



The **IT** University  
of Copenhagen

# Syntactic Theory for Bigraphs

**Troels Christoffer Damgaard**

**Supervisor: Lars Birkedal**

# Syntactic Theory for Bigraphs

Troels Christoffer Damgaard  
The Bigraphical Programming Languages Group  
The IT University of Copenhagen

November 2006

## **Abstract**

I investigate and develop theory for term languages for a variant of bigraphs with binding, thus building the formal foundation for a (term-based) tool for bigraphical reactive systems.

I present two main results (developed with co-authors). First, I give an axiomatization of structural congruence (graph equivalence) for binding bigraphs. Along the way, I devise a term language for binding bigraphs and prove a series of normal form theorems for binding bigraphs. Second, using these results, I give a complete inductive characterization of matching in bigraphs — essentially, for describing when and where a bigraphical reaction rule can be applied.

I include an introduction to the goals of my Ph.D. project and explain how it relates to the goals of the BPL project. Moreover, I outline a number of future challenges and include a literature study for future work.

# Contents

<b>I</b>	<b>Overview of Completed and Proposed Work</b>	<b>3</b>
<b>1</b>	<b>Preface</b>	<b>3</b>
<b>2</b>	<b>Background</b>	<b>4</b>
2.1	The BPL Project . . . . .	4
2.2	A Brief Introduction to Bigraphs . . . . .	4
<b>3</b>	<b>Completed Work</b>	<b>9</b>
3.1	An Equational Theory for Binding Bigraphs . . . . .	11
3.2	An Inductive Characterization of Matching in Binding Bigraphs	13
<b>4</b>	<b>Proposed Work</b>	<b>14</b>
4.1	Overview . . . . .	14
<b>5</b>	<b>A Tool for BRSs</b>	<b>17</b>
<b>6</b>	<b>Litterature Study for Proposed Work</b>	<b>18</b>
6.1	Background . . . . .	19
6.2	Related Work on Bigraphs . . . . .	20
6.3	Term Rewriting . . . . .	23
6.3.1	Comparing TRSs and BRSs . . . . .	24
6.4	Graph Transformation . . . . .	25
6.4.1	Extensions . . . . .	27
6.4.2	Applications and Tools . . . . .	30
6.4.3	Simulating BRSs with Graph Transformation Systems	32
<b>7</b>	<b>Conclusion</b>	<b>33</b>
	<b>References</b>	<b>34</b>
<b>II</b>	<b>Axiomatizing Binding Bigraphs</b>	<b>42</b>
<b>III</b>	<b>Matching of Bigraphs</b>	<b>66</b>

## Part I

# Overview of Completed and Proposed Work

## 1 Preface

This is my qualification report for the A part of the 4-year Ph.D. programme at the IT University of Copenhagen. Formally, the report counts as my Master’s thesis and it describes core points of my research during the first two years of my Ph.D.

This qualification report is composed of three parts. Part I contains an introduction to my work, proposals for future work, and a literature study. Part II, “Axiomatizing Binding Bigraphs”, was published as [DB06]. Part III is an extended version of “Matching of Bigraphs” [BDGM06]. I presented this paper at the GT-VC workshop 2006 and it is currently under publication. Part II and Part III are presented unchanged; thus they are self-contained. Both contain introductions to bigraphs and bigraphical reactive systems and sections on related work.

The reader is expected to have a basic knowledge of calculi for mobile processes. In contrast, no knowledge of term rewriting and graph transformation is presupposed. Sections concerned with term rewriting and graph transformation contain (brief) introductions. Finally, as concerns bigraphs, Part I provides an informal introduction to bigraphs and their reactive systems — an expanded version of the brief introduction to bigraphs in Part II. I suggest, that you read Parts II and III before reading Section 4 of Part I, on proposals for future work. For the reader unfamiliar with bigraphs, I suggest that you read through Section 2 of Part III after reading the introduction in Part I, which contains a more thorough introduction to binding bigraphs.

For quick reference, the Appendix in Part II contains a terse recap of binding bigraphs.

## 2 Background

In this section, I give an outline of the background for my Ph.D. project. I briefly describe the goals of the Bigraphical Programming Languages (BPL) project, which I participate in; and, I give an informal introduction to bigraphs and their reactive systems.

### 2.1 The BPL Project

The BPL project is founded on the thesis that programming and specification languages for mobile and distributed systems may be based on the theoretical foundation of bigraphs. Thus, we seek to ease understanding and formal analysis of mobile systems [Bir04, BBD<sup>+</sup>06]. In particular, there has been a focus on context-aware systems in the domain of mobile ubiquitous computing; it is a long-term aim of the theory of bigraphs to represent easily and reason about such systems [Mil06b].

I investigate theory and techniques that support testing the thesis of the BPL project. To explain in more detail my work, I shall need to introduce the theory of bigraphs and their reactive systems.

In the following section, I give a brief informal introduction to bigraphs. The reader already familiar with bigraphs may skip this section. For a more thorough account, see Section 2 of Part III.

### 2.2 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 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 shows an example of a concrete 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.

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.

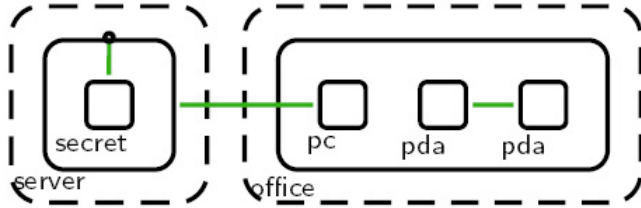


Figure 1:  $E$  — a bigraph model of an office

Bigraphs can contain *sites* (gray boxes), and inner or outer *names* (in red). Both roots and sites are ordered. The bigraph  $F$  in Figure 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 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.

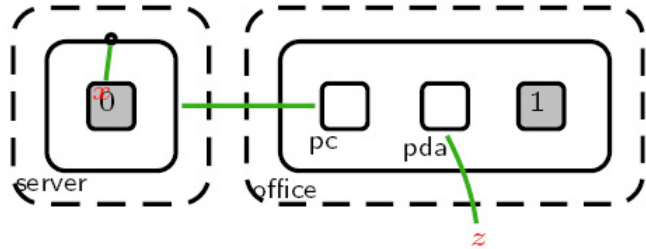


Figure 2:  $F$  — a bigraph context

A bigraph comprises two structures, a *place graph*, which is a forest of nodes, rooted at the roots, and with sites occurring 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 4 shows the place and link graph of  $F$  — letting  $r$ 's and  $s$ 's denote roots and sites, respectively.

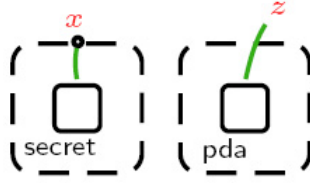


Figure 3:  $G$  — a bigraph that composes with  $F$

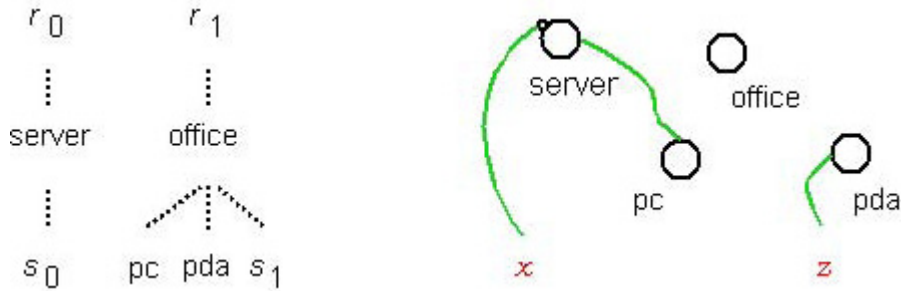


Figure 4: Place and link graph of  $F$

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 (viz. Definition 13 in Part II). 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. I discuss the motivation for introducing binding in Section 3.

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  $E = FG$  or  $E = F \circ G$ .

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  $Y$ ,  $\vec{X} = (X_0, \dots, X_{n-1})$  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 outeface) 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 outeface 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  $\langle 0, (\cdot), \emptyset \rangle \rightarrow \langle 2, (\{x\}, \emptyset), \{x, z\} \rangle$ .

We see that  $F$  and  $G$  compose, because the innerface of  $F$  is equal to the outeface 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  $I$ ,  $\text{id}_I : I \rightarrow I$ , maps  $n$  sites severally to  $n$  roots and map all inner names  $y \in Y$  to equal outer names<sup>1</sup>.

Both  $E$  and  $G$  have no sites or inner names, their innerface are  $\langle 0, (\cdot), \emptyset \rangle$ . 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. They correspond to closed terms (e.g., in term rewriting, see Section 6.3). To reflect this importance, we write the innerface  $\langle 0, (\cdot), \emptyset \rangle$  of agents simply as  $\epsilon$ ; and, in the rest of the report, agents are given names in lowercase.

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. A *parallel product*  $\parallel$ , which aliases common outer names, can be straightforwardly derived from  $\otimes$ .

In Parts II and III, we shall be particularly concerned with three classes of bigraphs, which prove important in a decompositional analysis of bigraphs: *Prime* bigraphs are those with only a single root, and only local inner names. For *discrete* bigraphs all links to a global outer name are one-one, while *name-discrete* bigraphs, are those where all links to both local and global outer names are one-one (refer to Definition 14 of II for the full definition of discreteness).

---

<sup>1</sup>Formally, a bigraph is defined as a pairing of a place graph and a link graph. Eliding some detail, taking interfaces as objects and bigraphs as morphisms we have a category of binding bigraphs. The identity bigraphs are the categorical identities. See the Appendix of Part II for full details.

We build bigraphical reactive systems (BRSs) by giving a set of reaction (or rewriting) rules; expressed essentially as a pair of bigraphs, such as those in Figure 5.

