter 5, we describe a provably sound and complete matching algorithm for bigraphical reactive systems. The algorithm forms the basis of the implementation of the BPL tool, a first implementation of bigraphical reactive systems. In the pre-study in Chapter 6, we lay the foundation for using bigraphical calculi to experiment with models and languages for biological systems. Finally, in Chapter 7, we develop the $\mathcal{C}$-calculus, a novel language based on bigraphs that allows a user to experiment with reactions inside and among biological cells.

We aim the dissertation at readers who are familiar with process calculi for distributed and mobile systems, in particular, the family stemming broadly from the work by Milner on CCS [Mil89] and the π-calculus [Mil99]. We also assume familiarity with calculi that introduce explicit representations of locality, such as the family of ambient calculi launched by Cardelli and Gordon [CG98, CG00].

1.1 Papers

In order of appearance, the following papers are included in this dissertation:

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

[DGBM07] Troels Christoffer Damgaard, Arne J. Glenstrup, Lars Birkedal, and Robin Milner. An inductive characterization of matching in binding bigraphs. Manuscript submitted for publication, September 2007.

[GDBH07] Arne J. Glenstrup, Troels Christoffer Damgaard, Lars Birkedal, and Espen Højsgaard. An implementation of bigraph matching. Manuscript submitted for publication, October 2007.

[DK08] Troels C. Damgaard and Jean Krivine. A generic language for biological systems based on bigraphs. Technical Report TR-2008-115, IT University of Copenhagen, Rued Langgaards Vej 7, DK-2300 Copenhagen V, December 2008.

[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.

The papers are presented in their original form, except for minor typographical changes and the inlining of erratta for [DB06] (as described in the preface of Chapter 3). The second paper [DGBM07] is an extended and revised version of [BDGM07], which provides more explanations and examples, and includes extensive details for the proof of completeness of the characterization, including a self-contained section on the algebraic properties of wirings and parallel product. The third paper [GDBH07] is a direct continuation of the paper on matching. Both the second and third paper are under submission for publication. The tech reports [DK08] and [DDK08] are more recent work, and will form the basis for later publication.

The following papers have been omitted from the dissertation.

[DD05b] Søren Debois and Troels C. Damgaard. Bigraphs by Example. Technical Report TR-2005-61, IT University of Copenhagen, March 2005.

[DB05] Troels C. Damgaard and Lars Birkedal. Axiomatization of binding bigraphs (revised). Technical Report 71, IT University of Copenhagen, October 2005.

[BBD$^+$06] Lars Birkedal, Mikkel Bundgaard, Troels Christoffer Damgaard, Søren Debois, Ebbe Elsborg, Arne John Glenstrup, Thomas Troels Hildebrandt, Robin Milner, and Henning Niss. Bigraphical programming languages for pervasive computing. In Thomas Strang, Vinny Cahill, and Aaron Quigley, editors, *Proceedings of Pervasive 2006 International Workshop on Combining Theory and Systems Building in Pervasive Computing*, pages 653–658, May 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.

The tech report [DD05b] is a basic study of the modelling capabilities of bigraphs; it contains a series of encodings of smaller examples in bigraphical reactive systems. The papers [DB05] and [BDGM07] are superseded by the included papers [DB06] and [DGBM07] (although [DB05] includes full details for some proofs omitted from [DB06]). Finally, the publication [BBD$^+$06] is a position paper.

1.2 Background

In Section 1.2.1, we give a quick informal introduction to bigraphs with binders and their reactive systems. (The section is essentially an expanded version of the brief introduction to bigraphs in Chapter 3.) Our main purpose is to provide the reader unfamiliar with bigraphs with enough background to follow the discussion of our work in Chapter 2. In Section 1.2.2, we provide some pointers to some of the fundamental studies and results underlying the bigraphical framework. For each of the papers in the dissertation, we discuss further related work in Chapter 2.

The reader already familiar with binding bigraphs may wish to skip Section 1.2.1. For a more formal account, we refer the reader to Section 2 in Chapter 4. For quick reference, the Appendix in Chapter 3 contains a terse recap of the main definitions underlying binding bigraphs; below, we shall provide the reader with a few pointers to those formal definitions.

1.2.1 A Brief Introduction to Bigraphs

The theory of bigraphs has been developed by Milner and colleagues [JM04, LM04, Mil05, Mil06b]. Bigraphs and their reactive systems have been developed as a graphical model of computation that focuses on both mobile *locality* and *connectivity*. The theory has been been developed with two principal aims: (1) to be able to model directly important aspects of ubiquitous systems, and (2) to provide a unification of existing theories by developing a general theory, in which many existing calculi for concurrency and mobility may be represented, with a uniform behavioural theory. The latter is achieved by representing the dynamics of bigraphs by an abstract definition of reaction rules from which a labelled transition system may be derived in such a way that an associated bisimulation relation is a congruence relation.

We can use bigraphs to model systems. Figure 1.1 shows an example of a bigraph model. It consists of two *roots* (dashed boxes), *nodes* (solid boxes), and *links* (green lines). Each node has a *control* (in sans serif) indicating the number and type of *ports* for linkage. Ports can be either *free* or *binding* — the latter indicated by circular attachments. To determine the set of controls, we can use in a bigraphical reactive system, we give a bigraphical *signature*.

The bigraph E is supposed to model a part of a building. In one location, there is a server with a secret inside. The secret is linked to a binding port of the server. The free port of the server is linked (supposedly by some kind of network connection) to a pc inside an office in another location. The office also contains two pdas linked to each other.

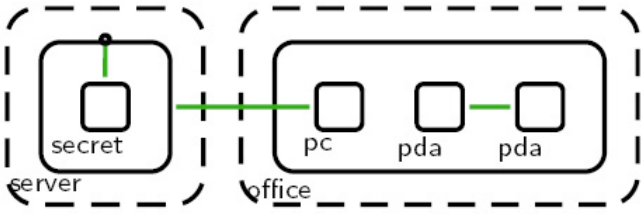


Figure 1.1: E — a bigraph model of an office

Bigraphs can contain *sites* (gray boxes), and inner or outer *names* (in red). Both roots and sites are ordered. The bigraph F in Figure 1.2 has two sites numbered 0 and 1, and two inner names, x *located* at site 0 and z *global* (i.e., not located). The bigraph G in Figure 1.3 has two corresponding outer names, x located at its first root, and z global. We also use circular attachments to denote that x is local to the

first root.

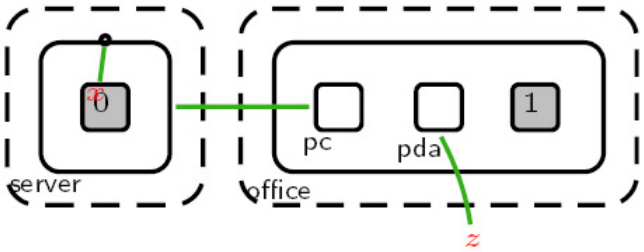


Figure 1.2: F — a bigraph context

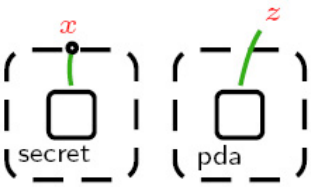


Figure 1.3: G — a bigraph that composes with F

A bigraph comprises two structures, a *place graph*, which is a forest of nodes with unordered siblings, rooted at the roots, and with sites occuring only as leaves; and a *link graph*, which is a hypergraph connecting ports of the nodes with each other and with inner and outer names. Figure 1.4 shows the place and link graph of F — letting r's and s's denote roots and sites, respectively.

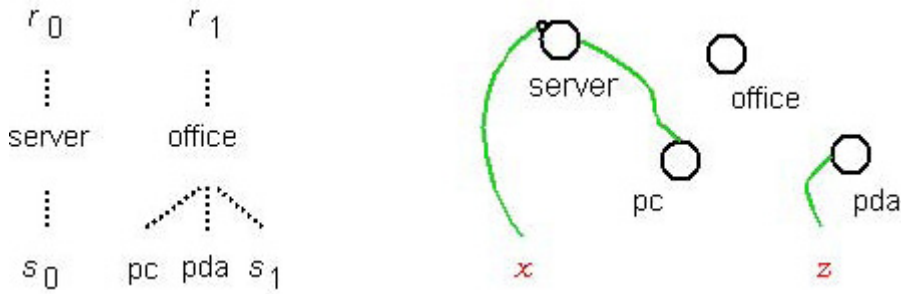


Figure 1.4: Place and link graph of F

Milner originally worked on *pure* bigraphs [Mil01, Mil06b], which only have global names and free ports. In pure bigraphs the place and link graphs are entirely orthogonal.

The bigraphs E, F and G are examples of *binding* bigraphs, which enforce a *scoping* discipline. We call binding ports and local names *binders* — the circular attachment is used to emphasize their likeness. The scope rule for binding bigraphs

requires that all inner names or ports linked to a binder must be located *below* (in the place graph) the binder (cf. Definition 13 in Chapter 3). For instance, the leftmost link in F adheres to this discipline: the binder is located on the server node, and this node is located above the site 0, where x is located. On the other hand, the global inner name z could not be linked to any binder, as z is a global (i.e., non-located) inner name. We discuss the motivation for introducing binding in Section 1.2.2.

Bigraphs are composable structures. We can compose F and G by plugging the sites of F with the roots of G, joining equal inner and outer names; F and G compose to form E. We write $E = F \circ G$, or just $E = FG$.

Not all bigraphs are composable. Bigraphs F and G compose exactly because G has roots and outer names corresponding to the sites and inner names of F. The inner and outer *interfaces* of a bigraph registers this information, and hence determines which bigraphs can be composed. An interface has the form $\langle n, \vec{X}, Y \rangle$ for an integer n, a set of names Y, and an array of disjoint subsets of the names Y, $\vec{X} = (X_0, \ldots, X_{n-1})$ of length n. The bigraph F has the inner interface (or innerface) $\langle 2, (\{x\}, \emptyset), \{x, z\} \rangle$. The first and third component registers that F has 2 sites, and that it has the inner names x and z, respectively. The second component, the array $(\{x\}, \emptyset)$, registers for each site the local names for that site; hence, the innerface of F registers that x is local to the first site (and that no other names are local). The outer interface (or outerface) of F is $\langle 2, (\emptyset, \emptyset), \emptyset \rangle$ registering that F has 2 roots, and that F has no outer names. The second component of an outerface registers for each root the names local to that root; thus, as F has no outer names, there are no local names. We write $F : \langle 2, (\{x\}, \emptyset), \{x, z\} \rangle \rightarrow \langle 2, (\emptyset, \emptyset), \emptyset \rangle$, and call this the interface of the entire bigraph. Bigraph G, on the other hand, has the interface $\langle 0, (), \emptyset \rangle \rightarrow \langle 2, (\{x\}, \emptyset), \{x, z\} \rangle$.

We see that F and G compose, because the innerface of F is equal to the outerface of G. The identity for composition on a particular interface is called the *identity* bigraph. Given an interface $I = \langle n, \vec{X}, Y \rangle$, the identity bigraph on I, $\mathsf{id}_I : I \rightarrow I$, maps n sites severally to n roots and map all inner names $y \in Y$ to equal outer names.

Formally, a bigraph is defined as a pairing of a place graph and a link graph. Eliding some detail, taking interfaces as objects, bigraphs as morphisms, and the identity bigraphs as the categorical identities, we have essentially a category of binding bigraphs.[2] (See also the Appendix in Chapter 3 for more details.)

Both E and G have no sites or inner names, their innerfaces are $\langle 0, (), \emptyset \rangle$. We say that E and G are *ground* and call them *agents*. Agents are important, as reaction in a bigraphical reactive system can only take place on agents. To reflect this importance, we write the innerface $\langle 0, (), \emptyset \rangle$ of agents simply as ϵ.

We can also combine bigraphs with a *tensor product* (denoted by $\otimes$), which is simply juxtapositioning of roots. For tensor product we require that both inner and outer names be disjoint. From tensor product, we can derive straightforwardly a

[2] We are eliding here the discussion of *abstract* vs. *concrete* bigraphs. Only the former constitute a category, concrete bigraphs form a so-called supported pre-category, as composition is not always defined. We discuss this further in the discussion on Syntax for Bigraphs in Section 2.1.

parallel product || that aliases common outer names, and, a *prime product* | that aliases common outer names and merges several roots into one root.

In Chapters 3, 4, and 5, we shall be particularly concerned with three classes of bigraphs, which prove important in a decompositional analysis of bigraphs: (1) *prime* bigraphs are those with only a single root, and only local inner names; (2) *discrete* bigraphs are bigraphs, where all links to a global outer name are one-to-one; and, (3) *name-discrete* bigraphs, are those where all links to both local and global outer names are one-to-one (refer to Definition 14 in Chapter 3 for the full definition of discreteness).

We build bigraphical reactive systems (BRSs) by giving a signature (to determine the controls) and a set of reaction (or rewriting) rules. Reaction rules are expressed essentially as a pair of bigraphs, such as those in Figure 1.5.

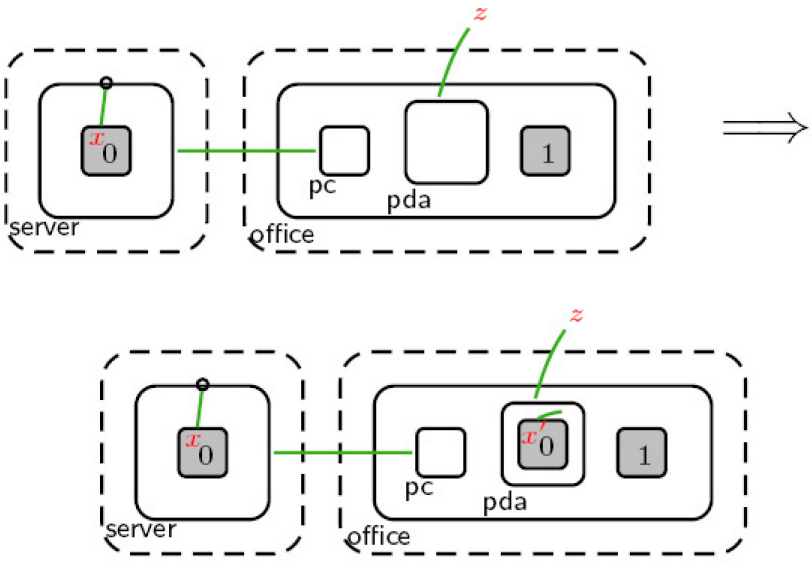


Figure 1.5: A bigraph reaction rule

Eliding some details, we might interpret this rule as saying: If a pc in some office is linked to a server, a pda in the same office may use the pc as a gateway to copy data from the server.

The left-hand side of a rule is called the *redex* and the right-hand side the *reactum*. We use rules to rewrite bigraph agents. To do so, we *match* the rule to some part of the agent, and we find a bigraph *parameter* to fill the sites of the redex — essentially, to supply the missing inner structure — and find a bigraph *context* to supply the missing outer structure.

Suppressing some detail, a redex R matches a ground agent a, if a decomposes, such that,

$$a = C\,(R \otimes \mathsf{id}_Z)\,d,$$

for a context C and a discrete parameter d.[3] The bigraph id_Z is shorthand for the identity on the interface $\langle 0, (), Z\rangle$, and serves to allow the parameter d to export some extra names Z past the redex, and directly to the context C.

The reaction rule in Figure 1.5 can be used to rewrite E, essentially with G as parameter and with the empty context.[4]

1.2.2 The Foundation for Bigraphs

The theory of bigraphs and BRSs stem from work on deriving labelled transition systems (LTSs) for reactive systems [Sew02, LM00, JM03], and from an effort by Milner and coworkers to provide a general theory able to unify behavioral aspects of mobile process calculi [Mil06b, JM04]. Thus, in the collective eyes of the mobile process calculi community, bigraphs and BRSs is a "grand unifying model" (quoting Nestmann [Nes06]).

Bigraphs and BRSs have been designed inherently by the needs of a meta-calculus. When modelling in BRSs, we are given the freedom to choose both our own static structures *and* the reconfiguration semantics for those structures. We essentially implement a domain-specific calculus for modelling both the static structure and the dynamic semantics directly. This has consequences for the modelling primitives and mechanisms we are given to work with. As we have reviewed in an earlier litterature study (in [Dam06]), from a modelling perspective bigraphs and BRSs have much in common with the fields of term rewriting [BN98, TeR03] and algebraic (or double-pushout) graph transformation [Roz97, REKE99, REKM99]. We shall discuss some selected relations to term rewriting and graph transformation when discussing related work in Section 2.1.2.

Up until now, there has been a considerable amount of work on recapturing aspects of mobile process calculi in the CSP, CCS, and ACP tradition of process algebra for modelling and reasoning about concurrent systems [Bae05]. There has been less work on designing novel models or languages based on bigraphs. We postpone the discussion of both the recapture of existing calculi, and of the development of novel models or languages, to the section on related work in Section 2.2.1.

In the remainder of this section, we discuss the background for bigraphs with binders, and comment on a number of fundamental results that seek to strengthen or extend the bigraphical framework.

Pure Bigraphs and Bigraphs with Binders

Pure bigraphs treat connectivity (i.e., the link graph) and locality (i.e., the place graph) orthogonally, and do not include any notion of scope or locality of linkage. The intuition underlying the place graph is to model the structure of the abstract

[3] Formally, the context also has to be *active* (see Section 2, Chapter 4).

[4] This is only essentially correct. Choosing G as the parameter would require us to choose $\mathsf{id}_Z = \mathsf{id}_{\{z\}}$, but then $R \otimes \mathsf{id}_Z$ is not defined (recall that $\otimes$ requires outer names to be disjoint). We can simply rename z in G to z', though, and provide a context that rewires z' and z from R, and we have a valid match.

syntax tree of a term, while the intuition for links is to provide bigraphical languages with a first-class model of pure names. Open (or free) links are assigned names, and closed links, which have no name, correspond to α-convertible names. (As a side-remark, our work in Chapter 6 formalizes that intuition.) By taking links as primitives, bigraphs also incorporate the power to model directly explicit substitutions of names as link graphs.

Binding bigraphs introduce mechanisms to allow us to relate and constrain the two constituent structures. As outlined in Section 1.2.1, binding bigraphs add binding ports to nodes, an associated scope rule, and names localized to roots to transfer the scope discipline across composition.

Building on the intuition for pure links given above, bound links provide bigraphs with a model of lexically scoped *bound* names, or, in programming language-terms, lexically scoped name-valued formal variables. The scope discipline ensures essentially, that the context A in the composition $A\,B$ cannot make global links in B local; it may however make local links in B global (in bigraph-lingo, by *concretizing* them). The intuition here is that a context may only *instantiate* bound names, not treat already instantiated names as variables.

Introducing locality and binding allows us to model both *copying* of linkage inside a parameter, and *change* of linkage connected to a parameter. Thus, binding and localized linkage appear naturally in modelling, as it allows us more control over parametric reaction.

As has been shown by Jensen and Milner for the asynchronous π-calculus [JM04], and by Bundgaard and Sassone for the typed polyadic π-calculus [BS06], introducing binders in combination with located names provides us with enough expressivity to faithfully capture process calculi like the π-calculus. This concerns both the internal reactive semantics and the labelled semantics.[5] Names bound by a ν or by an input-prefix in the π-calculus are captured in binding bigraphs using closed, or bound and closed links, respectively. We underline the point: In the π-calculus, we typically use the term *bound* names to refer to both names bound by a ν—such as x in P in $\nu x.P$—and, names bound by a binding prefix—such y in P in $x(y).P$. It is important to note that in binding bigraphs these two notions are treated separately. In short, the intuition is: ν-bound names correspond to closed links; names, which may be substituted for other names, correspond to bound links.

In the study of bigraphical example models [DD05b], we give another example of the modelling power of binding bigraphs based on a case-study of using binding bigraphs to model event-driven systems.

More recently, Milner has investigated extensions of binding bigraphs, where names are allowed to be multi-located, that is, located at several roots or several sites [Mil04, Mil06a]. The research on extensions of binding bigraphs with a more fine-grained control over the locality of names, is motivated by the lacking ability to write convenient wide reaction rules working on bound linkage. In binding bigraphs names can only be located at a single root. This means that names shared across

[5] At the least, up to differences between what is modelled as part of the structural congruence and as part of the reaction rules.

roots in wide redexes or reactums must be global. When we match a redex to an agent, the scope rule prevents us from treating global names in the redex as local in the context. Consequentially, in binding bigraphs, we cannot write wide rules working on bound linkage, because we need to explicitly show a binder in the rule. In graph terms, we need to show a common ancestor of the points connected to a bound link. Binding bigraphs with multi-located names lift this restriction by allowing names shared across roots to be local, and thus bound. Centrally, Milners encoding of the λ-calculus [Mil06a] uses this feature to encode succinctly non-local substitution (referred to as "substitution at-a-distance").

Sortings and Types

Sortings [Deb08] allow us to prune the bigraphs we consider, by asserting that we only consider our reactive system to contain those bigraphs that adhere to a sorting discipline. We could, for example, disallow all bigraphs with nodes of control K beside each other (such as $\mathsf{K}|\mathsf{K}$). Sortings are defined categorically via a functor from a sorted category into an unsorted category. Thus, a sorting works by restricting which bigraphs may be composed; consequentially, sorted reactive systems only contain well-sorted bigraphs.

Variants of sortings have been applied extensively for pruning the derived labelled transition systems for encodings of various calculi into bigraphs to recapture in the encoding the contextual equivalence of the original calculi (e.g., for CCS [Mil06b], typed π [BS06], Homer [BH06], and Mobile Ambients [Jen06][6]). It has also been applied to the domain of context-aware systems in [BDE$^+$06], where sortings are used for modelling *context-aware* reaction rules, by requiring for particular reaction rules that the context be sorted according to a particular sorting.

Work by Debois, Birkedal, and Hildebrandt has focused on giving a unified treatment of sortings. In [BDH06], a general theory of sortings for reactive systems over categories was defined, and it was conjectured that an extension to bigraphs was feasible. In particular, a framework was established for so-called *decomposable predicate* sortings, those sortings given by a predicate P preserved under decomposition. Recently, the lift to bigraphical reactive systems was completed [BDH08], and a refined theory based on so-called *closure sortings* is established. The scoping restrictions imposed by the above-mentioned binding variants of bigraphs can also be expressed as sortings. Stemming from work in the dissertation by Debois [Deb08], in [BDH08] it is shown that binding bigraphs with multi-located names can be recaptured via a sorting. Recall, however, that the definition of *reaction* is also changed for bigraphs with binders. In particular, all variants of binding bigraphs allow parameters with arbitrary bound wiring. This cannot be expressed directly via sorting.

In [EHS08a] (under publication as [EHS08b]) Elsborg, Hildebrandt, and Sangiorgi launch an investigation of types for bigraphs defined inductively over the structure of terms for bigraphs. The aim is to be able to model more easily than

[6] Refer to our comments on related work in Section 2.2.1 for an overview of bigraph-based languages.

sortings existing type systems for process calculi, which are often defined over the structure of terms. As an example, the authors develop an i/o-type system with subtyping for a finite π-calculus without summation. It is also a hope that type systems defined over the structure of terms may be more easily computer-verified.

Directed Bigraphs

Over a series of papers [GM07a, GM07b, GM08a, GM08b], Grohmann and Miculan have launched an investigation of *directed bigraphs*—a generalization of pure bigraphs where the link graph is symmetric. Directed bigraphs generalize the existing notion of link graph in bigraphs such that linking is oriented (i.e., directed). Formally, in standard bigraphs, the link graph maps inner names and ports to outer names and edges. Thus, there is an asymmetry between the notion of inner and outer names; the direction of links is restricted to *ascend* a bigraph, so to speak. In directed link graphs, the authors add sets of so-called *downwards* names to the interfaces of bigraphs. Thus, in directed bigraphs a port can be linked both upwards and downwards. This provides link graphs with a symmetric structure with regard to composition.

In [GM07b] it is shown that the fusion calculus without replication (due to Parrow and Victor [PV98]) may be encoded faithfully in directed bigraphs. In [GM08a], the authors provide a term language for directed bigraphs, and finally, in [GM08b], the authors study directed bigraphs over *polarized* signatures, adding negative ports that allow directed bigraphs to represent controlled access to resources modelled by edges.

We should also remark that interestingly, as noted by Debois [Deb08], directed bigraphs is the only known variant of bigraphs, whose structure cannot be recaptured using sortings.

Spatial Logics

In [CMS05b, CMS05a, CMS07] Conforti, Macedonio, and Sassone have initiated research on a framework, BiLog, for static spatial logics, which instantiate directly to (pure) bigraphical structure as a composition of a place graph and a link graph logic. The basic logic is defined on bigraph terms and is inspired and firmly supported by the axiomatization of pure bigraphs in [Mil05]. It is an *intensional* logic, in that it coincides with structural congruence, which, in general, is strongly more fine grained than any behavorial equivalence one might want to consider. Towards controlling the structural inspection power of the logic, a notion of *transparency* is defined; essentially, allowing one to consider certain parts of the bigraphical structure as *opaque* or indistinguishable. In essence, we may define certain nodes of certain controls to be opaque, blocking inspection of a term at these nodes and their descendants.

Recent, yet unpublished, work [CMS06], delve into more exemplifying instantiations of the logic, and discuss how the framework may be extended to encompass also dynamic behavior. Interestingly, it is shown that by utilizing the strong intensionality of the logic, for particular cases, standard temporal modalities such as

the next-step modality can be expressed in the logic as it is. Essentially, based on an analysis which characterizes all reacting contexts and given a particular set of reaction rules, it is possible to define a characteristic formula for every rule. This is shown for a simple encoding of CCS and an associated behavioral logic $\mathcal{L}_{\text{spat}}$ (introduced in [CL04]).

Chapter 2

Summary

In this section, we give an overview and discussion of the papers in Chapters 3 through 7.

We discuss the papers in two steps as outlined in the introduction: A first step that discusses the establishment of a theoretical foundation for developing a tool for experimenting with bigraphs and BRSs (in Section 2.1); and a second step, which concerns the use of bigraphs for biological modelling and presents the development of a concrete language based on bigraphs (in Section 2.2). In Section 2.3, we conclude and summarize the contributions, and in Section 2.4, we provide pointers for future work.

2.1 Towards a Tool for Working with Bigraphs

As outlined in the introduction, our first step focuses on establishing the foundation for implementing a prototype tool for experimenting with bigraphs.

In Chapter 3, we start by developing a firm syntactic foundation for working with the structure of bigraphs with binders. We develop a term language, a set of normal form theorems, and an associated equational theory on terms that captures graph isomorphism on the term level. Subsequently, in Chapter 4, we develop a complete inductive characterization of matching, that is, for describing when and where a redex matches an agent.

To start with, the work yields convenient syntax-based and inductive proof-techniques for reasoning about equality and matching in binding bigraphs. However, it also yields a formal foundation for the implementation of a tool for working with, and reasoning about, BRSs. Such an implementation needs an internal representation of bigraphs, and prominently needs to find matches based on this representation. The work presented in Chapters 3 and 4 provides the necessary foundation for choosing a syntactic representation of bigraphs.

Thus, concluding this step, in Chapter 5, we describe a provably sound and complete matching algorithm for bigraphical reactive systems. The algorithm forms the basis of the implementation of the BPL Tool, a first implementation of bigraphical reactive systems.

Before we turn to the discussion of the papers, let us dwell for a moment on the basis for our approach. In particular, we should discuss briefly why we focus on syntax for a graphical formalism.

Syntax for Bigraphs Formally, bigraphs come in two flavours, *concrete* bigraphs, where nodes and edges are equipped with identities (or support); and, *abstract* bigraphs, where identities have been quotiented away. We may think of concrete bigraphs as vehicles for defining the abstract bigraphs.

Essentially, the idea is that we start by assigning concrete identities to nodes and edges to give the parent and link maps underlying place and link graphs. We obtain from this so-called *concrete* place and link graphs. A concrete bigraph is a pairing of a concrete place graph and a concrete link graph.

Concrete bigraphs form a so-called supported pre-category, not a category. This is because composition is defined only for concrete bigraphs whose constituents do not share identities. We are interested in the structure of a bigraph, however, not in the concrete identities of nodes and edges. Those identities are essentially there to state the underlying maps.[1] Further, as an artefact of this manner of giving the link map, concrete link graphs may contain unconnected edges (so-called *idle* edges). Such edges are essentially "junk". Formally, *lean-support equivalence*, denoted $\eqslantless$, is defined such that $G_0 \eqslantless G_1$ iff G_0 and G_1 differ only by a bijection between their nodes and non-idle edges; idle edges are disregarded entirely. The class of *abstract* bigraphs, those bigraphs we are really interested in, are defined as $\eqslantless$-equivalence classes of concrete bigraphs. Abstract bigraphs form a full category.

In reasoning about equality for bigraphs (as we do in Chapter 3) or about matching a redex to an agent (as we do in Chapter 3), an algebraic presentation of abstract bigraphs has certain conveniencies. Rather than reasoning, for example, about equivalence of (abstract) bigraphs based on the representation sketched above, a term representation with an associated structural congruence allows us standard syntax-based equational reasoning. This is convenient, because we can find a small language for describing bigraphs and because it helps us describe in separation, in the equational theory, the possible ways the elements of the language may interact to create equal bigraphs. For instance, the elision of idle edges is captured concisely in a single axiom expressing the interaction between two elementary wirings.

Based on these considerations, we choose to study how the algebraically based understanding of equality for binding bigraphs (that we develop in Chapter 3) may also provide us with an understanding of matching. That study is documented in Chapter 4 and continued in Chapter 5.

[1] Giving identities or *support* to nodes and edges also equip the supported pre-category of concrete bigraphs with *relative pushouts*. They would otherwise be missing, because we would lack any notion of sharing of nodes or edges between bigraphs. This is important for the theories for labelled bisimulations and sortings for bigraphs. These topics are, however, out of the scope of this dissertation. We direct the reader to the works by Milner [Mil06b] or the recent dissertation by Debois [Deb08], for more on these topics.

2.1.1 Axiomatizing Binding Bigraphs

In Chapter 3, we develop a term language for binding bigraphs, an associated axiomatization of graph isomorphism for bigraphs, and a series of normal form theorems for binding bigraphs. This is done as a conservative extension of the work for pure bigraphs by Milner [Mil05]. The introduction of binding and locality turns out to require some reworking on the level of the term language; in turn, this invokes changes for both the normal forms and the equational theory.

Our Approach

The term language, the normal form and the equational theory for *pure* bigraphs utilizes the orthogonality of the place and link graph to full effect [Mil05]. Let us start by giving a brief overview of that work.

The term language provides three *placings* essentially for constructing place graphs, two *linkings* for constructing link graphs, and a single construct, an *ion*, for adding nodes. The combinators, composition and tensor product, stem directly from the categorical foundation for bigraphs. The normal form for pure bigraphs is based on so-called *discrete* decomposition; here a bigraph is decomposed into (1) a *discrete bigraph*—a place graph with a trivial link graph (all links are linked to separate outer names), and (2) a *wiring*—a bigraph with an empty place graph. In other words, the effect is to reach a description that decomposes a bigraph into its underlying place and link graph. This manner of decomposition eases the proof of completeness of the normal form considerably. Importantly, the normal form is also engineered towards establishing the completeness of the associated equational theory.

The equational theory for pure bigraphs itself is stated as a set of axioms. Most axioms are derived from the categorical foundation of composition and tensor product (e.g., axioms for associativity and commutativity). The separation of place and link graph in the term language supports a separation of the axioms concerned with place and link graph structure into three axioms expressing the basic equalities for placings and four axioms for linkings. A single axiom is concerned with ions. The separation of placings and linkings in the term language, and in the discrete decomposition provided by the normal form, is also used intrinsically in the proof of completeness of the equational theory.

In Chapter 3, we work with binding bigraphs, where the place and link graph are no longer orthogonal—the locality of names and binding ports, which serve to constrain the link graph, is dependent on the place graph. Hence, we cannot hope to decompose place and link graph entirely in a term language or a normal form for binding bigraphs. On the other hand, binding bigraphs are simply a refinement on pure bigraphs, in the sense that if we "forget" the binding of ports and locality of names, we get a valid pure bigraph.[2] We would expect a term language, a normal

[2]Formally, there is an obvious forgetful functor from the category of binding bigraphs to the category of pure bigraphs.

form, and, an equational theory to reflect this. In other words, we seek to develop a conservative extension of the algebraic theory for pure bigraphs.

This requires a careful extension of the term language and the normal form developed for pure bigraphs. It proves non-trivial to extend the term language and normal form to engineer them towards establishing and proving complete the equational theory. Centrally, we take care to define the term language and the normal form, such that we can prove completeness of the equational theory by structural induction.

In the term language, we change a placing to allow for local names, add a constant, the *concretion* $\ulcorner X \urcorner$, that maps local names X to global names X, and define a new combinator, the *abstraction* $(Y)\,P$, that converts global outer names Y in the bigraph P to local outer names.[3] Furthermore, to account for binding ports on nodes, we allow a *binding* ion $\mathsf{K}_{\vec{y}(\vec{X})}$ to bind severally sets of (local) inner names $X_0, \ldots, X_{b-1}$ to each of the b binding ports of the node K.

Subsequently, we develop a normal form, where binding is introduced as "late" as possible, and which is based centrally on a correspondent to (pure) discreteness called *name*-discreteness. In name-discrete bigraphs, all linkage to both global and local outer names is required to be trivial, that is, linked one-to-one to outer names. The term language and normal form is engineered such that it is easy to define a *syntactic* correspondent to name-discrete bigraphs called *linearity*. Linearity imposes a simple syntactical restriction on the type of linkings allowed in a term. It is easily verified that linearity is an inductive property with regard to the term language. We utilize this correspondence to full effect to allow the proof of completeness of the equational theory to be established by structural induction.

To illustrate the engineering of the term language and normal form, consider as an example the binding ion. One could have defined the binding ion to only allow single inner names (not *sets* of names) to be bound to the binding ports of nodes. In fact, the term language (with this variant of the ion) is complete for binding bigraphs, and we could have easily defined a complete normal form. However, we would not have been able to define linearity as a simple syntactic correspondent of name-discreteness; thus complicating the establishment of the normal forms and the equational theory.

In stating the equational theory, we extend the axioms for pure bigraphs with six axioms concerned with abstractions, concretions, and, binding ions, and we refine the axioms concerned with the changed placing. The main theorems state soundness and completeness of the equational theory. These results yield syntax-based equational techniques for reasoning about equivalence of bigraphs.

Discussion and Related Work

Our equational theory and associated term language for binding bigraphs closes an open problem by Milner in [Mil05]. We have already discussed in detail the work by

[3] Formally, in this case P is required to be *prime*, i.e., to have a single root and no local inner names.

Milner on the axiomatization of pure bigraphs [Mil05]; and, in Section 1.2, we have given pointers to some other fundamental results developed for bigraphs. Below, we comment on the relations between our work and some of these results.

The soundness and completeness of the axiomatization of directed bigraphs by Grohmann and Miculan [GM08a] is verified by techniques similar to ours.

The work by Elsborg, Hildebrandt, and Sangiorgi on type systems inductively built over terms for (binding) bigraphs [EHS08a], is based on the term language we develop for binding bigraphs. Consequentially, our equational theory is heavily used to establish the validity of the theory.

As discussed in Section 1.2.2, Milner has suggested a variant of bigraphs with binders and multi-located names. The equational theory treated in Chapter 3 is for bigraphs with binders and singularly located names. It is an open question to extend our work to encompass also multi-located names. We conjecture that the extension is not difficult, and that the difficulty lies in introducing binding in the first place.

The axiomatization of pure bigraphs [Mil05], has proven an important foundation for the analysis of pure bigraphs, and, as such, has been the main inspiration for BiLog, the framework for spatial logics for bigraphical structure [CMS05b] by Conforti, Macedonio, and Sassone (as discussed in Section 1.2.2). Our results extend this foundation to binding bigraphs, and thus yields an obvious line of future work, namely to extend BiLog to binding bigraphs.

Finally, another line of related work is based on the resemblance between the algebraic presentation of bigraphs and term graph rewriting systems. The algebraic presentation of pure and binding bigraphical structure resembles the algebraic presentation of term graphs by Corradini and Gadduci in [CG99a]. (For more on relations to term graphs, see our comments on related work in the following section.)

2.1.2 An Inductive Characterization of Matching in Binding Bigraphs

In Chapter 4, we tackle the central problem of matching the left-hand side of a reaction rule, the redex, to an agent. Our approach is to initially *characterize* solutions to the matching problem, and then later (as we shall discuss in the following section) to investigate how to specialize this into an algorithmic approach.

Our Approach

As discussed above, an algebraic representation is convenient for reasoning about equality. It allows us to build an understanding of equality by treating in separation the ways in which elementary bigraphs and combinators may work together to form equal bigraphs.

In the same manner, we look to utilize the term language to separate concerns in handling the matching problem. Recall that (omitting some details) we need to solve the following problem: Given a reaction rule $R \to R'$ and an agent a, if $a = C\,(R \otimes \mathsf{id}_Z)\,d$, we say that the rule $R \to R'$ *matches* a. In other words, when

a can be decomposed into a context C, the redex R, and a parameter d, we have a match. In that case, we can perform a reaction $a \to C\,(R' \otimes \mathsf{id}_Z)\,d$.

Because of the richness of the bigraphical model, the decomposition of a is nontrivial—even more so, for binding bigraph reactive systems, where links may be either local (i.e., linked to a located name or to a binding port) or global. For instance, links that are global in the agent may be matched by local links in the redex since the context C may concretize them (i.e., make them global); and, parameters may contain nontrivial local links that may also, through concretization, end up matching global links in the agent. As a small example of the interplay between local and global links, consider a rule with left-hand side: $\mathsf{K}(\mathsf{id}_1) \mid \mathsf{id}_{(x)}$.[4] We may read this rule as matching a node of control K, possibly with something inside the site id_1, beside a local link named x, located at a site. However, even though the link and the node are next to each other in the redex, in the agent they need not be related; in fact, in the agent the link need not be matched to a local link—or be present at all.[5] Observe also, that links that are open in the redex may be matched to closed links in the agent (but not the other way around). Wide rules—rules, where R contains several regions—also contribute to the complexity; as does the fact that links are hyper-edges with unordered points, and that place graphs have unordered children. For instance, we allow left-hand sides such as $\mathsf{K}(\mathit{merge}_2)$, a K node with two sibling sites inside. That redex, when matched to an agent, nondeterministically splits any content of a K node into 2 parts.

To tackle the decomposition of the agent, we exploit that we already have an inductive way of building binding bigraphs as terms. Our approach in Chapter 4 is to look for an inductive characterization of how to construct a context C and a parameter d, by induction on the structure of the agent a and the redex R, the input to the matching problem. We define the concept of a *matching sentence*—a new representation of a match. A matching sentence defines a relation on components of the agent, redex, context, and parameter. Matching sentences are defined such that valid sentences correspond to valid matches, and we look to characterize this relation inductively.

Our main result is a set of rules that suffice for deriving all valid sentences. The main theorems state soundness and completeness of the rules.

The term language characterization of bigraphs helps us give a concise set of rules. As for the equational theory it helps us separate concerns. For instance, the special concerns in matching local and global linkage is collected in one rule (the rule LSUB), while the matching of closed links is handled in another rule (the rule CLOSE). Matching sentences also utilize the insight gained from the treatment of discrete decomposition in the normal forms, by separating global links from place graphs and local links.

[4] Formally, in the following, assume that K has *active* control, i.e., may contain other nodes and allows reaction to occur inside it.

[5] Why? Because the context may either concretize the link to match it to a global link in the agent; or, because the context may concretize *and* close the link while, at the same time, the parameter introduces the located name x via an idle (i.e., not connected) local name—in effect, making the link an idle edge that is discarded via elision.

Discussion and Related Work

To start with, our results yield an inductive method for reasoning about matching in binding bigraphs. Moreover, in combination with our earlier work, we have also established a foundation for basing a theoretically well-founded *implementation* of bigraphs and BRSs directly on terms. This allows us to base such an implementation on a provably complete representation and on provably correct algorithms. We shall discuss this work in Section 2.1.3.

Incidentally, the proof of completeness of the characterization serves as a thorough example of the utilization of the decompositional analysis provided by the normal form and the proof-techniques for reasoning about equivalence (i.e., as developed in Chapter 3).

In the section on related work in Chapter 4, we comment on the relation of our work to general graph pattern matching algorithms. Below, we outline the relations to the long-running traditions of term rewriting and graph transformation systems (see [Dam06], for a more comprehensive survey).

Bigraphical reactive systems are related to *term rewriting* systems [BN98, TeR03]. A term rewriting system (TRS) is a collection of rewrite rules on terms. Terms are constructed inductively over sets of function symbols and variables. Rewrite rules are pairs of terms $L \to R$, such that all variables in R also occur in L. A rule $L \to R$ matches a term t iff for a substitution σ on the variables of L and a subterm t' of t, we have $\sigma(L) = t'$. We rewrite (or reduce) a redex by replacing $t' = \sigma(L)$ with $\sigma(R)$.

Though TRSs lack any treatment of names, they bear a close likeness to reactive systems over place graphs. There are a few important differences, though. Recall that we consider place graphs equal up to permutation of siblings. Terms of TRSs are naturally considered equal iff they are syntactically equal. In matching terms quotiented by an underlying equational theory place graphs are more reminiscent of *rewriting logic*, that is, term rewriting modulo a set of static equivalences. A survey by Meseguer [Mes96] outlined the use of rewriting logic as semantic models for concurrent systems. In bigraphs we are also allowed to write *wide* rules working on place graphs. This allows us to model elegantly distributed reaction, including non-local mobility and copying, as we do not need to show explicitly the parent structure of the subtrees matched by the patterns in the redex. Finally, a special aspect of bigraph sites (that play the role of variables in rules) is that we may match a set of subtrees to a single site in the redex. Further studies are needed to determine whether and how, we may employ some of the sophisticated techniques for matching and rewriting developed in the context of rewriting logic, for instance, for the system Maude [Mau, CDE+01, CDE+03].

The long tradition of algebraic graph transformation systems (GTSs) based on double-pushout (DPO) constructions [Roz97, REKE99, REKM99] (for a recent survey, see [EEPT06]), initiated by Ehrig, Pfender and Schneider [EPS73], are also related to bigraphical reactive systems. The class of all graphs as objects and of (total) graph homomorphisms form a category **Graphs**. A graph transformation rule

$p = (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ with lefthand-side L and right-hand side R, consists of graphs L, K, R and mono (i.e., injective) morphisms l, r. We can think of the *interface* K as the intersection of L and R. A (direct) graph transformation $G \stackrel{p,m}{\Longrightarrow} H$ goes from a graph G to a graph H by p and a *match* morphism $m : L \to G$. We may think of the match morphism m as identifying nodes and edges in L with nodes and edges in G. Given m, we transform G by p, essentially by removing from G those nodes and edges not present in K, and to construct H, we add the nodes and edges in $R \setminus K$. Formally, the matching and the construction of H are defined by two pushout diagrams essentially built from the morphism $k = m \circ l : K \to G$ (hence the name double-pushout).

Notably, standard graph transformation systems do not allow *parametric* reconfiguration rules. When matching a bigraphical redex R to an agent a, $a = C\,(R \otimes \mathsf{id}_Z)\,d$, we naturally rely on the hierarchical structure of the place graph to split the surroundings of R in a context above, C, and a parameter below, d. In contrast, a priori, we have no way to give meaning to *parameters* for arbitrary graphs. Consequentially, we cannot express directly in GTS rules, copying and deletion of arbitrary subgraphs in (basic) GTSs. Some approaches to *hierarchical* graph transformation systems have been investigated. In [DHP02], Drewes et al. investigate a DPO approach that allows recursive nesting of graphs in special hyperedges. The work supports rule schemas with so-called *frame* variables, which can be instantiated to graphs, to allow directly copying and deletion of subgraphs. As opposed to bigraphs, though, edges cannot cross boundaries of frames. Palach [Pal04] proposes a framework that imposes a separate nesting relation directly on components of a (flat) graph (bringing us closer to bigraphs). The framework allows edges crossing nesting boundaries, but the framework does not yet include a supporting notion of parametric rules.

More studies are needed to determine whether the work on, for instance, hierarchical graph transformations may inform the work on bigraphical reactive systems. There are, in fact, studies that show that the links between the theoretical underpinnings for BRSs and graph transformation may be strengthened. As reported by Ehrig in [Ehr02], the underlying categorical model and use of pushouts are different, but may possibly be conjoined.[6]

Finally, the field of term graph rewriting has roots in both term rewriting and graph transformation. Simply put, a term graph rewriting system is a TRS that is lifted and reduced under GTS semantics. When lifting terms to graphs, edges represent the nesting relation. Variables are represented using special nodes, and the structure of graphs is used to represent sharing of variables directly. When applying rules with terms lifted to graphs this naturally results in shared, rather than copied,

[6] As noted in [Dam06], a common ground may be found, perhaps using a 2-categorical approach as used by Gadduci et al. to represent graph rewriting in [GHL99], or, by Sassone and Sobocinski for deriving labels for reactive systems [SS02]. Furthermore, a recent line of work founded by Ehrig and Koenig on deriving labels and capturing observational equivalences in the DPO approach to graph transformation is directly inspired by the work on deriving labelled transitions systems for BRSs [EK04].

subterms, something which cannot be expressed in basic TRSs. The utilization of sharing allows very efficient implementations of functional programming languages such as *Clean* [Cle, PE93].

As noted in Section 2.1.1, the algebraic presentation of the structure of bigraphs resembles the algebraic presentation of term graphs [CG99a] by Corradini and Gadduci. The work in *loc.cit.* is used as the basis of a 2-categorical presentation of term graph rewriting [CG99b]. This resemblance could be used as the foundation for looking for a stronger categorical foundation for the description of matching and rewriting in bigraphs.

2.1.3 An Implementation of Bigraph Matching

The term language for binding bigraphs gives us a complete representation language for expressing bigraphs, and the normal form is a uniform representation based directly on this manner of expressing bigraphs. The inductive characterization of matching, as outlined above, gives us an inductive method for proving a match valid. It can also be used, however, as a foundation for implementing matching in a tool for BRSs.

The paper in Chapter 5 describes our work on deriving a provably sound and complete algorithm for bigraph matching from the characterization. The algorithm is implemented in the BPL Tool, a prototype implementation of bigraphical reactive systems. In the paper, we also describe the tool and illustrate it with an example.

Our Approach

The characterization in Chapter 4 uses the term language and normal forms to separate concerns when we build up valid matches, and for decomposing the involved bigraphs (i.e., agent, context, redex, and parameter) in smaller parts. However, matching sentences still express a relation on expressions up to structural congruence. So, in Chapter 5, to recast the inference rules to work directly on terms, we could simply add a single rule to allow application of structural congruence. However, this yields a wildly nondeterministic inference system as we might need to apply arbitrary structural congruence laws in every step to infer a match. For mechanically finding valid matches, this is hardly practical.

Consequentially, to specialize the characterization into an algorithm for mechanically *finding* matches, we define *normal inferences*. Normal inferences are inferences that are complete in the sense that all valid matching sentences can be inferred, but suitably restricted, such that inferences can be built mechanically. In particular, normal inference definitions for term matching need to spell out *how* and *where* to apply structural congruence. As a main trick, we define normal inferences that require each inference to start by rewriting the input terms (i.e., the redex and the agent) to be on normal form. This allows us to incorporate structural congruence axioms into two rules, the rule for introducing tensor products, and the rule that handles the merging of a product into one bigraph root. In the main theorem in Chapter 5, we prove a grammar for normal inferences sound and complete.

It is feasible to build normal inferences mechanically, and an algorithm to derive such inferences formes the core of the BPL Tool. The tool is a reference implementation of binding bigraph matching and rewriting, and gives us a toolbox for experimenting with bigraphs. The implementation is written in Standard ML, consists of a parser, normalisation and matching kernel, and includes web and command line user interfaces (see [BPL07]). To ensure correctness, the implementation is faithful to the theory, implementing directly the inference systems developed for matching, normalisation and auxiliary operations; essentially implementing one SML function for each inference rule.

Discussion and Related Work

Through our work, we have built an implementation that is theoretically well-founded and whose core algorithm for matching is provably correct. The tool has been used for experimenting with semantics for several bigraphically derived languages; including the $\mathcal{C}$-calculus, developed in Chapter 7, and in the CosmoBiz-project [BGH$^+$08a, BGH$^+$08b] to experiment with higher-order variants of WS-BPEL [TC07] formalized in binding bigraphs. Alas, the implementation is not very efficient. As is apparent from the discussions in the sections above, we have focused on allowing the full generality of reactive systems over binding bigraphs; our first concern has been on soundness and completeness of the algorithm, not efficiency-issues. We comment on several possibilities for future lines of work in Section 2.4.

As discussed under related work in Section 2.1.2, bigraphical reactive systems bear a resemblance to term rewriting systems and graph transformation systems, and some of the techniques employed for the implementation of these might inspire future work on a dedicated tool for BRSs. Due to the longer history of the fields, implementations such as the previously mentioned Maude-system [Mau, CDE$^+$01, CDE$^+$03] (based on rewriting logic), the Clean-language [Cle, PE93] (based on term graph rewriting), and graph transformation tools, such as the AGG (Attributed Graph Grammar) System [AGG, ERT99] and the PROGRES (PROgrammed Graph Rewriting Systems) [PRO, SWZ99], build on considerable amounts of work. In these fields, the development of tools tend to have been driven by varius applications. As needs have arisen in applications, so tools and techniques have evolved. For instance, applications with associated implementations include semantics for functional languages such as Clean [NSvP91], verification of object-oriented systems using graph-transformation systems such as GROOVE [Ren04], generation of diagram editors from graph grammar specifications of visual languages [Min03], and, as suggested in [VSV05], the theoretical foundation of model transformation in model driven development.

This points to a strong argument for also letting the needs of a particular application drive and inform further development on tools for bigraphs. As discussed in the following sections, in Chapters 6 and 7, we launch the idea that bigraphical reactive systems may be used for modelling biochemical reactions at the level of cells. We discuss and motivate a number of significant simplifications and restrictions on

the kinds of bigraphs and reaction rules that we need consider for such systems.

2.2 Developing a Bigraphical Language

In this section, we give an overview of our work on developing a concrete bigraphical language. This effort constitutes the second step in our test of bigraphs as a foundation for the development of languages.

Previous efforts have shown that the bigraphical framework allows the recapture of the semantics of a wide variety of calculi.[7] This is clearly a strength, and it also helps us in developing novel calculi, as it allows us to experiment with a wide variety of different semantics. As we have explained, in our earlier efforts on the BPL Tool we have meticulously made sure to include all the degrees of freedom of BRSs over binding bigraphs. This allows us to give tool-aid for that kind of experimentation using all the expressivity of BRSs.

As we have discussed in Section 2.1.2, the downside of full generality is that it has repercussions for matching. The bigraphical framework allows rules, which may seem too exotic. However, at the level of semantic framework our approach has been to refrain from disallowing any degree of freedom. The encodings of various calculi and experimentation with different bigraphical models have shown that sometimes we might be able to prove certain invariants for our model that make seemingly exotic rules useful. Or, we might simply want to start by giving an abstract semantics that allows "too many" reactions; and then later on refine that semantics. Had we made restrictions on the bigraphical framework before working on the BPL Tool, we could have risked hindering such experimentation for some languages.

However, in the process of developing a *specific* language, our experience has been that as part of the process of finding the right abstractions for a language, it is a natural step to try to single out a subset of BRSs to work with. For instance, for modelling low-level biochemical reactions, it turns out that linear rules are sufficient. Cutting away selected parts of bigraphical freedom benefits the presentation of the developed calculus, and may form the foundation for more efficient implementation later on.

In the following sections we shall discuss our investigation of the usage of bigraphs and BRSs for modelling biological interaction. In Section 2.2.1, we discuss our work on settling on a family of bigraphs and BRSs that we can base our development of a concrete language on. This work is presented in the paper in Chapter 6. In Section 2.2.2, we summarize our development of a concrete language, the $\mathcal{C}$-calculus, that allows a user to experiment with reactions inside and among biological cells. This work is presented in the paper in Chapter 7.

[7] In the section on related work below, we discuss some notable examples where the behavioral theories for existing calculi have been recaptured.

2.2.1 A Generic Language for Biological Systems based on Bigraphs

Since Regev, Shapiro, and Silverman suggested using the π-calculus for modelling biochemical processes [RSS01], there have been numerous suggestions for using process calculi for modelling biological systems at the molecular level. In the pre-study in Chapter 6, we lay the foundation for using bigraphical calculi to experiment with models and languages for biological systems.

Our Approach

First, we need to give a brief outline of the kind of languages that we are aiming for.

We set our aim towards languages that allow *rule-based modelling*, that is, allow a user of the language to write certain kinds of reaction rules themselves. This way of modelling seems particularly well-suited for modelling biochemical reaction patterns. It mimics the understanding and level of description of basic biochemical reactions that biochemists tend to use.

The κ-calculus [DL04] by Danos and Laneve, is a prominent example of a language that has championed rule-based modelling for biological systems. It focuses on 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].

Bigraphs and BRSs seem well-suited as a foundation for developing a calculus that allows rule-based modelling. By leaving part of the job of writing reaction rules to a user of the language, we can leverage the meta-modelling capabilities of bigraphs to allow a user to instantiate a domain-specific sub-calculus for the investigation of a particular biological problem. In summary, the $\mathcal{C}$-calculus, which we shall discuss in the following section, is going to be a family of bigraphical languages, where we will allow the user a level of freedom in choosing reaction rules.

Turning to our work in Chapter 6, we pave the way by discussing and identifying a subset of bigraphical calculi suitably restricted for modelling biological interaction. We call the resulting family $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi. (We let the name signify that we instantiate a particular calculus by giving a signature Σ and a set of rules $\mathcal{R}$; the $\mathcal{B}$, of course, stands for bigraphical.) We take the time to develop and treat carefully an alternative and self-contained description of the syntax, and give an operational semantics for $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi void of most bigraphical idiosyncracies. First of all, this little experiment illustrates that such alternative presentations of bigraphical calculi *may*, in fact, be developed. It lets us describe the $\mathcal{C}$-calculus without referring too much to bigraphs. Importantly, the description of $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi may also serve as the basis for dedicated and more efficient implementations. We have taken care to exploit, in the characterization of the operational semantics for $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi, the restrictions of the language and the reaction rules. We sketch below, the restrictions

we have made on $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi.

In short, $\mathcal{B}^{\Sigma,\mathcal{R}}$ has a lightly sugared syntax for *processes* that correspond to bigraphs with one root and no inner names or idle outer names (in short, link-epi primes); so-called process *groups* are essentially products of processes. In the syntax for processes, we choose a standard set of operators—prominently including parallel product, name hiding, and prefixing—and derive standard definitions for free and bound names, structural congruence and substitution (defined essentially on top of bigraphical composition). As a main simplification, we do not include processes that correspond to pure link graphs; they would essentially be representatives of explicit substitutions (on names). In other words, there is no correspondent of, say, the bigraph y/X among $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes. Observe that in bigraphs, we essentially have substitution residing *inside* the language—bigraphical composition is an operator. In the syntax for $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi we define substitution as an *operation* instead of having it be an operator. These are some of the central simplifications that we make for $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi, whose repercussions, to the syntax, the structural congruence relation (in particular), and the operational semantics, helps us bring the presentation more in line with standard tradition for process calculi.

For reaction rules, we require all rules to be linear and prime (i.e., to be only over processes, not process groups). We give a small set of rules for contextualizing reactions. We simply close reactions under syntactic constructions, structural congruence, and also under *bijective* (not general) renaming of free names—consequentially, we call the contextualization *non-aliasing*. The non-aliasing contextualization is motivated by the inclusion of side-conditions, which we shall discuss now.

It turns out, that the dynamics of a few special rules for diffusion in the $\mathcal{C}$-calculus are more concisely expressed with the usage of a modicum of contextual *side-conditions* on reaction rules. Such side-conditions are well-known from other meta-language frameworks (such as in graph transformation systems [EPS73, Roz97, REKE99, REKM99]), but have not been investigated for bigraphs before. Hence, in Chapter 6, we give a first treatment of reaction rules that allow testing of arbitrary side-conditions on parameters. For defining such side-conditions sensibly on links from parameters, it turns out that reaction under non-aliasing contexts is important.

In a (pure) bigraph reaction, parameters are required to be discrete (i.e., with no shared names), leaving to the context to perform all aliasing. (For process calculi, this corresponds to pushing all ν-binders to the top of an expression, before reaction.) As we explain in the paper, the choice to make bigraphical matching work in that manner resolves an ambiguity in the duplication of links in parameters for non-linear rules.

For our purposes, that means, that if we wish to define side-conditions that test sharing among links stemming from parameters, we would need to test the context also. However, as explained above, we are only concerned with *linear* rules. As a consequence, the ambiguity does not arise in our setting. Hence, we choose to give an alternative definition of matching for bigraphs, that requires aliasing to occur inside parameters, and disallows aliasing in contexts. That suffices to allow us to define also side-conditions on parameters for linear bigraphical reaction.

The main result of the paper is to 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. We finish off by illustrating that with non-aliasing semantics the (nondeterministic) κ-calculus may be faithfully captured as a $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculus.

Discussion and Related Work

Before turning to related work, let us discuss two specific choices in our development of the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework.

First, for the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework and for the $\mathcal{C}$-calculus, we decided not to use any of the variants of binding for bigraphs. We experimented with the use of binding for capturing some of the invariants of the model for biological phenomena in the $\mathcal{C}$-calculus—as captured in well-formedness conditions (see the following section). However, the fact that binding has been developed for capturing lexical scoping of names, ends up working against that use. The scoping constraints for binding are asymmetric, in the sense that bound links are barred from extending *outside* the location of their binder; there is no restriction to downwards extension of links, however. For modelling biological phenomena, we really need an abstraction that captures *distance*, a symmetric concept. Links represent chemical bonds governed by a variety of fairly weak and highly distance-limited forces. Consequentially, links in the $\mathcal{C}$-calculus are limited to connecting highly co-located entities. Though to a limited degree we have been able to construct encodings that enable binding to capture some of the necessary constraints, binding is really not the right abstraction for the job. Thus, we elected to work in pure bigraphs, and formulate the well-formedness constraints as a series of conditions directly on the (bi)graphs—essentially, a sorting. This, in turn, has the side-effect that in verifying the preservation of those well-formedness constraints, we use bigraphical reasoning. This is both a strength of the dual bigraph and term nature of $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes; but also underlines that the $\mathcal{B}^{\Sigma,\mathcal{R}}$ presentation cannot currently stand by itself.

Secondly, in retrospect, the capture of numbered sites and associated composition ends up being somewhat cumbersome. Though, as explained above, we demoted composition to be an operation—in the form of substitution—we decided to keep sites in the language, in the form of numbered variables. To match the behavior of sites in bigraphs, this requires us to consider processes with variables up to order-preserving renumbering of the variables, and include this in the structural congruence relation. This is essentially to mimic that taking the tensor product of two bigraphs A and B invokes a similar reordering on the numbers of sites in $A \otimes B$. As a consequence, for defining conveniently substitution (i.e., the correspondent of bigraphical composition), we also need to stratify the numbering of the variables, that is, to renumber variables to $0, 1, 2, \ldots$. Formally, we can capture this by taking the least order-preserving renumbering.

The $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework was an experiment to try to give an alternative presentation of bigraphs and BRSs, which exploits a particular subset of bigraphs and reaction

rules. In evaluating that experiment, we feel that the capture of variables and substitution is perhaps the least satisfying. It does not match the typical usage of term-valued variables in calculi such as the applied π-calculus [AF01], and still smacks more of an abstraction for contexts with numbered holes (which sites were developed to capture) than it does of variables for carrying parameters in reaction rules. In future work on the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework, and on presenting the syntax and semantics of bigraphical calculi, we need to try to find more elegant ways of treating variables for parametric rules.

Below, we expand the discussion on related work in the paper. In order for the discussion to be coherent and self-contained, we shall need to repeat a few paragraphs from the paper.

Much effort has been made to recapture existing calculi as BRSs, in particular, on recapturing contextual equivalences using the machinery for bigraphically derived bisimulations via the derivation of minimal labels [Mil06b]. (More recently, Bonchi et al. have suggested an alternative approach, called a *saturated* semantics [BKM06], and motivated that in some cases minimal labels are not sufficient.)

Milner and Jensen recaptured the behavioral equivalences for a version of the asynchronous π-calculus [JM04]. In the monograph by Jensen [Jen06], Jensen extends the results to the full π-calculus with summation and replication; and also treats a bigraphical encoding of the ambient calculus [CG98, CG00]. Bundgaard and Sassone treat the polyadic π-calculus with capability types (as introduced by Pierce and Sangiorgi [PS96]) in bigraphs [BS06]. All these encodings make use of bigraphs with binders and use a variety of bigraphical sortings (as discussed in Section 1.2.2) to recapture the original contextual equivalences given for these calculi.

In [Mil06b], Milner encodes a finite variant of CCS (i.e., without replication) in pure bigraphs. It turns out, that the derived bisimilarity for the encoding does not match the original bisimilarity given for CCS [Mil80]. To recapture the original notion of bisimilarity given for CCS, Milner restricts to labels corresponding to contexts without substitutions (i.e., link-mono contexts). Correspondingly, the original bisimilarity given for CCS is not preserved under substitution. Interestingly, those contexts match our non-aliasing contexts.

In our work, we have focused on reaction and dynamic correspondences, not bisimulation. We wished 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 for $\mathcal{B}^{\Sigma,\mathcal{R}}$, also. In doing so, we may try to use the bigraphical framework for deriving labels and bisimulation. However, as we have modified the definition of the reaction relation to incorporate (negative) side-conditions and restricted the contextualization of ground reaction to link-mono contexts, it needs to be studied how to update the framework accordingly. In doing so, one may look to the experiences, by Rangel et al. [RKE08], on deriving labels for graph transformation systems that have rules with negative application conditions (using the similar, so-called *borrowed context* approach [EK06]). Such studies may also be a first step towards relating the bigraphical theory for bisimulation congruences to the studies on meta-theoretical theorems concerned with establish-

ing congruential behavioral equivalences for syntactic rule formats for structural operational semantics [MRG07].

In [Mil06a], Milner gives an encoding of an (untyped) λ-calculus with explicit substitutions, Λ_{sub} and starts an investigation of confluence properties of BRSs. Milner uses the identities for nodes and edges in concrete bigraphs to analyze if, and how, two ground redexes occurring in a ground bigraph overlap. Ó Conchúir continues the examination of Milners λ-calculus [Con06], and of confluence, and compares it to another λ-calculus with explicit substitutions λxgc by Bloo and Rose [BR95].

Bundgaard and Hildebrandt builds upon the treatment of explicit substitutions (by Milner in [Mil06a]) and nested named locations (by Jensen in [Jen06]), to give a bigraphical semantics for the calculus Higher-Order Mobile Embedded Resources (Homer) in [BH06], a higher-order calculus (i.e., allowing process-passing) with nested locations that allows active mobility and duplication of processes, and has local names.

Finally, in [GM08a], Grohmann and Miculan use and motivate their directed variant of bigraphs by giving an encoding of the Fusion calculus.

There have been a few lines of work that focus on the development of novel models or languages based on bigraphs.

An evaluation of the aim of using bigraphs for representing context-aware systems in the domain of mobile ubiquitous systems was initiated in [BDE+06] and continued in [Els06] by Elsborg, who defines and models so-called plato-graphical models as BRSs. Along the way the authors encode and analyze a MiniML-like calculus with references. 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 brief, the project aims to extend the current state of the art for business process languages to handle also mobile and distributed workflows.

The project stems from work in the Ph.D. thesis of Bundgaard [Bun07] on models, reasoning techniques, and, types for higher-order mobile embedded resources, and from work by Hildebrandt et al. [HNO06], on giving an encoding of a subset of the business process execution language (BPEL) in the bigraphically based framework Reactive XML [HNOW06]. 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.

2.2.2 A Language for the Cell

In Chapter 7, we conclude our investigation of bigraphical languages by presenting a novel language, the $\mathcal{C}$-calculus. The $\mathcal{C}$-calculus allows a user to work with models that capture low-level interaction inside and among cells, the basic building blocks

of all life. In the $\mathcal{C}$-calculus, we focus on two main actors of cells, membranes and proteins.

The introduction in the paper includes a brief overview of the biology governing cells, proteins, and membranes, providing a background for the motivation for our choices in the calculus.

Our Approach

Early in the development of the $\mathcal{C}$-calculus, we phrased a slogan for the kind of modelling that we wanted to allow: "*Modelling by rule-refinement*". We wanted to allow users of the $\mathcal{C}$-calculus considerable freedom in choosing the reaction rules to govern reconfiguration. As discussed in the previous section, the inspiration was the succes of the κ-calculus in treating a wide variety of protein-protein reaction patterns. However, we did not want to lay upon the shoulders of modellers the task of inventing their own abstractions of basic biological events. This is hardly any better than just presenting users, say, biochemists, with the entire framework of bigraphs. Instead, we set an aim to characterize a fixed set of canonical biological *actions*, and encapsulate each action in a reaction rule. The task of the user is then to choose actions and decorate them with application conditions; this corresponds to instantiating a reactive system by refining the core reaction rules.

Having set a goal for the calculus, we start by settling on a model for proteins and membranes.

We aim to capture protein-protein interaction on a level comparable to that of the κ-calculus, but extended to a multi-compartment environment. To allow for transmembrane-proteins, in the $\mathcal{C}$-calculus proteins are represented as clusters of so-called protein *domains* sharing a common name.[8] Protein domains are, bigraphically speaking, atomic nodes. This name is a formal representative of the protein backbone. Domain-domain bonds, that let proteins form complexes, are also represented via name-sharing.

Compartments are formed by formal *membranes*. Bigraphically, membranes are modelled by active nodes, that is, nodes that may contain other nodes and, in which reaction may occur. To capture faithfully the dynamics of fusing or dividing membranes, we include in the $\mathcal{C}$-calculus also binary *channels* between membranes. Channels allows us to capture an observable intermediate state in fusion and division. In this intermediate state the connected compartments may exchange material regulated by diffusion. Formally, channels are formed by *gates* sharing a common name. (Gates are another kind of atomic nodes.) Channels can be seen as abstractions for the various molecular-structures that bind together (real) fusing or dividing membranes.

We instantiate the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework for the $\mathcal{C}$-calculus, and call processes in the $\mathcal{C}$-calculus *solutions*. We want to emphasize for users of the $\mathcal{C}$-calculus that solu-

[8] In short, domains are biological abstractions of the active regions of proteins, i.e., the parts of the protein that binds together with other proteins to form complexes. See the Introduction in Chapter 7 for more details.

tions have an equally formal interpretation as (bi)graphs, so we include a section that outlines this relation. Importantly, the grammar for the $\mathcal{C}$-calculus (i.e., that inherited from the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework) still allows us to write expressions for many solutions, which may be hard to interpret biologically. So we define and motivate at length a set of *well-formedness* conditions to single out those solutions, which we can interpret biologically.

Having settled on a static model, our next task is to characterize a set of core rules (called *generators* in the paper) that each encapsulate a single biological action. These actions include protein-level reconfiguration, such as forming or breaking complexation-links, or changing the conformation of a protein; or, they correspond to a variety of biologically based membrane-level reconfigurations.

We need to take special care with those core rules which involve bulk transport of sub-solutions; such transport is modelled by relocating parameters in parametric rules. The central rule that allows transport encapsulates *diffusion* of complexes, that is, connected components of proteins. The underlying molecular forces that govern protein backbones and protein complexation are severely restricted by locality; we capture this in the well-formedness conditions. To ensure that we preserve well-formedness we develop a contextual side-condition that checks the parameters. We formulate the side-condition using a small function that essentially computes the connected component of a user-specified set of proteins T. However, as protein-backbones may cross membranes, we need to find *only* the part of connected component that is co-located with T. Furthermore, as we do not want to allow *membranes* to diffuse, in some cases we want to disallow diffusion, namely when the connected component needs to include a membrane to hold all connected proteins. The side-condition also needs to check for this. In all, the treatment of diffusion requires some care to settle on a model that is both logically sound and biologically valid; consequentially, we spend quite some space on motivating and developing the core rule that encapsulates it.

Finally, to allow a user to give application conditions for core rules, we define *refinement*. We choose to treat and define refinement for general $\mathcal{B}^{\Sigma,\mathcal{R}}$-rules (to allow future development of the $\mathcal{C}$-calculus). Bigraphically speaking, we essentially allow rules $L \to R$ to be refined to $C\,L\,D \to C\,R\,D$; that is, we allow a user to add extra context to rules. That corresponds exactly to giving extra conditions on the reaction encapsulated in $L \to R$, in the sense that we simply restrict the applicability of the rule, without changing the semantics.

The main theorem of the paper states that any refinement of a core rule preserves well-formedness.

In the paper we also illustrate the use of the $\mathcal{C}$-calculus with two examples. We develop a model that illustrates simple cross-membrane signalling via a G-protein coupled receptor protein, and we develop a larger model of the formation of clathrin-coated cytoplasmic vesicles, through budding from the plasma membrane (i.e., the cell-wall).

In conclusion, to develop a model in the $\mathcal{C}$-calculus the main task of the user is to pick and choose biological actions, encapsulated in a toolbox of core reaction rules,

and then refine them by giving contextual application conditions for rules. The main theorem ensures us that all reactions will stay within a fragment of solutions that we can interpret biologically. In essence, we feel that we have fulfilled the goal set out by our slogan, to allow "*modelling by rule-refinement*".

Discussion and Related Work

The extension of a κ-like calculus with dynamic compartments that support both fusion and fission is a novel contribution, in itself. The $\mathcal{C}$-calculus is, to our knowledge, the first calculus to capture an observable intermediate state for fusing and dividing compartments. The channel abstraction and its behavior is modelled over research that shows that both when cells fuse or divide, membranes will temporarily be in a partially fused state with a neck consisting of partially fused membrane material. Below, we comment on some related efforts on using formal calculi to model biology. First, however, let us comment on the $\mathcal{C}$-calculus itself.

Overall, our focus in developing the $\mathcal{C}$-calculus has been to make choices to simplify practical modelling of biological phenomena on the level of proteins and membranes. Ideally, with the help of a little syntactic and graphical sugar, we expect biochemists to be able to specify and work with $\mathcal{C}$-calculus models themselves. The description of the transportation rules (the rules for diffusion and pinch[9]) required the most effort, to give a biologically faithful model that allows the specification of convenient rules. As we discuss in the paper, we could have elided the use of any side-conditions by taking only rules that allow the transport of *statically* specified closed protein complexes. That would have been computationally complete, in the sense that it allows transport of arbitrary complexes.

However, it would also be quite impractical to use, as in many cases a central regulation mechanism of diffusion is based on certain proteins or complexes that act as *chaperones*. Chaperones essentially allow anything that they bind to to diffuse; and, there is, in general, a huge number of possible complexes that such chaperones may bind to. In such situations it would be unfeasible to require a modeller to enumerate and specify fully all the possible complexes that a chaperone was known to bind to. So, to allow the specification of more practical rules, we set the goal to allow rules that say something along the lines of: "any complex(es) C co-located with, and bound to a chaperone given by some T may diffuse"; and we only want to require the user to specify the chaperone in T. That goal drove us through a series of models, which ended up in the core rule for diffusion having a special side-condition that essentially collects the complex(es) C. The side-condition required some formal footwork, to make it behave correctly. Though this complicates the presentation of the final versions of the transport rules somewhat, on balance, we feel that the convenience and level of abstraction allowed by this treatment justifies the added complexity. In any case, in an implementation of the $\mathcal{C}$-calculus, the mechanics of computing the side-condition would be an implementation-detail that

[9]The *pinch* rule is the other rule in the $\mathcal{C}$-calculus (apart from the diffusion-rule) that needs care, as it encapsulates the simultaneous creation of a new membrane as well as transport.

a user need not worry about.

We can also remark that, though in the paper we focus only on transport of protein complexes, we have also considered versions of the calculus, where we generalize the rule for pinching a membrane to allow also transfer of membranes. This would allow users to conveniently model, in a single rule and thus at a relatively high level of abstraction, cellular division of eukaryotic cells—cells that have nuclei and membrane-encapsulated organelles. We can model that, by allowing the core rule encapsulating pinching to transport also membranes. In producing such extensions of the calculus, we can benefit from having isolated the computation of the complex in a side-condition: We need only reimplement the side-condition to collect also membranes to capture cellular division.

Finally, tying in to our earlier work, we should also remark that the BPL Tool was useful for experimenting with different semantics for the $\mathcal{C}$-calculus. We expect future work on developing a dedicated implementation of the $\mathcal{C}$-calculus, to be able to exploit, in particular, the well-formedness conditions. In well-formed solutions, we prune further the bigraphs considered in the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework, and remove many of the complexities that haunt general bigraph matching. Knowing that well-formedness is preserved across reaction allows an implementation to optimize for this, for instance, to utilize that all links except protein backbones are binary, and that backbone links (the only hyper-links) are restricted to one or two specific shapes.

We now turn to related work; we expand the discussion on related work in the paper. In order for the discussion below to be coherent and self-contained, we repeat some paragraphs from the paper.

Several languages have been proposed in order to model biomolecular systems. We have already discussed how the protein-model of the $\mathcal{C}$-calculus is inspired by the κ-calculus. The Bio-κ-calculus by Laneve and Tarissan extends the κ-calculus with basic brane-inspired membranes [LT06]. However, splitting of membranes is a non-atomic procedure, which requires a considerable amount of (computationally expensive) encoding. In comparison, in the $\mathcal{C}$-calculus we have taken the atomic *pinch* primitive for membrane division, and isolated any associated splitting of proteins in a side-condition.

The idea to use formal calculi for mobile and distributed systems was launched by Regev, Shapiro, and Silverman. They use the π-calculus to represent and simulate biomolecular processes underlying protein signalling networks [RSS01]. They continue that work in the stochastic version of the π-calculus [PRSS01] (as treated by Priami in [Pri95]) taking into account both the time and probability of biochemical reactions. Proteins are encoded as parallel products of a series of domains, encoded via channels. As in the $\mathcal{C}$-calculus, name-sharing encodes both protein-backbones and complexation. As is standard in the π-calculus, locality (i.e., membrane-encapsulated compartments) is encoded via the sharing of names, and mobility is encoded via name-passing. Conformational change is also encoded via mobility. The basic combinators of π-calculus suffice to encode sequential events, independent events (via parallel), or mutually exclusive events (via sum). The line

of work on using languages inspired by the stochastic π-calculus is continued in the school of work on β-binders [PQ05] lead by Priami and co-authors. In particular, β-binders add syntactic constructs for modelling membranes, but do not offer any solution to the problem of fission.

In all, the π-calculus family suffice to model protein signalling at a relatively high level of abstraction. However, the relative simplicity of the basic π-calculus requires the modeller to invent and maintain her own abstractions of biological entities and events. The single reaction rule of the π-calculus also requires us to encode all biological reactions via name-passing. Even further, since our model is entirely contained in a π-calculus process, this requires us to make explicit in the process the pre-conditions for all events. In essence, we need to mix the specification of the static model with our encoding of the dynamics for a model. This has the benefit of giving us a model that may be easier to analyze, since we have elaborated the possible events in the term for the process. On the other hand, it may also make it harder to discover new and unexpected behavior. In the $\mathcal{C}$-calculus, we are inspired by the κ-calculus by allowing the user to disassociate the static model (the solutions) from the dynamics (the reaction rules). We view this as one of the key advantages of *rule-based modelling*. We should also remark, however, that by using the π-calculus one benefits also from the multitude of tools developed for variants of the π-calculus.

In Bio-ambients [RPS$^+$04], Regev et al. extend their earlier work by developing a modified version of the ambient calculus developed by Cardelli and Gordon [CG98, CG00]. The ambient calculus is extended to include parent-to-child communication as well as intra-ambient communication. The calculus is also equipped with a variety of dual capabilities to allow actions such as *entry*, *exit*, and *merging* of ambients. Named ambients model membrane-enclosed compartments; the ambient abstraction is also used to model proteins, allowing *merge* to model certain kinds of complex-formation and membrane fusion. The calculus encodes cross-membrane proteins by parent-child communication—by allowing proteins to communicate outside the membrane they are enclosed in. This has the side-effect of requiring the user to *not* model cross-membrane proteins via ambients, but instead as naked processes. Also, protein complex breakage is complicated by using *merge* to model complex-formation. To allow complex-breakage, the authors suggest to use instead ambient entry to model complex-formation (introducing an asymmetry in the relationship behind proteins in a complex); or, to still use *merge*, but use private channels to keep information on original protein-structure. The latter solution, in turn, results in complex breakage being a multi-step procedure that essentially "gathers" the proteins before disassociating them. (An analogous procedure would be needed to model membrane division.) In its model of biological entities, the Bio-ambients calculus is a step closer to the $\mathcal{C}$-calculus. The calculus has a separate syntactic abstractions for compartments; this is then used to model both membranes, proteins, and protein complexes. As for the π-calculus, the reaction rules for the calculus is hardwired.