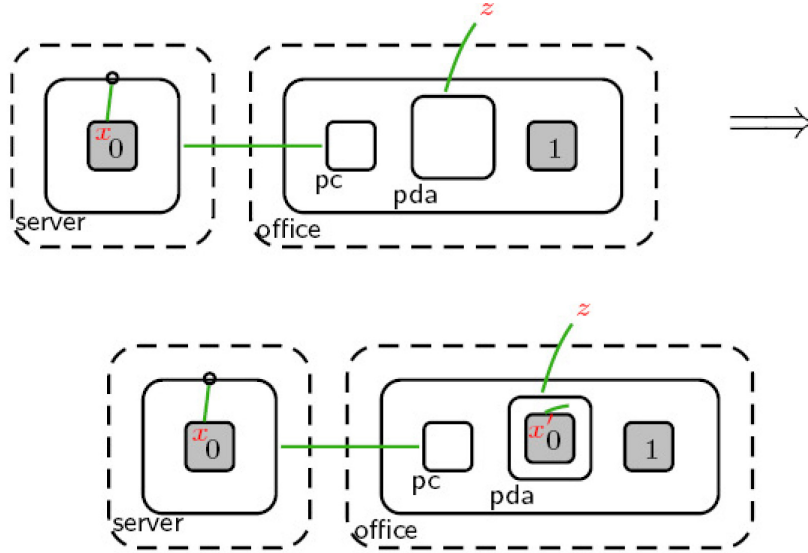


Figure 5: A bigraph reaction rule

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,

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

for a context  $C$  and a discrete parameter  $d^2$ . The bigraph  $\text{id}_Z$  is shorthand for the identity on the interface  $\langle 0, (), Z \rangle$ , and serves simply to export extra names  $Z$  of  $d$  “through” the redex.

<sup>2</sup>Formally, the context has to be *active* (see Section 2 of Part III).

The reaction rule in Figure 5 can be used to rewrite  $E$ , essentially with  $G$  as parameter and with the empty context<sup>3</sup>.

### 3 Completed Work

In this section, I give an overview of my work (see Parts II and III for detailed accounts). Moreover, I highlight those characteristics of bigraphs that support my course of investigation.

In the next section on proposed future work (Section 4), I discuss on-going and future work on a tool for bigraphs.

My main work has focused on establishing *syntactic* representations and associated theory. For binding bigraphs, I have developed

- a term language,
- a set of normal form theorems,
- an equational theory on terms that captures graph isomorphism on the term level, and
- a complete inductive characterization of matching (i.e., for describing when and where a redex matches an agent).

This work was developed together with co-authors (see Parts II and III) and extends the foundational work of Milner [Mil05], who gives a term language, a set of normal form theorems, and an associated equational theory for pure bigraphs.

The motivation for this work is two-fold: First, the work yields convenient syntax-based and inductive proof-techniques for reasoning about equality and matching in binding bigraphs. Second, in the BPL group, we are collectively working towards implementing 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. My work provides the necessary foundation for choosing an entirely syntactic representation of bigraphs. This is on-going work, and I give further details on this work in Section 5.

But why investigate syntax for an inherently graphical formalism? And why is binding important?

---

<sup>3</sup>This is only essentially correct. Choosing  $G$  as the parameter would require us to choose  $\text{id}_Z = \text{id}_{\{z\}}$ , but then  $R \otimes \text{id}_Z$  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 for Bigraphs** Graph isomorphism for bigraphs corresponds closely to *static* or *structural congruence* for process algebras for concurrent, distributed, and mobile systems. Structural congruence is sometimes thought of as simply a presentational convenience of the statement of the operational semantics; and sometimes as reflecting a notable difference of abstraction with regard to the dynamic part of the semantics. Choosing the latter view, we give a structural congruence relation to say that we intend no semantic difference between two terms  $a|b$  and  $b|a$ , for example. It is simply an artefact of our term-based representation of the process algebra that we can distinguish these two processes syntactically. It is a main virtue of encodings of process algebra into bigraphs that structural congruence resolves to graph isomorphism in the bigraphical encoding.

But an algebraic presentation also has certain conveniences. To motivate this for bigraphs, I have to explain in a little more detail than in Section 2.2, how bigraphs are formally defined (see the Appendix of Part II for full details). We define place and link graphs directly by giving the parent and link maps between collections of nodes, (hyper-)edges and names. To give the parent and link map, we assign nodes and edges concrete identities; hence, we call such place and link graphs *concrete*. We define a concrete bigraph as a pairing of a concrete place graph and a concrete link graph.

We are interested in the structure of a bigraph, though, not in the concrete identities of nodes and edges, they are simply necessary to state the underlying maps. Further, as an artefact of this manner of giving the link map, we can create concrete link graphs that contain unconnected edges (so-called *idle* edges). Hence, we define *lean-support equivalence*, denoted  $\simeq$ , such that  $G_0 \simeq G_1$  iff  $G_0$  and  $G_1$  differ only by a bijection between their nodes and non-idle edges; idle edges are disregarded entirely. We define *abstract* bigraphs, those bigraphs we are really interested in, as  $\simeq$ -equivalence classes of concrete bigraphs.

And now, rather than reasoning, for example, about equivalence of (abstract) bigraphs based on the representation sketched above, a term-based representation with an associated structural congruence allows us simple syntax-based equational reasoning.

Since the place graph of a bigraph is a forest it is possible to devise a fairly simple term language for bigraphs. General graphs, as used in graph transformation systems, do not share this property (see Section 6.4).

**Binding is Important for Modelling** Pure bigraphs aim to treat connectivity and locality orthogonally, and hence, do not include any notion of

scope or locality of linkage [Mil06b]. Binding introduces a mechanism that allows us to relate and constrain the two constituent structures.

Binding bigraphs add binding ports on nodes, an associated scope discipline, and localized names to transfer the scoping discipline across composition. Milner is currently working on formulating a variant of bigraphs, where names are allowed to be *multi*-located [Mil06a] (see Section 6.2).

Introducing locality and binding allows us to model both *copying* of linkage in a parameter, and *change* of linkage upon a parameter. Thus, binding and localized linkage appear naturally in modelling, as it allows us more control over parametric reaction. Binding is essential for the description of parametric reduction in, for example, the  $\pi$  and  $\lambda$  calculus. Milner and Jensen study an encoding of an asynchronous variant of the  $\pi$  calculus [JM04], while in recent work Bundgaard and Sassone extend this work to a typed polyadic variant [BS06]. In [Mil06a], Milner encodes and studies a  $\lambda$  calculus with explicit substitution (using intrinsically binding and multilocalized names — see Section 6.2).

In [DD05b], we give examples and an extended discussion on the modelling power of binding bigraphs based on a case-study of using binding bigraphs to model event-driven systems.

### 3.1 An Equational Theory for Binding Bigraphs

Together with coauthors, I give a term language for binding bigraphs, an associated axiomatization of graph isomorphism, 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 proves to require a reworking of the foundation of the term language, and, in particular, the normal form and equational theory.

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. 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 are simply composition and tensor product. The normal form for pure bigraphs is based on so-called *discrete* decomposition, decomposing a bigraph as a discrete bigraph composed with a *wiring* — a bigraph which contain only linkage. This essentially corresponds to a decomposition of the bigraph into its underlying place and link graph. This manner of decomposing a bigraph 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 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, support 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 the discrete decomposition provided by the normal form is also used intrinsically in the proof of completeness of the equational theory.

For binding bigraphs 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 is simply a refinement of pure bigraphs, in the sense that if we “forget” the binding of ports and locality of names, we get a valid pure bigraph<sup>4</sup>. We would expect a term language, a normal form and an equational theory that reflects this. In other words, we seek to develop a conservative extension of the syntactic 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, *concretion*, that maps local names to global names, and define a new combinator, *abstraction*, that converts global outer names to local outer names. Furthermore, to account for binding ports on nodes, we allow the *binding* ion to bind a set of (local) inner names to each of the binding ports of a node.

We develop a normal form, where binding is introduced as “outermost” as possible, and which is based centrally on *name*-discreteness (i.e., where all linkage to both global and local outer names is one-one). 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.

---

<sup>4</sup>Formally, there is an obvious forgetful functor from the category of binding bigraphs to the category of pure bigraphs.

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 nodes. In fact, the term language (with this variant of ion) is complete for binding bigraphs, and we could have easily defined a complete normal form. But we would not have been able to define linearity as a simple syntactic correspondent of name-discreteness.

In stating the equational theory, we extend the axioms for pure bigraphs with six axioms concerned with binding and local names, and 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. Incidentally, the proof of completeness in Part III 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.

### 3.2 An Inductive Characterization of Matching in Binding Bigraphs

As mentioned in Section 3, the definition of a binding bigraph is quite unwieldy for reasoning about equality. Furthermore, the definition of matching (as sketched in Section 2.2) tells us nothing about how to *construct* a context  $C$  and parameter  $d$ , given an agent  $a$  and a redex  $R$ . Hence, our approach is to initially *characterize* solutions to the matching problem, and later investigate how to *specialize* this into an algorithmic approach.

We already have an inductive way of building bigraphs as terms. Hence, it seems a valid approach to look for an *inductive* characterization of how to construct  $C$  and  $d$ , by induction on  $a$  and  $R$ , the input to the matching problem. We define the concept of a *matching sentence*, which is a new representation of a match. A matching sentence defines a *relation* on components of the agent, redex, context, and parameter. We look to characterize this relation inductively. Matching sentences are defined such that valid sentences correspond to valid matches.

Our main result is a set of operational rules that suffice for deriving all valid sentences. The extended version of the original paper in Part III includes an appendix with extensive details on the proof of completeness.

Our results immediately yield an inductive manner of reasoning about matching in binding bigraphs. But combined with our earlier work, we have also established a foundation for basing an *implementation* of bigraphs and BRSs directly on terms. This would allow us to base such an implementation on provably complete representations and on provably correct algorithms. The work on such an implementation is currently underway (see Section 5).

This concludes the overview of my completed work, which is described in full detail in Parts II and III. In the following section, I outline my proposal for future work, and give an overview of some related fields, results, and techniques.

## 4 Proposed Work

In this section, I propose future work. Several lines of work involve the utilization or adaptation of results, techniques, or tools from other studies on bigraphs or studies in other fields. To explain these lines of work in greater detail, I first need to introduce these results, techniques or tools.