Cardelli continues the investigation of membrane interactions in the family of

brane calculi [Car04a]. Here, the focus is aimed mainly at finding good abstractions for membranes and their behavior. Molecular structures are limited to atomic structures attached to membrane-surfaces. Cardelli identifies a set of basic membrane-actions that is motivated by the hydrophilic and hydrophobic properties of membranes, and coin the concept "bitonality" to describe the constraints. In capturing a small set of membrane-actions in the $\mathcal{C}$-calculus, we are inspired by the brane calculi, and, in particular, by the *projective* variant of the brane calculi, by Danos and Pradellier [DP04]. Projective descriptions (which we explain and define in detail in the paper) let us describe conveniently sets of rules for reactions involving regions separated by membrane-surfaces, while eliding the *orientation* of those separating membranes. As the family of brane calculi contain only simple molecular structures attached to membranes, the reactions to model fusion and division of membranes (as well as transport) become equally simple. The $\mathcal{C}$-calculus differs notably, in our treatment of multi-compartment proteins, and our allowance for an observable intermediate state in fusion and fission. Also, as discussed above, in the $\mathcal{C}$-calculus the static model is dissassociated from the dynamic one.

In the recent paper on Bitonal Membrane Systems [Car08], Cardelli continues the line of work on membrane calculi, and investigates further the notion of projectivity and orientation to unify membrane-reactions that from a local perspective (a so-called local *patch* of the membranes) involve the same pattern of reconfiguration.

Finally, also related is the school of work on P-systems [Pau02] initiated by Paun. P-systems are essentially rewriting systems with locations at a high level abstraction. The aim of P-systems is more angled towards developing a general computational model *inspired* by biology, than towards giving a language for describing and investigating biology, however.

2.3 Conclusion

In this Ph.D. dissertation, we have made a number of contributions that together fulfill the stated goals: (1) to develop a sound formal foundation for a tool for experimenting with bigraphical languages; and, (2) to develop a novel bigraphical language for the domain of cellular biology.

In the first main body of work, we have focused on establishing the theoretical foundation for a tool for experimenting with bigraphs. In the paper presented as Chapter 3, we have given a syntactic foundation for understanding and working with the structure of binding bigraphs. We have developed a term language for binding bigraphs, a set of normal form theorems, and an equational theory that axiomatizes bigraph isomorphism. This work yields powerful syntax-based proof-techniques for reasoning about bigraph equivalence on the term level. It has informed the development of inductive type systems for bigraphs [EHS08a, EHS08b], and has formed the basis for reasoning about matching on the term level.

In Chapter 4, we have presented a set of rules that together inductively characterize matching for the full theory of bigraphical reactive systems over binding bigraphs, that is, for determining when the left-hand side of a bigraphical rule matches an agent. The characterization is precise in the sense that it is proven both sound and complete with regard to the abstract definition of matching. Our work provides both inductive proof-techniques for reasoning about matching, and paves the way for developing and *proving correct* a matching algorithm based on terms for binding bigraphs. The inductive characterization relies intrinsically on the normal form theorems for binding bigraphs developed in Chapter 3 to tackle decomposition of the agent, and to allow us to analyze matching of a general bigraph by considering expressions for its link and place graph constituents.

In Chapter 5, we have described, and proven sound and complete, a matching algorithm for bigraphical reactive systems over binding bigraphs. The algorithm builds directly on the characterization presented in the previous chapter, but is specialized to integrate applications of structural congruence in special rules. We define a grammar for *normal* inferences and prove that normal inferences are complete, in the sense that all valid matches may be derived. A matching algorithm to build normal inferences mechanically forms the core of the BPL Tool, a first implementation of bigraphical reactive systems. In the chapter, we have also described the tool and illustrated it with an example.

In the second line of work, our focus has been to develop a concrete new language based on bigraphs. The work presented in Chapter 6 forms a pre-study that lays the foundation for using bigraphical theory to model biological interaction at the level of proteins and cells. In the study, we treat how to extend linear bigraphical reaction rules to include negative contextual side-conditions. We also develop a self-contained presentation of the operational semantics for a subset of bigraphical calculi inspired by the style of standard structural operational semantics.

Finally, in Chapter 7, we have presented the $\mathcal{C}$-calculus, a novel language for modelling low-level interaction at the level of biological cells. The language focuses

on two main actors of cells, proteins and membranes, and allows users the freedom to instantiate a domain-specific sub-calculus for the study of particular biological applications. The $\mathcal{C}$-calculus also introduces a novel abstraction, *channels*, for modelling partially fused membranes, and uses these to give a novel treatment of transport of complexes through diffusion and parametric pinching of buds.

2.4 Future Work

There are several lines of possible future work, and we have already commented on a number of them above. Some are direct continuations of the work in Chapters 3 through 7, and, some involve the utilization or adaption of results, techniques, or tools from other studies on bigraphs, or is inspired by research in other fields. Below, we collect and outline some prominent directions.

Bigraphs with Binders

As discussed in Section 1.2.2, Milner has suggested a variant of bigraphs with multi-located names [Mil04]. It seems fairly straightforward to extend the axiomatization presented in Chapter 3, and the characteriziation of matching presented in Chapter 4 to this generalization. We conjecture that the complexity lies in introducing binders in the first place.

In all, it seems that the final word on *how* to introduce binding in bigraphs has yet to be said. In an upcoming book on bigraphs [Mil09], Milner discusses a variant where binders have been disassociated entirely from ports on nodes (and where names can still be multi-located); instead, binders take the form of atomic nodes themselves. As discussed in *loc. cit.*, this variant appears to preempt the previous suggestions, and in addition allows binding links to connect to sibling ports. It will be an important topic for future research to give a definitive treatment of binding for bigraphs.

Logics, Sortings, and Types

Both static logics and dynamic logics are of considerable future importance for stating properties and reasoning about bigraphs and BRSs. As discussed in Section 1.2.2, Milners axiomatization of pure bigraphs has been used a foundation for BiLog [CMS05b], a framework for spatial logics that instantiates to bigraphical structure. An obvious line of future work would be to utilize the axiomatization of binding bigraphs structure presented in Chapter 3 to extend the BiLog-framework to the binding setting. It will also be important to work toward extending the framework with more natural dynamic reasoning capabilities, than those discussed in [CMS06]. We should not, perhaps, hope to find easily a general result relating the derived bisimilarity and a dynamic logic for BiLog and BRSs, in the style of the Hennesy-Milner logical characterization of bisimilarity for the π-calculus [HM80]. For that, it seems, that the intensionality of the BiLog framework at present is too pronounced.

More experience is also needed to determine whether and how we might hope to develop tool-support for some of the features presented by a logic. Prominent tools such as the Concurrency Workbench [CWB, CPS93], and the Mobility Workbench [MWB, Vic94] are concerned with efficiently representing CCS and the π-calculus, respectively, and, particularly with analyzing and verifying behavioral

properties. Spatial logics promise to lend us a language for expressing such properties for bigraphs and bigraphical reactive systems.

Relatedly, as discussed in Section 1.2.2, sortings [Deb08] and types [EHS08a, EHS08b] for bigraphs may also be important topics to consider in future work on tool-support. Sortings have been a necessary component in constraining BRSs to include less "junk" when recapturing behavioral theories for various calculi. Both typings and sortings could probably be used to simplify and make more efficient the computation of the matching problem, as we need only consider well-sorted or well-typed matches.

The inclusion of tool-support for either sortings, types, or logics seem to require a considerable amount of work. For sortings, in particular, this would at the least require work on how to express sortings, but also on how to intelligently combine sorting with matching and rewriting. This may be simpler for inductive types [EHS08a, EHS08b] whose structure is related directly to the term language for bigraphs. On the other hand, the theory is currently less developed. For spatial logics, one would most probably need to determine an interesting subset of the logic, which it would be feasible to implement tool-support for checking.

Matching and Implementation

The work presented in Chapters 4 and 5 constitutes a firm formal basis for the BPL Tool [BPL07]. Our choice to base a matching algorithm on normal forms for term has aided us in producing a formally verifiable algorithm. However, while the direct utilization of the results on syntactic theories yields provably correct representations and algorithms, it leaves ample room for efficiency-improvements.

First of all, it is interesting to note that each definition of normal inference corresponds to a different algorithmic approach to matching. It might even be worthwhile extending the set of rules to allow even more freedom in choosing a definition of normal inference. There are several possibilities for adding rules that allow shortcutting obviously invalid or valid matches. At the very least, one might look for a revised normal form, more suited for efficient matching and rewriting. We are also currently considering rephrasing the rules to derive a set of constraints for wirings (the three first components of a matching sentence), which could be fed to a constraint solving algorithm, instead of directly building matches for them, as required by the vanilla rules.

Centrally, there is also a need to investigate techniques for intelligently combining matching and rewriting. For instance, one of the main pillars of efficiency in the tool developed for simulating the κ-calculus is, that matches for all reaction rules are precomputed and easily *updated* (i.e., not recomputed) when reaction occurs [DFFK07]. Such an approach could also be implemented as a heuristic approach, to maintain instead a list of approximations for possible matchings.

Also, as suggested by [DD05a], it might be worthwhile to consider approaches where matching is driven even more aggresively by the place graph. (In *loc.cit.*, we show that for pure bigraphs with epi redexes, given a match in the place graph there

is at most one compatible link graph match.)

In the BPL group, we have also considered preliminary work on a generic abstract machine for bigraphs. In essence, the idea is to combine the insights from the inductive characterization of matching, as presented in Chapter 4, with a simplified representation for bigraphs and their rewriting semantics, resembling the presentation given in Chapter 6. Ultimately, one might hope that given one of the encodings of the π-calculus, such an abstract machine might be instantiable to something resembling Turners abstract machine for the π-calculus [Tur96].

Finally, as discussed under related work in Section 2.1.2, another line of future work is to investigate whether one might employ techniques for matching or rewriting from the fields of term or graph rewriting. Also, on the level of the algebraic presentation, the resemblance between the axiomatization of term graph structure in [CG99a]—which is used as the basis of a 2-categorical presentation of term graph rewriting [CG99b]—leads to a line of work that may yield an alternative categorical foundation for the description of matching and rewriting in bigraphs.

Bigraph-based Languages

In Chapters 6 and 7, we discuss and develop the usage of bigraphs for modelling biochemical reactions. This work has driven the treatment of negative side-conditions for bigraphical reaction rules and the development of an alternative presentation of the rewriting semantics for bigraphs. We strongly believe that future development of the bigraphical framework should be driven and informed by such experimentation with novel languages and models based on bigraphs. Thus, we hope that our work on bigraphs for biology may serve as the basis for further studies, not only on languages for biology, but also on the bigraph framework itself.

One particular line of future work, stemming from the work on negative side-conditions, is to investigate how the derivation of labels and bisimulation behaves under this modification. As discussed in the section on related work in Section 2.2.1, in doing so, one might look to the research by Rangel et al. on deriving labels for graph transformation systems with rules with negative application conditions [RKE08].

A Language for the Cell

In Section 5.1 in Chapter 7, we discuss suggestions for future work on the $\mathcal{C}$-calculus, at length. We summarize some of those suggestions below.

Immediate possibilities include the generalization of the description of pinching to allow us to model cellular division of eukaryotic cells (i.e., cells with nuclei) and further experimentation with larger examples.

Ultimately, to work with such experiments, we aim to build a dedicated implementation for simulation of the $\mathcal{C}$-calculus. The generic BPL Tool allows experimentation with a side-condition-free fragment of the $\mathcal{C}$-calculus. However, a dedicated implementation can be optimized directly for the calculus, utilizing the well-formedness constraints to full effect. As we have essentially developed the calculus as a (considerable) extension of the κ-calculus, we may hope to utilize some

of the techniques developed for the implementation of the κ-calculus [DFFK07] to build an efficient implementation.

Towards that purpose, we have also made a preliminary investigation of causality for pure bigraphs, which we expect to be instantiable for the $\mathcal{C}$-calculus. Essentially, we aim to be able to describe causal relations in terms of modification-testing dependencies, in which we may describe classical notions such as precedence, weak permutation, casuality, and concurrency. Recall, that we discussed above, the implementation of the κ-calculus, whose efficiency stems, in particular, from a mechanism for updating precomputed matches when reaction occurs. That update-mechanism relies essentially on an understanding of the causal relation between reaction rules.

Finally, recent results by Milner, Krivine, and Troina [KMT08] ensures us, that we may extend the $\mathcal{C}$-calculus to a stochastic setting. Several proposals in the field of calculi for biology have illustrated that stochastics is important as it allows for more accurate quantitative biological modelling [DFF$^+$07, DPPQ06, PRSS01].

Chapter 3

Axiomatizing Binding Bigraphs

Abstract

We axiomatize the congruence relation for binding bigraphs and prove that the generated theory is complete. In doing so, we define a normal form for binding bigraphs, and prove that it is unique up to certain isomorphisms.

Our work builds on Milner's axioms for pure bigraphs. We have extended the set of axioms with five new axioms concerned with binding, and we have altered some of Milner's axioms for ions, because ions in binding bigraphs have names on both their inner and outer faces. The resulting theory is a conservative extension of Milner's for pure bigraphs.

Preface This chapter is a reprint of the journal paper [DB06] which, in turn, is a journal version of the Tech Report [DB05]. The paper is presented unchanged, except for fixing two minor errors found after the publication of the paper (cf. footnotes 1 and 2). The paper was co-authored with Lars Birkedal from the IT University of Copenhagen.

3.1 Introduction

Over the last decade, Robin Milner and co-workers have developed a theory of bigraphical reactive systems, see [JM04, Mil05, Mil06b]. Bigraphical reactive systems (BRSs) provide a graphical model of computation in which both locality and connectivity are prominent. In essence, 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. Bigraphs can be reconfigured by means of *reaction rules*. A *bigraphical reactive system* consists of set of bigraphs and a set of reaction rules.BRSs have been developed with two principal aims: (1) to model ubiquitous systems by focusing on mobile connectivity (the link graph) and mobile locality (the place graph), and (2) to provide a unification of existing theories by developing a general theory, in which many existing calculi for concurrency and mobility may be represented, with a uniform behavioural theory. The latter is

achieved by representing the dynamics of bigraphs by reaction rules from which a labelled transition system may be derived in such a way that the associated bisimulation relation is a congruence. The unification has recovered existing behavioural theories for the π-calculus [JM04], the ambient calculus [Jen06], and has contributed to that for Petri nets [LM04]. Thus the evaluation of the second aim has so far been encouraging. In [BDE$^+$05] Birkedal et al. initiate an evaluation of the first aim, in particular it is shown how to give bigraphical models of context-aware systems.

As suggested and argued in [JM04, Bir04, BDE$^+$06, BDE$^+$05] it would be very useful to have an implementation of the dynamics of bigraphical reactive systems to allow experimentation and simulation. In the Bigraphical Programming Languages research project at the IT University, we are working towards such an implementation.

An implementation of bigraphical reactive systems must, of course, work on some data structure representing bigraphs. An obvious possibility is to represent bigraphs by *bigraphical expressions* that denote bigraphs. This is particularly feasible if (1) the bigraphical expressions are defined inductively (by a grammar, say), such that algorithms may operate inductively on the representation; and (2) there are normal forms for bigraphical expressions and axioms for determining whether two bigraphical expressions denote the same bigraph, such that algorithms may operate on normal form representations, and may be founded on principles of equational reasoning. There *is* such an axiomatization of the so-called *pure* bigraphs with these properties [Mil05]. In the present paper we extend the axiomatization for pure bigraphs to *binding bigraphs*, a wider class of bigraphs better suited for the representation of calculi and systems involving binding, e.g., the π-calculus, and prove that our axiomatization has the above mentioned desired properties. In particular, we prove the axiomatization complete and prove that our notion of normal form is unique up to certain specified isomorphisms. Our axiomatization is a conservative extension of Milner's.

For reasons of brevity, we refer the reader to the papers cited above for more background information and motivation than can be included here. In particular, we shall need to assume some familiarity with pure and binding bigraphs as described in [JM04] and with the axiomatization of pure bigraphs [Mil05] — we do, however, include an informal description of bigraphs in the following section and we have included the formal definition of binding bigraphs in 3.A.

The remainder of the paper is organized as follows. In the following section we introduce bigraphs by example. In Section 3.3 we define elementary forms of bigraphs and arrive at a semantic normal form theorem, which expresses how every bigraph may be decomposed into a composite of elementary forms. In Section 3.4 we present our term language for binding bigraphs and the accompanying equational theory. We arrive at a theorem which states soundness and compleness of the equational theory. We present some examples of bigraphs and their corresponding normal forms in Section 3.5 — we recommend that the reader refers to these examples from time to time when reading the earlier more technical sections. We comment on some related and further work in Section 3.6. Finally, 3.A contains a

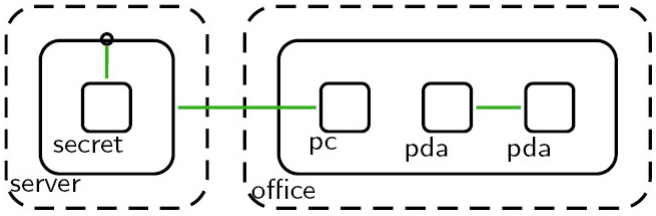


Figure 3.1: A – a bigraph model of an office in a building

summary of the definitions of binding bigraphs. We have omitted detailed proofs from this paper, they can be found in the companion technical report [DB05].

3.2 Bigraphs by Example

We introduce the most basic terminology and properties for bigraphs, by giving a small example of a bigraph. We refer the reader to 3.A for all formal definitions.

The bigraph A is bigraph model of an office containing a pc and two pdas. The pc is linked (supposedly by some kind of network connection) to server containing a secret located somewhere else. We say that A consists of **roots** (dashed boxes), **nodes** (solid boxes), and **links** (lines). Each node has a **control** written beside it. The control indicates the number and type of ports for linkage on the node. Ports can be either **free** or **binding** — the latter indicated by circular attachments.

Bigraphs can contain **sites** (sometimes called holes), and/or inner or outer names. The bigraph B has two sites, numbered 0 and 1, and two inner names, x **located** at site 0 and z **global** (i.e., not located). C has two outer names, x located at its first root, and x global.

We can compose B and C by plugging the sites of B with the roots of C. The bigraphs B and C **compose** to form A. We write $A = B \circ C$. Bigraph A is said to be **ground** as it has no holes or inner names.

Binding bigraphs enforce a **scope** discipline on linkage connected to a binding port: All **peers** (names or ports) linked to a binding port or located outer name, must be nested within the node or root (see Definition 3.A.6).

Not all bigraphs are composable. B and C composes exactly, because C has a root, outer name (local and global), for each corresponding site and inner name (local and global) of B. The **interfaces** of a bigraph registers this, and hence determines which bigraphs can be composed. We write $B : \langle 2, (\{x\}, \emptyset), \{x, z\}\rangle \to \langle 2, (\emptyset, \emptyset), \emptyset\rangle$ and $C : \langle 0, (\emptyset), \emptyset\rangle \to \langle 2, (\{x\}, \emptyset), \{x, z\}\rangle$.

We can also combine bigraphs by an associative **tensor product** (denoted by $\otimes$), which works simply by juxtaposition of roots. For tensor product we require only that both inner and outer names be disjoint.

Finally, in the following we will be particularly concerned with three classes of bigraphs — **prime** bigraphs are those with only a single root, and only local inner

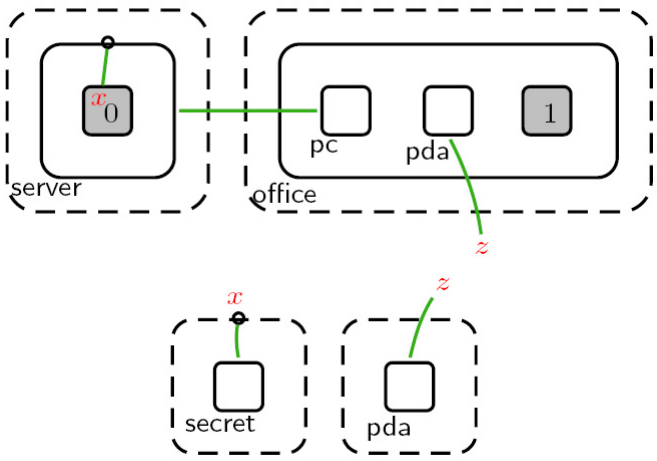


Figure 3.2: B and C – bigraphs that compose to form A

names. For **discrete** bigraphs all linkage upon global names is one-one while **name-discrete** bigraphs, are those where *all* linkage upon all names is one-one (refer to Definition 3.A.7 for the full definition of discreteness).

For more involved examples of bigraphical models including dynamics, we refer the reader to the tech report [DD05b].

3.3 Elementary Bigraphs and Normal Form

We start by defining **placings** corresponding closely to the placings defined for pure bigraphs in [Mil05]. We shall use placings to define the class of terms for bigraphs that denote place graphs paired with identities on local names.

$$\begin{array}{rcll} 1 & : & \epsilon \to 1 & \text{a barren root,} \\ \mathit{join} & : & 2 \to 1 & \text{join two sites,} \\ \gamma_{m,n,(\vec{X},\vec{Z})} & : & \langle m+n, \vec{X}\vec{Z}, \{\vec{X}\} \uplus \{\vec{Z}\}\rangle \to \langle m+n, \vec{Z}\vec{X}, \{\vec{X}\} \uplus \{\vec{Z}\}\rangle & \\ & & & \text{transpose } m \text{ with } n \text{ places preserving names.} \end{array}$$

Note that 1 and *join* are defined exactly as for pure bigraphs, while $\gamma_{m,n,(\vec{X},\vec{Z})}$ lets a set of local inner names be linked to corresponding outer names, in the only way allowed by the scope rule (see Definition 3.A.6).

We use π and ρ to range over **permutations**, placings generated by composition and tensor product from $\gamma_{m,n,(\vec{X},\vec{Z})}$.

For $I_i = \langle m_i, \vec{X}^i_B, \{\vec{X^i_B}\} \uplus X^i_F\rangle$ $(i \in \{0,1\})$ we define

$$\gamma_{I_0,I_1} \stackrel{\text{def}}{=} \gamma_{m_0,m_1,(\vec{X^0_B},\vec{X^1_B})} \otimes \mathsf{id}_{X^0_F} \otimes \mathsf{id}_{X^1_F}.$$

Using *join* we define the bigraph merge_m that joins m sites:

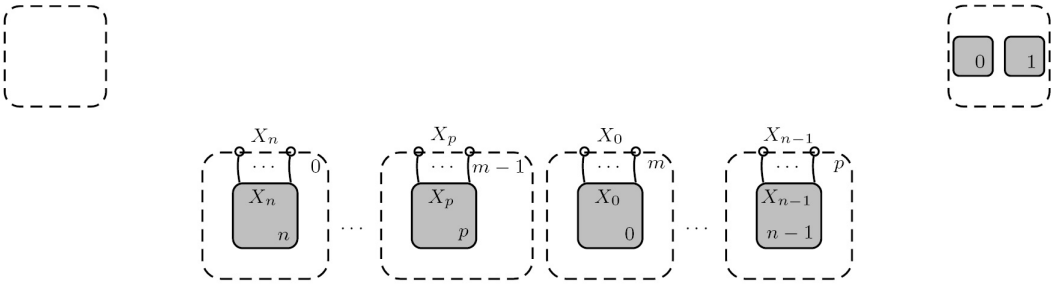


Figure 3.3: 1, *join*, and $\gamma_{m,n,(\vec{X},\vec{Z})}$ (using the abbreviation $p = m + n - 1$)

Definition 3.3.1 (merge)**.** For all $m \geq 0$ we define $merge_m : m \to 1$ recursively, by

$$\begin{array}{rcl} merge_0 & \stackrel{\text{def}}{=} & 1 \\ merge_{m+1} & \stackrel{\text{def}}{=} & join(\mathsf{id}_1 \otimes merge_m). \end{array}$$

Note that $merge_1 = \mathsf{id}_1$ and thus $merge_2 = join$.

A **linking** is a (pure) link graph $X \to Y$ that has no nodes. All linkings can be expressed in terms of the following two kinds:

$$\begin{array}{rcll} /x & : & x \to \epsilon & \text{closure,} \\ y/X & : & X \to y & \text{substitution } x \mapsto y, \text{ for all } x \in X. \end{array}$$

A closure closes a single link. For $X = \{x_0, \ldots, x_{k-1}\}$ and $k > 0$ we define a multiple closure $/X \stackrel{\text{def}}{=} /x_0 \otimes \cdots \otimes /x_{k-1}$. For $Y = \{y_0, \ldots, y_{k-1}\}$, $k > 0$, and disjoint sets $X_0, \ldots, X_{k-1}$ we define a multiple substition

$$\vec{y}/\vec{X} \stackrel{\text{def}}{=} y_0/X_0 \otimes \ \cdots \otimes \ y_{k-1}/X_{k-1}.$$

Note that a substitution need not be surjective (i.e., we allow $X = \emptyset$), thus the dual of closure – name introduction $y : \epsilon \to y$ – is a substitution. A **renaming** is a bijective (multiple) substitution, i.e., each X_i above is a singleton. A **wiring** is a bigraph with zero width (and hence no local names) generated by composition and tensor of $/x$ and y/X.

We let ω range over wirings, σ range over (multiple) substitutions and α and β range over renamings. Often we do not distinguish notationally between a name and the singleton set containing the name. With this convention $\vec{y}/\vec{x}$ is a renaming when $\vec{y} = y_0, \ldots, y_{k-1}$ and $\vec{x} = x_0, \ldots, x_{k-1}$, for some k.

A **simple concretion** is a discrete prime which maps a set X of local inner names severally to equally named global outer names.

$$\ulcorner X \urcorner \ : \ (X) \to \langle X \rangle \quad \text{concretion.}$$

Note that a special case of a simple concretion is $\mathsf{id}_1 = \ulcorner \emptyset \urcorner$.

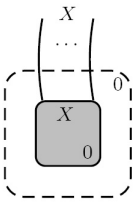


Figure 3.4: $\ulcorner X \urcorner$

An **abstraction** $(X)-$ is a *construction*, defined on every prime P that localizes a subset of the global names of P. For every prime $P : I \to \langle (Y_B), Y \rangle$, let

$$(X)P \quad : I \to \langle (Y_B \uplus X), Y \rangle \quad \text{abstraction on } P,$$

where $X \subseteq Y \setminus Y_B$.

Note that the scope rule is necessarily respected since the inner face of P is required to be local as P is prime. Abstractions are in some sense dual to concretions, and the axioms concerning abstraction and concretion reflect this (see axioms (B2) and (B3) in Table 3.1)

Using abstraction we can express concretions in the sense of [JM04]: We define a general **concretion** $\ulcorner Y \urcorner^X : \langle 1, (X \uplus Y), X \uplus Y \rangle \to \langle 1, (X), X \uplus Y \rangle$ in terms of a simple concretion and abstraction as $\ulcorner Y \urcorner^X \stackrel{\text{def}}{=} (X)\ulcorner X \uplus Y \urcorner$. Towards succinct statement of the normal form, we define $\ulcorner \alpha \urcorner \stackrel{\text{def}}{=} (\alpha \otimes \mathsf{id}_1)\ulcorner X \urcorner$ (where $\alpha : X \to$).

With the help of linkings we get **local wirings** — bigraphs that by composition can change the linkage of local names. We define a **local renaming** (for vectors of names $\vec{y}$ and $\vec{x}$, s.t. $|\vec{y}| = |\vec{x}|$) by $(\vec{y})/(\vec{x}) \stackrel{\text{def}}{=} (\vec{y})((\vec{y}/\vec{x} \otimes \mathsf{id}_1)\ulcorner \{\vec{x}\} \urcorner)$. We extend this notation to multiple substitutions and define $(\vec{y})/(\vec{X}) \stackrel{\text{def}}{=} (\vec{y})((\vec{y}/\vec{X} \otimes \mathsf{id}_1)\ulcorner X \urcorner)$ (for $X = \{\vec{X}\}$).

Just as plain substitutions can introduce idle global names, local substitutions can introduce idle local names when their underlying global substitution is not surjective (e.g., $(y)/(\emptyset)$).

We let $\alpha^{\mathbf{loc}}$ and $\sigma^{\mathbf{loc}}$ range over local renamings and substitutions, respectively. We shall need to take the preimage of a local substitution $\sigma^{\mathbf{loc}}$ of a vector of namesets $\vec{X}$. Formally:

Definition 3.3.2 (Preimage of a local wiring)**.** Let $\sigma^{\mathbf{loc}}_{\mathbf{u}}$ be the link map (which is a function) of $\sigma^{\mathbf{loc}}$. For a set of names X, define $(\sigma^{\mathbf{loc}})^{-1}(X)$ to be the preimage $(\sigma^{\mathbf{loc}}_{\mathbf{u}})^{-1}(X)$ and define $(\sigma^{\mathbf{loc}})^{-1}(\vec{X})$ to be the vector of namesets resulting from taking the preimage of $\sigma^{\mathbf{loc}}$ pointwise for each set in $\vec{X}$.

We can generate all isomorphisms in the category of binding bigraphs using permutations π, renamings α, and local renamings $\alpha^{\mathbf{loc}}$(see [JM04, Proposition 9.2b] for the definition of isomorphism in the category of binding bigraphs):

Proposition 3.3.3. *Every binding bigraph isomorphism,* $\iota : \langle m, \vec{Z}, \{\vec{Z}\} \uplus U\rangle \to \langle m, \vec{X}, \{\vec{X}\} \uplus Y\rangle$ *(of width* m*) may be expressed in the following form*

$$\iota = (\pi \otimes \alpha)(\nu_0 \otimes \cdots \otimes \nu_{m-1} \otimes \mathsf{id}_U)$$

where these requirements hold:

- $m = |\vec{X}| = |\vec{Z}|$,
- $\alpha : U \to Y$,
- $\forall i \in m : \nu_i = (\vec{x_i})/(\vec{z_i})$ *for* $\vec{X} = (\{\vec{x}_0\}, \ldots, \{\vec{x}_{m-1}\})$, *and* $\vec{Z} = (\{\vec{z}_0\}, \ldots, \{\vec{z}_{m-1}\})$.

For a control $K : b \to f \in \mathcal{K}$, let $\vec{y}$ be a sequence of distinct names, and $\vec{X}$ a sequence of sets of distinct names, s.t. $|\vec{X}| = b$ and $|\vec{y}| = f$.

A **binding ion** $K_{\vec{y}(\vec{X})} : \langle 1, (X), X\rangle \to \langle 1, (\emptyset), Y\rangle$ is a prime bigraph with a single node of control K with free ports linked severally to global outer names $\vec{y}$, and each binding port $i \in b$ linked to all local inner names in X_i. Figure 3.5 shows a binding ion.

$$K_{\vec{y}(\vec{X})} \quad : \quad (X) \to \langle Y\rangle \quad \text{a binding ion}$$

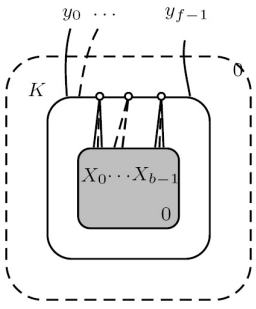


Figure 3.5: A binding ion

This definition of binding ion is a straightforward generalization of the **free discrete ion** defined in [JM04, Chapter 11]. We can recapture the latter by requiring every set in X to be a singleton. When $\vec{X} = (\{x_0\}, \ldots, \{x_{b-1}\})$, we overload our notation and write $K_{\vec{y}(\vec{x})}$ to mean a free discrete ion.

Definition 3.3.4. For any name-discrete prime $P : I \to \langle 1, (X), X \uplus Z\rangle$ and ion $K_{\vec{y}(\vec{X})}$, we define a **free discrete molecule** as

$$(K_{\vec{y}(\vec{X})} \otimes \mathsf{id}_Z)P : I \to \langle 1, (\emptyset), \{\vec{y}\} \uplus Z\rangle$$

Note that even though we use the more general binding ion in the definition above, our definition of free discrete molecule is equal to the one given in [JM04, Chapter 11], in the sense that it covers the same set of bigraphs.

As P in the above definition is discrete and prime it is easily seen that M is also discrete and prime. In fact:

Proposition 3.3.5. *A free discrete molecule is a name-discrete prime bigraph with a single outermost node.*

This proposition relies on both name-discreteness and discreteness being preserved by composition and tensor (Lemma 3.A.8). Vice versa, we have:

Proposition 3.3.6. *Any free discrete prime bigraph with a single outermost node is a free discrete molecule.*

3.3.1 A Normal Form for Binding Bigraphs

In the following section we present our binding discrete normal form theorem for graphs. This *semantic* theorem states that every binding bigraph can be decomposed in certain ways. We shall use it as the basis for the establishment of a corresponding *syntactic* definition of normal form for our term language for binding bigraphs, which we introduce in Section 3.4.

We aim to base our normal form on a variant of discreteness, as in [Mil05], simply as this allows a clean separation between the constituent components of a bigraph. Our main aim is to prove completeness for an equational theory over a term language for binding bigraphs. To that end it will be central to formulate an inductive property of expressions that characterizes our chosen variant of discreteness syntactically. Alas, discreteness is not preserved under composition with abstractions and concretions. Indeed, consider a discrete bigraph D with width n. $(\bigotimes_{i<n} {}^{\ulcorner}X_i{}^{\urcorner})D$ is not discrete, if D is not name-discrete. Conversely, given a nondiscrete prime $P : I \to \langle (X), X \uplus Y \rangle$, $(Y)P : I \to (X \uplus Y)$ is discrete. Hence, we turn to name-discreteness.

Recall that a bigraph is name-discrete (Definition 3.A.7) if every free link is an outer name and has exactly one point, and every bound link is either an edge, or (if it is an outer name) has exactly one point. This is a simple specialization of the discreteness property. As a consequence, it is easy to verify that both abstraction and composition with concretions preserve both name-discreteness and non-name-discreteness. Name-discreteness still allows arbitrary linking upon *bound* edges, and exactly for that reason, we have chosen to take the binding ion (as defined above) as a constant in our term language. Syntactically, this allows us to restrict the usage of substitutions to define a simple inductive property that characterizes name-discreteness.

Theorem 3.3.7 (Semantic binding discrete normal form)**.**

1. *Any free discrete molecule $M : I \to \langle 1, \{\vec{y}\} \uplus Z \rangle$ can be expressed as*

$$\left(K_{\vec{y}(\vec{X})} \otimes \mathsf{id}_Z\right) P$$

where $P : I \to \langle 1, (\{\vec{X}\}), \{\vec{X}\} \uplus Z\rangle$ *is a name-discrete prime.*

This expression is unique up to renaming of the local names on the innerface of the ion, and (correspondingly) on the outer face of prime P*. Hence, any other such expression for* M *takes the form*

$$\left(K_{\vec{y}(\vec{X'})} \otimes \mathsf{id}_Z\right) P'$$

where the following requirements hold:

- *there exists a local renaming* $\alpha^{\mathbf{loc}} : (\{\vec{X'}\}) \to (\{\vec{X}\})$ *s.t.* $K_{\vec{y}(\vec{X})}\alpha^{\mathbf{loc}} = K_{\vec{y}(\vec{X'})}$*, and*
- $P = (\alpha^{\mathbf{loc}} \otimes \mathsf{id}_Z)P'$*.*

2. *Any name-discrete prime* $P : I \to \langle 1, (Y_B), Y\rangle$ *may be expressed as*

$$(Y_B)\left(merge_{n+k} \otimes \mathsf{id}_Y\right)\left(\ulcorner \alpha_0 \urcorner \otimes \cdots \otimes \ulcorner \alpha_{n-1} \urcorner \otimes M_0 \otimes \cdots \otimes M_{k-1}\right)\pi$$

where every $M_i : J_i \to \langle Y_i^{\mathbf{M}}\rangle$ *is a free discrete molecule, and for renamings* $\alpha_i : X_i \to Y_i^{\mathbf{C}}$*, we have* $Y = (\biguplus_{i\in n} Y_i^{\mathbf{C}}) \uplus \biguplus Y_i^{\mathbf{M}}$*.*

The expression for P *is unique up to reordering of the concretions and molecules, and the ordering of the sites inside the molecules; the permutation changes accordingly to preserve the innerface. Formally, any other such expression for* P *takes the form*

$$(Y_B)\left(merge_{n+k} \otimes \mathsf{id}_Y\right)\left(\ulcorner \alpha'_0 \urcorner \otimes \cdots \otimes \ulcorner \alpha'_{n-1} \urcorner \otimes M'_0 \otimes \cdots \otimes M'_{k-1}\right)\pi'$$

where the following requirements hold:

- *There exist permutations* ρ*,* ρ_i $(i \in k)$*,* ρ'*, s.t.*
  - $\ulcorner \alpha'_i \urcorner = \ulcorner \alpha_{\rho(i)} \urcorner$
  - $M'_i = M_{\rho(i)}\rho_i$*,*
  - $(\mathsf{id}_{(X'_0)} \otimes \cdots \otimes \mathsf{id}_{(X'_{n-1})} \otimes \rho_0 \otimes \cdots \otimes \rho_{k-1})\pi' = \rho'\pi$*.*
- *Furthermore, let* $\vec{l}$ *denote the vector of inner widths of the product* $((\alpha_0 \otimes \mathsf{id}_1)\ulcorner X_0 \urcorner \otimes \ldots \otimes (\alpha_{n-1} \otimes \mathsf{id}_1)\ulcorner X_{n-1} \urcorner \otimes M_0 \otimes \cdots \otimes M_{k-1})$*, let* $\vec{X'} = (X'_0, \ldots, X'_{k-1})$*, and let* $\vec{X} = (X_0, \ldots, X_{n-1})$*.*

  Then ρ' *is determined uniquely by* ρ*,* $\vec{l}$*,* $\vec{X}$*, and* $\vec{X'}$ *as* $\rho' = \overline{\rho}_{\vec{l},\vec{X'}\vec{X}}$ *as defined in Lemma 3.4.2.*

3. *Any name-discrete bigraph* D *(of outer width* n*) can be expressed as*

$$(P_0 \otimes \cdots \otimes P_{n-1})\,\pi \otimes \alpha$$

where every P_i *is a name-discrete prime,* α *is a renaming, and* π *is a permutation.*

This expression is unique up to reordering of the sites in the primes; the permutation changes accordingly to preserve the innerface. Hence, any other such expression of D takes the form

$$\left(P'_0 \otimes \cdots \otimes P'_{n-1}\right) \pi' \otimes \alpha$$

where there exists permutations ρ_i, $(i \in n)$, s.t. $P'_i = P_i \rho_i$, and $(\rho_0 \otimes \cdots \otimes \rho_{n-1})\pi' = \pi$.

4. *Any bigraph $G : I \to \langle n, \vec{Y}_B, \{\vec{Y_B}\} \uplus Y_F \rangle$ can be expressed as*

$$\left(\bigotimes_{i<n} (\vec{y_i})/(\vec{X}_i) \otimes \omega\right) D$$

where $D : I \to \langle n, \vec{X}, X \uplus Z \rangle$ is name-discrete, $\omega : Z \to Y_F$ is a wiring, and $\bigotimes_{i<n}(\vec{y_i})/(\vec{X}_i) : (\vec{X}) \to (\vec{Y_B})$ is a local substitution of width n on the bound names of D.

The expression is unique up to (local and global) renamings on the innerface of the wiring and (correspondingly) on the outerface of D. Hence, any other such expression of G takes the form

$$\left(\bigotimes_{i<n} (\vec{y_i})/(\vec{X'}_i) \otimes \omega'\right) D'$$

where there exists a renaming α s.t. $\omega' = \omega\alpha$, and n local renamings $\alpha_i^{\mathbf{loc}} : (\vec{X'}_i) \to (\vec{X}_i)$, s.t. $(\bigotimes_{i<n}(\vec{y_i})/(\vec{X}_i)) \bigotimes_{i<n} \alpha_i^{\mathbf{loc}} = (\bigotimes_{i<n}(\vec{y_i})/(\vec{X'}_i))$, and $\left(\bigotimes_{i<n} \alpha_i^{\mathbf{loc}} \otimes \alpha\right) D' = D$.

Furthermore, for every class of expressions the expression given is well defined and generates only *bigraphs of the appropriate type.*

See [DB05] for a proof of the theorem. The proof is simply a detailed analysis of the structure of possible decompositions of binding bigraphs.

3.4 Binding Bigraph Expressions and Axioms

The set of **binding bigraph expressions** is defined as the smallest set of expressions built by composition, tensor product, and abstraction (on primes) from identities and the constants we have just introduced:

$$1 \qquad \mathit{join} \qquad \gamma_{m_0,m_1,(\vec{X}_0,\vec{X}_1)} \qquad /x \qquad y/X \qquad \ulcorner X \urcorner \qquad K_{\vec{y}(\vec{X})}$$

Each expression (implicitly) has two interfaces of the form $\langle m, \vec{X}, Y \rangle$ which determine when tensor product, composition, and abstraction are well defined (according

to the requirements stated formally in 3.A). The interface and the bigraph an expression denotes can be determined by induction. As usual, we write $\vDash E = F$ to mean that the expression $E = F$ is **valid**; and $\vdash E = F$ if the equation is **provable**.

In [Mil05] Milner stated and proved a set of axioms complete for pure bigraph expressions. We extend that result and prove the set of axioms in Table 3.1 complete for binding bigraph expressions.[1] Every pure bigraph expression as defined by Milner [Mil05] trivially corresponds to a binding bigraph expression as defined above. Our axiomatic theory is a conservative extension of Milner's in the sense that any two pure bigraph expressions are provably equal in Milner's theory iff the corresponding expressions are provably equal in our theory. (Formally, this is easy to prove using soundness and completeness of the two theories and the fact that the embeddings of pure bigraphs into binding bigraphs and pure bigraph expressions into binding bigraph expressions are both full and faithful). We proceed by defining and proving the theory complete for increasingly larger classes of expressions.

Note that as tensor product is defined only when name sets of the interfaces are disjoint, and as abstraction is defined only on prime bigraphs with the abstracted names in the outer face, we only require the equations to hold when both sides are defined.

3.4.1 Preliminaries

Lemma 3.4.1 (Wiring commutes with all binding bigraph expressions)**.** *For all bigraph expressions G and for all wirings $\omega \vdash G \otimes \omega = \omega \otimes G$.*

By essentially iterating axiom C9, we can push a permutation "through" a product of primes, permuting the order in which they appear in the product, and producing a permutation that reorders the sites in the primes to preserve the inner face.

Lemma 3.4.2 (Push-through lemma)**.** *For n primes P_i*

$$P_i \quad : \quad \langle m_i, \vec{X}_i, \{\vec{X}_i\}\rangle \to \langle 1, (Y_i^{\mathrm{B}}), Y_i\rangle,$$

and permutation π, there exists a permutation $\overline{\pi}_{\vec{m},\vec{X}}$, which depends solely on π, $\vec{m}$, and $\vec{X} = (\vec{X}_0, \ldots, \vec{X}_{n-1})$, s.t.,[2]

$$\vdash \pi \circ (P_0 \otimes \cdots \otimes P_{n-1}) = (P_{\pi^{-1}(0)} \otimes \cdots \otimes P_{\pi^{-1}(n-1)}) \circ \overline{\pi}_{\vec{m},\vec{X}}.$$

3.4.2 $\mathbf{Place_L}$ expressions

Let $\mathbf{Place_L}$ expressions be all expressions in the term language generated by $\circ$, and $\otimes$ from $bmerge_{m,\vec{X}}$ (defined below) and $\gamma_{I,J}$. Thus, $\mathbf{Place_L}$ consists of all

[1] In the published version, the axiom (C10) was mistakenly left out.

[2] In the published version, the following equation mistakenly used $P_{\pi(i)}$ instead of $P_{\pi^{-1}(i)}$.

Categorical axioms

(C1)	$A\ \mathsf{id}_I$	$=$ A	$= \mathsf{id}_J\ A$	$(A : I \to J)$
(C2)	$A(BC)$	$=$	$(AB)C$	
(C3)	$A \otimes \mathsf{id}_\epsilon$	$=$ A	$= \mathsf{id}_\epsilon \otimes A$	
(C4)	$A \otimes (B \otimes C)$	$=$	$(A \otimes B) \otimes C$	
(C5)	$\mathsf{id}_I \otimes \mathsf{id}_J$	$=$	$\mathsf{id}_{I \otimes J}$	
(C6)	$(A_1 \otimes B_1)(A_0 \otimes B_0)$	$=$	$(A_1 \circ A_0) \otimes (B_1 \circ B_0)$	
(C7)	$\gamma_{I,\epsilon}$	$=$	id_I	
(C8)	$\gamma_{J,I}\ \gamma_{I,J}$	$=$	$\mathsf{id}_{I \otimes J}$	
(C9)	$\gamma_{I,K}(A \otimes B)$	$=$	$(B \otimes A)\gamma_{H,J}$	$(A : H \to I, B : J \to K)$
(C10)	$\gamma_{I \otimes J,K}$	$=$	$(\gamma_{I,K} \otimes \mathsf{id}_J)(\mathsf{id}_I \otimes \gamma_{J,K})$	

Link axioms

(L1)	x/x	$=$	id_x
(L2)	$/y \circ y/x$	$=$	$/x$
(L3)	$/y \circ y$	$=$	id_ϵ
(L4)	$z/(Y \uplus y)(\mathsf{id}_Y \otimes y/X)$	$=$	$z/(Y \uplus X)$

Place axioms

(P1)	$join(1 \otimes \mathsf{id}_1)$	$=$	id_1
(P2)	$join(join \otimes \mathsf{id}_1)$	$=$	$join(\mathsf{id}_1 \otimes join)$
(P3)	$join\, \gamma_{1,1,(\emptyset,\emptyset)}$	$=$	$join$

Binding axioms

(B1)	$(\emptyset)P$	$=$	P	
(B2)	$(Y)^{\ulcorner} Y^{\urcorner}$	$=$	$\mathsf{id}_{(Y)}$	
(B3)	$({}^{\ulcorner} X^{\urcorner Z} \otimes \mathsf{id}_Y)(X)P$	$=$	P	$(P : I \to \langle 1, (Z), Z \uplus X \uplus Y \rangle$
(B4)	$(((Y)(P)) \otimes \mathsf{id}_X)G$	$=$	$(Y)(P \otimes \mathsf{id}_X)G$	
(B5)	$(X \uplus Y)P$	$=$	$(X)((Y)P)$	

Ion axioms

(N1)	$(\mathsf{id}_1 \otimes \alpha)K_{\vec{y}(\vec{X})}$	$=$	$K_{\alpha(\vec{y})(\vec{X})}$
(N2)	$K_{\vec{y}(\vec{X})}\sigma^{\mathbf{loc}}$	$=$	$K_{\vec{y}((\sigma^{\mathbf{loc}})^{-1}(\vec{X}))}$

Table 3.1: Axioms for binding bigraphs

expressions denoting place graphs paired with identities on local names. We shall start by proving that the theory is complete for $\mathbf{Place_L}$ expressions.

To that end, we extend the place merging expression *join* to local interfaces.

Definition 3.4.3 (binding join)**.** For sets of names X and Y let $bjoin_{(X,Y)}$, the *binding join* bigraph, be defined as

$$bjoin_{(X,Y)} \stackrel{\text{def}}{=} (X \uplus Y)((join \otimes \mathsf{id}_{X \uplus Y}) \circ (\ulcorner X \urcorner \otimes \ulcorner Y \urcorner)).$$

We also define an iterated version

Definition 3.4.4 (binding merge)**.** For all $m \geq 0$ we define $bmerge_{m,\vec{X}}$ recursively, by

$$\begin{aligned} bmerge_{0,()} &\stackrel{\text{def}}{=} 1 \\ bmerge_{m+1,\vec{X}Y} &\stackrel{\text{def}}{=} bjoin_{(\{\vec{X}\},Y)} \circ (bmerge_{m,\vec{X}} \otimes \mathsf{id}_Y) \end{aligned}$$

Binding join and merge behave similarly as their underlying place expressions when composed with permutations or themselves (refer the place graph axioms of Table 3.1), though, as they have (local) names on their faces their interplay with names is not as simple. The lemma below reflects this, and also states that merging a product of binding merges yields a binding merge.

Lemma 3.4.5.

$$\begin{aligned} &\vdash bjoin_{(X_1,X_0)} \circ \gamma_{1,1,(X_0,X_1)} = bjoin_{(X_0,X_1)}, \\ &\vdash bmerge_{m,\pi(\vec{X})} \circ \pi = bmerge_{m,\vec{X}}, \\ &\vdash bmerge_{k,\vec{X}} \circ (\textstyle\bigotimes_{i<k} bmerge_{m_i,\vec{X_i}}) = bmerge_{m,\vec{X}}, \end{aligned}$$

where in the last equation $m = \sum_{i<k} m_i$ *and* $\vec{X} = \vec{X_0} \ldots \vec{X_{k-1}}$.

Using binding merge, we define and prove sufficient a normal form for $\mathbf{Place_L}$ expressions.

Lemma 3.4.6 (Normal form for $\mathbf{Place_L}$ expressions)**.** *For every* ***Place*$_\mathbf{L}$** *expression* E

$$\vdash E \;=\; (bmerge_{m_0,\vec{X_0}} \otimes \cdots \otimes bmerge_{m_{k-1},\vec{X_{k-1}}}) \circ \pi$$

for some $k \geq 0$ *and permutation expression* π *s.t. the composition is well defined.*

With the help of Lemma 3.4.5 the proof is simple by induction on the structure of expressions.

Note that in a strict symmetric monoidal category the categorical axioms are known to be complete for $\circ$ and $\otimes$ of the symmetries $\gamma_{I,J}$ — hence the theory is complete for permutations.

Full completeness for $\mathbf{Place_L}$ expressions follows with the help of the uniqueness properties stated in Theorem 3.3.7. These yield a number of equations which are provable within the theory.

Proposition 3.4.7 (Completeness for $\mathbf{Place_L}$ expressions)**.** *If* $\vdash E = \bigotimes_{i<k} bmerge_{m_i,\vec{X}_i} \circ \pi$ *and* $\vdash F = \bigotimes_{j<l} bmerge_{n_j,\vec{Y}_j} \circ \pi'$ *and* $\models E = F$*, then* $\vdash E = F$*.*

3.4.3 $\mathbf{Link_G}$ expressions

We now consider the class of global link expressions, those bigraph expressions generated by composition and tensor of closure and substitution. We will refer to this collection of expressions as $\mathbf{Link_G}$. Our term language for binding bigraphs has the same constructs for linking as the language used by Milner for pure bigraphs [Mil05]. Since we also have the exact same axioms for global link expressions, it is easily seen that the proof that the axiomatic theory for the binding bigraph term language is complete for global link expressions is entirely the same.

Proposition 3.4.8 (Link completeness)**.** *The theory is complete for link expressions.*

3.4.4 Linear Bigraph Expressions

We now define an important kind bigraph expressions – **linear** expressions, which we shall prove to be a syntactic analogue to name-discrete bigraphs, in the sense that any name-discrete bigraph has a linear expression.

Definition 3.4.9 (Linearity)**.** A binding bigraph expression is **linear** iff it contains only linkings of the form y/x.

In other words, in linear expressions all substitutions are renamings, and there are no closures. This is an inductive property with respect to the term language, which we will utilize to full effect in the following sections.

We start by establishing some basic properties of linear expressions. The proofs of the following lemmas are all by induction on the structure of expressions.

Lemma 3.4.10. *If* E *is linear expression, then* $\vdash E = E' \otimes \alpha$*, where* E' *is linear and has local innerface.*

Lemma 3.4.11. *If* $E : \langle m, \vec{U}, \{\vec{U}\}\rangle \to \langle n, \vec{Y}, \{\vec{Y}\} \uplus V\rangle$ *is a linear expression with local innerface, then*

$$\vdash E \circ \bigotimes_{i<m} (\vec{u_i})/(\vec{Z_i}) = \left(\left(\bigotimes_{i<n} (\vec{y_i})/(\vec{X_i})\right) \otimes \mathsf{id}_V\right) \circ E',$$

for some $\vec{y}$*,* $\vec{X}$*, and* E' *where* E' *is linear with local innerface.*

We shall use the following proposition to show completeness for ion-free expressions in the following section. Importantly, it also constitutes a step towards a syntactic normal form for bigraph expressions, analogous to the semantic normal form we established in Theorem 3.3.7, item 4.

Proposition 3.4.12 (Underlying linear expression)**.** *For any expression G denoting a bigraph of outer width n, there exists a wiring ω, a linear expression E, and a local renaming $\bigotimes_{i<n}(\vec{y}_i)/(\vec{X}_i)$, s.t.,*

$$\vdash G = (\bigotimes_{i<n}(\vec{y}_i)/(\vec{X}_i) \otimes \omega) \circ E.$$

The proof is by structural induction on G, using the lemmas above [DB05].

3.4.5 Ion-free Expressions

Let us now consider ion-free expressions – all expressions in our term language, that does not contain ions ($K_{\vec{y}(\vec{X})}$). We proceed as above, by showing that ion-free expressions can be decomposed into simpler expressions.

Lemma 3.4.13. *If $E = E_1 \circ E_2$ or $E = E_1 \otimes E_2$ is linear, ion-free, and with local inner and outer face, then E_1 and E_2 are also linear and ion-free with local inner and outer face.*

Lemma 3.4.14. *If E is linear and ion-free of width n with local inner and outer face, then $\vdash E = \bigotimes_{i<n}(\vec{y}_i)/(\vec{x}_i) \circ G^P$, where $G^P \in \boldsymbol{Place_L}$.*

Lemma 3.4.15. *If E is linear and ion-free, then there exists concretions, E', and a renaming α s.t. $\vdash E = (\bigotimes_{i<n} {}^{\ulcorner}X_i{}^{\urcorner Z_i} \circ E') \otimes \alpha$, with E' linear and ion-free and local inner and outer face.*

With the help of the above lemmas we can now establish a normal form for ion-free expressions.

Lemma 3.4.16 (A normal form for ion-free expresssions)**.** *For all ion-free expressions G of width n*

$$\vdash G = \omega \otimes \left(\bigotimes_{i<n} (Y_i) \left((\rho \otimes \mathsf{id}_1) \circ {}^{\ulcorner}X_i{}^{\urcorner} \right) \right) \circ G^P.$$

where $G^P \in \boldsymbol{Place_L}$.

Completeness for ion-free expressions follows easily.

Corollary 3.4.17 (The theory is complete for ion-free expressions)**.**

3.4.6 Syntactic Normal Form

Corresponding to the four classes of normal forms in Theorem 3.3.7 we define four classes of syntactic normal forms for binding bigraph expressions:

Definition 3.4.18 (Syntactic binding discrete normal form (BDNF))**.**

$$\begin{array}{lrcl}
\text{MDNF} & M & ::= & (K_{\vec{y}(\vec{X})} \otimes \mathsf{id}_Z)P \\
\text{PDNF} & P & ::= & (X)\left(merge_{n+k} \otimes \mathsf{id}_Y\right) \\
 & & & \quad (\ulcorner \alpha_0 \urcorner \otimes \cdots \otimes \ulcorner \alpha_{n-1} \urcorner \otimes M_0 \otimes \cdots \otimes M_{k-1})\,\pi \\
\text{DDNF} & D & ::= & (P_0 \otimes \cdots \otimes P_{n-1})\pi \otimes \alpha \\
\text{BDNF} & B & ::= & (\bigotimes_{i<n}(\vec{y_i})/(\vec{X_i}) \otimes \omega)D.
\end{array}$$

The proofs of the following lemmas go by induction on the number of ions. As we have established completeness for ion-free expressions, we have the base case.

Lemma 3.4.19 (All BDNF forms are closed under composition with isos)**.**

We also need that DBDNF expressions are closed under composition.

Lemma 3.4.20 (DBDNF is closed under composition)**.** *For all composable* DBDNF*s* C, D*, there exists a* DBDNF D'*, s.t.* $\vdash D \circ C = D'$*.*

We now state the correspondence between our semantic normal form (Theorem 3.3.7) and the syntactic normal form above. Moreover, we state that linearity is, in fact, a syntactic correspondent to name-discreteness (item 3 in the following proposition):

Proposition 3.4.21 (provable normal forms)**.** *Let* E *be a linear expression, and* G *any expression.*

1. *If* E *denotes a free discrete molecule, then* $\vdash E = M$ *for some* MDNF*.*
2. *If* E *denotes a name-discrete prime, then* $\vdash E = P$ *for some* PDNF P*.*
3. $\vdash E = D$ *for some* DDNF D*.*
4. $\vdash G = B$ *for some* BDNF B*.*

We are now able to state the formal completeness proposition, using our results for linear expressions to bridge the gap to the full binding bigraph term language.

As we have laboured to establish a correspondence between each level of BDNF form and each level of the semantic normal form, in the proofs we are able to proceed by case analysis on the form of the bigraph the expression denotes, and then apply the uniqueness properties spelled out in Theorem 3.3.7 to yield a number of equations that are provable within our theory. We refer to the companion technical report [DB05] for more details on the proofs.

Proposition 3.4.22 (Linear completeness)**.** *If* E *and* E' *are linear expressions and* $E = E'$*, then* $\vdash E = E'$*.*

Theorem 3.4.23 (Soundness and Completeness)**.** *For all binding bigraph expressions* E *and* F*,* $\vDash E = F$ *iff* $\vdash E = F$*.*

3.5 Term Language and Normal Forms – by Example

We shall use our examples from Section 3.2 to give a few examples of the term language and the syntactic binding discrete normal form.

Using a modicum amount of shorthand, an expression for C is $((x)(\mathrm{secret}_x \circ 1)) \otimes \mathrm{pda}_z \circ 1$, while B can be expressed for example as

$$\begin{array}{l}(\mathsf{id}_2 \otimes /\{e0, e1\}) \circ \big((\mathrm{server}_{e0(\{x\})}) \otimes \\ \quad (\mathsf{id}_{\langle 1,e1\rangle} \otimes /\{f0, f1\})(\mathrm{office} \circ 1 \otimes \mathsf{id}_{\{e1,f0,f1\}}) \\ \quad (\mathit{merge}_3 \otimes \mathsf{id}_{\{e1,f0,f1\}})(\mathrm{pc}_{e1} \circ 1 \otimes \mathrm{pda}_{f0} \circ 1 \otimes \mathrm{pda}_{f1} \circ 1))\big).\end{array}$$

Hence, A can be expressed, simply by putting a $\circ$ between the two expressions for B and C.

On the other hand, giving an expression on BDNF for either bigraph, requires us to break it down into molecules, prime parts and not use non-linear linkage except at the topmost level. As an example, we give an expression on normal form for B:

$$\begin{array}{l}(\mathsf{id}_2 \otimes /\{e0, e1\} \otimes /\{f0, f1\}) \circ \\ \quad \big(\ (\emptyset)(\mathit{merge}_1 \otimes \mathsf{id}_{e0})(\mathrm{server}_{e0(\{x\})} \circ (x)(\mathit{merge}_1 \otimes \mathsf{id}_x)(\ulcorner x \urcorner)) \otimes \\ \quad\quad (\emptyset)(\mathit{merge}_1 \otimes \mathsf{id}_{\{e1,f0,f1\}})(\mathrm{office} \circ \\ \quad\quad\quad (\emptyset)(\mathit{merge}_3 \otimes \mathsf{id}_{\{e1,f0,f1\}})(\mathrm{pc}_{e1} \circ 1 \otimes \mathrm{pda}_{f0} \circ 1 \otimes \mathrm{pda}_{f1} \circ 1)) \\ \quad \otimes \ \mathsf{id}_\epsilon\big).\end{array}$$

Here we do not show identity permutations, and we write 1 instead of PBDNF for 1 (which is $(\emptyset)(\mathit{merge}_0 \otimes \mathsf{id}_\epsilon)$).

3.6 Related and Further Work

Bigraphical reactive systems are related to graph transformation systems using the double pushout construction [Ehr79] and, recently, it has also been investigated how to derive bisimulation congruences in the double pushout approach to graph rewriting [EK04].

Recent work on spatial logics [CMS05b] for pure bigraphs utilizes the axiomatization of pure bigraphs by Milner [Mil05]. An obvious line of further work is to utilize the algebraic theory presented here for binding bigraphs to extend the spatial logics to binding.

As mentioned in the introduction, jointly with the other members of our Bigraphical Programming Languages group, we are are currently working on an implementation of bigraphical reactive systems.

Further work is needed to relate tools based on graph rewriting to our work on Bigraphical Programming Languages.

Currently our experimental implementation of bigraphical reactive systems represents bigraphs internally by *normal form bigraphical expressions* that denote bigraphs. We have also developed a proposal for a surface language which users can

use to define bigraphical reactive systems — expressions of the surface language denote binding bigraphs and can thus be transformed to binding discrete normal forms: the proofs of the normal form theorems of this paper are constructive in nature and thus define algorithms than can be used to transform arbitrary bigraph expressions into normal form.

The core problem of implementing the dynamics of bigraphical reactive systems is the *matching problem*, that is, to determine for a given bigraph and reaction rule whether and how the reaction rule can be applied to rewrite the bigraph.

The abstract semantic definition of matching, as defined in the theory of bigraphs [JM04], is roughly as follows (omitting many details): Given a reaction rule with redex R and reactum R' (with R and R' both bigraphs), and a bigraph A (the agent to be rewritten), if $A = C \circ R \circ d$, then it can be rewritten to $C \circ R' \circ d$. Here $\circ$ denotes composition of bigraphs. In other words, if the reaction rule *matches* A, in the sense that A can be decomposed into a context C, redex R and a parameter d, then A can be rewritten.

Phrased in terms of binding bigraph *expressions*, the decision problem for matching is then roughly the following. Given binding bigraph expressions R, A, C, and d, determine whether $\vDash A = C \circ R \circ d$ holds. We have worked out an *inductive characterization* of when $\vDash A = C \circ R \circ d$ holds, by induction on the normal forms for A and R (the input to a matching algorithm). It is a precise characterization in the sense that it is both sound and complete. This provides a detailed analysis of the matching problem, and paves the way for developing and proving correct an actual matching algorithm (which, given A and R, must find a C and d such that $\vDash A = C \circ R \circ d$ holds). We will report on our work on the inductive characterization and on an actual matching algorithm in a subsequent paper.

We intend to use the implementation of bigraphical reactive systems to evaluate also in practice how well bigraphical models of ubiquitous systems [BDE+05] work.

Axknowledgements We are grateful for useful discussions of this work with all members of the BPL group at the IT University of Copenhagen, in particular Arne Glenstrup and Søren Debois; and with Robin Milner.

3.A Definition of Binding Bigraphs

We recall the definition of binding bigraphs [JM04].

Definition 3.A.1 (binding signature)**.** A **binding signature** $\mathcal{K}$ is a set of **controls**. For each $K \in \mathcal{K}$ it provides a pair of finite ordinals: the **binding arity** $\mathrm{ar}_{\mathbf{b}}(K) = h$ and the **free arity** $\mathrm{ar}_{\mathbf{f}}(K) = k$. We write $\mathrm{ar}(K) = \mathrm{ar}_{\mathbf{b}}(K) + \mathrm{ar}_{\mathbf{f}}(K)$.

Definition 3.A.2 (binding interface)**.** A **binding interface** $I = \langle m, loc, X \rangle$, consists of a **width** m, a finite set of **names** X, and a **locality map** $loc : X \to m \uplus \perp$, which associates some of the names in X with a location in m; if $loc(x) = i \in m$, we say x is **located** at i or **local** to i. When $loc(x) = \perp$ we say x is **global**.

For an interface $I = \langle m, loc, X\rangle$ we shall typically represent the locality map by a vector of disjoint subsets $\vec{X} = (X_0, \ldots, X_{m-1})$, where X_i is the set of names local to $i \in m$. If I is global, meaning that all names in I are global, then we may write I simply as $\langle m, X\rangle$; just m, if $X = \emptyset$; or just X, if $m = 0$.

We call I **prime** if $m = 1$. In that case, we shall sometimes write I as $\langle (X), Y\rangle$; just (X), if it is local; or just $\langle Y\rangle$, if it is global.

We use ϵ to denote the interface $\langle 0, (), \emptyset\rangle$.

A binding bigraph will have two binding interfaces and will be a pairing of a **place graph**, and a **link graph** following a structural requirement, the **scope rule** (see Definition 3.A.6).

We start by calling to mind the definitions of place graphs and link graphs.

Definition 3.A.3 (place graph)**.** A **(concrete) place graph** over signature $\mathcal{K}$ $G = (V, ctrl, prnt) : m \to n$ has an **inner width** m and an **outer width** n, both finite ordinals; a finite set V of nodes with a control map $ctrl : V \to \mathcal{K}$; and a **parent map** $prnt : m \uplus V \to V \uplus n$. The parent map is **acyclic**, i.e., $prnt^k(v) \neq v$, for all $k > 0$ and $v \in V$.

The parent map $prnt$ represents a forest of n unordered trees. The widths m and n of $G : m \to n$ index G's **sites** $0, \ldots, m-1$ and **roots** $0, \ldots, n-1$, respectively. We use ϵ to denote the width 0. A place graph with inner width 0 is called an **agent**.

Place graphs are composed as follows. Let $G_i = (V_i, ctrl_i, prnt_i) : m_i \to m_{i+1}$ $(i \in \{0, 1\})$ be place graphs with $V_0 \cap V_1 = \emptyset$; then $G_1 \circ G_0 \stackrel{\text{def}}{=} (V, ctrl, prnt)$, where $V = V_0 \uplus V_1$, $ctrl = ctrl_0 \uplus ctrl_1$, and $prnt = (\mathsf{id}_{V_0} \uplus prnt_1) \circ (prnt_0 \uplus \mathsf{id}_{V_1})$.

The identity place graph at m is $\mathsf{id}_m \stackrel{\text{def}}{=} (\emptyset, \emptyset, \mathsf{id}_m) : m \to m$.

The tensor product $I \otimes J$ of two interfaces $I = m$ and $J = n$ is simply $m + n$, and the tensor product of two place graphs $F : k \to l$ and $G : m \to n$ with disjoint node sets is $F \otimes G : k + m \to l + n$. It consists of placing the two forests side-by-side (see [JM04, Definition 7.5] for a formal definition). Note that $\mathsf{id}_\epsilon = \mathsf{id}_0$ is the unit for $\otimes$, in the sense that $F \otimes \mathsf{id}_\epsilon = \mathsf{id}_\epsilon \otimes F = F$, for all place graphs F. Thus, an iterated tensor product $F_0 \otimes \cdots \otimes F_{k-1}$ equals id_ϵ in case $k = 0$.

Two concrete place graphs G_0 and G_1 are said to be **support equivalent**, $G_0 \eqsim G_1$, if they differ only by a bijection between their node sets. An **abstract place graph** is an $\eqsim$-equivalence class of concrete place graphs. Composition and identity of abstract place graphs is given by composition and identity of concrete place graphs, and this provides a well-defined **category of place graphs** with interfaces as objects and abstract place graphs as morphisms. The induced tensor product on abstract place graphs, defined by $[F]_\eqsim \otimes [G]_\eqsim \stackrel{\text{def}}{=} [F \otimes G]_\eqsim$, makes it into a strict symmetric monoidal category.

Definition 3.A.4 (link graph)**.** A (**concrete**) **link graph** G over a signature $\mathcal{K}$, is a tuple $(V, E, ctrl, link) : X \to Y$ with finite sets of nodes V, edges E, **inner names** X, and **outer names** Y. As place graphs it has a control map $ctrl : V \to \mathcal{K}$. The function $link : X \uplus P \to E \uplus Y$ maps **points**, i.e., inner names X and ports $P = \sum_{v \in V} \mathrm{ar}(ctrl\, V)$ of G to **links**, i.e., outer names Y and edges E.

We call a link **idle** if it has no preimage under *link*. An outer name is an **open** link, and an edge is a **closed** link. A point is called **open** if its link is open, otherwise closed. Further, we call two distinct points on the same link **peers**.

The composition of two link graphs $G_i = (V_i, E_i, ctrl_i, link_i) : X_i \to X_{i+1}$ ($i \in \{0,1\}$) is defined when $V_0 \cap V_1 = \emptyset$ and $E_0 \cap E_1 = \emptyset$; and is then $G_1 \circ G_0 \stackrel{\text{def}}{=} (V, E, ctrl, link) : X_0 \to X_2$; where $V = V_0 \uplus V_1$, $E = E_0 \uplus E_1$, $ctrl = ctrl_0 \uplus ctrl_1$, and $link = (\mathsf{id}_{E_0} \uplus\ link_1) \circ (link_0 \uplus \mathsf{id}_{P_1})$.

The identity link graph at X is $\mathsf{id}_X \stackrel{\text{def}}{=} (\emptyset, \emptyset, \emptyset, \mathsf{id}_X) : X \to X$.

The tensor product of two link graph interfaces X and Y is the disjoint union, $X \uplus Y$, and is defined only when X and Y are disjoint. Tensor product of link graphs $G_i = (V_i, E_i, ctrl_i, link_i) : X_i \to Y_i$ is the disjoint union of the underlying constituents $G_0 \otimes G_1 \stackrel{\text{def}}{=} (V_0 \uplus V_1, E_0 \uplus E_1, ctrl_0 \uplus ctrl_1,$
$link_0 \uplus link_1) : X_0 \otimes X_1 \to Y_0 \otimes Y_1$, and is defined only when the interfaces are defined.

Definition 3.A.5 (binding bigraph)**.** A (**concrete**) **binding bigraph** $G = (V, E, ctrl, G^{\mathbf{P}}, G^{\mathbf{L}}) : I \to J$ over a signature $\mathcal{K}$ has an **inner interface** (or **inner face**) $I = \langle m, loc_I, X\rangle$ and an **outer interface** (or **outer face**) $J = \langle n, loc_J, Y\rangle$. Here V, E and $ctrl$ are finite sets of nodes, edges, and a control map $ctrl : V \to \mathcal{K}$, exactly as for link graphs.

The fourth component $G^{\mathbf{P}} = (V, ctrl, prnt) : m \to n$ is a place graph, while the fifth $G^{\mathbf{L}} = (V, E, ctrl, link) : X \to Y$ is a link graph.

We require that G adheres to the **scope rule** below.

Definition 3.A.6 (scope rule)**.** Let the **binders** of G be the binding ports of nodes in V and the local names of its outer face J.

If p is a binder located at a node or root w, then for all peers p' of p, $loc(p') = w'$ must imply $w' = prnt^k_{G^{\mathbf{P}}}(w)$, for some $k > 0$.

We say that a link is **bound** if it contains a binder, otherwise **free**. As usual, we extend this terminology to the points in the link. A binding bigraph $G : I \to J$ is said to be **free** if its outer face J is global, i.e., the image of loc_J is $\perp$.

A binding bigraph G is given by its underlying place $G^{\mathbf{P}}$ and link graph $G^{\mathbf{L}}$ and its binding interfaces I and J. We write $G = \langle G^{\mathbf{P}}, G^{\mathbf{L}}\rangle : I \to J$. We shall sometimes use a variant of the 5-tuple notation where we inline the components unique to the place graph and link graph components, i.e., $G = (V, E, ctrl, prnt, link) : I \to J$.

We define a notation for the underlying set of vectors of names: Given a vector of disjoint name sets $\vec{Y}$, $\{\vec{Y}\}$ denotes the disjoint union of the sets in the vector. Composition and tensor product of concrete binding bigraphs $G_i = \langle G_i^{\mathbf{P}}, G_i^{\mathbf{L}}\rangle : I_i \to J_i$ are given by composition and tensor product of their underlying place and link graphs, and by the tensor product of binding interfaces. We have only to explain the latter: Tensor product of binding interfaces $I_i = \langle m_i, \vec{X}_i, X_i\rangle$ is $I_0 \otimes I_1 \stackrel{\text{def}}{=} \langle m_0 + m_1, \vec{X}_0\vec{X}_1, \{\vec{X}_0\} \uplus \{\vec{X}_1\}\rangle$ (letting juxtaposition denote vector concatenation), and is defined when the name sets are disjoint. Hence, if the bigraphs above have disjoint node and edge sets, $G_1 \circ G_0 \stackrel{\text{def}}{=} \langle G_1^{\mathbf{P}} \circ G_0^{\mathbf{P}}, G_1^{\mathbf{L}} \circ G_0^{\mathbf{L}}\rangle : I_0 \to J_1$ is defined

if $I_1 = J_0$; and $G_1 \otimes G_0 \stackrel{\text{def}}{=} \langle G_1^{\mathbf{P}} \otimes G_0^{\mathbf{P}}, G_1^{\mathbf{L}} \otimes G_0^{\mathbf{L}} \rangle \ : \ I_0 \otimes I_1 \ \rightarrow \ J_0 \otimes J_1$ if the tensor products of the interfaces are defined. (See [JM04, Chapter 11] for more details.)

The identity for composition is given by a pairing of the identities for composition for place graphs and link graphs. If $I = \langle m, loc, X \rangle$ then
$\mathsf{id}_I \stackrel{\text{def}}{=} \langle \mathsf{id}_m, \mathsf{id}_X \rangle : I \rightarrow I$.

We shall use the following notation for iterated tensor product:
$\bigotimes_{i<n} P_i = P_0 \otimes P_1 \otimes \cdots \otimes P_{n-1}$. The identity for tensor is id_ϵ; thus, an iterated tensor product $P_0 \otimes \ldots \otimes P_{n-1}$ equals id_ϵ in case $n = 0$. Composition binds tighter than tensor product, and abstraction $(Y)P$ and $\bigotimes$ binds as far right as possible.

We say that two concrete binding bigraphs G_0 and G_1 are **lean-support equivalent**, denoted $G_0 \eqslantless G_1$ iff they differ only by a bijection between their nodes and their non-idle edges; idle edges are disregarded entirely.

Abstract binding bigraphs are $\eqslantless$-equivalence classes of concrete binding bigraphs. Composition, tensor and identity of abstract binding bigraphs are given by composition, tensor and identity of the underlying concrete bigraphs. Taking interfaces as objects and abstract binding bigraphs as morphisms we have a **category of binding bigraphs**. Finally, a **ground** bigraph is a bigraph with inner face ϵ. We shall also refer to such a bigraph as an **agent**. A bigraph $G : I \rightarrow J$ is called **prime**, if I is local and J is prime.

We shall need to consider and distinguish several forms of **discreteness**, which we define below.

Definition 3.A.7 (Variants of discreteness)**.**

- We say that a bigraph is **discrete** iff every free link is an outer name and has exactly one point.
- A bigraph is **name discrete** iff it is discrete and every bound link is either an edge, or (if it is an outer name) has exactly one point.

Note that name-discrete implies discrete. Name-discreteness is defined to impose exactly the same level of constraints on local and global linkage upon names. We utilize this in the normal form we define. Discreteness and name-discreteness share several nice properties.

Lemma 3.A.8. *If A and B are discrete, then $A \otimes B$, $(Y)A$, and AB are also discrete. The same holds for name-discrete bigraphs A and B.*

Chapter 4

An Inductive Characterization of Matching in Binding Bigraphs

Abstract

We analyze the matching problem for bigraphs. In particular, we present a sound and complete inductive characterization of matching in bigraphs with binding. Our results yield a specification for a provably correct matching algorithm, as needed by our prototype tool implementing bigraphical reactive systems.

Preface This chapter contains an extended and revised version of the extended abstract "Matching of Bigraphs" [BDGM07] presented at the GT-VC workshop 2006. The paper was co-authored with Lars Birkedal and Arne J. Glenstrup from the IT University of Copenhagen, and with Robin Milner from the University of Cambridge, UK.

4.1 Introduction

Over the last decade, a theory of bigraphical reactive systems has been developed [JM04, Mil05, Mil06b]. Bigraphical reactive systems (BRSs) provide a graphical model of computation in which both locality and connectivity are prominent. In essence, 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. Bigraphs can be reconfigured by means of *reaction rules*. Loosely speaking, a *bigraphical reactive system* consists of a set of bigraphs and a set of reaction rules, which can be used to reconfigure the set of bigraphs. BRSs have been developed with principally two aims in mind: (1) to be able to model directly important aspects of ubiquitous systems by focusing on mobile connectivity and mobile locality, and (2) to provide a unification of existing theories by developing

a general theory, in which many existing calculi for concurrency and mobility may be represented, with a uniform behavioural theory. The latter is achieved by representing the dynamics of bigraphs by an abstract definition of reaction rules from which a labelled transition system may be derived in such a way that an associated bisimulation relation is a congruence relation. The unification has recovered existing behavioural theories for the π-calculus [JM04], the ambient calculus [Jen06], and has contributed to that for Petri nets [LM04]. Thus the evaluation of the second aim has so far been encouraging. Birkedal et al. has begun to adress the first aim, in particular, to show how to give bigraphical models of context-aware systems [BDE$^+$06].

As suggested and argued in [JM04, BDE$^+$06, BBD$^+$06] it would be very useful to have an implementation of the dynamics of bigraphical reactive systems to allow experimentation and simulation. In the Bigraphical Programming Languages research project at the IT University, we have been working towards such an implementation. The core problem of implementing the dynamics of bigraphical reactive systems is the *matching problem*, that is, to determine for a given bigraph and reaction rule whether and how the reaction rule can be applied to rewrite the bigraph. The topic of the present paper is to analyze the matching problem. We report on an implementation based on the work presented here elsewhere [GDBH07].

In Figure 4.1 we show several bigraphs. Consider the bigraph named a. It is intended to model two buildings, one belonging to a corporation and one belonging to a consultancy group. Inside the buildings are laptops with data nested inside folders. The nesting structure depicts the place graph. Links are used to name the buildings and, moreover, to model the association of folders to network channels. The laptop shown in the middle is intended to belong to a consultant working for the corporation — the consultant has a folder, containing some data, which is directly connected to a laptop in the consultancy (the link shown to the left) and a folder with a connection over the corporate backbone to another laptop in the corporation (the link shown to the right). There are two kinds of network channels in this example; those local to a building (i.e., over a building backbone) and global channels (presumably across the internet). The fact that folders are connected over the corporation backbone is expressed by linking those folders to a so-called binding port on the corporation building, indicated by the circle. A binding port of a node imposes a scoping discipline to ensure that links connected to the port will be constrained to connections within the node.