In Section 4.1, I give an overview of the proposed work. In Section 5, I outline the on-going work on a tool for BRSs. In Section 6, I give an overview of a number of studies and fields that are important to consider for future work, and provide more detail on the lines of work that involve these.

### 4.1 Overview

The following proposals are listed in order of interest. In particular, I find that the first three are the most important to consider at present. Hence, these are also the most detailed proposals (although with some details postponed for later sections).

**A Prototype Tool for BRSs** As mentioned in Section 3, we are currently working on a prototype of a tool for BRSs. This work is based directly on the work on terms for (binding) bigraphs, as sketched in Sections 3.1 and 3.2, and detailed in Parts II and III.

The results on syntactic theories yields provably correct representations and algorithms, but leaves ample room for efficiency-improvements. Centrally, there is a need to investigate techniques for intelligently combining matching and rewriting. At the least, this requires work on a revised normal form, more suited for efficient matching and rewriting.

In Section 5, I provide more details on our current work on a tool.

### **Investigation and Adaptation of Related Tools and Techniques**

The work on bigraphs and BRSs stem from an effort, by Milner and coworkers, to provide a general theory able to unify and represent many behavioral aspects of mobile process calculi. Consequentially, there has been quite a lot of work on recapturing aspects of mobile process calculi. But exactly because bigraphs and BRSs has been developed as a meta-calculus, the manner in which we model in BRSs does not resemble the manner in which we model in mobile process calculi.

From a modelling perspective, the theory is more closely related to *term rewriting* and *graph transformation*. There is a lot of experience in these fields, both on theory and on practical applications. However, links to these fields have so far been less investigated. These fields are important to understand from an implementation-perspective, though, as their modelling approach (tree and graph based, respectively) and reconfiguration mechanics (rule-based) are similar to BRSs. Hence, it appears that we may adapt in our work on a tool for BRSs, tools and techniques developed for term rewriting and graph transformation.

In Sections 6.3 and 6.4, I initiate an investigation of term rewriting and graph transformation. To understand how bigraphs and BRSs relate to these fields, we need to understand the fields in greater detail. The section also gives pointers to tools and techniques that could be the starting points of further investigation. Section 6.1 expands on the discussion of relatedness with term rewriting and graph transformation, thus providing a motivation for the focus of the survey.

**Simulating BRSs with Graph Transformation Systems** Considering the long history of work on general graph transformation [EEPT06], it is natural to ask whether we could, for example, use a graph transformation tool as a basis for implementing BRSs. This would probably preclude us from utilizing much of the work based on syntactic representations, though. Thus, it would require us to establish other methods for ensuring validity. It is still important to consider, though, as it would allow us to reuse a number of existing resources. Consequentially, I have initiated an investigation of how to *compile* a BRS to a (standard) graph transformation system. I expect to be able to establish a (weak) dynamic correspondence between the reactive systems.

I comment on this, and give indications of the difficulties involved, in

the final section on graph transformation (Section 6.4.3).

**Bigraphs with Multilocal Names** The final word on *how* to introduce binding into bigraphs has yet to be said. As mentioned, Milner is working on formulating a variant of bigraphs with multilocal names. It seems straightforward to extend my work on an equational theory and on matching to multilocal names; the complexity lies in introducing binding in the first place. Hence, this result seems fairly easy to establish, but we need more experience with concrete examples, for me to determine whether this line of work is worth the effort.

In Section 6.2, I give more details on the work on, and motivation for, bigraphs with multilocal names.

**Spatial Logics for Binding Bigraphs** 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, a framework for spatial logics for bigraphical structure [CMS05b]. Our results extend this foundation to binding bigraphs, and thus yields an obvious line of future work, namely to extend BiLog to binding bigraphs. Further, currently BiLog only allows us to express static properties. It will also be important to work toward extending BiLog with dynamic reasoning capabilities.

This line of work is important — as we hope to include tool-support for reasoning about systems — but one could hope that the authors of [CMS05b] continue this work themselves.

In Section 6.2, I give a brief overview of the present state of BiLog, and comments on possible extensions.

**Tool-support for Sortings and Spatial Logics** We also plan to give tool-support for some of the important concepts and techniques that is being developed for bigraphs, highlighted in Section 6.2. For instance, the theory of *sortings* expands on the aim of being able to constrain and control the structure of bigraphs. *Spatial logics* promise to lend us a language for expressing properties of bigraphs and bigraphical reactive systems. The inclusion of tool-support for either sortings or logics requires a considerable amount of work. For sortings, this would at the least require work on how to express sortings; but also on how to intelligently combine sorting with matching and rewriting. For spatial logics, this initially requires the extension of the theory to binding bigraphs (see above). Secondly, one would

most probably need to determine an interesting subset of the logic, which it would be feasible to implement tool-support for checking.

In Section 6.2, I give an overview on the work on sortings and spatial logics.

**Investigating the Categorical Foundation of Matching** Finally, another possible line of work is based on the resemblance between the algebraic presentation of bigraphs and *term graph* rewriting systems.

As was kindly pointed out to me recently by A. Corradini, the algebraic presentation of bigraphical structure in [Mil05] and Part II resembles the algebraic presentation of term graphs in [CG99a]. 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. This line of work seems less applicable for a tool for BRSs, though, so I shall give this lower priority.

In Section 6.4.1, I discuss term graph rewriting systems and provide a few more details on the work mentioned above.

This concludes the brief overview of proposed future work. In the following sections, I provide more details for each of these proposals.

## 5 A Tool for BRSs

It is a main aim of the BPL project to build tool-support for working with, and reasoning about, BRSs. Currently, in the BPL group, we are collectively working towards that aim by implementing a prototype rewriting engine for binding bigraphs, based directly on the work on term languages for binding bigraphs in Parts II and III.

The term language for binding bigraphs lends 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 gives us an inductive method of proving a match valid. But it can also be used as a foundation for implementing matching in a tool for BRSs. As I shall sketch below, I am currently working on *normal* inferences for matching on *terms*. (Note that the characterization given in Part III is for bigraphs, not terms for bigraphs.)

First, to recast the inference rules to work on terms, I simply add a single rule to allow application of structural congruence. This yields a wildly

nondeterministic inference system as we might need to apply structural congruence between every step to infer a match. (This is reminiscent of problems in implementing *rewriting logic*, term rewriting modulo a set of static equivalences, see Section 6.3.1 for more details.)

Consequentially, as the second step, to specialize the characterization into an algorithm for mechanically *finding* matches, I 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, I define normal inferences that require each inference to start by rewriting the term to be on normal form.

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.

Thus, our implementation is theoretically well-founded, but, alas, not necessarily very efficient. Therefore it should be noted that currently we are considering efficiency-issues only at the level of being able to run minimal examples.

Plans for future work on the tool include major revisions on matching and rewriting, in particular on techniques for intelligently combining these. Also, as suggested by [DD05a], it might be worthwhile to consider approaches where matching is driven even more aggressively by the place graph. (In *loc.cit.*, we show that for pure bigraphs with epi redices, given a match in the place graph there is at most one compatible link graph match.)

Future aims also include support for *sortings*, *spatial logics*, and *multi-located names* as detailed in Section 4.1.

## 6 Litterature Study for Proposed Work

This section contains a litterature study fields, results, and techniques, which will be the basis for further investigation. It also provides a background for a number of my proposed lines of work (see Section 4.1), and discusses these in greater detail.

In Section 6.1 I discuss the motivation for focusing the main parts of this study on term rewriting and graph transformation. In Section 6.2, I give an overview of studies and results for bigraphs, which are important for my further studies. In Sections 6.3 and 6.4 I give brief surveys of term rewriting

and graph transformation, respectively.

I emphasize that these surveys are preliminary, and that the areas and projects highlighted here will serve as the basis for further studies. To determine links and possible adaptation between bigraphical theory and other fields, we still need a deeper understanding of bigraphs and BRSs as a modelling tool. To that end, it will be important to develop more worked-out and well-understood examples. I hope that tool-support, as promised by the current work on a prototype implementation, shall be a central aid in doing so.

## 6.1 Background

The theory of bigraphs and BRSs stem from work on deriving labelled transition systems (LTSs) for reactive systems [Sew02, LM00, JM03], and on 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]).

Consequentially, there has been quite a lot of work on recapturing aspects of mobile process calculi; I highlight some examples in Section 6.2. And significantly, in that sense, bigraphs and BRSs have been designed inherently by the needs of a meta-calculus. Whether one investigates BRSs as the foundation for representing and capturing semantics of mobile process calculi, or, whether one seeks to model *directly* aspects of ubiquitous systems, this design of bigraphs and BRSs is important. Centrally, it has consequences for the modelling primitives and mechanisms we are given to work with. As I try to illustrate in Sections 6.3 and 6.4, from a modelling perspective bigraphs and BRSs have much in common with term rewriting and graph transformation. More so, in fact, than with specific mobile process calculi.

When modelling in BRSs, we get to give 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. First, we define our own set of primitives and operators, and then we construct the reconfiguration patterns for these using reaction rules. Compare this, for example, with the graph transformation language PROGRES, which can be seen as a specification-language for both statics and dynamics of domain-specific databases (see Section 6.4.2). And contrast this with mobile process calculi, which almost all derive from the (CSP, CCS, and ACP) tradition of process algebra for modelling and

reasoning about concurrent systems [Bae05]. In a process algebra, we are given a proto-programming language with a fixed set of primitives and operators to model our data-structures, and use the hard-wired semantics of that particular calculus to model our intended behavior.

The differences in *how* we model are particularly important for our work on a tool for BRSs. It is not easily seen how we may directly employ experiences with the implementation of tools or programming languages based on mobile process calculi.

Prominent tools such as the Concurrency Workbench [CWB, CPS93], and the Mobility Workbench [MWB, Vic94] are concerned with efficiently representing CCS and the  $\pi$ -calculus, respectively, and, particularly on analyzing and verifying bisimilarity based on that representation. In the BPL project, we are currently concerned with representing and simulating a reactive system, which is given by a signature and a set of rewrite-rules that are defined by the user. The associated bisimulation relation is generated semi-automatically (something which we might hope to give partial tool-support for in the future), but, as sketched in Section 6.2, the techniques that need to be employed for tailoring the bisimulation are still very much in development.

Because of the likeness of the modelling mechanisms of BRSs and graph and term rewriting, I conclude, that when looking to employ techniques and tools inspired by work in other fields, we should currently look towards those developed for term and graph rewriting.

As a final side-remark; note, that there are 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. A common ground may be found, perhaps using a 2-categorical approach as used to represent graph rewriting in [GHL99] or, for deriving labels for reactive systems [SS02]. Furthermore, recent work on deriving labels and capturing observational equivalences in the standard DPO approach to graph transformation is directly inspired by the work on deriving labelled transitions systems for BRSs [EK04].

## 6.2 Related Work on Bigraphs

An evaluation of the aim of using bigraphs for representing mobile ubiquitous systems, was initiated in [BDE<sup>+</sup>06] and continued in [Els06] by Elsborg, who defines and models as BRSs so-called Plato-graphical models, in the

process encoding and analysing 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.

Case-studies of bigraphs, such as those above, drive the need for development of novel theory and techniques.

**Bigraphs with Multilocal Names** I have already discussed the motivation for introducing binding and local names in Section 3.

Milner is currently investigating extensions of binding bigraphs, where names are allowed to be multilocal, that is, located at several roots or several sites (see, e.g., [Mil04] and [Mil06a]). The on-going work in [Mil06a] is an investigation of confluence properties of BRSs, by studying an encoding of a  $\lambda$ -calculus with explicit substitutions.

The research on extensions of binding bigraphs with a more fine-grained control of the locality of names, is motivated by the loss of ability to write convenient wide reaction rules working on bound linkage. In binding bigraphs, names shared across roots in wide redices or reactums of rules must be global. When trying to find a match between a redex and an agent, global names in the redex cannot end up being bound in the context, as this would require the context bigraph to link a global inner name to a binding port or a local outer name; thus, breaking the scope rule. Consequentially, in binding bigraphs, we cannot write wide rules working on bound linkage without explicitly showing the binder in the rule. Essentially, bigraphs with multilocal names add this feature, by allowing shared names to be local. Centrally, the encoding of the  $\lambda$ -calculus of [Mil06a] uses this feature to encode succinctly non-local substitution.

**Sortings** The theory of *sortings* (or *typings*) for bigraphs and general reactive systems have been extended in [BDH06], and applied to the domain of context-aware systems in [BDE<sup>+</sup>06]. In particular, a framework is established for so-called *decomposable predicate* sortings, those sortings given by a predicate  $P$  preserved under decomposition<sup>5</sup>. As locality and binding, sortings allow us to prune the bigraphs we consider, simply by asserting that we only consider our reactive system to contain those morphisms (i.e., bigraphs)

---

<sup>5</sup>Strictly, the results in [BDH06] are for reactive systems over categories, thus not directly applicable to the precategory setting of bigraphs. The authors state, though, that “We believe the extension of our work [to reactive systems over precategories] to be straightforward but cumbersome, but we have yet to justify that belief.”

that adhere to the sorting discipline. We could, for example, disallow all bigraphs with nodes of control  $K$  in parallel (such as  $K \mid K$ ). In [BDE<sup>+</sup>06] 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, for example, requiring that no  $B$ s occur in the context of a reaction.

Variants of sortings have been applied extensively for pruning the derived labelled transition systems yielded by encoding particular calculi in bigraphs to recapture in the encoding the contextual equivalence of the original calculi (for CCS [Mil06b], typed  $\pi$  [BS06], Homer [BH06], and Mobile Ambients [Jen06]). Not surprisingly, the scoping restrictions imposed by the above-mentioned binding variants of bigraphs can be expressed as sortings. (The definition of reaction is also changed for binding bigraphs, though — see [JM04, Chapter 12] or Part III — in particular, to allow parameters with arbitrary bound wiring. This cannot be expressed directly as a sorting.)

From an implementation point-of-view, sorting is an important future topic to consider; most importantly, as it has been a necessary component in constraining models to include less “junk”, and, as such it promises to simplify the matching problem (as we need only consider well-sorted matches).

**Spatial Logics** In [CMS05b, CMS05a] the authors 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 behavioral 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. Similar to the *active/passive* kinding of controls, we define certain nodes of certain controls as opaque, blocking inspection of a term at these nodes and their descendants.

Recent, yet unpublished, work [CMS06], delve more into exemplifying instantiations of the logic, and discussing how the framework may be extended to encompass also dynamic behavior. Interestingly, they show 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 characterize 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]).

Both static logics and dynamic logics are of major future importance for stating properties and reasoning about bigraphs and BRSs. An obvious line of future work would be to utilize the work on binding bigraphs 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. 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 Hennessy-Milner logical characterization of bisimilarity for the  $\pi$ -calculus. For that, it seems, that the intensionality of the BiLog framework at present is too pronounced. More experience is needed to evaluate this, though, and also to determine whether and how we might hope to include tool-support for some of the features presented by a logic.

### 6.3 Term Rewriting

There is a long tradition of *term rewriting* systems and a comprehensive literature on the subject; two standard references to the field are [BN98], and more recently [TeR03]. Let us briefly recall the basic theory of term rewriting systems, and discuss the relation to bigraphs and bigraphical reactive systems.

A term rewriting system (TRS) is a collection of *rewrite rules* on terms. We construct terms, as usual, inductively over sets of function symbols and variables. Rewrite rules are pairs of terms  $L \rightarrow R$ , such that all variables in  $R$  also occur in  $L$ . TRS's can be used to rewrite on closed terms (i.e., terms with no variables)<sup>6</sup>.

We determine whether we can rewrite a term  $t$  by a TRS  $\mathcal{T}$  of rules  $L_i \rightarrow R_i$  in the following manner: We try to find one or more *matches* between left-hand sides  $L_i \in \mathcal{T}$  of each rule and  $t$ . We have a match iff for a substitution  $\sigma$  on the variables of  $L_i$  and a subterm  $t'$  of  $t$ , we have

---

<sup>6</sup>Rewriting on open terms (i.e., terms with free variables) can also be considered, and are called *narrowing* rewrite systems. The name refers to the narrowing (or, instantiation) of the variables in the open term being rewritten on, when we match a left-hand side of a rule to the term. Such systems have been studied in the context of logical programming languages. Rewriting in BRSs works on agents, corresponding to closed terms, so, at present, we shall not be particularly concerned with this kind of TRSs.

$\sigma(L_i) = t'$ ; or in words, iff a subterm of  $t$  matches a substitution-instance of  $L_i$ . Such a subterm  $t'$  is called a *redex* (conflicting somewhat with the terminology in BRSs where the left-hand side itself is called a redex). We rewrite (or reduce) a redex by replacing  $t' = \sigma(L_i)$  with  $\sigma(R_i)$ . A term  $t$  is said to be in *normal form* with respect to a given TRS  $\mathcal{T}$ , if no rule in  $\mathcal{T}$  can be used to rewrite  $t$ . Again, the terminology for bigraphs might confuse slightly. In this report we refer mainly to a *static* normal form (as given by Theorem 1 in Part II). This is an important notion for bigraphs, because bigraph terms are considered up to structural congruence. On the contrary, for BRSs a normal form with respect to the reduction rules (a *dynamic* normal form) is less important to consider, since we are interested in reactive systems, which, in general, we do not wish to consider as reaching an end-state.

It is typical to require of left-hand sides  $L$  that they cannot be only a variable, thus disallowing arbitrary introduction of  $R$  structure everywhere; and that they cannot be *comparing*, i.e., contain multiple occurrences of the same variable.

Term rewriting systems have been studied as a vehicle to giving semantics to several kinds of programming languages, and there have been a natural interest in investigating confluence properties of term rewriting systems. Recall, that a rewriting system is confluent iff for every two reduction sequences  $t \rightarrow^* t_1$  and  $t \rightarrow^* t_2$ , there exists a  $t_3$ , s.t.  $t_1 \rightarrow^* t_3$  and  $t_2 \rightarrow^* t_3$ . In general, TRSs are not confluent, but confluent sub-classes have been identified. For instance, a TRS is confluent if it is non-comparing, and *non-ambiguous* (i.e., all left-hand sides are non-overlapping).

As a natural extension, different kinds of *reduction strategies* have been studied. Reduction strategies are functions that, given a TRS and a term, determines the redex to reduce next. Of special interest here (see below) are *parallel* reduction strategies, which have the possibility of determining two or more redices to be reduced next, in arbitrary order. The parallelism allowed by this strategy corresponds to an interleaving view of concurrent execution.

### 6.3.1 Comparing TRSs and BRSs

The basic setup of TRSs bears a close likeness with *place*-graph reactive systems; essentially, collections of parametric reaction rules reconfiguring tree-shaped structure with node controls corresponding to function symbols, and (numbered) sites corresponding to (named) variables. There are a few important differences, though. Recall that we consider place graphs equal

up to permutation of siblings, which corresponds to considering terms for place graphs equal up to structural congruence (as determined by the categorical and place graph axioms, viz. Part II). Terms of TRSs are naturally considered equal iff they are syntactically equal. Further, we are allowed to write *wide* rules working on place graphs. The latter allows us to model directly *distributed* reaction, which is more than a parallel reduction strategy allows us. Thus, we can model elegantly non-local moves and copying, as we do not need to show explicitly the parent structure of the subtrees matched by the patterns in the left-hand side.

The first point is a basis for studying more closely the relationship between bigraph reactive systems and term rewriting systems modulo a set of (static) equivalences. This is essentially what the field of *rewriting logic* takes as its starting point. There is a considerable amount of studies on rewriting logic, and, in particular, also as semantic models for concurrent systems; see, for example, [Mes96]. In particular, we could learn from studies on efficient implementation strategies for pattern matching and rewriting over associative and commutative theories. One approach, at least, is to employ the Knuth-Bendix completion algorithm [EB70] to transform the set of equations (over terms) into a confluent term rewriting system. This is as opposed to, for example, employing a (static) normalization strategy, as we are currently pursuing in the prototype of the implementation of the BPL tool. Further studies are surely needed to determine whether and how, we employ some of the sophisticated techniques developed in the context of rewriting logic.