The abstract semantic definition of matching, as defined in the theory of bigraphs [JM04], is roughly as follows (omitting many details): Given a reaction rule (R, R') and a bigraph a, if $a = C \circ (R \otimes \mathsf{id}_Z) \circ d$, then it can be rewritten to $C \circ (R' \otimes \mathsf{id}_Z) \circ d$. We call a the *agent*. The constituents of the rule, R, the *redex*, and R', the *reactum*; and, the *context* C and *parameter* d are also bigraphs; $\circ$ is vertical composition of bigraphs, while $\otimes$ juxtaposes bigraphs horizontally. Z is a set of names exported by d. In short, we say that if the reaction rule *matches* the agent a, in the sense that a can be decomposed into a context C, redex R and a parameter d, then a can be rewritten.

Consider again the example in Figure 4.1. There is a reaction rule expressed by

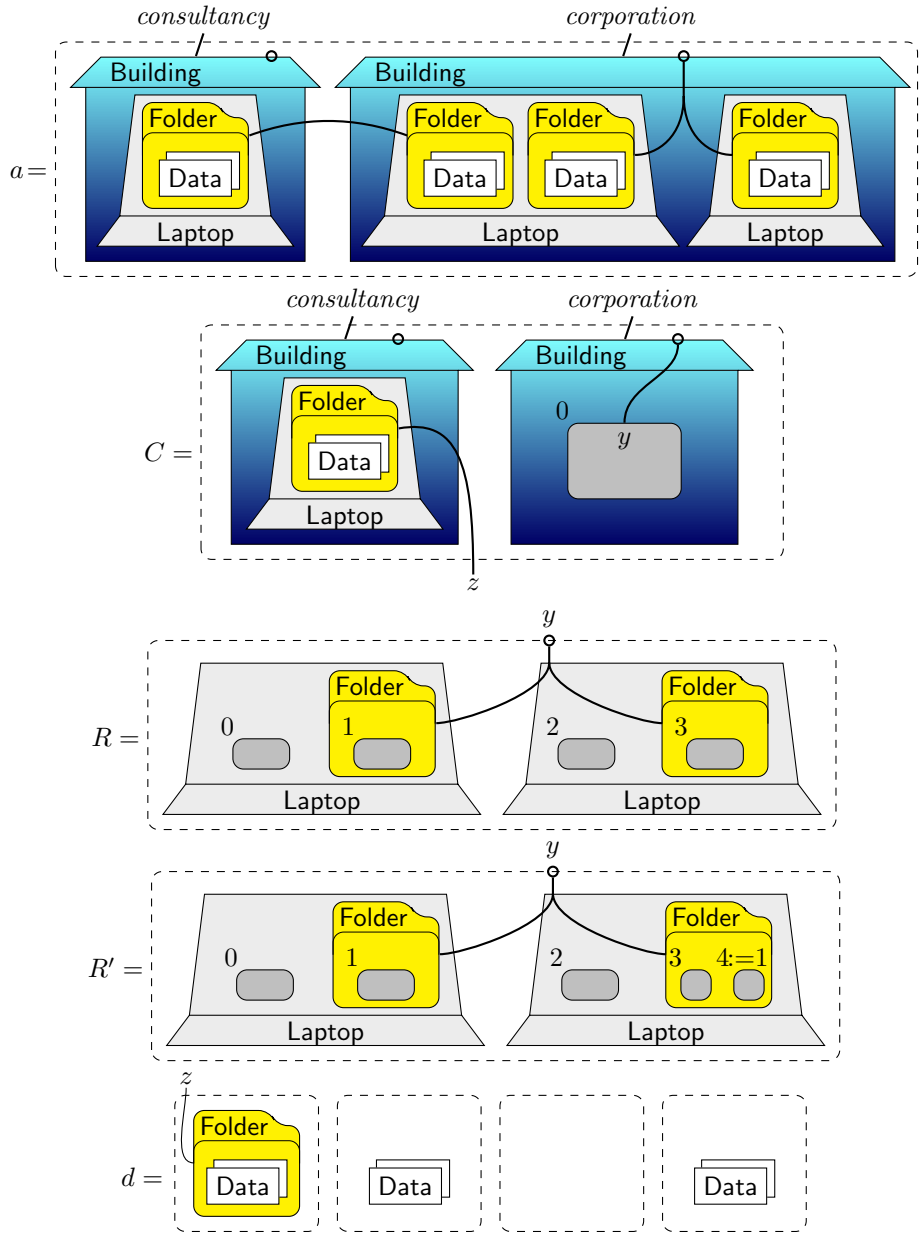


Figure 4.1: A bigraph $a = C \circ (\mathsf{id}_z \otimes R) \circ d$. Reaction rule $R \to R'$ copies data between connected folders.

the redex R and the reactum R'. The intention of the reaction rule is to allow copying of data only between folders on co-located laptops; we require also that the folders be connected over a network connection (note the link in R between the two folders), allowing that link to be possibly connected to a building backbone (expressed by linking the link to the so-called local name y). The agent a can be written as a composition of C, R and d — formally, $a = C \circ (R \otimes \mathsf{id}_z) \circ d$. Composition works by (1) plugging the roots of R and d into the holes (aka sites) of C respectively R; (2) fusing together the connections between folder and z (in d) and z and folder (in C), removing the name z in the process; and (3) fusing together the connection between the local name y and the two folders in R and the name y and the bound port in C, removing the name y in the process. Note the use of id_z in the composition $a = C \circ (R \otimes \mathsf{id}_z) \circ d$; it allows a name z from the parameter d to be passed around the redex and be attached to something in the context C. The reactum R' contains a copy of the site numbered 1 in R, expressing that data is copied between the shared folders. The sites numbered 0 and 2 in R allow the reaction rule to apply also when the laptops contain other folders than the two that are connected. Thus a can be rewritten using the reaction rule to another agent a' like a but with two data items in the rightmost laptop (the agent a' is not shown in Figure 4.1).

In the present paper we provide an *inductive characterization* of when $a = C \circ (R \otimes \mathsf{id}_Z) \circ d$ holds, by induction on the structure of a, R, C and d. It is a precise characterization in the sense that it is both sound and complete with respect to the abstract definition. This provides a detailed analysis of the matching problem, and give a specification for developing and *proving correct* an actual matching algorithm (which, given a and R, must find C, d, and Z such that $a = C \circ (R \otimes \mathsf{id}_Z) \circ d$ holds). We further include a discussion of how one may derive matching algorithms directly from our inductive characterization.

Our inductive characterization is based on normal form theorems for binding bigraphs [DB06], which express how general bigraphs may be decomposed into a composition of simpler graphs. The normal form theorems and also the inductive characterization we present here is based on *discrete* decompositions of bigraphs. Discrete bigraphs are bigraphs with only a simple form of linkage. To a large extent, this allows us to analyze matching of a general bigraph by considering its link graph and place graph separately.

Of course, the matching problem is closely related to the NP-complete graph embedding problem. Thus we analyze the embedding problem for a restricted class of graphs, and our inductive characterization makes good use of the algebraic presentation of such graphs [Mil05, DB06]. Importantly, by providing an abstract characterization founded in well-established theory for bigraphs, we expect to be able to combine or adapt more easily our approach to theory and techniques being developed for bigraphs; for instance, sortings (simple type disciplines) on bigraphs could be a source of early search elimination [BDH06].

The remainder of this paper is organized as follows: In Section 4.2 we give an informal description of binding bigraphs. The main contributions of this paper are in Section 4.3, where we present our inductive characterization of matching, and

in the Appendix, where we give the proof of completeness of the characterization. To illustrate how the matching rules work together, in Section 4.4, we provide an inference tree for inferring the match in the example in Figure 4.1. Section 4.5 discusses how the inductive characterization yields a specification for a provably correct algorithm for matching. In the final sections we discuss related and future work, and conclude.

An extended abstract of this paper was presented at the GT-VC 2006 workshop. This extended and revised version fixes a few errors in the earlier presentations, provides more explanations and examples, and notably includes extensive details for the proof of completeness of the characterization and supporting lemmas, including a selfcontained section on the algebraic properties of wirings and parallel product.

4.2 Binding Bigraphs

In the following section, we present binding bigraphs fairly thoroughly, but we leave out formal details inessential for the present paper; for a more complete presentation, see [JM04] or [DB06].

4.2.1 Concrete Bigraphs

A concrete binding bigraph G consists of a *place graph* $G^{\mathbf{P}}$ and a *link graph* $G^{\mathbf{L}}$. The place graph is an ordered list of trees indicating *location*, with roots $r_0, \ldots, r_n$, nodes $v_0, \ldots, v_k$, and a number of special leaves $s_0, \ldots, s_m$ called *sites*, while the link graph is a general graph over the node set $v_0, \ldots, v_k$ extended with *inner names* $x_0, \ldots, x_l$, and equipped with hyperedges, indicating *connectivity*.

We usually illustrate the place graph by nesting nodes, as shown in the upper part of Figure 4.2 (ignore for now the interfaces denoted by " $: \cdot \to \cdot$"). A *link* is a

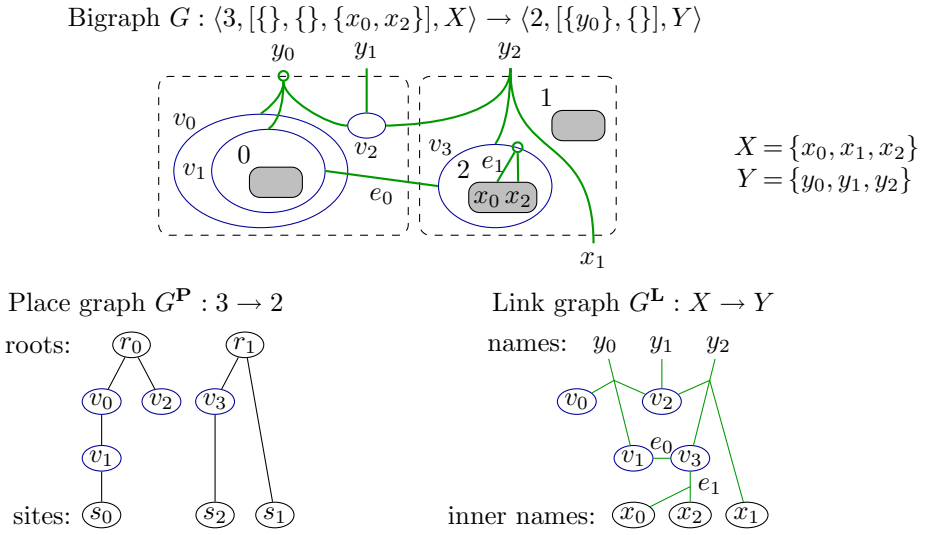


Figure 4.2: Example bigraph illustrated by nesting and as place and link graph.

hyper edge of the link graph, either an internal *edge* e_0 or a *name* y. Those that are names are called *open*, edges are called *closed* links. Names and inner names can be *global* or *local*, the latter being located at a specific root or site, respectively. In Figure 4.2, y_0 is located at r_0, indicated by a small ring, and x_0 and x_2 are located at s_2, indicated by writing them within the site. Global names like y_1 and y_2 are drawn anywhere at the top, while global inner names like x_1 are drawn anywhere at the bottom. A link, including internal edges like e_1 in the figure, can be located with one *binder* (the ring), in which case it is a *bound link,* otherwise it is *free*. However, a bound link must satisfy the *scope rule*, a simple structural requirement that all points (see below) of the link lie within its location (in the place graph), except for the binder itself. This prevents y_2 and e_0 in the example from being bound.

4.2.2 Controls

Every node v has a *control* K indicated by $v : K$, which determines a binding and free arity $K : b \to f$. In the example of Figure 4.2, we could have $v_i : K_i, i = 0, 1, 2, 3$, where $K_0 : 0 \to 1$, $K_1 : 0 \to 2$, $K_2 : 0 \to 3$, $K_3 : 1 \to 2$. The arities determine the number of bound and free *ports* of the node, to which bound and free links, respectively, are connected. Ports and inner names are collectively referred to as *points*.

4.2.3 Abstract Bigraphs

While concrete bigraphs with named nodes and internal edges are the basis of bigraph theory [JM04], our prime interest is in *abstract bigraphs,* equivalence classes of concrete bigraphs that differ only in the names of nodes and internal edges[1]. Abstract bigraphs are illustrated with their node controls, as shown in Figure 4.1 with Building, Laptop, etc. In what follows, "bigraph" will thus mean "abstract bigraph."

4.2.4 Interfaces

Every bigraph G has two *interfaces* I and J, written $G : I \to J$, where I is the *inner face* and J the *outer face*. An interface is a triple $\langle m, \vec{X}, X \rangle$, where m is the *width* (the number of sites or roots), X the entire set of local and global names, and $\vec{X}$ are disjoint subsets of X indicating the locations of each local name, cf. Figure 4.2. We let $\epsilon = \langle 0, [\,], \{\} \rangle$; when $m = 1$ the interface is *prime*, and if all $x \in X$ are located by $\vec{X}$, the interface is *local*. As in [Mil04] we write $G : \to J$ or $G : I \to$ for $G : I \to J$ when we are not concerned about about I or J, respectively.

A bigraph $G : I \to J$ is called *ground*, or an *agent*, if $I = \epsilon$, *prime* if I is local and J prime, and a *wiring* if $m = n = 0$, where m and n are the widths of I and J, respectively. For $I = \langle m, \vec{X}, X \rangle$, bigraph $\mathsf{id}_I : I \to I$ consists of m roots, each root r_i containing just one site s_i, and a link graph linking each inner name $x \in X$ to name x.

[1] Formally, we also disregard *idle* edges: edges not connected to anything.

4.2.5 Discrete and Regular Bigraphs

We say that a bigraph is *discrete* iff every free link is a name and has exactly one point. The virtue of discrete bigraphs is that any connectivity by internal edges must be bound, and node ports can be accessed individually by the names of the outer face. In Figure 4.1, only R, R' and d are discrete, because the free internal edges of a and C have two points. Further, a bigraph is *name-discrete* iff it is discrete and every bound link is either an edge, or (if it is an name in the outer face) has exactly one point. Note that name-discrete implies discrete.

A bigraph is *regular* if, for all nodes v and sites i, j, k with $i \leq j \leq k$, if i and k are descendants of v, then j is also a descendant of v. Further, for roots $r_{i'}$ and $r_{j'}$, and all sites i and j where i is a descendant of $r_{i'}$ and j of $r_{j'}$, if $i \leq j$ then $i' \leq j'$. The bigraphs in the figures are all regular, the permutation in Table 4.1 is not. The virtue of regular bigraphs is that permutations can be avoided when composing them from basic bigraphs.

4.2.6 Tensor Product, Parallel Product, and Composition

For bigraphs G_1 and G_2 that share no names or inner names, we can make the *tensor product* $G_1 \otimes G_2$ by juxtaposing their place graphs, constructing the union of their link graphs, and increasing the indexes of sites in G_2 by the number of sites of G_1. For instance, bigraph d of Figure 4.1 is a tensor product of four primes. We write $\bigotimes_i^n G_i$ for the iterated tensor $G_0 \otimes \cdots \otimes G_{n-1}$, which, in case $n = 0$, is id_ϵ.

The *parallel product* $G_1 \,||\, G_2$ is like the tensor product, except global names can be shared: if y is shared, all points of y in G_1 and G_2 become the points of y in $G_1 \,||\, G_2$.

We can *compose* bigraphs $G_2 : I \to I'$ and $G_1 : I' \to J$, yielding bigraph $G_1 \circ G_2 : I \to J$, by plugging the sites of G_1 with the roots of G_2, eliminating both, and connecting names of G_2 with inner names of G_1—as in Figure 4.1, where $a = C \circ (\mathsf{id}_z \otimes R) \circ d$. In the following, we will omit the '$\circ$', and simply write $G_1 G_2$ for composition, letting it bind tighter than tensor product.

4.2.7 Active, Passive and Atomic Controls

In addition to arity, each control is assigned a *kind*, either atomic, active or passive, and describe nodes according to their control kinds. We require that atomic nodes contain no nodes except sites; any site being a descendant of a passive node is *passive*, otherwise it is *active*. If all sites of a bigraph G are active, G is *active*.

For Figure 4.1 we could have $\mathsf{Data} : \mathsf{atomic}(0 \to 0)$, $\mathsf{Folder} : \mathsf{passive}(0 \to 1)$, $\mathsf{Laptop} : \mathsf{active}(0 \to 0)$, $\mathsf{Building} : \mathsf{active}(1 \to 1)$.

4.2.8 Bigraphical Reactive Systems

Bigraphs in themselves model two essential parts of context: locality and connectivity. To model also *dynamics,* we introduce *bigraphical reactive systems* (BRS) as a collection of *rules.* Each rule $R \to_\varrho R'$ consists of a regular *redex* $R : I \to J$, a

reactum $R' : I' \to J$, and an *instantiation* ϱ, mapping each site of R' to a site of R. Interfaces $I = \langle m, \vec{X}, X\rangle$ and $I' = \langle m', \vec{X'}, X'\rangle$ must be local, and are essentially related by $X'_i = X_{\varrho(i)}$.[2] We illustrate ϱ by a '$i := j$', as shown in Figure 4.1, whenever $\varrho(i) = j \neq i$. Given an instantiation ϱ and a discrete bigraph $d = d_0 \otimes \cdots \otimes d_k$ with prime d_i's, we let $\varrho(d) = d_{\varrho(0)} \otimes \cdots \otimes d_{\varrho(k)}$, allowing copying, discarding and reordering parts of d.

Given an agent a, a *match* of redex R is a decomposition $a = C(\mathsf{id}_Z \otimes R)d$, with active context C, discrete parameter d and its global names Z. Dynamics is achieved by transforming a into a new agent $a' = C(\mathsf{id}_Z \otimes R')d'$, where $d' = \varrho(d)$—an example is shown in Figure 4.1. This definition of a match is as in [JM04], except that we here also require R to be regular. The restriction to regular redexes R, which simplifies the inductive characterization, does not limit the set of possible reactions, because sites in R and R' can be renumbered to render R regular.

4.2.9 Notation, Basic Bigraphs, and Abstraction

In the sequel, we will use the following notation: $\uplus$ denotes union of sets required to be disjoint; we write $\{\vec{Y}\}$ for $Y_0 \uplus \cdots \uplus Y_{n-1}$ when $\vec{Y} = Y_0, \ldots, Y_{n-1}$, and similarly $\{\vec{y}\}$ for $\{y_0, \ldots, y_{n-1}\}$. For interfaces, we write n to mean $\langle n, [\emptyset, \ldots, \emptyset], \emptyset\rangle$, X to mean $\langle 0, [\,], X\rangle$, $\langle X\rangle$ to mean $\langle 1, [\{\}], X\rangle$, (X) to mean $\langle 1, [X], X\rangle$, and $(\vec{X})$ to mean $\langle n, \vec{X}, \{\vec{X}\}\rangle$, when the length of $\vec{X}$ is n.

Any bigraph can be constructed by applying composition, tensor product and *abstraction* to identities (on all interfaces) and a minimal set of basic bigraphs [DB06]. Given a prime P, the abstraction operation localizes a subset of its outer names. The scope rule is necessarily respected since the inner face of a prime P is required to be local, so all points of P are located within its root. The abstraction operator is denoted by $(\cdot)\cdot$ and reaches as far right as possible.

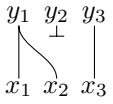
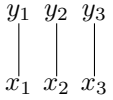
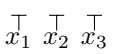
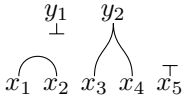
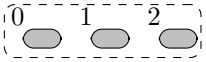
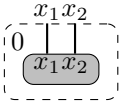
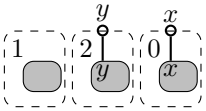
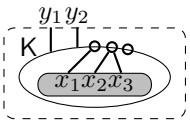
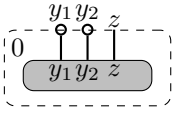
We illustrate the basic bigraphs and abstraction in Table 4.1. Substitutions introduce open linking and closures create edges. Renamings are one-one substitutions, while wirings ω range over all expressions built by composition and tensor from substitutions and closures. A *merge* bigraph merges sites into a single root, a concretion maps local inner names to global outer names, while a permutation serves to produce any ordering of sites inside roots. The ion introduces nodes, mapping free ports to global outer names and bound ports to sets of local inner names.

The resulting inductive language for building bigraphs is fairly heavy, but it is easy to derive more sugared languages, resembling closely standard notation used for mobile process calculi, based on these basic bigraphs and operators. In this paper, we shall not be concerned much with concrete notation, though; we refer the interested reader to [BDE$^+$06] or the forthcoming tutorial book on bigraphs by Milner.

We shall only use a few basic conventions for shortening expressions. For a

[2] When copying sites with local names, one has to also rename local names of copied sites, which makes the relation slightly less straightforward. We elide the details here, as they are not central to matching, and refer the reader to [JM04, Chapter 12] for the full formal details.

Table 4.1: Basic bigraphs, abstraction, and metavariables ranging over bigraphs.

	Notation	Example	
Substitution σ	$\vec{y}/\vec{X} : X \to Y$	$[y_1, y_2, y_3]/[\{x_1, x_2\}, \{\}, \{x_3\}] =$	
Renaming α, β	$\vec{y}/\vec{x} : X \to Y$	$[y_1, y_2, y_3]/[x_1, x_2, x_3] =$	
Closure	$/X : X \to \{\}$	$/\{x_1, x_2, x_3\} =$	
Wiring ω	$\omega : X \to Y$	$([y_1, y_2]/[y_1, y_2] \otimes /\{z_1, z_2\})$ $[y_1, z_1, y_2, z_2]\,/$ $[\{\}, \{x_1, x_2\}, \{x_3, x_4\}, \{x_5\}] =$	
Merge	$merge_n : n \to 1$	$merge_3 =$	
Concretion	$\ulcorner X \urcorner : (X) \to \langle X \rangle$	$\ulcorner \{x_1, x_2\} \urcorner =$	
Permutation $\pi_{\vec{X}}, \pi$	$\{i \mapsto j, \ldots\}_{\vec{X}} :$ $(\vec{X}) \;\to\; (\pi(\vec{X}))$	$\{0 \mapsto 2, 1 \mapsto 0, 2 \mapsto 1\}_{[\{x\},\emptyset,\{y\}]} =$	
Ion	$K_{\vec{y}(\vec{X})} : (\{\vec{X}\}) \to \langle \{\vec{y}\} \rangle$	$\mathsf{K}_{[y_1,y_2]([\{x_1\},\{x_2,x_3\},\{\}])} =$	
Abstraction	$(Y)P : I \to \langle 1, [Y], Z \uplus Y \rangle$	$(\{y_1, y_2\})\ulcorner \{y_1, y_2, z\} \urcorner =$	

renaming $\alpha : X \to Y$, we write $\ulcorner\alpha\urcorner$ to mean $(\alpha \otimes \mathsf{id}_1)\ulcorner X\urcorner$, and when $\sigma : U \to Y$, we let $\widehat{\sigma} = (Y)(\sigma \otimes \mathsf{id}_1)\ulcorner U\urcorner$. We write substitutions $\vec{y}/[\emptyset, \ldots, \emptyset] : \epsilon \to Y$ as Y. For permutations, when used in any context, $\pi_{\vec{X}} G$ or $G\pi_{\vec{X}}$, $\vec{X}$ is given entirely by the interface of G; hence, we shall typically elide the names of $\pi_{\vec{X}}$ and write only π. Note that $[]/[] = /\emptyset = \pi_0 = \mathsf{id}_\epsilon$ and $merge_1 = \ulcorner\emptyset\urcorner = \pi_1 = \mathsf{id}_1$, where π_i is the nameless permutation of width i.

To conclude this section, we illustrate the basic bigraphs and operations by showing expressions for some of the bigraphs in the previous examples. The bigraph of Figure 4.2 can be expressed as

$$\begin{array}{rcl} G & = & (\omega \otimes (\{y_0\})(y_0/Y_0 \otimes \mathsf{id}_1)\ulcorner Y_0\urcorner))\,(((Y_0)P_1) \otimes P_2 \otimes y_2/x_1)\,, \text{ where} \\ \omega & = & (/e_0 \otimes \mathsf{id}_{\{y_1,y_2\}})[y_1, y_2, e_0]/[\{y_1\}, \{y_2, y_2', y_2''\}, \{e_0, e_0'\}], \quad Y_0 = \{y_0, y_0', y_0''\} \\ P_1 & = & (\mathsf{id}_{\{y_0,y_1,y_2',e_0\}} \otimes merge_2)\left((\mathsf{id}_{\{y_0,e_0\}} \otimes K_{0[y_0']})K_{1[y_0,e_0]} \otimes K_{2[y_0'',y_1,y_2']}\, merge_0\right) \\ P_2 & = & (\mathsf{id}_{\{e_0',y_2''\}} \otimes merge_2)(K_{3[e_0',y_2'']([\{x_0,x_2\}])} \otimes \ulcorner\emptyset\urcorner), \end{array}$$

and for Figure 4.1 we have $a = (\mathsf{id}_{\{consultancy,corporation\}} \otimes /z)\,(p_1 \,||\, p_2)$, where

$$\begin{array}{rcl} p_1 & = & (\mathsf{id}_z \otimes \mathsf{Building}_{[consultancy]([\{\}])}\mathsf{Laptop})\mathsf{Folder}_{[z]}\mathsf{Data}\, merge_0 \\ p_2 & = & (\mathsf{id}_z \otimes \mathsf{Building}_{[corporation]([\{y_1,y_2\}])})(\{y_1, y_2\})(\mathsf{id}_{\{z,y_1,y_2\}} \otimes merge_2)\,(p_2' \otimes p_2'') \\ p_2' & = & (\mathsf{id}_{\{z,y_1\}} \otimes \mathsf{Laptop}\, merge_2)(\mathsf{Folder}_{[z]}\mathsf{Data} merge_0 \otimes \mathsf{Folder}_{[y_1]}\mathsf{Data}\, merge_0) \\ p_2'' & = & (\mathsf{id}_{y_2} \otimes \mathsf{Laptop})\mathsf{Folder}_{[y_2]}\mathsf{Data}\, merge_0 \end{array}$$

4.3 Inductive Characterization of Matching

In this section we present our inductive characterization of matching. To ease the presentation we shall disregard the requirement that the context in a match must be active (it is straightforward to extend the presentation to include that requirement). To simplify notation we shall write id for local identity bigraphs, without a subscript showing the interface, when it is clear from the context what interface is intended. Furthermore, we use the name *molecule* for a prime with just one outermost node.

4.3.1 Discrete Decomposition

We base our characterization on discrete decomposition of bigraphs, which separates global (or free) wiring from the place graph and local wiring. The following proposition expresses how any bigraph may be decomposed into a global wiring ω, and a discrete bigraph D (cf. Section 4.2.5).

Proposition 4.3.1 (Discrete decomposition)**.** *Any bigraph G can be decomposed into a composition of the following form*

$$G = (\omega \otimes \mathsf{id})(D \otimes \mathsf{id}_Y),$$

where D is discrete and with local innerface. Any other decomposition of G on this form takes the form $G = (\omega' \otimes \mathsf{id})(D' \otimes \mathsf{id}_Y)$, where $\omega' = \omega(\alpha \otimes \mathsf{id}_Y)$ and $D' = (\alpha^{-1} \otimes \mathsf{id})D$, for suitable α.

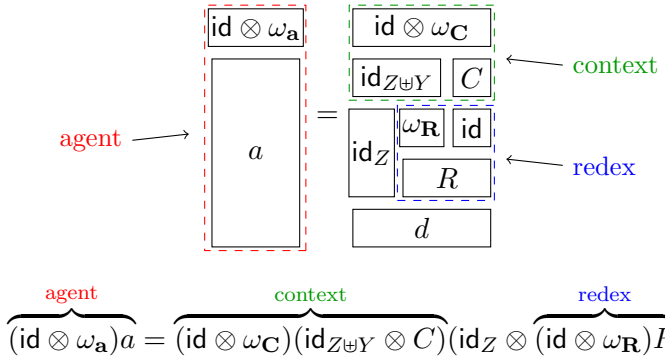


Figure 4.3: Illustrating valid matching sentences

The proof follows easily from the normal form theorem in [DB06], which also gives a number of normal forms for name-discrete, prime and molecular bigraphs. We shall not go into detail with these forms in this section, but as they are central in proving completeness of the characterization, we shall return to them in the Appendix.

4.3.2 Matching Sentences

We now define matching sentences and rules for deriving valid matching sentences.

Definition 4.3.2 (Matching sentence)**.** A *matching sentence* is a 7-place relation among wirings and bigraphs, written $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \hookrightarrow C, d$, where $\omega_{\mathbf{a}}$, $\omega_{\mathbf{R}}$, $\omega_{\mathbf{C}}$ are wirings, and a, R, C, d are discrete bigraphs, R and C have local inner faces, and R is regular.

Definition 4.3.3 (Valid matching sentence)**.** A matching sentence $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \hookrightarrow C, d$, where $\omega_{\mathbf{R}} :\rightarrow Y$, and d has global outer names Z, is *valid*, denoted $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash a, R \hookrightarrow C, d$, iff

$$(\mathsf{id} \otimes \omega_{\mathbf{a}})a \quad = \quad (\mathsf{id} \otimes \omega_{\mathbf{C}})(C \otimes \mathsf{id}_Y \otimes \mathsf{id}_Z)\,(\mathsf{id}_Z \otimes (\mathsf{id} \otimes \omega_{\mathbf{R}})\,R)\,d.$$

where unqualified identities are local and determined from their context.

Note that for a valid sentence $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \hookrightarrow C, d$, if we let $a' = (\mathsf{id} \otimes \omega_{\mathbf{a}})a$, $C' = (\mathsf{id} \otimes \omega_{\mathbf{C}})(C \otimes \mathsf{id}_Y \otimes \mathsf{id}_Z)$, and $R' = (\mathsf{id} \otimes \omega_{\mathbf{R}})R$, then $a' = C'(R' \otimes \mathsf{id}_Z)d$. Conversely, if, for general a', C', R', d we have a match $a' = C'(R' \otimes \mathsf{id}_Z)d$, then by Proposition 4.3.1, we can decompose a', C', and R' and obtain a corresponding valid sentence. Thus, valid sentences precisely capture the abstract definition of matching.

Rearranging a few identities, we can illustrate the discrete decomposition of the agent, context, and redex in a valid matching sentence schematically as in Figure 4.3. We draw bigraph composition as vertical composition, and tensor product as horizontal juxtaposition.

$$\text{PERM}\ \frac{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, \bigotimes_i^m P_{\pi^{-1}(i)} \hookrightarrow C, (\overline{\pi} \otimes \mathsf{id})d}{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, \bigotimes_i^m P_i \hookrightarrow C\pi, d}$$

$$\text{PAR}\ \frac{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \mid\mid \omega \vdash a, R \hookrightarrow C, d \qquad \omega_{\mathbf{b}}, \omega_{\mathbf{S}}, \omega_{\mathbf{D}} \mid\mid \omega \vdash b, S \hookrightarrow D, e}{\omega_{\mathbf{a}} \mid\mid \omega_{\mathbf{b}}, \omega_{\mathbf{R}} \mid\mid \omega_{\mathbf{S}}, \omega_{\mathbf{C}} \mid\mid \omega_{\mathbf{D}} \mid\mid \omega \vdash a \otimes b, R \otimes S \hookrightarrow C \otimes D, d \otimes e}$$

$$\text{LSUB}\ \frac{\sigma_{\mathbf{a}} \otimes \omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \sigma_{\mathbf{C}} \otimes \omega_{\mathbf{C}} \vdash p, R \hookrightarrow P, d}{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash (\widehat{\sigma_{\mathbf{a}}} \otimes \mathsf{id})(Z)p, R \hookrightarrow (\widehat{\sigma_{\mathbf{C}}} \otimes \mathsf{id})(U)P, d}, \begin{array}{l}\sigma_{\mathbf{a}} : Z \to \\ \sigma_{\mathbf{C}} : U \to\end{array}$$

$$\text{MERGE}\ \frac{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \hookrightarrow C, d}{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash (\mathit{merge} \otimes \mathsf{id})a, R \hookrightarrow (\mathit{merge} \otimes \mathsf{id})C, d},\ a \text{ global}$$

$$\text{ION}\ \frac{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash ((\vec{v})/(\vec{X}) \otimes \mathsf{id})p, R \hookrightarrow ((\vec{v})/(\vec{Z}) \otimes \mathsf{id})P, d}{\sigma \mid\mid \omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \sigma\alpha \mid\mid \omega_{\mathbf{C}} \vdash (K_{\vec{y}(\vec{X})} \otimes \mathsf{id})p, R \hookrightarrow (K_{\vec{u}(\vec{Z})} \otimes \mathsf{id})P, d}, \begin{array}{l}\alpha = \vec{y}/\vec{u} \\ \sigma : \{\vec{y}\} \to\end{array}$$

$$\text{SWITCH}\ \frac{\omega_{\mathbf{a}}, \mathsf{id}_\epsilon, \omega_{\mathbf{C}}(\alpha\sigma \otimes \omega_{\mathbf{R}} \otimes \mathsf{id}_Z) \vdash p, \mathsf{id} \hookrightarrow P, d}{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash p, (\widehat{\sigma} \otimes \mathsf{id})(W)P \hookrightarrow \ulcorner \alpha \urcorner, d}, \begin{array}{l}\sigma : W \to U \\ d : \langle m, \vec{X}, X \uplus Z \rangle\end{array}$$

$$\text{CLOSE}\ \frac{\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \mathsf{id}_Y \otimes \sigma_{\mathbf{C}} \vdash a, R \hookrightarrow C, d}{(\mathsf{id} \otimes /(Y \uplus X))\sigma_{\mathbf{a}}, (\mathsf{id} \otimes /Y)\sigma_{\mathbf{R}}, (\mathsf{id} \otimes /X)\sigma_{\mathbf{C}} \vdash a, R \hookrightarrow C, d}, \begin{array}{l}\sigma_{\mathbf{C}} :\to Z \uplus X \\ \sigma_{\mathbf{R}} :\to U \uplus Y\end{array}$$

Figure 4.4: Rules for matching binding bigraphs

4.3.3 Rules for Matching

In Figure 4.4 and Figure 4.5, we present a set of rules and axioms for inferring matching sentences. In PAR we require further that the tensor products of all discrete components be defined. Also, in the premises of the rules PERM and ION, and in the conclusion of rules MERGE, ION, and SWITCH we require the id's to have width 0 (hence be link graph identities). This determines them entirely from the context.

We now explain each of the rules. To illustrate how the matching rules work together, in the following section (Section 4.4) we provide an inference tree for inferring the match in the example depicted in Figure 4.1.

The PERM rule simply pushes a permutation on the inside of the context through the redex, permuting the discrete primes, and producing a pushed-through permutation $\overline{\pi}$, depending on π and the innerface of the redex, as stated in the push-through lemma [DB06, Lemma 2].

$$\textsc{prime-axiom}\ \frac{\alpha : X \to U \quad \beta : V \to Z \quad \sigma : U \uplus Z \to \quad \tau : Y \to X \quad p : \langle Y \uplus Z \rangle}{\sigma(\alpha\tau \otimes \beta), \mathsf{id}_\epsilon, \sigma \vdash p, \mathsf{id}_{(X)} \hookrightarrow \ulcorner \alpha \urcorner, (X)(\tau \otimes \beta \otimes \mathsf{id}_1)p}$$

$$\textsc{wiring-axiom}\ \frac{}{y, Y, y/Y \vdash \mathsf{id}_\epsilon, \mathsf{id}_\epsilon \hookrightarrow \mathsf{id}_\epsilon, \mathsf{id}_\epsilon}$$

Figure 4.5: Axioms for matching binding bigraphs

The PAR rule explains how to match a product, given two valid matches. The two valid matches are allowed to *share* some context wiring ω, if the redices share (global) names. Figure 4.6 illustrates the conclusion of a match using PAR. The wirings are depicted above the underlying discrete bigraphs: a product of two agents a and b containing a single node; a product of two contexts C and D, which contain only a site; and, a product of two redices R and S containing a single node (for this example, the parameters are empty). As the parallel product of the redex wirings $\omega_{\mathbf{R}}$ and $\omega_{\mathbf{S}}$ already map the links from w_1 and w_2 to a shared name w, the link from w is shared wiring ω (while the links from x and y_1 are in $\omega_{\mathbf{a}} = \omega \,||\, \omega_{\mathbf{C}}$, and the links from y_2 and z are in $\omega_{\mathbf{b}} = \omega \,||\, \omega_{\mathbf{D}}$).

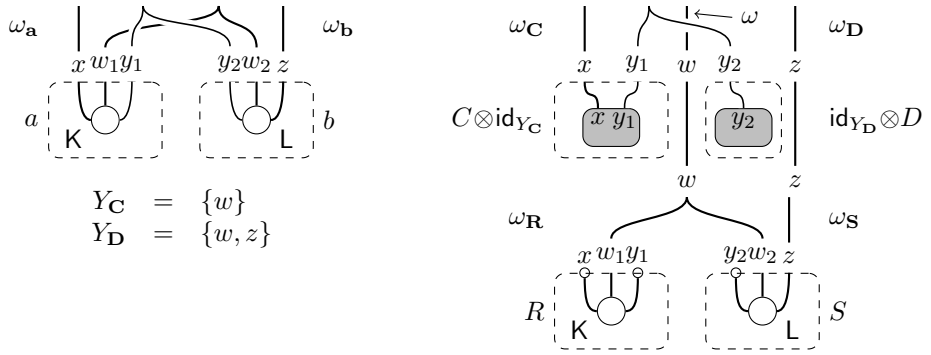


Figure 4.6: Matching a product using the PAR rule

The LSUB rule allows us to match any discrete prime by matching an underlying *free* prime with the wiring of agent and context extended with the underlying global substitutions $\sigma_{\mathbf{a}}$ and $\sigma_{\mathbf{C}}$. In other words, this rule expresses that we can match a single-rooted bigraph with *local* names by matching the corresponding free bigraph (i.e., forgetting the locality of the names).

The MERGE rule simply states that if we can match (global) bigraphs with several roots, then we can *merge* those roots into a single root, and still have a valid match.

The ION rule states intuitively that if we have two valid matches with primes in agent and context, we can compose both primes with an ion (i.e., a node with wiring) and still have a valid match. For any given match of discrete primes, we can compose with ions $K_{\vec{y}(\vec{X})}$ or $K_{\vec{u}(\vec{Z})}$, if we extend the wirings of agents and contexts

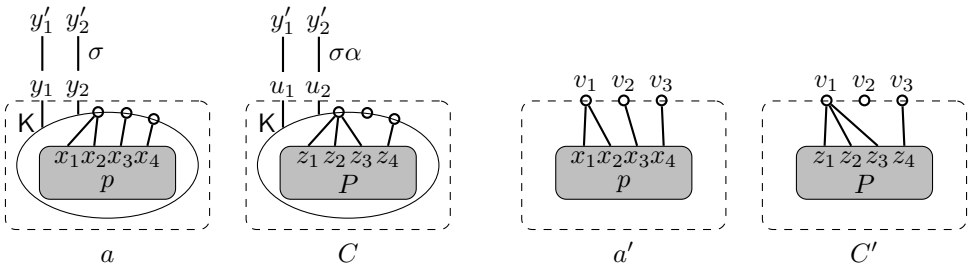


Figure 4.7: Matching an ion in the agent a with context C by matching a' with context C'

with isomorphic wiring on the outer names $\vec{y}$ and $\vec{u}$; stated in the rule by requiring that we extend with σ and $\sigma\alpha$ (where $\alpha = \vec{y}/\vec{u}$). For example, if we seek to match the agent $a = (\mathsf{id} \otimes K_{\vec{y}(\vec{X})})p$ with a context $C = (\mathsf{id} \otimes K_{\vec{u}(\vec{Z})})P$, then it suffices to consider matching of $a' = (\vec{v})/(\vec{X})p$ with a context $C' = (\vec{v})/(\vec{Z})$, as illustrated in Figure 4.7. The local linkage remaining in a' and C' is the local substitutions underlying the ions in a and C.

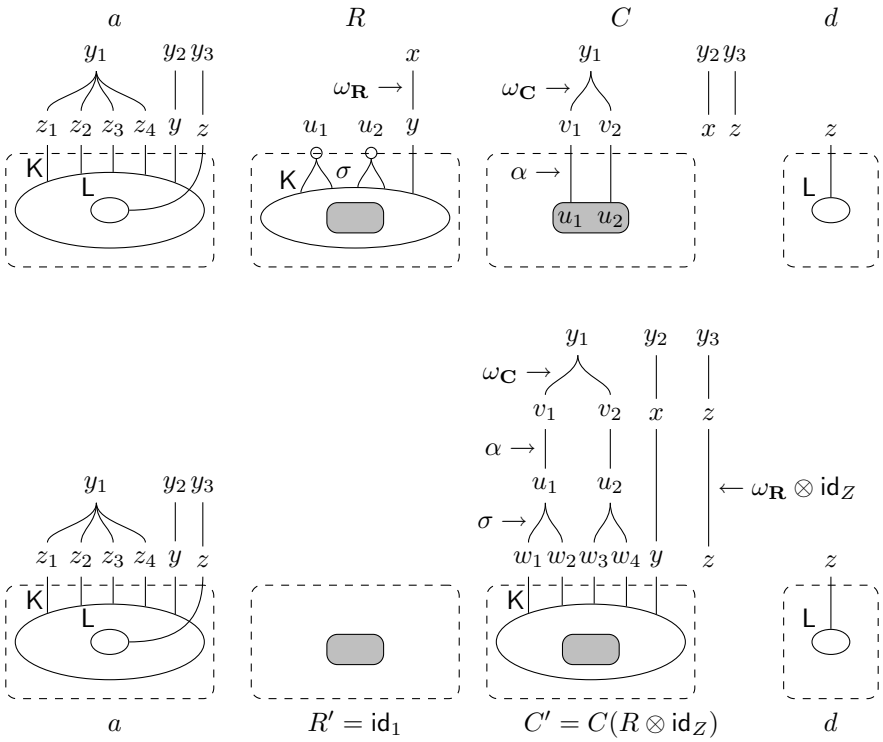


Figure 4.8: Matching $a = C(R \otimes \mathsf{id}_Z)d$ by matching $a = C'(R' \otimes \mathsf{id}_Z)d$ using the SWITCH rule.

Given an agent and considering an inference tree operationally bottom up, the rules specify how to decompose the agent while *constructing* the corresponding con-

text (cf., e.g., the ION rule). At the point where the root of the redex is matched, the SWITCH rule is applied, switching the redex into context position, so that further decomposition of the agent *checks* that the redex matches. A redex root needs to be matched when the only remainder of the context is a site, possibly with some local linkage. Thus, when inferring a match, every rule except SWITCH can be used in two modes: one where the agent and redex are given, resulting in a context and parameter; and, after a SWITCH, one where the agent and context are given, resulting in a parameter. This is reflected in the fact, that in the premiss of SWITCH, in the matching sentence the redex-position is id and the redex-wiring is empty, since we are now only concerned with checking the redex, and constructing the parameter. (Note, that this means that for $d : \langle 1, (Y), Y \uplus Z \rangle$, the unspecified id in the context-wiring is necessarily id_Z.)

In Figure 4.8, we depict an application of SWITCH. As illustrated the agent does not change when applying the rule; we simply try to infer a match between the current agent structure a, and a new context C' constructed as the composition of the former context C composed with the former redex R. As required, the new redex R' is an identity and the parameter d is preserved.

The PRIME-AXIOM and WIRING-AXIOM axioms are the base cases of matching inferences. The latter axiom is used to match bigraphs of zero width (i.e., bigraphs which only contain wiring). By iterated application of this axiom and PAR, we handle matching of any idle names (i.e., outer names not connected to anything). Intuitively, this axiom allows us to only be concerned with wiring with no idle names (i.e., epi wiring) in all other rules (cf., Note 4.A.21 in the appendix, immediately following the proof of Lemma 4.3.8 concerned with WIRING-AXIOM.)

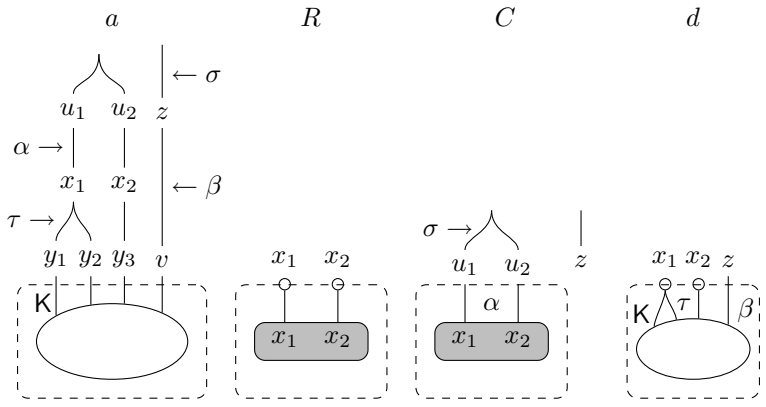


Figure 4.9: Matching $a = C(R \otimes \mathsf{id}_Z)d$ by using PRIME-AXIOM.

The PRIME-AXIOM handles the case where we have matched all nodes in the redex and context; this is the case when only sites remain. The axiom expresses this for primes only; as PAR allows us to combine several valid matches with product, most other rules simply need to be concerned with the prime case. Hence, the axiom requires both redex and context to be single sites (in the context allowing renaming of local names), and requires the agent and parameter place graph to be equal and

their wiring to be compatible.

Figure 4.9 depicts an instance of PRIME-AXIOM. The agent and parameter contain a node of the same control K, and the wirings τ, β, α, and σ, which are composed to determine the wiring of $C(R \otimes \mathsf{id}_Z)d$, matches the wiring of a (which in the figure is split accordingly, to illustrate this match).

Finally, the CLOSE rule allows us to infer a match for bigraphs where all global links are open, and "close" this match by replacing names in wirings with edges, cf. Figure 4.10.

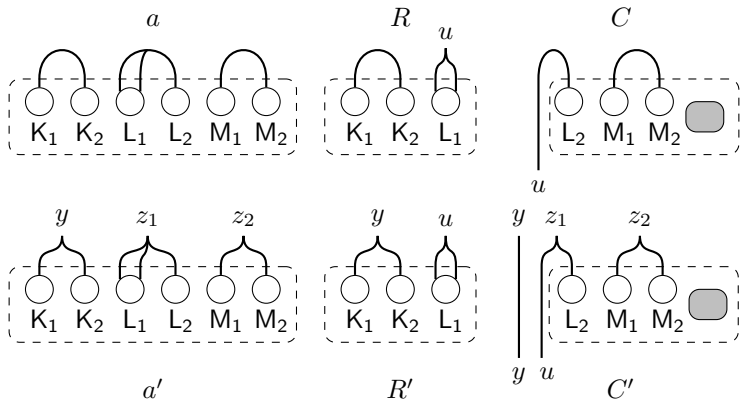


Figure 4.10: Matching closed links within and between redex and context

4.3.4 Soundness and Completeness of the Characterization

The two following theorems state that the rules constitute a sound and complete characterization of matching.

Theorem 4.3.4 (Soundness)**.** *The rules for matching in Figures 4.4 and 4.5 are sound, that is, any derivable matching sentence is valid.*

Proof. Straightforward, by standard algebraic manipulations. □

The completeness theorem is proved by induction on the size of valid sentences, which is defined as follows.

Definition 4.3.5 (Size of a matching sentence)**.** The size of a matching sentence $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \hookrightarrow C, d$ is the number of ions in a.

The following lemmas express how a valid sentence may be derived by applications of inference rules to valid sentences of lesser or equal size. The proofs proceed by first decomposing the components of the given valid sentence, then defining the components of the valid sentence(s) claimed to exist and, finally, verifying that (1) the sentences claimed to exist really are valid and (2) that the given sentence can indeed be derived as claimed. The decompositions are obtained via Proposition 4.3.1 and other normal forms for binding bigraphs, and the verifications proceed using

uniqueness results for normal forms based on those found in [DB06]. We give extensive details for the proofs of these lemmas in the Appendix.

Lemma 4.3.6. *Every valid sentence* $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash a, R \hookrightarrow C, d$ *is provable using the* CLOSE *and the* PERM *rules on a valid sentence, of equal size, of the form* $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash a, S \hookrightarrow \bigotimes_i^n P_i, e$.

Lemma 4.3.7. *Every valid sentence* $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash a, R \hookrightarrow \bigotimes_i^n P_i, d$, *with each* P_i *prime and discrete, is provable using the* PAR *rule on valid sentences, of lesser or equal size, of the form* $\sigma^0_{\mathbf{a}}, \sigma^0_{\mathbf{R}}, \sigma^0_{\mathbf{C}} \,||\, \sigma^{\mathbf{S}}_{\mathbf{C}} \vDash p, S \hookrightarrow P_0, e$ *and* $\sigma^1_{\mathbf{a}}, \sigma^1_{\mathbf{R}}, \sigma^1_{\mathbf{C}} \,||\, \sigma^{\mathbf{S}}_{\mathbf{C}} \vDash a', R' \hookrightarrow \bigotimes_{i=1}^n P_i, e'$. *All substitutions mentioned above are required to be epi (i.e., with no idle names).*

Lemma 4.3.8. *Every valid sentence* $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash a, R \hookrightarrow \mathsf{id}_\epsilon, d$ *is provable using* PAR *and* WIRING-AXIOM.

Lemma 4.3.9. *Every valid sentence* $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash p, R \hookrightarrow P, d$, *with* p *and* P *prime and discrete, is provable using the* LSUB *rule on a valid sentence, of lesser or equal size, of the form* $\sigma'_{\mathbf{a}}, \sigma'_{\mathbf{R}}, \sigma'_{\mathbf{C}} \vDash p', R \hookrightarrow P', d$, *where* p' *and* P' *are discrete free primes. All substitutions mentioned above are required to be epi (i.e., with no idle names).*

Lemma 4.3.10. *Every valid sentence* $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash p, R \hookrightarrow Q, d$, *with* p *and* Q *discrete and free primes, is provable using the* MERGE, PAR *(iterated), and* SWITCH *rules on valid sentences, each of lesser or equal size, and each on one of two forms:*

- $\sigma'_{\mathbf{a}}, \sigma'_{\mathbf{R}}, \sigma'_{\mathbf{C}} \vDash p^{\mathbf{N}}, \mathsf{id} \hookrightarrow P^{\mathbf{N}}, e$, *where* $p^{\mathbf{N}}$ *and* $P^{\mathbf{N}}$ *are free discrete primes,*
- $\sigma'_{\mathbf{a}}, \sigma'_{\mathbf{R}}, \sigma'_{\mathbf{C}} \vDash m, S \hookrightarrow M, e$, *where* m *and* M *are free discrete molecules.*

All substitutions mentioned above are required to be epi (i.e., with no idle names).

Lemma 4.3.11. *Every valid sentence* $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash m, R \hookrightarrow M, d$, *with* m *and* M *free discrete molecules, is provable using the* ION *rule on a valid sentence* $\sigma'_{\mathbf{a}}, \sigma'_{\mathbf{R}}, \sigma'_{\mathbf{C}} \vDash p, R \hookrightarrow P, d$, *of lesser size, where* p *and* P *are discrete primes. All substitutions mentioned above are required to be epi (i.e., with no idle names).*

Lemma 4.3.12. *Every valid sentence* $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash p, \mathsf{id} \hookrightarrow P, e$, *with* p *and* P *free discrete primes, is provable using the* MERGE *and* PAR *(iterated) rules on valid sentences of equal or lesser size, which are either instances of rule* PRIME-AXIOM *or of the form* $\sigma'_{\mathbf{a}}, \sigma'_{\mathbf{R}}, \sigma'_{\mathbf{M}} \vDash m, R \hookrightarrow M, d$. *All substitutions mentioned above are required to be epi (i.e., with no idle names).*

Theorem 4.3.13 (Completeness)**.** *The rules for matching in Figures 4.4 and 4.5 are complete, that is, any valid matching sentence can be derived from the rules.*

Proof. By induction on the size of a sentence. By the lemmas above, we have that all valid sentences with size n can be derived from valid sentences of the form $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash m, R \hookrightarrow M, d$, with m and M free discrete molecules, of size less than or equal to n. By Lemma 4.3.11, these can be derived from sentences of size less than n. □

4.4 An Example: Inferring a Match

In this section, we give an inference tree for inferring the match in the example depicted in Figure 4.1. To fit the inference tree in three reasonably small figures (Figures 4.11, 4.12, and 4.13), we use a more humble visual style, than in Figure 4.1, to depict roots, nodes and names.

Roots are only drawn when there are more than one; in that case we use a dashed separating line to indicate separate roots (see for example the conclusion of PAR in Figure 4.11). Controls of nodes are indicated with the shape (and colour) of the node: Buildings are (blue) rectangles, laptops are (gray) rectangles with rounded corners, folders are (yellow) circles, and data-nodes are black squares with a D inside. Instead of the name *consultancy* we use n and instead of *corporation* we use c. Finally, we do not depict the basic redex R and parameter d, which are illustrated already in Figure 4.1.

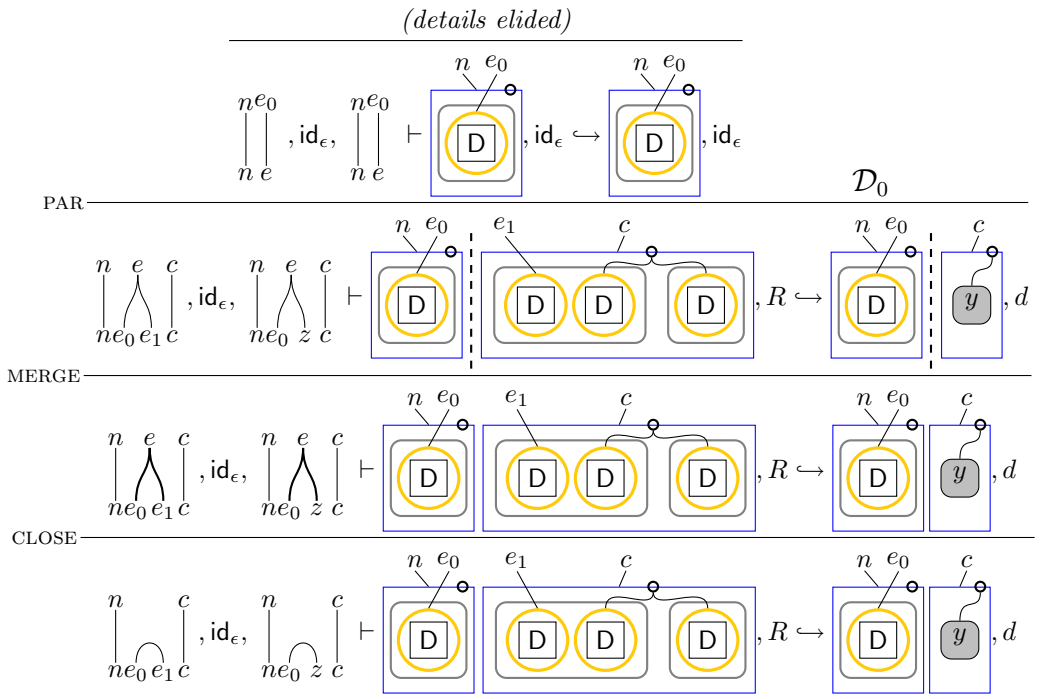


Figure 4.11: Inferring the match in Figure 4.1

We build the inference bottom up and start by decomposing a and C discretely to obtain a sentence that we aim to prove as the conclusion of an application of CLOSE in Figure 4.11. (Note that in contrast to a and C, bigraphs R and d are already discrete as depicted in Figure 4.1.) The application of CLOSE allows us to match and introduce the edge between names e_0 and e_1 in the agent, and between names e_0 and z in the context. We are building an inference bottom up, so in the premiss we simply introduce a fresh outer name e to map these names to.

Next, we aim to use an application of PAR to pair up two inferences of matches

between top-level nodes of the agent and the context. The top-level nodes of the agent and the context are in the same root, though, so after using PAR to pair up two such inferences, we need to apply MERGE to merge the roots introduced by PAR into a single root.

Matching the left root of the agent to the left root of the context is a simple matter, as the agent and context are isomorphic, both in wiring and underlying discrete bigraph, while the redex and parameter are empty (consequentially, we leave out the details for that subderivation). The derivation $\mathcal{D}_0$ is depicted in Figure 4.12.

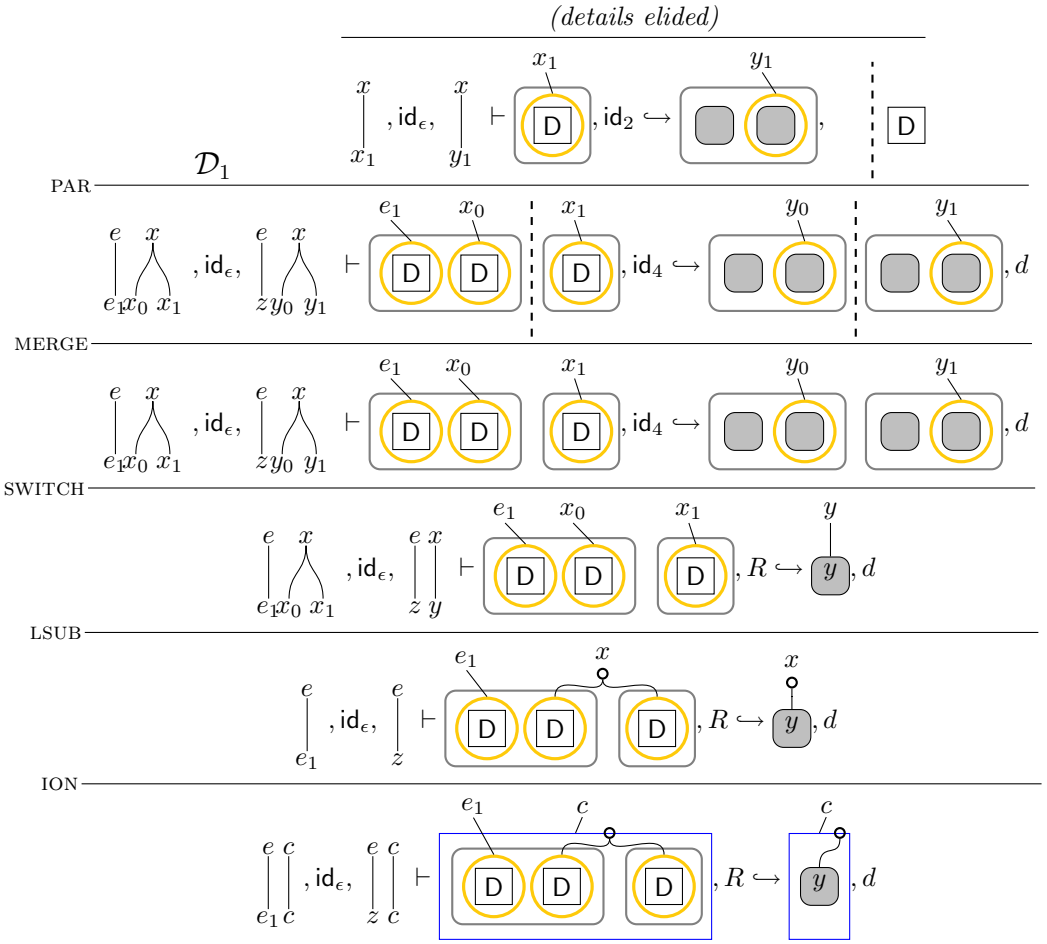


Figure 4.12: The derivation $\mathcal{D}_0$

In the conclusion of the derivation $\mathcal{D}_0$ we use an application of ION to match the top-level building node of the right root of the agent to the building node of the right root of the context. Reading the application of the ION rule bottom up, we see that removing the building node in both agent and context requires us to match and remove the global wiring upon free ports of the building node (e/e_1 and e/z, respectively), and introduce a common local outer name x to map linkage upon

binding ports to.

We immediately proceed to remove that local name. The LSUB rule allows to introduce local names and local wiring; so when building up an inference tree, we simply introduce new global names (in this case, x_0 and x_1 in the agent, and y in the context), and wire those names through global wiring instead.

In the resulting premiss of LSUB the only remainder of the context is a concretion $\ulcorner y \urcorner$ (i.e., a single site), so we can SWITCH to matching the redex. This means that we need to try to infer a match between the current agent structure (in the derivation) and the remainder of the context (i.e., the context wiring and $\ulcorner y \urcorner$) composed with the redex R — with identity redex and the same parameter d. The composition $\ulcorner y \urcorner R$ makes global the name y of R, which is subsequentially wired to an x in the context-wiring. Consequentially, in the context in the premiss of SWITCH we have R with two global names y_0 and y_1, which are wired through the link $x/\{y_0, y_1\}$. The redex in the premiss is id_4 to match the four roots of the parameter d.

Now, we aim again to use an application of PAR to pair up two inferences of matches between the top-level nodes of the current agent and context. After using PAR to pair up those inferences, we need again to apply MERGE to merge the roots introduced by PAR into a single root. The remaining two derivations for matching folders with data in the agent to folders and data in the (former) redex and parameter are very similar; we show the leftmost derivation $\mathcal{D}_1$ in Figure 4.13.

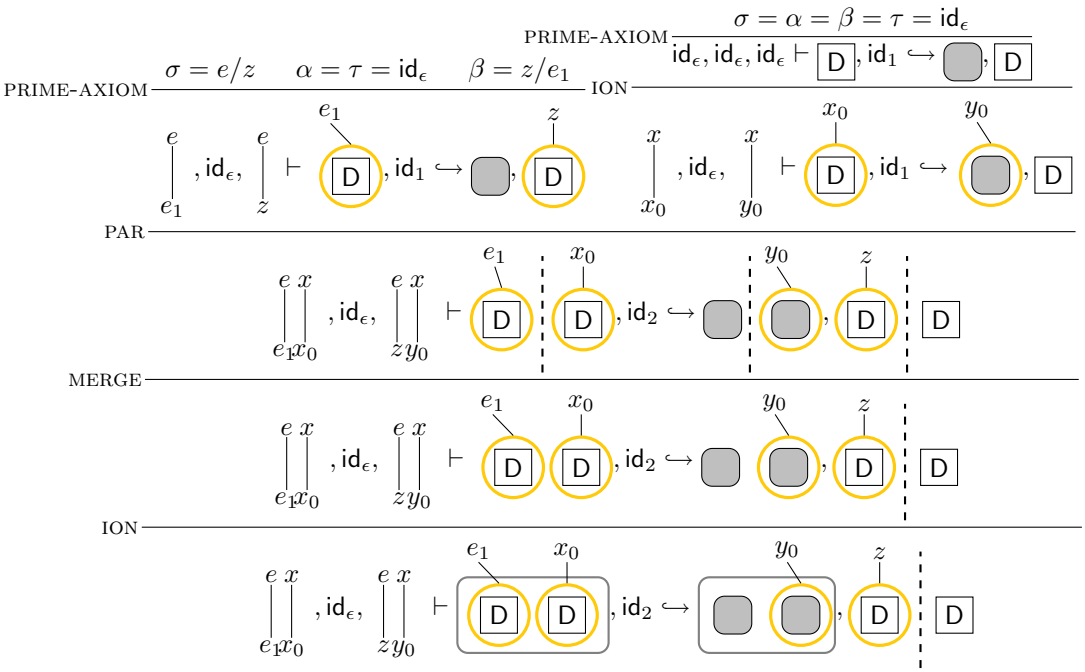


Figure 4.13: The derivation $\mathcal{D}_1$

We conclude the derivation $\mathcal{D}_1$ as we did $\mathcal{D}_0$, namely with an application of ION, in this case, to match and remove a laptop node. Folder nodes have no ports, so here we need not be concerned with wiring.

Another application of the rules MERGE and PAR allows to combine two derivations, the left of which is an instance of PRIME-AXIOM. The PRIME-AXIOM requires both redex and context to be single sites (in the context allowing renaming of local names), and requires the agent and parameter place graph to be equal and their wiring to be compatible. In this case, only the names e_1 and z differ, but as they are internal (i.e., disappear when composing the wiring with the discrete underlying bigraph), we can construct suitable wirings $\sigma = e/z$ and $\beta = z/e_1$ to verify that we have a valid instance of PRIME-AXIOM.

In the remaining rightmost derivation we need a single application of ION, before being able to conclude the entire derivation with a very simple instance of PRIME-AXIOM, as data nodes have no ports to connect wiring to.

4.5 Towards Algorithms for Matching

The completeness theorem tells us that we can find all valid matching sentences by applications of the rules for matching. Thus the rules for matching define an algorithm for matching, for instance easily expressed in Prolog, which simply operates by searching for inference trees using the rules.

Although we can base a matching algorithm directly upon the matching rules, we do not claim that an efficient matching algorithm has to be so based. We have introduced matching rules for a dual purpose: first, to characterize matching structurally and inductively in order to understand it (in particular, to understand the relation to representations based on normal forms and to understand exactly where choices between different matches can be made during matching); second, to provide a specification from which to begin the search for truly efficient matching algorithms, and to verify them against. This rigorous approach to matching is justified, in our view, because matching will be the workhorse of any implementation of bigraph dynamics.

In practice, one is, of course, interested in minimizing unnecessary blind search, and thus, for instance, only search for inference trees of a certain form. Indeed, one can show that it suffices to consider so-called *normal inference trees*, which put restrictions on the order in which the inference rules are applied (such as, e.g., always concluding with the CLOSE rule). We shall not include a formal definition of normal inference trees here, but rather discuss some of the possibilities for defining normal inference trees. We first remark that to retain completeness, any definition of normal inference must, of course, ensure no loss of provability. Looking at the formulations of the lemmas leading up to the completeness theorem, we see that there are indeed several possibilities for the definition of normal inference tree. For example, from Lemma 4.3.6 we see that we are free to conclude each inference tree with CLOSE and then PERM or vice versa. Further, in several rules we are allowed to propagate closed links, even though CLOSE intuitively makes that unnecessary. We have chosen to leave this freedom in the rule system and instead comment on how we could *extend* the set of rules to allow even more freedom in chosing our definition of normal inference tree. This is important when thinking about implementations,

as each definition of normal inference tree corresponds to a different algorithmic approach to matching.

One may say that the current set of rules naturally give rise to normal inferences that are a mix between matching the link graph "lazily"or "eagerly". Instead of the CLOSE rule, one could have amended the PAR and ION rules (those with $||$ in the conclusion) such that they would also handle matching of closures. This would have allowed true "by need" link-matching. Conversely, one could have amended the CLOSE rule to also compare substitutions, allowing us to consider matching of discrete bigraphs up to renamings (i.e., isomorphisms) on their outerfaces. If we amended the LSUB and SWITCH rules to work accordingly, this would actually preclude the need for the wirings $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}}$ in matching sentences. It seems, though, that the tedious complexity added into these rules would mean that we would gain little in removing complexity from the rules as a whole. Anyhow, these changes would allow us to define a variant of normal inferences, which would be "strict" in the link graph, in that we would immediately be able to reject possible matches based on the link graph (instead of the place graph).

Another possibility would be to add a rule GLOB, allowing us to match all wiring stemming from a *single* prime as global wiring. This idea seems to indicate that matching in *local* bigraphs [Mil04] (where there is no global linkage but instead multilocated names) could be handled similarly, by recasting the rules to work on local links and just locating names at all roots where they occur.

4.5.1 Representations of Bigraphs

An implementation of matching must, of course, represent bigraphs in some way. One possibility is to represent bigraphs directly by place and link graphs, and then implement the normal form lemmas, which express how bigraphs may be decomposed into simpler bigraphs; then matching can proceed by induction on the decomposed graph. In general, however, the "decomposition functions" return *sets* of possible decompositions, because normal forms are only unique up to certain permutations. (For example, $merge(M_1 \otimes M_2) = merge(M_2 \otimes M_1)$.) A matching implementation needs to explore all the possible decompositions. This can be made explicit formally, by phrasing the inductive characterization of matching not on bigraphs but on *expressions* (i.e., syntax) for binding bigraphs. Doing so forces us to add an inference rule, which allows one to replace any expression in a matching sentence $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \hookrightarrow C, d$, say a, by a', when a' is provably equal to a via the axioms for structural equality of bigraphs [DB06]. Doing so clearly yields a complete set of rules on bigraphical expressions, but yields a wildly nondeterministic inference system as we might need to apply equality axioms between every step to infer a match. (This is reminiscent of problems arising when implementing rewriting logic, i.e., term rewriting modulo a set of static equivalences.) Consequentially, normal inference trees for rules based on syntax, needs to spell out *how* and *where* to apply equality axioms. The definition of normal inference trees will then *formally explicate* all the possibilities that a matching algorithm needs to explore.

4.5.2 A Prototype Implementation

Based on the considerations above, we have worked out a definition of normal inference tree for matching bigraph expressions and proved it complete. In particular, we also utilize normal forms for expressions by defining normal inferences that require each inference to start by rewriting the term to be on normal form. This restricts considerably the set of expressions that the normal inferences need be concerned with.

This definition of normal inferences is the basis of a prototype implementation of a tool for working with bigraphical reactive systems. We report on this in a forthcoming paper [GDBH07]. The prototype tool is also available online at:

`http://tiger.itu.dk:8080/bplweb/`

4.6 Related and Future Work

Bigraphical reactive systems are related to general graph transformations systems; see [EEPT06] for a recent comprehensive overview of graph transformation systems. In particular, bigraph matching is strongly related to the general graph pattern matching (GPM) problem, so general GPM algorithms might be applicable [Ull76, Fu97, LV02, Zün94]. Due to the special structure of bigraphs, general GPM algorithms are expected to be inefficient, although some GPM tools [VVF05] use heuristic search strategies that might be able to discover and exploit bigraph structure.

A special aspect of bigraphs is that we may match a set of subtrees with a single node (site) in the redex, and match multiple redex roots in different places within the agent. Fu [Fu97] handles such wildcard nodes and multiple patterns, but directly applying his algorithm is not straightforward, as he attacks the problem of tree isomorphism of rooted graphs unfolded to finite unbounded depths. The subtree isomorphism problem [Sel77, Val02, ST99] is simpler than GPM, but applying it directly to the place graphs of bigraphs would not exploit the constraints imposed by the link graphs. Rather, efficient implementations of bigraph matching should be derived from the initial implementation by experimenting with different normal inference tree definitions, and combining it with subtree isomorphism algorithms. The inductive characterization provided here will make it easier to prove an actual algorithm correct. Finally, as noted in the introduction, by providing an abstract characterization founded in well-established theory for bigraphs, we expect to be able to combine or adapt more easily our approach to theory and techniques being developed for bigraphs.

For a more detailed account of related work, in particular on relations between BRSs, graph transformations, term rewriting and term graph rewriting, see [Dam06, Section 6].

Future work on bigraph matching include investigating how we may combine, for instance, our approach with sortings on bigraphs [BDH06], which could be a source of early search elimination. We are also considering rephrasing the rules to

derive a set of constraints for wirings (the three first components of a matching sentence), which could be fed to a constraint solving algorithm, instead of matching them online as the rules.

4.7 Conclusion

We have presented a sound and complete inductive characterization of matching for binding bigraphs. The characterization provides a formal specification for a matching algorithm for binding bigraphs; and, even further, the specification has already served as the basis for an implementation of a prototype tool for working with BRSs.

Acknowledgement This work was funded in part by the Danish Research Agency (grant no.: 2059-03-0031) and the IT University of Copenhagen (the LaCoMoCo project).

4.A Proofs of Completeness

We give extensive details for the proof of Lemmas 4.3.6 through 4.3.12. In proving the main lemmas underpinning the completeness of the matching rules, we shall need a number of properties of binding bigraph structure.

4.A.1 Algebraic Properties of Parallel Product

The *parallel product* $G \,||\, H$ of two bigraphs $G : X \to Y \uplus Z$ and $H : U \to V \uplus Z$, with $X \cap U = Y \cap V = \emptyset$, is given by taking the tensor of the place graphs and the union of the link graph maps.

The equational properties of wiring, composition and tensor product has already been investigated [Mil05, DB06]; in this section we build on this foundation to state directly also a number of properties of wiring and $||$, which shall allow us a number of convenient equational manipulations. We shall mainly use equational reasoning to prove the properties, in the process illustrating the convenience allowed by the axiomatizations and the normal form for links (cf., *loc. cit.*).

We start by giving some simple equivalent forms of the definition of the parallel product.

Lemma 4.A.1 (Parallel product)**.** *For* $Z = \{z_0, \ldots, z_{n-1}\}$

$$\begin{aligned} G \,||\, H &= \big((\textstyle\bigotimes_i^{|Z|} z_i/\{z_i, z_i'\}) \otimes \mathsf{id}_Y \otimes \mathsf{id}_V\big)\big(G \otimes ((\textstyle\bigotimes_i^{|Z|} z_i'/z_i) \otimes \mathsf{id}_V)H\big) \\ &= \big(\vec{z}/(\vec{z}, \vec{z'}) \otimes \mathsf{id}_Y \otimes \mathsf{id}_V\big)\big(G \otimes (\vec{z'}/\vec{z} \otimes \mathsf{id}_V)H\big) \\ &= \big(\sigma \otimes \mathsf{id}_Y \otimes \mathsf{id}_V\big)\big(G \otimes (\alpha \otimes \mathsf{id}_V)H\big), \end{aligned}$$

where in the second equation we introduce some shorthand notation and in the third equation σ and α are given by the previous equations.

Proof. (*Omitted*) Routine from [JM04, Def. 9.13]. □

Sometimes, we shall need to split up a wiring by its inner- or outerface and analyze it in smaller parts.

When splitting up a wiring $\omega : U \uplus V \to X \uplus Y$ by its innerface or outerface, respectively, we get, for δ a suitable closure,

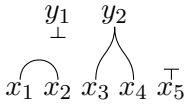
$$\omega = (\delta \otimes \mathsf{id})(\sigma_0 \,||\, \sigma_1) \qquad \sigma_0 : U \to \qquad \sigma_1 : V \to$$

respectively

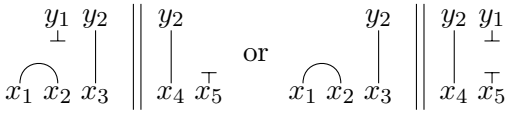
$$\omega = \omega_0 \otimes \omega_1 \qquad \omega_0 :\to X \qquad \omega_1 :\to Y,$$

for δ, σ_0, and σ_1; and ω_0 and ω_1, which are constrained by ω, U, V, X and Y.

Those subwirings (i.e., the substitutions, closures, and wirings) of a splitting are *not* in general determined uniquely by ω, U and V, or ω, X and Y. For example, when splitting

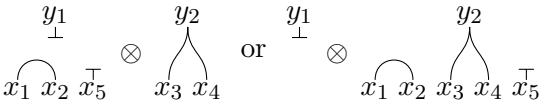
$\omega =$ 

by $\{x_1, x_2, x_3\}$ and $\{x_4, x_5\}$, we get either



When splitting by the innerface, we are free to distribute the idle names of ω. When the wirings have no idle names (i.e., are epi) the splitting is unique. We state this formally in Lemma 4.A.5.

When splitting by the outerface, it is closures — that introduce edges, that constitute the problem. For instance, there are four valid splittings of ω above by $\{y_1\}$ and $\{y_2\}$, corresponding to the four possible ways of distributing the two closed links between the two wirings. We might, for instance, choose to put both closures in either the lefthand or the righthand wiring:



Splittings by the outerface of pure substitutions (i.e., with no closures), on the other hand, are unique; we state this formally in Lemma 4.A.6.

We start by recording that for wirings $\otimes$ and $||$ commute.