A good concrete starting point for further studies is the system Maude [Mau, CDE<sup>+</sup>01, CDE<sup>+</sup>03]. It is supported by a strong research background on rewriting logic, in particular, a number of more recent studies by Eker on efficient AC matching and rewriting [Eke00, Eke03]. Furthermore, an extension, Mobile Maude [DELM00], supports mobile computation.

Last, but not least, an important derivative of TRSs are term *graph* rewriting systems: essentially, lifting TRSs to reduce under a graph transformation semantics. There are some links to the work on normal forms and an equational theory for bigraphs, and I shall return to term graph rewriting systems in the next section, after having introduced graph transformations systems.

## 6.4 Graph Transformation

The studies of *graph*, rather than term, reconfiguration also has a long history. There are several approaches, but the longest running tradition

is the *algebraic* approach based on double-pushout (DPO) constructions initiated by Ehrig, Pfender and Schneider [EPS73]. For a recent introduction, see [EEPT06]; for the comprehensive standard reference to the field, see the handbooks [Roz97, REKE99, REKM99]). The name *algebraic* stems from the fact, that a directed graph  $G = (V, E, s, t)$  (where  $s : E \rightarrow V$  and  $t : E \rightarrow V$  are functions assigning source and target nodes for edges) can be seen as a special case of an algebra with two base sets  $V, E$  and operations  $s, t$ . Equally, graph homomorphisms  $f : G_1 \rightarrow G_2$  are constructed pointwise, and are special cases of algebra homomorphisms  $f = (f_V : V_1 \rightarrow V_2, f_E : E_1, E_2)$ .

A graph transformation system (GTS) consists of a set of graph transformation rules working on directed graphs. As for term rewriting systems, a rule (or production)  $p$  is essentially a pair of graphs, a left-hand side  $L$  and a right-hand side  $R$ . Rules are given with a common *interface*  $K$ , the intersection of  $L$  and  $R$ ; explicitly stating the nodes and edges which are preserved when rewriting by  $p$ . Formally,  $p = (L, K, R)$ , where  $L \cup K = L$  and  $R \cup K = R$ .

As the name double-pushout suggests, the theory has a categorical presentation. We briefly recap some of the basic definitions and extensions. The class of all graphs as objects and of (total) graph homomorphisms form a category **Graphs**. Categorically, a rule  $p = (L \xleftarrow{l} K \xrightarrow{r} R)$  consists of graphs  $L, K, R$  and mono (injective) morphisms  $l, r$ . A (direct) graph transformation  $G \xrightarrow{p, m} H$  from a graph  $G$  to a graph  $H$  by  $p$  and a *match* morphism  $m : L \rightarrow G$ , is given by the following DPO diagram,

$$\begin{array}{ccccc}
 L & \xleftarrow{l} & K & \xrightarrow{r} & R \\
 \downarrow m & & \downarrow k & & \downarrow n \\
 G & \xleftarrow{f} & D & \xrightarrow{g} & H
 \end{array}$$

where both squares are pushouts. The match morphism  $m$  serves to identify 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  $K$  (giving us the graph  $D$ ), and to construct  $H$ , we add the nodes and edges in  $R \setminus K$ . It is a standard restriction to require that the match morphism  $m$  be mono; thus, for example, disallowing the match morphism  $m$  to identify two nodes of  $L$  with a single node in  $G$ .

As opposed to term rewriting systems (TRSs) and BRSs, graph transformation systems do *not* naturally allow *parametric* reconfiguration rules. Our understanding of parameters in TRSs and BRSs, naturally rely on the hierarchical structure of terms and the place graph, respectively. In contrast, a priori, we have way to give meaning to *parameters* for arbitrary graphs. Consequentially, we cannot express directly in rules, copying and deletion of arbitrary subgraphs in (basic) GTSs. Not surprisingly, though, the notion of parametric reconfiguration appear in extensions to the basic theory for GTSs. In the next section, I give particular mention to one, *hierarchical* graph transformation, and we shall encounter the problem again in the context of the concurrent version of the programming language *Clean* (see Section 6.4.1), which is implemented using *term graph rewriting*.

Finally, it should be noted, that as for term rewriting systems, confluence properties of GTSs have been studied in great detail, characterizing precisely when direct graph transformations can be applied in arbitrary order, leading to the same result. This leads naturally to a notion of parallelism, that corresponds to that allowed by a *parallel reduction strategy* for term rewriting systems (see Section 6.3).

#### 6.4.1 Extensions

There exists numerous basic extensions to the theory, adding for example, *types*, *application conditions*, and *layered rules*. The algebraic approach to transformation of graphs generalizes to a theory of so-called *high-level replacement* (HLR) systems, which permits reconfiguration of more complex structures such as, for example, Petri nets, algebraic specifications, and hierarchical graphs (the latter, which we shall return to below). The theory of HLR systems is supported by recent work on adhesive categories by Lack and Sobocinski [LS04], and can be instantiated, for example, to typed *attributed* graph transformation systems. As the development of such extensions have been driven by the needs of concrete models and from practical experience with tools, and, as such, are of central importance to our project, I outline the properties of the extensions mentioned.

Transformation of *typed* graphs can be considered; first, by giving a particular *type* graph  $TG$ ; second, by considering all graphs  $G$  together with a *type* morphism  $type = (type_V, type_E) : G \rightarrow TG$ , which relates nodes of  $G$  to type-nodes of  $TG$  and edges of  $G$  to type-edges of  $TG$ ; and finally, by restating definitions and propositions (as sketched above), for the category **Graphs<sub>TG</sub>** of typed graphs over  $TG$ . Labelled graphs fall out as a special case of typed graphs.

*Application conditions* for matches restrict the cases where the left-hand side of a rule matches a graph. Of practical interest is, in particular, *negative* application conditions (NACs), which allow us to display graph configurations that we wish to *prevent* from transforming.

*Layered* rules are simply distributed into a sequence of layers. In each layer, rules are applied until we reach a graph which cannot be reconfigured more. Finally, *attributed* graphs allows us to attach attributes over an arbitrary algebra to nodes and edges. Formally, this requires another extension of the theoretical underpinnings into the category  $\mathbf{AGraphs}_{ATG}$  of attributed graphs over an *attributed* type graph  $ATG$ , which is then used to instantiate the general theory developed for HLR systems. On a more practical note, and as an interesting and non-trivial example of the utilization of attributes, the AGG tool (see Section 6.4.2) allows us to attribute graphs with Java objects and types.

**Hierarchical graph transformation** As we are concerned with bigraphs, another notable extension is the work on *hierarchical* graph transformation. Several approaches have been investigated; an interesting approach is [DHP02], which smoothly extends the DPO approach for transformation from flat to hierarchical graphs, which allow recursive nesting of graphs in special hyperedges. The work is motivated by practical applications for programming and system specification, which the authors claim should benefit from a notion of hierarchy in the structure. In particular, the manner in which hierarchy is introduced, lends itself easily to a mechanism for hiding or abstracting from subgraphs; a property, which is important for the visualization of large graphs. As for bigraphs, the idea is to add this notion *explicitly* in the model, rather than encode (and maintain) the hierarchy in (flat) graphs. Further, the notion of hierarchical transformation supports an introduction of rule *schemas* with *frame* variables, which can be instantiated with graphs, to allow directly copying and deletion of subgraphs. As opposed to bigraphs, though, edges cannot cross boundaries of frames.

Other approaches, such as [Pal04], work by imposing a nesting relation directly on components of a (flat) graph; we might, for example, allow nesting of the nodes (like for bigraphs). The framework in *loc.cit.* is also founded on the DPO approach to transformation, and care is taken to allow flexibility in allowing the framework to be specializable to particular subclasses such as, for example, node-nesting or (hyper-)edge nesting. The framework immediately 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 these, or other notions of hierarchical graph transformations could be employed to strengthen the relation between GTSs and BRSs.

**Term graph rewriting** Term graph rewriting is not really an extension, but an entire subfield of research with 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. A standard approach is to lift terms by interpreting their tree shape as a graph, with atoms as labelled nodes, variables as unlabelled nodes, and edges representing the nesting relation — making sure to link multiple occurrences of the same variable to the *same* node. Thus, the natural term graph rewriting system is really a term *dag* rewriting system. When applying rules, with terms lifted in this manner, it naturally results in *sharing*, rather than copying, of subterms, something which cannot be expressed in basic TRSs.

Both term rewriting and term graph rewriting systems have been studied as the basis for efficient implementations of the  $\lambda$ -calculus. The utilization of sharing allows very efficient implementations of functional programming languages. The book [PE93] introduces the fields of term, graph and term graph rewriting systems with the specific aim of introducing *Clean*, a purely functional language. For our purposes, *Concurrent Clean* [Cle, NSvP91, Pv98], the extension to concurrent distributed domains, appears particularly interesting. It is a natural choice to seek maximal sharing when concerned with an efficient implementation of the  $\lambda$ -calculus, but in a distributed, mobile setting one sometimes need to actually copy and move objects. Thus, as a pragmatic solution Concurrent Clean simply introduces a separate construct to *copy*, rather than share, terms. We might perhaps need to look to such solutions, if the need arises for introducing sharing and dag-place graph structure.

Another interesting point, is the relatedness between the algebraic presentation of bigraphical structure in [Mil05] and Part II (i.e.,[DB06]), and the algebraic presentation of term graphs in [CG99a]. Based on an analysis of atomic term graphs and normal forms, a categorical foundation for term graphs is established, showing that term graphs over a signature  $\Sigma$  are in one-one correspondence with the morphisms of the free so-called *graph-substitution*-monoidal category generated by  $\Sigma$ . In the paper [CG99b], this treatment of term graphs is used as the basis of a 2-categorical description of rewriting. As suggested to me, it might make sense to take this work as an inspiration to start looking for a stronger categorical foundation for the

description of matching and rewriting in bigraphs. This is surely a topic for future consideration; it seems, initially, that this would lead in the same general direction as that taken by the 2-categorical approach to derive labels for reactive systems, as described in [SS02].

#### 6.4.2 Applications and Tools

Graph transformation is a field which has been driven by a number of applications. These applications have consequently driven the development of tools for graph transformations. Applications with associated implementations include semantics for functional languages such as Clean [NSvP91], verification of object-oriented 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.

There exists a number of general graph transformation tools. Some of the techniques employed for the implementation of these might inspire future work on a dedicated tool for BRSs; or they might be employed directly for simulating BRSs. As examples of such tools, I highlight two below, which constitute quite different approaches to employing graph transformation.

**The AGG System** The AGG (Attributed Graph Grammar) System [AGG, ERT99], developed at the Technical University of Berlin, is at the core a rewriting engine for typed attributed graph systems using negative application conditions. The modelling approach and implementation follow quite closely the categorical foundation as described above<sup>7</sup>, seeking to provide a general framework for working with graph transformation. Thus, the AGG engine utilizes a library for computing *colimits* efficiently (recall that a pushout is a just a special case of a colimit), developed in [Wol98]. The matching engine underlying AGG recasts graph pattern matching as a constraint satisfaction problem (CSP). In [Rud00], Rudolf shows a technique for obtaining an equivalent CSP for a given graph matching problem. The main idea to take the nodes and edges of the pattern (left-hand side) graph to be the variables of the CSP, to take the nodes and edges of the working graph as the domain of these variables, and to establish a translation of the restrictions that apply for graph maps to be a homomorphism into a set of constraints for the CSP. It would be interesting to investigate whether this approach would work for bigraphs.

---

<sup>7</sup>Strictly, rule application is computed using the so-called *single-pushout* approach, rather than the double-pushout approach.

AGG is implemented entirely in Java, and (as mentioned) supports attributing nodes and edges with Java objects and types; further, rules may also be attributed with attributes that are evaluated during rule application. This enables AGG to be used either through its own visual editing interface; or to allow (external) applications to employ the transformation engine in AGG. One such application is the Tiger project (Transformation-based generation of modeling environments) [Tig, EEHT05] a tool, also developed at Berlin, that allows the generation of visual editor plugins for the Eclipse tool [Ecl] from graph-grammar specifications of a visual language.

**The PROGRES Tool** PROGRES (PROgrammed Graph Rewriting Systems) [PRO, SWZ99] has a longer history than AGG, and, is actually more a full programming language based on graph transformation — but can also be viewed as a specification-language for domain-specific databases determining both static structure and dynamic behavior. It includes a graphical interactive environment such as AGG. It is, as the name hints at, engineered towards a *programmed* approach to using graph transformation as a means of both modelling and actual programming. Thus, it presents the user with more control structures for controlling rewriting, and the modelling environment allows more powerful modelling mechanics than the rule-based rewriting of typed, attributed nodes and edge allowed by AGG. We can establish *inheritance hierarchies* of node classes (even allowing multiple inheritance), *derived* relationships — representing often-used paths of a given graph, UML-like *multiplicity constraints* for edges, write functions that test attribute values under pattern matching, and centrally we are allowed to define complex queries and procedures that may update data. As graph transformation is employed centrally in running these queries, and following the programming approach to graph transformation, it is natural to allow so-called *parameter-passing* of nodes between rules. Under consecutive rule application, parameter-passing allows pattern matching to speed up in the subsequent rewriting steps, as it can directly reuse nodes passed as parameters rather than recalculating them.

The parameter-passing approach is further supported by the pattern matching algorithm, as detailed in [Zün94]. The heuristic algorithm detailed here solves graph pattern matching directly, as opposed to the CSP-approach for AGG. In doing this, a central problem is to choose the right *search plan* for comparing the components of a pattern graph with the components of a working graph. Interestingly, during compilation the pattern graph is transformed (using graph transformation(!)) to a graph where the

operations of all possible search plans are represented explicitly, together with a representation of the original components of the pattern graph. The algorithm then uses a quite sophisticated cost model for the basic operations in performing pattern matching (such as testing an attribute, traversing an edge, or enumerating all nodes of a specific type). Based on this cost model, at compile-time a greedy algorithm generates search plans for the different rules.

Based on a recent survey [VSV05], benchmarking different tools based on graph transformation, the techniques employed in PROGRES are more efficient in practice than those in AGG. Recent work [VVF05] on graph-pattern matching suggests incorporating (at compile-time) domain-specific knowledge into the technique for generating search plans, allowing statistical knowledge provided, for example, by a domain expert, to influence the cost model. This may even be employed at run-time to influence dynamically the cost-model.

#### 6.4.3 Simulating BRSs with Graph Transformation Systems

As graph transformation talks about transformation of structure at a very general level, it is natural to ask whether we could use GTSs as a vehicle for implementing BRSs; essentially, whether we may *compile* a BRS to a GTS establishing a suitable (weak) dynamic correspondence between the systems. This is important to consider as it would allow us to reuse existing technology developed for GTSs.

Consequentially, I have initiated and am currently working on such an investigation. More specifically, I investigate how we may faithfully encode a *particular* BRS into a (standard) typed, attributed GTS. I define a translation of BRSs simply by translating each of the constituents of a BRS — an agent, a set of reaction rules, and a signature. Towards fulfilling the practical goals of this investigation, I have chosen to restrict myself to work with the set of GTS concepts that are present in the tool AGG. The main goal is to establish a dynamic correspondence between the reaction semantics of a (possibly suitably restricted) set of BRSs and their translations into GTSs.

It is a strong aim to preserve graph isomorphism of bigraphs as graph isomorphism in the translated graph. As argued in Section 3, this has traditionally been noted as one of the main advantages of encodings of several process calculi as both BRSs and GTSs, and we would not easily want to let go of this feature. The lack of parametric reconfiguration in basic GTSs is a natural barrier in representing BRSs faithfully, though. My preliminary experience shows, however, that we might need to slacken this constraint

(i.e., of preserving graph isomorphism), if we aim to transfer from BRSs concepts such as arbitrary copying and deletion of subgraphs. More work is needed to confirm this, though.

## 7 Conclusion

This concludes the first part of this report — with an overview of my completed work and my proposals for future work. In summary, I have developed for binding bigraphs,

- a term language,
- a set of normal form theorems,
- an equational theory on terms that captures graph isomorphism on the term level, and
- a complete inductive characterization of matching for describing when and where a redex matches an agent.

This work was developed together with co-authors and extends the work on pure bigraphs by Milner [Mil05]. The remainder of this report — Parts II and III — is devoted to presenting this work in more detail.

Moreover, I have presented and discussed a set of proposals for future work, giving particular focus to those that are applicable for a future tool for BRSs. Finally, I have presented a preliminary literature study focusing mainly on term rewriting and graph transformation; as argued in Section 6.1, from a modelling perspective BRSs, term rewriting systems and graph transformation systems have much in common.

**Acknowledgements** This work was developed with the support of the entire BPL group at the IT University of Copenhagen. I am grateful for many useful discussions with all the members of the group. In particular, I extend my warm thanks to my supervisor Lars Birkedal, and to Thomas Hildebrandt, Henning Niss, and Søren Debois who have all read earlier drafts of this report. Their critique and ideas have improved it considerably.

I wish to thank my loving parents, Eva and Hans Christian, and my brother, Stephan, for their everlasting support, sparring, and patience. Finally, I wish to thank my beloved fiancée, Lena, who continues to challenge me intellectually and support me affectionately — both financially and emotionally.

## References

- [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<sup>+</sup>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<sup>+</sup>06] Lars Birkedal, Søren Debois, Ebbe Elsborg, Thomas Hildebrandt, and Henning Niss. Bigraphical models of context-aware systems. In Luca Aceto and Anna Ingólfssdóttir, 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.
- [BDGM06] Lars Birkedal, Troels C. Damgaard, Arne John Glenstrup, and Robin Milner. Matching of bigraphs. In *Proceedings of Graph Transformation for Verification and Concurrency Workshop 2006*, Electronic Notes in Theoretical Computer Science. Elsevier, August 2006. To appear.
- [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.
- [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 LaCo-MoCo research project. In *Second UK UbiNet Workshop, Cambridge*, May 2004. position paper.
- [BN98] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, New York, NY, USA, 1998.
- [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.
- [CDE<sup>+</sup>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<sup>+</sup>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.
- [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.
- [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, 2005.
- [CMS06] Giovanni Conforti, Damiano Macedonio, and Vladimiro Sassone. Bilog: Spatial Logics for Bigraphs. *Information and Computation*, 2006. Submitted for publication.
- [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/>.
- [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, February 2005.
- [DD05b] Søren Debois and Troels C. Damgaard. Bigraphs by Example. Technical Report TR-2005-61, IT University of Copenhagen, March 2005.
- [DELM00] Francisco Durán, Steven Eker, Patrick Lincoln, and José Meseguer. Mobile Maude. volume 1882 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.
- [DHP02] Frank Drewes, Berthold Hoffmann, and Detlef Plump. Hierarchical graph transformation. *J. Comput. Syst. Sci.*, 64(2):249–283, 2002.
- [EB70] Knuth D. E. and Bendix P. B. Simple word problems in universal algebra. In *Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967)*, pages 263–297. Pergamon Press, 1970.
- [Ecl] Eclipse project. <http://www.eclipse.org/>.
- [EEHT05] Karsten Ehrig, Claudia Ermel, Stefan Hänsgen, and Gabriele Taentzer. Generation of visual editors as eclipse plug-ins. In *ASE '05: Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, pages 134–143, New York, NY, USA, 2005. ACM Press.