Lemma 4.A.2 (Tensor and parallel product commute for wirings)**.** *For all wirings $\omega_{\mathbf{a}} :\to Z \uplus Y_{\mathbf{a}}$, $\omega_{\mathbf{b}} :\to Z \uplus Y_{\mathbf{b}}$, $\omega_{\mathbf{c}} :\to X \uplus Y_{\mathbf{c}}$, and $\omega_{\mathbf{d}} :\to X \uplus Y_{\mathbf{d}}$, with all inner faces mutually disjoint, $X \cap Z = \emptyset$, and $(Y_{\mathbf{a}} \cap Y_{\mathbf{c}}) = (Y_{\mathbf{b}} \cap Y_{\mathbf{d}}) = \emptyset$, it holds that*

$$(\omega_{\mathbf{a}} \,||\, \omega_{\mathbf{b}}) \otimes (\omega_{\mathbf{c}} \,||\, \omega_{\mathbf{d}}) = (\omega_{\mathbf{a}} \otimes \omega_{\mathbf{c}}) \,||\, (\omega_{\mathbf{b}} \otimes \omega_{\mathbf{d}}).$$

Proof. Equationally,

$$\begin{aligned}
&(\omega_{\mathbf{a}} \mid\mid \omega_{\mathbf{b}}) \otimes (\omega_{\mathbf{c}} \mid\mid \omega_{\mathbf{d}}) \\
&\quad = (\sigma_Z \otimes \mathsf{id})(\omega_{\mathbf{a}} \otimes (\alpha_Z \otimes \mathsf{id})\omega_{\mathbf{b}}) \otimes (\sigma_X \otimes \mathsf{id})(\omega_{\mathbf{c}} \otimes (\alpha_X \otimes \mathsf{id})\omega_{\mathbf{d}}) \\
&\quad = (\sigma_Z \otimes \mathsf{id})(\omega_{\mathbf{a}} \otimes (\alpha_Z \omega_{\mathbf{b}}^Z \otimes \omega'_{\mathbf{b}})) \otimes (\sigma_X \otimes \mathsf{id})(\omega_{\mathbf{c}} \otimes (\alpha_X \omega_{\mathbf{d}}^X \otimes \omega'_{\mathbf{d}})) \\
&\quad = (\sigma_Z \otimes \sigma_X \otimes \mathsf{id})(\omega_{\mathbf{a}} \otimes \omega_{\mathbf{c}} \otimes (\alpha_Z \omega_{\mathbf{b}}^Z \otimes \alpha_X \omega_{\mathbf{d}}^X \otimes \omega'_{\mathbf{d}} \otimes \omega'_{\mathbf{b}})) \\
&\quad = (\sigma_Z \otimes \sigma_X \otimes \mathsf{id})(\omega_{\mathbf{a}} \otimes \omega_{\mathbf{c}} \otimes (\alpha_Z \otimes \alpha_X \otimes \mathsf{id})(\omega_{\mathbf{b}}^Z \otimes \omega'_{\mathbf{d}} \otimes \omega_{\mathbf{d}}^X \otimes \omega'_{\mathbf{b}})) \\
&\quad = (\omega_{\mathbf{a}} \otimes \omega_{\mathbf{c}}) \mid\mid (\omega_{\mathbf{b}} \otimes \omega_{\mathbf{d}}).
\end{aligned}$$

— where the first and the last equality are instances of the third equation of Lemma 4.A.1, and $\omega_{\mathbf{b}}^Z$, $\omega'_{\mathbf{b}}$, $\omega_{\mathbf{d}}^X$, and $\omega'_{\mathbf{d}}$ arises from splitting $\omega_{\mathbf{b}}$ and $\omega_{\mathbf{d}}$ by the outerface. □

We state and prove Lemma 4.A.3, a cancellation property of parallel product and idle names, only to help us prove Lemma 4.A.4. The latter lemma tells us, that should the parallel product of two pairs of wirings with matching interfaces be equal, then those pairs were equal in the first place.

Lemma 4.A.3. *For wirings $\omega^i = X^i \to Y^i \uplus Z$ $(i \in \{0,1\})$ with $X^0 \cap X^1 = Y^0 \cap Y^1 = \emptyset$,*

$$(\omega^0 \mid\mid \omega^1)(X^0 \otimes \mathsf{id}_{X^1}) = Y^0 \otimes \omega^1.$$

Proof. Equationally,

$$\begin{aligned}
(\omega^0 \mid\mid \omega^1)(X^0 \otimes \mathsf{id}_{X^1}) &= \omega^0 X^0 \mid\mid \omega^1 \\
&= (Y^0 \otimes Z) \mid\mid \omega^1 \\
&= Y^0 \otimes (\sigma_Z \otimes \mathsf{id}_{Y^1})(Z \otimes (\alpha_Z \otimes \mathsf{id}_{Y^1})\omega^1) \\
&= Y^0 \otimes (\sigma_Z \otimes \mathsf{id}_{Y^1})(Z \otimes \alpha_Z \omega_Z^1 \otimes \omega_{Y^1}^1) \\
&= Y^0 \otimes \sigma_Z(Z \otimes \alpha_Z \omega_Z^1) \otimes \omega_{Y^1}^1 \\
&= Y^0 \otimes \alpha_Z^{-1} \alpha_Z \omega_Z^1 \otimes \omega_{Y^1}^1 \\
&= Y^0 \otimes \omega^1.
\end{aligned}$$

where $\sigma_Z \stackrel{\text{def}}{=} \vec{z}/(\vec{z}, \vec{z'})$, $\alpha_Z \stackrel{\text{def}}{=} \vec{z'}/\vec{z}$ (as defined in Lemma 4.A.1), and $\omega_Z^1 \otimes \omega_{Y^1}^2 \stackrel{\text{def}}{=} \omega^1$ is a splitting by the outerface of ω^1, s.t. $\omega_Z^1 :\to Z$ and $\omega_{Y^1}^1 :\to Y^1$.

The fifth equality resolves the composition (equationally, by iterated application of axiom L4 of [DB06]). For our particular purpose, the axiom can be instantiated to state for each $z \in Z$ that $z/(z, z')(z \otimes \mathsf{id}_{z'}) = z/z'$. Iterating this, we have also that $\sigma_Z(Z \otimes \mathsf{id}_{Z'}) = \alpha_Z^{-1} = \vec{z}/\vec{z'}$. □

Lemma 4.A.4. *Given $\omega_{\mathbf{c}}^i, \omega_{\mathbf{a}}^i : X^i \to Y^i \uplus Z$, $(i = 0, 1)$ with $X^0 \cap X^1 = Y^0 \cap Y^1 = \emptyset$, we have that*

$$\omega_{\mathbf{c}}^0 \mid\mid \omega_{\mathbf{c}}^1 = \omega_{\mathbf{a}}^0 \mid\mid \omega_{\mathbf{a}}^1 \qquad \textit{iff} \qquad \omega_{\mathbf{c}}^0 = \omega_{\mathbf{a}}^0 \quad \textit{and} \quad \omega_{\mathbf{c}}^1 = \omega_{\mathbf{a}}^1.$$

Proof. ($\Leftarrow$) Immediate.
($\Rightarrow$) Composing on both sides of the assumed equation with equal terms, we have,

$$\begin{aligned}
(/Y^0 \otimes \mathsf{id}_{Y^1 \uplus Z})(\omega_{\mathbf{c}}^0 \mid\mid \omega_{\mathbf{c}}^1)(X^0 \otimes \mathsf{id}_{X^1}) &= (/Y^0 \otimes \mathsf{id}_{Y^1 \uplus Z})(Y^0 \otimes \omega_{\mathbf{c}}^1) &= \omega_{\mathbf{c}}^1, \text{ and} \\
(/Y^0 \otimes \mathsf{id}_{Y^1 \uplus Z})(\omega_{\mathbf{a}}^0 \mid\mid \omega_{\mathbf{a}}^1)(X^0 \otimes \mathsf{id}_{X^1}) &= (/Y^0 \otimes \mathsf{id}_{Y^1 \uplus Z})(Y^0 \otimes \omega_{\mathbf{a}}^1) &= \omega_{\mathbf{a}}^1.
\end{aligned}$$

— and analogously for $\omega^1_{\mathbf{c}}$ and $\omega^1_{\mathbf{a}}$ (using Lemma 4.A.3 to resolve the first equalities). □

And now we can state and prove the two lemmas mentioned earlier, which tell us when a splitting of wiring is unique.

Lemma 4.A.5 (Splitting by the innerface)**.** *The splitting of $\omega : U \uplus V \to$ (i.e., with $U \cap V = \emptyset$) by its innerface into $\omega = (\delta \otimes \mathsf{id})(\sigma_0 \otimes \sigma_1)$ with $\sigma_0 : U \to$ and $\sigma_1 : V \to$ is unique (up to iso) if ω is epi.*

Proof. Suppose wlog that $\omega : U \uplus V \to Z$. We analyse the underlying link-function link_ω of ω. When ω is epi, the link-function has codomain exactly $\mathit{link}_\omega(U \uplus V) = Z \uplus E$, where E is the set of edges in ω. (There are no edges in E with no preimage in link_ω, as we are concerned with abstract bigraphs, which contain no such idle edges.)

The link-function of σ_0 must, for each, $u \in U$, either contain $u \mapsto \mathit{link}_\omega(u)$ if $\mathit{link}_\omega(u) \in Z$, or $u \mapsto z_u$ for a fresh name z_u if $\mathit{link}_\omega(u) \in E$; in the latter case, δ must also contain $z_u \mapsto \emptyset$. The same goes for σ_1.

We can freely choose the names for transferring closed links, such as z_u; having chosen those names, both the interfaces of σ_0 and σ_1 are also determined, since Z contains only images of $U \uplus V$. By Lemma 4.A.4 this determines them wholly. □

Lemma 4.A.6 (Splitting by the outerface)**.** *The splitting of $\sigma :\to X \uplus Y$ (i.e., with $X \cap Y = \emptyset$) by its outerface into $\sigma = \sigma_0 \,||\, \sigma_1$ with $\sigma_0 :\to X$ and $\sigma_1 :\to Y$ is unique.*

Proof. Suppose wlog that $\sigma : U \to X \uplus Y$. Since σ is a substitution its underlying link-function is defined precisely on $\mathit{link}_\sigma : U \to X \uplus Y$(i.e., there are no ports or names in the link-function).

The relation $\mathit{link}^{-1}_\sigma$ relates to each $x \in X$ and $y \in Y$ a (possibly empty) set $U_x \subseteq U$ or $U_y \subseteq U$, respectively. Since link_σ is a function, it is clear that all these sets are distinct and disjoint.

Hence, we can construct mechanically the domain U_0 of link_{σ_0} by taking the union of the preimage of the names in X, i.e., $U_0 = \mathit{link}^{-1}_\sigma(X)$; and this set is disjoint from $U_1 = \mathit{link}^{-1}_\sigma(Y)$. This procedure determines the interfaces of σ_0 and σ_1 uniquely, hence, by Lemma 4.A.4 it determines them wholly. □

Finally, we record a few further convenient properties of the interplay of substitutions with parallel product.

Lemma 4.A.7. *When both sides are defined, we have:*

$$\sigma \,||\, \sigma\alpha = \sigma(\mathsf{id} \,||\, \alpha), \tag{4.1}$$

$$\sigma(\omega_0 \,||\, \omega_1) = \sigma\omega_0 \,||\, \sigma\omega_1, \tag{4.2}$$

$$(\sigma_0 \,||\, \sigma_1 \,||\, \sigma_2)(\omega_0 \,||\, \omega_1) = (\sigma_0 \,||\, \sigma_2)\omega_0 \,||\, (\sigma_1 \,||\, \sigma_2)\omega_1. \tag{4.3}$$

Proof. (4.1) By definition of $||$ and normal form for σ (see, [DB06]).
(4.2) Follows easily from (4.1).
(4.3) Immediate from the earlier properties. □

4.A.2 Valid Matching Sentences and Normal Forms

We start by stating two propositions, which can be derived from the normal form theorem for binding bigraphs [DB06, Theorem 1(2)].

Proposition 4.A.8. *Any discrete bigraph D of width n with local innerface can be decomposed such that*

$$D = \big(\bigotimes_i^n (\widehat{\sigma_i} \otimes \mathsf{id}) P_i\big)\pi,$$

where the P_i's are name-discrete and prime. Any other decomposition on this form of D takes the form $\big(\bigotimes_i^n (\widehat{\sigma_i'} \otimes \mathsf{id}) P_i'\big)\pi'$, where, for some $\widehat{\alpha_i}$, ρ_i, for all i, $P_i' = (\widehat{\alpha_i}^{-1} \otimes \mathsf{id}) P_i \rho_i$ $(\bigotimes_i^n \rho_i)\pi' = \pi$, and $\widehat{\sigma_i'} = \widehat{\sigma_i}\widehat{\alpha_i}$.

The normal forms of name-discrete primes and free discrete molecules can be found in *loc. cit.*

One can decompose binding ions $K_{\vec{y}(\vec{X})}$ into $K_{\vec{y}(\vec{u})} \bigotimes_i^n (u_i)/(X_i)$. Such decompositions will be useful because of the following property, which is a corollary of the normal form for free discrete molecules.

In analyzing molecules, it shall be useful that the uniqueness property for free discrete molecules actually holds also for all free molecules.

Proposition 4.A.9. *For primes P and Q, if*

$$\big(K_{\vec{y}(\vec{x})} \otimes \mathsf{id}\big)\, P = \big(K_{\vec{y}(\vec{z})} \otimes \mathsf{id}\big)\, Q,$$

then for $\alpha = \vec{x}/\vec{z}$, $K_{\vec{y}(\vec{x})}\widehat{\alpha} = K_{\vec{y}(\vec{z})}$ and $P = (\widehat{\alpha} \otimes \mathsf{id})Q$.

Proof. (*Omitted*) Follows easily from normal form for primes and molecules. □

We give a number of convenient equivalent forms for Definition 4.3.3 of valid matching sentences. Both are simply results of equational manipulations of the original form. In particular, the second is more compact, while the third separates global linkage from discrete bigraphs. In the proofs, we shall refer to the following Fact instead of Definition 4.3.3.

Fact 4.A.10 (Valid matching sentence — with equivalent forms)**.** A matching sentence $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R(\to C, d$, where $\omega_{\mathbf{R}} : U \to Y$, for C with global outer names V, and d with global outer names Z, is *valid*, denoted $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash a, R(\to C, d$, iff

$$\begin{aligned}(\mathsf{id} \otimes \omega_{\mathbf{a}})a &= (\mathsf{id} \otimes \omega_{\mathbf{C}})(C \otimes \mathsf{id}_Y \otimes \mathsf{id}_Z)\,(\mathsf{id}_Z \otimes (\mathsf{id} \otimes \omega_{\mathbf{R}})\,R)\,d \\ &= (\mathsf{id} \otimes \omega_{\mathbf{C}})((C \otimes \omega_{\mathbf{R}})R \otimes \mathsf{id}_Z)d \\ &= (\mathsf{id} \otimes \omega_{\mathbf{C}}(\mathsf{id}_V \otimes \omega_{\mathbf{R}} \otimes \mathsf{id}_Z))((C \otimes \mathsf{id}_U)R \otimes \mathsf{id}_Z)d.\end{aligned}$$

where unqualified identities are local and determined from the context.

The bigraph $((C \otimes \mathsf{id}_U)R \otimes \mathsf{id}_Z)d$ (i.e., the composition of the underlying discrete bigraphs of context, redex and parameter in the last form given in Fact 4.A.10) is *not* discrete in general. Discreteness is not preserved by composition — in composing discrete D and E, local non-discrete linkage of E may be made global by composition with D. On the other hand, the expression is a ground product of primes where all edges are bound. We record some properties of these kinds of bigraphs, which follow easily from the normal form theorems based on discrete decomposition.

We start by giving a simple restatement of Proposition 3.2, eliding the details of the representation of discrete primes.

Corollary 4.A.11. *Any discrete bigraph D with local innerface of width n can be decomposed such that*

$$D = (\bigotimes_i^n P_i)\pi,$$

where P_i are discrete prime. Any other decomposition on this form of D can be written as $(\bigotimes_i^n P'_i)\pi'$, where for some ρ_i, $P'_i = P_i\rho_i$ and $(\bigotimes_i^n \rho_i)\pi' = \pi$.

We call bigraphs with only bound edges *globally open*.

Definition 4.A.12 (Globally open)**.** G is *globally open* iff all edges in G are bound.

The following property of globally open bigraphs allows us to use the decomposition in Fact 4.A.10 to establish a relation between the wiring of the agent and the wirings of the context, redex and parameter, even though the latter is not discretely decomposed. (We state the proposition only for ground bigraphs, as that is enough for our purposes.)

Proposition 4.A.13 (Semi-discrete decomposition)**.** *If $(\mathsf{id} \otimes \omega_{\mathbf{d}})d = (\mathsf{id} \otimes \omega_{\mathbf{b}})b$, where both identities are local, d is discrete and b is globally open, then there exists a substitution σ, s.t., $\omega_{\mathbf{d}} = \omega_{\mathbf{b}}\sigma$ and $(\mathsf{id} \otimes \sigma)d = b$.*

Proof. Follows easily from Proposition 4.3.1.

By Proposition 4.3.1, and since b is globally open, there exists $\sigma_{\mathbf{e}}$ and e, where e is discrete, s.t., $b = (\mathsf{id} \otimes \sigma_{\mathbf{e}})e$. By uniqueness of the normal form of Proposition 4.3.1, we have

$$\begin{aligned} \omega_{\mathbf{d}} &= \omega_{\mathbf{b}}\sigma_{\mathbf{e}}\beta, \\ d &= (\mathsf{id} \otimes \beta^{-1})e. \end{aligned}$$

We simply take $\sigma = \sigma_{\mathbf{e}}\beta$, and we are done. □

We define products of primes with arbitrary ordering of the sites and call them *link-contained*, as they are characterized precisely by having no links that span several roots. In particular, a link-contained bigraph of width 1 is simply a prime.

Definition 4.A.14 (Link-contained)**.** G is *link-contained* iff G is a product of primes with arbitrary ordering of the sites.

In other words, G (of width n) can be written as $(\bigotimes_i^n P_i)\pi$, where each P_i is prime.

Link-containedness is simply a generalization of discreteness.

Lemma 4.A.15 (Discreteness implies link-containedness)**.**

Proof. (*Omitted*) Immediate from Definition 4.A.14 and normal form for discrete. □

A virtue of link-containedness (as opposed to discreteness) is that it is preserved by composition.

Lemma 4.A.16 (Link-containedness is preserved by composition)**.** *If D and E are link-contained, then DE is link-contained.*

Proof. (*Omitted*) Routine. □

With regard to Fact 4.A.10, Lemmas 4.A.15 and 4.A.16 ensures us that $((C \otimes \mathsf{id}_U)R \otimes \mathsf{id}_Z)d$ is link-contained.

Link-contained bigraphs share essentially all the nice properties with discrete bigraphs, that we used for establishing their normal forms. From Proposition 4.3.1 and Corollary 4.A.11, we derive a normal form for link-contained bigraphs.

Corollary 4.A.17 (Normalform for link-contained bigraphs)**.** *If G is link-contained, then G can be expressed as*

$$G = (\bigotimes_{i}^{n}(\omega_i \otimes \mathsf{id})P_i)\pi,$$

where each P_i is prime and discrete. Further, any other expression for G on this format is of the form

$$(\bigotimes_{i}^{n}(\omega'_i \otimes \mathsf{id})Q_i)\pi',$$

where $(\forall i \in n)$ there exists α_i and ρ_i, s.t. $\omega_i = \omega'_i\alpha_i$, $Q_i = (\alpha_i \otimes \mathsf{id})P_i\rho_i$, and $(\bigotimes_i^n \rho_i)\pi' = \pi$.

Proof. (*Omitted*) Follows easily from the definition of link-contained and the normal forms for bigraphs and discrete bigraphs. □

Finally, we state the following simple property of abstraction.

Lemma 4.A.18.

$$(U)P = (U)Q \text{ iff } P = Q$$

Proof.
($\Leftarrow$) Immediate.
($\Rightarrow$) Nearly immediate; equationally, by composing with equal concretions:

$$P = (\ulcorner U \urcorner \otimes \mathsf{id})(U)P = (\ulcorner U \urcorner \otimes \mathsf{id})(U)Q = Q.$$

□

4.A.3 Proofs of Lemmas 4.3.6 through 4.3.12

In this section we give the proofs for the lemmas which support the main theorem on completeness, i.e., Theorem 4.3.13. For ease of reading, we repeat each lemma immediately before each proof.

In the following proofs, we use equational techniques as allowed by the axiomatization of structural congruence of binding bigraphs [DB06]. In particular, in manipulating terms, we shall need to introduce quite a lot of id's on interfaces determinable from the context. We shall adopt the convention of using only unqualified local identities, when the interface is determinable, and inessential for the analysis. For global identities, we shall strive to use the metavariables introduced in Fact 4.A.10 — i.e., we use V's for identities on context wiring, U's for identities on redex wiring, and Z's for identities on parameter-wiring.

We start by proving two sublemmas, of which Lemma 4.3.6 will be a simple corollary. As CLOSE and PERM work solely on the three link graph and four discrete components, respectively, we proceed by proving a lemma for each of these rules.

Lemma 4.A.19. *Every valid sentence $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash a, R(\to C, d$ is a consequence by* PERM *on a valid sentence, of equal size, of the form $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash a, S(\to \bigotimes_i^n Q_i, e$, where each Q_i is prime (and discrete).*

Proof. By Fact 4.A.10 and Corollary 4.A.11, C can be decomposed directly as $(\bigotimes_i^n Q_i)\pi$, while (regular) R can be decomposed as $\bigotimes_i^m P_i$ (for prime and discrete Q_i, P_i).

Applying these decompositions and the push-through lemma of [DB06] we find by standard manipulations

$$\begin{aligned}(\mathsf{id} \otimes \omega_{\mathbf{a}})a &= (\mathsf{id} \otimes \omega_{\mathbf{C}}) \left(\left(\bigotimes_i^n Q_i \right) \pi \otimes \omega_{\mathbf{R}} \right) \left(\left(\bigotimes_i^m P_i \right) \otimes \mathsf{id}_Z \right) d, \\ &= (\mathsf{id} \otimes \omega_{\mathbf{C}}) \left(\left(\bigotimes_i^n Q_i \right) \otimes \omega_{\mathbf{R}} \right) \left(\left(\bigotimes_i^m P_{\pi^{-1}}(i) \right) \otimes \mathsf{id}_Z \right) (\overline{\pi} \otimes \mathsf{id}_Z) d.\end{aligned}$$

Choosing $S = \bigotimes_i^m P_{\pi^{-1}}(i)$ and $e = (\overline{\pi} \otimes \mathsf{id}_Z)d$ by Fact 4.A.10 we have a valid sentence $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash a, S(\to \bigotimes_i^n Q_i, e$, which taken as the premise in the PERM yields the required sentence as conclusion. □

Lemma 4.A.20. *Every valid sentence $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash a, R(\to C, d$ is a consequence by* CLOSE *on a valid sentence, of equal size, of the form $\sigma'_{\mathbf{a}}, \sigma'_{\mathbf{R}}, \sigma'_{\mathbf{C}} \vDash a, R(\to C, d$.*

Proof. We may write $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}$ and $\omega_{\mathbf{C}}$ as (by the normal form for linkage, cf. [Mil05])

$$(\mathsf{id} \otimes /Y_{\mathbf{a}})\sigma_{\mathbf{a}} \stackrel{\text{def}}{=} \omega_{\mathbf{a}}, \quad (\mathsf{id} \otimes /Y_{\mathbf{R}})\sigma_{\mathbf{R}} \stackrel{\text{def}}{=} \omega_{\mathbf{R}}, \quad (\mathsf{id} \otimes /Y_{\mathbf{C}})\sigma_{\mathbf{C}} \stackrel{\text{def}}{=} \omega_{\mathbf{C}}.$$

(Note, that for this proof (only) we use unqualified global identies.)

We have $|Y_{\mathbf{a}}| = |Y_{\mathbf{R}}| + |Y_{\mathbf{C}}|$; as, in particular, the number of free edges in the agent must be equal to the number of free edges in the context composed with the redex

and the parameter. The discrete bigraphs d, C and R contain no free edges, so the free edges must be created entirely by $\omega_{\mathbf{R}}$ and $\omega_{\mathbf{C}}$. Hence, there exists a renaming $\alpha : Y_{\mathbf{a}} \to Y_{\mathbf{R}} \uplus Y_{\mathbf{C}}$. Assuming validity of the original sentence, we calculate

$$(\mathsf{id} \otimes (\mathsf{id} \otimes \alpha)\sigma_{\mathbf{a}})a = (\mathsf{id} \otimes (\sigma_{\mathbf{C}} \otimes \mathsf{id}_{Y_{\mathbf{R}}})(\mathsf{id}_V \otimes \sigma_{\mathbf{R}} \otimes \mathsf{id}_Z))((C \otimes \mathsf{id}_U)R \otimes \mathsf{id}_Z)d,$$

which by Fact 4.A.10 means that the sentence

$$(\mathsf{id} \otimes \alpha)\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \otimes \mathsf{id}_{Y_{\mathbf{R}}} \vDash a, R(\to C, d,$$

is valid. This sentence is on the required form, and, checking, we see that applying CLOSE to this sentence, we arrive at

$$(\mathsf{id} \otimes /(Y_{\mathbf{C}} \uplus Y_{\mathbf{R}}))(\mathsf{id} \otimes \alpha)\sigma_{\mathbf{a}}, (\mathsf{id} \otimes /Y_{\mathbf{R}})\sigma_{\mathbf{R}}, (\mathsf{id} \otimes /Y_{\mathbf{C}})\sigma_{\mathbf{C}} \vDash a, R(\to C, d,$$

which, since by construction $/(Y_{\mathbf{C}} \uplus Y_{\mathbf{R}})(\mathsf{id} \otimes \alpha) = /Y_{\mathbf{a}}$, is equal to

$$\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash a, R(\to C, d.$$

□

Lemma (4.3.6)**.** Every valid sentence $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash a, R \hookrightarrow C, d$ is provable using the CLOSE and the PERM rules on a valid sentence, of equal size, of the form $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash a, S \hookrightarrow \bigotimes_i^n P_i, e$.

Proof. Immediate by combining Lemma 4.A.20 and Lemma 4.A.19. □

Lemma (4.3.7)**.** Every valid sentence $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash a, R \hookrightarrow \bigotimes_i^n P_i, d$, with each P_i prime and discrete, is provable using the PAR rule on valid sentences, of lesser or equal size, of the form $\sigma^0_{\mathbf{a}}, \sigma^0_{\mathbf{R}}, \sigma^0_{\mathbf{C}} \,||\, \sigma^{\mathbf{S}}_{\mathbf{C}} \vDash p, S \hookrightarrow P_0, e$ and $\sigma^1_{\mathbf{a}}, \sigma^1_{\mathbf{R}}, \sigma^1_{\mathbf{C}} \,||\, \sigma^{\mathbf{S}}_{\mathbf{C}} \vDash a', R' \hookrightarrow \bigotimes_{i=1}^n P_i, e'$. All substitutions mentioned above are required to be epi (i.e., with no idle names).

Proof. By Fact 4.A.10 and Corollary 4.A.11, a can be decomposed as $\bigotimes_i^n p_i$ (with discrete and prime p_i). We immediately utilize that, assuming validity of the original sentence, the width of a and $\bigotimes_i^n P_i$ must be equal.

We have, by Fact 4.A.10 (using, as usual, unqualified local identities and introducing $C \stackrel{\text{def}}{=} \bigotimes_{i=1}^n P_i$)

$$(\mathsf{id} \otimes \sigma_{\mathbf{a}}) \bigotimes_i^n p_i = (\mathsf{id} \otimes \sigma_{\mathbf{C}}(\mathsf{id}_{V_0} \otimes \mathsf{id}_V \otimes \sigma_{\mathbf{R}} \otimes \mathsf{id}_Z))((P_0 \otimes C \otimes \mathsf{id}_U)R \otimes \mathsf{id}_Z)d.$$

By Proposition 4.A.13, there exists a substitution $\sigma^{\mathbf{L}}$, s.t.,

$$\sigma_{\mathbf{a}} = \sigma_{\mathbf{C}}(\mathsf{id}_{V_0} \otimes \mathsf{id}_V \otimes \sigma_{\mathbf{R}} \otimes \mathsf{id}_Z)\sigma^{\mathbf{L}}, \tag{4.4}$$

$$\text{and} \quad (\mathsf{id} \otimes \sigma^{\mathbf{L}}) \bigotimes_i^n p_i = ((P_0 \otimes C \otimes \mathsf{id}_U)R \otimes \mathsf{id}_Z)d. \tag{4.5}$$

We can partition the redex according to the innerfaces of P_0 and C; and then partition the parameter according to the partioning of the redex. From (4.5) we can derive

$$(\mathsf{id} \otimes \sigma^{\mathbf{L}}) \bigotimes_{i}^{n} p_i = ((P_0 \otimes \mathsf{id}_{U_0})R_0 \otimes \mathsf{id}_{Z_0})d_0 \otimes ((C \otimes \mathsf{id}_{U'}) \bigotimes_{i=1}^{m} R_i \otimes \mathsf{id}_{Z'}) \bigotimes_{i=1}^{l} d_i, \tag{4.6}$$

introducing a few more convenient metavariables: For the partioned parts of the redex and parameter, we let $\bigotimes_i^m R_i \stackrel{\text{def}}{=} R$ and $\bigotimes_i^l d_i \stackrel{\text{def}}{=} d$ (where each R_i and d_i is a product of primes). Also, in the following, we shall use the shorthands $R' \stackrel{\text{def}}{=} \bigotimes_{i=1}^m R_i$ and $d' \stackrel{\text{def}}{=} \bigotimes_{i=1}^l d_i$.

From (4.6) we see that we must be able to partition $\sigma^{\mathbf{L}}$ equally. The substitution must be on the form $\sigma^{\mathbf{L}} = \left(\sigma^{\mathbf{L}}_{\mathbf{P_0}} \otimes \sigma^{\mathbf{L}}_{\mathbf{R_0}} \otimes \sigma^{\mathbf{L}}_{\mathbf{d_0}}\right) \otimes \left(\sigma^{\mathbf{L}}_{\mathbf{C}} \otimes \sigma^{\mathbf{L}}_{\mathbf{R'}} \otimes \sigma^{\mathbf{L}}_{\mathbf{d'}}\right)$, as we can split $\sigma^{\mathbf{L}}$ according to the outerfaces of $((P_0 \otimes \mathsf{id}_{U_0})R_0 \otimes \mathsf{id}_{Z_0})d_0$ and $((C \otimes \mathsf{id}_{U'})R' \otimes \mathsf{id}_{Z'})d'$. (This splitting is unique; as are all the other splittings of wiring in this lemma, since we work only on epi substitutions — cf. Lemma 4.A.5 and Lemma 4.A.6.)

By Corollary 4.A.17, as the bigraphs are ground (and hence, there are no permutations to consider), we find

$$(\mathsf{id} \otimes \sigma^{\mathbf{L}}_{\mathbf{P_0}} \otimes \sigma^{\mathbf{L}}_{\mathbf{R_0}} \otimes \sigma^{\mathbf{L}}_{\mathbf{d_0}})p_0 = ((P_0 \otimes \mathsf{id}_{U_0})R_0 \otimes \mathsf{id}_{Z_0})d_0 \tag{4.7}$$

$$\text{and} \quad (\mathsf{id} \otimes \sigma^{\mathbf{L}}_{\mathbf{C}} \otimes \sigma^{\mathbf{L}}_{\mathbf{R'}} \otimes \sigma^{\mathbf{L}}_{\mathbf{d'}}) \bigotimes_{i=1}^{n} p_i = ((C \otimes \mathsf{id}_{U'})R' \otimes \mathsf{id}_{Z'})d'. \tag{4.8}$$

We can also split $\sigma_{\mathbf{a}}$ by its innerface, which it shares with $\sigma^{\mathbf{L}}$ (cf. (4.4)), and define

$$\left(\sigma^{\mathbf{P_0}}_{\mathbf{a}} \mid\mid \sigma^{\mathbf{R_0}}_{\mathbf{a}} \mid\mid \sigma^{\mathbf{d_0}}_{\mathbf{a}}\right) \mid\mid \left(\sigma^{\mathbf{C}}_{\mathbf{a}} \mid\mid \sigma^{\mathbf{R'}}_{\mathbf{a}} \mid\mid \sigma^{\mathbf{d'}}_{\mathbf{a}}\right) \stackrel{\text{def}}{=} \sigma_{\mathbf{a}}.$$

(In the following we also use $\sigma^{\mathbf{R}}_{\mathbf{a}} \stackrel{\text{def}}{=} \sigma^{\mathbf{R_0}}_{\mathbf{a}} \mid\mid \sigma^{\mathbf{R'}}_{\mathbf{a}}$.)

Equally, we can split $\sigma_{\mathbf{R}}$ by its innerface into $\sigma_{\mathbf{R_0}} || \sigma_{\mathbf{R'}}$ (corresponding to the outerface of $\sigma^{\mathbf{L}}_{\mathbf{R_0}}$ and $\sigma^{\mathbf{L}}_{\mathbf{R'}}$, respectively). Finally, we split $\sigma_{\mathbf{C}}$ by its innerface (cf. (4.4)), to define

$$\sigma^{\mathbf{P_0}}_{\mathbf{C}} \mid\mid \sigma^{\mathbf{d_0}}_{\mathbf{C}} \mid\mid \sigma^{\mathbf{R}}_{\mathbf{C}} \mid\mid \sigma^{\mathbf{C}}_{\mathbf{C}} \mid\mid \sigma^{\mathbf{d'}}_{\mathbf{C}} \stackrel{\text{def}}{=} \sigma_{\mathbf{C}}.$$

We cannot immediately split $\sigma^{\mathbf{R}}_{\mathbf{C}}$ into two, as we could for $\sigma^{\mathbf{R}}_{\mathbf{a}}$. This part of the context-wiring needs special care.

Now (4.4) can be expressed like this (rearranging terms slightly to compose matching wirings)

$$\begin{aligned}
\sigma^{\mathbf{P_0}}_{\mathbf{a}} \mid\mid \sigma^{\mathbf{d_0}}_{\mathbf{a}} \mid\mid \sigma^{\mathbf{R}}_{\mathbf{a}} \mid\mid \sigma^{\mathbf{C}}_{\mathbf{a}} \mid\mid \sigma^{\mathbf{d'}}_{\mathbf{a}} &= (\sigma^{\mathbf{P_0}}_{\mathbf{C}} \mid\mid \sigma^{\mathbf{d_0}}_{\mathbf{C}} \mid\mid \sigma^{\mathbf{R}}_{\mathbf{C}} \mid\mid \sigma^{\mathbf{C}}_{\mathbf{C}} \mid\mid \sigma^{\mathbf{d'}}_{\mathbf{C}}) \circ \\
&\quad \sigma^{\mathbf{L}}_{\mathbf{P_0}} \otimes \sigma^{\mathbf{L}}_{\mathbf{d_0}} \otimes (\sigma_{\mathbf{R_0}}\sigma^{\mathbf{L}}_{\mathbf{R_0}} \mid\mid \sigma_{\mathbf{R'}}\sigma^{\mathbf{L}}_{\mathbf{R'}}) \otimes \sigma^{\mathbf{L}}_{\mathbf{C}} \otimes \sigma^{\mathbf{L}}_{\mathbf{d'}} \\
&= \sigma^{\mathbf{P_0}}_{\mathbf{C}}\sigma^{\mathbf{L}}_{\mathbf{P_0}} \mid\mid \sigma^{\mathbf{d_0}}_{\mathbf{C}}\sigma^{\mathbf{L}}_{\mathbf{d_0}} \mid\mid \sigma^{\mathbf{R}}_{\mathbf{C}}(\sigma_{\mathbf{R_0}}\sigma^{\mathbf{L}}_{\mathbf{R_0}} \mid\mid \sigma_{\mathbf{R'}}\sigma^{\mathbf{L}}_{\mathbf{R'}}) \mid\mid \\
&\quad \sigma^{\mathbf{C}}_{\mathbf{C}}\sigma^{\mathbf{L}}_{\mathbf{C}} \mid\mid \sigma^{\mathbf{d'}}_{\mathbf{C}}\sigma^{\mathbf{L}}_{\mathbf{d'}}.
\end{aligned}$$

We note that by Lemma 4.A.4, as each pair of wirings of the parallel product have equal interfaces, they are equal (i.e., $\sigma_{\mathbf{a}}^{\mathbf{P_0}} = \sigma_{\mathbf{C}}^{\mathbf{P_0}}\sigma_{\mathbf{P_0}}^{\mathbf{L}}$, $\sigma_{\mathbf{a}}^{\mathbf{d_0}} = \sigma_{\mathbf{C}}^{\mathbf{d_0}}\sigma_{\mathbf{d_0}}^{\mathbf{L}}$, ...).

We now consider the substitution working on the redex stemming from the context, $\sigma_{\mathbf{C}}^{\mathbf{R}}$, which needs a little extra care. We have, by the arguments above, in particular $\sigma_{\mathbf{a}}^{\mathbf{R}} = \sigma_{\mathbf{C}}^{\mathbf{R}}(\sigma_{\mathbf{R_0}}\sigma_{\mathbf{R_0}}^{\mathbf{L}} \parallel \sigma_{\mathbf{R'}}\sigma_{\mathbf{R'}}^{\mathbf{L}})$.

Splitting $\sigma_{\mathbf{C}}^{\mathbf{R}}$ by the outerfaces of $\sigma_{\mathbf{R_0}}$ and $\sigma_{\mathbf{R'}}$, we get

$$\sigma_{\mathbf{a}}^{\mathbf{R}} = \sigma_{\mathbf{a}}^{\mathbf{R_0}} \parallel \sigma_{\mathbf{a}}^{\mathbf{R'}} = (\sigma_{\mathbf{C}}^{\mathbf{R_0}} \parallel \sigma_{\mathbf{C}}^{\mathbf{R'_0}} \parallel \sigma_{\mathbf{C}}^{\mathbf{R'}})(\sigma_{\mathbf{R_0}}\sigma_{\mathbf{R_0}}^{\mathbf{L}} \parallel \sigma_{\mathbf{R'}}\sigma_{\mathbf{R'}}^{\mathbf{L}}),$$

as, in general, $\sigma_{\mathbf{R_0}}$ and $\sigma_{\mathbf{R'}}$ can share names in their outerface; $\sigma_{\mathbf{C}}^{\mathbf{R'_0}}$ works on those shared names.

For ease of notation, we break our metavariable conventions for substitutions temporarily, and introduce $\phi_{\mathbf{C}}^{\mathbf{R}} = \sigma_{\mathbf{C}}^{\mathbf{R_0}} \parallel \sigma_{\mathbf{C}}^{\mathbf{R'_0}}$ and $\psi_{\mathbf{C}}^{\mathbf{R}} = \sigma_{\mathbf{C}}^{\mathbf{R'_0}} \parallel \sigma_{\mathbf{C}}^{\mathbf{R'}}$. Applying Lemma 4.A.7(4.2), we have

$$\sigma_{\mathbf{a}}^{\mathbf{R_0}} \parallel \sigma_{\mathbf{a}}^{\mathbf{R'}} = (\phi_{\mathbf{C}}^{\mathbf{R}}\sigma_{\mathbf{R_0}}\sigma_{\mathbf{R_0}}^{\mathbf{L}} \parallel \psi_{\mathbf{C}}^{\mathbf{R}}\sigma_{\mathbf{R'}}\sigma_{\mathbf{R'}}^{\mathbf{L}}),$$

and, applying Lemma 4.A.4, we find that

$$\sigma_{\mathbf{a}}^{\mathbf{R_0}} = \phi_{\mathbf{C}}^{\mathbf{R}}\sigma_{\mathbf{R_0}}\sigma_{\mathbf{R_0}}^{\mathbf{L}}, \tag{4.9}$$

$$\sigma_{\mathbf{a}}^{\mathbf{R'}} = \psi_{\mathbf{C}}^{\mathbf{R}}\sigma_{\mathbf{R'}}\sigma_{\mathbf{R'}}^{\mathbf{L}}. \tag{4.10}$$

And now, finally, we are set to utilize what we have learnt from these somewhat tedious symbol manipulations. Composing on both sides of (4.7) with $\mathsf{id} \otimes \left(\sigma_{\mathbf{C}}^{\mathbf{P_0}} \parallel \phi_{\mathbf{C}}^{\mathbf{R}}\sigma_{\mathbf{R_0}} \parallel \sigma_{\mathbf{C}}^{\mathbf{d_0}}\right)$, and using the equalities for the substitutions that we have found above, we derive

$$\begin{aligned}
&\left(\mathsf{id} \otimes \left(\sigma_{\mathbf{C}}^{\mathbf{P_0}} \parallel \phi_{\mathbf{C}}^{\mathbf{R}}\sigma_{\mathbf{R_0}} \parallel \sigma_{\mathbf{C}}^{\mathbf{d_0}}\right)\right)(\mathsf{id} \otimes \sigma_{\mathbf{P_0}}^{\mathbf{L}} \otimes \sigma_{\mathbf{R_0}}^{\mathbf{L}} \otimes \sigma_{\mathbf{d_0}}^{\mathbf{L}})p_0 = \\
&\quad \left(\mathsf{id} \otimes \left(\sigma_{\mathbf{C}}^{\mathbf{P_0}} \parallel \phi_{\mathbf{C}}^{\mathbf{R}}\sigma_{\mathbf{R_0}} \parallel \sigma_{\mathbf{C}}^{\mathbf{d_0}}\right)\right)((P_0 \otimes \mathsf{id}_{U_0})R_0 \otimes \mathsf{id}_{Z_0})d_0 \\
\iff \quad &\left(\mathsf{id} \otimes \left(\sigma_{\mathbf{a}}^{\mathbf{P_0}} \parallel \sigma_{\mathbf{a}}^{\mathbf{R_0}} \parallel \sigma_{\mathbf{a}}^{\mathbf{d_0}}\right)\right)p_0 = \left(\mathsf{id} \otimes \left((\sigma_{\mathbf{C}}^{\mathbf{P_0}} \parallel \phi_{\mathbf{C}}^{\mathbf{R}} \parallel \sigma_{\mathbf{C}}^{\mathbf{d_0}})(\mathsf{id}_{V_0} \otimes \sigma_{\mathbf{R_0}} \otimes \mathsf{id}_{Z_0})\right)\right) \circ \\
&\qquad\qquad ((P_0 \otimes \mathsf{id}_{U_0})R_0 \otimes \mathsf{id}_{Z_0})d_0
\end{aligned}$$

Analogous manipulations from (4.8) and, using in particular (4.10), ensures us that we have valid sentences

$$\sigma_{\mathbf{C}}^{\mathbf{P_0}} \parallel \sigma_{\mathbf{a}}^{\mathbf{R_0}} \parallel \sigma_{\mathbf{a}}^{\mathbf{d_0}}, \sigma_{\mathbf{R_0}}, \sigma_{\mathbf{C}}^{\mathbf{P_0}} \parallel \sigma_{\mathbf{C}}^{\mathbf{d_0}} \parallel \sigma_{\mathbf{C}}^{\mathbf{R_0}} \parallel \sigma_{\mathbf{C}}^{\mathbf{R'_0}} \vDash p_0, R_0(\to P_0, d_0,$$

and

$$\sigma_{\mathbf{a}}^{\mathbf{C}} \parallel \sigma_{\mathbf{a}}^{\mathbf{R'}} \parallel \sigma_{\mathbf{a}}^{\mathbf{d'}}, \sigma_{\mathbf{R'}}, \sigma_{\mathbf{C}}^{\mathbf{C}} \parallel \sigma_{\mathbf{C}}^{\mathbf{d'}} \parallel \sigma_{\mathbf{C}}^{\mathbf{R'}} \parallel \sigma_{\mathbf{C}}^{\mathbf{R'_0}} \vDash \bigotimes_{i=1}^{n} p_i, R'(\to \bigotimes_{i=1}^{n} P_i, d',$$

which by PAR yields the original sentence. □

Lemma (4.3.8)**.** Every valid sentence $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash a, R \hookrightarrow \mathsf{id}_\epsilon, d$ is provable using PAR and WIRING-AXIOM.

Proof. Since $C = \mathsf{id}_\epsilon$, a must have width 0, hence also $a = \mathsf{id}_\epsilon$ the only discrete bigraph with local innerface (in this case, actually ground innerface) of width 0. Similarly, R and d must have width 0, hence be id_ϵ. We analyze the wirings, in turn:

Agent By Fact 4.A.10, the agent is expressible as $(\sigma_{\mathbf{a}} \otimes \mathsf{id}_\epsilon)\mathsf{id}_\epsilon$, hence $\sigma_{\mathbf{a}} = Y$ for some Y.

Context Equally, by Fact 4.A.10, the context is $(\sigma_{\mathbf{C}} \otimes \mathsf{id}_\epsilon)(\mathsf{id}_\epsilon \otimes \mathsf{id}_U)$ (for some names U stemming from the redex (wiring)). Hence, $\sigma_{\mathbf{C}} : U \to Y$.

Redex And finally, also by Fact 4.A.10, redex is $(\sigma_{\mathbf{R}} \otimes \mathsf{id}_\epsilon)\mathsf{id}_\epsilon$, hence $\sigma_{\mathbf{R}} : \epsilon \to U = U$.

By the arguments above, the original sentence must be on the form,

$$Y, U, \sigma_{\mathbf{C}} \vDash \mathsf{id}_\epsilon, \mathsf{id}_\epsilon(\to \mathsf{id}_\epsilon, \mathsf{id}_\epsilon \quad (\sigma_{\mathbf{C}} : U \to Y).$$

By induction on the size of Y it is immediate that this sentence is derivable by $|Y|-1$ applications of PAR from sentences, which are instances of WIRING-AXIOM. □

note 4.A.21. Iterating Lemma 4.3.7 allows us to break any product of discrete primes in context and agent into prime parts, resulting in sentences of the form

$$\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash p, R(\to P, d,$$

for (discrete) primes p, P and epi substitutions $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}}$. We treat in Lemma 4.3.7 only wirings with no idle names; we handle the idle names by (iterated) application of Lemma 4.3.8, where there are no primes in the context or agent.

Hence, in the following lemmas all wiring is epi substitutions, and we are guaranteed that any splitting of wiring is unique.

Lemma (4.3.9)**.** Every valid sentence $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash p, R \hookrightarrow P, d$, with p and P prime and discrete, is provable using the LSUB rule on a valid sentence, of lesser or equal size, of the form $\sigma'_{\mathbf{a}}, \sigma'_{\mathbf{R}}, \sigma'_{\mathbf{C}} \vDash p', R \hookrightarrow P', d$, where p' and P' are discrete free primes. All substitutions mentioned above are required to be epi (i.e., with no idle names).

Proof. We have (by Fact 4.A.10 and Proposition 4.A.8),

$$p = (\widehat{\sigma_{\mathbf{q}}} \otimes \mathsf{id}_W)(X^{\mathbf{q}})q, \quad \text{and} \quad P = (\widehat{\sigma_{\mathbf{Q}}} \otimes \mathsf{id}_V)(X^{\mathbf{Q}})Q,$$

where q and Q are free primes.

Hence, given validity of the original sentence; by standard manipulations,

$$\begin{aligned}
& (\sigma_{\mathbf{a}} \otimes \mathsf{id})(\widehat{\sigma_{\mathbf{q}}} \otimes \mathsf{id}_W)(X^{\mathbf{q}})p &=& (\sigma_{\mathbf{C}} \otimes \mathsf{id}) \circ \\
& & & ((((\widehat{\sigma_{\mathbf{Q}}} \otimes \mathsf{id}_V)(X^{\mathbf{Q}})Q) \otimes \sigma_{\mathbf{R}})R \otimes \mathsf{id}_{X^{\mathbf{q}}})d \\
\iff \quad & (\sigma_{\mathbf{a}} \otimes \mathsf{id})(X)(\sigma_{\mathbf{q}} \otimes \mathsf{id}_W \otimes \mathsf{id})p &=& (\sigma_{\mathbf{C}} \otimes \mathsf{id}) \circ \\
& & & ((((X)(\sigma_{\mathbf{Q}} \otimes \mathsf{id}_V \otimes \mathsf{id})Q) \otimes \sigma_{\mathbf{R}})R \otimes \mathsf{id}_{X^{\mathbf{q}}})d \\
\iff \quad & (X)(\sigma_{\mathbf{a}} \otimes \sigma_{\mathbf{q}} \otimes \mathsf{id})p &=& (X)(\sigma_{\mathbf{C}} \otimes \sigma_{\mathbf{Q}} \otimes \mathsf{id})((Q \otimes \sigma_{\mathbf{R}})R \otimes \mathsf{id}_{X^{\mathbf{q}}})d,
\end{aligned}$$

assuming $\widehat{\sigma_{\mathbf{q}}}$ (hence, necessarily also $\widehat{\sigma_{\mathbf{Q}}}$) has outerface (X).

Now, by Lemma 4.A.18, we also have

$$(\sigma_{\mathbf{a}} \otimes \sigma_{\mathbf{q}} \otimes \mathsf{id})p = (\sigma_{\mathbf{C}} \otimes \sigma_{\mathbf{Q}} \otimes \mathsf{id})((Q \otimes \sigma_{\mathbf{R}})R \otimes \mathsf{id}_{X^{\mathbf{q}}})d.$$

Hence, choosing $\sigma'_{\mathbf{a}} = \sigma_{\mathbf{a}} \otimes \sigma_{\mathbf{q}}$, $\sigma'_{\mathbf{C}} = \sigma_{\mathbf{C}} \otimes \sigma_{\mathbf{Q}}$, $p' = q$, and $P' = Q$, we have a valid sentence, which by LSUB yields the original sentence. □

Lemma (4.3.10)**.** Every valid sentence $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash p, R \hookrightarrow Q, d$, with p and Q discrete and free primes, is provable using the MERGE, PAR (iterated), and SWITCH rules on valid sentences, each of lesser or equal size, and each on one of two forms:

- $\sigma'_{\mathbf{a}}, \sigma'_{\mathbf{R}}, \sigma'_{\mathbf{C}} \vDash p^{\mathbf{N}}, \mathsf{id} \hookrightarrow P^{\mathbf{N}}, e$, where $p^{\mathbf{N}}$ and $P^{\mathbf{N}}$ are free discrete primes,
- $\sigma'_{\mathbf{a}}, \sigma'_{\mathbf{R}}, \sigma'_{\mathbf{C}} \vDash m, S \hookrightarrow M, e$, where m and M are free discrete molecules.

All substitutions mentioned above are required to be epi (i.e., with no idle names).

Proof. By Fact 4.A.10, and the propositions on normal forms (for namediscrete primes, [DB06, Theorem 1(2)] and for discrete, Proposition 4.A.8), we see that,

$$\begin{aligned} p &= (merge \otimes \mathsf{id}) \bigotimes_i^k m_i, \\ Q &= (merge \otimes \mathsf{id}) \left(\left(\bigotimes_i^n \ulcorner \alpha_i \urcorner \right) \otimes \bigotimes_i^m M_i \right) \pi, \\ \text{and} \quad R &= \bigotimes_i^l P_i, \end{aligned}$$

as R is regular. Then by Fact 4.A.10 and Proposition 4.A.13, there exists a substitution $\sigma^{\mathbf{L}}$, s.t.,

$$\sigma_{\mathbf{a}} = \sigma_{\mathbf{C}}(\sigma_{\mathbf{R}} \otimes \mathsf{id}_V \otimes \mathsf{id}_Z)\sigma^{\mathbf{L}}, \tag{4.11}$$

and

$$\begin{aligned} &(\mathsf{id} \otimes \sigma^{\mathbf{L}})(merge \otimes \mathsf{id}) \bigotimes_i^k m_i \\ &\quad = (merge \otimes \mathsf{id}) \left(\left(\left(\left(\bigotimes_i^n \ulcorner \alpha_i \urcorner \right) \otimes \bigotimes_i^m M_i \right) \pi \otimes \mathsf{id}_U \right) \left(\bigotimes_i^l P_i \right) \otimes \mathsf{id}_Z \right) d. \end{aligned} \tag{4.12}$$

Working on (4.12) we get (applying the push-through-lemma [DB06]),

$$
\begin{aligned}
&(\mathsf{id} \otimes \sigma^{\mathbf{L}})\,(\mathit{merge} \otimes \mathsf{id}) \bigotimes_i^k m_i \\
&\quad = (\mathit{merge} \otimes \mathsf{id}) \left(\left(\left(\bigotimes_i^n \ulcorner \alpha_i \urcorner \right) \otimes \left(\bigotimes_i^m M_i \right) \otimes \mathsf{id}_U \right) \left(\bigotimes_i^l P_{\pi^{-1}(i)} \right) \otimes \mathsf{id}_Z \right) \overline{\pi} d \\
&\quad = (\mathit{merge} \otimes \mathsf{id}) \left(\left(\bigotimes_i^n (\ulcorner \alpha_i \urcorner \otimes \mathsf{id}) P_{\pi^{-1}(i)} \right) \otimes \left(\bigotimes_i^m (M_i \otimes \mathsf{id}) S_i \right) \otimes \mathsf{id}_Z \right) \overline{\pi} d \\
&\quad = (\mathit{merge} \otimes \mathsf{id}) \left(\left(\bigotimes_i^n \left((\ulcorner \alpha_i \urcorner \otimes \mathsf{id}) P_{\pi^{-1}(i)} \otimes \mathsf{id} \right) e_i \right) \otimes \bigotimes_i^m \left((M_i \otimes \mathsf{id}) S_i \otimes \mathsf{id} \right) f_i \right), \\
&\quad = (\mathit{merge} \otimes \mathsf{id}) \left(\left(\bigotimes_i^n t_i \right) \otimes \bigotimes_i^m u_i \right), \qquad (4.13)
\end{aligned}
$$

where we define

$$
\bigotimes_i^m S_i \stackrel{\text{def}}{=} \bigotimes_{i=n}^l P_{\pi^{-1}(i)}, \quad \left(\bigotimes_i^n e_i \right) \otimes \bigotimes_i^m f_i \stackrel{\text{def}}{=} \overline{\pi}\, d,
$$

$$
\begin{aligned}
\forall i \in n \quad & t_i \stackrel{\text{def}}{=} \left((\ulcorner \alpha_i \urcorner \otimes \mathsf{id}) P_{\pi^{-1}(i)} \otimes \mathsf{id} \right) e_i, \text{ and} \\
\forall i \in m \quad & u_i \stackrel{\text{def}}{=} \left((M_i \otimes \mathsf{id}) S_i \otimes \mathsf{id} \right) f_i.
\end{aligned}
$$

Each e_i, S_i and f_i are determined by the innerfaces of the corresponding $P_{\pi^{-1}(i)}$, M_i and S_i, respectively.

Since $\sigma^{\mathbf{L}}$ shares global outerface with the product of the t_i's and u_i's, we can partition it according to the outerfaces of those primes.

$$
\left(\bigotimes_i^n \sigma_i^{\mathbf{t}} \right) \otimes \bigotimes_i^m \sigma_i^{\mathbf{u}} = \sigma \qquad (4.14)
$$

We do not want to break the redex and parameter entirely into molecules in this step. Instead, as each m_i represent a top-level node of the agent, it is easily seen that we can permute and partition the m_i's into n products, p_i, and m molecules, n_i, which match the place-graph structure in each t_i and u_i, respectively:

$$
\left(\bigotimes_i^n p_i \right) \otimes \bigotimes_i^m n_i \stackrel{\text{def}}{=} (\rho \otimes \mathsf{id}) \bigotimes_i^k m_i \qquad (4.15)
$$

for some permutation ρ. Since $\sigma^{\mathbf{L}}$ was also partitioned according to the outerfaces of t_i and u_i, the outerface of each p_i will match the innerface of each $\sigma_i^{\mathbf{t}}$, and the outerface of each n_i will match the innerface of each $\sigma_i^{\mathbf{u}}$.

More formally, we can easily prove a little lemma for ground molecules and primes, saying that

$$(merge \otimes \mathsf{id}) \bigotimes_i^n m_i = (merge \otimes \mathsf{id}) \bigotimes_i^m p_i$$

iff (for some ρ, k_i)

$$\forall\, i \in m \quad p_i = (merge \otimes \mathsf{id}) \left(\bigotimes_j^{k_i} m_{\rho(j)} \right).$$

The point is, that when we have one of the products under a *merge* split wholly into molecules, we can express each prime p_i just by taking each of the molecules m_i that match it.

Consequentially, by combining (4.14) and (4.15), we find that

$$(\forall i \in n)\ (\sigma_i^{\mathbf{t}} \otimes \mathsf{id}) p_i = t_i \tag{4.16}$$

$$\text{and} \quad (\forall i \in m)\ (\sigma_i^{\mathbf{u}} \otimes \mathsf{id}) n_i = u_i \tag{4.17}$$

Now we concern ourselves with the wirings of equation (4.11). We split $\sigma_{\mathbf{a}}$ by the innerface according to the innerfaces of the $\sigma_i^{\mathbf{t}}$'s and $\sigma_i^{\mathbf{u}}$'s, to get

$$\sigma_{\mathbf{a}} = \left(\Vert_i^n \sigma_{i,\mathbf{a}}^{\mathbf{t}} \right) \,||\, \left(\Vert_i^m \sigma_{i,\mathbf{a}}^{\mathbf{u}} \right).$$

We also split $\sigma_{\mathbf{C}}(\sigma_{\mathbf{R}} \otimes \mathsf{id}_V \otimes \mathsf{id}_Z)$ accordingly

$$\begin{aligned}
&\sigma_{\mathbf{C}}(\sigma_{\mathbf{R}} \otimes \mathsf{id}_V \otimes \mathsf{id}_Z) \left(\left(\bigotimes_i^n \sigma_i^{\mathbf{t}} \right) \otimes \bigotimes_i^m \sigma_i^{\mathbf{u}} \right) \\
&\quad = \sigma_{\mathbf{C}} \left(\left(\Vert_i^n \sigma_{i,\mathbf{R}}^{\mathbf{t}} \right) \,||\, \left(\Vert_i^m \sigma_{i,\mathbf{R}}^{\mathbf{u}} \right) \otimes \mathsf{id}_V \otimes \mathsf{id}_Z \right) \left(\left(\bigotimes_i^n \sigma_i^{\mathbf{t}} \right) \otimes \bigotimes_i^m \sigma_i^{\mathbf{u}} \right) \\
&\quad = \left(\left(\Vert_i^n \sigma_{i,\mathbf{C}}^{\mathbf{t}} (\sigma_{i,\mathbf{R}}^{\mathbf{t}} \otimes \mathsf{id}) \right) \,||\, \left(\Vert_i^m \sigma_{i,\mathbf{C}}^{\mathbf{u}} (\sigma_{i,\mathbf{R}}^{\mathbf{u}} \otimes \mathsf{id}) \right) \right) \left(\left(\bigotimes_i^n \sigma_i^{\mathbf{t}} \right) \otimes \bigotimes_i^m \sigma_i^{\mathbf{u}} \right)
\end{aligned} \tag{4.18}$$

in the first step splitting $\sigma_{\mathbf{R}}$, s.t.

$$\sigma_{\mathbf{R}} = \left(\Vert_i^n \sigma_{i,\mathbf{R}}^{\mathbf{t}} \right) \,||\, \left(\Vert_i^m \sigma_{i,\mathbf{R}}^{\mathbf{u}} \right),$$

and in the second step using Lemma 4.A.7(4.3) (iterated) to split shared wiring in $\sigma_{\mathbf{C}}$ into $n+m$ substitutions according to $\sigma_{i,\mathbf{R}}^{\mathbf{t}}$ and $\sigma_{i,\mathbf{R}}^{\mathbf{u}}$.

As each $\sigma_{i,\mathbf{a}}^{\mathbf{t}}$ and $\sigma_{i,\mathbf{a}}^{\mathbf{u}}$ has the same interfaces as $\sigma_{i,\mathbf{C}}^{\mathbf{t}}(\sigma_{i,\mathbf{R}}^{\mathbf{t}} \otimes \mathsf{id})$ and $\sigma_{i,\mathbf{C}}^{\mathbf{u}}(\sigma_{i,\mathbf{R}}^{\mathbf{u}} \otimes \mathsf{id})$, respectively, by Lemma 4.A.4 they are equal. By equational manipulations identical

to those concluding the proof of Lemma 4.3.7, we can infer from (4.16), (4.17), and (4.18), that,

$$(\forall i \in n)\ \left(\sigma^{\mathbf{t}}_{i,\mathbf{a}} \otimes \mathsf{id}\right) p_i = \left(\sigma^{\mathbf{t}}_{i,\mathbf{C}} \left(\sigma^{\mathbf{t}}_{i,\mathbf{R}} \otimes \mathsf{id}\right) \otimes \mathsf{id}\right) t_i$$
$$\text{and } (\forall i \in m)\ \left(\sigma^{\mathbf{u}}_{i,\mathbf{a}} \otimes \mathsf{id}\right) n_i = \left(\sigma^{\mathbf{u}}_{i,\mathbf{C}} \left(\sigma^{\mathbf{u}}_{i,\mathbf{R}} \otimes \mathsf{id}\right) \otimes \mathsf{id}\right) u_i.$$

We check and find, that we have $n + m$ valid sentences which by PAR (iterated) yields the original sentence by a single application of MERGE.

Finally, we note that each sentence corresponding to the first n equations,

$$\sigma^{\mathbf{t}}_{i,\mathbf{a}}, \sigma^{\mathbf{t}}_{i,\mathbf{R}}, \sigma^{\mathbf{t}}_{i,\mathbf{C}} \vDash p_i, P_{\pi^{-1}(i)}(\rightarrow \ulcorner \alpha_i \urcorner, e_i,$$

is a consequence by SWITCH of the sentence,

$$\sigma^{\mathbf{t}}_{i,\mathbf{a}}, \mathsf{id}_\epsilon, \sigma^{\mathbf{t}}_{i,\mathbf{C}} \left(\alpha_i \sigma_{\pi^{-1}(i)} \otimes \sigma^{\mathbf{t}}_{i,\mathbf{R}} \otimes \mathsf{id}\right) \vDash p_i, \mathsf{id}(\rightarrow P^{\mathbf{N}}_{\pi^{-1}(i)}, e_i$$

for $P_i = (\widehat{\sigma_i} \otimes \mathsf{id})\,(Y_i) P^{\mathbf{N}}_i$; where $P^{\mathbf{N}}_{\pi^{-1}(i)}$ is discrete, free and prime. □

Lemma (4.3.11)**.** Every valid sentence $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash m, R \hookrightarrow M, d$, with m and M free discrete molecules, is provable using the ION rule on a valid sentence $\sigma'_{\mathbf{a}}, \sigma'_{\mathbf{R}}, \sigma'_{\mathbf{C}} \vDash p, R \hookrightarrow P, d$, of lesser size, where p and P are discrete primes. All substitutions mentioned above are required to be epi (i.e., with no idle names).

Proof. By Fact 4.A.10, and the normal form for free discrete molecules (cf. [DB06]), we can express m as

$$m = \left(K_{\vec{y}(\vec{X})} \otimes \mathsf{id}\right) q,$$

and M as

$$M = \left(K_{\vec{u}(\vec{W})} \otimes \mathsf{id}\right) Q,$$

where q and Q are namediscrete and prime.[3]

Assuming validity of the original sentence, by Fact 4.A.10 and Proposition 4.A.13, there exists a substitution $\sigma^{\mathbf{L}}$, s.t.,

$$\sigma_{\mathbf{a}} = \sigma_{\mathbf{C}}(\mathsf{id}_V \otimes \sigma_{\mathbf{R}} \otimes \mathsf{id}_Z)\sigma^{\mathbf{L}}. \tag{4.19}$$

$$\text{and} \quad (\mathsf{id} \otimes \sigma^{\mathbf{L}}) \left(K_{\vec{y}(\vec{X})} \otimes \mathsf{id}\right) q = \left(\left(\left(K_{\vec{u}(\vec{W})} \otimes \mathsf{id}\right) Q \otimes \mathsf{id}_U\right) R \otimes \mathsf{id}_Z\right) d. \tag{4.20}$$

Since the K-node on the righthand-side is wired discretely to $\vec{u}$'s, the lefthand-side must match this. Hence, we must have

$$\sigma^{\mathbf{L}} = \vec{u}/\vec{y} \otimes \sigma^{\mathbf{L}}_1.$$

[3]Note, that for this proof only, we break our convention of only eliding interfaces on local identities. The identities introduced in the expressions for m and M are global, but as they are inessential for the analysis, to ease the notational clutter, we shall elide the interfaces for these two identities.

We define,

$$\widehat{\sigma}_{(\vec{X})} \stackrel{\text{def}}{=} \bigotimes_{i}^{|\vec{X}|}(x_i)/(X_i) \quad \text{and} \quad \widehat{\phi}_{(\vec{W})} \stackrel{\text{def}}{=} \bigotimes_{i}^{|\vec{W}|}(x_i)/(W_i),$$

such that we can decompose the ions, and express (4.20) as,

$$\left(K_{\vec{u}(\vec{x})} \otimes \mathsf{id}\right)\left(\widehat{\sigma}_{(\vec{X})} \otimes \sigma_1^{\mathbf{L}}\right) q = \left(K_{\vec{u}(\vec{x})} \otimes \mathsf{id}\right)\left(\left(\left(\widehat{\phi}_{(\vec{W})} \otimes \mathsf{id}\right) Q \otimes \mathsf{id}_U\right) R \otimes \mathsf{id}_Z\right) d. \tag{4.21}$$

Applying Proposition 4.A.9, we have,

$$\left(\widehat{\sigma}_{(\vec{X})} \otimes \sigma_1^{\mathbf{L}}\right) q = \left(\left(\left(\widehat{\phi}_{(\vec{W})} \otimes \mathsf{id}\right) Q \otimes \mathsf{id}_U\right) R \otimes \mathsf{id}_Z\right) d. \tag{4.22}$$

We split $\sigma_{\mathbf{a}}$ and $\sigma_{\mathbf{C}}$ into the wiring linked to the K-node (i.e., the $\vec{y}$'s, and the $\vec{u}$'s, respectively) and a remainder.

$$\sigma_{\mathbf{a}}^{\mathbf{K}} \,||\, \sigma_{\mathbf{a}}^{\mathbf{r}} \stackrel{\text{def}}{=} \sigma_{\mathbf{a}},$$
$$\text{and} \quad \sigma_{\mathbf{C}}^{\mathbf{K}} \vec{u}/\vec{y} \,||\, \sigma_{\mathbf{C}}^{\mathbf{r}} \left(\sigma_{\mathbf{R}} \otimes \mathsf{id}\right) \sigma_1^{\mathbf{L}} \stackrel{\text{def}}{=} \sigma_{\mathbf{C}} \left(\mathsf{id}_V \otimes \sigma_{\mathbf{R}} \otimes \mathsf{id}_Z\right) \left(\vec{u}/\vec{y} \otimes \sigma_1^{\mathbf{L}}\right).$$

From (4.19), with the help of Lemma 4.A.4, then also, as the interfaces of the corresponding substitutions match,

$$\sigma_{\mathbf{a}}^{\mathbf{K}} = \sigma_{\mathbf{C}}^{\mathbf{K}} \vec{u}/\vec{y},$$
$$\text{and} \quad \sigma_{\mathbf{a}}^{\mathbf{r}} = \sigma_{\mathbf{C}}^{\mathbf{r}} \left(\mathsf{id}_V \otimes \sigma_{\mathbf{R}} \otimes \mathsf{id}_Z\right) \sigma_1^{\mathbf{L}}.$$

Composing on both sides of (4.22) with $(\mathsf{id} \otimes \sigma_{\mathbf{C}}^{\mathbf{r}} (\mathsf{id}_V \otimes \sigma_{\mathbf{R}} \otimes \mathsf{id}_Z))$ and using the equalities defined above, we find,

$$\left(\mathsf{id} \otimes \sigma_{\mathbf{a}}^{\mathbf{r}}\right)\left(\widehat{\sigma}_{(\vec{X})} \otimes \mathsf{id}\right) q$$
$$= \left(\mathsf{id} \otimes \sigma_{\mathbf{C}}^{\mathbf{r}} \left(\mathsf{id}_V \otimes \sigma_{\mathbf{R}} \otimes \mathsf{id}_Z\right)\right)\left(\left(\left(\widehat{\phi}_{(\vec{W})} \otimes \mathsf{id}\right) Q \otimes \mathsf{id}_U\right) R \otimes \mathsf{id}_Z\right) d.$$

Choosing $\sigma'_{\mathbf{a}} = \sigma_{\mathbf{a}}^{\mathbf{r}}$, $\sigma'_{\mathbf{R}} = \sigma_{\mathbf{R}}$, $\sigma'_{\mathbf{C}} = \sigma_{\mathbf{C}}^{\mathbf{r}}$, $p = \left(\widehat{\sigma}_{(\vec{X})} \otimes \mathsf{id}\right) q$, and

$$P = \left(\left(\left(\widehat{\phi}_{(\vec{W})} \otimes \mathsf{id}\right) Q \otimes \mathsf{id}_U\right) R \otimes \mathsf{id}_Z\right) d,$$

we have a valid sentence, which by ION yields the original sentence. □

Lemma (4.3.12)**.** Every valid sentence $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash p, \mathsf{id} \hookrightarrow P, e$, with p and P free discrete primes, is provable using the MERGE and PAR (iterated) rules on valid sentences of equal or lesser size, which are either instances of rule PRIME-AXIOM or of the form $\sigma'_{\mathbf{a}}, \sigma'_{\mathbf{R}}, \sigma'_{\mathbf{M}} \vDash m, R \hookrightarrow M, d$. All substitutions mentioned above are required to be epi (i.e., with no idle names).

Proof. (*Sketch*) The proof is similar to the proof of Lemma 4.3.10, but simpler as the redex is a (local) identity. This also implies $\sigma_{\mathbf{R}} = \mathsf{id}_\epsilon$, since for some local identity $(\mathsf{id} \otimes \sigma_{\mathbf{R}})R$ must be defined. For the concretions in P, instead of arriving at sentences which are derivable by SWITCH, each such sentence is an instance of PRIME-AXIOM. We treat these basecases in more detail below.

The analysis for the wirings is simplified as $\sigma_{\mathbf{R}}$ is an identity. We find (focusing only on the wiring concerned with the concretions of P),

$$(\forall i \in n)\ \sigma^{\mathbf{t}}_{i,\mathbf{a}} = \sigma^{\mathbf{t}}_{i,\mathbf{C}}\sigma^{\mathbf{t}}_i. \tag{4.23}$$

Performing an analysis analogous to that in the proof of Lemma 4.3.10 we have (instead of (4.16))

$$(\forall i \in n)\ (\sigma^{\mathbf{t}}_i \otimes \mathsf{id})p_i = (\alpha_i \otimes \mathsf{id} \otimes \mathsf{id}_{Z_i})\,(\ulcorner U_i \urcorner \otimes \mathsf{id}_{Z_i})\,e_i, \tag{4.24}$$

for renamings $\alpha_i : X_i \to U_i$, and primes $e_i : \langle Z_i \uplus U_i \rangle$ stemming from the parameter.

From (4.24), we can derive an expression for e_i in terms of p_i. For each $i \in n$, we have

$$\begin{array}{rrcl} & (\sigma^{\mathbf{t}}_i \otimes \mathsf{id})p_i & = & (\alpha_i \otimes \mathsf{id} \otimes \mathsf{id}_{Z_i})\,(\ulcorner U_i \urcorner \otimes \mathsf{id}_{Z_i})\,e_i \\ \iff & (\alpha^{-1} \otimes \mathsf{id} \otimes \mathsf{id}_{Z_i})(\sigma^{\mathbf{t}}_i \otimes \mathsf{id})p_i & = & (\ulcorner U_i \urcorner \otimes \mathsf{id}_{Z_i})\,e_i \\ \iff & (X_i)(\alpha^{-1} \otimes \mathsf{id} \otimes \mathsf{id}_{Z_i})(\sigma^{\mathbf{t}}_i \otimes \mathsf{id})p_i & = & e_i \end{array} \tag{4.25}$$

Since we know e_i is discrete, from (4.25) we see that $\sigma^{\mathbf{t}}_i$ must be a renaming on all names not linked to U_i (to be localized by the abstraction). Hence, for each $i \in n$,

$$\sigma^{\mathbf{t}}_i = \sigma^{\mathbf{t}'}_i \otimes \beta_i,$$

for $\beta_i :\to Z_i$ and $\sigma^{\mathbf{t}'}_i :\to U_i$.

By composing on both sides of (4.24) with $(\sigma^{\mathbf{t}}_{i,\mathbf{C}} \otimes \mathsf{id})$ and utilizing the forms for e_i from (4.25)

$$\begin{aligned}(\forall i \in n)\ &(\sigma^{\mathbf{t}}_{i,\mathbf{C}}(\sigma^{\mathbf{t}'}_i \otimes \beta_i) \otimes \mathsf{id})p_i \\ &= (\sigma^{\mathbf{t}}_{i,\mathbf{C}} \otimes \mathsf{id})(\ulcorner \alpha_i \urcorner \otimes \mathsf{id}_{Z_i})(X_i)(\alpha_i^{-1} \otimes \mathsf{id} \otimes \mathsf{id}_{Z_i})(\sigma^{\mathbf{t}}_i \otimes \mathsf{id})p_i\end{aligned} \tag{4.26}$$

To be strict, we also need to refer to equation (4.23) to see that $\sigma^{\mathbf{t}}_{i,\mathbf{C}}(\sigma^{\mathbf{t}'}_i \otimes \beta_i)$ is, in fact, wiring stemming from the agent. In other words, (4.23) gives an equation of the agent wiring in terms of the wiring from the context and parameter.

The n equations stated in (4.26), means that we have n valid sentences on the form

$$\sigma^{\mathbf{t}}_{i,\mathbf{C}}(\sigma^{\mathbf{t}'}_i \otimes \beta_i), \mathsf{id}_\epsilon, \sigma^{\mathbf{t}}_{i,\mathbf{C}} \vDash p_i, \mathsf{id}_{(X_i)}(\to \ulcorner \alpha_i \urcorner, (X_i)(\alpha_i^{-1} \otimes \mathsf{id} \otimes \mathsf{id}_{Z_i})(\sigma^{\mathbf{t}}_i \otimes \mathsf{id})p_i.$$

It is easily verified that those sentences are instances of PRIME-AXIOM (choosing, in particular, τ equal to $\alpha_i^{-1}\sigma^{\mathbf{t}'}_i$). □

Chapter 5

An Implementation of Bigraph Matching

Abstract

We describe a provably sound and complete matching algorithm for bigraphical reactive systems. The algorithm has been implemented in our BPL Tool, a first implementation of bigraphical reactive systems. We describe the tool and present a concrete example of how it can be used to simulate a model of a mobile phone system in a bigraphical representation of the polyadic π calculus.

Preface This chapter contains a paper currently submitted for publication. The paper was co-authored with Arne J. Glenstrup, Lars Birkedal, and Espen Højsgaard from the IT University of Copenhagen.

5.1 Introduction

The theory of bigraphical reactive systems [JM04] provides a general meta-model for describing and analyzing mobile and distributed ubiquituous systems. Bigraphical reactive systems form a graphical model of computation in which graphs embodying both locality and connectivity can be reconfigured using *reaction rules*. So far it has been shown how to use the theory for recovering behavioural theories for various process calculi [JM04, Jen06, LM04] and how to use the theory for modelling context-aware systems [BDE$^+$06].

In this paper we describe the core part of our BPL Tool, a first prototype implementation of bigraphical reactive systems, which can be used for experimenting with bigraphical models.

The main challenge of implementing the dynamics of bigraphical reactive systems is the *matching problem*, that is, to determine for a given bigraph and reaction rule whether and how the reaction rule can be applied to rewrite the bigraph. When studying the matching problem in detail, one finds that it is a surprisingly tricky problem (it is related to the NP-complete graph embedding problem).

Therefore we decided early on to study the matching problem quite formally and base our prototype implementation on a provably correct specification. In previous work [BDGM07, DGBM07], we gave a sound and complete inductive characterization of the matching problem for bigraphs. Our inductive characterization was based on normal form theorems for binding bigraphs [DB06].

In the present paper we extend the inductive characterization from graphs to a *term* representation of bigraphs. A single bigraph can be represented by several structurally congruent bigraph terms. Using an equational theory for bigraph terms [DB06], we essentially get a non-deterministic matching algorithm operating on bigraph terms. However, such an algorithm will be wildly non-deterministic and we thus provide an alternative, but still provably sound and complete, characterization of matching on terms, which is more suited for mechanically finding matching. In particular, it spells out how and where to make use of structural congruences.

We have implemented the resulting algorithm in our BPL Tool, which we briefly describe in Section 5.6. We also present an example of a bigraphical reactive system, an encoding of the polyadic π calculus, and show how it can be used to simulate a simple model of a mobile phone system.

Bigraphical reactive systems are related to general graph transformation systems; [EEPT06] provide a recent comprehensive overview of graph transformation systems. In particular, bigraph matching is related to the general graph pattern matching (GPM) problem, so general GPM algorithms might also be applicable to bigraphs [Ull76, Fu97, LV02, Zün94]. As an alternative to implementing matching for bigraphs, one could try to formalize bigraphical reactive systems as graph transformation systems and then use an existing implementation of graph transformation systems. Some promising steps in this direction have been taken [SS05], but they have so far fallen short of capturing precisely all the aspects of binding bigraphs. For a more detailed account of related work, in particular on relations between BRSs, graph transformations, term rewriting and term graph rewriting, see the Thesis of [Dam06, Section 6].

The remainder of this paper is organized as follows. In Section 5.2 we give an informal presentation of bigraphical reactive systems and in Section 5.3 we present our matching algorithm: we first recall the graph-based inductive characterization, then we develop a term-based inductive characterization, which forms the basis for our implementation of matching. In Section 5.4 we describe how our implementation deals with the remaining nondeterminism and in Section 5.5 we discuss a couple of auxiliary technologies needed for the implementation of the term-based matching rules. In Section 5.6 we finally describe the BPL Tool and present an example use of it. We conclude and discuss future work in Section 5.7.

5.2 Bigraphs and Reactive Systems

In the following, we present bigraphs informally; for a formal definition, see the work by [JM04] and [DB06].

5.2.1 Concrete Bigraphs

A concrete binding bigraph G consists of a *place graph* $G^{\mathbf{P}}$ and a *link graph* $G^{\mathbf{L}}$. The place graph is an ordered list of trees indicating *location*, with roots $r_0, \ldots, r_n$, nodes $v_0, \ldots, v_k$, and a number of special leaves $s_0, \ldots, s_m$ called *sites*, while the link graph is a general graph over the node set $v_0, \ldots, v_k$ extended with *inner names* $x_0, \ldots, x_l$, and equipped with hyper edges, indicating *connectivity*.