- [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.
- [Ehr02] Hartmut Ehrig. Bigraphs meet double pushouts. *Bulletin of the EATCS*, 78:72–85, 2002.
- [EK04] Hartmut Ehrig and Barbara König. Deriving bisimulation congruences in the DPO approach to graph rewriting. In Walukiewicz [Wal04], pages 151–166.
- [Eke00] S. Eker. Fast matching in combinations of regular equational theories. In J. Meseguer, editor, *Electronic Notes in Theoretical Computer Science*, volume 4. Elsevier Science Publishers, 2000.
- [Eke03] Steven Eker. Associative-commutative rewriting on large terms. In Robert Nieuwenhuis, editor, *Rewriting Techniques and Applications (RTA 2003)*, volume 2706 of *Lecture Notes in Computer Science*, pages 14–29. Springer-Verlag, June 2003.
- [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.
- [GHL99] Fabio Gadducci, Reiko Heckel, and Mercè Llabrés. A bi-categorical axiomatisation of concurrent graph rewriting. *Electronic Notes in Theoretical Computer Science*, 29, 1999.
- [Jen06] Ole H. Jensen. *Mobile Processes in Bigraphs*. PhD thesis, Univ. of Cambridge, 2006. Forthcoming.

- [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.
- [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.
- [LS04] Stephen Lack and Pawel Sobocinski. Adhesive categories. In Walukiewicz [Wal04], pages 273–288.
- [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.
- [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.

- [Min03] Mark Minas. Visual specification of Visual Editors with Visual-DiaGen. In John L. Pfaltz, Manfred Nagl, and Boris Böhlen, editors, *AGTIVE*, volume 3062 of *Lecture Notes in Computer Science*, pages 473–478. Springer, 2003.
- [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.
- [PE93] Rinus Plasmeijer and Marko Van Eekelen. *Functional Programming and Parallel Graph Rewriting*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1993.
- [PRO] PROGRES homepage. <http://www-i3.informatik.rwth-aachen.de/research/projects/progres/>.
- [Pv98] M. Plasmeijer and M. van Eekelen. Language report: Concurrent Clean. Technical report, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, June 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.
- [Roz97] Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. 1: Foundations*. World Scientific, Singapore, 1997.
- [Rud00] Michael Rudolf. Utilizing constraint satisfaction techniques for efficient graph pattern matching. In *TAGT'98: Selected papers from the 6th International Workshop on Theory and Application of Graph Transformations*, pages 238–251, London, UK, 2000. Springer-Verlag.
- [Sew02] Peter Sewell. From rewrite rules to bisimulation congruences. *Theoretical Computer Science*, 274(1–2):183–230, 2002.
- [SS02] V. Sassone and P. Sobocinski. Deriving bisimulation congruences: A 2-categorical approach. In *Proceedings of Express '02*, number 68(2) in *Electronic Notes in Theoretical Computer Science*, May 2002.
- [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.
- [TeR03] TeReSe. *Term Rewriting Systems*. Number 55 in *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.
- [Tig] Tiger project. <http://tfs.cs.tu-berlin.de/tigerprj/>.
- [Vic94] Björn Victor. *A Verification Tool for the Polyadic  $\pi$ -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.
- [Wal04] Igor Walukiewicz, editor. *Foundations of Software Science and Computation Structures, 7th International Conference, FOS-SACS 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29 - April 2, 2004, Proceedings*, volume 2987 of *Lecture Notes in Computer Science*. Springer, 2004.
- [Wol98] D. Wolz. *Colimit Library for Graph Transformations and Algebraic Development Techniques*. PhD thesis, Technische Universität, Berlin, 1998.
- [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.

## Part II

# Axiomatizing Binding Bigraphs

This part consists of the journal paper “Axiomatizing Binding Bigraphs” [DB06]. The paper is presented unchanged. As we have found two minor errors after the publication of the paper, I include an erratta-sheet after the main body of the paper.

# Axiomatizing Binding Bigraphs

T.C. Damgaard  
*IT University of Copenhagen*  
*Denmark*  
*tcd@itu.dk*

L. Birkedal  
*IT University of Copenhagen*  
*Denmark*  
*birkedal@itu.dk*

**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.

**ACM CCS Categories and Subject Descriptors:** D.3.1. Formal Definitions and Theory, F.3.2. Process Models.

**Key words:** graphical models of computation, bigraphs, axioms for static congruence.

## 1. Introduction

Over the last decade, Robin Milner and co-workers have developed a theory of bigraphical reactive systems, see [9, 12, 13]. 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  $\pi$ -calculus [9], the ambient calculus [10], and has contributed to that for Petri nets [11]. Thus the evaluation of the second aim has so far been encouraging. In [3] 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 [9, 2, 1, 3] 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 [12]. 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  $\pi$ -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 [9] and with the axiomatization of pure bigraphs [12] — we do, however, include an informal description of bigraphs in the following section and we have included the formal definition of binding bigraphs in Appendix A.

The remainder of the paper is organized as follows. In the following section we introduce bigraphs by example. In Section 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 4 we present our term language for binding bigraphs and the accompanying equational theory. We arrive at a theorem which states soundness and completeness of the equational theory. We present some examples of bigraphs and their corresponding normal forms in Section 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 6. Finally, Appendix A contains a summary of the definitions of binding bigraphs. We have omitted detailed proofs from this paper, they can be found in the companion technical report [5].

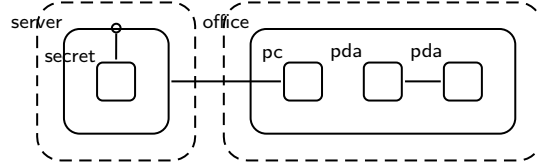


Figure 2.1: A – a bigraph model of an office in a building

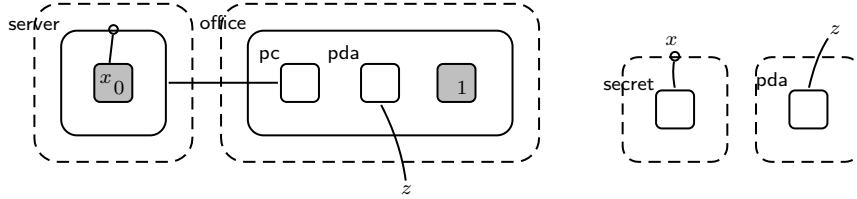


Figure 2.2: B and C – bigraphs that compose to form A

## 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 Appendix A for all formal definitions.

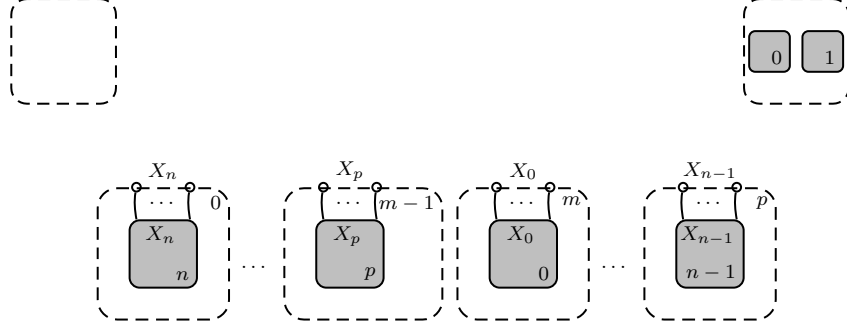
The bigraph  $A$  is bigraph model of an office containing a pc and two pdas. The pc is linked (supposedly by some kind of network connection) to server containing a secret located somewhere else. We say that  $A$  consists of **roots** (dashed boxes), **nodes** (solid boxes), and **links** (lines). 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 13).

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  $B : \langle 2, (\{x\}, \emptyset), \{x, z\} \rangle \rightarrow \langle 2, (\emptyset, \emptyset), \emptyset \rangle$  and  $C : \langle 0, (\emptyset), \emptyset \rangle \rightarrow \langle 2, (\{x\}, \emptyset), \{x, z\} \rangle$ .



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

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 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 14 for the full definition of discreteness).

For more involved examples of bigraphical models including dynamics, we refer the reader to the tech report [6].

### 3. Elementary Bigraphs and Normal Form

We start by defining **placings** corresponding closely to the placings defined for pure bigraphs in [12]. We shall use placings to define the class of terms for bigraphs that denote place graphs paired with identities on local names.

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

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 13).

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

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

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

Using *join* we define the bigraph  $\text{merge}_m$  that joins  $m$  sites:

**Definition 1** (merge). *For all  $m \geq 0$  we define  $\text{merge}_m : m \rightarrow 1$  recursively, by*

$$\begin{aligned} \text{merge}_0 &\stackrel{\text{def}}{=} 1 \\ \text{merge}_{m+1} &\stackrel{\text{def}}{=} \text{join}(\text{id}_1 \otimes \text{merge}_m). \end{aligned}$$

Note that  $\text{merge}_1 = \text{id}_1$  and thus  $\text{merge}_2 = \text{join}$ .

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

$$\begin{aligned} /x &: x \rightarrow \epsilon \quad \text{closure,} \\ y/X &: X \rightarrow y \quad \text{substitution } x \mapsto y, \text{ for all } x \in X. \end{aligned}$$

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

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

Note that a substitution need not be surjective (i.e., we allow  $X = \emptyset$ ), thus the dual of closure – name introduction  $y : \epsilon \rightarrow y$  – 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  $\omega$  range over wirings,  $\sigma$  range over (multiple) substitutions and  $\alpha$  and  $\beta$  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  $\vec{y} = y_0, \dots, y_{k-1}$  and  $\vec{x} = x_0, \dots, x_{k-1}$ , 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.

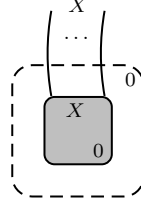
$$\lceil X \rceil : (X) \rightarrow \langle X \rangle \quad \text{concretion.}$$

Note that a special case of a simple concretion is  $\text{id}_1 = \lceil \emptyset \rceil$ .

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

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

where  $X \subseteq Y \setminus Y_B$ .