We usually illustrate the place graph by nesting nodes, as shown in the upper part of Figure 5.1 (ignore for now the interfaces denoted by " $: \cdot \to \cdot$"). A *link*

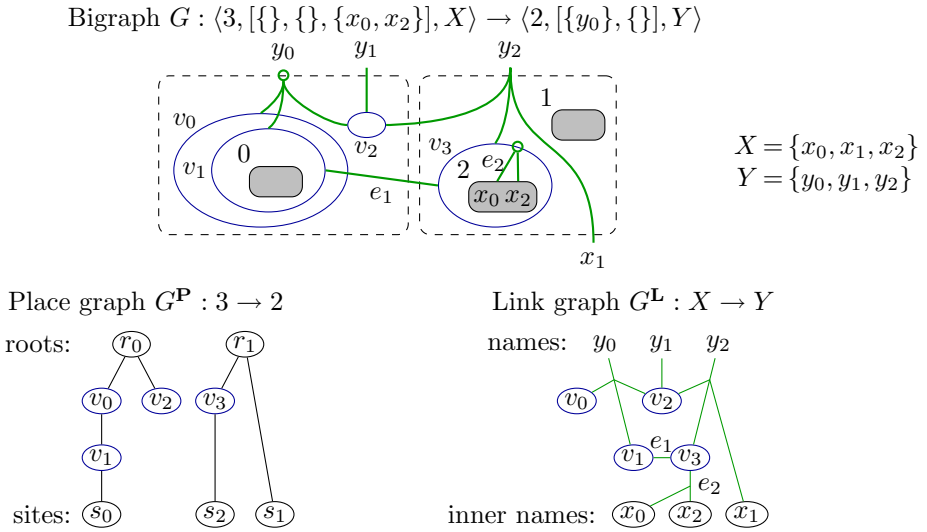


Figure 5.1: Example bigraph illustrated by nesting and as place and link graph.

is a hyper edge of the link graph, either an internal *edge* e or a *name* y. Links that are names are called *open*, those that are edges are called *closed*. Names and inner names can be *global* or *local*, the latter being located at a specific root or site, respectively. In Figure 5.1, y_0 is located at r_0, indicated by a small ring, and x_0 and x_2 are located at s_2, indicated by writing them within the site. Global names like y_1 and y_2 are drawn anywhere at the top, while global inner names like x_1 are drawn anywhere at the bottom. A link, including internal edges like e_2 in the figure, can be located with one *binder* (the ring), in which case it is a *bound link*, otherwise it is *free*. However, a bound link must satisfy the *scope rule*, a simple structural requirement that all points (cf. next paragraph) of the link lie within its location (in the place graph), except for the binder itself. This prevents y_2 and e_1 in the example from being bound.

5.2.2 Controls and Signatures

Every node v has a *control* K, indicated by $v : K$, which determines a binding and free arity. In the example of Figure 5.1, we could have $v_i : K_i, i = 0, 1, 2, 3$, where arities are given by $K_0 : 1$, $K_1 : 2$, $K_2 : 3$, $K_3 : 1 \to 2$, using $K : f$ as a shorthand for

$K : 0 \to f$. The arities determine the number of bound and free *ports* of the node, to which bound and free links, respectively, are connected. Ports and inner names are collectively referred to as *points*.

In addition to arity, each control is assigned a *kind*, either atomic, active or passive, and describe nodes according to their control kinds. We require that atomic nodes contain no nodes except sites; any site being a descendant of a passive node is *passive*, otherwise it is *active*. If all sites of a bigraph G are active, G is *active*.

A collection of controls with their associated kinds and arities is referred to as a *signature*.

5.2.3 Abstract Bigraphs

While concrete bigraphs with named nodes and internal edges are the basis of bigraph theory [JM04], our prime interest is in *abstract bigraphs,* equivalence classes of concrete bigraphs that differ only in the names of nodes and internal edges[1]. Abstract bigraphs are illustrated with their node controls (see Figure 5.13 in Section 5.6). In what follows, "bigraph" will thus mean "abstract bigraph."

5.2.4 Interfaces

Every bigraph G has two *interfaces* I and J, written $G : I \to J$, where I is the *inner face* and J the *outer face*. An interface is a triple $\langle m, \vec{X}, X \rangle$, where m is the *width* (the number of sites or roots), X the entire set of local and global names, and $\vec{X}$ indicates the locations of each local name, cf. Figure 5.1. We let $\epsilon = \langle 0, [], \{\} \rangle$; when $m = 1$ the interface is *prime*, and if all $x \in X$ are located by $\vec{X}$, the interface is *local*. As in the work by [Mil04] we write $G : \ \to J$ or $G : I \to$ for $G : I \to J$ when we are not concerned about about I or J, respectively.

A bigraph $G : I \to J$ is called *ground*, or an *agent*, if $I = \epsilon$, *prime* if I is local and J prime, and a *wiring* if $m = n = 0$, where m and n are the widths of I and J, respectively. For $I = \langle m, \vec{X}, X \rangle$, bigraph $\mathsf{id}_I : I \to I$ consists of m roots, each root r_i containing just one site s_i, and a link graph linking each inner name $x \in X$ to name x.

5.2.5 Discrete and Regular Bigraphs

We say that a bigraph is *discrete* iff every free link is a name and has exactly one point. The virtue of discrete bigraphs is that any connectivity by internal edges must be bound, and node ports can be accessed individually by the names of the outer face. Further, a bigraph is *name-discrete* iff it is discrete and every bound link is either an edge, or (if it is a name) has exactly one point. Note that name-discrete implies discrete.

A bigraph is *regular* if, for all nodes v and sites i, j, k with $i \leq j \leq k$, if i and k are descendants of v, then j is also a descendant of v. Further, for roots $r_{i'}$ and $r_{j'}$, and all sites i and j where i is a descendant of $r_{i'}$ and j of $r_{j'}$, if $i \leq j$ then

[1] Formally, we also disregard *idle* edges: edges not connected to anything.

$i' \leq j'$. The bigraphs in the figures are all regular, the permutation in Table 5.1 is not. The virtue of regular bigraphs is that permutations can be avoided when composing them from basic bigraphs.

5.2.6 Product and Composition

For bigraphs G_1 and G_2 that share no names or inner names, we can make the *tensor product* $G_1 \otimes G_2$ by juxtaposing their place graphs, constructing the union of their link graphs, and increasing the indexes of sites in G_2 by the number of sites of G_1. We write $\bigotimes_i^n G_i$ for the iterated tensor $G_0 \otimes \cdots \otimes G_{n-1}$, which, in case $n = 0$, is id_ϵ.

The *parallel product* $G_1 \,||\, G_2$ is like the tensor product, except global names can be shared: if y is shared, all points of y in G_1 and G_2 become the points of y in $G_1 \,||\, G_2$.

The *prime product* $G_1 \,|\, G_2$ is like the parallel product, except the result has just one root (also when G_1 and G_2 are wirings), produced by merging any roots of G_1 and G_2 into one.

We can *compose* bigraphs $G_2 : I \to I'$ and $G_1 : I' \to J$, yielding bigraph $G_1 \circ G_2 : I \to J$, by plugging the sites of G_1 with the roots of G_2, eliminating both, and connecting names of G_2 with inner names of G_1. In the following, we will omit the '$\circ$', and simply write $G_1 G_2$ for composition, letting it bind tighter than tensor product.

5.2.7 Notation, Basic Bigraphs, and Abstraction

In the sequel, we will use the following notation: $\uplus$ denotes union of sets required to be disjoint; we write $\{\vec{Y}\}$ for $Y_0 \uplus \cdots \uplus Y_{n-1}$ when $\vec{Y} = Y_0, \ldots Y_{n-1}$, and similarly $\{\vec{y}\}$ for $\{y_0, \ldots, y_{n-1}\}$. For interfaces, we write n to mean $\langle n, [\emptyset, \ldots, \emptyset], \emptyset\rangle$, X to mean $\langle 0, [], X\rangle$, $\langle X\rangle$ to mean $\langle 1, [\{\}], X\rangle$ and (X) to mean $\langle 1, [X], X\rangle$.

Any bigraph can be constructed by applying composition, tensor product and abstraction to identities (on all interfaces) and a set of basic bigraphs, shown in Table 5.1 [DB06]. For permutations, when used in any context, $\pi_{\vec{X}} G$ or $G\pi_{\vec{X}}$, $\vec{X}$ is given entirely by the interface of G; in these cases we simply write $\pi_{\vec{X}}$ as π.

Given a prime P, the abstraction operation localises a subset of its outer names. Note that the scope rule is necessarily respected since the inner face of a prime P is required to be local, so all points of P are located within its root. The abstraction operator is denoted by $(\cdot)\cdot$ and reaches as far right as possible.

For a renaming $\alpha : X \to Y$, we write $\ulcorner\alpha\urcorner$ to mean $(\alpha \otimes \mathsf{id}_1)\ulcorner X\urcorner$, and when $\sigma : U \to Y$, we let $\widehat{\sigma} = (Y)(\sigma \otimes \mathsf{id}_1)\ulcorner U\urcorner$. We write substitutions $\vec{y}/[\emptyset, \ldots, \emptyset] : \epsilon \to Y$ as Y.

Note that $[]/[] = /\emptyset = \pi_0 = \mathsf{id}_\epsilon$ and $merge_1 = \ulcorner\emptyset\urcorner = \pi_1 = \mathsf{id}_1$, where π_i is the nameless permutation of width i.

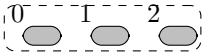
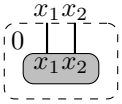
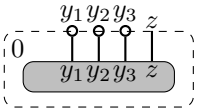
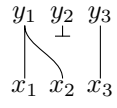
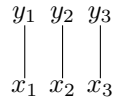
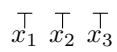
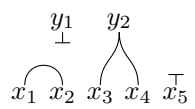
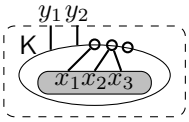
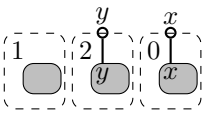
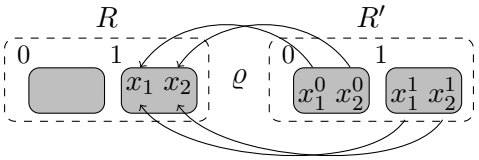
	Notation	*Example*	
Merge	$merge_n : n \to 1$	$merge_3 =$	
Concretion	$\ulcorner X \urcorner : (X) \to \langle X \rangle$	$\ulcorner \{x_1, x_2\} \urcorner =$	
Abstraction	$(Y)P : I \to \langle 1, [Y], Z \uplus Y \rangle$	$(\{y_1, y_2\})(\{y_3\})\ulcorner \{y_1, y_2, y_3, z\} \urcorner =$	
Substitution σ	$\vec{y}/\vec{X} : X \to Y$	$[y_1, y_2, y_3]/[\{x_1, x_2\}, \{\}, \{x_3\}] =$	
Renaming α, β	$\vec{y}/\vec{x} : X \to Y$	$[y_1, y_2, y_3]/[x_1, x_2, x_3] =$	
Closure	$/X : X \to \{\}$	$/\{x_1, x_2, x_3\} =$	
Wiring ω	$(\mathsf{id} \otimes /Z)\sigma : X \to Y$	$(\mathsf{id}_{\{y1,y2\}} \otimes /\{z_1, z_2\})$ $[y_1, z_1, y_2, z_2]/$ $[\{\}, \{x_1, x_2\}, \{x_3, x_4\}, \{x_5\}] =$	
Ion	$K_{\vec{y}(\vec{X})} : (\{\vec{X}\}) \to \langle \{\vec{y}\} \rangle$	$\mathsf{K}_{[y_1,y_2]([\{x_1\},\{x_2,x_3\},\{\}])} =$	
Permutation $\pi_{\vec{X}}$	$\{i \mapsto j, \ldots\}_{\vec{X}}$ $: \langle m, \vec{X}, X \rangle \to \langle m, \pi(\vec{X}), X \rangle$	$\{0 \mapsto 2, 1 \mapsto 0, 2 \mapsto 1\}_{[\{x\},\emptyset,\{y\}]} =$	

Table 5.1: Basic bigraphs, the abstraction operation, and variables ranging over bigraphs.

5.2.8 Bigraphical Reactive Systems

Bigraphs in themselves model two essential parts of context: locality and connectivity. To model also *dynamics,* we introduce *bigraphical reactive systems* (BRS) as a collection of *rules.* Each rule $R \xrightarrow{\varrho} R'$ consists of a regular *redex* $R : I \to J$, a *reactum* $R' : I' \to J$, and an *instantiation* ϱ, mapping each site of R' to a site of R, and mapping local names in I' to those of I, as illustrated in Figure 5.2. Interfaces



$$\varrho = [1\&[x_1^0 \mapsto x_1, x_2^0 \mapsto x_2], \\ 1\&[x_1^1 \mapsto x_1, x_2^1 \mapsto x_2]]$$

Figure 5.2: A reaction rule

$I = \langle m, \vec{X}, X \rangle$ and $I' = \langle m', \vec{X}', X' \rangle$ must be local, and are related by $X'_i = X_{\varrho(i)}$, where ϱ must be a bijection between X'_i and $X_{\varrho(i)}$. We illustrate ϱ by '$i := j$', whenever $\varrho(i) = j \neq i$, or, alternatively, by listing $[\varrho(0), \ldots, \varrho(m' - 1)]$. Given an instantiation ϱ and a discrete bigraph $d = d_0 \otimes \cdots \otimes d_k$ with prime d_i's, we let $\varrho(d) = d_{\varrho(0)} \otimes \cdots \otimes d_{\varrho(k)}$, allowing copying, discarding and reordering parts of d.

Given an agent a, a *match* of redex R is a decomposition $a = C(\mathsf{id}_Z \otimes R)d$, with active context C and discrete parameter d with its global names Z. Dynamics is achieved by transforming a into a new agent $a' = C(\mathsf{id}_Z \otimes R')d'$, where $d' = \varrho(d)$, cf. Figure 5.3. This definition of a match is as given by [JM04], except that we

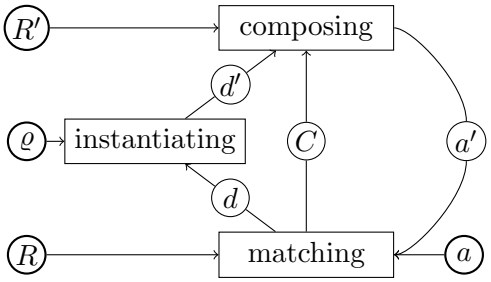


Figure 5.3: The reaction cycle

here also require R to be regular. This restriction to regular redexes R simplifies the inductive characterization of matching without limiting the set of possible reactions, as sites in R and R' can be renumbered to render R regular.

5.2.9 Bigraph Terms and Normal Forms

Expressing bigraphs as terms composed by product, composition and abstraction over basic bigraph terms, [DB06] showed that bigraphs can be expressed on normal forms uniquely up to certain permutations and renamings. Further, they showed

equivalence of term and bigraph equality, which will allow us in Section 5.3.2 to base our implementation on terms rather than graphs.

In this work, we use the normal forms shown in Figure 5.4, which are unique up to permutation of S_i's and renaming of names not visible on the interfaces. Regular bigraphs are expressed by the same forms, with the permutations removed.

$$
\begin{array}{rll}
M & ::= (\mathsf{id}_Z \otimes K_{\vec{y}(\vec{X})})N & \textit{molecule} \\
S & ::= \ulcorner \alpha \urcorner \quad | \quad M & \textit{singular top-level node} \\
G & ::= (\mathsf{id}_Y \otimes \mathit{merge}_n)(\bigotimes_i^n S_i)\pi & \textit{global discrete prime} \\
N & ::= (X)G & \textit{name-discrete prime} \\
P, Q & ::= (\mathsf{id}_Z \otimes \widehat{\sigma})N & \textit{discrete prime} \\
D & ::= \alpha \otimes (\bigotimes_i^n P_i)\pi & \textit{discrete bigraph} \\
B & ::= (\omega \otimes \mathsf{id}_{(\vec{X})})D & \textit{binding bigraph}
\end{array}
$$

Figure 5.4: Normal forms for binding bigraphs

5.3 Matching

In this section we develop the theory underlying the matching engine. First, we express matching inference using a graph representation; this representation is the basis for correctness proofs. Then we transform it to be based on a term representation more amenable to implementation, but in such a way that correctness is preserved—achieving an implementation proven correct in great detail.

5.3.1 Inferring Matches using a Graph Representation

For simplicity, we will first consider just place graphs to explain the basic idea behind matching inference.

Matching Place Graphs

A place graph match is captured by a matching sentence:

Definition 5.3.1 (Matching Sentence for Place Graphs)**.** A *matching sentence* for place graphs is a 4-tuple of bigraphs $a, R\text{`} \to C, d$, all are regular except C, with a and d ground. A sentence is *valid* iff $a = CRd$.

We infer place graph matching sentences using the inference system given in Figure 5.5. Traversing an inference tree bottom-up, the agent is decomposed, while constructing the context, using the ION, MERGE and PAR rules. The PERM rule permutes redex parts to align tensor factors with corresponding agent factors.

At the point in the agent where a redex root should match, leaving a site in the context, the SWITCH rule is applied, switching the roles of the context and redex. This allows the remaining rules to be reused (above the switch rule) for checking

$$\text{PRIME-AXIOM}\ \frac{}{p, \mathsf{id} \hookrightarrow \mathsf{id}, p} \qquad \text{ION}\ \frac{p, R \hookrightarrow P, d}{Kp, R \hookrightarrow KP, d} \qquad \text{SWITCH}\ \frac{p, \mathsf{id} \hookrightarrow P, d}{p, P \hookrightarrow \mathsf{id}_1, d}$$

$$\text{PAR}\ \frac{a, R \hookrightarrow C, d \quad b, S \hookrightarrow D, e}{a \otimes b, R \otimes S \hookrightarrow C \otimes D, d \otimes e} \qquad \text{PERM}\ \frac{a, \bigotimes_i^n P_{\pi^{-1}(i)} \hookrightarrow C, \overline{\pi} d}{a, \bigotimes_i^n P_i \hookrightarrow C\pi, d}$$

$$\text{MERGE}\ \frac{a, R \hookrightarrow C, d}{merge\, a, R \hookrightarrow merge\, C, d}$$

Figure 5.5: Inference rules for deriving place graph matches

that the redex matches the agent. When a site in the redex is reached, whatever is left of the agent should become (a part of) the parameter—this is captured by the PRIME-AXIOM rule.

For a match with a redex $R : m \to n$ consisting of n nontrivial (i.e., non-identity) primes, the inference tree will contain m applications of PRIME-AXIOM and n applications of SWITCH. Further, between any leaf and the root of the inference tree, SWITCH will be applied at most once. The structure of a matching inference tree will thus generally be as illustrated in Figure 5.6; rules applied above SWITCH

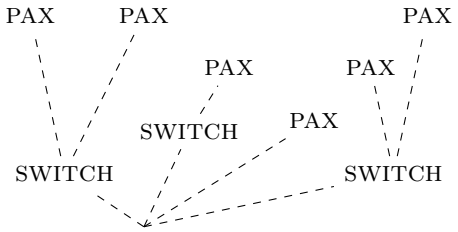


Figure 5.6: The general structure of an inference tree for matching (PAX = PRIME-AXIOM)

match agent and redex structure, while rules applied below match agent and context structure.

Matching Binding Bigraphs

Turning now to consider binding bigraphs, we extend the matching sentences to cater for links:

Definition 5.3.2 (Matching Sentence for Binding Bigraphs)**.** A *(binding bigraph) matching sentence* is a 7-tuple of bigraphs: $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R\text{`} \to C, d$, where a, R, C and d are discrete with local inner faces, all regular except C, with a and d ground. It is valid iff $(\mathsf{id} \otimes \omega_{\mathbf{a}})a = (\mathsf{id} \otimes \omega_{\mathbf{C}})(\mathsf{id}_{Z \uplus V} \otimes C)(\mathsf{id}_Z \otimes (\mathsf{id} \otimes \omega_{\mathbf{R}})R)d$.

This definition separates the wirings, leaving local wiring in a, R, C and d, while keeping global wiring of agent, redex and context in $\omega_{\mathbf{a}}$, ω_R and ω_C, respectively.

The validity property shows how a valid matching sentence relates to a match, as illustrated in Figure 5.7.

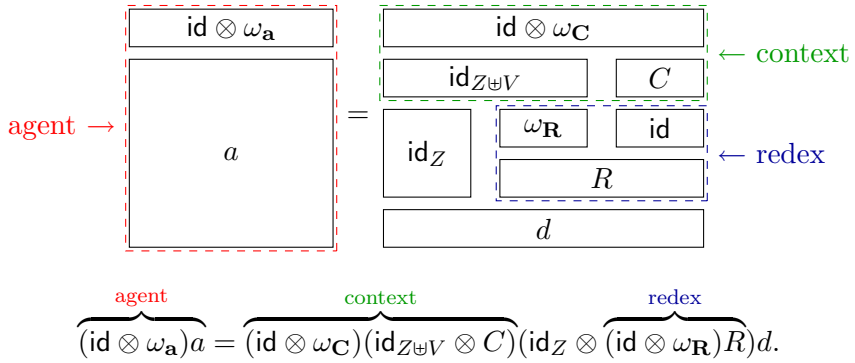


Figure 5.7: Decomposition of the bigraphs of a valid matching sentence

To reach a system for inferring valid matching sentences for binding bigraphs, we simply augment the place graph rules with wirings as shown in Figure 5.8, and add three rules for dealing with purely wiring constructs, shown in Figure 5.9. A detailed explanation of the rules is available in the literature [DGBM07], along with proofs of soundness and completeness of the inference system.

5.3.2 Inferring Matches using a Term Representation

While the graph representation of matching sentences is useful for constructing a relatively simple inference system amenable to correctness proofs, it is not sufficient for an implementation based on syntax, that is, bigraph terms. One bigraph can be represented by several different bigraph terms that are structurally congruent by the axiom rules: $a = a \otimes \mathsf{id}_0 = merge_1\, a$, $a \otimes (b \otimes c) = (a \otimes b) \otimes c$ and $merge(a \otimes b) = merge(b \otimes a)$. If, for instance, we were to match agent $a = merge((\mathsf{K} \otimes \mathsf{L}) \otimes \mathsf{M})$ with redex $R = \mathsf{K}$, we would first need to apply the axioms to achieve $R = merge((\mathsf{K} \otimes \mathsf{id}_0) \otimes \mathsf{id}_0)$ before being able to apply the MERGE and PAR rules.

In the following, we recast the matching sentences to be tuples of 3 wirings and 4 bigraph *terms* $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_C \vdash a, R \rightsquigarrow C, d$, with the same restrictions and validity as before, interpreting the terms as the bigraphs they represent. Given this, adding just this one rule would be sufficient to achieve completeness of the inference system:

$$\text{STRUCT}\ \frac{a \equiv a' \qquad R \equiv R' \qquad C \equiv C' \qquad h \equiv h' \qquad \omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a', R' \rightsquigarrow C', h'}{\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, R \rightsquigarrow C, h}$$

The STRUCT rule says that we can apply structural congruence to rewrite any term a, R, C or h to a term denoting the same bigraph. With the help of the equational theory for determining bigraph isomorphism on the term level [DB06], we have essentially a nondeterministic algorithm for matching bigraph terms—implementable in say, Prolog. A brief glance at the equational theory, shows us, though, that the associative and commutative properties of the basic operators of the language

$$\text{PRIME-AXIOM}\ \frac{\sigma : W \uplus U \to \quad \beta : Z \to U \quad \alpha : V \to W \quad \tau : X \to V \quad p : \langle X \uplus Z \rangle}{\sigma(\beta \otimes \alpha\tau), \mathsf{id}_\epsilon, \sigma \vdash p, \mathsf{id}_{(V)} \hookrightarrow \ulcorner \alpha \urcorner, (\beta \otimes \widehat{\tau})(X)\, p}$$

$$\text{ION}\ \frac{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash ((\vec{v})/(\vec{X}) \otimes \mathsf{id}_U)\, p, R \hookrightarrow ((\vec{v})/(\vec{Z}) \otimes \mathsf{id}_W)\, P, d \qquad \alpha = \vec{y}/\vec{u} \qquad \sigma : \{\vec{y}\} \to}{\sigma \,||\, \omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \sigma\alpha \,||\, \omega_{\mathbf{C}} \vdash (K_{\vec{y}(\vec{X})} \otimes \mathsf{id}_U)\, p, R \hookrightarrow (K_{\vec{u}(\vec{Z})} \otimes \mathsf{id}_W)\, P, d}$$

$$\text{SWITCH}\ \frac{\begin{array}{c}\sigma : W \to U \quad P : \ \to \langle W \uplus Y \rangle \quad d : \langle m, \vec{X}, X \uplus Z \rangle \\ \omega_{\mathbf{a}}, \mathsf{id}_\epsilon, \omega_{\mathbf{C}}(\sigma \otimes \omega_{\mathbf{R}} \otimes \mathsf{id}_Z) \vdash p, \mathsf{id} \hookrightarrow P, d\end{array}}{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash p, (\widehat{\sigma} \otimes \mathsf{id}_Y)(W)\, P \hookrightarrow \ulcorner U \urcorner, d}$$

$$\text{PAR}\ \frac{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \,||\, \omega \vdash a, R \hookrightarrow C, d \qquad \omega_{\mathbf{b}}, \omega_{\mathbf{S}}, \omega_{\mathbf{D}} \,||\, \omega \vdash b, S \hookrightarrow D, e}{\omega_{\mathbf{a}} \,||\, \omega_{\mathbf{b}}, \omega_{\mathbf{R}} \,||\, \omega_{\mathbf{S}}, \omega_{\mathbf{C}} \,||\, \omega_{\mathbf{D}} \,||\, \omega \vdash a \otimes b, R \otimes S \hookrightarrow C \otimes D, d \otimes e}$$

$$\text{PERM}\ \frac{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, \bigotimes_i^m P_{\pi^{-1}(i)} \hookrightarrow C, (\overline{\pi} \otimes \mathsf{id})\, d}{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, \bigotimes_i^m P_i \hookrightarrow C\pi, d}$$

$$\text{MERGE}\ \frac{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \hookrightarrow C, d}{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash (\mathit{merge} \otimes \mathsf{id}_Y)\, a, R \hookrightarrow (\mathit{merge} \otimes \mathsf{id}_X)\, C, d}$$

Figure 5.8: Place graph rules (shaded) augmented for deriving binding bigraph matches

$$\text{WIRING-AXIOM}\ \frac{}{y, X, y/X \vdash \mathsf{id}_\epsilon, \mathsf{id}_\epsilon \hookrightarrow \mathsf{id}_\epsilon, \mathsf{id}_\epsilon}$$

$$\text{ABSTR}\ \frac{\begin{array}{c}\sigma_{\mathbf{a}} : Z \to W \quad p : \langle Z \uplus Y \rangle \quad \sigma_{\mathbf{C}} : U \to W \quad P : \ \to \langle U \uplus X \rangle \\ \sigma_{\mathbf{a}} \otimes \omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \sigma_{\mathbf{C}} \otimes \omega_{\mathbf{C}} \vdash p, R \hookrightarrow P, d\end{array}}{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash (\widehat{\sigma_{\mathbf{a}}} \otimes \mathsf{id}_Y)(Z)p, R \hookrightarrow (\widehat{\sigma_{\mathbf{C}}} \otimes \mathsf{id}_X)(U)P, d}$$

$$\text{CLOSE}\ \frac{\begin{array}{c}\sigma_{\mathbf{a}} : \ \to U \uplus Y_R \qquad \sigma_{\mathbf{R}} : \ \to V \uplus Y_R \qquad \sigma_{\mathbf{C}} : \ \to W \uplus Y_C \\ \sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \mathsf{id}_{Y_{\mathbf{R}}} \otimes \sigma_{\mathbf{C}} \vdash a, R \hookrightarrow C, d\end{array}}{(\mathsf{id}_U \otimes /(Y_{\mathbf{R}} \uplus Y_{\mathbf{C}}))\sigma_{\mathbf{a}}, (\mathsf{id}_V \otimes /Y_{\mathbf{R}})\sigma_{\mathbf{R}}, (\mathsf{id}_W \otimes /Y_{\mathbf{C}})\sigma_{\mathbf{C}} \vdash a, R \hookrightarrow C, d}$$

Figure 5.9: Added inference rules for deriving binding bigraph matches

would yield a wildly nondeterminstic inference system, since we would need to apply structural congruence between every step to infer a match. This is reminiscent of the problems in implementing *rewriting logic*, that is, term rewriting modulo a set of static equivalences [Mau, CDE$^+$01, CDE$^+$03]. Consequentially, we abandon the fully general STRUCT rule. For the purposes of stating the completeness theorem below, we shall need to refer to sentences derived from the ruleset for bigraphs (i.e., from section 5.3.1) recast to terms with the help of the STRUCT rule above. We shall write such sentences $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, R \rightsquigarrow_{\mathrm{S}} C, h$ for wirings $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}}$ and terms a, R, C and h.

Definition 5.3.3. For wirings $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}}$ and terms a, R, C and h, sentences $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, R \rightsquigarrow_{\mathrm{S}} C, h$ range over sentences derived from the rules of Figure 5.9—reading a, R, C and h as terms—extended with the STRUCT rule.

Instead, to specialize the characterization into a more efficient algorithm for mechanically finding matches, we define *normal inferences*. Normal inferences are classes of inferences that are complete in the sense that all valid matching sentences can be inferred, but suitably restricted, such that inferences can be built mechanically. In particular, normal inference definitions for term matching need to spell out *how* and *where* to apply structural congruence. As a main trick, we utilize a variant of the normal forms proven complete for binding bigraphs (cf. Section 5.2.9), lending us a set of uniform representations of classes of bigraphs based directly on terms for bigraphs; we define normal inferences that require each inference to start by rewriting the term to be on normal form.

Before giving the format for normal inferences, we incorporate structural congruence axioms into PRODUCT and MERGE rules. We derive rules for iterated tensor product and permutations under merge, arriving at the inference system shown in Figure 5.10. In this inference system, the terms in the conclusion of every rule except DNF is in some normal form as given by Figure 5.4, where e is a discrete prime (p) or global discrete prime (g). An expression $[\![t]\!]^G$ means term t expressed on G-normal form—for instance, $[\![\ulcorner \alpha \urcorner]\!]^G$ means $(\mathsf{id}_Y \otimes \mathit{merge}_1)(\bigotimes_i^1 \ulcorner \alpha \urcorner)$—and similarly for the remaining normal forms. The expression $\bar{\varrho}(n, m)$ denotes the set of n-m-partitions. An n-m-partition ϱ is a partition of $\{0, \ldots, n-1\}$ into m (possibly empty) subsets, and for $i \in m$, ϱ_i is the ith subset. Given a metavariable $\mathcal{X}$, $\overline{\mathcal{X}}$ ranges over iterated tensor products of $\mathcal{X}$'es. As indicated by the superscript, rules $\mathrm{PER}^{\mathcal{E}}$, $\mathrm{PAR}^{\mathcal{E}}_{\mathbf{n}}$ and $\mathrm{PAR}^{\mathcal{E}}_{\equiv}$ can be used either on discrete primes p and P or global discrete primes g and G.

The main differences from the preceding inference system is that we have replaced the binary PAR rule by two iterative PAR rules, $\mathrm{PAR}^{\mathcal{E}}_{\mathbf{n}}$ and $\mathrm{PAR}^{\mathcal{E}}_{\equiv}$, and specialised the MERGE rule into a rule, MER, that makes the partitioning of children in an agent node explicit. The $\mathrm{PAR}^{\mathcal{E}}_{\equiv}$ rule splits up an iterated tensor product into a number of products matching agent factors, while $\mathrm{PAR}^{\mathcal{E}}_{\mathbf{n}}$ performs the actual inductive inference on each of the factors. (Note, by the way, that $\mathrm{PAR}^{\mathcal{E}}_{\equiv}$ and $\mathrm{MER}^{\mathcal{E}}_{\equiv}$ correspond just to particular instances of the STRUCT-rule, that we abandoned above.)

Furthermore, note that the usage of the previous WIRING-AXIOM-rule for intro-

$$\text{PAX}\ \frac{\sigma : W \uplus Z \to \quad \alpha : V \to W \quad \tau : X \to V \quad g : \langle X \uplus Z \rangle}{\sigma(\mathsf{id}_Z \otimes \alpha\tau), \mathsf{id}_\epsilon, \sigma \vdash g, [\![\mathsf{id}_{(V)}]\!]^{\overline{P}} \rightsquigarrow [\![\ulcorner \alpha \urcorner]\!]^G, [\![(\mathsf{id}_Z \otimes \widehat{\tau})(X)g]\!]^{\overline{P}}}$$

$$\text{ION}\ \frac{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash (\mathsf{id} \otimes (\vec{v})/(\vec{X}))n, \overline{P} \rightsquigarrow (\mathsf{id} \otimes (\vec{v})/(\vec{Z}))N, \overline{q} \quad \alpha = \vec{y}/\vec{u} \quad \sigma : \{\vec{y}\} \to}{(\sigma \mid\mid \sigma^{\mathbf{a}}), \sigma^{\mathbf{R}}, (\sigma\alpha \mid\mid \sigma^{\mathbf{C}}) \vdash [\![(\mathsf{id}_U \otimes K_{\vec{y}(\vec{X})})n]\!]^G, \overline{P} \rightsquigarrow [\![(\mathsf{id}_W \otimes K_{\vec{u}(\vec{Z})})N]\!]^G, \overline{q}}$$

$$\text{SWX}\ \frac{\sigma : W \to U \quad G : \ \to \langle W \uplus Y \rangle \quad \overline{q} : \langle m, \vec{X}, X \uplus Z \rangle \quad \sigma^{\mathbf{a}}, \mathsf{id}_\epsilon, \sigma^{\mathbf{C}}(\mathsf{id}_Z \otimes \sigma \otimes \sigma^{\mathbf{R}}) \vdash g, [\![\bigotimes_i^n \mathsf{id}_{(X_i)}]\!]^{\overline{P}} \rightsquigarrow G, \overline{q}}{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash g, [\![(\mathsf{id}_Y \otimes \widehat{\sigma})(W)G]\!]^{\overline{P}} \rightsquigarrow [\![\ulcorner U \urcorner]\!]^G, \overline{q}}$$

$$\text{PAR}^{\mathcal{E}}_{\mathbf{n}}\ \frac{\sigma' : I_{\mathbf{R}} \to I_{\mathbf{a}} \quad (\forall i \in n)\ \sigma_i^{\mathbf{a}}, \sigma_i^{\mathbf{R}}, \sigma \mid\mid \sigma_i^{\mathbf{C}} \vdash e_i, \overline{P}_i \rightsquigarrow E_i, \overline{q}_i}{\big(I_{\mathbf{a}} \mid\mid \|_i^n \sigma_i^{\mathbf{a}}\big), \big(I_{\mathbf{R}} \mid\mid \|_i^n \sigma_i^{\mathbf{R}}\big), \big(\sigma' \mid\mid \sigma \mid\mid \|_i^n \sigma_i^{\mathbf{C}}\big) \vdash \bigotimes_i^n e_i, \bigotimes_i^n \overline{P}_i \rightsquigarrow \bigotimes_i^n E_i, \bigotimes_i^n \overline{q}_i}$$

$$\text{PAR}^{\mathcal{E}}_{\equiv}\ \frac{P'_{ij} = P_{j+\sum_{r\in i} l_r} \quad q'_{ij} = q_{j+\sum_{r\in i} k_r} \quad P'_{ij} : \langle k_{ij}, \vec{X}_{ij} \rangle \to \quad k_i = \sum_{j\in l_i} k_{ij} \quad \sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash \bigotimes_i^n e_i, \bigotimes_i^n \bigotimes_j^{l_i} P'_{ij} \rightsquigarrow \bigotimes_i^n E_i, \bigotimes_i^n \bigotimes_j^{k_i} q'_{ij}}{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash \bigotimes_i^n e_i, \bigotimes_i^m P_i \rightsquigarrow \bigotimes_i^n E_i, \bigotimes_i^{m'} q_i}$$

$$\text{PER}^{\mathcal{E}}\ \frac{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash \overline{e}, \bigotimes_i^n Q_{\pi^{-1}(i)} \rightsquigarrow \overline{E}, \bigotimes_i^m q_{\bar{\pi}^{-1}(i)}}{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash \overline{e}, \bigotimes_i^n Q_i \rightsquigarrow \overline{E}\pi, \bigotimes_i^m q_i}$$

$$\text{MER}\ \frac{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash \bigotimes_i^m (\mathsf{id} \otimes \mathit{merge}) \bigotimes_{j\in\varrho_i, \varrho\in\bar{\varrho}(n,m)} m_j, \overline{P} \rightsquigarrow (\bigotimes_i^m [\![S_{\pi^{-1}(i)}]\!]^G)\bar{\pi}, \overline{q}}{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash (\mathsf{id} \otimes \mathit{merge}) \bigotimes_i^n m_i, \overline{P} \rightsquigarrow (\mathsf{id} \otimes \mathit{merge}) \bigotimes_i^m S_i, \overline{q}}$$

$$\text{ABS}\ \frac{\sigma_L^{\mathbf{a}} : Z \to W \quad \sigma_L^{\mathbf{C}} : U \to W \quad G : \ \to \langle U \uplus X \rangle \quad \sigma_L^{\mathbf{a}} \otimes \sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma_L^{\mathbf{C}} \otimes \sigma^{\mathbf{C}} \vdash g, \overline{P} \rightsquigarrow G, \overline{q}}{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash (\mathsf{id} \otimes \widehat{\sigma_L^{\mathbf{a}}})(Z)g, \overline{P} \rightsquigarrow (\mathsf{id} \otimes \widehat{\sigma_L^{\mathbf{C}}})(U)G, \overline{q}}$$

$$\text{CLO}\ \frac{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \mathsf{id}_{Y_{\mathbf{R}}} \otimes \sigma^{\mathbf{C}} \vdash \overline{p}, \overline{P} \rightsquigarrow \overline{Q}\pi, \overline{q}}{(\mathsf{id} \otimes /(Y_{\mathbf{R}} \uplus Y_{\mathbf{C}}))\sigma^{\mathbf{a}}, (\mathsf{id} \otimes /Y_{\mathbf{R}})\sigma^{\mathbf{R}}, (\mathsf{id} \otimes /Y_{\mathbf{C}})\sigma^{\mathbf{C}} \vdash \overline{p}, \overline{P} \rightsquigarrow \overline{Q}\pi, \overline{q}}$$

$$\text{DNF}\ \frac{a \equiv \overline{p} \quad R \equiv \overline{P} \quad C \equiv \overline{Q}\pi \quad h \equiv \overline{q} \quad \overline{p}, \overline{P}, \overline{Q}, \overline{q} \text{ are on normal form} \quad R \text{ is regular} \quad \omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash \overline{p}, \overline{P} \rightsquigarrow \overline{Q}\pi, \overline{q}}{\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, R \rightsquigarrow C, h}$$

Figure 5.10: Inference rules for binding bigraph terms

ducing idle linkage has been inlined to a side-condition on a slightly generalized PAR-rule (i.e., the $\text{PAR}^{\mathcal{E}}_{\mathbf{n}}$-rule). The σ' in that rule allows us to introduce idle linkage in redex and agent, and link them in context; as previously allowed by the WIRING-AXIOM-rule. Hence, $\text{PAR}^{\mathcal{E}}_{\mathbf{n}}$ also serves as an axiom, introducing 0-ary products of id_ϵ's on G- and P-normal forms.

While this inference system is more explicit about partitioning tensor products (in the MER and $\text{PAR}^{\mathcal{E}}_{\equiv}$ rules), there is still a lot of nondeterministic choice left in the *order* in which the rules can be applied. To limit this, we define normal inferences based, essentially, on the order rules were used in the proof of completeness [DGBM07]. We derive a sufficient order that still preserves completeness:

Definition 5.3.4 (Normal Inference)**.** A *normal inference* is a derivation using the term matching rules of Figure 5.10 in the order specified by the grammar given in Figure 5.11.

$$\mathcal{D}_{\mathbf{G}} ::= \begin{cases} \text{PAX}\ \dfrac{}{\cdots} \\ \text{ABS}\ \dfrac{\mathcal{D}_{\mathbf{P}}}{\cdots} \\ \text{ION}\ \dfrac{}{\cdots} \\ \text{SWX}\ \dfrac{\mathcal{D}'_{\mathbf{P}}}{\cdots} \end{cases} \qquad \mathcal{D}'_{\mathbf{G}} ::= \begin{cases} \text{PAX}\ \dfrac{}{\cdots} \\ \text{ABS}\ \dfrac{\mathcal{D}'_{\mathbf{P}}}{\cdots} \\ \text{ION}\ \dfrac{}{\cdots} \end{cases}$$

$$\mathcal{D}_{\mathbf{P}} ::= \begin{cases} \mathcal{D}_{\mathbf{G}} \\ \text{PAR}^{\mathbf{G}}_{\mathbf{n}}\ \dfrac{\mathcal{D}_{\mathbf{G}} \quad \cdots \quad \mathcal{D}_{\mathbf{G}}}{\cdots} \\ \text{PAR}^{\mathbf{G}}_{\equiv}\ \dfrac{}{\cdots} \\ \text{PER}^{\mathbf{G}}\ \dfrac{}{\cdots} \\ \text{MER}\ \dfrac{}{\cdots} \end{cases} \qquad \mathcal{D}'_{\mathbf{P}} ::= \begin{cases} \mathcal{D}'_{\mathbf{G}} \\ \text{PAR}^{\mathbf{G}}_{\mathbf{n}}\ \dfrac{\mathcal{D}'_{\mathbf{G}} \quad \cdots \quad \mathcal{D}'_{\mathbf{G}}}{\cdots} \\ \text{PAR}^{\mathbf{G}}_{\equiv}\ \dfrac{}{\cdots} \\ \text{PER}^{\mathbf{G}}\ \dfrac{}{\cdots} \\ \text{MER}\ \dfrac{}{\cdots} \end{cases}$$

$$\mathcal{D}_{\mathbf{B}} ::= \begin{array}{l} \text{ABS}\ \dfrac{\mathcal{D}_{\mathbf{P}}}{\cdots} \quad \cdots \quad \text{ABS}\ \dfrac{\mathcal{D}_{\mathbf{P}}}{\cdots} \\ \text{PAR}^{\mathbf{P}}_{\mathbf{n}}\ \dfrac{}{\cdots} \\ \text{PAR}^{\mathbf{P}}_{\equiv}\ \dfrac{}{\cdots} \\ \text{PER}^{\mathbf{P}}\ \dfrac{}{\cdots} \\ \text{CLO}\ \dfrac{}{\cdots} \\ \text{DNF}\ \dfrac{}{\cdots} \end{array}$$

Figure 5.11: Grammar for normal inferences for binding bigraphs with start symbol $\mathcal{D}_{\mathbf{B}}$

Now we can give the main theorem stating that normal inferences are sufficient for finding all valid matches. The following theorem states formally for every sentence derivable with the ruleset for bigraphs recast to bigraph terms by extending with STRUCT, that such a sentence is also derivable as a normal inference.

Theorem 5.3.5 (Normal inferences are sound and complete)**.** *For wirings* $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}}$ *and terms* a, R, C, d*, we can infer* $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, R \rightsquigarrow_{\mathrm{S}} C, d$ *iff we can infer* $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, R \rightsquigarrow C, d$ *using a normal inference.*

Proof. (*Sketch*) By induction over the structure of the derivation of the sentence $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, R \rightsquigarrow_{\mathrm{S}} C, d$. We case on the last rule used to conclude this sentence. By the induction hypothesis (IH), we can conclude a normal derivation of the sentence used for concluding $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, R \rightsquigarrow_{\mathrm{S}} C, d$.

STRUCT: By IH, we can construct a normal derivation of $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a', R' \rightsquigarrow C', d'$, with $a = a'$, $R' = R$, $C' = C$ and $d' = d$. This normal derivation can be used directly to conclude also $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a', R' \rightsquigarrow C', d'$.

PRIME-AXIOM: We produce the needed normal inference by starting with an application of PAX, which introduces the needed prime bigraphs and wiring—that is, each term being equal up to structural congruence to the sentence concluded with PRIME-AXIOM. Now we proceed to build the needed normal inference by a building first a $\mathcal{D}_{\mathbf{P}}$ and then a $\mathcal{D}_{\mathbf{B}}$-inference. All steps add only term structure to match a particular normal form, while not changing the denotation of the terms.

ION: By IH, we can construct a normal derivation of $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash ((\vec{v})/(\vec{X}) \otimes \mathsf{id}_U)p, R \rightsquigarrow ((\vec{v})/(\vec{Z}) \otimes \mathsf{id}_W)P, d$. For this case, we have to unroll that normal derivation up across the $\mathcal{D}_{\mathbf{B}}$ production except for the last ABS-step, concluding with a $\mathrm{PAR}_1^{\mathbf{P}}$ step (since we know p and P are prime). We now have a $\mathcal{D}_{\mathbf{P}}$ normal inference with an added ABS-step, which we can use for concluding an ION-step introducing our needed ion. Referring to the grammar in Figure 5.11, we see that this produces a $\mathcal{D}_{\mathbf{G}}$-inference, which we have to lead through two series of PAR-PER-MER steps (and one ABS-step), to produce a full normal inference.

SWITCH: This case needs a little extra care. First, we point out two properties of normal derivations: (i) any $\mathcal{D}_{\mathbf{G}}$ and $\mathcal{D}_{\mathbf{P}}$ inference without SWX is also a $\mathcal{D}'_{\mathbf{G}}$ or $\mathcal{D}'_{\mathbf{P}}$ inference, respectively; and, (ii) any sentence, $\omega_{\mathbf{a}}, \mathsf{id}_\epsilon, \omega_{\mathbf{C}} \vdash a, \mathsf{id} \rightsquigarrow C, h$ has a normal derivation with no SWX-steps. Both are easily verified.

Now, by the IH we can construct a normal derivation of a sentence $\omega_{\mathbf{a}}, \mathsf{id}_\epsilon, \omega_{\mathbf{C}}(\sigma \otimes \omega_{\mathbf{R}} \otimes \mathsf{id}_Z) \vdash p, \mathsf{id} \rightsquigarrow P, d$ for global P. By property (ii), we can assume that this normal derivation does not contain any applications of SWX. We unroll this normal derivation up across the whole $\mathcal{D}_{\mathbf{B}}$ production, . This leaves us with a $\mathcal{D}_{\mathbf{P}}$-type normal derivation, which by property (i), we can use also as $\mathcal{D}'_{\mathbf{P}}$ derivation. Hence, we can apply SWX to obtain a $\mathcal{D}_{\mathbf{G}}$ derivation. We proceed to build first a $\mathcal{D}_{\mathbf{P}}$ type inference, and then a $\mathcal{D}_{\mathbf{B}}$ type inference, in particular applying again ABS to introduce local linkage in p.

PAR: By IH, we can construct normal derivations of $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \,||\, \omega \vdash a, R \rightsquigarrow C, d$ and $\omega_{\mathbf{b}}, \omega_{\mathbf{S}}, \omega_{\mathbf{D}} \,||\, \omega \vdash b, S \rightsquigarrow D, e$. Each of these normal derivations we can unroll up to the last application of $\mathrm{PAR}_{\mathbf{n}}^{\mathbf{P}}$ $\mathcal{D}_i$ and $\mathcal{E}_j$, applied for concluding these $\mathrm{PAR}_{\mathbf{n}}^{\mathbf{P}}$ steps. To construct the required normal inference we simply let instead a single $\mathrm{PAR}_{\mathbf{n}}^{\mathbf{P}}$ step utilize all of the normal inferences $\mathcal{D}_i$ and $\mathcal{E}_j$.

PERM: By IH, we can construct a normal derivation of $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, \bigotimes_i^m P_{\pi(i)} \rightsquigarrow C, (\overline{\pi} \otimes \mathsf{id}_Z)d$. Unrolling this normal derivation up through the applications of DNF,

CLO, and $\text{PER}^{\mathbf{P}}$, we can edit the $\text{PER}^{\mathbf{P}}$-step to also move the permutation π to the context.

MERGE: By IH, we can construct a normal derivation of $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \rightsquigarrow C, d$ for global a and C. We unroll this derivation up across the $\mathcal{D}_{\mathbf{B}}$ production to obtain n $\mathcal{D}_{\mathbf{P}}$-derivations (for a and C of width n). We may consider these as $\mathcal{D}_{\mathbf{G}}$-derivations, also. We combine these in a single application of $\text{PAR}^{\mathbf{G}}_{\mathbf{n}}$, and, after a $\text{PAR}^{\mathbf{P}}_{\equiv}$ and a PER-step, we apply MER to merge the roots as required by the case. We conclude by adding term structure to the terms of this $\mathcal{D}_{\mathbf{P}}$-inference as required by the normal form and lead it through the steps to produce a $\mathcal{D}_{\mathbf{B}}$-derivation.

WIRING-AXIOM: As sketched in the text above, introduction of idle names is now handled by $\text{PAR}^{\mathbf{P}}_{n}$. For this case, we simply start with a $\text{PAR}^{\mathbf{P}}_{0}$-step and proceed through the grammar for $\mathcal{D}_{\mathbf{B}}$ to produce a normal inference as needed.

ABSTR: By IH, we can construct a normal derivation of $\sigma_{\mathbf{a}} \otimes \omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \sigma_{\mathbf{C}} \otimes \omega_{\mathbf{C}} \vdash p, R \rightsquigarrow P, d$. We unroll this normal derivation up across the entire $\mathcal{D}_{\mathbf{B}}$-inference to obtain a $\mathcal{D}_{\mathbf{P}}$ type inference. (We know there is only one $\mathcal{D}_{\mathbf{P}}$-inference, as p and P are prime.) We construct the required $\mathcal{D}_{\mathbf{B}}$ inference by starting with a modified ABS-step, where we introduce the required abstractions and local substitutions.

CLOSE: By IH, we can construct a normal inference for a sentence with only substitutions (i.e., with no closed links). We simply unroll this normal inference up across the CLO-step, and instead, to produce the needed normal inference, close the needed names in a new CLO-step. □ □

Normal inferences are sufficiently restricted such that we can base our prototype implementation on mechanically constructing them.

5.4 Nondeterminism

Given these term-based rules and the normal inference grammar, proven correct matching has been expressed in an operational, that is, implementable, form. However, there is still a fair amount of nondeterminism left, but fortunately we can clearly identify where it occurs:

Grammar selection: Which branches to select for $\mathcal{D}_{\mathbf{G}}$, $\mathcal{D}_{\mathbf{P}}$, $\mathcal{D}'_{\mathbf{G}}$ and $\mathcal{D}'_{\mathbf{P}}$.

Tensor grouping: How to group the tensor product in $\text{PAR}_{\equiv}$.

Children partitioning: How to partition molecules in MER.

Prime permutation: How to permute redex primes in PER.

Context-redex-parameter wiring: How to choose Z, α and τ in PAX.

Mapping closed links: How to find an appropriate decomposition of agent wiring in CLO such that closed agent links are matched correctly with closed redex links (i.e., determining $\sigma^{\mathbf{a}}$ and $Y_{\mathbf{R}}$).

When implementing matching, the challenge is to develop a heuristic that will handle typical cases well. In general, an agent-redex pair can lead to many different matches, so in our implementation we return for every inference rule a lazy list of possible matches.

To handle nondeterminism, we return possible matches as follows, bearing in mind that operationally speaking, rules applied below SWX are given agent and redex, while rules above SWX are given agent (, redex) and context:

Grammar selection: For $\mathcal{D}_{\mathbf{G}}$ and $\mathcal{D}_{\mathbf{P}}$, we concatenate the returned lazy lists returned from matching each branch in turn. However, if PAX succeeds, there is no reason to attempt a SWX match, as no new matches will result.

For $\mathcal{D}_{\mathbf{G}}{}'$ and $\mathcal{D}_{\mathbf{P}}{}'$, we try each branch in turn, returning the first branch that succeeds, as later branches will not find any new matches.

Tensor grouping: For given m and n in $\text{PAR}^{\mathcal{E}}_{\equiv}$, we compute all the ways of splitting $[0,\ldots,m-1]$ into n (possibly empty) subsequences, trying out matching for each split. Note that this need only be done for applications of $\text{PAR}^{\mathcal{E}}_{\equiv}$ below the SWX rule.

Children partitioning: For given m and n in MER, we compute all the ways of partitioning $\{0,\ldots,m-1\}$ into n (possibly empty) sets, trying out matching for each partitioning.

Prime permutation: For given n in $\text{PER}^{\mathcal{E}}$, we compute all n-permutations, trying out matching for each permutation. This is done for applications of $\text{PER}^{\mathcal{E}}$ below the SWX rule; above, similar permutations are computed in the MER rule.

Context-redex-parameter wiring: Given global agent wiring, we compute the ways of decomposing it into $\sigma(\mathsf{id}_Z \otimes \alpha\tau)$, returning a match for each decomposition.

Mapping closed links: We split agent wiring into named and closed links, and postpone the actual mapping of each closed link to redex or context links until some constraint, given by ION or PAX produces it.

Note that even after limiting nondeterminism in this way, we can still in general find several instances of the same match, reached by different inference trees, as we are computing abstract bigraph matches using concrete representations. For instance, matching redex $R = \mathsf{K1}$ in agent $a = \mathit{merge}(\mathsf{K1}\otimes\mathsf{K1})$ produces matches with context $C_1 = \mathit{merge}(\mathsf{id}_1 \otimes \mathsf{K1})$ and context $C_2 = \mathit{merge}(\mathsf{K1} \otimes \mathsf{id}_1)$.

5.5 Auxiliary Technologies

A number of auxiliary technologies are needed for implementing the match inference system presented here, notably the normalising and regularising operations needed

in the DNF rule. While they do not represent the most advanced part of bigraph matching, their correctness is vital for achieving a correct implementation.

5.5.1 Normalising

We define a normalisation relation $t \downarrow_{\mathbf{B}} t'$ for bigraph terms (details are given in Figure 5.21 of Appendix 5.A.1), with the following property:

Proposition 5.5.1. *For any bigraph terms t, t', if t represents a bigraph b and $t \downarrow_{\mathbf{B}} t'$, then t' represents b as well, and is on B-normal form given in Figure 5.4.*

The relation recursively normalises subterms, then recombines the results; for tensor product, the rule stated is

$$\text{Bten}\ \frac{\begin{array}{c} t_i \downarrow_{\mathbf{B}} (\omega_i \otimes \mathsf{id}_{(\vec{Y}_i)})D_i \qquad D_i \equiv \alpha_i \otimes (\bigotimes_{j\in n_i} P_i^j)\pi_i : I_i \to \langle n_i, \vec{Y}_i, Y_i\rangle \\ \omega = \bigotimes_{i\in n} \omega_i \quad \alpha = \bigotimes_{i\in n} \alpha_i \quad \mathsf{id}_{(\vec{Y})} = \bigotimes_{i\in n} \mathsf{id}_{(\vec{Y}_i)} \quad \pi = \bigotimes_{i\in n} \pi_i \\ P = \bigotimes_{j\in n} \bigotimes_{i\in n_j} P_i^j \qquad\qquad D \equiv \alpha \otimes P\pi \end{array}}{\bigotimes_{i\in n} t_i \downarrow_{\mathbf{B}} (\omega \otimes \mathsf{id}_{(\vec{Y})})D}.$$

We find that the expression $\bigotimes_{j\in n} \bigotimes_{i\in n_j} P_i^j$ in general will lead to name clashes, because we can only assume that outer, not inner names, of the ω_i's are disjoint.

One solution could be to rename names on P_i^j's outer face in the Bten rule. However, as Bten is applied recursively at each level of tensor product, this would lead to multiple renamings of the same names, causing inefficiency. Instead, we precede normalisation by a renaming phase described in the following; it will prevent name clashes in normalisation.

5.5.2 Renaming

While renaming names used in a term might look trivial at first sight, it is in fact not entirely straightforward. First, inner and outer names of a term must not be renamed, or we would be representing a different bigraph. Second, we cannot even require of a renamed term that all internal names are unique, as a normalised subterm can contain several instances of the same name, due to the use of id_Y in the normal form.

Thus, we need to identify a more refined notion of internal horizontal uniqueness, where a name can be reused vertically in link compositions, but not horizontally in tensor products. To this end, given a term t, we conceptually replace all occurrences of $/X$ by $e_1/x_1 \otimes \cdots \otimes e_n/x_n$, and $K_{\vec{y}(\vec{X})}$ by $K_{\vec{y}(\vec{e}/\vec{X})}$, in effect naming uniquely each closed link. We then define a function *linknames*, mapping terms to link namers (details are given in Figure 5.22 of Appendix 5.A.2). Using this function we define a predicate *normalisable*, which identifies terms whose tensor products and compositions do not produce subterms with name clashes, and is preserved by normalisation (details are given in Figure 5.23 of Appendix 5.A.2):

Proposition 5.5.2. *For any bigraph term t, if normalisable(t), there exists a t' such that $t \downarrow_{\mathbf{B}} t'$ and normalisable(t').*

For the actual renaming, we define inductively a renaming judgment $U \vdash \alpha, t \downarrow_{\boldsymbol{\beta}} t', \beta \dashv V$, where U is a set of used names and α renames t's inner names to those of t', while β renames t's outer names to those of t' and V extends U with names used in t' (details are given in Figure 5.24 of Appendix 5.A.2).

We can show that renaming preserves the bigraph, and enables normalisation:

Proposition 5.5.3. *Given a term t representing a bigraph $b : \langle m, \vec{X}, X\rangle \to \langle n, \vec{Y}, Y\rangle$, we can derive $X \cup Y \vdash \mathsf{id}_X, t \downarrow_{\boldsymbol{\beta}} t'', \beta - |V$ for some t'', β, V, and set $t' = ((\beta^{\mathbf{glob}})^{-1} \otimes (\widehat{\beta^{\mathbf{loc}}})^{-1})t''$; then t' represents b, and normalisable(t').*

5.5.3 Regularising

As a regular bigraph can be expressed as a term containing permutations, we must define *regularising* to represent it as a permutation-free term. This is done by splitting the permutations in the D- and G-normal forms, recursively pushing them into the subterms where they reorder the tensor product of S_i's.

While D's permutation π must be a tensor product of π_i's—otherwise the bigraph would not be regular—G's permutation, on the other hand, need not be so. However, as the bigraph is regular, it must be possible to split it into a major permutation $\underline{\pi}^{\vec{X}}$ and n minor permutations $\pi_i^{\vec{X}}$, based on the local inner faces, $\vec{X}$, of the S_i's. Then $\underline{\pi}^{\vec{X}}$ is elided by permuting the S_i's, and each $\pi_i^{\vec{X}}$ permutation is handled recursively in its S_i (details are given in Figure 5.25 of Appendix 5.A.3).

We can show that regularisation is correct:

Proposition 5.5.4. *Given a term t representing a regular bigraph b, we can infer $t' \to t'$, for some t' where t' contains no nontrivial permutations, and t' represents b.*

A detailed illustration of the entire reaction cycle involving the preceding transformation technologies can be seen in Figure 5.12.

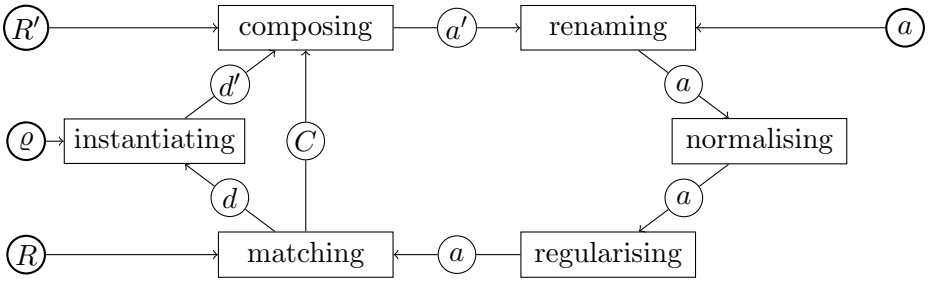


Figure 5.12: Details of the reaction cycle

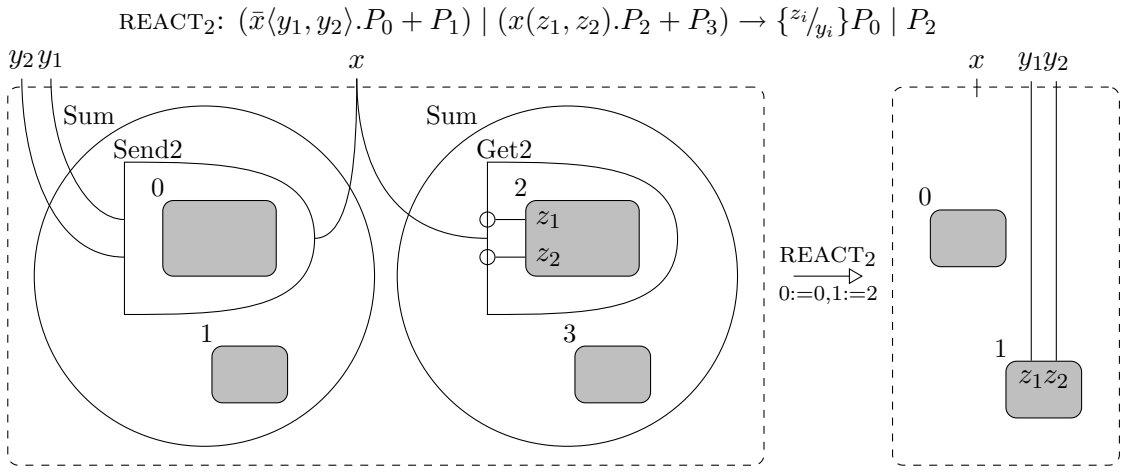
5.6 Tool Implementation and Example Modelling

We have implemented a BPL Tool as a reference implementation of binding bigraph matching, and as a toolbox for experimenting with bigraphs. It is written in SML, consists of parser, normalisation and matching kernel, and includes web and command line user interfaces [BPL07].

To ensure correctness, we have implemented normalisation, renaming, regularisation and matching faithfully by implementing one SML function for every inference rule—in the case of matching, two: one for applications above and one for below the SWX rule.

The BPL Tool handles normalisation, regularisation, matching and reaction for the full set of binding bigraphs, and allows construction of simple tactics for prescribing the order in which reaction rules should be applied. The following example output is taken verbatim from the command line interface, which is based on the SMLNJ interactive system; omitted details are indicated by "`[...]`".

As an example, we model the polyadic π calculus, running the mobile phone system introduced in Milner's π book [Mil99]. The calculus can be modeled by a family of reaction rules $\{\text{REACT}_i \mid i = 0, 1, \ldots\}$, one for each number of names that are to be communicated in a reaction [JM04]; REACT_2 is shown in Figure 5.13.




```
val REACT2 = "REACT2" :::
  Sum o (Send2[x,y1,y2] `|` idp(1)) `|` Sum o (Get2[x][[z1],[z2]] `|` idp(1))
  --[0 |-> 0, 1 |-> 2]--|>
  (y1/z1 * y2/z2 * x//[] * idp(1)) o (idp(1) `|` `[z1, z2]`);
```

Figure 5.13: π calculus reaction rule shown as bigraphs and BPL value.

The signature for the nodes modelling the calculus and the mobile phone system is constructed using `passive` and `atomic` functions as shown in Figure 5.14. For this system, we only need `Send` and `Get` nodes for REACT_0 and REACT_2. Note that all reaction rule nodes are passive, preventing reaction within a guarded expression.

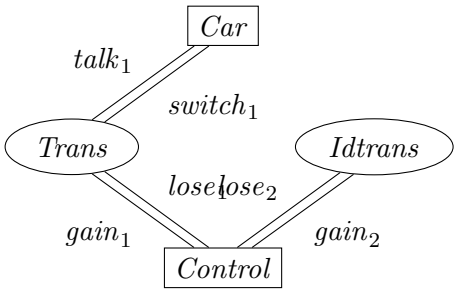
```
(* Pi calculus nodes *)
val Sum   = passive0 ("Sum")
val Send0 = passive ("Send0" -: 0 + 1)
val Get0  = passive ("Get0"  =: 0 --> 1)
val Send2 = passive ("Send2" -: 2 + 1)
val Get2  = passive ("Get2"  =: 2 --> 1)
```

```
(* Mobile phone system nodes *)
val Car      = atomic ("Car" -: 2)
val Trans    = atomic ("Trans" -: 4)
val Idtrans  = atomic ("Idtrans" -: 2)
val Control  = atomic ("Control" -: 8)
```

Figure 5.14: Signature for π calculus and mobile phone system nodes.

The system consists of a car, one active and one idle transmitter, and a control centre, as shown in Figure 5.15.

```
- val System1 = simplify (
      Car[talk1,switch1]
  `|` Trans[talk1,switch1,gain1,lose1]
  `|` Idtrans[gain2,lose2]
  `|` Control[lose1,talk2,switch2,gain2,
              lose2,talk1,switch1,gain1]);
```




```
val System1 =
  (lose1//[lose1_83, lose1_98] * talk2/talk2_82 * switch2/switch2_81
    * gain2//[gain2_80, gain2_95] * lose2//[lose2_7f, lose2_94]
    * talk1//[talk1_7e, talk1_9b, talk1_a5]
    * switch1//[switch1_7d, switch1_9a, switch1_a4]
    * gain1//[gain1_7c, gain1_99]) o merge(4) o
  (Car[talk1_a5, switch1_a4] *
   Trans[talk1_9b, switch1_9a, gain1_99, lose1_98] *
   Idtrans[gain2_95, lose2_94] *
   Control[lose1_83, talk2_82, switch2_81, gain2_80, lose2_7f, talk1_7e,
           switch1_7d, gain1_7c])
  : 0 -> <{lose1, talk2, switch2, gain2, lose2, talk1, switch1, gain1}> : bgval
-
```

Figure 5.15: Definition of the mobile phone system, $System_1$

Internally, a prime product constructed using the `` `|` `` operator is represented by a wiring and $merge_2$ composed with a binary tensor product. The function `simplify` applies various heuristics for producing human-readable bigraph terms, in this case for a prime product of four factors.

The definition of these nodes and connections, shown in Figure 5.16, allows the control centre to switch *Car* communication between the two transmitters (supposedly when the car gets closer to the ilde than the active transmitter), and allows the car to talk with the active transmitter. Note that in the BPL tool, we define a node by a rule that unfolds an atomic node into a bigraph corresponding to the defining

Defining equation	*BPL definition*

$Car(talk, switch) \stackrel{\text{def}}{=}$
$\overline{talk}.\ Car\langle talk, switch\rangle$
$+\ switch(t, s).\ Car\langle t, s\rangle$
	`val DEF_Car = "DEF_Car" :::`
	`  Car[talk,switch]`
	`  ----|>`
	`  Sum o (Send0[talk] o Car[talk,switch]`
	`  '|' Get2[switch][[t],[s]]`
	`  o (<[t,s]> Car[t,s]))`

$Trans(talk, switch, gain, lose) \stackrel{\text{def}}{=}$
$talk\ .\ Trans\langle talk, \underline{switch}, gain, lose\rangle$
$+\ lose(t, s).\overline{switch}\langle t, s\rangle$
$.\ Idtrans\langle gain, lose\rangle$
	`val DEF_Trans = "DEF_Trans" :::`
	`  Trans[talk,switch,gain,lose]`
	`  ----|>`
	`  Sum o (Get0[talk][] o`
	`  Trans[talk,switch,gain,lose]`
	`  '|' Get2[lose][[t],[s]]`
	`  o (<[t,s]> Sum o`
	`  Send2[switch,t,s]`
	`  o Idtrans[gain,lose]))`

$Idtrans(gain, lose) \stackrel{\text{def}}{=}$
$gain(t, s).\ Trans\langle t, s, gain, lose\rangle$
	`val DEF_Idtrans = "DEF_Idtrans" :::`
	`  Idtrans[gain, lose]`
	`  ----|>`
	`  Sum o Get2[gain][[t],[s]]`
	`  o (<[t,s]> Trans[t,s,gain,lose])`

$Control(lose_1, talk_2, switch_2, gain_2,$
$lose_2, talk_1, switch_1, gain_1) \stackrel{\text{def}}{=}$
$\overline{lose_1}\langle talk_2, switch_2\rangle.\overline{gain_2}\langle talk_2, switch_2\rangle$
$.\ Control\langle lose_2, talk_1, switch_1, gain_1,$
$lose_1, talk_2, switch_2, gain_2\rangle$
	`val DEF_Control = "DEF_Control" :::`
	`  Control[lose1,talk2,switch2,gain2,`
	`  lose2,talk1,switch1,gain1]`
	`  ----|>`
	`  Sum o Send2[lose1,talk2,switch2]`
	`  o Sum o Send2[gain2,talk2,switch2]`
	`  o Control[lose2,talk1,switch1,gain1,`
	`  lose1,talk2,switch2,gain2]`

Figure 5.16: Definitions of Car, Trans, Idtrans and Control nodes.

π calculus expression.

Our BPL definition of the initial system in Figure 5.15, $System_1$, is the folded version; as BPL matching is complete, querying the tool reveals the four possible unfolding matches, illustrated in Figure 5.17. Here `mkrules` constructs the inter-

```
- val rules = mkrules [REACT0, REACT2, DEF_Car, DEF_Trans,
                       DEF_Idtrans, DEF_Control];
[...]
- print_mv (matches rules System1);
[{rule = "DEF_Car",
 context
  = (lose1//[lose1_d3, lose1_d6] * talk2/talk2_d2 * switch2/switch2_d1
      * gain2//[gain2_d0, gain2_d5] * lose2//[lose2_cf, lose2_d4]
      * talk1//[talk, talk1_ce, talk1_d9]
      * switch1//[switch, switch1_cd, switch1_d8]
      * gain1//[gain1_cc, gain1_d7]) o
    (merge(4) o
     (Trans[talk1_d9, switch1_d8, gain1_d7, lose1_d6] *
      Idtrans[gain2_d5, lose2_d4] *
      Control[lose1_d3, talk2_d2, switch2_d1, gain2_d0, lose2_cf,
              talk1_ce, switch1_cd, gain1_cc])),
 parameter = idx0},
{rule = "DEF_Control", [...] },
{rule = "DEF_Idtrans", [...] },
{rule = "DEF_Trans", [...] }]
```

Figure 5.17: Determining which rules match $System_1$.

nal representation of a rule set, and `print_mv` prettyprints a lazy list of matches, produced by the `matches` function.

Using `react_rule` that simply applies a named reaction rule, and `++` that runs its arguments sequentially, we construct a tactic, `TAC_unfold`, for unfolding all four nodes once, shown in Figure 5.18. Applying this tactic using function `run`, we get an unfolded version of the system.

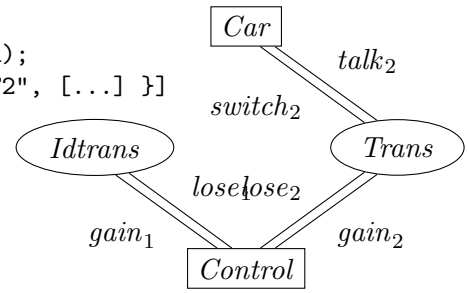
Querying the BPL Tool for all possible matches in the unfolded system reveals exactly the switch and talk actions, initiated by REACT_2 and REACT_0 rules, respectively, cf. Figure 5.19. Applying the π calculus reaction rules for switching, we arrive at $System_2$, where *Car* communication has been switched to the other transmitter, as witnessed by the outer names to which *Car* ports link, as well as the order of names to which *Control* ports link.

This concludes our description of the example highlighting how we can use the BPL Tool to experiment with bigraphical reactive systems.

```
- val TAC_unfold =
  react_rule "DEF_Car"     ++ react_rule "DEF_Trans"   ++
  react_rule "DEF_Idtrans" ++ react_rule "DEF_Control";
[...]
- val System1_unfolded = run rules TAC_unfold System1;
val System1_unfolded =
  (lose1//[lose1_3f9, lose1_419, lose_441, lose_459, lose_45d]
    * talk2//[talk2_3f8, talk2_40f, talk2_418]
    * switch2//[switch2_3f7, switch2_40e, switch2_417]
    * gain2//[gain2_3f6, gain2_410, gain_431, gain_438]
    * lose2//[lose2_3fd, lose_430]
    * talk1//[talk1_3fc, talk_460, talk_465, talk_482, talk_485]
    * switch1//[switch1_3fb, switch_447, switch_45f, switch_480, switch_481]
    * gain1//[gain1_3fa, gain_442, gain_45e]) o merge(4) o
  (Sum o merge(2) o
   (Send0[talk_485] o Car[talk_482, switch_481] *
    Get2[switch_480][[t_47d], [s_47c]] o
    (<[s_47c, t_47d]> Car[t_47d, s_47c])) *
   Sum o merge(2) o
   (Get0[talk_465] o Trans[talk_460, switch_45f, gain_45e, lose_45d] *
    Get2[lose_459][[t_446], [s_445]] o
    (<[s_445, t_446]>
      Sum o (Send2[switch_447, t_446, s_445] o Idtrans[gain_442, lose_441]))) *
   Sum o Get2[gain_438][[t_433], [s_432]] o
   (<[s_432, t_433]> Trans[t_433, s_432, gain_431, lose_430]) *
   Sum o
   (Send2[lose1_419, talk2_418, switch2_417] o
    (Sum o
     (Send2[gain2_410, talk2_40f, switch2_40e] o
      Control[lose2_3fd, talk1_3fc, switch1_3fb, gain1_3fa, lose1_3f9,
              talk2_3f8, switch2_3f7, gain2_3f6]))))
  : 0 -> <{lose1, talk2, switch2, gain2, lose2, talk1, switch1, gain1}> : agent
```

Figure 5.18: Unfolding $System_1$, using the TAC_unfold tactic.

```
- print_mv (matches rules System1_unfolded);
[{rule = "REACT0", [...] }, {rule = "REACT2", [...] }]
```




```
[...]
- val TAC_switch =
  react_rule "REACT2"      ++ (* Control tells Trans to lose.   *)
  react_rule "REACT2"      ++ (* Control tells Idtrans to gain. *)
  react_rule "REACT2";        (* Trans tells Car to switch.     *)
[...]
- val System2 = run rules TAC_switch System1_unfolded;
val System2 =
  (lose1//[lose1_86a, lose_8c0] * talk2//[t_858, talk2_869, t_8bf]
    * switch2//[s_857, switch2_868, s_8be] * gain2//[gain_856, gain2_867]
    * lose2//[lose_855, lose2_86e] * talk1/talk1_86d * switch1/switch1_86c
    * gain1//[gain1_86b, gain_8c1]) o merge(4) o
  (Idtrans[gain_8c1, lose_8c0] * Car[t_8bf, s_8be] *
   Control[lose2_86e, talk1_86d, switch1_86c, gain1_86b, lose1_86a, talk2_869,
           switch2_868, gain2_867] * Trans[t_858, s_857, gain_856, lose_855])
  : 0 -> <{lose1, talk2, switch2, gain2, lose2, talk1, switch1, gain1}> : agent
-
```

Figure 5.19: Checking possible matches, then switching to $System_2$, using the TAC_switch tactic.

5.7 Conclusion and Future Work

We have developed a provably sound and complete inference system over bigraph terms for inferring legal matches of bigraphical reactive systems. Moreover, we have implemented our BPL Tool, the first implementation of bigraphical reactive systems. We have demonstrated a simple, but concrete, example of how the tool can be used to simulate bigraphical models. We have found it very useful to base this first implementation of bigraphical reactive systems so closely on the developed theory—this has naturally given us greater confidence in the implementation, but the implementation work has also helped to debug the developed theory.

There are lots of interesting avenues for future work. While the current implementation of BPL Tool is efficient enough to experiment with small examples, we will try to make it more efficient by using a number of different techniques: we plan to investigate how to prune off invalid matches quickly, for instance by making use of sorting information [BDH06]. Moreover, we will investigate to what extent we can capture the link graph matching via a constraint-based algorithm.

We also plan to investigate smarter ways of combining matching and rewriting. As a starting point, we have made it possible for users to combine *tactics* to inform the tool in which order it should attempt to apply reaction rules.

Jean Krivine and Robin Milner are currently investigating stochastic bigraphs, which will be particularly important for simulation of real systems. We hope that our detailed analysis of matching for binding bigraphs will make it reaonably straightforward to extend it to stochastic bigraphs.

5.A Auxiliary Technologies Details

5.A.1 Normalising

We define a normalisation relation $t \downarrow_{\mathbf{B}} t'$ for elementary bigraphs: $merge_n, {}^{\ulcorner}X^{\urcorner}, \vec{y}/\vec{X}$, $K_{\vec{y}(\vec{X})}$, and π as shown in Figure 5.20, and inductively for operations: abstraction $(X)P$, product $\bigotimes_i^n B_i$ and composition $B_1 B_2$ as shown in Figure 5.21, where the notation $\sigma|^Y$ means $\{X \mapsto y \in \sigma \mid y \in Y\}$.

5.A.2 Renaming

Let a *link namer* be a map μ mapping every link l (outer name or edge) in its domain to a pair (E, X), where E is a set of names used internally to compose the link, and X are the inner names linking to l. We let $\mathcal{V}_i(Y, \mu) = \bigcup_{y \in Y, y \mapsto (X_1, X_2) \in \mu} X_i$ and define link namer composition by

$$\begin{array}{rcl} \mu_1 \circ \mu_2 & = & \{y_1 \mapsto (E_1 \cup X_1 \cup V_1, V_2) \mid y_1 \mapsto (E_1, X_1) \in \mu_1 \wedge V_i = \mathcal{V}_i(X_1, \mu_2)\} \\ & \cup & \{y_2 \mapsto (E_2, X_2) \in \mu_2 \mid \forall y_1 \mapsto (E_1, X_1) \in \mu_1 : y_2 \notin X_1\}, \end{array}$$

essentially composing links of μ_1 with those of μ_2, and adding closed links from μ_2.

We then define a function *linknames*, mapping terms to link namers, by the equations given in Figure 5.22. By using the link namers of immediate subterms,

$$\text{Bmer}\ \frac{N \equiv (\emptyset)(\mathsf{id}_\emptyset \otimes merge_n)\bigotimes_{i\in n}{}^{\ulcorner}\mathsf{id}_\emptyset{}^{\urcorner} \qquad P \equiv (\mathsf{id}_\emptyset \otimes {}^{([])}/_{([])})N \qquad D \equiv \mathsf{id}_\emptyset \otimes (\bigotimes_{i\in 1} P)\mathsf{id}_n}{merge_n \downarrow_{\mathbf{B}} (\mathsf{id}_\emptyset \otimes \mathsf{id}_{[\emptyset]})D}$$

$$\text{Bcon}\ \frac{\begin{array}{c} N \equiv (\emptyset)(\mathsf{id}_X \otimes merge_1)\bigotimes_{i\in 1}(\mathsf{id}_X \otimes \mathsf{id}_1)^{\ulcorner}X^{\urcorner} \\ P \equiv (\mathsf{id}_X \otimes {}^{([])}/_{([])})N \qquad D \equiv \mathsf{id}_\emptyset \otimes (\bigotimes_{i\in 1} P)\mathsf{id}_{(X)} \end{array}}{{}^{\ulcorner}X^{\urcorner} \downarrow_{\mathbf{B}} (\mathsf{id}_X \otimes \mathsf{id}_{[\emptyset]})D}$$

$$\text{Bwir}\ \frac{}{\vec{y}/\vec{X} \downarrow_{\mathbf{B}} (\vec{y}/\vec{X} \otimes \mathsf{id}_{[]})(\mathsf{id}_X \otimes \mathsf{id}_0\mathsf{id}_0)}$$

$$\text{Bion}\ \frac{\begin{array}{c} X = \{\vec{X}\} \qquad Y = \{\vec{y}\} \qquad M \equiv (\mathsf{id}_\emptyset \otimes K_{\vec{y}(\vec{X})})(X)(\mathsf{id}_X \otimes merge_1)\bigotimes_{i\in 1}(\mathsf{id}_X \otimes \mathsf{id}_1)^{\ulcorner}X^{\urcorner} \\ N \equiv (\emptyset)(\mathsf{id}_Y \otimes merge_1)\bigotimes_{i\in 1} M \qquad P \equiv (\mathsf{id}_Y \otimes {}^{([])}/_{([])})N \qquad D \equiv \mathsf{id}_\emptyset \otimes (\bigotimes_{i\in 1} P)\mathsf{id}_{(X)} \end{array}}{K_{\vec{y}(\vec{X})} \downarrow_{\mathbf{B}} (\mathsf{id}_Y \otimes \mathsf{id}_{[\emptyset]})D}$$

$$\text{Bper}\ \frac{\begin{array}{c} Y_i = \{\vec{y}_i\} \qquad N_i \equiv (Y_i)(\mathsf{id}_{Y_i} \otimes merge_1)\bigotimes_{j\in 1}(\mathsf{id}_{Y_i} \otimes \mathsf{id}_1)^{\ulcorner}Y_i{}^{\urcorner} \\ P_i \equiv (\mathsf{id}_\emptyset \otimes \widehat{\vec{y}_i/\vec{y}_i})N_i \qquad\qquad D \equiv \mathsf{id}_\emptyset \otimes (\bigotimes_{i\in m} P_i)\pi \end{array}}{\pi : \langle m, \vec{X}, X\rangle \to \langle m, \vec{Y}, X\rangle \downarrow_{\mathbf{B}} (\mathsf{id}_\emptyset \otimes \mathsf{id}_{\vec{Y}})D}$$

Figure 5.20: Inference rules for normalising elementary bigraph expressions

we can determine whether a term can be normalised without name clashes. To this end, we define a predicate *normalisable* by the equations given in Figure 5.23. We basically just require, that at no level in the term does two different links share any internal names.

Renaming is achieved by the judgment $U \vdash \alpha, t \downarrow_{\boldsymbol{\beta}} t', \beta \dashv V$, where U is a set of used names and α renames t's inner names to those of t', while β renames t's outer names to those of t' and V extends U with names used in t'. The system of rules for inferring this judgment is given in Figure 5.24.

5.A.3 Regularising

The system of rules for inferring a permutation-free term representing a regular bigraph is given in Figure 5.25.

$$\text{Babs}\ \frac{\begin{array}{c} b \downarrow_{\mathbf{B}} (z/W \otimes \mathsf{id}_{([Y])})(\mathsf{id}_\emptyset \otimes (\bigotimes_{i\in 1}(\mathsf{id}_Z \otimes \widehat{\vec{y}/\vec{X}})(W)G)\mathsf{id}_I) \\ \vec{z}^X = [z_j \leftarrow \vec{z} \mid z_j \in X] \qquad \vec{z}^{\bar{X}} = [z_j \leftarrow \vec{z} \mid z_j \notin X] \\ \vec{W}^X = [W_j \leftarrow \vec{W} \mid z_j \in X] \quad \vec{W}^{\bar{X}} = [W_j \leftarrow \vec{W} \mid z_j \notin X] \\ W^X = \{\vec{W}^X\} \qquad W^{\bar{X}} = \{\vec{W}^{\bar{X}}\} \qquad U = \{\vec{y}\vec{z}^X\} \\ N \equiv (W^X \cup W)G \qquad P \equiv (\mathsf{id}_{W^{\bar{X}}} \otimes \widehat{\vec{y}\vec{z}^X/\vec{X}\vec{W}^X})N \end{array}}{(X)b \downarrow_{\mathbf{B}} (z^{\bar{X}}/W^{\bar{X}} \otimes \mathsf{id}_{([U])})(\mathsf{id}_\emptyset \otimes (\bigotimes_{i\in 1} P)\mathsf{id}_I)}$$

$$\text{Bten}\ \frac{\begin{array}{c} b_i \downarrow_{\mathbf{B}} (\omega_i \otimes \mathsf{id}_{(\vec{Y}_i)})D_i \qquad D_i \equiv \alpha_i \otimes (\bigotimes_{j\in n_i} P_i^j)\pi_i : I_i \to \langle n_i, \vec{Y}_i, Y_i\rangle \\ \omega = \bigotimes_{i\in n} \omega_i \quad \alpha = \bigotimes_{i\in n} \alpha_i \quad \mathsf{id}_{(\vec{Y})} = \bigotimes_{i\in n} \mathsf{id}_{(\vec{Y}_i)} \quad \pi = \bigotimes_{i\in n} \pi_i \\ P = \bigotimes_{j\in n} \bigotimes_{i\in n_j} P_i^j \qquad D \equiv \alpha \otimes P\pi \end{array}}{\bigotimes_{i\in n} b_i \downarrow_{\mathbf{B}} (\omega \otimes \mathsf{id}_{(\vec{Y})})D}$$

$$\text{Ccom}\ \frac{\sigma = (\mathsf{id}_Z \otimes \alpha)(\mathsf{id}_Z \otimes y/X)}{(\mathsf{id}_Z \otimes (\alpha \otimes \mathsf{id}_1)^{\ulcorner}Y^{\urcorner}) \bigotimes_{i\in 1}(\mathsf{id}_Z \otimes \widehat{\vec{y}/\vec{X}})(X)(\mathsf{id}_U \otimes merge_n)\bar{S} \downarrow_{\bar{\mathbf{S}}} \sigma, \bar{S}}$$

$$\text{Mcom}\ \frac{\begin{array}{c} (\mathsf{id}_Z \otimes N)\bar{P} \downarrow_{\mathbf{N}} \sigma, N' \qquad \vec{X}' = \sigma^{-1}(\vec{X}) \\ Z' = \sigma^{-1}(Z) \quad Y' = \sigma^{-1}(Y) \quad \sigma' = \mathsf{id}_{\{\vec{y}\}} \otimes \sigma\lfloor^{Z\uplus Y} \end{array}}{(\mathsf{id}_Z \otimes (\mathsf{id}_Y \otimes K_{\vec{y}(\vec{X})})N)\bar{P} \downarrow_{\bar{\mathbf{S}}} \sigma', \bigotimes_{i\in 1}(\mathsf{id}_{Z'\uplus Y'} \otimes K_{\vec{y}(\vec{X}')})N'}$$

$$\text{Ncom}\ \frac{\begin{array}{c} P_i : \langle m_i, \vec{X}_i, X_i\rangle \to \langle 1, (Y_i), Y_i \uplus W_i\rangle \\ \bigotimes_{i\in n} \bar{P}_i = \bigotimes_{i\in k} P_i \quad \bar{P}_i : I_i \to \langle n_i, \vec{Y}_i, \{\vec{Y}_i\} \uplus Z_i\rangle \quad (\mathsf{id}_{Z_i} \otimes S_i)\bar{P}_i \downarrow_{\bar{\mathbf{S}}} \sigma_i, \bar{S}_i \\ \bar{S} = \bigotimes_{i\in n} \bar{S}_i : I \to \langle n', Z' \uplus Y'\rangle \qquad \sigma = \bigotimes_{i\in n} \sigma_i \\ X' = \sigma^{-1}(X) \qquad Z' = \sigma^{-1}(Z) \qquad Y' = \sigma^{-1}(Y) \end{array}}{\big(\mathsf{id}_Z \otimes (X)(\mathsf{id}_Y \otimes merge_n)\bigotimes_{i\in n} S_i\big)\bigotimes_{i\in k} P_i \downarrow_{\mathbf{N}} \sigma, (X')(\mathsf{id}_{Z'\uplus Y'} \otimes merge_{n'})\bar{S}}$$

$$\text{Pcom}\ \frac{(\mathsf{id}_Z \otimes N)\bar{P} \downarrow_{\mathbf{N}} \sigma, N' \quad W = \sigma^{-1}(Z \uplus Z') \quad \vec{X}' = \sigma^{-1}(\vec{X}) \quad \sigma' = \sigma\lfloor^{Z\uplus Z'}}{(\mathsf{id}_Z \otimes (\mathsf{id}_{Z'} \otimes \widehat{\vec{y}/\vec{X}})N)\bar{P} \downarrow_{\mathbf{P}} \sigma', (\mathsf{id}_W \otimes \widehat{\vec{y}/\vec{X}'})N'}$$

$$\text{Bcom}\ \frac{\begin{array}{c} b_1 \downarrow_{\mathbf{B}} \Big(\omega_1 \otimes \mathsf{id}_{(\vec{U}^1)}\Big)D_1 : \langle m', \vec{X}', X' \uplus Z\rangle \to \langle n, \vec{U}^1, U^1 \uplus W\rangle \\ b_2 \downarrow_{\mathbf{B}} \Big(\omega_2 \otimes \mathsf{id}_{(\vec{U}^2)}\Big)D_2 : \langle m, \vec{X}, X \uplus U\rangle \to \langle m', \vec{U}^2, U^2 \uplus Z\rangle \\ D_1 \equiv \alpha_1 \otimes \Big(\bigotimes_{i\in n} P_i^1\Big)\pi_1 : \langle m', \vec{X}', X' \uplus Z\rangle \to \langle n, \vec{U}^1, U^1 \uplus V^1 \uplus W^1\rangle \\ D_2 \equiv \alpha_2 \otimes \Big(\bigotimes_{i\in m'} P_i^2\Big)\pi_2 : \langle m, \vec{X}, X \uplus U\rangle \to \langle m', \vec{U}^2, U^2 \uplus V^2 \uplus W^2\rangle \\ P_i^1 : \langle m_i', \vec{X}_i', X_i'\rangle \to \langle (U_i^1), U_i^1 \uplus V_i^1\rangle \qquad P_i^2 : \langle m_i'', \vec{X}_i'', X_i''\rangle \to \langle (U_i^2), U_i^2 \uplus V_i^2\rangle \\ \omega_1 : V^1 \uplus W^1 \to W \qquad \omega_2 : V^2 \uplus W^2 \to Z \qquad \alpha_1 : Z \to W^1 \qquad \alpha_2 : U \to W^2 \\ V^2 = \biguplus_{i\in m'} V_i^2 \qquad \bigotimes_{i\in m'} P^2_{\pi_1^{-1}(i)} = \bigotimes_{i\in n} \bar{P}_i \qquad \bar{P}_i : I_i' \to \langle m_i', \vec{X}_i', X_i' \uplus Z_i'\rangle \\ (\mathsf{id}_{Z_i'} \otimes P_i^1)\bar{P}_i \downarrow_{\mathbf{P}} \sigma_i, P_i \quad \sigma = \mathsf{id}_U \otimes \bigotimes_{i\in n} \sigma_i \quad \omega = \omega_1(\alpha_1\omega_2(\alpha_2 \otimes \mathsf{id}_{V^2}) \otimes \mathsf{id}_{V^1})\sigma \\ \pi = \overline{\pi_1}_{\vec{X}''}\pi_2 \qquad D \equiv \mathsf{id}_U \otimes \big(\bigotimes_{i\in n} P_i\big)\pi \end{array}}{b_1 b_2 \downarrow_{\mathbf{B}} \Big(\omega \otimes \mathsf{id}_{(\vec{U}^1)}\Big)D}$$

Figure 5.21: Inference rules for normalising bigraph abstraction, product and composition expressions

$$
\begin{array}{lcl}
\mathit{linknames}(\mathit{merge}_n) & = & \{\} \\
\mathit{linknames}(\ulcorner X \urcorner) & = & \{x \mapsto (\{\}, \{x\}) \mid x \in X\} \\
\mathit{linknames}(\vec{y}/\vec{X}) & = & \{y_i \mapsto (\{\}, X_i) \mid i \in |\vec{y}|\} \\
\mathit{linknames}(K_{\vec{y}(\vec{e}/\vec{X})}) & = & \{y_i \mapsto (\{\}, \{\}) \mid i \in |\vec{y}|\} \cup \{e_i \mapsto (\{\}, X_i) \mid i \in |\vec{X}|\} \\
\mathit{linknames}(\pi : \rightarrow \langle m, \vec{X}, X \rangle) & = & \{x \mapsto (\{\}, \{x\}) \mid x \in X\} \\
\mathit{linknames}((Y)P) & = & \mathit{linknames}(P) \\
\mathit{linknames}(\bigotimes_i t_i) & = & \bigcup_i \mathit{linknames}(t_i) \\
\mathit{linknames}(t_1 t_2) & = & \mathit{linknames}(t_1) \circ \mathit{linknames}(t_2)
\end{array}
$$

Figure 5.22: Function for determining which names are used internally to compose a link

$$
\begin{array}{lcl}
\mathit{normalisable}(\mathit{merge}_n) & = & \mathit{true} \\
\mathit{normalisable}(\ulcorner X \urcorner) & = & \mathit{true} \\
\mathit{normalisable}(\vec{y}/\vec{X}) & = & \mathit{true} \\
\mathit{normalisable}(K_{\vec{y}(\vec{e}/\vec{X})}) & = & \mathit{true} \\
\mathit{normalisable}(\pi : \rightarrow \langle m, \vec{X}, X \rangle) & = & \mathit{true} \\
\mathit{normalisable}((Y)P) & = & \mathit{normalisable}(P) \\
\mathit{normalisable}(\bigotimes_i t_i) & = & \bigwedge_i \mathit{normalisable}(t_i) \\
 & & \wedge(\forall i \neq j : E_i \cap E_j = \emptyset) \\
 & & \text{where} \quad \mu_i = \mathit{linknames}(t_i) \\
 & & \qquad\quad E_i = \bigcup_{y \mapsto (E,X) \in \mu_i} E \\
\mathit{normalisable}(t_1 t_2) & = & \mathit{normalisable}(t_1) \wedge \mathit{normalisable}(t_2) \\
 & & \wedge(\forall l_1 \neq l_2 : \mu_{\mathbf{E}}(l_1) \cap \mu_{\mathbf{E}}(l_2) = \emptyset) \\
 & & \text{where} \quad \mu_i = \mathit{linknames}(t_i) \\
 & & \qquad\quad \mu = \mu_1 \circ \mu_2 \\
 & & \qquad\quad \mu_{\mathbf{E}}(l) = E, \text{ if } \mu(l) = (E, X)
\end{array}
$$

Figure 5.23: Function for determining whether a (well-formed) term is normalisable

$$\text{Rmer}\ \frac{}{U \vdash \mathsf{id}_\emptyset, \mathit{merge}_n \downarrow_{\boldsymbol{\beta}} \mathit{merge}_n, \mathsf{id}_\emptyset - |U} \qquad \text{Rcon}\ \frac{X' = \alpha(X)}{U \vdash \alpha, \ulcorner X \urcorner \downarrow_{\boldsymbol{\beta}} \ulcorner X' \urcorner, \alpha - |U}$$

$$\text{Rwir}\ \frac{\begin{array}{cc} Z = \{\vec{z}\} & Z \cap U = \emptyset \quad |Z| = |\vec{z}| = |\vec{y}| \\ \vec{X}' = \alpha(\vec{X}) & \beta = \{y_i \mapsto z_i\} \end{array}}{U \vdash \alpha, \vec{y}/\vec{X} \downarrow_{\boldsymbol{\beta}} \vec{z}/\vec{X}', \beta - |U \cup Z}$$

$$\text{Rion}\ \frac{\begin{array}{cc} Z = \{\vec{z}\} & Z \cap U = \emptyset \quad |Z| = |\vec{z}| = |\vec{y}| \\ \vec{X}' = \alpha(\vec{X}) & \beta = \{y_i \mapsto z_i\} \end{array}}{U \vdash \alpha, K_{\vec{y}(\vec{X})} \downarrow_{\boldsymbol{\beta}} K_{\vec{z}(\vec{X}')}, \beta - |U \cup Z}$$

$$\text{Rper}\ \frac{X' = \alpha(X) \quad \vec{X}' = \alpha(\vec{X}) \quad \vec{Y}' = \alpha(\vec{Y})}{U \vdash \alpha, \pi : \langle m, \vec{X}, X\rangle \to \langle m, \vec{Y}, X\rangle \downarrow_{\boldsymbol{\beta}} \pi : \langle m, \vec{X}', X'\rangle \to \langle m, \vec{Y}', X'\rangle, \alpha - |U}$$

$$\text{Rabs}\ \frac{U \vdash \alpha, t \downarrow_{\boldsymbol{\beta}} t', \beta - |V \quad X' = \beta(X)}{U \vdash \alpha, (X)t \downarrow_{\boldsymbol{\beta}} (X')t', \beta - |V}$$

$$\text{Rten}\ \frac{t_i : \langle m_i, \vec{X}_i, X_i\rangle \to J_i \quad \alpha_i = \alpha \restriction_{X_i} \quad U_i \vdash \alpha_i, t_i \downarrow_{\boldsymbol{\beta}} t_i', \beta_i - |U_{i+1} \quad \beta = \bigotimes_{i\in n} \beta_i}{U_0 \vdash \alpha, \bigotimes_{i\in n} t_i \downarrow_{\boldsymbol{\beta}} \bigotimes_{i\in n} t_i', \beta - |U_n}$$

$$\text{Rcom}\ \frac{U_1 \vdash \alpha_1, t_2 \downarrow_{\boldsymbol{\beta}} t_2', \beta_1 - |U_2 \quad U_2 \vdash \beta_1, t_1 \downarrow_{\boldsymbol{\beta}} t_1', \beta_2 - |V_2}{U_1 \vdash \alpha_1, t_1 t_2 \downarrow_{\boldsymbol{\beta}} t_1' t_2', \beta_2 - |V_2}$$

Figure 5.24: Renaming rules

$$\alpha\ \frac{}{\ulcorner \alpha \urcorner \mathsf{id}_{(X)}\text{`} \to \ulcorner \alpha \urcorner} \qquad \text{M}\ \frac{N\pi\text{`} \to N'}{(\mathsf{id}_Z \otimes K_{\vec{y}(\vec{X})})N\pi\text{`} \to (\mathsf{id}_Z \otimes K_{\vec{y}(\vec{X})})N'}$$

$$\text{N}\ \frac{S_i : \langle m_i, \vec{X}_i\rangle \to J_i \quad \pi = \underline{\pi'}^{\vec{X}} \quad S_i \pi_i'^{\vec{X}}\text{`} \to S_i'}{((X)(\mathsf{id}_Y \otimes \mathit{merge}_n)\bigotimes_{i\in n} S_i)\pi'\text{`} \to (X)(\mathsf{id}_Y \otimes \mathit{merge}_n)\bigotimes_{i\in n} S'_{\pi(i)}}$$

$$\text{D}\ \frac{\pi = \bigotimes_{i\in n} \pi_i \quad \pi_i : I_i' \to I_i \quad N_i : I_i \to J_i \quad N_i\pi_i\text{`} \to N_i'}{\alpha \otimes (\bigotimes_{i\in n}(\mathsf{id}_{Z_i} \otimes \widehat{\vec{y}_i/\vec{X}_i})N_i)\pi\text{`} \to \alpha \otimes \bigotimes_{i\in n}(\mathsf{id}_{Z_i} \otimes \widehat{\vec{y}_i/\vec{X}_i})N_i'}$$

$$\text{B}\ \frac{D\text{`} \to D'}{(\omega \otimes \mathsf{id}_{(\vec{X})})D\text{`} \to (\omega \otimes \mathsf{id}_{(\vec{X})})D'}$$

Figure 5.25: Removing nontrivial permutations from regular bigraphs.

Chapter 6

A Generic Language for Biological Systems based on Bigraphs

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.

Preface This chapter contains the tech report [DK08]. The report was co-authored with Jean Krivine currently at Harvard Medical School.

6.1 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 biologists) 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, Mil06b]. 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 BioAm-

bients. 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 [Mil06b], 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 [GM07b]), 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 proposals [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 [Car04b], 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.

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

In Section 6.2, 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 6.3, we make the effort to treat carefully a family 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 6.4, we formally state the relationship between $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi and BRSs; and we conclude this paper by illustrating in Section 6.5, 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 6.3), and the section on encoding the κ-calculus as a $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculus (Section 6.5), at readers who know little or next to nothing about bigraphs. In Section 6.5, 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 6.2) and the section concerned with establishing that $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi are, in fact, a certain kind of bigraphical calculi (Section 6.4), we adress mainly to readers who are familiar with the basic setup of pure bigraphs (as can be gotten from, say, [Mil06b]).

6.2 Bigraphical Preliminaries

In this section, we start by giving brief and informal recap of necessary concepts of the theory for pure bigraphs [Mil06b]. 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.

6.2.1 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 $a \to a'$ with (R, R'), when we have $a = C(R \otimes \mathsf{id})d \to C(R' \otimes \mathsf{id})d' = a'$ 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.

6.2.2 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] 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.

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 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 $(\mathsf{K} \otimes \mathsf{id})\Box_0 \to (\mathsf{K} \otimes \mathsf{id})(\Box_0 \mid \Box_0)$, to the agent $a = /x\,(\mathsf{K} \otimes \mathsf{id})(\mathsf{M}_x \mid \mathsf{M}_x)$ do we copy the binder or not? This corresponds precisely to instantiating the parameter as either $d = /x\,(\mathsf{M}_x \mid \mathsf{M}_x)$ under the empty context or as $d = (\mathsf{M}_{x_1} \mid \mathsf{M}_{x_2})$ 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 6.3) we are mostly interested in side-conditions testing the outer names of parameters.

Definition 6.2.1 (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 6.2.2 (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 6.3 to build a self-contained characterization in structural operational semantics-style of how bigraphical calculi evolve under non-aliasing semantics.

6.3 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 6.3.1 (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 $\mathsf{L} : \mathrm{atomic}(2)$ and $\mathsf{M} : \mathrm{active}(0)$—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 $\mathsf{L}_{y,x} \mid \mathsf{M}.\mathsf{L}_{y,z}$ 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

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

Figure 6.1: Notational conventions—names, variables and controls

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.

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 6.1. In Definition 6.3.2, we define formally processes, and in Definition 6.3.3, 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 6.3.4 formally defines the well-formedness criterion.

Definition 6.3.2 (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 6.3.3 (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 6.3.4 (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 $\mathsf{M}.0$.

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 $V_{45} \mid V_{56}$ and $V_0 \mid V_1$. 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 6.3.5 (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 6.3.6 (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 $V_0 \mid V_1$.

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

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

For groups we extend the definition above to include also:

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

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 $\mathrm{fn}(M) \cap \mathrm{fn}(N) \neq \emptyset$.

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 6.3.8 (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 $M \equiv N$, $\mathrm{fn}(M) = \mathrm{fn}(N)$.

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 6.3.9 (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 6.3.10 (width of group)**.** For

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

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

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 6.3.11 (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,

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

As each variable is a leaf in C it is easy to see that the substituted term respects the grammar (i.e., in Definition 6.3.2 and in Definition 6.3.3). 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 6.3.12 (total substitution)**.** Substitution $C \cdot D$ of variables in C by processes in D is defined when $|\operatorname{var}(C)| = \operatorname{width}(D)$ and $\operatorname{bn}(C) \cap \operatorname{fn}(D) = \operatorname{bn}(C) \cap$

[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.

$\mathrm{bn}(D) = \mathrm{fn}(C) \cap \mathrm{bn}(D) = \emptyset$. 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 6.3.13 (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 6.3.14 generalizes Definition 6.3.12 to the cases, where D has fewer processes than C has variables.

Definition 6.3.14 (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.

[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.

6.3.1 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.

Definition 6.3.15 (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 6.3.16 records a small set of rules which together characterize reactions for processes.[7]

Definition 6.3.16 (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

[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.

relation $\rightarrow$, the least binary relation over $\mathcal{P}^{\Sigma}$, s.t.

$$\text{RULE}\frac{(M \rightarrow N, \varphi) \in \mathcal{R} \quad \exists G \text{ s.t. } P = M \cdot G \text{ and } P' = N \cdot G \quad \varphi(G) \text{ satisfied}}{P \rightarrow P'}$$

$$\text{PAR}\frac{P \rightarrow P'}{P \mid Q \rightarrow P' \mid Q} \qquad \text{PREFIX}\frac{P \rightarrow P'}{\mathsf{M}_{\vec{y}}.P \rightarrow \mathsf{M}_{\vec{y}}.P'}$$

$$\text{CLOSE}\frac{P \rightarrow P'}{(x)P \rightarrow (x)P'} \qquad \text{STRUCT}\frac{Q \equiv P \quad P \rightarrow P' \quad P' \equiv Q'}{Q \rightarrow Q'}$$

$$\text{SUBST}\frac{P \rightarrow P' \quad \exists \tilde{x}.\alpha : \text{fn}(P) \leftrightarrow \tilde{x}}{\alpha(P) \rightarrow \alpha(P')},$$

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 $P \rightarrow P'$, $\text{fn}(P) = \text{fn}(P')$.

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

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

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 $\mathsf{K}_z \mid \mathsf{K}_z \mid \mathsf{M} \rightarrow \mathsf{K}_z \mid \mathsf{M}.\mathsf{K}_z$, 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

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

which contradicts our intention with the side-condition.

In Section 6.4, 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.

[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.

6.4 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.

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 6.3.16. 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 6.4.2, a static correspondence verifying that $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes correspond to certain kinds of bigraphs with a single root (primes). In Lemma 6.4.3, 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 6.4.4, we characterize $\mathcal{B}^{\Sigma,\mathcal{R}}$-reaction via substitution. By way of this characterization, in Theorem 6.4.5, we may formally state and verify the dynamic correspondence, we are looking for.

6.4.1 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 $\mathcal{B}^{\Sigma,\mathcal{R}}$.

[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.

Lemma 6.4.1 (normal form for link-epi, prime bigraphs)**.** *All link-epi primes $P^{\mathbf{B}}$ may be expressed on the following normal form*

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

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.)*

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 6.3.9 and the normal form for link-epi primes above, it is easy to check the following property.

Proposition 6.4.2 ($\mathcal{B}^{\Sigma,\mathcal{R}}$-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 6.4.2 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 6.4.3 ($\mathcal{B}^{\Sigma,\mathcal{R}}$-substitution is bigraphical composition)**.** $(C \cdot D)^{\mathbf{B}} = (C^{\mathbf{B}} \mid\mid \mathsf{id}_Y) D^{\mathbf{B}}$*, for* $Y = \mathrm{fn}(D^{\mathbf{B}})$.

6.4.2 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 6.2.2). 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 6.3.16 allows to rewrite.

Lemma 6.4.4 (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:

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

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 6.3.9) this is straightforward. □

Having thus characterized reactions via substitution, which in turn corresponds to bigraphical composition (Lemma 6.4.3) we may bridge the gap to bigraphical reaction.

Theorem 6.4.5 ($\mathcal{B}^{\Sigma,\mathcal{R}}$ 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 6.2.1 and Definition 6.2.2.*

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 6.4.4, 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 6.4.3 that $\mathcal{B}^{\Sigma,\mathcal{R}}$-substitution is bigraphical dotting; name-hiding and substitution correspond to composition with closure and renaming, respectively.

We choose $C = (/\tilde{x} \otimes \mathsf{id}_{\langle 1, Y \uplus Z\rangle})(O^{\mathbf{B}} \parallel \mathsf{id}_Y)(\alpha \otimes \mathsf{id}_1) = (/\tilde{x} \otimes \mathsf{id}_{\langle 1, Y \uplus Z\rangle})(O^{\mathbf{B}} \parallel \alpha)$, 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 6.4.4.

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

6.5 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.

6.5.1 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:

$$\begin{array}{rcll} S, T & ::= & 0 & \text{empty solution} \\ & | & A(\rho) & \text{protein} \\ & | & S, T & \text{group} \\ & | & (x)\,(S) & \text{new name} \end{array}$$

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 6.3.7.

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 6.3.8.

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 6.2, 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:

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

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 6.3.16; 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):

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

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

6.5.2 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 6.5.2, and solutions, in Definition 6.5.3, to $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes. In Proposition 6.5.4, we state formally, that this translation respects and reflects structural congruence. Definition 6.5.5 states that rules are simply translated pointwise, and in Proposition 6.5.6 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 6.5.7, and verify it in Lemma 6.5.8.

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

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

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 6.5.2 (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 $\mathbb{N}$.

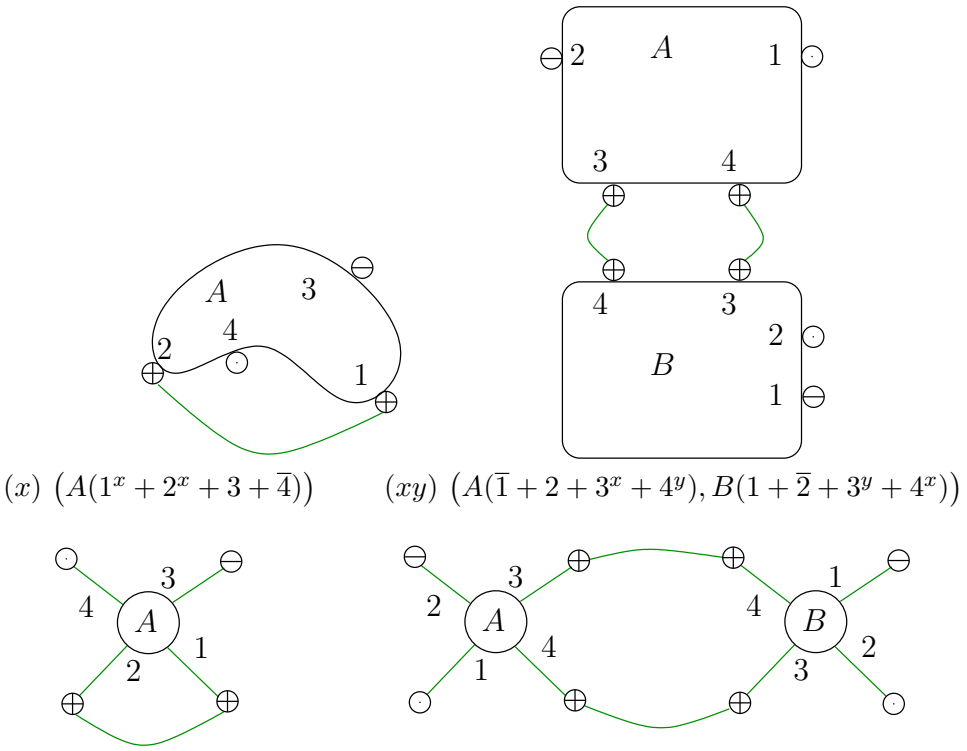


Figure 6.2: Examples: κ-terms as $\kappa^{\mathcal{B}}$-processes.

Definition 6.5.3 (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 $\beta : n \to \tilde{b}' = \{i \mapsto b_i \mid \rho(i) \text{ defined}\}$.

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 6.2 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 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.

[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.

Proposition 6.5.4 (static correspondence)**.** *$S \equiv T$ (for κ-$\equiv$) if and only if $[\![S]\!] \equiv [\![T]\!]$ (for $\mathcal{B}^{\Sigma,\mathcal{R}}$-$\equiv$).*

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

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 6.5.6 (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

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

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

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

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 6.4.4), 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. □

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 6.5.7 (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 6.5.8 (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 6.5.3.

2. By the normal form (Proposition 6.3.9) 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. □

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.

6.6 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 [Mil06b]).

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 [Mil06b]. (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 concerned

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.

Chapter 7

A Language for the Cell

Abstract

We introduce a formal language, the $\mathcal{C}$-calculus, for modelling low-level interaction inside and among cells, the basic building blocks of all known life. We focus on two main actors of cells, proteins and membranes. Proteins are represented as clusters of *domains* sharing a common hidden name; domain-domain bonds are also represented via name-sharing. Compartments are formed by formal *membranes*. We treat also *channels* between membranes allowing transport of proteins, allowing us to capture an observable intermediate state in cell fusion or division, regulated by diffusion. We illustrate the calculus by giving two example models. We exemplify the basic constituents of the calculus, by developing a model of simple cross-membrane signalling via a G-protein coupled receptor protein. We continue by developing a model illustrating part of the endocytic pathway—the formation of clathrin-coated cytoplasmic vesicles, through budding from the plasma membrane (the cell-wall).

Preface This part contains the tech report [DDK08]. The report was co-authored with Jean Krivine, Harvard Medical School, and with Vincent Danos, University of Edinburgh.

7.1 Introduction

In this paper, we introduce a formal language, the $\mathcal{C}$-calculus, for modelling low-level interaction inside and among cells, the basic building blocks of all known life. We start by giving a brief overview of the biology of cells, and discuss the level of abstraction that we aim to capture in the calculus. In the $\mathcal{C}$-calculus, we focus on two main actors of cells, membranes and proteins.

Membranes In nature, biological membranes are vital for separating cells from their surroundings, for confining dangerous compounds to secure compartments, for serving as vehicles of transport, or, for promoting certain reactions by raising

the local concentration of reactants. The membrane of a cell consists mostly of a double layer of proteins and lipids (fat-like molecules). Membranes may have a variety of molecules attached to them or embedded within them. Notably, such molecules may be *transmembrane* proteins, which serve to propagate signals across the membrane barrier; or, they may be various molecules, which serve as pumps for moving different smaller molecules in and out of the cell. The molecules attached to membranes also serve to differentiate the two sides of a membrane and to stabilize it. Organelles ("small organs") inside eukaryotic cells (cells with nuclei) are also wrapped in protective membranes, as are *vesicles* transporting material between cells.

Biological membranes are dynamic entities—under certain conditions, they may form *buds*, new small membrane-enclosed compartments that grow from a particular location on the membrane. Budding may serve to create vesicles, for instance. A membrane may also undergo *fission* and divide entirely in two, in the process also dividing the content in the enclosed compartment.[1] Both budding and cell division are typically complicated and regulated processes involving various kinds of machinery (depending on the type of membranes and division). For instance—in what is sometimes known as asexual reproduction of eukaryotic cells—when two eukaryotic cells divide, initially the chromosomes (the structures carrying DNA) in the cell nucleus are carefully separated into two nuclei. This process is called *mitosis*. In a following step called *cytokinesis*, the cytoplasm (the contents inside the cellular membrane) as well as the organelles and the cellular membrane are split up to form two new daughter cells.

Membranes may also *fuse* with other membranes. This occurs, for instance, when viruses infect cells, and it serves to create such structures as the long fibers of human skeletal muscles. When two cells fuse together, their membranes meet at one point and create a connection between the membranes. Eventually, the two membranes will form one single, continuous membrane that surrounds the contents of both cells.

Research has confirmed that both when cells fuse or divide, membranes will temporarily be in a partially fused state with a *neck* consisting of partially fused membrane material.[2] This neck acts as a bridge connecting two compartments and it typically allows some material to be exchanged by diffusion. Even further, it also happens that cells abort fusion (or fission). For instance, two cells may partially fuse, exchange some material by diffusion, and then part again.

[1] As in computer science, biological terminology tends to be overloaded. In this paper, we shall use *fission* as a generic term to refer to any process by which membranes divide. In doing so, we stretch the terminology typically employed by biochemists a bit, but it is convenient to use the term to describe the reverse of fusion. We stress that our usage of fission should not be confused with so-called *binary fission*—the specific process by which prokaryotic cells (cells without a nuclei) divide.

[2] For instance, studies on cellular division in nematode worms show that an intermediate state may indeed be observed during cell fusion [Pod06]. Other studies have focused on different models for the intermediate structures in membrane fission [KK03]. Some studies also hypothesize that the shapes of the intermediate protein structures, which form the neck itself, may drive and serve to distinguish the process of fission and fusion [KC02].

In the $\mathcal{C}$-calculus, compartments are formed by formal membranes. To capture faithfully the dynamics of fusion and fission, we include in the $\mathcal{C}$-calculus also binary *channels* between membranes. Channels are, to our knowledge, a novel abstraction that allows us to capture an observable intermediate state in fusion and fission. In this intermediate state the connected compartments may exchange material regulated by diffusion. Formally, channels are formed by *gates* sharing a common name; gates can be seen as abstractions for the various molecular-structures that constitute the neck.

Membranes-enclosed cells are large structures compared to the typical protein. While a eukaryotic cell may be around $10-100\mu$m, the typical diameter of a globular protein—like hemoglobin, for instance—is around 5nm. The difference in scale of mass is far larger. Thus, from the perspective of cells and membranes, proteins can be considered essentially atomic.

Proteins Proteins are linear chains of amino-acids. The string of amino-acids is known as the *backbone* of a protein. Gene-sequences in the DNA enclosed in all cells determines the amino-acid sequence for all the proteins that a cell can produce. Production (synthesis) of proteins involves *transcribing* the DNA to mRNA, various kinds of *post-transcriptional* modification, and *translating* the mRNA to proteins by way of small cellular factories called *ribosomes*. Finally, most proteins fold into three-dimensional structures called the *conformation* or the *tertiary* structure of the protein. The tertiary structure may in turn hide or reveal some parts of the protein.[3] The entire process of synthesis of proteins is complex, self-organized and, as nearly all biological phenomena, highly regulated by various molecular signals.

The peptide bonds forming the backbone make proteins relatively stable structures. Proteins may also form weaker bonds with other proteins to form multi-protein *complexes*. These bonds form on particular parts of the proteins referred to as *domains* or *sites*.

In the $\mathcal{C}$-calculus, our aim is to capture a domain-level description suitable for describing protein-protein interaction in a multi-compartment environment. To allow for transmembrane-proteins, in the $\mathcal{C}$-calculus proteins are represented as clusters of domains sharing a common name. This name is a formal representative of the backbone. Domain-domain bonds are also represented via name-sharing.

Formal foundation and sources of inspiration Formally, the calculus is founded on bigraphical reactive systems (due to Milner et al. [JM04, Mil06b]) through the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework treated by Damgaard and Krivine [DK08] (see below). Thus, the $\mathcal{C}$-calculus admits a formally founded graphical language; and we inherit results for a tacit term syntax with well-understood structural congruence properties [Mil05, DB06], and a well-understood behavioral theory. Even further, due to

[3] Traditionally, biochemists refer to four distinct aspects of structure for proteins. *Primary* structure, the amino-acid sequence; *secondary* structure, describing regular local structures (such as so-called *alpha-helixes* or *beta-sheets*); *tertiary* structure, the 3-d structure; and, *quaternary* structure, the structure of *complexes* formed from several proteins.

recent results by Milner, Krivine, and Troina [KMT08], a stochastic extension of the nondeterministic calculus presented in this paper, is well-founded. Implementation of bigraphs has also been investigated [BDGM07], and a prototype implementation is available [BPL07].

The $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework (presented in [DK08]) paves the way for our usage of bigraphical calculi for modelling biological interaction, in particular, by extending bigraphical reaction rules to allow testing of contextual negative side-conditions. It also introduces a self-contained and generic presentation of the operational semantics for a subset of bigraphical calculi, in a structural style more in line with how process calculi are typically presented. We shall make good use of this groundwork in this paper, eliding most discussion of bigraphical idiosyncracies, and focusing on the presentation and motivation of the calculus itself.

We are inspired by and combine features from several existing languages, in particular, the κ calculus [DL04], a rule-based language capturing domain-level protein-protein interaction. We base our domain-level model of proteins closely on our experience with capturing the κ-calculus in the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework (in [DK08]), only here generalized to a multi-compartment setting. A main aim of the work presented in this paper, has been to extend a κ-like language with dynamic compartments.

The model for membranes and their dynamics is influenced, in particular, by the brane-family of calculi [Car04b, DP04], which deals solely with membrane interaction; and the Bioambients [RPS$^+$04], one of the first calculi to investigate at the same time both molecular and membrane-level interaction.

We discuss in more detail the inspiration sources for the $\mathcal{C}$-calculus both in the main sections of the paper, and in Section 7.5.2.

An overview of the paper Having given an overview of the biology of cells and introduced the background that the $\mathcal{C}$-calculus builds on, in the next section we present the $\mathcal{C}$-calculus static model. In Sections 7.2.1 and 7.2.2, we introduce and motivate the model of proteins and membranes, and in Section 7.2.3 we define the grammar for $\mathcal{C}$-calculus *solutions* and define an associated structural congruence relation. In Section 7.2.4, we discuss how structural congruence classes of solutions correspond to bigraphs. This paves the way for a graph-based description of *well-formed* solutions, which we define and motivate in Section 7.2.5. We shall define well-formedness to ensure that well-formed solutions are those that we can interpret biologically.

In Section 7.3, we turn to the operational semantics of the calculus. The κ-calculus has been one of several advocates for what we may call *rule-based* modelling. Rules can be understood and manipulated separately, are easily visualized, and, easily express possible overlapping behaviors of certain configurations. On the other hand, we need to uphold certain well-formedness invariants to ensure that we can interpret model configurations sensibly as biological states. Hence, we advocate a middle-ground for the $\mathcal{C}$-calculus: *Modelling by rule refinement*. We shall give a set of rule-generators that each express a core biological action for membranes and proteins. The key idea is that a domain expert, say a bio-chemist, may pick

and refine those core generators to give a domain-specific sub-calculus for the study of a particular biological application. He does this by giving a set of application conditions that express when an action may be applied.

We start by defining $\mathcal{C}$-calculus reactive systems and reaction rules in Section 7.3.1. In Section 7.3.2 we describe formally so-called *projective* descriptions of rules, inspired by the Projective Brane Calculus by Pradelier and Danos [DP04], and, more recently, by the *patch* reactions treated by Cardelli for Bitonal Systems [Car08]. They let us describe conveniently sets of rules for reactions involving regions separated by one or two membrane-surfaces, while eliding the orientation of those separating membranes. In Sections 7.3.3 and 7.3.4, we introduce rules for protein-protein interaction and membrane (or channel) reconfiguration, respectively. In Section 7.3.5, we define refinements of rules and settle on a definition of those rules that we allow in the $\mathcal{C}$-calculus. Finally, in Section 7.3.6 we show that the allowable rules preserve well-formedness.

In Section 7.4, we illustrate the calculus with two examples. We exemplify the basic constituents of the calculus, by developing a model of simple cross-membrane signalling via a G-protein coupled receptor protein. We continue by developing a model illustrating part of the endocytic pathway—the formation of clathrin-coated cytoplasmic vesicles, through budding from the plasma membrane (the cell-wall).

In Section 7.5, we conclude and discuss some related and future work.

In Appendix 7.A, we include a series of definitions for solutions inherited directly from the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework for such properties as free names, normal forms, and substitution.

7.2 The $\mathcal{C}$-calculus—a Model of Proteins and Membranes

In Sections 7.2.1 and 7.2.2, we introduce and motivate the foundation of the $\mathcal{C}$-calculus—the model of proteins and membranes. In Section 7.2.3, we define formally the grammar for the $\mathcal{C}$-calculus and its associated structural congruence relation. In Section 7.2.4, we discuss how solutions up to structural congruence correspond to bigraphs, and, finally, in Section 7.2.5 we define and motivate well-formedness for the $\mathcal{C}$-calculus.

7.2.1 Proteins

In the $\mathcal{C}$-calculus, proteins are represented as groups of interaction *domains* sharing a common name, which we call the *backbone*. Domain-domain bonds, which let proteins form complexes, are also represented via name-sharing. While in the κ-calculus proteins are atomic entities, domains are the atoms of the $\mathcal{C}$-calculus. Domains are represented via atomic function symbols, and thus depicted as atomic nodes (i.e., they cannot contain other nodes).

Formally, we derive the set of domains from a protein-signature. Protein-names are drawn from a fixed set $\mathcal{P}$, which we require to be disjoint from all other kinds of names.

Definition 7.2.1 (protein-signature)**.** A protein-signature, $\Sigma^{\mathcal{P}}$, is a map from $\mathcal{P}$ to $\mathbb{N}^2$.

The protein-signature, $\Sigma^{\mathcal{P}}$, maps each protein-name p to a pair of integers (a, r), where a is the *arity* of the protein—it determines the number of domains of p—and, r is the *receptor-arity* of p. We explain the receptor-arity below.

Definition 7.2.2 (domains)**.** Given a protein-signature $\Sigma^{\mathcal{P}}$, let $\mathcal{D} \subseteq \mathcal{P} \times \mathbb{N}$, the set of domains induced by $\Sigma^{\mathcal{P}}$, be given as

$$\mathcal{D} = \{(\mathsf{p}, i) \mid \mathsf{p} \in \mathcal{P},\ \Sigma^{\mathcal{P}}(\mathsf{p}) = (a, r),\ \text{and } 0 \leq i < a\}.$$

We choose a simple concrete language for expressing the domains of proteins. Suppose the protein-name pro has the signature $(4, 1)$. The arity 4 determines that there are four domains in a protein of type pro. We write its four domains as $\mathsf{pro0}, \mathsf{pro1}, \mathsf{pro2}, \mathsf{pro3}$. We interpret the receptor-arity for a protein-name in the following manner: If the receptor-arity r is larger than zero, pro is a transmembrane protein, and all well-formed instances of pro must take the shape of two subunits separated by a single membrane: one with domains 0 to $r - 1$, and one with the domains numbered r to $n - 1$.

Protein domains have one or two ports for connecting to named *links*. All domains of a protein share a common *backbone*-link connected to their first port, and pairs of bound domains may share a *complexation*-link on a second port. A domain has an associated complexation-state, which determines their current ability to form complexes. It may be *bound* to a link x; it may be in a *hidden* state due to the overall conformation of the protein it is part of; or it may be currently unbound, but *visible* and ready to form new complexation-links.[4]

We write $\mathsf{pro}i_a^x$ for the ith domain of a pro protein connected to the backbone a and bound to the complexation-link x; we write $\mathsf{pro}i_a$ for the domain in its visible, but unbound state; and, we write $\overline{\mathsf{pro}i}_a$ for the domain in its hidden state. Figure 7.1 shows how we illustrate these three kinds of domains. Figure 7.2 contains an illustration of an instance of the entire protein pro, which respects the protein-signature. Figure 7.2) contains an illustration of a well-formed instance of a closed protein pro. (We formally introduce the name-hiding operator (x) and the syntax for membranes in the following sections.) [5]

[4] We have some freedom in choosing exactly how to model as state. In particular, there is a subtle difference in electing to model unbound, *visible* domains (i) via a separate state as inspired by the κ-calculus, or (ii) via closed unary links. Model (i) allows us to use less names, while model (ii) allows us to write a reconfiguration rule, which matches both bound and unbound domains. In this paper, we have chosen model (i), thus building on the intuition from the κ-calculus.

[5] Our model and naming scheme for proteins and their domains is essentially a minimal extension to a multi-locality setting of the $\mathcal{B}^{\Sigma,\mathcal{R}}$-model of κ-calculus proteins described in [DK08]. Centrally, the atom representing the protein backbone has been substituted for a shared link, allowing us to conveniently model transmembrane proteins.

Though the protein-model and naming scheme is simplistic, it is sufficient for describing many interesting examples (as we shall see later in Section 7.4). However, one may easily extend the protein model with more complex kinds of state, as done for later versions of the κ calculus [DFF$^+$07, DFFK07], or with more complex naming schemes, for instance, for proteins or do-

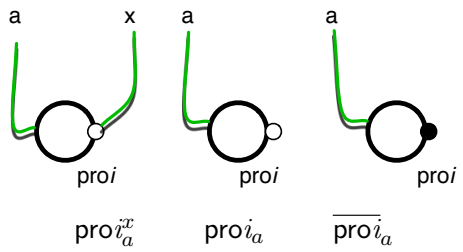
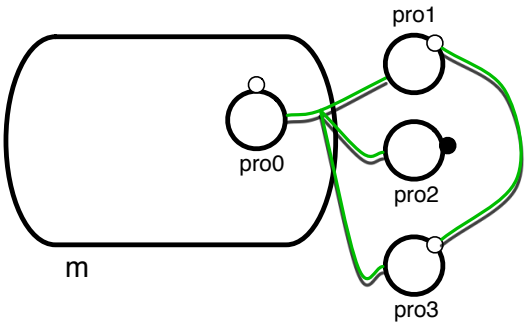


Figure 7.1: Domain states.



$(a, x)\,\mathsf{m}[\mathsf{pro0}_a] \mid \mathsf{pro1}^x_a \mid \overline{\mathsf{pro2}}_a \mid \mathsf{pro3}^x_a$

Figure 7.2: A well-formed instance of pro.

7.2.2 Adding Dynamic Compartments

To model membranes and their dynamics in the $\mathcal{C}$-calculus, we add two kinds of function symbols to the language. In keeping with the tradition set out by Cardelli et. al. for ambient and brane-calculi, we represent formal membranes by square brackets $[\cdots]$. The syntactic space between the brackets induces a region in which other entities, including other membranes, may float. As for proteins, it is convenient to allow different kinds of membranes to be distinguished by names, allowing us to distinguish, for instance, a vesicle from the plasma membrane encapsulating a cell. For example, we may write an expression to model that a plasma membrane contains an empty vesicle: $\mathsf{plasma}[\mathsf{ves}[]]$. We suppose that such membrane-names are drawn from another fixed pool of names, $\mathcal{M}$, disjoint from every other kind of names.

We add also *gates* to the language. They are the endpoints of binary *channels* between membranes. Gates, and the channels they form, are an abstraction of the molecular structure forming bonds between fusing or dividing membranes. We shall discuss the channel abstraction in more detail in a separate section below. Gates have one port to connect gates pairwise; for instance, the expression $(x)\,\mathsf{ves}[\triangle_x] \mid \mathsf{ves}[\triangle_x]$ denotes a channel formed between sibling vesicles.

---

mains. For instance, in later versions of the κ-calculus (see *loc. cit.*), the capability to write such rules is added by introducing a small subtyping lattice for states and incorporating subtyping into the matching relation.

While our gates, $\triangle_g$, are anonymous, one could certainly consider also allowing gates to be named, as domains and membranes. This would allow us to distinguish different kinds of channels, potentially convenient for some models. In this paper, however, we shall focus on introducing the channel abstraction itself, and for that purpose, anonymous gates suffice.

On the Channel Abstraction

Compared to existing calculi treating membranes [Car04b, RPS+04, PQ05, LT06, Car08], channels allow us to capture an observable semi-fused intermediate state in cell fusion or fission. In this state, the membranes are still separate, but may exchange material regulated by diffusion. To capture partial fusion or fission as described in the introduction, we need to keep the material in each compartment separate.

In Figure 7.3, we illustrate the two most basic configurations of semi-fused membranes, and show how they are represented in the calculus with the help of channels. On the left, the surfaces of two co-located membranes are partially fused; on the bottom-left we show the $\mathcal{C}$-calculus representation of that situation. They may be cells in the process of fusing or dividing; or it may be a cell in the process of budding a small vesicle. On the right, we have a similar situation, only with one membrane inside the other. In both cases, the partial fusing of the membranes allows diffusion of material across the point where the membrane surfaces touch. On the left, the channel allows diffusion between the two compartments inside the membranes. On the right, the channel allows transport between the surroundings and the innermost compartment.

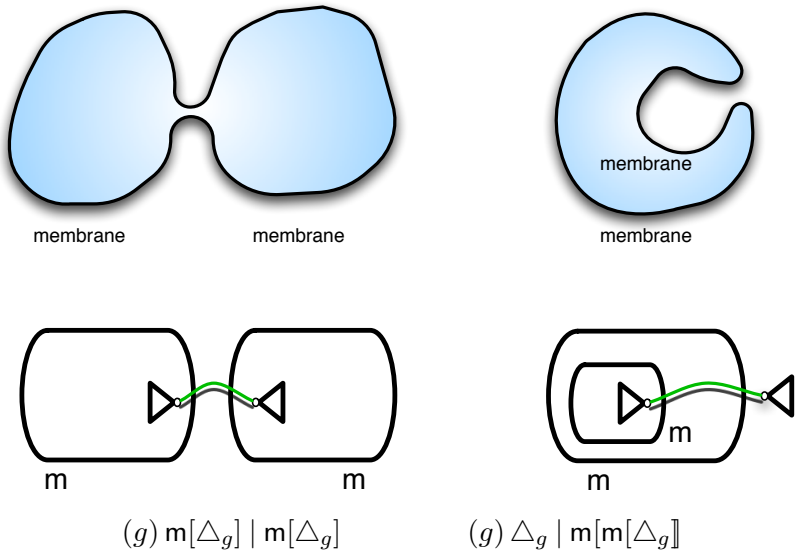


Figure 7.3: Fused membranes, and their representation via channels in the $\mathcal{C}$-calculus.

In the topmost part of Figure 7.4, we illustrate the three different stages in cell fusion and fission that we wish to capture. The arrows indicate, that while the process of fusing two membranes is deterministic (resulting in a single membrane containing all the entities of both compartments), the division of the entities inside a single compartment is a nondeterministic process.

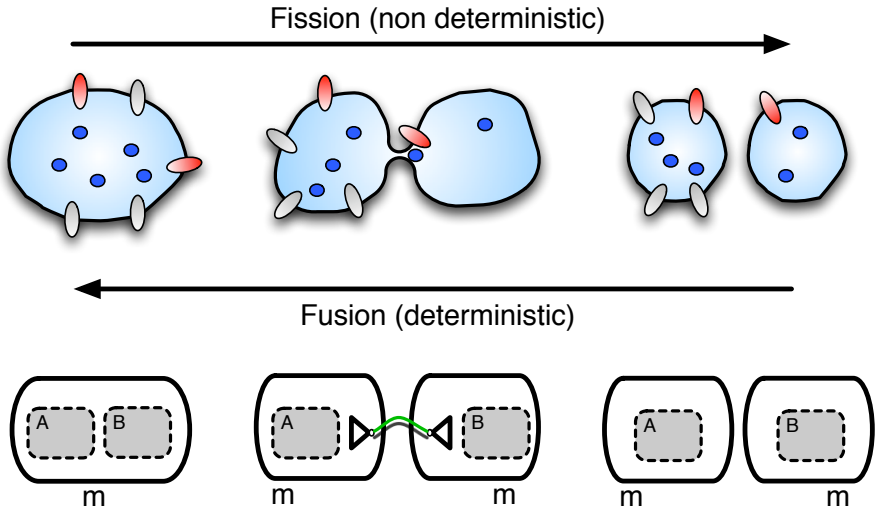


Figure 7.4: Representing an observable intermediate state—for cell fusion and division.

In nature, various kinds of cell division may be identified. For instance, due to their different structure, the division processes are fundamentally different between eukaryotic cells and prokaryotic cells. Common to them all are that they are quite complicated and highly regulated processes—in particular, for determining the division of contained entities. By adding an intermediate state, we allow users of the $\mathcal{C}$-calculus to model such regulatory behavior during the process of division, in more detail. In particular, it allows rules for diffusion to regulate the final division of entities in the newly formed compartments.

At the bottom of Figure 7.4, we illustrate the three different stages as they are captured in the $\mathcal{C}$-calculus. We focus on membranes and gates, and use variables to parameterize the model over the contents of the membranes. We shall return to these stages, when we discuss the dynamics of the $\mathcal{C}$-calculus in Section 7.3.

As a final remark, we may also note that while the main motivation for channels is to represent an intermediate step in division regulated by (nondeterministic) diffusion, one might also envision using channels to model more stable kinds of transport channels, such as so-called *tunneling nano-tubes* (TNTs) [RSM+04, GBG08]. TNTs are long and stable binary channels between co-located membranes that were recently discovered and bear a striking resemblance to our binary gated channels.[6]

[6]We discuss TNTs in more detail under future work (see Section 7.5.1.)

7.2.3 Signature and Syntax

In the previous section, we have informally introduced proteins, membranes, and gates and their syntax. In this section, we formally define the signature and grammar of the $\mathcal{C}$-calculus.

Signature

The signature of the $\mathcal{C}$-calculus defines the classes of function symbols in $\mathcal{C}$-calculus, and their type and number of ports for linkage.

Definition 7.2.3 ($\mathcal{C}$-calculus signature)**.** Given a protein-signature $\Sigma^{\mathcal{P}}$ and a set of membrane-names $\mathcal{M}$, the signature for the $\mathcal{C}$-calculus is

$$\begin{array}{rcll} \Sigma & = & \mathcal{D} \times \{\mathrm{v}, \mathrm{h}\} & : \ \mathrm{atomic}(1) \\ & \uplus & \mathcal{D} \times \{\mathrm{b}\} & : \ \mathrm{atomic}(2) \\ & \uplus & \mathcal{M} & : \ \mathrm{active}(0) \\ & \uplus & \{\triangle\} & : \ \mathrm{atomic}(1) \end{array}$$

where $B : \mathit{kind}(n)$ is short for stating that the elements of B are assigned the type *kind* and n ports.

The signature formally records what we have stated informally above. Function symbols from the set of domains, $\mathcal{D}$ (as defined in Definition 7.2.2), are atomic and exists in three states, visible (v), hidden (h), or bound (b). They all have one port (for linking to a protein backbone) and in their bound state an extra port (linked to a complexation-link). The function symbols used for membranes have no ports, but are active—i.e., they may nest other solutions inside them—and gates are atomic and have a single port (linked to another gate).

Grammar

We call the basic computing units in the $\mathcal{C}$-calculus *solutions*, and we shall define their grammar below. We use a small set of well-known operators from process calculi equipped with pure names, in particular, we take an operator for hiding names, (x), and parallel product, $|$. We shall also quotient the grammar over a structural congruence relation ensuring, for instance, that bound names are alpha-convertible and that parallel product is associative and commutative as usual. In formally defining solutions, we presuppose an unbounded supply of pure names (disjoint from all other kinds of names); we use lowercase italic letters $a, b, c, \ldots$ for pure names.

For defining parametric reaction rules in the following section, we shall also need solutions with variables (sometimes called holes), and groups—ordered sequences of solutions that may share names. Variables in the $\mathcal{C}$-calculus are numbered, i.e., they are drawn from a countable set of variables $\mathbb{V} = \{\Box_0, \Box_1, \ldots\}$. Formally, we require for any solution, that all variables in a solution are distinct (in the following, we assume all solutions and groups to respect this requirement). We let metavariables

$x, y, z, \ldots$	pure names
$A, B, C, \ldots$	variables
d, e	generic domains
m, n	generic membranes

Figure 7.5: Notational conventions—names, variables, domains, and membranes

$A, B, C, \ldots, X, Y, Z, \ldots$ range over variables. In practice, we shall not be concerned with the actual numbers used for variables, only their order (i.e., relatively to the other variables) in a solution. We record this in the structural congruence relation below.

Formally, the $\mathcal{C}$-calculus is a $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculus [DK08]. This means that the following grammar, definitions for structural congruence, substitution, etc. for the $\mathcal{C}$-calculus is derived directly from the $\mathcal{C}$-calculus signature, and the generic grammar given for $\mathcal{B}^{\Sigma,\mathcal{R}}$-processes. In this paper, we shall elide the full formal treatment of certain definitions derived directly from the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework, such as variables and their ordering, substitution, definitions of free and bound names, and certain normal forms. In this section, we limit ourselves to informal introductions. For a formal treatment, we refer the reader to the paper introducing $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculi [DK08]; for easy reference, we have included in Appendix 7.A the central definitions inherited from $\mathcal{B}^{\Sigma,\mathcal{R}}$.

In the remainder of the paper we shall overload the following names to be metavariables for domain and membrane-names: d and e are generic domain-names, m and n are generic membrane-names. Note, that this also means that later on, when we give rules using these names, formally these rules should be read as schemas for rules using concrete names. We summarize these metavariable conventions in Figure 7.5.

In Definition 7.2.4, we define formally solutions, and in Definition 7.2.5, we define solution groups. A solution is a special case of a group—namely, a group with exactly one solution. We use this in definitions and properties, and overload the symbols we use for operations such as structural congruence and substitution to work for both solutions and groups.

Definition 7.2.4 (solutions)**.** Solutions are collections of protein-domains, and channel-gates nested inside compartments induced by membranes. A solution may either by empty; a combination of two colocated solutions; a solution with a hidden name x; a domain $\mathsf{d} \in \mathcal{D}$ in one of three possible states linked to backbone b and (when bound) to complexation-link x; a gate $\triangle$ linked to a channel-link g; a membrane $\mathsf{m} \in \mathcal{M}$; or, a variable X.

We define solutions inductively as follows

$$
\begin{array}{rcll}
\mathsf{S},\mathsf{T} & ::= & 0 & \text{the empty solution} \\
& | & \mathsf{S} \mid \mathsf{T} & \text{parallel product} \\
& | & (x)\,\mathsf{S} & \text{new name} \\
& | & \mathsf{D} & \text{a domain} \\
& | & \triangle_g & \text{a channel-gate} \\
& | & \mathsf{m}[\mathsf{S}] & \text{a membrane} \\
& | & X & \text{variable}
\end{array}
$$

where

$$
\begin{array}{rcll}
\mathsf{D} & ::= & \mathsf{d}_b^x & \text{a bound domain} \\
& | & \mathsf{d}_b & \text{a visible and unbound domain} \\
& | & \overline{\mathsf{d}}_b & \text{a hidden domain}
\end{array}
$$

Definition 7.2.5 (groups)**.** Groups are defined inductively as

$$
\begin{array}{rcll}
\mathsf{G},\mathsf{F} & ::= & \mathsf{S} & \text{a single solution} \\
& | & (x)\,\mathsf{G} & \text{new name} \\
& | & \mathsf{G} \parallel \mathsf{F} & \text{wide parallel product} \\
& | & \epsilon & \text{the empty group}
\end{array}
$$

Call solutions or groups without variables *ground*. Call nonground solutions *presolutions*, and nonground groups *pregroups*—in general, we call nonground solutions or groups, *contexts*. We write $\mathcal{S}^{\Sigma}$ for the solutions over the signature Σ.

We use parentheses for grouping as usual. We let $\mid$ bind tighter than $\parallel$, which in turn binds tighter than the operator (x). When $\tilde{x} = \{x_1, \cdots, x_n\}$ we write $(x_1 \cdots x_n)\,\mathsf{S}$ or $(\tilde{x})\,\mathsf{S}$ to mean $(x_1) \cdots (x_n)\,\mathsf{S}$. Finally, as usual, we shall typically elide the 0 in empty membranes, for instance, writing $\mathsf{ves}[\,]$ instead of $\mathsf{ves}[0]$.

As usual, the new name operator $(x)\,\mathsf{S}$ is a binder on pure names, that is, instances of the pure name x in S are α-convertible. We define inductively the set of free, $\mathrm{fn}(G)$, or bound (pure) names, $\mathrm{bn}(G)$, of an expression as usual (cf. Definition 7.A.3 in Appendix 7.A). (Recall that names for proteins, and membranes are distinct and different from the pure names). We call a solution or group *closed* if all its pure names are bound; and *open* if it is not closed. Generally, we shall refer to pure names as just names, qualifying protein-names and membrane-names. We say that two solutions S and T are *connected* if they share free names, that is, if $\mathrm{fn}(\mathsf{S}) \cap \mathrm{fn}(\mathsf{T}) \neq \emptyset$.

Structural congruence We quotient solutions and groups according to a structural congruence relation, $\equiv$, enforcing prominently that names in the scope of a binder are α-convertible and that (x) floats freely in a term, as long as we do not capture free instances of the name x. We also allow order-preserving reordering of variables, for instance, that $\Box_0 \mid \Box_1 \equiv \Box_0 \mid \Box_2$. This formalizes the notion that we are only concerned with ordering of variables internal to an expression (cf. Definition 7.A.1 in Appendix 7.A). Furthermore, we make parallel product associative,

allow reordering, and stipulate that we may introduce (or delete) empty solutions. For groups, we make wide parallel product associative and make ϵ the neutral element.

Definition 7.2.6 (structural congruence)**.** Structural congruence, $\equiv$, on solutions 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, $|$, is associative and commutative with 0 as neutral element;
- wide parallel product, $||$, is associative with ϵ as neutral element;

and including the following *scope extrusion* laws

$$
\begin{array}{rcll}
\mathsf{m}[(x)\,\mathsf{S}] & \equiv & (x)\,\mathsf{m}[\mathsf{S}] & \text{(extrusion - mem)} \\
\mathsf{S} \mid (x)\,\mathsf{T} & \equiv & (x)\,\mathsf{S} \mid \mathsf{T} \quad \text{if } x \notin \mathrm{fn}(\mathsf{S}) & \text{(extrusion - par)} \\
((x)\,\mathsf{S}) \mid\mid \mathsf{T} & \equiv & (x)\,\mathsf{S} \mid\mid \mathsf{T} \quad \text{if } x \notin \mathrm{fn}(\mathsf{T}) & \text{(extrusion - wide par left)} \\
\mathsf{S} \mid\mid (x)\,\mathsf{T} & \equiv & (x)\,\mathsf{S} \mid\mid \mathsf{T} \quad \text{if } x \notin \mathrm{fn}(\mathsf{S}) & \text{(extrusion - wide par right)} \\
(x)(y)\,\mathsf{S} & \equiv & (y)(x)\,\mathsf{S} & \text{(reordering)} \\
(x)\,\mathsf{S} & \equiv & \mathsf{S} \quad \text{if } x \notin \mathrm{fn}(\mathsf{S}) & \text{(elision)}
\end{array}
$$

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

Substitution Variables in a solution S may be substituted for a group of solutions G. In the $\mathcal{C}$-calculus, we shall find use for both total substitution $\mathsf{S} \cdot \mathsf{G}$, and $\mathsf{S} \triangleleft \mathsf{G}$. The treatment of (variables and) substitution is transferred directly from the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework, so we introduce substitution only informally here (for easy reference, the definitions are included in Appendix 7.A, cf. Definitions 7.A.7 and 7.A.9).

We may, for instance, substitute three variables in a solution S by a group G of three solutions. Essentially, the resultant solution is computed by exchanging the ith ordered variable in S by the ith solution in G. If S has exactly three variables, the substitution is *total*, and we write $\mathsf{S} \cdot \mathsf{G}$ for the result. If S has more than three variables, the substitution is *partial*, and we write $\mathsf{S} \triangleleft \mathsf{G}$ for the result; remaining variables in S are reordered to have higher numbers than any variable in G. We distinguish the two kinds of substitution because total substitutions are associative while partial substitutions are not.

Notably, in parametric reaction rules, variables play the role of placeholders serving only to carry parameters across a reaction. For this usage, neither the numbers nor the ordering of variables matters (as we shall close the reaction relation under substitution for arbitrarily ordered groups of solutions). We require only that the same variables be used on both sides of a reaction rule. Therefore in reaction rules, we overload the usage of the metavariables $A, B, C, \ldots$ to allow them to be used instead of concrete variables (thus eliding the numbers of each variable).

7.2.4 Solutions as Graphs

For programming in the $\mathcal{C}$-calculus, and for giving and checking many properties, it is convenient to have a concise and formal term-base language as we have defined in the previous section. For visualization and for expressing certain properties and conditions it is convenient also to consider solutions as certain kinds of graphs.

Up until now, we have used a number of illustrations to illustrate solutions. Since the $\mathcal{C}$-calculus is formally founded on bigraphs, through the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework, these illustrations are actually based on a fully formal graphical language for solutions and groups, both ground and nonground. Formally, structural equivalence classes of solutions and groups correspond to certain bigraphs over a corresponding bigraph signature; hence, we may directly use the graphical language for bigraphs to describe $\mathcal{C}$-calculus-solutions. This is formally stated in [DK08, Proposition 5.4]; we shall not go into detail with that here.

For expressing and checking well-formedness conditions, it is however convenient to be able to describe and reason about solutions from a graph-based understanding of solutions. Hence, below we describe informally how solutions correspond to certain *bi*graphs—the combination of a tree and a hyper-graph sharing nodes.

First of all, every function symbol in the signature corresponds to the kind (or *control*) for a node in a bigraph. Eliding names, we may view a solution as a tree—sometimes called the *place* graph—with unordered children (parallel product is commutative) induced by the containment relation for membranes, whose leaves are domains, gates, variables, or empty solutions. In other words, the tree is simply the abstract syntax tree of the expression for a solution, only with unordered children to model directly the properties of the parallel product. Groups of solutions correspond to place graphs that are forests, instead.

Name-sharing between domains and gates creates a separate graph—sometimes called the *link* graph—of (hyper-)links across the tree. In closed solutions every link is an unnamed edge, while in open solutions, the links that correspond to free names are named.

When we depict bigraphs, instead of drawing the two graphs of bigraphs separately, we draw them in one picture, indicating the place graph with containment—just as we have done in the illustrations of $\mathcal{C}$-calculus solutions up until now (e.g., in Figure 7.2).

Finally, it may be convenient to point out, that graphically, we may think of substitution as inserting subtrees in the holes made by variables. Reactions, which we shall consider later, induce changes in the graph by creating or deleting nodes, by creating or severing links, or by moving entire subtrees (by way of parameters in reaction rules).

7.2.5 Well-formedness

The grammar for the $\mathcal{C}$-calculus allows us to write expressions for many solutions, which may be hard to interpret biologically. We wish to restrict to those solutions, which respect the motivations that we have given for each type of construct in the

language.

To capture such solutions we give a set of well-formedness criteria. It is convenient to state the requirements on solutions as properties of the (bi)graphs that $\mathcal{C}$-calculus solutions correspond to. The key theorem in the following section, which presents the operational semantics of the $\mathcal{C}$-calculus, states that well-formedness is preserved by the reaction relation induced by the class of reaction-rules that we allow. In verifying this theorem, we shall also make use of the bigraphical underpinning of the $\mathcal{C}$-calculus. We motivate each of the criteria in the paragraphs following the definition.

Definition 7.2.7 (well-formedness)**.** We say that a solution is well-formed, if

- (*link sorting*) links are well-sorted, i.e., for any two function symbols u, v, if the ith port of u and the jth port of v is part of the same link, then u and v are either two gates, or two domains and $j = i$;
- (*binary channels*) all gates are linked pairwise;
- (*binary complex*) complexation-links are binary;
- (*local complex*) complexation-links are mono-located, i.e., if two domains d and e are linked by their complexation-ports, then they are siblings;
- (*fixed backbone*) for any protein pro with signature (n, r), every domain $\mathsf{pro}i$ is linked by its backbone port to exactly $n-1$ domains $\mathsf{pro0}$, ..., $\mathsf{pro}i\text{-}1$, $\mathsf{pro}i{+}1$, ..., and $\mathsf{pro}n\text{-}1$.
  Further, if $r = 0$, all those domains of are siblings; else if $r > 0$, the domains $\mathsf{pro0}$, ..., $\mathsf{pro}r\text{-}1$ are siblings, and the domains $\mathsf{pro}r$, ..., $\mathsf{pro}n\text{-}1$ are siblings, and these two subunits of the protein are separated by a single membrane.
- (*bitonality*) all connected gates are separated by either 0 or 2 membranes.

Each kind of function symbol represents different types of entities. We require links to be *sorted*, such that links only connect entities from the same class. This ensures us that we only have links that we can interpret as protein backbones, domain-domain complexation links, and channels.

We require that channels and complex-formation links be *binary*. We have already discussed the motivation behind the channel-abstraction. Domain-domain complexation links represent a variety of weak forces attracting certain parts of proteins to each other. While at some level of abstraction, one could conceiveably consider domains interacting with more than one other domain at a time, we shall disregard that possibility in this exposition and require that complexation-links be binary.

We require also for complexation links that they be *local*, that is, that they do not cross membrane borders. The weakness of the forces working to establish complexes also enforce a high degree of locality. Only the backbone of a protein may cross membranes, the biological correspondent being that the amino-acid structure of a transmembrane protein actually penetrates the membrane.

We require that the backbone link of a protein is *fixed*, such that all instances of proteins respect their protein-signature (cf. Definition 7.2.1). This simply means, that for any protein pro with signature $(n, 0)$, all instances of pro takes the form

$$\mathsf{pro0} \mid \ldots \mid \mathsf{pro}n\text{-}1.$$

For any protein pro with signature (n, r) for $r > 0$, the following two kinds of configurations are well-formed

$$\mathsf{pro0} \mid \ldots \mid \mathsf{pro}r\text{-}1 \mid \mathsf{m}[\mathsf{pro}r \mid \ldots \mid \mathsf{pro}n\text{-}1 \mid \mathsf{S}]$$
$$\mathsf{m}[\mathsf{pro0} \mid \ldots \mid \mathsf{pro}r\text{-}1 \mid \mathsf{S}] \mid \mathsf{pro}r \mid \ldots \mid \mathsf{pro}n\text{-}1,$$

for some membrane-name $\mathsf{m} \in \mathcal{M}$ and arbitrary well-formed solutions S.

Thus, we may at all times identify the protein to which a domain belongs by its backbone.

Finally, we require that channels respect *bitonality*. This requirement deserves a bit more discussion.

On Bitonality

As discussed earlier, membranes consist of a lipid bilayer, two layers of lipids 'back-to-back'. Such a layer works as an *amphipathic* layer, that is, a layer with both hydrophilic and hydrophobic parts. This layer induces a strong barrier for fluids on the two sides of the membrane. In fact, these forces are also a main stabilizing factor of a membrane.

A direct consequence is that, in a calculus seeking to capture membrane interaction, we should try to enforce that entities may not cross membrane barriers, and that the reconfigurations of membranes only happen in such a way that the orientation of membranes is respected. As discussed by Cardelli for brane calculi [Car04b], we may abstractly capture this by considering the compartments induced by membranes to be coloured alternately black and white according to their nesting depth; and, essentially, then requiring of our operational semantics that it keeps black entities and white entities separate.

We should remark also that, while bitonality serves as good design abstraction, many important reactions in nature do, however, *not* respect bitonality. For instance, the plasma membrane of cells, is penetrated by many structures, some of which work as channels for the direct intake of smaller molecules.

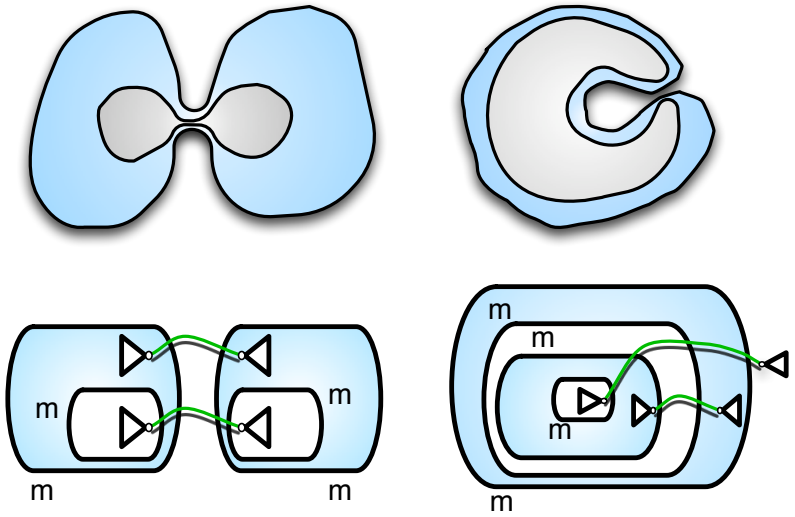
In the $\mathcal{C}$-calculus, any kind of transport between compartments involves channels, and we may enforce a correspondent to the bitonality constraint by requiring that channels form and exist between compartments separated by two membrane surfaces. This allows the basic configurations of membranes and channels that we have illustrated in Figure 7.3, but disallows a variety of more exotic configurations.

The basic configurations that we wish to disallow are those with channels of odd length, such as

$$(g) \triangle_g \mid \mathsf{m}[\triangle_g].$$

Although, as discussed, in nature limited transport across membranes *do* occur, we wish to keep channels as an abstraction for the neck of partially fused membranes.

In Figure 7.6, we illustrate two other prominent configurations that are not well-formed. We give the corresponding $\mathcal{C}$-calculus solutions below each illustration. In both cases, it is the h channel that is the perpetrator; it has been stretched, so to speak. On the left, two fusing membranes have stretched across a channel between another pair of partially fused membranes. On the right, we have a similar situation, only with the direction of the fusing membranes inverted.



$(g,h)\,\mathsf{m}[\triangle_g \mid \mathsf{m}[\triangle_h]\!] \mid \mathsf{m}[\triangle_g \mid \mathsf{m}[\triangle_h]\!]$ $\qquad$ $(g,h)\,\triangle_h \mid \mathsf{m}[\triangle_g \mid \mathsf{m}[\mathsf{m}[\triangle_g \mid \mathsf{m}[\triangle_h]\!]]\!]$

Figure 7.6: Examples of disallowed membrane configurations, and their representation via channels in the $\mathcal{C}$-calculus.

We disallow these configurations in our definition of bitonality, not because they would allow colours to mix (only channels of odd length would do that), but because we would struggle to interpret them biologically. Though biological membranes are quite deformable, their plasticity still makes configurations such as those depicted in Figure 7.6 very improbable. The examples are prominent, however; we need to take care in the treatment of reaction rules that allow creating or transporting membranes, to disallow reactions that lead to these configurations from well-formed ones. Consider, for instance, this (well-formed) configuration

$$(g,h)\,\mathsf{m}[\triangle_g \mid \mathsf{m}[\triangle_h] \mid \mathsf{m}[\triangle_h]\!] \mid \mathsf{m}[\triangle_g].$$

If we allowed one of the membranes with a gate $\triangle_h$ to be transported across the g channel, we get the leftmost configuration in Figure 7.6. We may write a similar wellformed pre-configuration for the rightmost configuration. We shall discuss how to avoid such mishaps in Section 7.3.

As a special case we allow degenerate channels such as

$$\mathsf{U} = (g)\,\mathsf{m}[\triangle_g \mid \triangle_g].$$

As shall be apparent when we introduce the operational semantics, they may be formed when colocated membranes form several channels between themselves, before fusing. For instance, we shall allow the following chain of reactions to form the solution U:

$$\mathsf{m}[] \mid \mathsf{m}[] \to (g)\,\mathsf{m}[\triangle_g] \mid \mathsf{m}[\triangle_g] \to (g,h)\,\mathsf{m}[\triangle_g \mid \triangle_h] \mid \mathsf{m}[\triangle_g \mid \triangle_h] \to (g)\,\mathsf{m}[\triangle_g \mid \triangle_g].$$

We may illustrate this chain of reactions as in Figure 7.7.

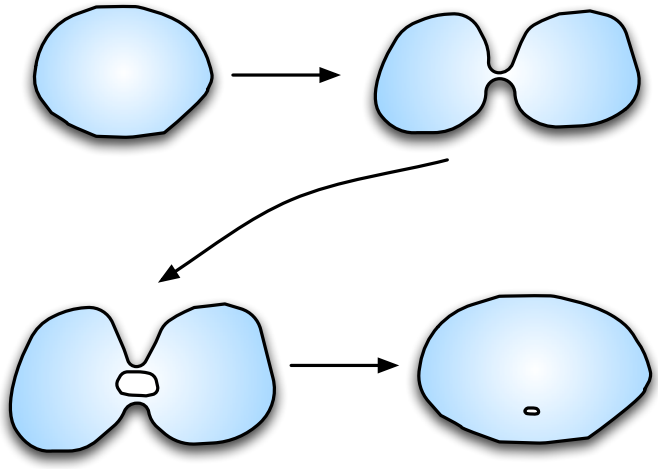


Figure 7.7: Multi-touch may form degenerate channels.

The first reaction is a basic instance of a so-called *touch*; so is the second reaction. The third reaction is a *fusion* around the h channel, which leaves the g channel as a residual of the first touch.

One could contemplate giving side-conditions on touch-rules to disallow the second reaction; it turns out that this is not sufficient for preserving well-formedness across reaction, however. Multi-channels may be formed by fusion as consequence of a sequence of earlier touch events, each of which should be locally allowable, in the sense that no multi-channels are created. A minimal situation is illustrated in Figure 7.8. In general, testing for multi-channels is highly non-local requiring a traversal of the entire link graph of channels. And disallowing the final fusion in Figure 7.8 would be counter-intuitive with regard to biological consistency. Indeed, though multi-touches and the shape of membranes in the second step of Figure 7.8 are highly unlikely to happen in nature, they are not impossible.

For the reasons discussed above, we choose to allow multi-touches. This keeps our reaction rules simpler, and it allows us to model membranes still separate, but in different stages of fusing. In turn, the design choice has the side-effect that degenerate channels can be formed. Such degenerate channels are, however, harmless; no diffusion or reconfiguration may occur along them.[7]

[7]We may even consider extending the structural congruence with a rule to "garbage collect"

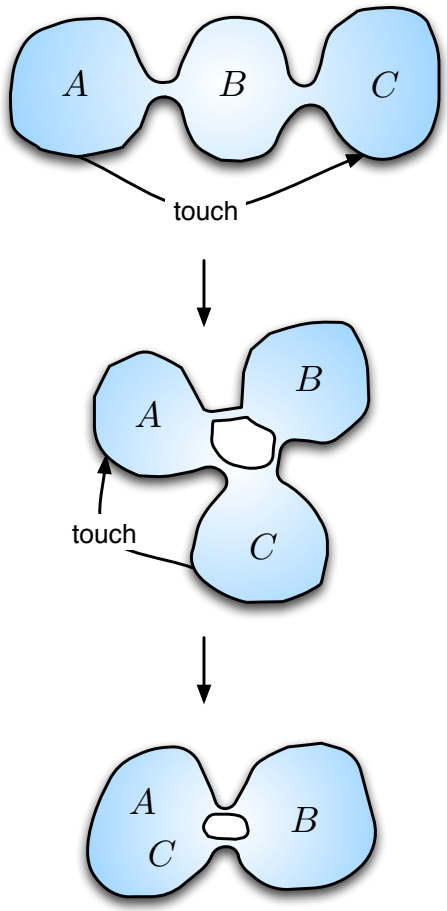


Figure 7.8: A multi-touch as a consequence of earlier touches.

7.3 Dynamics

In this section, we introduce the operational semantics of the $\mathcal{C}$-calculus.

A user develops a model in the $\mathcal{C}$-calculus by selecting a set a proteins and membranes to work with (by giving a protein-signature, and a set of membrane-names), and a set of reaction rules. We cannot allow any collection of reaction rules; such a loose policy would not allow us to maintain well-formedness. This would also lay upon the shoulders of modellers the task of inventing their own abstractions of biological events; this in turn would complicate, for instance, composition of different models, something which we would like to be an easy task. Instead, we characterize in this section a fixed set of canonical biological *actions*, encapsulated in classes of certain reaction rules. We allow modellers to refine these actions by

them, i.e.: $(g) \triangle_g \mid \triangle_g \equiv 0$. However, that would remove the 1-1 correspondence between solutions and bigraphs, in turn preventing us from using bigraphical reasoning in the proofs. Hence, we leave this idea for future work.

giving contextual preconditions for rules. In all, we characterize a set of core rules, which provide modellers with a toolbox of generic actions that can be incorporated into a model by specializing them to a particular signature. Given this toolbox of actions, a $\mathcal{C}$-calculus reactive system can essentially be designed by domain experts (rather than computer scientists).

In summary, modelling in $\mathcal{C}$-calculus is done by giving a signature and choosing and refining a set of core rules over this signature. Thus, as set out in the Introduction, the slogan for modelling in the $\mathcal{C}$-calculus is: *Modelling by rule refinement.*

First, in Section 7.3.1 we define $\mathcal{C}$-calculus reactive system and explain reaction rules. In Section 7.3.2, we treat so-called *projective* descriptions of rules. They let us describe conveniently sets of rules for reactions involving regions separated by membrane-surfaces, while eliding the *orientation* of those separating membranes. We continue to introduce reaction rules; they divide into two categories. In Section 7.3.3, we discuss rules that involves protein-protein interaction, that is, forming or breaking complexation-links, changing the state of domains, or creating or deleting proteins. These kinds of rules work essentially in the κ-like fragment of the calculus, and, as we have treated an encoding of the κ-calculus before [DK08], we shall go into less detail with these kinds of rules. In Section 7.3.4, we discuss rules involving membrane (or channel) reconfiguration; the rules for transport of material along channels need special care—we shall discuss and motivate them at length. In Section 7.3.5, we formally define the allowable rules, and, in Section 7.3.6 we verify that any allowable rule preserves well-formedness.

7.3.1 $\mathcal{C}$-calculus Reactive Systems

In the $\mathcal{C}$-calculus, we work with reaction rules of the following format:$(\mathsf{L} \to \mathsf{R}, \varphi)$, where L and R are expressions for solutions and φ is a side-condition. The expressions may have a number of variables, which serve to carry parameters unchanged across from the left-hand side to the right-hand side. The side-condition φ is a (possible empty) predicate that tests parameters. We shall use side-conditions for expressing contextual conditions on the surroundings of L and R for the rules handling transport.

The $\mathcal{C}$-calculus reaction rules are fairly simple (as compared to the generality allowed for bigraphical rules, say). We need only consider rules, where free names are preserved, that is, $\mathrm{fn}(\mathsf{L}) = \mathrm{fn}(\mathsf{R})$, and where variables occur exactly once on the left-hand side and on the right-hand side (i.e., all rules are linear). These are exactly the kind of rules, we have treated in the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework [DK08], and we may instantiate the theory for reaction and reactive systems from that work. We outline how reaction works below. (As verified in *loc. cit.*, it also follows that $\mathcal{C}$-calculus reaction is bigraphical reaction modulo side-conditions, which ensures us that we may reason (bi)graphically in proofs.)

Loosely, any ground solution S that is a substitution-instance of the left-hand side of a rule may be rewritten with that rule. Given a rule $(\mathsf{L} \to \mathsf{R}, \varphi)$, if for some (ground) group G, we have $\mathsf{S} \equiv \mathsf{L} \cdot \mathsf{G}$, we may perform a reaction $\mathsf{S} \to \mathsf{S}' \equiv \mathsf{R} \cdot \mathsf{G}$, if $\varphi(\mathsf{G})$ is satisfied. We say that S *matches* L with the *parameter* G, since G consists of

the group of solutions that we will substitute for the variables in R. Reactions may occur in any solution context. We make this explicit in Definition 7.3.1.

We contextualize reactions according to standard tradition for process calculi. We close reactions under syntactic constructions, structural congruence, and also under (bijective) renaming of free names. Definition 7.3.1 records a set of rules that together gives a simple characterization of the reactive system over solutions. The definition is a straight instantiation of the generic operational semantics given for $\mathcal{B}^{\Sigma,\mathcal{R}}$ (in [DK08, Definition 3.16]). Free names are preserved across reaction, that is, if $\mathsf{S} \to \mathsf{S}'$, $\mathrm{fn}(\mathsf{S}) = \mathrm{fn}(\mathsf{S}')$.

As outlined in the introduction to this section, we do not wish, however, to allow all kinds of reaction rules. Therefore, in the Definition 7.3.1, we have restricted to a set of (as yet unspecified) *allowable* reaction rules. Informally, we wish to allow rules that encapsulates a single biological action, such as breaking a domain-domain bond or fusing two partially fused membranes. The following sections shall be concerned with introducing core rules that incorporate biological actions (Sections 7.3.3 and 7.3.4).

Definition 7.3.1 (reactive system)**.** Given a set of *allowable* reaction rules $\mathcal{R}$ and a signature Σ, $\mathcal{T}^{\mathcal{R}}_{\Sigma}$ the reactive system associated with $\mathcal{R}$, is given by the reaction relation $\to$, the least binary relation over $\mathcal{S}^{\Sigma}$, s.t.

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

$$\textsc{par}\frac{\mathsf{S} \to \mathsf{S}'}{\mathsf{S} \mid \mathsf{T} \to \mathsf{S}' \mid \mathsf{T}} \qquad \textsc{membrane}\frac{\mathsf{S} \to \mathsf{S}'}{\mathsf{m}[\mathsf{S}] \to \mathsf{m}[\mathsf{S}']}$$

$$\textsc{close}\frac{\mathsf{S} \to \mathsf{S}'}{(x)\mathsf{S} \to (x)\mathsf{S}'} \qquad \textsc{struct}\frac{\mathsf{T} \equiv \mathsf{S} \qquad \mathsf{S} \to \mathsf{S}' \qquad \mathsf{S}' \equiv \mathsf{T}'}{\mathsf{T} \to \mathsf{T}'}$$

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

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

However, we also wish to allow a modeller to test parts of the surroundings, to give application conditions for reactions. We allow a modeller to refine core rules by adding extra context to rules. It turns out that we may capture refinement succinctly with substitution. We define formally refinement and allowable rules in Section 7.3.5 (allowing us to explain and motivate refinement with a concrete example using a core rule introduced in the previous sections). Figure 7.9 illustrates schematically how refinement works. Suppose we have the core rule $\mathsf{L} \to \mathsf{R}$, where L and R have two variables each (depicted as dark triangles inside L and R). The user may refine that rule by substituting both sides L and R into an external context, T (with one variable), and by substituting a common group of solutions into the variables of L and R, yielding the rule $\mathsf{T} \cdot \mathsf{L} \cdot \mathsf{F} \to \mathsf{T} \cdot \mathsf{R} \cdot \mathsf{F}$. For full generality, we still need to treat rules with side-conditions, however; we return to refinement in detail in Section 7.3.5.

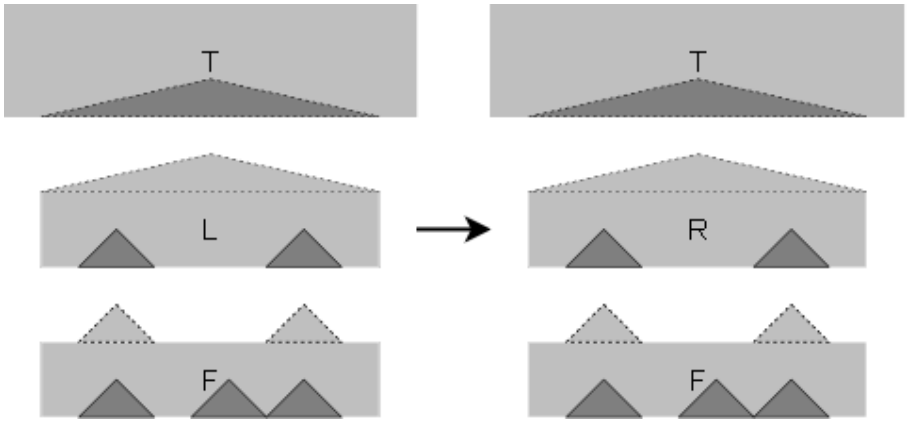
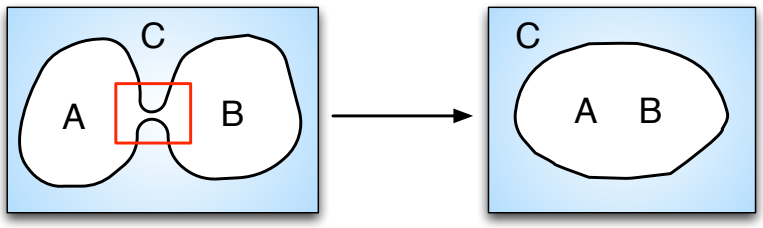


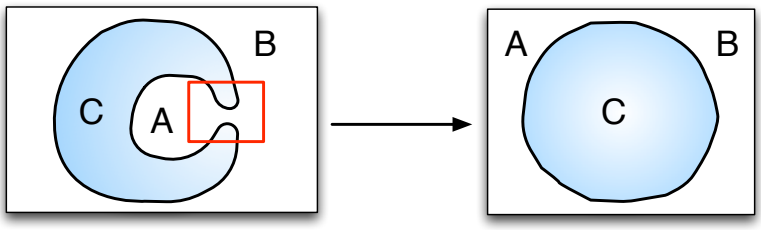
Figure 7.9: Illustrating refinement of a rule.

7.3.2 Projective Descriptions

The model of membranes in the $\mathcal{C}$-calculus is influenced by Cardellis brane calculi [Car04b]. We build, however, also on the insights reported by Pradalier and Danos on the so-called *projective* brane calculi [DP04], and, more recently, *patch* reactions by Cardelly [Car08]. Their insights were to note, that for membrane calculi all reactions that are described for membranes occur irrespective of membrane orientation. For instance, the reaction



is allowed, and so is



If we focus on the local patches of membranes that are merging (indicated by the rectangular lenses), we may observe that from a *local* perspective, the preconditions for the reactions look highly similar. Only the *orientation* of the curvature of the

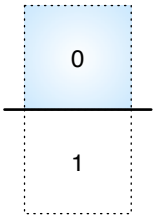
membranes is different—a property, which is hardly detectable in nature.[8]

Indeed, some descriptions of biological reactions abstract away from the orientation of the membrane, describing instead only the conditions local to a patch of a membrane wall. It does, of course, matter whether a signal propagates inwards or outwards across the cellular membrane; the central observation is, that this is *not* due to the orientation of the cellular membrane.

We wish to allow such descriptions for $\mathcal{C}$-calculus reaction rules; adopting the terminology introduced by Pradalier and Danos, we call the descriptions *projective*. To that end, we start by defining lateral and horizontal *projection* of binary groups of solutions.

Projective Contexts and Projective Groups

In essence, projective descriptions, involve two regions separated by a membrane wall, as depicted below:



We may interpret such descriptions as a schema for two reaction rules, one for each orientation of the separating membrane. In Figure 7.10, we illustrate the two ways one may interpret a description of two regions separated by a single membrane wall. As depicted, it simply boils down to determining which of the regions numbered 0 or 1 contain the other. We may colour the regions in accordance with the colouring scheme discussed for the bitonality constraint in the previous section, and call the descriptions *heterogenous*, as they involve differently coloured regions.

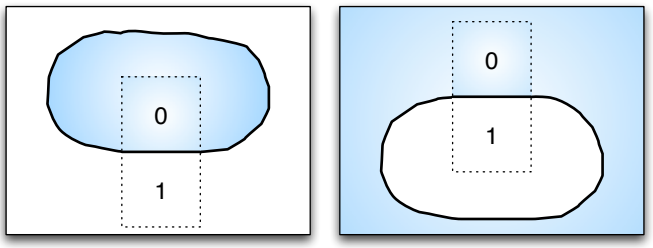
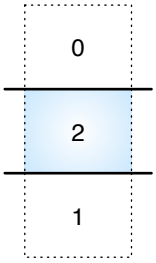


Figure 7.10: Heterogenous projective contexts illustrated.

Similarly projective descriptions of reactions involving regions separated by two membranes, involve three regions as depicted below:

[8] Although, as reported in [HR05], curvature *does* occasionally matter and may serve to distinguish and sort vesicles by size.



We call such descriptions *homogenous* as the regions 0 and 1 have equal colour. In Figure 7.11, we illustrate the three ways one may orient the membranes, such that the regions 0 and 1 are separated (three, as the first is symmetric). Consequentially,

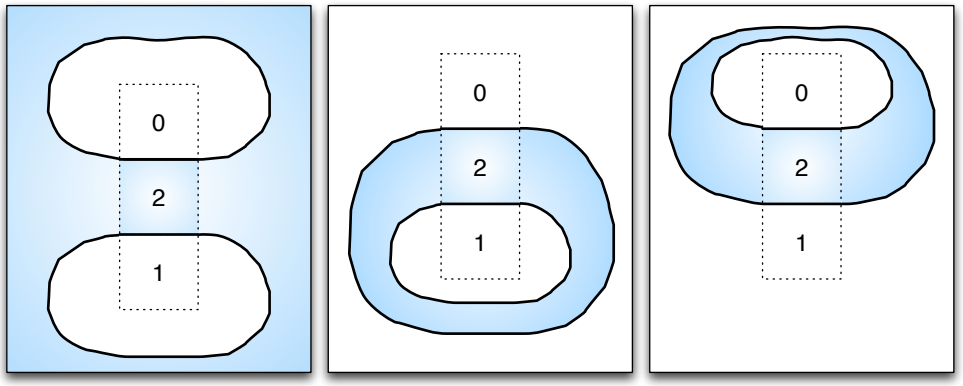


Figure 7.11: Homogenous projective contexts illustrated.

we may read homogenous projective descriptions as a schema for three rules.

We may generate projections by substitution into sets of *projective contexts*, defined to match each of the projections in Figures 7.10 and 7.11. Formally, we let the two sets of projective contexts be parameterized over one or two names (for heterogenous and homogenous contexts, respectively) to allow projection across any kind of membrane(s).

Definition 7.3.2 (Projective contexts)**.** For all $\mathsf{m}, \mathsf{n} \in \mathcal{M}$, the *heterogenous* projective contexts $\eta_{\downarrow}^{\mathsf{m}}$ and $\eta_{\uparrow}^{\mathsf{m}}$, and the *homogenous* projective contexts $\omega_{\downarrow}^{\mathsf{m},\mathsf{n}}$, $\omega_{\uparrow}^{\mathsf{m},\mathsf{n}}$, and $\omega_{\leftrightarrow}^{\mathsf{m},\mathsf{n}}$ are

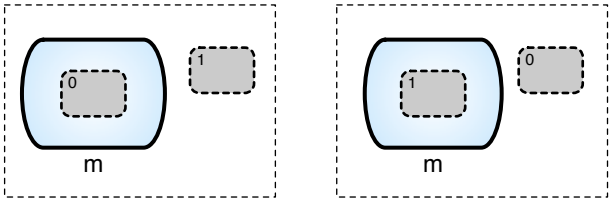
$$\begin{array}{llll} \omega_{\downarrow}^{\mathsf{m},\mathsf{n}} & = \mathsf{m}[\mathsf{n}[\Box_1] \mid \Box_2] \mid \Box_0 & \eta_{\downarrow}^{\mathsf{m}} & = \mathsf{m}[\Box_1] \mid \Box_0 \\ \omega_{\uparrow}^{\mathsf{m},\mathsf{n}} & = \mathsf{m}[\mathsf{n}[\Box_0] \mid \Box_2] \mid \Box_1 & \eta_{\uparrow}^{\mathsf{m}} & = \mathsf{m}[\Box_0] \mid \Box_1 \\ \omega_{\leftrightarrow}^{\mathsf{m},\mathsf{n}} & = \mathsf{m}[\Box_0] \mid \mathsf{n}[\Box_1]. & & \end{array}$$

The sets of generic homogenous and heteregonous projective contexts are

$$\begin{array}{ll} \omega_\delta = & \{\mathsf{S} \mid \exists \mathsf{m}, \mathsf{n} \in \mathcal{M}.\mathsf{S} = \omega_\delta^{\mathsf{m},\mathsf{n}}\}, \text{for } \delta \in \{\uparrow, \downarrow\} \\ \eta_\delta = & \{\mathsf{S} \mid \exists \mathsf{m}, \mathsf{n} \in \mathcal{M}.\mathsf{S} = \eta_\delta^{\mathsf{m}}\}, \text{for } \delta \in \{\uparrow, \downarrow, \leftrightarrow\}. \end{array}$$

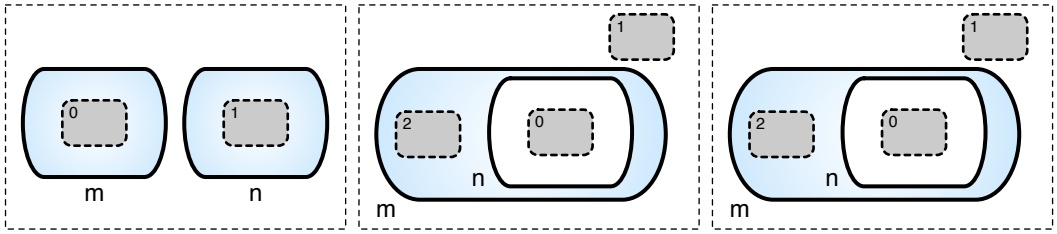
We shall typically use the sets of generic projective contexts, because we want to refer to the set of projective contexts generated by all valid membrane-names; for instance, $\omega_\downarrow = \{\mathsf{S} \mid \exists \mathsf{m}, \mathsf{n} \in \mathcal{M}.\mathsf{S} = \omega_\downarrow^{\mathsf{m},\mathsf{n}}\}$. In the following we shall allow ourselves a convenient sloppyness: to use directly in expressions the unqualified projective contexts, such as in the expression $\omega_\downarrow \triangleleft \mathsf{G}$. Formally, this means that we are talking about a set of solutions $\{\mathsf{S} \triangleleft \mathsf{G} \mid \exists \mathsf{m}, \mathsf{n} \in \mathcal{M}.\mathsf{S} = \omega_\downarrow^{\mathsf{m},\mathsf{n}}\}$. This shall be particularly convenient when writing schemas for rules.[9]

We depict the contexts defined in Definition 7.3.2 in Figures 7.12 and 7.13 (colouring the compartments for effect).



$\eta_\uparrow^{\mathsf{m}} = \mathsf{m}[\Box_0] \mid \Box_1$ $\eta_\downarrow^{\mathsf{m},} = \mathsf{m}[\Box_1] \mid \Box_0$

Figure 7.12: Heterogenous projective contexts.



$\omega_\leftrightarrow^{\mathsf{m},\mathsf{n}} = \mathsf{m}[\Box_0] \mid \mathsf{n}[\Box_1]$ $\omega_\uparrow^{\mathsf{m},\mathsf{n}} = \mathsf{m}[\mathsf{n}[\Box_0] \mid \Box_2] \mid \Box_1$ $\omega_\downarrow^{\mathsf{m},\mathsf{n}} = \mathsf{m}[\mathsf{n}[\Box_1] \mid \Box_2] \mid \Box_0$

Figure 7.13: Homogenous projective contexts.

Now we may define heterogenous and homogenous projection of binary groups by way of substitution.

Definition 7.3.3. Given a binary group $\mathsf{G} = \mathsf{S} \mid \mathsf{T}$, its heterogenous projections is the sets of solutions generated by

$$\eta_\downarrow \triangleleft \mathsf{G}, \text{ and } \eta_\uparrow \triangleleft \mathsf{G},$$

and its homogenous projections is the sets of solutions generated by

$$\omega_\downarrow \triangleleft \mathsf{G}, \omega_\uparrow \triangleleft \mathsf{G}, \text{ and } \omega_\leftrightarrow \triangleleft \mathsf{G}.$$

[9] In the following definitions, we shall consistently use partial substitution $\mathsf{G} \triangleleft \mathsf{S}$, though total substitution $\mathsf{G} \cdot \mathsf{S}$ would be sufficient in some cases (specifically, for heterogenous contexts). (Recall, that $\triangleleft$ denotes partial substitution—see Section 7.2.3 and Def. 7.A.9).

Letting δ range over the directions $\{\uparrow, \downarrow, \leftrightarrow\}$ (up, down, and lateral), we may summarize Definition 7.3.3, by stating that the heterogenous projections are those characterized by

$$\eta_\delta \lhd \mathsf{G},$$

while the homogenous projections can be characterized as

$$\omega_\delta \lhd \mathsf{G}.$$

Projective Rules

We utilize our characterization of projective groups, to define projective descriptions of reaction rules.

We define projective *generators*, projective descriptions that constitute schemas for rules, which we *project* to form concrete rules.

Let $\rho \in \{\omega, \eta\}$, and recall that $\delta \in \{\uparrow, \downarrow, \leftrightarrow\}$. We write projective generators as

$$\mathsf{rule}^\rho : \mathsf{L} \to_\rho \mathsf{R},$$

using the subscripted ρ as reminder that these are rule-schemas that await a projective direction before application.

From generators, we generate *projected* rules. For homogenous generators we have

$$\mathsf{rule}^\omega_\delta : \omega_\delta \lhd \mathsf{L} \to \omega_\delta \lhd \mathsf{R},$$

and for heterogenous generators we have

$$\mathsf{rule}^\eta_\delta : \eta_\delta \lhd \mathsf{L} \to \eta_\delta \lhd \mathsf{R}.$$

For some generators, we shall declare once and for all, that they are either heterogenously or homogenously projective, and elide the projection, ρ, from the name of the rule. We depict generators in correspondence with the illustrations for projective groups above, separating groups with one or two horizontal lines denoting the projected membranes.

Expanding on the illustration of refinement in Figure 7.9, Figure 7.14 illustrates schematically how generators are projected to form concrete rules, before giving application conditions for them. In the illustration, L and R are groups containing two solutions, and we suppose that $\mathsf{L} \to_\rho \mathsf{R}$ is a heterogenous generator. By inserting the group into the heterogenous projective contexts, we may obtain (projected) rules. These rules we may refine as usual.

7.3.3 Protein-protein Interaction

Our domain-level model of proteins is closely related to that of the κ-calculus, only generalized to a distributed setting by using backbone links instead of nodes to connect domains of the same protein. In the $\mathcal{C}$-calculus, we shall allow essentially the same kind of reactions as in the κ-calculus, that is, rules creating or destroying

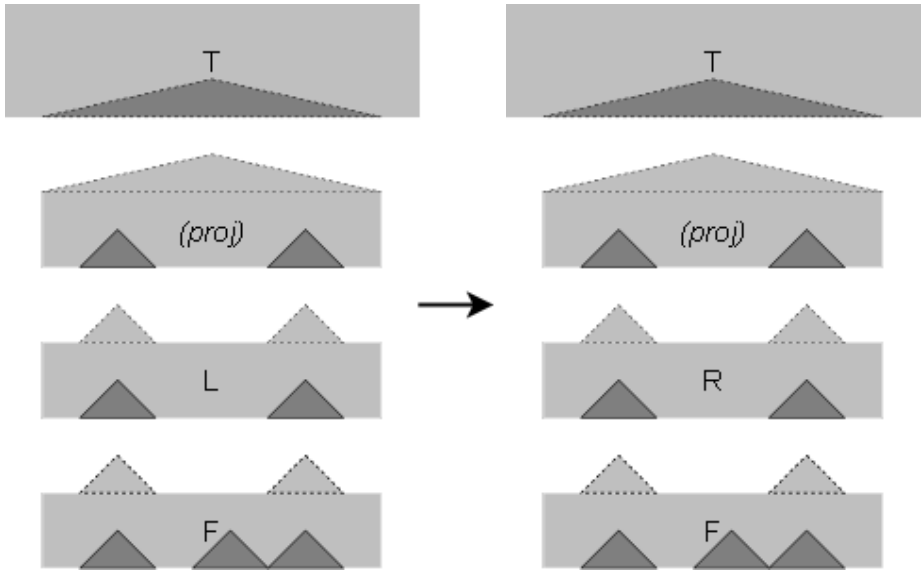


Figure 7.14: Illustrating refinement of projective generators.

protein complexation-links and/or changing domain state. We also take rules that create or delete proteins, whose domains are all unbound (i.e., visible or hidden).

For the κ-calculus it has been discussed in detail, how one may restrict to rules with a suitable level of atomicity (see [DL04]). This level of atomicity may be enforced by restricting to so-called *monotone* and *anti-monotone* reaction rules. As we have essentially imported a generalized version of the core κ-calculus inside the $\mathcal{C}$-calculus, we may adopt such (anti-)monotonicity requirements more or less directly. We elide such treatment here, as the restrictions are not central for the main theorem—the preservation of well-formedness—that we wish to establish; and because we wish to focus on the extension of a κ-like calculus with dynamic compartments.

We take instead a simpler set of rules that allow to us capture core κ-reactions. Definition 7.3.4 defines base domain-level rules; a set of rules, which each incorporate a single core biological action.

Definition 7.3.4 (base domain-level rules)**.** For arbitrary domains d and e, we allow the following rules:

$$\begin{array}{lrcl}
\textsc{bind}: & \mathsf{d}_a \mid \mathsf{e}_b & \rightarrow & (x)\,(\mathsf{d}^x_a \mid \mathsf{e}^x_b) \\
\textsc{break}: & \mathsf{d}_a \mid \mathsf{e}_b & \leftarrow & (x)\,(\mathsf{d}^x_a \mid \mathsf{e}^x_b) \\
\textsc{show}: & \overline{\mathsf{d}}_a & \rightarrow & \mathsf{d}_a \\
\textsc{hide}: & \overline{\mathsf{d}}_a & \leftarrow & \mathsf{d}_a.
\end{array}$$

We allow also rules for *synthesis*, protein-creation, and for *degradation*, protein-deletion. These kinds of rules come in two variants depending on the signature of

the protein involved. For proteins p with $\Sigma^{\mathcal{P}}(\mathsf{p}) = (a, 0)$, they take the forms:

$$\begin{array}{lll} \textsc{synth}: & 0 \rightarrow & (b)\,\big(\dot{\mathsf{p}}\dot{0}_b \mid \ldots \mid \mathsf{p}\dot{a}\text{-}1_b\big) \\ \textsc{degrade}: & 0 \leftarrow & (b)\,\big(\dot{\mathsf{p}}\dot{0}_b \mid \ldots \mid \mathsf{p}\dot{a}\text{-}1_b\big)\,, \end{array}$$

where we use the shorthand $\dot{\mathsf{p}i}$ to range over visible or hidden domains $\mathsf{p}i$, i.e., $\dot{\mathsf{p}i} \in \{\mathsf{p}i, \overline{\mathsf{p}i}\}$. (In other words, we allow SYNTH and DEGRADE rules to create or delete domains that are unbound—whether visible or hidden.)

For receptor-proteins p, with $\Sigma^{\mathcal{P}}(\mathsf{p}) = (a, r)$ for $r > 0$, synthesis and degradation are described by the following projective generators:

$$\begin{array}{lll} \textsc{synth}: & 0 \,||\, 0 \rightarrow_\eta & (b)\,\big(\dot{\mathsf{p}}\dot{0}_b \mid \ldots \mid \mathsf{p}\dot{r}\text{-}1_b \,||\, \dot{\mathsf{p}}r_b \mid \ldots \mid \mathsf{p}\dot{a}\text{-}1_b\big) \\ \textsc{degrade}: & 0 \,||\, 0 \leftarrow_\eta & (b)\,\big(\dot{\mathsf{p}}\dot{0}_b \mid \ldots \mid \mathsf{p}\dot{r}\text{-}1_b \,||\, \dot{\mathsf{p}}r_b \mid \ldots \mid \mathsf{p}\dot{a}\text{-}1_b\big) \end{array}$$

The rules are pairwise inverses. We can summarize the actions in the reaction rules as follows:

- BIND establishes binary complexation links between visible and co-located domains;
- BREAK breaks closed complexation links between (bound) domains;
- HIDE/SHOW changes the domain-state between visible and hidden;
- SYNTH allows creation of the domains and the backbone that constitutes a protein; all domains should be unbound (i.e., visible or hidden); and,
- DEGRADE allows deletion of the domains and the backbone that constitutes a protein; all domains should be unbound (i.e., visible or hidden).

For SYNTH (and DEGRADE), we require further that the protein is created (or deleted) in a configuration, which respects its protein signature (cf. Definition 7.2.1).

Together the actions incorporated into the base rules allow us to capture reaction for proteins comparable to those in the κ-calculus, extended to a multi-compartment setting, but with considerably tighter atomicity-requirements on reactions. In the κ-calculus, one may, for instance, break several complexation-links in unison or synthesize proteins and bind them to other proteins in one go. It is not particularly hard to relax the atomicity requirements we have given on domain-level rules in the $\mathcal{C}$-calculus; however, in this paper, we shall make do with a few extra rules combining the actions captured in the base domain-level rules.

Definition 7.3.5 (domain-level rules)**.** Domain-level rules are those described by BIND, BREAK, HIDE, SHOW, SYNTH, and, DEGRADE above, and the following rules (for arbitrary domains d, e and f):

$$\begin{array}{lllll} \textsc{bind+hide}: & \mathsf{d}_a \mid \mathsf{e}_b \mid \mathsf{f}_c & \rightarrow & (x)\,\big(\mathsf{d}^x_a \mid \mathsf{e}^x_b \mid \bar{\mathsf{f}}_c\big) \\ \textsc{break+show}: & \mathsf{d}_a \mid \mathsf{e}_b \mid \mathsf{f}_c & \leftarrow & (x)\,\big(\mathsf{d}^x_a \mid \mathsf{e}^x_b \mid \bar{\mathsf{f}}_c\big) \\ \textsc{bind+show}: & \mathsf{d}_a \mid \mathsf{e}_b \mid \bar{\mathsf{f}}_c & \rightarrow & (x)\,(\mathsf{d}^x_a \mid \mathsf{e}^x_b \mid \mathsf{f}_c) \\ \textsc{break+hide}: & \mathsf{d}_a \mid \mathsf{e}_b \mid \bar{\mathsf{f}}_c & \leftarrow & (x)\,(\mathsf{d}^x_a \mid \mathsf{e}^x_b \mid \mathsf{f}_c)\,. \end{array}$$

Together the extra kinds of rules, BIND+SHOW, BREAK+HIDE, BIND+HIDE, and BREAK+SHOW, allows us to model that creation or deletion of a complexation-link may invoke a change of conformation on a protein.

One can also combine other of the base rules, obtaining rules such as SYNTH+BIND to synthesize a protein *and* bind one of its domains to an existing domain, or BIND+BIND to allow two complexation-links to be created in one go. For the concrete examples we shall discuss in Section 7.4, the set of rules captured in Definition 7.3.5 is sufficient, however.

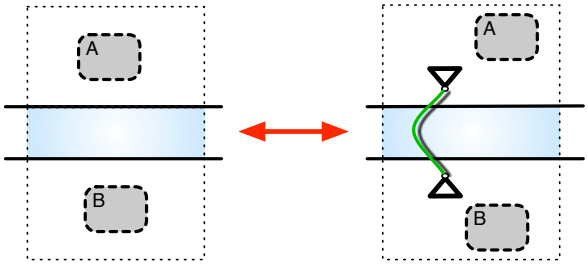
7.3.4 Membrane Reconfiguration

Membrane-based actions model structural reorganization of membranes such as fusion or fission. As we discussed in Section 7.2.5, as studied by pioneering works [DP04, Car08], oriented membrane interactions can be factored into a small set of projective interactions, which do not take into account the global orientation of the involved membranes. We capture this by giving generators for projective rules as defined in Section 7.3.2.

Touch and Part

We start by describing generators incorporating the dual actions of establishing and breaking a channel between colocated membranes. In terms of the bitonality constraint, these involve regions separated by two membrane surfaces; hence the generators are homogenous, that is, we generate rules by composing with the homogenous contexts (cf. Definition 7.3.2).

The TOUCH and PART generators are each others inverses; they are depicted in Figure 7.15. The PART generator allows us to model the first step of a fusion process. It creates a channel between two colocated membranes, bringing them into a partially fused state.



TOUCH ($\rightarrow_\omega$) and PART ($\leftarrow_\omega$): $A \,||\, B \;\leftrightarrow_\omega\; (x)(\triangle_x \mid A \,||\, \triangle_x \mid B)$

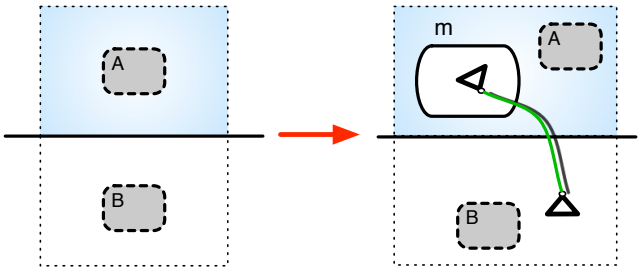
Figure 7.15: The homogenous generators TOUCH and PART.

The PART generator is the counterpart of TOUCH; it allows one membrane to detach from another by severing a channel that links the two.

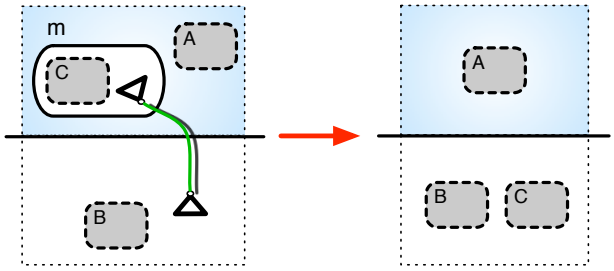
Bud and Merge

In this section, we treat generators incorporating the actions of creating and destroying a membrane. These generators are heterogenous; they involve a single membrane, which bends and *buds* a new small compartment; or, which *merges* entirely with another compartment that it is partially fused with.

The BUD generator, on the left in Figure 7.16, describes how a membrane may start to divide to create a new compartment.



BUD: $A \,||\, B \;\rightarrow_\eta\; (x)(A \mid \mathsf{m}[\triangle_x] \,||\, B \mid \triangle_x)$



MERGE: $(x)(A \mid \mathsf{m}[\triangle_x \mid C] \,||\, \triangle_x \mid B) \rightarrow_\eta A \,||\, C \mid B$

Figure 7.16: The heterogenous generators BUD and MERGE.

The MERGE generator, on the right in Figure 7.16, describes how a membrane may finish its fusion with another membrane. A membrane merges into another by releasing its content, C, into the second membrane. The MERGE generator clearly generates left inverses of BUD (take C to be empty).

Note, that the MERGE generator applies even if the involved membranes have channels—to other membranes in the context, or to the projected membrane containing B. (The latter case, is the one we have discussed already in Section 7.2.5.) This neatly captures that after merging two membranes, the resulting membrane inherits the partial fusings of both membranes.

The BUD generator creates an empty compartment that is still connected to the original membrane via a channel. This channel will allow us to model exchange of materials between the two compartments (by diffusion, which we shall treat below). For simple models where empty vesicles are budded from the cellular membrane, this may be sufficient. However, in most natural cases, a bud is formed around

something, for instance, a complex bound to the cell wall. (For a concrete example, see the model of the formation of clathrin-coated vesicles in Section 7.4.2.) To allow this, we need to allow the (deterministic) transfer of a parameter into the newly formed membrane. To fullfill that need, in the following section we define the PINCH generator, a generalization of the BUD generator.

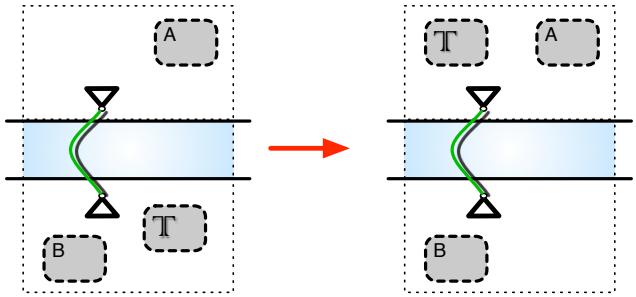
Transport Rules

In this section, we shall discuss how to encapsulate *transport*. We define two generators, one that encapsulates the action of diffusion, and another, the PINCH generator, a generalization of BUD introduced above. These generators involve transport of complexes across channels, and we need to take care to uphold well-formedness. (The MERGE generator already allows transport, of course, but in a manner non-problematic with regard to well-formedness.)

We start by developing a model of transport for the generator encapsulating diffusion. We discuss and motivate in some detail the choices that we make; also to show that the generators that we give are not cast in iron, but may be slanted towards different scenarios depending on the biological model one has in mind. We round off by applying our model of transport to generalize BUD to PINCH.

The action we need to encapsulate is fairly simple: "Should any complex find itself beside a (non-degenerate) channel, it may travel across it." From this specification it is immediate that the generator is homogenous (since well-formed channels go between compartments separated by two membrane-surfaces), and needs to be parametric in the cargo, that is, the complex (or complexes) that it allows to diffuse. Furthermore, in nature diffusion is typically highly regulated, so we need to allow a modeller to specify that cargo-parameter.

Our first attempt is the following rule:



$$(x)\,(A \mid \triangle_x \,||\, \triangle_x \mid \mathbb{T} \mid B) \rightarrow_\omega (x)\,(A \mid \mathbb{T} \mid \triangle_x \,||\, \triangle_x \mid B), \text{ for any closed complexes } \mathbb{T}$$

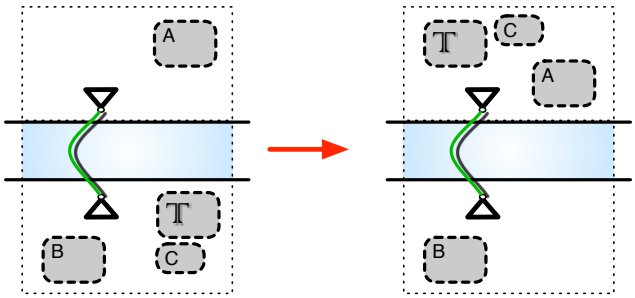
Here $\mathbb{T}$ denotes the cargo-parameter given by the modeller. We certainly preserve well-formedness if we restrict $\mathbb{T}$ to be a closed product of zero or more domains; that is just another way to require that $\mathbb{T}$ contain zero or more fully specified complexes. In other words, formally we require that $\mathbb{T}$ be on the form:

$$\mathbb{T} \equiv (\tilde{y})(\mathsf{d}_{b_1}^{x_1} \mid \cdots \mid \mathsf{d}_{b_k}^{x_k}),$$

where all for i, $x_i \in \tilde{y}$ and $b_i \in \tilde{y}$.

The generator above is computationally complete, in the sense that it allows transport of arbitrary complexes. So we might leave it at that. It is, however, also somewhat impractical to use; there is a huge number of possible configurations of complexes (biologically speaking, *species*) in which a particular protein may be a part. Furthermore, in many cases a central regulation mechanism of diffusion consists of certain proteins or complexes that act as "chaperones"; they essentially allow anything that they bind to to pass. In such situations it would be unfeasible to require a modeller to enumerate all the possible complexes that a chaperone was known to bind to, as well as being a somewhat unfaithful capture of the biological knowledge. In essence, to help practical modelling, we want to be able to give a rule to say that: "any complex(es) C bound to a chaperone (given in $\mathbb{T}$) may diffuse", without having to specify any more than $\mathbb{T}$.

Consequentially, we need to allow $\mathbb{T}$ to be open, to allow it to drag one or more complexes along with it. Our next attempt is then the following rule:



$$\frac{\mathrm{fn}(\mathbb{T} \mid C) = \emptyset}{(x)\,(A \mid \triangle_x \,||\, \triangle_x \mid \mathbb{T} \mid C \mid B) \rightarrow_\omega (x)\,(A \mid \mathbb{T} \mid C \mid \triangle_x \,||\, \triangle_x \mid B)}$$

$\mathbb{T}$ still denotes the user-given parameter, while C contains the complexes that are dragged along with $\mathbb{T}$. Again, we require that $\mathbb{T}$ be on the form

$$\mathbb{T} \equiv (\tilde{y})(\mathsf{d}_{b_1}^{x_1} \mid \cdots \mid \mathsf{d}_{b_k}^{x_k}),$$

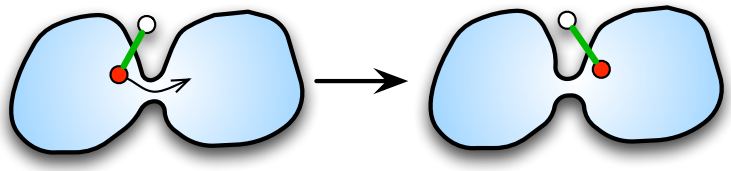
but this time stipulating no conditions on the names (thus allowing $\mathbb{T}$ to be open). Instead, we require that the product of $\mathbb{T}$ and C be closed. We state this formally as a contextual side-condition, as it needs to be checked for the unknown parameter C before a reaction may occur.[10]

This version of the diffusion-rule allows us to give diffusion-rules that only mention chaperones, and not the cargo that they drag along. However, the generator may also allow any solution *not* connected to $\mathbb{T}$ to travel along in C, thus undermining the possibility of modelling highly regulated chaperoning.

The generator above is also too fine-grained to allow a phenomenon known as *receptor-sliding*.When a vesicle is forming on a cell membrane, receptors tied to the

[10] A central contribution of the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework was to work out how such contextual side-conditions may be adopted for bigraphically based calculi (see [DK08]).

membrane wall may on some occasions *slide* from the cytoplasm to the vesicle. The figure below illustrates the pivot-motion that the receptor-protein follows:



Our generator, as stated above, locks receptors to their membranes; to allow receptors to travel, we need to allow an open complex to travel along a channel. In conclusion, our second version is both too coarse, allowing any context to be dragged along, and in a sense also too fine-grained, disallowing receptor-sliding.

Taking a lesson from these failings, let us try to specify more precisely what it is that we would like to drag along with $\mathbb{T}$: We want to capture the *local*, *minimal* context of the parameter $\mathbb{T}$. By local, we mean colocated with domains in $\mathbb{T}$; by minimal, we mean to restrict to domains connected to domains in $\mathbb{T}$. In other words, we want to capture the *local connected component* of domains in $\mathbb{T}$. Let us first try to formulate a non-constructive side-condition to capture this specification:

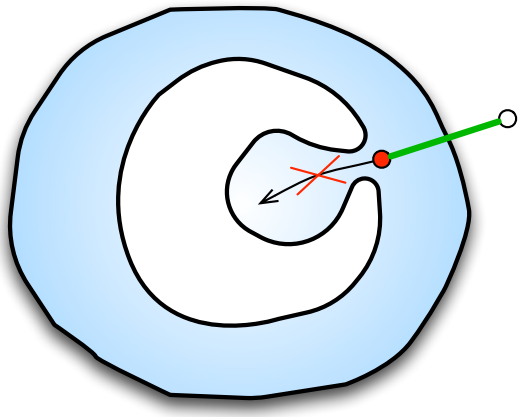
$$\frac{B \equiv B^+ \mid B^- \qquad B^+ \text{ minimal s.t. } \mathrm{fn}(B^-) \cap \mathrm{fn}(\mathbb{T} \mid B^+) = \emptyset \qquad \delta = \downarrow : \mathrm{fn}(\mathbb{T} \mid B^+) = \emptyset}{(x)\,(A \mid \triangle_x \,||\, \triangle_x \mid B^- \mid B^+ \mid \mathbb{T}) \;\rightarrow_\omega\; (x)\,(A \mid \triangle_x \mid \mathbb{T} \mid B^+ \,||\, \triangle_x \mid B^-)},$$

allowing the same form for $\mathbb{T}$ as in our previous attempt.

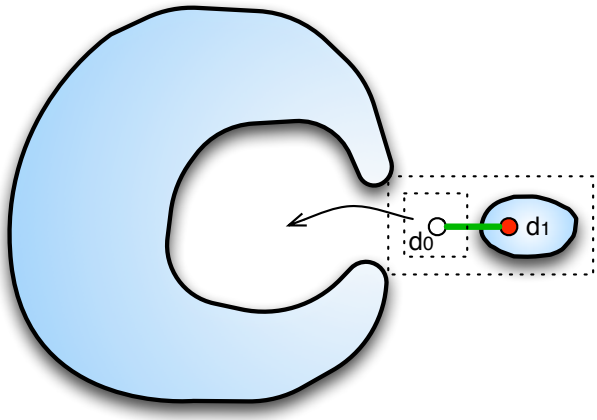
First of all, note that instead of allowing any split of surroundings of $\mathbb{T}$ on the left-hand side (as in the previous rule), we specify in the side-condition that the parameters B^- and B^+ should be a certain split of the surroundings of $\mathbb{T}$. Here, we have simply stated informally, that B^+ must be "minimal", such that there are no links between $\mathbb{T}$ plus its dragged cargo B^+, and B^-, the part of the context left behind.[11]

Note also, that we have kept the side-condition requiring closure of the entire cargo $\mathbb{T} \mid B^+$, for the downwards projection of the generator. This breaks from the intuition that membrane reconfiguration is projective, which we described in Section 7.3.2. Indeed, it turns out that in the $\mathcal{C}$-calculus, because we model cross-membrane links, such as receptor-backbones and channels, the intuition underpinning projectivity does not hold for a certain kind of limit case. And, it is for that reason that we keep the side-condition for the downwards projection. The limit case, which the side-condition prevents, essentially breaks down to a simple geometrical property, and can be illustrated as below:

[11] Note, that non-aliasing matching as introduced and discussed in [DK08] is necessary to infer the lack of any immediate links from the lack of name-sharing expressed by the side-condition.



We wish to transfer the domain d_0 (indicated by the dotted rectangle) down the channel. Trying to compute the minimal surroundings of d_0, we find that it is tied to a membrane in the context that encapsulates also the membranes with the channel. Without breaking links or violating well-formedness we cannot perform any transfer; the problem is precisely that the tied membrane *surrounds* the partially fused membranes. If the tied membrane had had the inverse projection, a transfer would in principle be possible, if we allow membranes to be diffused:



In this case, we may compute a closed cargo to be transfer along with d_0 (indicated by the outer dotted rectangle).

The conclusion is, that membrane curvature *does* matter when our transfer is directed downwards. We cannot drag along enclosing membranes; thus the side-condition restricts to closed cargoes for the downwards projection.

The fact remains that the side-condition is still non-constructive and informally stated, however. More critically, our analysis also highlights another flaw: the minimal context of $\mathbb{T}$ may contain a membrane, as illustrated above.

In other words, as the generator is stated above, we also admit the dragging of membranes in B^+ bound to receptors in $\mathbb{T}$. Though this preserves well-formedness as we have defined it in Definition 7.2.7, it is hardly biologically feasible. Though

on some occasions tiny vesicles may travel along certain special channels (such as in tunneling nano-tubes [RSM+04]), our channels are intended mainly as an abstraction allowing transport of protein complexes, not membranes. The difference in scale between proteins and membranes is such that, broadly speaking, the "pulling power" of forces affecting proteins cannot drag along a membrane. In a situation such as the one illustrated above, the most likely scenario would probably be that the tied membrane prevented any transfer from happening at all; while for certain applications, a more likely scenario might be that any tied complexation-links be broken. In this paper, we shall make the first choice, as it equips diffusion with a transactional guarantee.

Consequentially, in giving a concrete definition of "minimality", we would like to capture also that membranes are too large to travel in channels. However, as the discussion above illustrates, it is probably beneficial not to hardwire exactly the collection of the cargo. Hence, we choose to isolate and implement the final version of the side-condition as a small function that computes the surroundings. For certain applications this function may then be tweaked. We may, for instance, specify in this function that under certain conditions the transport is not possible; thus allowing us to prevent membrane-dragging.

We cannot get around, however, that computing the connected component of a node in a graph requires iteration. In our case, we want to find a particular subset of the connected component of the user-parameter $\mathbb{T}$ (following backbone-links and complexation-links); namely the part of it that is colocated with $\mathbb{T}$. Formally, we can define the function that computes the local connected component by taking the least fixed point of a function that computes the immediate (local) neighbours; as usual, we may then compute this least fixed point by iterating the latter function.

We start by defining collect(T, S), the function that collects those top-level domains in S (i.e., not nested inside membranes), which are connected with domains in T.

Definition 7.3.6 (collecting immediate local neighbours)**.** The function collect(T, S) is defined for any solution T and any open solution

$$\mathsf{S} \equiv (\mathsf{S}_0 | \ldots | \mathsf{S}_{n-1}),$$

where each S_i is either a domain, a gate or a membrane (empty or with another solution inside).

The function is defined as follows (iterating over S from right to left):

$$\begin{array}{rcl}
\mathrm{collect}(\mathsf{T}, 0) & = & (\mathsf{T}, 0) \\
\mathrm{collect}(\mathsf{T}, \mathsf{S} \mid \mathsf{d}_b^x) & = & \mathrm{collect}(\mathsf{T}, \mathsf{S}) \mid (0, \mathsf{d}_b^x), \text{ if } \{x, b\} \cap \mathrm{fn}(\mathsf{T}) = \emptyset \\
\mathrm{collect}(\mathsf{T}, \mathsf{S} \mid \mathsf{d}_b) & = & \mathrm{collect}(\mathsf{T}, \mathsf{S}) \mid (0, \mathsf{d}_b), \text{ if } \{b\} \notin \mathrm{fn}(\mathsf{T}) \\
\mathrm{collect}(\mathsf{T}, \mathsf{S} \mid \overline{\mathsf{d}_b}) & = & \mathrm{collect}(\mathsf{T}, \mathsf{S}) \mid (0, \overline{\mathsf{d}_b}), \text{ if } \{b\} \notin \mathrm{fn}(\mathsf{T}) \\
\mathrm{collect}(\mathsf{T}, \mathsf{S} \mid \triangle_g) & = & \mathrm{collect}(\mathsf{T}, \mathsf{S}) \mid (0, \triangle_g) \\
\mathrm{collect}(\mathsf{T}, \mathsf{S} \mid \mathsf{m}[\mathsf{S}']) & = & \mathrm{collect}(\mathsf{T}, \mathsf{S}) \mid (0, \mathsf{m}[\mathsf{S}']),
\end{array}$$

where $\mid$ is lifted pointwise to tuples of solutions.

Formally, the normal form for solutions ensures us that collect(T,S) is defined for all open solutions S (cf. Definition 7.A.4 in Appendix 7.A). The function collect(T,S) returns another tuple of solutions $(\mathsf{T}',\mathsf{S}')$, where those domains in S connected to T (i.e., sharing names) have been removed from S and added to T, to form T'. This can be summed up in the following little lemma.

Lemma 7.3.7. *If* collect$(\mathsf{T},\mathsf{S}) = (\mathsf{T}',\mathsf{S}')$ *then there exists some solution* S''*, s.t.* $\mathsf{T}' \equiv \mathsf{T} \mid \mathsf{S}''$ *and* $\mathsf{S} \equiv \mathsf{S}' \mid \mathsf{S}''$*.*

In other words, we always have collect$(\mathsf{T},\mathsf{S} \mid \mathsf{S}') \equiv (\mathsf{T} \mid \mathsf{S},\mathsf{S}')$, for some S. It is also easy to see that collect(T,S) has least fixed points; either collect(T,S) removes part of S, making further progress towards the base case for collect$(\mathsf{T},0)$; or nothing is removed from S, and we are done.

Lemma 7.3.8. *The function* collect *has least fixed points.*

The least fixed point can be computed by iterating collect*; call the least fixed point* collect**.*

And finally, we can define lcc, the function that computes the *draggable* local connected component. We define lcc(T,S) as a partial function, that uses collect* to compute the local connected component $(\mathsf{T}^*,\mathsf{S}^*)$, and then tests whether the resulting solutions are still connected, that is, still share names. If so, this is because domains are tied across a membrane, and we let lcc be undefined in this case.

Definition 7.3.9 (draggable local connected component)**.** For

$$\mathsf{T} \equiv (\tilde{x})\,\mathsf{T}',$$

where T' is a parallel product of domains, and

$$\mathsf{S} \equiv (\tilde{y})\,\mathsf{S}',$$

where S' is any open solution, s.t. $\tilde{x} \cap \tilde{y} = \tilde{x} \cap \mathrm{fn}(\mathsf{S}) = \tilde{y} \cap \mathrm{fn}(\mathsf{T}) = \emptyset$,

$$\mathrm{lcc}(\mathsf{T},\mathsf{S}) = \begin{cases} ((\tilde{y})\,\mathsf{T}^*, (\tilde{y})\,\mathsf{S}^*) & \text{if collect}^*(\mathsf{T},\mathsf{S}') = (\mathsf{T}^*,\mathsf{S}^*) \text{ and } \mathrm{fn}(T^*) \cap \mathrm{fn}(S^*) = \emptyset \\ \bot & \text{else} \end{cases}$$

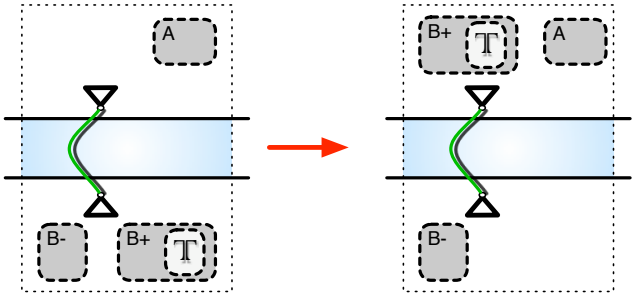
The restrictions on the bound and free names of T and S are purely technical, as we can always α-convert bound names to avoid any clashes.

We state in the following proposition the properties of the solutions computed by lcc. The proposition is straight-forwardly verified from the definitions and lemmas above.

Proposition 7.3.10. *Either* lcc$(\mathsf{T}, B^+ \mid B^-)$ *is undefined or* lcc$(\mathsf{T}, B^+ \mid B^-) \equiv (\mathsf{T} \mid B^+, B^-)$*, and* B^+ *is solution that contains only domains, such that every domain in* B^+ *can trace a path of (backbone or complexation) links to a domain in* T*, and such that no domain in* B^- *can trace such a path.*

In other words, the result of invoking lcc on the user-parameter and its context, $\mathrm{lcc}(\mathbb{T}, B^+|B^-)$ is either undefined, indicating that a tied membrane prevents transfer, or a result $(\mathbb{T} \mid B^+, B^-)$, such that B^+ is a parallel product of domains; and, all domains in B^+ are in the connected component of $\mathbb{T}$, while none of the domains in B^- are.

Having captured and isolated the computation of the solution that $\mathbb{T}$ drags along in lcc, we may now state the final generator for DIFF. Figure 7.17 contains the definition and an illustration that slightly overloads our graphical language to illustrate the computation of the draggable local connected component. We take Proposition 7.3.10 as verifying that we have implemented faithfully the last non-constructive specification of the generator.



DIFF($\mathbb{T}$):

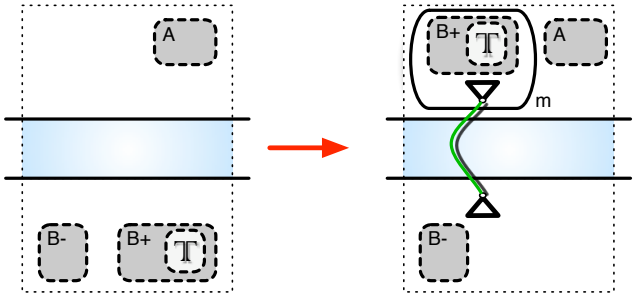
$$\frac{\mathrm{lcc}(\mathbb{T}, B^+ \mid B^-) \equiv (\mathbb{T} \mid B^+, B^-) \qquad \delta = \downarrow\colon \mathrm{fn}(\mathbb{T} \mid B^+) = \emptyset}{(x)\,(A \mid \triangle_x \,||\, \triangle_x \mid B^- \mid B^+ \mid \mathbb{T}) \;\rightarrow_\omega\; (x)\,(A \mid \triangle_x \mid \mathbb{T} \mid B^+ \,||\, \triangle_x \mid B^-)}$$

Figure 7.17: The homogenous generator DIFF.

We may similarly enhance the pinch-rule to allow also the transfer of an initial parameter, as discussed in the previous section. The resulting generator is depicted and defined in Figure 7.18.

As a final remark in this section, a note on expressivity: Not unlike the encoding of κ into micro-κ [DL04], we may implement the lcc-function using vanilla reaction rules encoding the traversal performed in lcc. It essentially requires the addition of a single control for temporarily gathering the connected connect; and a number of extra rules for recognizing preconditions corresponding to the left-hand sides of the lcc-function. We find, however, that the higher abstraction-level gained by taking the lcc-function is sufficiently important to justify our choice. Also, as the function is inductively given, it lends itself directly to implementation.

Table 7.1 summarizes all the generators introduced in this section for membrane reconfiguration.



$$\textsc{pinch}(\mathbb{T}):\ \frac{\mathrm{lcc}(\mathbb{T}, B^+ \mid B^-) \equiv (\mathbb{T} \mid B^+, B^-) \qquad \delta = \downarrow : \mathrm{fn}(\mathbb{T} \mid B^+) = \emptyset}{A \mid\mid B^- \mid B^+ \mid \mathbb{T} \ \rightarrow_\eta \ (x)(A \mid \mathsf{m}[\mathbb{T} \mid B^+ \mid G_x] \mid\mid B^- \mid \triangle_x)}$$

Figure 7.18: The heterogenous generator for parametric PINCH.

7.3.5 Refinement and Allowable Rules

In the previous sections we have introduced a series of generators all encapsulating a core biological action. And in Section 7.3.1, and Figures 7.9 and 7.14, we have introduced refinement formally. We now turning to defining refinement, formally.

The idea is, that we want to allow in $\mathcal{C}$-calculus any rule that is a suitable refinement of (a projection of) one of the generators for membrane actions or domain-level rules given in the previous section. All (projected) generators are certainly valid rules by themselves. However, we want to allow a biochemist to specify that, for instance, a vesicle may only TOUCH the plasma membrane, if a protein pro attached to its surface is bound to a receptor rec on the plasma membrane. Let is work out this example in detail.

From the informal specification above, we want a rule that might look like this:

$$\begin{array}{rcl} r_0 & = & \mathsf{pro0}_a^x \mid \mathsf{ves}[\ \ \mathsf{pro1}_a \mid A] \mid \mathsf{rec0}_b^x \mid \mathsf{plasma}[\ \ \mathsf{rec1}_b \mid B] \rightarrow \\ & & (y)\, \mathsf{pro0}_a^x \mid \mathsf{ves}[\triangle_y \mid \ \mathsf{pro1}_a \mid A] \mid \mathsf{rec0}_b^x \mid \mathsf{plasma}[\triangle_y \mid \mathsf{rec1}_b \mid B], \end{array}$$

where, for effect, we have aligned counterparts on the left- and right-hand sides. Let is spell out precisely in what sense r_0 is a refinement of the lateral projection of TOUCH.

First, the lateral projection of TOUCH, selecting the appropriate kinds of membranes, yields the following rule:

$$\mathsf{ves}[A] \mid \mathsf{plasma}[B] \rightarrow (y)\, \mathsf{ves}[\triangle_y \mid A] \mid \mathsf{plasma}[\triangle_y \mid B].$$

Next, in refining a generated rule, we want to allow refinement of the parameters; in practice, by substituting for any variable X, a solution S (that might in turn contain variables). We call this *inner* refinement of X. Refining the parameters A and B in the lateral projection of TOUCH, we get:

$$\mathsf{ves}[\mathsf{pro1}_a \mid A] \mid \mathsf{plasma}[\mathsf{rec1}_b \mid B] \rightarrow (y)\, \mathsf{ves}[\triangle_y \mid \mathsf{pro1}_a \mid A] \mid \mathsf{plasma}[\triangle_y \mid \mathsf{rec1}_b \mid B]$$

Homogenous generators

$$\textsc{part} \qquad (x)(\triangle_x \mid A \parallel \triangle_x \mid B) \;\to_\omega\; A \parallel B$$

$$\textsc{touch} \qquad A \parallel B \;\to_\omega\; (x)(\triangle_x \mid A \parallel \triangle_x \mid B)$$

$$\textsc{diff}(\mathbb{T}) \qquad \frac{\mathrm{lcc}(\mathbb{T}, B^+ \mid B^-) \equiv (\mathbb{T} \mid B^+, B^-) \qquad \delta = \downarrow\colon \mathrm{fn}(\mathbb{T} \mid B^+) = \emptyset}{(x)\,(A \mid \triangle_x \parallel \triangle_x \mid B^+ \mid B^- \mid \mathbb{T}) \;\to_\omega\; (x)\,(A \mid \triangle_x \mid \mathbb{T} \mid B^+ \parallel \triangle_x \mid B^-)}$$

Heterogenous generators

$$\textsc{merge} \qquad (x)(A \mid \mathsf{m}[\triangle_x \mid C] \parallel \triangle_x \mid B) \;\to_\eta\; A \parallel C \mid B$$

$$\textsc{pinch}(\mathbb{T}) \qquad \frac{\mathrm{lcc}(\mathbb{T}, B^+ \mid B^-) \equiv (\mathbb{T} \mid B^+, B^-) \qquad \delta = \downarrow\colon \mathrm{fn}(\mathbb{T} \mid B^+) = \emptyset}{A \parallel B^+ \mid B^- \mid \mathbb{T} \;\to_\eta\; (x)(A \mid \mathsf{m}[\mathbb{T} \mid B^+ \mid G_x] \parallel B^- \mid \triangle_x)}$$

for any $\mathbb{T} \equiv (\tilde{y})(\mathsf{d}_{b_1}^{x_1} \mid \cdots \mid \mathsf{d}_{b_k}^{x_k})$ and for lcc as given in Def. 7.3.9.

Table 7.1: Summary of the generators for membrane reconfiguration.

And finally, we want to allow *outer* refinement of a reaction rule by specifying parts of surrounding context, where the reaction occurs. In practice, for a rule $\mathsf{L} \to \mathsf{R}$, we want to allow any rule $\mathsf{L} \mid \mathsf{S} \to \mathsf{R} \mid \mathsf{S}$. By contextualizing the rule above with $\mathsf{pro0}_a^x \mid \mathsf{rec0}_b^x$, we get r_0.

We may sum up refinement conveniently with the help of substitution. First however, for full generality, we also need to consider rules with side-conditions.

Consider the following (somewhat artificial) rule and side-condition:[12]

$$r_1 = A \to \mathsf{p} \mid A,\ \varphi = \text{"}A \text{ does not contain a } \mathsf{p}.\text{"}$$

If we perform inner refinement on A to get

$$r_1' = \mathsf{p} \mid A \to \mathsf{p} \mid \mathsf{p} \mid A,\ \varphi' = ?,$$

what should the side-condition φ' say? We elect that A should be refined in the side-condition, as well. This yields: $\varphi' =$ "$(\mathsf{p} \mid A)$ does not contain a p." This in turn yields a valid rule (that, however, never applies, since the side-condition is equal to falsity). In other words, when substituting a parameter X for T, we also need to substitute that parameter in φ. We finally settle on the following definition for refinement:

Definition 7.3.11 (refinement)**.** For any rule $(\mathsf{L} \to \mathsf{R}, \varphi)$, any solution context S with one hole, and any group of solutions G, s.t., the number of solutions in G is equal to the number of variables in L and R, a refinement of this rule is $(\mathsf{S} \triangleleft \mathsf{L} \triangleleft \mathsf{G} \to$

[12] In writing side-conditions for reaction rules, it is convenient to overload the usage of variable-names in the rule, such as A, to refer to the solution that instantiates the variable.

$\mathsf{S} \lhd \mathsf{R} \lhd \mathsf{G}, \varphi')$. The side-condition φ' is true for a group of parameters F if $\varphi(\mathsf{G} \lhd \mathsf{F})$ holds.

We call the contexts S and G the outer and inner refinement, respectively. Together, we call them the *application conditions* used to refine the rule $\mathsf{L} \to \mathsf{R}$.

In our case, for the generators in Table 7.1, the only parameters mentioned in side-conditions are the B-parameters in PINCH and DIFF. In practice, for those rules, to ensure that refined rules not be unappliable, do not refine B^- to include a domain that is connected to $\mathbb{T}$, and do not refine B^+ to include a domain that is *not* connected to $\mathbb{T}$.

Our choice to refine also side-conditions ensures us that application conditions only restrict the applicability of rules. In particular, we easily verify the following convenient little lemma.

Lemma 7.3.12 (application conditions restrict rules)**.** *Suppose* $r' = (\mathsf{L}' \to \mathsf{R}', \varphi')$ *is a refinement of* $r = (\mathsf{L}, \mathsf{R}, \varphi)$ *as defined in Definition 7.3.11. Then, if* $\mathsf{T} \to \mathsf{T}'$ *by* r'*, then also* $\mathsf{T} \to \mathsf{T}'$ *by* r*.*

Suppose, we had *not* refined side-conditions also; then this lemma would not hold. (To see this, consider the reaction $\mathsf{p} \to \mathsf{p} \mid \mathsf{p}$ against the example rule r_1 and against the refined rule r_1' *without* refining also the application condition.)

Having treated the technicalities concerning refinement, we can sum up the allowable rules.

Definition 7.3.13 (allowable rules)**.** The allowable rules in the $\mathcal{C}$-calculus are refinements of

- domain-level rules, as defined in Definition 7.3.5; and,
- projections of the generators in Table 7.1.

In Section 7.4, we give several concrete examples of rules.

7.3.6 Preservation of Well-formedness

In this section, we establish the promised property that any reaction due to an allowable rule preserves well-formedness.

The main theorem is the following. We devote the rest of this section to prove this theorem.

Theorem 7.3.14 ($\mathcal{C}$-calculus reactive systems preserve well-formedness)**.** *If* S *is well-formed, and* $\mathsf{S} \to \mathsf{T}$*, then* T *is well-formed.*

Proof. Follows from Lemmas 7.3.12, 7.3.15, 7.3.16, and 7.3.17. □

All well-formedness conditions stipulate conditions on links—on the presence, arity, or sorting of links or on the locality of the ports they connect. That fact, and the nature of allowable $\mathcal{C}$-calculus rules, makes graph-based reasoning on solutions most

convenient. As explained in Section 7.2.4, solutions correspond to certain bigraphs. Further, as recorded in [DK08, Theorem 4.5], reaction for any $\mathcal{B}^{\Sigma,\mathcal{R}}$-calculus corresponds 1-1 to bigraphical reaction (under so-called non-aliasing contexts). Hence, due to the founding of $\mathcal{C}$-calculus on bigraphs, we can directly apply (bi)graphical reasoning to verify the following propositions.

We need to check the change(s) induced by a reaction rule against each of the well-formedness conditions.

In outline, we proceed as follows: Most allowable rules, specifically domain-level rules and those generated from TOUCH, PART, and BUD create or break links (plus some domains or membranes) contained entirely inside the left-hand side or right-hand side of a rule. Consequentially, it is easily checked that they cannot break well-formedness. We start by stating this in Lemmas 7.3.15 and 7.3.16. Those rules that allow bunched transportation of subsolutions, contained in parameters of the rules, need a bit more care. We check that rules generated from MERGE are innocuous; and, for the rules generated from DIFF and PINCH we check that our development in Section 7.3.4 has been sound. We handle these three transport rules in Lemma 7.3.17. Together the three lemmas imply that all reactions due to allowable rules preserve well-formedness, as stated in Theorem 7.3.14.

A simple, but convenient corollary of Lemma 7.3.12 is that any reactive system $\mathcal{T}'$ over refined rules will be a sub-system of a reactive system $\mathcal{T}$ over the corresponding unrefined rules. It follows that, if we prove preservation of well-formedness for reactive systems over core rules without refinements, i.e., domain-level rules and the projections of the generators in Table 7.1 without any application conditions, then, in particular, any reactive systems using refined rules also preserves well-formedness. Hence in the following lemmas, we shall simply check unrefined rules.

Lemma 7.3.15 (domain-level rules preserve well-formedness)**.** *If* S *is well-formed and* $\mathsf{S} \to \mathsf{S}'$ *by a domain-level rule, then* S' *is well-formed.*

Proof. Below we sketch the proof in some detail. The proof is essentially a straightforward case-analysis of a number of cases corresponding to the kinds of links in the bigraphs that correspond to well-formed solutions. We thus rely on bigraphical reasoning. In this paper, we have not introduced bigraphical theory in full; however, we refer the reader to the informal summary given in Section 7.2.4.

From Definition 7.3.5 we know that all domain-level rules $\mathsf{L} \to \mathsf{R}$ incorporate one change (for the base rules) or two changes. Those changes translate directly to changes which relate the (bi)graphs S to S'.

Let us consider first, what occurs under SYNTH and DEGRADE.

- SYNTH All domains affected by this reaction have just been created in S and are unbound. The only created link is the protein backbone, and it is explicitly required to respect the well-formedness condition (*fixed backbone*), together with the created domains it links.

- DEGRADE No complexation-links are created or deleted by this reaction. A backbone link is deleted, and it is explicitly required that every domain in a protein is deleted together with it (ensuring us that (*fixed backbone*) is upheld).

Now, we analyze each of the base rules, which change existing or newly created domains or links, for their effect on a domain in S and consider well-formedness.

- BIND Suppose d in S is affected by this reaction. Then it is visible and will be bound to a co-located d' in S'. In effect, S' will have an added link compared to S. We check this link against the well-formedness conditions; the three interesting conditions are (*binary complex*), (*link sorting*), and (*local complex*). They are all upheld, as it is explicitly required that the link be between two complexation-ports of co-located domains. The remaining conditions are voidly upheld.

- BREAK Suppose d in S is affected by this reaction. Then it is bound, and will be visible in S' (as well as another d' in S). In effect, S' will have lost a link compared to S. Checking against the well-formedness conditions, it easily seen that no conditions are violated. In particular, note that since we explicitly require the entire link to be matched and deleted in BREAK, no faulty unary links can be created (which would violate (*binary complex*)).

- HIDE/SHOW These reactions change no links, making them non-problematic with regard to well-formedness.

Finally, from the check of HIDE/SHOW, it easily seen that the combinations of BIND and BREAK with one of HIDE and SHOW (i.e., BIND+SHOW, BIND+HIDE, BREAK+SHOW, and, BREAK+HIDE) also preserve well-formedness.

In conclusion, no domain-level reaction rule may induce a change in S, which breaks well-formedness in S'. □

We turn to checking the generators described in Section 7.3.4 and the changes they induce. We start by considering projections of the rules that do not involve transport.

Lemma 7.3.16 (non-transport generators preserve well-formedness)**.** *If* S *is well-formed and* $\mathsf{S} \to \mathsf{S}'$ *by a rule generated from* TOUCH*,* PART*, or* BUD*, then* S' *is well-formed.*

Proof. We sketch the proof.

Any reaction rule generated from TOUCH and PART induces exactly the following changes on S: The addition or deletion of two gates as well as the creation or deletion of a link between them. By definition, the homogenous projections will ensure us that these changes occur in compartments separated by exactly two membranes, in turn ensuring us that any such channel respects (*bitonality*). It is immediate from the definition of the generators that the conditions (*link sorting*) and (*binary channels*) are upheld.

Any reaction rule generated from BUD adds exactly one membrane as well as two connected gates (as for TOUCH above). The heterogenous projection in combination with the created membrane ensures that the gates respect (*bitonality*). (Again, it is immediate that the conditions (*link sorting*) and (*binary channels*) are upheld.) □

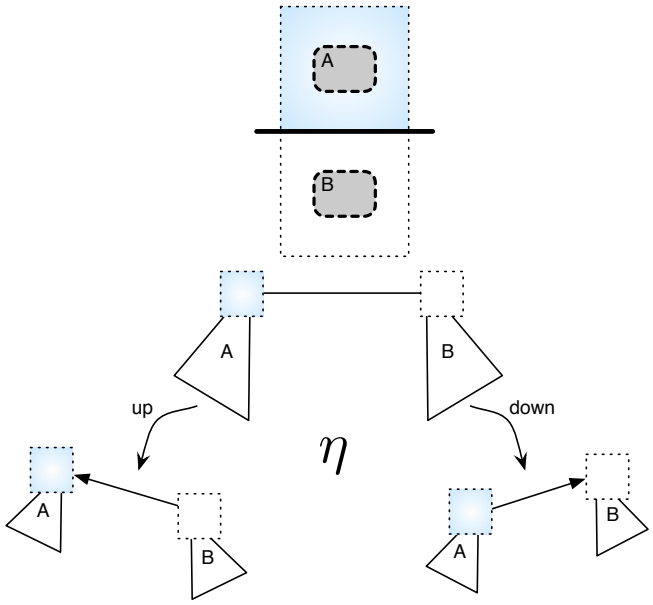


Figure 7.19: Depicting heterogeneous generators projectively, using unoriented parenthood-edges.

It remains for us to consider those rules, which move subtrees—captured in parameters in the rules—in the solution. To illustrate the reasoning in the following proof, we shall sketch the parts of the tree corresponding to the solution, where the reaction occurs.

We are reasoning about rules, which stem from generators projected to form two or three rules. To concisely illustrate and reason about all projections in one go, we may illustrate the subtrees corresponding to generators in the manner depicted for heterogenous projection in Figure 7.19 and for homogenous projection in Figure 7.20. Below, we refer to such illustrations as *projective*. At the top of the figures, projective groups are illustrate as in Section 7.3.2. When we consider solutions as trees whose parenthood-relation is given by membrane-containment, this corresponds to stating that one or two parenthood relations (for heterogenous and homogenous, respectively) are unspecified. We can conveniently illustrate that by leaving one or two parenthood-edges in the tree corresponding to a solution be unoriented. We depict that in the middle of Figures 7.19 and 7.20. At the bottom, we illustrate how the projections work to produce the two or three trees corresponding to the projective illustrations.

Lemma 7.3.17 (transport generators preserve well-formedness)**.** *If* S *is well-formed and* $\mathsf{S} \to \mathsf{S}'$ *by a rule generated from* MERGE, DIFF, *or* PINCH, *then* S' *is well-formed.*

Proof. Below we sketch and illustrate the proof in some detail. As for the proofs above, the proof is a case-analysis of a number of cases for the kinds of links in bi-

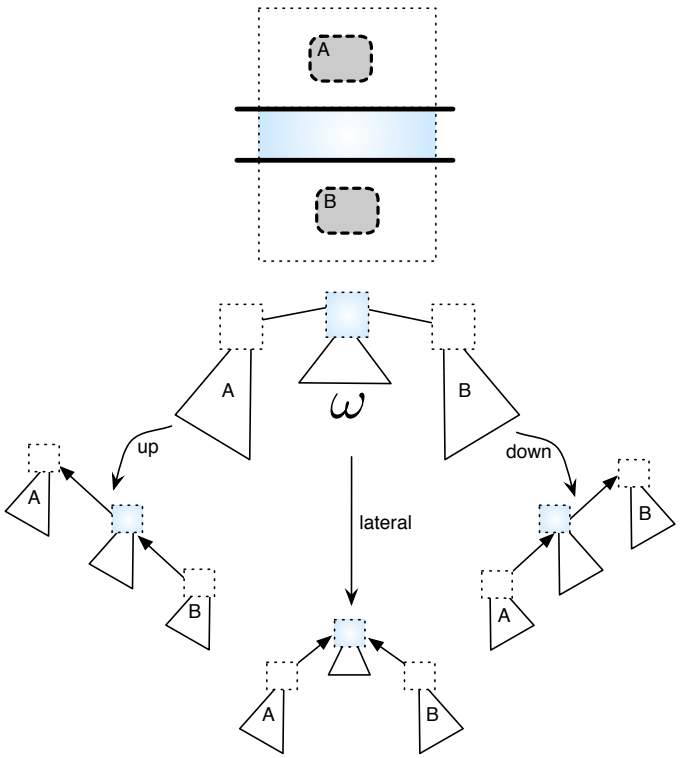


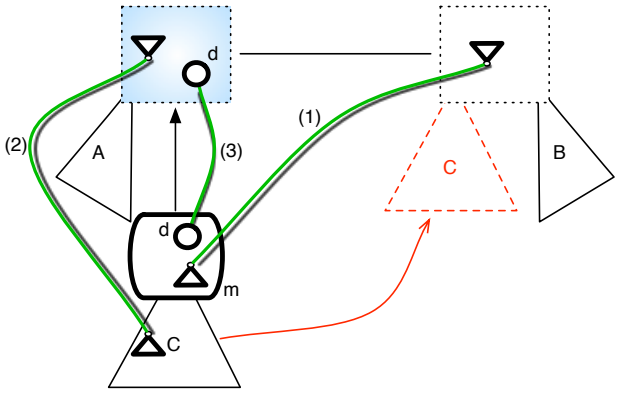
Figure 7.20: Depicting homogenous generators projectively, using unoriented parenthood-edges.

graphs corresponding to well-formed solution. We thus rely on bigraphical reasoning; we refer the reader to the intuition given in Section 7.2.4.

We consider the generators MERGE and PINCH, and discuss the links that we need to check with the help of projective illustrations as explained above. We elide a detailed sketch of the analysis for the case for DIFF, as the verification is similar to that for PINCH.

MERGE: A membrane and two connected channels are deleted; and the contents of the membrane (captured in the parameter C) is transfered to the other end of the channel. The deletion of the connected channels correspond to the action captured in PART, and we need not consider them further.

The projective illustration below illustrates the generator, and highlights representatives of links, which we need to consider.



On the bottom-left is the subtree corresponding to C. The arrow pointing to the right and the dotted subtree on the right indicates the transport that C will undergo as a consequence of MERGE. C is inside a membrane (to illustrate, we indicate this with a membrane M at the top of subtree illustrating C), which is inside a compartment, which contains another subtree (A). This compartment either has a parent or child (depending on the projection) containing the subtree B.

Those links that may prove problematic are those spanning regions, and it is clear that we need only consider those connected to nodes inside C. On the illustration, we have drawn representatives of each kind of link, that we need to consider.

Channels, whose gates lie within C, may come in two versions:

- Well-formed channels that are top-level in C may be connected to the region that C is transported into. Such an example is illustrated and tagged with (1) above. As a consequence of the transport such channels will be degenerate (i.e., their length will be 0), but well-formed in S'.

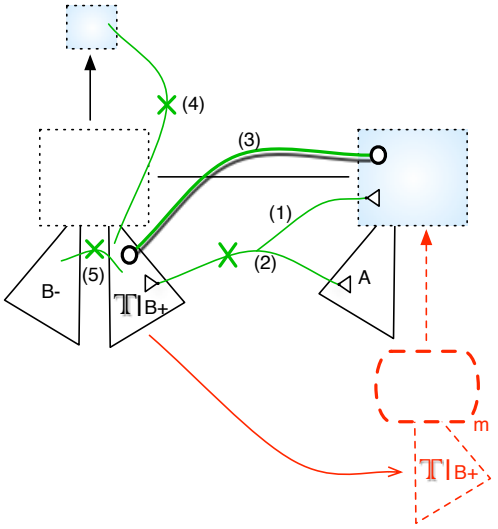
- Well-formed channels that are inside a top-level membrane in C may only be connected to the parent of C. We have illustrated and tagged such a link with (2) above. As can be seen from the illustration, after the transport of C, the channel will still span two membranes.

Every other well-formed channel is either contained entirely within or outside C.

The link tagged with (3) illustrates a protein backbone that crosses the membrane that is destroyed as a consequence of MERGE. Checking, we see that this backbone also preserves its well-formed after the transfer.

PINCH: A membrane and two connected channels are created, and the cargo $\mathbb{T}|B^+$ is transfered to the compartment inside the new membrane. It is simple to check that the creation of the membrane itself and the connected channels respect well-formedness. We concern ourselves with the links, whose endpoints change location.

We illustrate the generator projectively, and again we highlight representatives of links that we need to consider more carefully.



On the middle-left, we illustrate with a dotted region the compartment containing the user-parameter $\mathbb{T}$ and the parameter, B^+, it is dragging along. The parameter that remains, B^-, is a sibling. The compartment, that the new membrane is created in, is a parent or a child (depending on the projection) of the compartment that $\mathbb{T}|B^+$ stems from. We illustrate the transfer with an arrow, and on the bottom-right, we draw the newly created membrane and the transferred $\mathbb{T} \mid B^+$ with dotted lines.

The parent-compartment of the new membrane (on the top-right) has a subtree A. Finally, in case the projection is downwards (i.e., in case the undirected parent-edge is oriented from right to left), then we need to consider also links to the parent-region of the compartment containing $\mathbb{T}$, B^+, and B^- (illustrated on the top-left).

As for MERGE, we need concern ourselves mainly with links spanning compartments. However, we should remark that this is only because we have already verified (in Proposition 7.3.10) that the split of B into B^+ and B^- by lcc, ensures that there are no links, channels or backbones, between $\mathbb{T} \mid B^+$ and B^- (we illustrate such a link above with a crossed link and tag it with (5)).

Also, the side-condition $\mathrm{fn}(\mathbb{T} \mid B^+) = \emptyset$ given for the downwards projection, ensures that there can be no links to a parent region (we illustrate such a link with

a crossed link tagged with (4)).

Having treated the links that the side-conditions prevent, we turn to channels captured partially in $\mathbb{T} \mid B^+$. However, lcc ensures ensures that *no* gates may be captured in $\mathbb{T} | B^+$. Hence, the two possibilities for channels (illustrated with crossed lines and tagged with (1) and (2) above), are prevented. It remains again to consider backbones that span the projected membrane, such as the link tagged (3) above. After the PINCH they will instead span the newly created membrane—having followed the pivot-motion illustrated and discussed in Section 7.3.4—and will still be well-formed.

DIFF: The analysis is similar to that for PINCH. □

7.4 Examples

Now for some examples.

7.4.1 G-protein Coupled Receptor

We start by developing a model of cross-membrane signal transduction (i.e., propagation of signals across say a cell wall in a signalling pathway) via *G-protein coupled receptor proteins* (GCPRs). A GCPR is a kind of receptor belonging to a large family of certain transmembrane proteins—proteins integral to the cell wall that have protrusions on both sides. The family of GCPRs is large and diverse, but they all employ a similar technique to implement a mechanism that allows extracellular signals to cross the plasma membrane. In the following description and model of GCPRs we shall abstract away from many details to develop a model illustrating some of the central properties shared by GCPRs; in any case, the answers to many questions pertaining to GCPRs are not known.

The extracellular part of a GCPR, illustrated in Figure 7.21, constitutes a receptor site that is shaped to be able to bind certain *ligands* (signalling molecules). Its inner part is bound to a partner G-protein (a guanine nucleotide-binding protein). The G-protein in its inactive state is actually a little complex of three different protein subunits bound to each other: G_α, G_β, and G_γ.[13] The β and γ subunits are bound tightly together (together they are called the $G_{\beta\gamma}$-complex), while the α complex is relased as part of the activation of the GCPR. Particular to GCPRs are that they do not pass any substance through the membrane, but rely on structural changes to the folding shape of the receptor. The folding shape of a protein is called its *conformation*, hence such structural changes are known as conformational changes.

Loosely, the GCPR propagates a signal in the following manner: A ligand is bound to the receptor site activating the GCPR by changing the conformation of the receptor in such a way that the G-protein is activated, causing the G_α subunit to bind to a molecule of GTP (guanosine triphosphate). This in turn causes it to release

[13]For simplicity, we restrict to treating here heterotrimeric G proteins, also called "large" G-proteins.

itself from the rest of the G-protein, as illustrated in Figure 7.22. The remaining $G_{\beta\gamma}$-complex also detaches from the receptor. The release of the G_α subunit has exposed sites on the $G_{\beta\gamma}$-complex that may interact with other molecules that serve as effectors for further propagation of the signal. A receptor in an activated state may bind to other G-proteins and activate them. Depending on the stability of the ligand-receptor complex, the ligand is released at some point. At this point an inactive G-protein may again bind to it to return the entire complex to the initial inactive state.

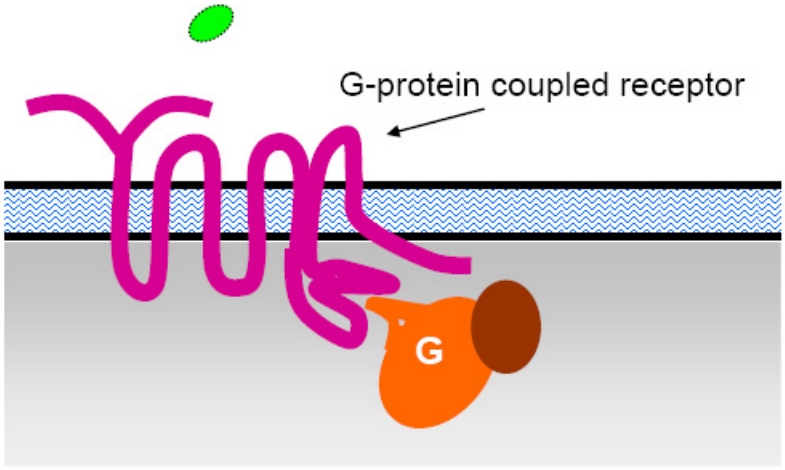


Figure 7.21: A G-protein receptor before binding to a ligand (*illustration from* [Coh07]).

A $\mathcal{C}$-calculus model We work with the following protein signature:

$$\{\mathsf{lig} : (1,0), \mathsf{re} : (2,1), \mathsf{gbc} : (3,0), \mathsf{ga} : (2,0), \mathsf{gtp} : (1,0), \mathsf{eff} : (1,0)\},$$

and a single membrane type plasma to model plasma membranes.

In Figure 7.23, we give seven rules that encapsulates a model of the core reactions involved in the firing of a G-protein coupled receptor. We explain the rules one-by-one, below.

- Rule 0: This rule allows a ligand lig to bind to the extra-cellular domain of the receptor, re0.
- Rule 1: The rule models the conformational change of the receptor induced by the ligand binding in rule 0, by activating the G_α subunit, specifically the domain ga1, able to bind to the GTP molecule, gtp. We make rule 1 η-projective, i.e., we intend that it may be applied if the two solutions in the groups in the left and right hand side are separated by exactly one membrane. In this manner, we capture conveniently and consisely the fact that these

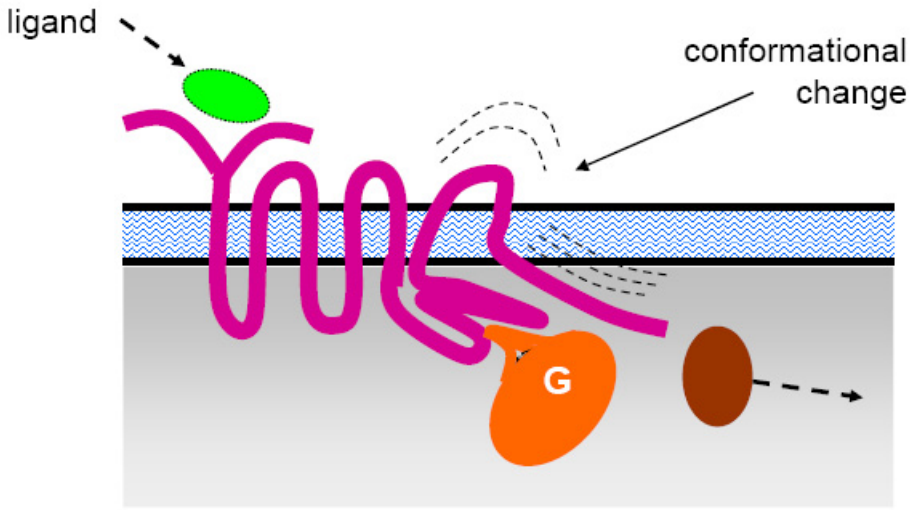


Figure 7.22: A G-protein receptor firing due to a conformational change, after binding to a ligand (*illustration from* [Coh07]).

transmembrane receptors transfer signals across the membrane, independent of the orientation of the membranes.

We should note, that though this serves as a nice illustration of projectivity, in nature the particular family of transmembrane receptors we have chosen, GCPRs, work only in the plasma membrane, working to propagate extracellular signals inwards.Note also, that we disassociate the ligand-binding event with the conformational change.

- Rule 2: In this rule the domain ga1 of the G_α subunit binds to the gtp.
- Rule 3: This rule allows the G_α subunit to break the assocation to the G-protein leaving the $G_{\beta\gamma}$-complex bound to the receptor. The dissassociation reveals (and activates) a part of the $G_{\beta\gamma}$-complex hidden by the G_α subunit. We model that by the activation of the domain gbc1. Note that rules 2 and 3 are not causally dependent. The ga1 may bind to gtp before or after the ga0 domain has broken the bond to gbc2 domain. In other words, we model that the G_α subunit may bind to the GTP molecule independently of the dissassocation from the $G_{\beta\gamma}$-complex.
- Rule 4: After the activation of gbc1, the $G_{\beta\gamma}$-complex may release itself from the receptor to seek out an effector to propagate the signal further. Rule 4 captures this breakage of the complexation-link between the internal receptor-domain, re1, and the domain of $G_{\beta\gamma}$, gbc0.
- Rule 5: And in rule 5 the $G_{\beta\gamma}$-complex, represented by the activated gbc1-domain, binds to some effector-domain—here abstractly represented as an eff-molecule. Note that for variation, here we have made the formation of the

$$
\begin{array}{ll}
\mathrm{r}_0 & \mathsf{lig}_l \mid \mathsf{re0}_r \;\to\; (x)\,\mathsf{lig}_l^x \mid \mathsf{re0}_r^x \\
\mathrm{r}_1 & (\mathsf{lig}_l^x \mid \mathsf{re0}_r^x) \mid\mid \left(\mathsf{re1}_r^y \mid \mathsf{gbc0}_b^y \mid \mathsf{gbc2}_b^z \mid \mathsf{ga0}_a^z \mid \overline{\mathsf{ga1}}_a\right) \to_\eta \\
 & (\mathsf{lig}_l^x \mid \mathsf{re0}_r^x) \mid\mid \left(\mathsf{re1}_r^y \mid \mathsf{gbc0}_b^y \mid \mathsf{gbc2}_b^z \mid \mathsf{ga0}_a^z \mid \mathsf{ga1}_a\right) \\
\mathrm{r}_2 & \mathsf{ga1}_a \mid \mathsf{gtp}_g \;\to\; (u)\,\mathsf{ga1}_a^u \mid \mathsf{gtp}_g^u \\
\mathrm{r}_3 & (z)\,\overline{\mathsf{gbc1}}_b \mid \mathsf{gbc2}_b^z \mid \mathsf{ga0}_a^z \mid \mathsf{ga1}_a^u \mid \mathsf{gtp}_g^u \;\to \\
 & \mathsf{gbc1}_b \mid \mathsf{gbc2}_b \mid \mathsf{ga0}_a \mid \mathsf{ga1}_a^u \mid \mathsf{gtp}_g^u \\
\mathrm{r}_4 & (x)\,\mathsf{re1}_r^x \mid \mathsf{gbc0}_b^x \mid \mathsf{gbc1}_b \;\to\; \mathsf{re1}_r \mid \mathsf{gbc0}_b \mid \mathsf{gbc1}_b \\
\mathrm{r}_5 & \mathsf{gbc1}_b \mid \mathsf{eff}_e \;\to\; (y)\,\mathsf{gbc1}_b^y \mid \mathsf{eff}_e^y \\
\mathrm{r}_6 & \mathsf{re1}_r \mid \mathsf{gbc0}_b \mid \overline{\mathsf{gbc1}}_b \mid \mathsf{gbc2}_b^z \mid \mathsf{ga0}_a^z \mid \overline{\mathsf{ga1}}_a \;\to \\
 & (x)\,\mathsf{re1}_r^x \mid \mathsf{gbc0}_b^x \mid \overline{\mathsf{gbc1}}_b \mid \mathsf{gbc2}_b^z \mid \mathsf{ga0}_a^z \mid \overline{\mathsf{ga1}}_a
\end{array}
$$

Figure 7.23: Rules for modelling a G-protein coupled receptor

bond between the effector and the $G_{\beta\gamma}$-complex dependent of the breakage of the bond between the receptor and the $G_{\beta\gamma}$-complex.

- Rule 6: Finally, rule 6 captures that a receptor with no G-protein may bind to another (inactive) G-protein.

7.4.2 Clathrin-dependent Endocytosis

For an example involving both formation and fusion of compartments, we turn to one of the celebrities in the world of research organisms, the illustrious nematode worm *Caenorhabditis elegans* (or *C. elegans* among friends). It exists in spades in common dirt, and is a multicellular eukaryote, with a small enough amount of cells (959 in adult hermaphrodites, 1031 in adult males) to have been extensively researched. This has made it possible to document in great detail the developemental lineage of every cell in the lifespan of a single organism. A key property for the tractability of such a study has also been that unlike mammals, for instance, in *C. elegans* the fate of cells are largely invariant between individual worms. The online compendium WormBook [WB] documents all aspects of research into these wondrous creatures.

We shall focus on endocytosis—the process by which cells internalize various extracellular material—as studied for *C. elegans*. The best studied endocytic pathway for both worms and mammals is the receptor-mediated endocytosis by formation of clathrin-coated vesicles. We base the following high-level presentation on the excellent review by Grant and Sato [GS06, Section 1] (available in the WormBook). The schematic illustration of clathrin-dependent endocytosis in Figure 7.4.2 we also borrow from *loc. cit.*

We may describe the process of clathrin-dependent endocytosis as follows: On the cell wall receptor proteins await, ready to bind to ligands, cargo molecules that the cell wishes to internalize. The purpose of the uptake of such cargo molecules

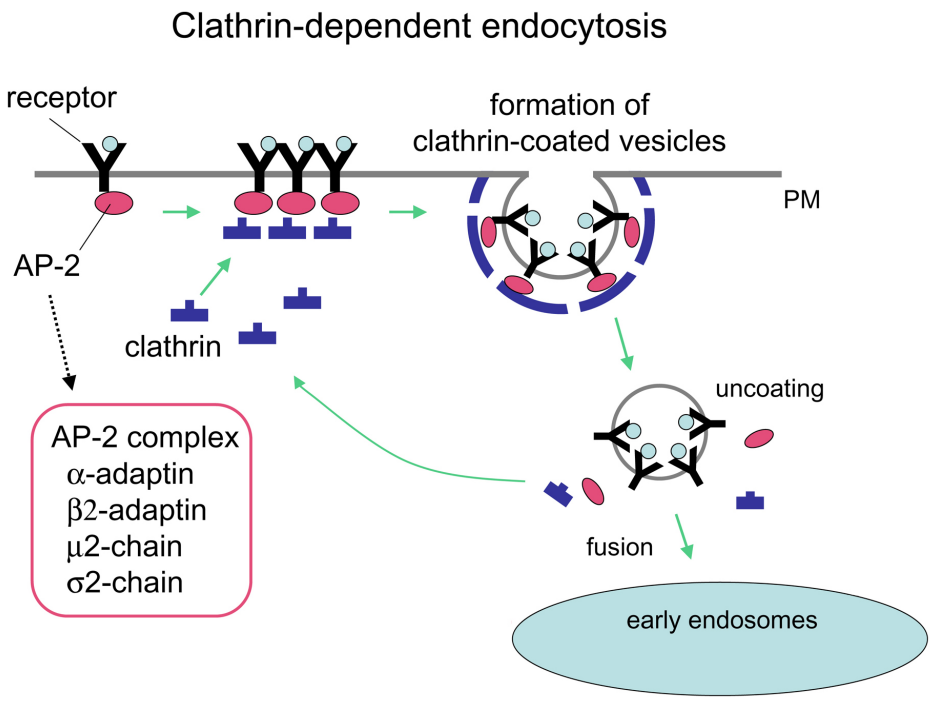


Figure 7.24: Mechanism of clathrin-dependent endocytosis. Clathrin and cargo molecules are assembled into clathrin-coated pits on the plasma membrane together with an adaptor complex called AP-2 that links clathrin with transmembrane receptors, concluding in the formation of mature clathrin-coated vesicles (CCVs). CCVs are then actively uncoated and transported to early/sorting endosomes (*illustration and caption from* [GS06]).

may be diverse, and may be part of signalling pathways, but may also, for instance, involve the uptake of molecules that the cell needs—such pathways inevitably lead the digested particles to degradation (i.e., destruction) in the lysosomes (with the suitable nickname, the "suicide sacs"). Over time, it so happens that clusters of receptors bound to ligands tend to form on the plasma membrane. The intracellular part of the receptors bind to adaptor complexes called AP2, who in turn bind to so-called clathrin lattices. The macro-shape of clusters of clathrin is thought to provide the down-force required to curve the plasma membrane into small inward buds. These inward buds are mechanically constricted and snipped off to form clathrin-coated vesicles floating freely in the cytoplasm. These coated vesicles are subsequently uncoated actively (with the help of chaperone protein hsc-70 and its compatriot the co-chaperone auxillin). With the aid of another GTPase, Rab5, these uncoated vesicles now fuse with each other and with so-called early endosomes, which serve a sorting purpose. These endosomes have a slightly reduced pH inside their membranes, which tends to cause the receptors to let go of their ligands. From the endosomes most receptors (some still with ligands bound) now form new vesicles

and eventually recycle to the cell wall; while most ligands continue their transport—possibly through further vesicle transport towards their endgoal, say, lysosomes.

At this point we shall leave our protagonists, and turn to developing a $\mathcal{C}$-calculus model. We underline again that, though detailed, the description above is quite high-level. Most steps in the pathway involve intricate biochemical steps; to name one, for instance, the snipping of buds happens through the intervention of a GTPase dynamin—it works essentially by forming a coil around the invaginated bud and then gradually, through a process of GTP hydrolysis, mechanically constricts until the bud is closed off to form a coated vesicle. In our model, we shall aim at hitting the same level of abstraction as in the description above, focusing mostly on illustrating how $\mathcal{C}$-calculus allows us to model induced vesicle formation, and later fusion of such vesicles with each other and with endosomes.

A $\mathcal{C}$-calculus model We make certain simplifications to make the model manageable: We model the 4-subunit complex AP2 as a single protein with only the two domains necessary for the reactions in this example. Also, we do not model exactly how the chaperone and co-chaperone, hsc-70 and auxillin, or the GTPase Rab5 work; we only require that they be present for certain reaction to take place. (For the same reason, we shall elide the mention of any complexation-ports on these three latter proteins.)

We arrive at the following protein signature:

$$\{\mathsf{lig} : (1,0), \mathsf{re} : (2,1), \mathsf{ap} : (2,0), \mathsf{cla} : (1,0), \mathsf{hsc\text{-}70} : (0,0), \mathsf{aux} : (0,0), \mathsf{rab} : (0,0)\}.$$

We take three membrane types plasma, ves, and end; the latter two types model endosomes and vesicles.

In Figure 7.25, we give a set of rules to model clathrin-dependent endocytosis as described above. Again, we explain the rules one-by-one.

- Rules 0, 1, and 2: These rules allow a ligand to bind to the extra-cellular receptor domain, $\mathsf{re0}$, (rule 0); allow the receptor-binding domain, $\mathsf{ap0}$, of the adaptin AP2 to bind to the intra-cellular receptor domain $\mathsf{re1}$, (rule 1) inducing a conformational change that allows clathrin, cla, to bind to the adaptin domain $\mathsf{ap1}$ (in rule 2).

- Rule 3: Rule 3 is a refinement of PINCH. (Since there is no need to repeat the generic side-condition for PINCH or DIFF, we just write by the rule $\varphi_{\text{PINCH}}(\mathbb{T} = \ldots)$ to remind that the rule has a side-condition.)

  We take the downwards projection of to model the clathrin-coating and forming of a vesicle inside a cell. The rule allows creations of buds of vesicle-type, $\mathsf{ves}[\ldots]$. We have refined the vanilla PINCH-rule to require at least one receptor coated with clathrin to be present for the bud to form, by setting $\mathbb{T} = \mathsf{re0}^x_r$, requiring the extra-cellular part of the receptor to be bound (to a ligand), and giving application conditions inside the plasma membrane to require the intra-cellular part to clathrin-coated. (For simplicity, we suppose that the downforce

$$
\begin{array}{ll}
\mathrm{r}_0 & \mathsf{lig}_l \mid \mathsf{re0}_r \to (x)\, \mathsf{lig}_l^x \mid \mathsf{re0}_r^x \\[1ex]
\mathrm{r}_1 & \mathsf{re1}_r \mid \mathsf{ap0}_a \mid \overline{\mathsf{ap1}}_a \to (x)\, \mathsf{re1}_r^x \mid \mathsf{ap0}_a^x \mid \mathsf{ap1}_a \\[1ex]
\mathrm{r}_2 & \mathsf{ap1}_a \mid \mathsf{cla}_c \to (x)\, \mathsf{ap1}_a^x \mid \mathsf{cla}_c^x \\[1ex]
\mathrm{r}_3 & B^- \mid B^+ \mid \mathsf{re0}_r^x \mid \mathsf{plasma}[\mathsf{re1}_r^y \mid \mathsf{ap0}_a^y \mid \mathsf{ap1}_a^z \mid \mathsf{cla}_c^z] \to \\
& (g)\, \triangle_g \mid B^- \mid \mathsf{plasma}[\mathsf{ves}[\triangle_g \mid B^+ \mid \mathsf{re0}_r^x] \mid \\
& \qquad\qquad \mathsf{re1}_r^y \mid \mathsf{ap0}_a^y \mid \mathsf{ap1}_a^z \mid \mathsf{cla}_c^z],\ \varphi_{\mathrm{PINCH}}(\mathbb{T} = \mathsf{re0}_r^x) \\[1ex]
\mathrm{r}_4 & (g)\ (\, B^- \mid B^+ \mid \mathsf{lig}_l^x \mid \triangle_g) \mid\mid (\triangle_g \mid A) \to_\omega \\
& (g)\ (\, B^- \mid \triangle_g) \mid\mid (\triangle_g \mid B^+ \mid \mathsf{lig}_l^x \mid A)\,, \qquad\qquad \varphi_{\mathrm{DIFF}}(\mathbb{T} = \mathsf{lig}_l^x) \\[1ex]
\mathrm{r}_5 & (g)\, \triangle_g \mid \mathsf{plasma}[A \mid \mathsf{re1}_r^y \mid \mathsf{ap0}_a^y \mid \mathsf{ap1}_a^z \mid \mathsf{cla}_c^z \mid \mathsf{re1}_s^v \mid \\
& \qquad\qquad \mathsf{ap0}_b^v \mid \mathsf{ap1}_b^w \mid \mathsf{cla}_d^w \mid \mathsf{ves}[\triangle_g \mid \mathsf{re0}_r^x \mid \mathsf{re0}_s^u \mid B]\,] \to \\
& \mathsf{plasma}[A \mid \mathsf{re1}_r^y \mid \mathsf{ap0}_a^y \mid \mathsf{ap1}_a^z \mid \mathsf{cla}_c^z \mid \mathsf{re1}_s^v \mid \\
& \qquad\qquad \mathsf{ap0}_b^v \mid \mathsf{ap1}_b^w \mid \mathsf{cla}_d^w \mid \mathsf{ves}[\mathsf{re0}_r^x \mid \mathsf{re0}_s^u \mid B]\,] \\[1ex]
\mathrm{r}_{6a} & (y, z)\, \mathsf{hsc\text{-}70}_h \mid \mathsf{aux}_b \mid \mathsf{re1}_r^y \mid \mathsf{ap0}_a^y \mid \mathsf{ap1}_a^z \mid \mathsf{cla}_c^z \to \\
& (z)\, \mathsf{hsc\text{-}70}_h \mid \mathsf{aux}_b \mid \mathsf{re1}_r \mid \mathsf{ap0}_a \mid \mathsf{ap1}_a^z \mid \mathsf{cla}_c^z \\[1ex]
\mathrm{r}_{6b} & (z)\, \mathsf{ap0}_a \mid \mathsf{ap1}_a^z \mid \mathsf{cla}_c^z \to \mathsf{ap0}_a \mid \mathsf{ap1}_a \mid \mathsf{cla}_c \\[1ex]
\mathrm{r}_{7a} & \mathsf{rab}_r \mid \mathsf{ves}[\, A\,] \mid \mathsf{ves}[\, B\,] \to (g)\, \mathsf{rab}_r \mid \mathsf{ves}[\, \triangle_g \mid A\,] \mid \mathsf{ves}[\, \triangle_g \mid B\,] \\[1ex]
\mathrm{r}_{7b} & \mathsf{rab}_r \mid \mathsf{ves}[\, A\,] \mid \mathsf{end}[\, B\,] \to (g)\, \mathsf{rab}_r \mid \mathsf{ves}[\, \triangle_g \mid A\,] \mid \mathsf{end}[\, \triangle_g \mid B\,] \\[1ex]
\mathrm{r}_{8a} & (g)\, \mathsf{ves}[\, \triangle_g \mid A\,] \mid \mathsf{ves}[\, \triangle_g \mid B\,] \to \mathsf{ves}[\, A \mid B\,] \\[1ex]
\mathrm{r}_{8b} & (g)\, \mathsf{ves}[\, \triangle_g \mid A\,] \mid \mathsf{end}[\, \triangle_g \mid B\,] \to \mathsf{end}[\, A \mid B\,] \\[1ex]
\mathrm{r}_9 & (x)\, \mathsf{end}[\, \mathsf{lig}_l^x \mid \mathsf{re0}_r^x\,] \to \mathsf{end}[\, \mathsf{lig}_l \mid \mathsf{re0}_r\,]
\end{array}
$$

Figure 7.25: Rules for modelling clathrin-dependent endocytosis

of a single clathrin-coating is enough to create a small bud.) The extra-cellular receptor-domain is pulled into the bud—dragging, in the surroundings, B^+, the bound ligand—and a channel is created.

- Rule 4: We refine the vanilla DIFF-generator to enable free diffusion of ligands in and out of the pinched bud. For variation, in this rule we just take every (homogenous) projection of the DIFF (indicated by keeping the ω-subscript on the reaction-arrow)—for every kind of membranes in the signature. This allows the rule to also allow diffusion among fused uncoated vesicles and endosomes later in the process—which we comment on below. Note, that should the ligand be bound to a receptor, then the receptor will be dragged into the bud (or out of the bud) as well as part of the surroundings, B^+. This may result in further or lessened clathrin-coating of a bud.

- Rule 5: Having pinched a bud via rule 3, rule 5 allows the bud to break off and part from the plasma membrane. Rule 5 is a straight refinement of the downwards projection of the PART-generator. For variation, we suppose, that the downforce of two clathrin-coatings is necessary to break off a bud. This necessitates the diffusion of more bound ligands into a formed bud—via rule 4.

- Rules 6a and 6b: These rules model, on a high level of abstraction, the uncoating of vesicles (as discussed above) in two steps. We simple require the presence of the two representatives of the chaperone hsc-70, hsc-70, and the co-chaperone auxillin, aux, for the uncoating reaction to occur.

- Rules 7a, 7b, 8a, and 8b: These rules allow the two-step procedure of TOUCH and MERGE, that results on fusion of uncoated vesicles with each other (rules 7a and 8a) and with endosomes-compartments, end[...] (rules 7b and 8b). For both fusions to occur, we require the GTPase Rab5, represented by rab, to be present for initiating the process. To illustrate the leverage we have in choosing refinements, for these rules, we take only the lateral projections of the TOUCH- and MERGE-rules; thus the reaction-arrows have no subscripts.

- Rule 9: Finally, rule 9 models that the cargo-ligand may be released when it has been enveloped by an endosome.

As a final remark, we may note, that one might also consider adding a rule to allow smaller clathrin-coated buds formed on the plasma membrane to merge, thus forming larger buds. This is probably a biologically viable alternative explanation to the formation of larger clusters of clathrin-coated receptors, resulting in larger buds.

7.5 Conclusion

We have presented and motivated in detail a novel calculus, the $\mathcal{C}$-calculus, for modelling low-level interaction inside and among cells, the basic building blocks of

all known life. The focus is on two main actors of cells, proteins and membranes.

The calculus takes inspiration from several calculi, in particular, the κ-calculus and the brane-calculi. The extension of a κ-like calculus with dynamic compartments is a novel contribution, in itself. The $\mathcal{C}$-calculus also introduces a novel abstraction, *channels*, for modelling partially fused membranes, and uses these to give a novel treatment of transport of complexes through diffusion and parametric pinching of buds. Furthermore, as opposed to most process calculi the $\mathcal{C}$-calculus allows domain experts considerable freedom in instantiating a domain-specific sub-calculus for the study of a particular biological application.

A user develops a model in the $\mathcal{C}$-calculus by selecting a set a proteins and membranes to work with and selects reaction rules from a toolbox of core rules, encapsulating basic biological actions. Subsequently, the core rules may be refined by giving contextual application conditions for rules.

In summary, the slogan for modelling in the $\mathcal{C}$-calculus is: *Modelling by rule refinement.*

7.5.1 Future Work

We should start by underlining that the reason for including names for domains and membranes was to be able to express directly the concrete examples in Section 7.4. The central parts of the calculus—the core actions, captured in the projective generators, and the capture of diffusion and fission (via PINCH)—could have been discussed without names and state for domains, and does not rely on the specific model chosen. In this first version of the calculus, the concrete model for representing names and state for domains (and membranes) has been chosen mainly because it allows readers familiar with the κ-calculus as presented in [DL04] to build on the intuition from there. In future work, we expect to revise the model for names and state, to allow the expression of rules that mention domains eliding their name or state, and, more generally, to allow a more uniform treatment of names and state for all the entities in the calculus.

The rules for transport in the $\mathcal{C}$-calculus allow only transport of proteins and complexes. However, by generalizing the PINCH generator, we may model cell-division in eukaryotic cells. As sketched in the introduction, this involves, in particular, the division of membrane-enclosed organelles into two daughter-cells. We may model this by allowing membranes and gates to travel across channels. Due to the isolation of the computation of the connected component in the lcc function, it suffices with a simple refinement of the PINCH-rule and of the lcc-function.

The generic tool for working with bigraphs, the BPL tool [BPL07], allows experimentation with the fragment of the $\mathcal{C}$-calculus without transport (i.e., the rules PINCH and DIFF). In experimenting with different semantics for the calculus, this has very been useful. However, as for the κ-calculus [DFFK07], a dedicated implementation can be optimized directly for the $\mathcal{C}$-calculus. For instance, knowing that well-formedness is preserved across reaction allows an implementation to optimize for this, utilizing, in particular, that all links except backbones are binary, and that

backbone links are restricted to one or two specific shapes.

A good way to mature the language is to investigate larger models. In particular, the level of abstraction in the $\mathcal{C}$-calculus leads us to believe that we may capture smoothly the steps from receptor transcription to placement (this is a part of the so-called exo-cytic pathway). This involves, in particular, the steps of transcription for exo-particles and transmembrane receptor proteins, which in turn involves recognition of partial proteins in the cytosol during transcription by ribosomes. As domains, not full proteins, are atoms in the $\mathcal{C}$-calculus, we should be able to capture these reactions by extending the allowable domain-level reactions. Recognition of a partial protein induces a lodging of the partial protein in the endoplasmic reticulum surrunding the nucleus—this is essentially a merging between two kinds of membrane-enclosed compartments. After final transcription, a vesicle is formed to encapsulate the exo-part of the protein, and the vesicle is transported through the (3-layer) Golgi apparatus sorting mechanism, and finally merged with the plasma membrane.

Our binary gated channels also bear a striking resemblance to a recent discovery [RSM$^+$04, GBG08], so-called *tunneling nanotubes* (TNTs), long and thin stable membrane-enclosed nanotubes (diameter of of 50 to 200nm, several cell diameters in length) tunneling through the divide between two cells. TNTs were originally reported to allow transport only of vesicles (i.e., essentially diffusion of vesicles), but seem later to have been found to permit direct intercellular transfer of organelles, cytoplasmic molecules and membrane components [GBG08]. Many things are as yet unknown about these entities and the mechanisms and conditions they work under. They seem to be correspondents of membrane channels observed in plants, so-called *plasmodesmata* (PD), membrane channels providing both membrane and cytoplasmic connectivity between cells. The paper by Gurke et al. [GBG08], provides a recent overview and summary of the current knowledge about TNTs, and contains an overview of three proposals on the nano-structure and mechanics of TNTs. It would be valuable to investigate whether, in the $\mathcal{C}$-calculus, we may model easily each of these three different proposals.

We have also made a preliminary investigation of the analysis of *causality* for pure bigraphs, which we expect to be instantiable for the $\mathcal{C}$-calculus. Essentially, we may give a presentation for describing causal relations in terms of modification-testing dependencies, in which we may describe classical notions such as precedence, weak permutation, casuality, and concurrency.

Finally, due to recent results by Milner, Krivine, and Troina [KMT08], a stochastic extension of the nondeterministic calculus presented in this paper, is well-founded.

7.5.2 Related Work

Several languages have been proposed in order to model biomolecular systems. We have already discussed in detail, how the $\mathcal{C}$-calculus is inspired by several previous efforts, in particular, by the κ-calculus [DL04]. The Bio-κ-calculus by Laneve and Tarissan extends the κ-calculus with basic brane-inspired membranes [LT06].

However, splitting of membranes is a non-atomic procedure, which requires a considerable amount of (computationally expensive) encoding. In comparison, in the $\mathcal{C}$-calculus we have taken the atomic *pinch* primitive for membrane division, and isolated any associated splitting of proteins in a side-condition.

The idea to use formal calculi for mobile and distributed systems was launched by Regev, Shapiro, and Silverman. They use the π-calculus to represent and simulate [RSS01] biomolecular processes underlying protein signalling networks. They continue that work in the stochastic version of the π-calculus [PRSS01] (as treated by Priami in [Pri95]) taking into account both the time and probability of biochemical reactions. Proteins are encoded as parallel products of a series of domains, encoded via channels. As in the $\mathcal{C}$-calculus, name-sharing encodes both protein-backbones and complexation. As is standard in the π-calculus locality (i.e., membrane-encapsulated compartments) is encoded via the sharing of names, and mobility is encoded via name-passing. Conformational change is also encoded via mobility. The basic combinators of π-calculus suffice to encode sequential events, independent events (via parallel), or, mutually exclusive events (via sum). The line of work on using languages inspired by the stochastic π-calculus is continued in the school of work on β-binders [PQ05] lead by Priami and co-authors. In particular, β-binders add syntactic constructs for modelling membranes, but do not offer any solution to the problem of fission.

In Bio-ambients [RPS$^+$04], Regev et al. extend their earlier work by developing a modified version of the ambient calculus developed by Cardelli and Gordon [CG98, CG00]. The ambient calculus is extended to include parent-to-child communication as well as intra-ambient communication. The calculus is also equipped with a variety of dual capabilities to allow actions such as *entry*, *exit*, and *merging* of ambients. Named ambients model membrane-enclosed compartments; the ambient abstraction is also used to model proteins, allowing merge to model certain kinds of complex-formation and membrane fusion. The calculus encodes cross-membrane proteins by parent-child communication—by allowing proteins to communicate outside the membrane they are enclosed in. This has the side-effect of requiring the user to *not* model cross-membrane proteins via ambients, but instead as naked processes. Also, protein complex breakage is complicated by using merge to model complex-formation.

Cardelli continues the investigation of membrane interactions in the family of brane calculi [Car04a]. Here, the focus is aimed mainly at finding good abstractions for membranes and their behavior. Molecular structures are limited to atomic structures attached to membrane-surfaces. Cardelli identifies a set of basic membrane-actions that is motivated by the hydrophilic and hydrophobic properties of membranes, and coin the concept "bitonality" to describe their constraints. In capturing a small set of membrane-actions in the $\mathcal{C}$-calculus, we are inspired by the brane calculi, and, in particular, by the *projective* variant of the brane calculi, by Danos and Pradellier [DP04]. Projective descriptions (which we explain and define in detail in the paper) let us describe conveniently sets of rules for reactions involving regions separated by membrane-surfaces, while eliding the *orientation* of those separating

membranes.

In the recent paper on Bitonal Membrane Systems [Car08], Cardelli continues the line of work on membrane calculi, and investigates further the notion of projectivity and orientation to unify membrane-reactions that from a local perspective (a so-called local *patch* of the membranes) involve the same pattern of reconfiguration.

Finally, also related is the school of work on P-systems [Pau02] initiated by Paun. P-systems are essentially rewriting systems with locations at a high level abstraction. The aim of P-systems is more angled towards developing a general computational model *inspired* by biology, than towards giving a language for describing and investigating biology, however.

Acknowledgements The authors are grateful for discussions with and suggestions from Lars Birkedal, Søren Debois, Luca Cardelli, and Robin Milner. 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 third author was a research fellow in the same group.

7.A Definitions Inherited from the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework

We include below, for easy reference, a few central definitions from the $\mathcal{B}^{\Sigma,\mathcal{R}}$-framework, instantiated to the signature used for the $\mathcal{C}$-calculus. For more extensive explanation and motivation, we refer the reader to [DK08].

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 solution $\Box_{45} \mid \Box_{56}$ from $\Box_0 \mid \Box_1$. All variables in terms 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.

Formally, we define order-preserving renumbering, and include it in the structural congruence relation. Write $\mathrm{nv}(C)$ for the set of numbers of variables in C.

Definition 7.A.1 (order-preserving renumbering)**.** Given a group G and an order-preserving and injective map $r : \mathrm{nv}(C) \to \mathbb{N}$, let $[\mathsf{G}]^r$ be the group G with all variables renumbered according to r.

Order-preserving renumberings are partial maps on natural numbers. We define also the *stratifying* renumbering, the renumbering that maps every variable to the number given by its relative order. To that end, we order the renumberings themselves, pointwise.

Definition 7.A.2 (stratifying renumbering)**.** The stratifying renumbering of variables in a group G is the least order-preserving renumbering defined on $\mathrm{nv}(\mathsf{G})$.

We write $[\mathsf{G}]$ for the group G renumbered according to this renumbering, and call $[\mathsf{G}]$ stratified.

Free and bound names The new name operator $(x)\,\mathsf{S}$ is a binder for pure names—instances of the name x in S are α-convertible. We define inductively the set of free or bound names of a term as usual.

Definition 7.A.3 (free and bound names)**.** For solutions S the free names $\mathrm{fn}(\mathsf{S})$ and the bound names $\mathrm{bn}(\mathsf{S})$ are defined inductively as:

$$\begin{array}{rclcrcl}
\mathrm{fn}(\mathsf{m}[\mathsf{S}]) &=& \mathrm{fn}(\mathsf{m}[\mathsf{S}]) & & \mathrm{bn}(\mathsf{m}[\mathsf{S}]) &=& \mathrm{bn}(\mathsf{S}) \\
\mathrm{fn}(\mathsf{d}_b^x) &=& \{x, b\} & & \mathrm{bn}(\mathsf{d}_b^x) &=& \emptyset \\
\mathrm{fn}(\mathsf{d}_b) &=& \{b\} & & \mathrm{bn}(\mathsf{d}_b) &=& \emptyset \\
\mathrm{fn}(\overline{\mathsf{d}}_b) &=& \{b\} & & \mathrm{bn}(\overline{\mathsf{d}}_b) &=& \emptyset \\
\mathrm{fn}(\triangle_g) &=& \{g\} & & \mathrm{bn}(\triangle_g) &=& \emptyset \\
\mathrm{fn}(\mathsf{S} \mid \mathsf{T}) &=& \mathrm{fn}(\mathsf{S}) \cup \mathrm{fn}(\mathsf{T}) & & \mathrm{bn}(\mathsf{S} \mid \mathsf{T}) &=& \mathrm{bn}(\mathsf{S}) \cup \mathrm{bn}(\mathsf{T}) \\
\mathrm{fn}((x)\,\mathsf{S}) &=& \mathrm{fn}(\mathsf{S}) \setminus x & & \mathrm{bn}((x)\,\mathsf{S}) &=& x \cup \mathrm{bn}(\mathsf{S}) \\
\mathrm{fn}(0) &=& \emptyset & & \mathrm{bn}(0) &=& \emptyset \\
\mathrm{fn}(A) &=& \emptyset & & \mathrm{bn}(A) &=& \emptyset
\end{array}$$

For groups we extend the definition above to include also:

$$\begin{array}{rclcrcl}
\mathrm{fn}(\mathsf{G} \mid\mid \mathsf{F}) &=& \mathrm{fn}(\mathsf{G}) \cup \mathrm{fn}(\mathsf{F}) & & \mathrm{bn}(\mathsf{G} \mid\mid \mathsf{F}) &=& \mathrm{bn}(\mathsf{G}) \cup \mathrm{bn}(\mathsf{F}) \\
\mathrm{fn}((x)\,\mathsf{G}) &=& \mathrm{fn}(\mathsf{G}) \setminus x & & \mathrm{bn}((x)\,\mathsf{G}) &=& x \cup \mathrm{bn}(\mathsf{G}) \\
\mathrm{fn}(\epsilon) &=& \emptyset & & \mathrm{bn}(\epsilon) &=& \emptyset
\end{array}$$

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 solutions or groups to bring every solution and group to a normal form, resembling the standard form for CCS [Mil80].

Proposition 7.A.4 (normal forms)**.** *Every solution S is structurally congruent to a normal form*

$$\mathsf{S} \equiv (\tilde{x})(\mathsf{S}_0 \mid \cdots \mid \mathsf{S}_{n-1})$$

where each $\mathsf{S}_0, \cdots, \mathsf{S}_{n-1}$ is a domain, a gate, or a membrane (empty or with another solution inside) containing no binders; and where $\tilde{x} \subseteq \mathrm{fn}(\mathsf{S}_0) \cup \cdots \cup \mathrm{fn}(\mathsf{S}_{n-1})$. (If $n = 0$, then $\mathsf{S}_0 \mid \cdots \mid \mathsf{S}_{n-1} \stackrel{\text{def}}{=} 0$, and if $\tilde{x} = \emptyset$ then the binder $(\tilde{x})$ is not there.)

Every group G is structurally congruent to a normal form

$$\mathsf{G} \equiv (\tilde{x})(\mathsf{S}_0 \mid\mid \cdots \mid\mid \mathsf{S}_{n-1})$$

where each $\mathsf{S}_0, \cdots, \mathsf{S}_{n-1}$ is non-empty, contain no binders, and is otherwise on (solution) normal form; and where $\tilde{x} \subseteq \mathrm{fn}(\mathsf{S}_0) \cup \cdots \cup \mathrm{fn}(\mathsf{S}_{n-1})$. (If $n = 0$, then $\mathsf{S}_0 \mid\mid \cdots \mid\mid \mathsf{S}_{n-1} \stackrel{\text{def}}{=} \epsilon$, 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 solutions (i.e., up to the commutative law for $\mid$).

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

Definition 7.A.5 (width of group)**.** For

$$\mathsf{G} \equiv_{\mathbf{N}} (\tilde{x})(\mathsf{S}_0 \mid\mid \cdots \mid\mid \mathsf{S}_{n-1})$$

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

Substitution Nonground solutions (and groups) have variables for which we may substitute other solutions.

We start by defining basic substitution of the variables in a group G by a solution S. (We define substitution only for groups, G. The definition for solutions follows as a special case.) Substitution is capture-avoiding, as usual.

Definition 7.A.6 (raw substitution)**.** Let φ be a bijective map from variables to solutions $\mathsf{S}_0, \mathsf{S}_1, \ldots, \mathsf{S}_{n-1}$.

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

$$
\begin{aligned}
\mathsf{m}[\mathsf{S}]\,\varphi &= \mathsf{m}[\mathsf{S}\varphi] \\
\mathsf{d}_b^x\,\varphi &= \mathsf{d}_b^x \\
\mathsf{d}_b\,\varphi &= \mathsf{d}_b \\
\overline{\mathsf{d}}_b\,\varphi &= \overline{\mathsf{d}}_b \\
\triangle_g\,\varphi &= \triangle_g \\
((x)\,\mathsf{S})\,\varphi &= (x)\,(\mathsf{S}\varphi) \\
(\mathsf{S} \mid \mathsf{T})\,\varphi &= (\mathsf{S}\varphi \mid \mathsf{T}\varphi) \\
0\,\varphi &= 0 \\
X\varphi &= \begin{cases} \mathsf{S}_i & \text{if } \varphi(X) = \mathsf{S}_i \\ X & \text{else} \end{cases} \\
\\
((x)\,\mathsf{G})\,\varphi &= (x)\,(\mathsf{G}\varphi) \\
(\mathsf{G} \mid\mid \mathsf{F})\,\varphi &= (\mathsf{G}\varphi \mid\mid \mathsf{F}\varphi) \\
\epsilon\,\varphi &= \epsilon.
\end{aligned}
$$

As each variable is a leaf in G it is easy to see that the substituted term respects the grammar (i.e., in Definition 7.2.4 and in Definition 7.2.5). In general, however, raw substitution is not well-behaved; it may lead to terms violating the requirement that numbered variables occur only once in expressions. 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, $\mathsf{G} \cdot \mathsf{F}$, the substitution of all variables in a group G with the solutions in a group F.

We define total substitution by pushing the binders of F to the top of the created term; hence, we also require that both bound and free names of G be distinct from bound names in F. This is simply a technical requirement, as we can always α-convert bound names of F to avoid any clashes.

Definition 7.A.7 (total substitution)**.** Substitution $\mathsf{G} \cdot \mathsf{F}$ of variables in G by solutions in F is defined when $|\operatorname{var}(\mathsf{G})| = \operatorname{width}(\mathsf{F})$ and $\operatorname{bn}(\mathsf{G}) \cap \operatorname{fn}(\mathsf{F}) = \operatorname{bn}(\mathsf{G}) \cap \operatorname{bn}(\mathsf{F}) = \operatorname{fn}(\mathsf{G}) \cap \operatorname{bn}(\mathsf{F}) = \emptyset$. In this case, for

$$\mathsf{F} \equiv_{\mathbf{N}} (\tilde{x})(\mathsf{S}_0 \mid\mid \cdots \mid\mid \mathsf{S}_{n-1})$$

let

$$\mathsf{G} \cdot \mathsf{F} = (\tilde{x})[\mathsf{G}]\{\Box_0 \mapsto \mathsf{S}_0, \cdots, \Box_{n-1} \mapsto \mathsf{S}_{n-1}\},$$

Note, that we use stratifying renumbering to renumber the variables in G before substituting, to ensure that the variables in G are numbered severally from 0 to $n-1$. Substitution is associative.

Proposition 7.A.8 (total substitution is associative)**.** $(\mathsf{G} \cdot \mathsf{F}) \cdot \mathsf{H} = \mathsf{G} \cdot (\mathsf{F} \cdot \mathsf{H})$.

We generalize our definition of substitution to partial substitution—where only some of the variables of the context G are substituted. Definition 7.A.9 generalizes Definition 7.A.7 to the cases, where F has fewer solutions than G has variables.

Definition 7.A.9 (partial substitution)**.** Partial substitution $\mathsf{G} \triangleleft \mathsf{F}$ of variables in G by solutions in F is defined when $|\operatorname{var}(\mathsf{G})| \geq \operatorname{width}(\mathsf{F})$ and (as for total substitution) when $\operatorname{bn}(\mathsf{G}) \cap \operatorname{fn}(\mathsf{F}) = \operatorname{bn}(\mathsf{G}) \cap \operatorname{bn}(\mathsf{F}) = \operatorname{bn}(\mathsf{G}) \cap \operatorname{fn}(\mathsf{F}) = \emptyset$. In this case, let

$$\mathsf{G} \triangleleft \mathsf{F} = \mathsf{G} \cdot ([\mathsf{F}] \mid\mid \Box_k \mid\mid \cdots \mid\mid \Box_{k+n}),$$

for $|\operatorname{var}(\mathsf{G})| - \operatorname{width}(\mathsf{F}) = n$ and $|\operatorname{var}(\mathsf{F})| = k$.

It is immediate from the definitions, that in the case where $|\operatorname{var}(\mathsf{G})| = \operatorname{width}(\mathsf{F})$, $\mathsf{G} \cdot \mathsf{F} \equiv \mathsf{G} \triangleleft \mathsf{F}$. In the case, where G has more variables, than F has solutions, F is extended with appropriately numbered variables.

Partial substitution is not in general associative; by convention we take it to be left-associative.

Bibliography

[AF01] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *POPL*, pages 104–115, 2001.

[AGG] AGG homepage. `http://tfs.cs.tu-berlin.de/agg/`.

[Bae05] Jos C. M. Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335(2-3):131–146, 2005.

[BBD+06] Lars Birkedal, Mikkel Bundgaard, Troels Christoffer Damgaard, Søren Debois, Ebbe Elsborg, Arne John Glenstrup, Thomas Troels Hildebrandt, Robin Milner, and Henning Niss. Bigraphical programming languages for pervasive computing. In Thomas Strang, Vinny Cahill, and Aaron Quigley, editors, *Proceedings of Pervasive 2006 International Workshop on Combining Theory and Systems Building in Pervasive Computing*, pages 653–658, May 2006.

[BDE+05] Lars Birkedal, Søren Debois, Ebbe Elsborg, Thomas Hildebrandt, and Henning Niss. Bigraphical Models of Context-aware Systems. Technical Report 74, IT University of Copenhagen, November 2005.

[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.

[BDH06] Lars Birkedal, Søren Debois, and Thomas Hildebrandt. Sortings for reactive systems. In Christel Baier and Holger Hermanns, editors, *CONCUR '06*, volume 4137 of *Lecture Notes in Computer Science*, pages 248–262. Springer, 2006.

[BDH08] Lars Birkedal, Søren Debois, and Thomas Hildebrandt. On the construction of sorted reactive systems. In Franck van Breugel and Marsha

Chechik, editors, *Proceedings of the 19th International Conference on Concurrency Theory 2008*, volume 5201 of *Lecture Notes in Computer Science*, pages 218–232. Springer-Verlag, August 2008.

[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.

[Bir04] Lars Birkedal. Bigraphical Programming Languages—a LaCoMoCo research project. In *Second UK UbiNet Workshop, Cambridge*, May 2004. position paper.

[BKM06] Filippo Bonchi, Barbara König, and Ugo Montanari. Saturated semantics for reactive systems. In *Proceedings of 21st IEEE Symposium on Logic in Computer Science (LICS'06)*, pages 69–80. IEEE Computer Society, 2006.

[BN98] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, New York, NY, USA, 1998.

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

[BR95] Roel Bloo and Kristoffer H. Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *In CSN-95: Computer Science in the Netherlands*, pages 62–72, November 1995.

[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.

[Bun07] Mikkel Bundgaard. *Semantics of Higher-Order Mobile Embedded Resources and Local Names*. PhD thesis, IT University of Copenhagen, 2007.

[Car04a] Luca Cardelli. Brane calculi. In Vincent Danos and Vincent Schächter, editors, *CMSB*, volume 3082 of *Lecture Notes in Computer Science*, pages 257–278. Springer, 2004.

[Car04b] Luca Cardelli. Brane calculi - interactions of biological membranes. In *Computational Methods in Systems Biology*, pages 257–278. Springer, 2004.

[Car08] Luca Cardelli. Bitonal membrane systems: Interactions of biological membranes. *Theor. Comput. Sci.*, 404(1-2):5–18, 2008.

[CDE+01] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José F. Quesada. Maude: Specification and programming in rewriting logic. *Theoretical Computer Science*, 2001.

[CDE+03] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. The Maude 2.0 System. In Robert Nieuwenhuis, editor, *Rewriting Techniques and Applications (RTA 2003)*, volume 2706 of *Lecture Notes in Computer Science*, pages 76–87. Springer-Verlag, June 2003.

[CG98] Luca Cardelli and Andrew D. Gordon. Mobile ambients. In Maurice Nivat, editor, *FoSSaCS*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer, 1998.

[CG99a] Andrea Corradini and Fabio Gadducci. An Algebraic Presentation of Term Graphs, via GS-Monoidal Categories. *Applied Categorical Structures*, 7:299–331, 1999.

[CG99b] Andrea Corradini and Fabio Gadducci. Rewriting on cyclic structures: Equivalence between the operational and the categorical description. *Theoretical Informatics and Applications*, 33(4/5):467–493, 1999.

[CG00] Luca Cardelli and Andrew Gordon. Mobile ambients. *Theoretical Computer Science*, 24:177–213, 2000.

[CL04] Luís Caires and Étienne Lozes. Elimination of quantifiers and undecidability in spatial logics for concurrency. In *CONCUR*, pages 240–257, 2004.

[Cle] The Clean System. `http://clean.cs.ru.nl/`.

[CMS05a] Giovanni Conforti, Damiano Macedonio, and Vladimiro Sassone. Bigraphical logics for XML. In *Proc. of Italian Symposium on Advanced Database Systems (SEBD'05)*, pages 392–399, 2005.

[CMS05b] Giovanni Conforti, Damiano Macedonio, and Vladimiro Sassone. Spatial Logics for Bigraphs. In *Proceedings of ICALP'05*, volume 3580 of *LNCS*, pages 766–778. Springer, June 2005.

[CMS06] Giovanni Conforti, Damiano Macedonio, and Vladimiro Sassone. Bilog: Spatial Logics for Bigraphs. *Information and Computation*, 2006. Submitted for publication.

[CMS07] Giovanni Conforti, Damiano Macedonio, and Vladimiro Sassone. Static BiLog: a unifying language for spatial structures. In Wojciech Penczek and Grzegorz Rozenberg, editors, *Half a century of inspirational research, honouring the scientific influence of Antoni Mazurkiewicz*, volume 80(1–3) of *Fundamenta Informaticae*, pages 91–110. IOS Press, 2007.

[Coh07] William Cohen. *A Computer Scientist's Guide to Cell Biology*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[Con06] Shane Ó Conchúir. Λ-sub as an explicit substitution calculus. Technical Report TR-2006-95, IT University of Copenhagen, September 2006.

[CPS93] Rance Cleaveland, Joachim Parrow, and Bernhard Steffen. The concurrency workbench: A semantics-based tool for the verification of concurrent systems. *ACM Trans. Program. Lang. Syst.*, 15(1):36–72, 1993.

[CWB] The Edinburgh Concurrency Workbench. `http://homepages.inf.ed.ac.uk/perdita/cwb/`.

[Dam06] Troels Christoffer Damgaard. Syntactic Theory for Bigraphs. Master's thesis, IT University of Copenhagen, December 2006. Master's Thesis.

[DB05] Troels C. Damgaard and Lars Birkedal. Axiomatization of binding bigraphs (revised). Technical Report 71, IT University of Copenhagen, October 2005.

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

[DD05a] Troels C. Damgaard and Søren Debois. Note on separately matching place and link graphs. Unpublished. Available on `http://www.itu.dk/people/tcd/docs/separate-matching.pdf`., February 2005.

[DD05b] Søren Debois and Troels C. Damgaard. Bigraphs by example. Technical Report TR-2005-61, IT University of Copenhagen, March 2005.

[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.

[DGBM07] Troels Christoffer Damgaard, Arne J. Glenstrup, Lars Birkedal, and Robin Milner. An inductive characterization of matching in binding bigraphs. Manuscript submitted for publication, September 2007.

[DHP02] Frank Drewes, Berthold Hoffmann, and Detlef Plump. Hierarchical graph transformation. *J. Comput. Syst. Sci.*, 64(2):249–283, 2002.

[DK08] Troels C. Damgaard and Jean Krivine. A generic language for biological systems based on bigraphs. Technical Report TR-2008-115, IT University of Copenhagen, Rued Langgaards Vej 7, DK-2300 Copenhagen V, December 2008.

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

[DP04] Vincent Danos and Sylvain Pradalier. Projective brane calculus. In *Proceedings of CMSB'04*, pages 134–148, 2004.

[DPPQ06] Pierpaolo Degano, Davide Prandi, Corrado Priami, and Paola Quaglia. Beta-binders for biological quantitative experiments. In *Proceedings of QAPL 2006*, volume 164 of *ENTCS*, pages 101–117, 2006.

[EEPT06] Hartmut Ehrig, Karsten Ehrig, Ulrike Prange, and Gabriele Taentzer. *Fundamentals of Algebraic Graph Transformation.* Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[Ehr79] Hartmut Ehrig. Introduction to the Algebraic Theory of Graph Grammars. In V. Claus, H. Ehrig, and G. Rozenberg, editors, *Graph-Grammars and Their Application to Computer Science and Biology*, volume 73 of *Lecture Notes in Computer Science*, pages 1–69, 1979.

[Ehr02] Hartmut Ehrig. Bigraphs meet double pushouts. *Bulletin of the EATCS*, 78:72–85, 2002.

[EHS08a] Ebbe Elsborg, Thomas T. Hildebrandt, and Davide Sangiorgi. Type Systems for Bigraphs. Technical Report 110, The IT University of Copenhagen, Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark, October 2008.

[EHS08b] Ebbe Elsborg, Thomas T. Hildebrandt, and Davide Sangiorgi. Type systems for bigraphs. In *Proceedings of the Fourth Symposium on Trustworthy Global Computing*. Springer, 2008. To appear.

[EK04] Hartmut Ehrig and Barbara König. Deriving bisimulation congruences in the DPO approach to graph rewriting. In Igor Walukiewicz, editor, *FoSSaCS*, volume 2987 of *Lecture Notes in Computer Science*, pages 151–166. Springer, 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.

[ERT99] C. Ermel, M. Rudolf, and G. Taentzer. The AGG approach: Language and Environment. In 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)*, pages 501–603. World Scientific, Singapore, 1999.

[Fu97] James Jianghai Fu. Directed graph pattern matching and topological embedding. *Journal of Algorithms*, 22(2):372–391, 1997.

[GBG08] Steffen Gurke, João F.V. Barroso, and Hans-Hermann Gerdes. The art of cellular communication: tunneling nanotubes bridge the divide. *Histochem Cell Biol.*, 129(5):539–550, May 2008.

[GDBH07] Arne J. Glenstrup, Troels Christoffer Damgaard, Lars Birkedal, and Espen Højsgaard. An implementation of bigraph matching. Manuscript submitted for publication, October 2007.

[GHL99] Fabio Gadducci, Reiko Heckel, and Mercè Llabrés. A bi-categorical axiomatisation of concurrent graph rewriting. *Electronic Notes in Theoretical Computer Science*, 29, 1999.

[GM07a] Davide Grohmann and Marino Miculan. Directed bigraphs. In *Proceedings of the 23rd Conference on the Mathematical Foundations of Programming Semantics (MFPS'07)*, volume 173 of *Electronic Notes in Theoretical Computer Science*, pages 121–137. Elsevier, 2007.

[GM07b] 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.

[GM08a] Davide Grohmann and Marino Miculan. An algebra for directed bigraphs. In Ian Mackie and Detlef Plump, editors, *Proceedings of the 4th International Workshop on Computing with Terms and Graphs 2008*, volume 203(1) of *Electronic Notes in Theoretical Computer Science*, pages 49–63. Elsevier, March 2008.

[GM08b] Davide Grohmann and Marino Miculan. Controlling resource access in directed bigraphs. In Claudia Ermel, Juan de Lara, and Reiko Heckel, editors, *Proceedings 7th International Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT 2008)*, volume 10 of *Electronic Communications of the EASST*. European Association of Software Science and Technology, 2008.

[GS06] B. D. Grant and M. Sato. Intracellular trafficking. *Wormbook*, January 2006. Available online at `http://www.wormbook.org`.

[HM80] Matthew Hennessy and Robin Milner. On observing nondeterminism and concurrency. In J. W. de Bakker and Jan van Leeuwen, editors, *ICALP*, volume 85 of *Lecture Notes in Computer Science*, pages 299–309. Springer, 1980.

[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.

[HNO06] Thomas Hildebrandt, Henning Niss, and Martin Olsen. Formalising business process execution with bigraphs and Reactive XML. In Paolo Ciancarini and Herbert Wiklicky, editors, *Proceedings of the 8th International Conference on Coordination Models and Languages (COORDINATION'06)*, volume 4038 of *Lecture Notes in Computer Science*, pages 113–129. Springer-Verlag, January 2006.

[HNOW06] Thomas T. Hildebrandt, Henning Niss, Martin Olsen, and Jacob W. Winther. Distributed reactive xml. *Electr. Notes Theor. Comput. Sci.*, 150(1):61–80, 2006.

[HR05] Reinhart Heinrich and Tom Rapoport. Generation of nonindentical compartments in vesicular transport systems. *The Journal of Cell Biology*, 168(2):271–280, January 2005.

[Jen06] Ole H. Jensen. Mobile processes in bigraphs, 2006. Monograph available at `http://www.cl.cam.ac.uk/~rm135/Jensen-monograph.html`.

[JM03] Ole H. Jensen and Robin Milner. Bigraphs and Transitions. In *Proceedings of POPL'03*, pages 38–49. ACM Press, 2003.

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

[KC02] Michael M. Kozlov and Leonid Chernomordik. The protein coat in membrane fusion: Lessons from fission. *Traffic*, 3(4):256–267, 2002.

[KK03] Yonathan Kozlovsky and Michael M. Kozlov. Membrane fisson: Model for intermediate structures. *Biophysical Journal*, 85(1):85–96, July 2003.

[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.

[LM00] James J. Leifer and Robin Milner. Deriving bisimulation congruences for reactive systems. In *Proceedings of CONCUR'00*, pages 243–258. Springer, 2000.

[LM04] James J. Leifer and Robin Milner. Transition systems, link graphs and Petri nets. Technical Report UCAM-CL-TR-598, University of Cambridge, August 2004.

[LT06] Cosimo Laneve and Fabien Tarissan. A simple calculus for proteins and cells. In *Proceedings of the Workshop MeCBIC'06.*, 2006.

[LV02] Javier Larrosa and Gabriel Valiente. Constraint satisfaction algorithms for graph pattern matching. *Mathematical Structures in Computer Science*, 12:403–422, 2002.

[Mau] The Maude System. `http://maude.cs.uiuc.edu/`.

[Mes96] Jose Meseguer. Rewriting logic as a semantic framework for concurrency: a progress report. In *International Conference on Concurrency Theory*, pages 331–372, 1996.

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

[Mil89] Robin Milner. *Communication and Concurrency*. International Series on Computer Science. Prentice Hall, 1989.

[Mil99] Robin Milner. *Communicating and Mobile Systems: The π-Calculus*. Cambridge U., 1999.

[Mil01] Robin Milner. Bigraphical reactive systems. In *CONCUR '01: Proceedings of the 12th International Conference on Concurrency Theory*, pages 16–35, London, UK, 2001. Springer-Verlag.

[Mil04] Robin Milner. Bigraphs whose names have multiple locality. Technical Report UCAM-CL-TR-603, University of Cambridge, Computer Laboratory, September 2004.

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

[Mil06a] Robin Milner. Local bigraphs and confluence: two conjectures. In Roberto Amadio and Iain Phillips, editors, *Proceedings of the 13th International Workshop on Expressiveness in Concurrency (EXPRESS'06)*, August 2006. To appear.

[Mil06b] 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.

[Min03] Mark Minas. Visual specification of Visual Editors with VisualDiaGen. In John L. Pfaltz, Manfred Nagl, and Boris Böhlen, editors, *AGTIVE*, volume 3062 of *Lecture Notes in Computer Science*, pages 473–478. Springer, 2003.

[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.

[MWB] The Mobility Workbench. `https://www.it.uu.se/research/group/mobility/mwb`.

[Nes06] Uwe Nestmann. Welcome to the jungle: A subjective guide to mobile process calculi. In Christel Baier and Holger Hermanns, editors, *CONCUR*, volume 4137 of *Lecture Notes in Computer Science*, pages 52–63. Springer, 2006.

[NSvP91] Eric Nocker, Sjaak Smetsers, Marko van Eekelen, and Rinus Plasmeijer. Concurrent Clean. In Aarts, Leeuwen, and Rem, editors, *Proc. of Parallel Architectures and Languages Europe (PARLE'91)*, volume 505, pages 202–219. Springer-Verlag, 1991.

[Pal04] Wojciech Palacz. Algebraic hierarchical graph transformation. *J. Comput. Syst. Sci.*, 68(3):497–520, 2004.

[Pau02] Gh. Paun. *Membrane Computing. An Introduction.* Springer, 2002. ISBN 3-540-43601-4.

[PE93] Rinus Plasmeijer and Marko Van Eekelen. *Functional Programming and Parallel Graph Rewriting.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1993.

[Pod06] Benjamin Podbilewicz. Cell fusion. *WormBook*, pages 1–32, January 2006. Available online at `http://www.wormbook.org`.

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

[Pri95] Corrado Priami. Stochastic π-calculus. *The Computer Journal*, 38(6):578–589, 1995.

[PRO] PROGRES homepage. `http://www-i3.informatik.rwth-aachen.de/research/projects/progres/`.

[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.

[PS96] Benjamin C. Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5):409–453, 1996.

[PV98] Joachim Parrow and Björn Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *LICS*, pages 176–185, 1998.

[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.

[Ren04] Arend Rensink. The GROOVE simulator: A Tool for State Space Generation. In J. Pfalz, M. Nagl, and B. Böhlen, editors, *Applications of Graph Transformations with Industrial Relevance (AGTIVE)*, volume 3062 of *Lecture Notes in Computer Science*, pages 479–485. Springer-Verlag, 2004.

[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.

[RSM+04] Amin Rustom, Rainer Saffrich, Ivanka Markovic, Paul Walther, and Hans-Hermann Gerdes. Nanotubular highways for intercellular organelle transport. *Science*, 303:1007–1010, 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.

[Sel77] Stanley M. Selkow. The tree-to-tree editing problem. *Information Processing Letters*, 6(6):184–186, 1977.

[Sew02] Peter Sewell. From rewrite rules to bisimulation congruences. *Theoretical Computer Science*, 274(1–2):183–230, 2002.

[SS02] V. Sassone and P. Sobocìnski. Deriving bisimulation congruences: A 2-categorical approach. In *Proceedings of Express '02*, number 68(2) in Electronic Notes in Theoretical Computer Science, May 2002.

[SS05] Vladimiro Sassone and Paweł Sobociński. Reactive systems over cospans. In *Proceedings of Logic in Computer Science (LICS'05)*, pages 311–320. IEEE Press, 2005.

[ST99] Ron Shamir and Dekel Tsur. Faster subtree isomorphism. *Journal of Algorithms*, 33(2):267–280, 1999.

[SWZ99] A. Schürr, A. Winter, and A. Zündorf. PROGRES: Language and Environment. In 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.

[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.

[Tur96] David N. Turner. *The Polymorphic Pi-calulus: Theory and Implementation*. PhD thesis, University of Edinburgh, 1996.

[Ull76] Jeffrey D. Ullman. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, 1976.

[Val02] Gabriel Valiente. *Algorithms on Trees and Graphs*. Springer, Berlin, 2002.

[Vic94] Björn Victor. *A Verification Tool for the Polyadic π-Calculus*. Licentiate thesis, Department of Computer Systems, Uppsala University, Sweden, May 1994. Available as report DoCS 94/50.

[VSV05] Gergely Varró, Andy Schürr, and Dániel Varró. Benchmarking for Graph Transformation. In *Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*, volume 00, pages 79–88, 2005.

[VVF05] Gergely Varró, Dániel Varró, and Katalin Friedl. Adaptive graph pattern matching for model transformations using model-sensitive search plans. In G. Karsai and G. Taentzer, editors, *GraMot 2005, International Workshop on Graph and Model Transformations*, Electronic Notes in Theoretical Computer Science, pages 191–205, 2005.

[WB] Online WormBook. `http://www.wormbook.org/`.

[Zün94] Albert Zündorf. Graph pattern matching in PROGRES. In J. Cuny, H. Ehrig, G. Engels, and G. Rozenberg, editors, *Proceedings of the 5th International Workshop on Graph-Grammars and their Application to Computer Science*, volume 1073 of *LNCS*, pages 454–468. Springer, 1994.