Figure 3.4:  $\ulcorner X \urcorner$ 

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 I)

Using abstraction we can express concretions in the sense of [9]: We define a general **concretion**  $\ulcorner Y \urcorner^X : \langle 1, (X \uplus Y), X \uplus Y \rangle \rightarrow \langle 1, (X), X \uplus Y \rangle$  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  $\ulcorner \alpha \urcorner \stackrel{\text{def}}{=} (\alpha \otimes \text{id}_1) \ulcorner X \urcorner$  (where  $\alpha : X \rightarrow$ ).

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  $(\vec{y})/(\vec{x}) \stackrel{\text{def}}{=} (\vec{y})((\vec{y}/\vec{x} \otimes \text{id}_1) \ulcorner \vec{x} \urcorner)$ . We extend this notation to multiple substitutions and define  $(\vec{y})/(\vec{X}) \stackrel{\text{def}}{=} (\vec{y})((\vec{y}/\vec{X} \otimes \text{id}_1) \ulcorner X \urcorner)$  (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  $\alpha^{\text{loc}}$  and  $\sigma^{\text{loc}}$  range over local renamings and substitutions, respectively. We shall need to take the preimage of a local substitution  $\sigma^{\text{loc}}$  of a vector of namesets  $\vec{X}$ . Formally:

**Definition 2** (Preimage of a local wiring). *Let  $\sigma_{\mathbf{u}}^{\text{loc}}$  be the link map (which is a function) of  $\sigma^{\text{loc}}$ . For a set of names  $X$ , define  $(\sigma^{\text{loc}})^{-1}(X)$  to be the preimage  $(\sigma_{\mathbf{u}}^{\text{loc}})^{-1}(X)$  and define  $(\sigma^{\text{loc}})^{-1}(\vec{X})$  to be the vector of namesets resulting from taking the preimage of  $\sigma^{\text{loc}}$  pointwise for each set in  $\vec{X}$ .*

We can generate all isomorphisms in the category of binding bigraphs using permutations  $\pi$ , renamings  $\alpha$ , and local renamings  $\alpha^{\text{loc}}$  (see [9, Proposition 9.2b] for the definition of isomorphism in the category of binding bigraphs):

**Proposition 1.** *Every binding bigraph isomorphism,  $\iota : \langle m, \vec{Z}, \{\vec{Z}\} \uplus U \rangle \rightarrow \langle m, \vec{X}, \{\vec{X}\} \uplus Y \rangle$  (of width  $m$ ) may be expressed in the following form*

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

where these requirements hold:

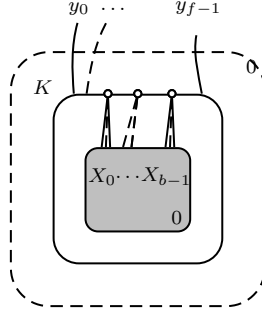
$$\circ m = |\vec{X}| = |\vec{Z}|,$$

- $\alpha : U \rightarrow Y$ ,
- $\forall i \in m : \nu_i = (\vec{x}_i)/(\vec{z}_i)$  for  $\vec{X} = (\{\vec{x}_0\}, \dots, \{\vec{x}_{m-1}\})$ ,  
and  $\vec{Z} = (\{\vec{z}_0\}, \dots, \{\vec{z}_{m-1}\})$ .

For a control  $K : b \rightarrow f \in \mathcal{K}$ , 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**  $K_{\vec{y}(\vec{X})} : \langle 1, (X), X \rangle \rightarrow \langle 1, (\emptyset), Y \rangle$  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.

$K_{\vec{y}(\vec{X})} : (X) \rightarrow \langle Y \rangle$  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 [9, Chapter 11]. We can recapture the latter by requiring every set in  $X$  to be a singleton. When  $\vec{X} = (\{x_0\}, \dots, \{x_{b-1}\})$ , we overload our notation and write  $K_{\vec{y}(\vec{x})}$  to mean a free discrete ion.

**Definition 3.** For any name-discrete prime  $P : I \rightarrow \langle 1, (X), X \uplus Z \rangle$  and ion  $K_{\vec{y}(\vec{X})}$ , we define a **free discrete molecule** as

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

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 [9, 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 2.** 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 13). Vice versa, we have:

**Proposition 3.** *Any free discrete prime bigraph with a single outermost node is a free discrete molecule.*

### 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 4.

We aim to base our normal form on a variant of discreteness, as in [12], 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  $P : I \rightarrow \langle (X), X \uplus Y \rangle$ ,  $(Y)P : I \rightarrow (X \uplus Y)$  is discrete. Hence, we turn to name-discreteness.

Recall that a bigraph is name-discrete (Definition 14) 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 1** (Semantic binding discrete normal form).

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

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

where  $P : I \rightarrow \langle 1, (\{\vec{X}\}), \{\vec{X}\} \uplus Z \rangle$  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*

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

where the following requirements hold:

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

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

$$(Y_B) (\text{merge}_{n+k} \otimes \text{id}_Y) (\ulcorner \alpha_0 \urcorner \otimes \cdots \otimes \ulcorner \alpha_{n-1} \urcorner \otimes M_0 \otimes \cdots \otimes M_{k-1}) \pi$$

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

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

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

where the following requirements hold:

- There exist permutations  $\rho, \rho_i$  ( $i \in k$ ),  $\rho'$ , s.t.
  - $\ulcorner \alpha'_i \urcorner = \ulcorner \alpha_{\rho(i)} \urcorner$
  - $M'_i = M_{\rho(i)} \rho_i$ ,
  - $(\text{id}_{(X'_0)} \otimes \cdots \otimes \text{id}_{(X'_{n-1})}) \otimes \rho_0 \otimes \cdots \otimes \rho_{k-1}) \pi' = \rho' \pi$ .
- Furthermore, let  $\vec{l}$  denote the vector of inner widths of the product  $((\alpha_0 \otimes \text{id}_1) \ulcorner X_0 \urcorner \otimes \cdots \otimes (\alpha_{n-1} \otimes \text{id}_1) \ulcorner X_{n-1} \urcorner \otimes M_0 \otimes \cdots \otimes M_{k-1})$ , let  $\vec{X}' = (X'_0, \dots, X'_{k-1})$ , and let  $\vec{X} = (X_0, \dots, X_{n-1})$ . Then  $\rho'$  is determined uniquely by  $\rho, \vec{l}, \vec{X}$ , and  $\vec{X}'$  as  $\rho' = \bar{\rho}_{\vec{l}, \vec{X}', \vec{X}}$  as defined in Lemma 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,  $\alpha$  is a renaming, and  $\pi$  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

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

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

(4) Any bigraph  $G : I \rightarrow \langle n, \vec{Y}_B, \{\vec{Y}_B\} \uplus Y_F \rangle$  can be expressed as

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

where  $D : I \rightarrow \langle n, \vec{X}, X \uplus Z \rangle$  is name-discrete,  $\omega : Z \rightarrow Y_F$  is a wiring, and  $\bigotimes_{i < n} (\vec{y}_i) / (\vec{X}_i) : (\vec{X}) \rightarrow (\vec{Y}_B)$  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  $\alpha$  s.t.  $\omega' = \omega\alpha$ , and  $n$  local renamings  $\alpha_i^{\text{loc}} : (\vec{X}'_i) \rightarrow (\vec{X}_i)$ , s.t.  $(\bigotimes_{i < n} (\vec{y}_i) / (\vec{X}_i)) \bigotimes_{i < n} \alpha_i^{\text{loc}} = (\bigotimes_{i < n} (\vec{y}_i) / (\vec{X}'_i))$ , and  $(\bigotimes_{i < n} \alpha_i^{\text{loc}} \otimes \alpha) D' = D$ .

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

See [5] for a proof of the theorem. The proof is simply a detailed analysis of the structure of possible decompositions of binding bigraphs.

#### 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:

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

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

In [12] 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 I complete for binding bigraph expressions. Every pure bigraph expression as defined by Milner [12] 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.

#### 4.1 Preliminaries

**Lemma 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 2** (Push-through lemma). *For  $n$  primes  $P_i$*

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

and permutation  $\pi$ , there exists a permutation  $\bar{\pi}_{\vec{m}, \vec{X}}$ , which depends solely on  $\pi$ ,  $\vec{m}$ , and  $\vec{X} = (\vec{X}_0, \dots, \vec{X}_{n-1})$ , s.t.,

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

#### 4.2 $\mathbf{Place}_{\mathbf{L}}$ expressions

Let  $\mathbf{Place}_{\mathbf{L}}$  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,  $\mathbf{Place}_{\mathbf{L}}$  consists of all expressions denoting place graphs paired with identities on local names. We shall start by proving that the theory is complete for  $\mathbf{Place}_{\mathbf{L}}$  expressions.

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

**Definition 4** (binding join). *For sets of names  $X$  and  $Y$  let  $bjoin_{(X, Y)}$ , the binding join bigraph, be defined as*

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

We also define an iterated version

**Categorical axioms**

$$\begin{aligned}
\text{(C1)} \quad & A \text{ id}_I = A = \text{id}_J A && (A : I \rightarrow J) \\
\text{(C2)} \quad & A(BC) = (AB)C \\
\text{(C3)} \quad & A \otimes \text{id}_\epsilon = A = \text{id}_\epsilon \otimes A \\
\text{(C4)} \quad & A \otimes (B \otimes C) = (A \otimes B) \otimes C \\
\text{(C5)} \quad & \text{id}_I \otimes \text{id}_J = \text{id}_{I \otimes J} \\
\text{(C6)} \quad & (A_1 \otimes B_1)(A_0 \otimes B_0) = (A_1 \circ A_0) \otimes (B_1 \circ B_0) \\
\text{(C7)} \quad & \gamma_{I,\epsilon} = \text{id}_I \\
\text{(C8)} \quad & \gamma_{J,I} \gamma_{I,J} = \text{id}_{I \otimes J} \\
\text{(C9)} \quad & \gamma_{I,K}(A \otimes B) = (B \otimes A)\gamma_{H,J} \quad (A : H \rightarrow I, B : J \rightarrow K)
\end{aligned}$$

**Link axioms**

$$\begin{aligned}
\text{(L1)} \quad & x/x = \text{id}_x \\
\text{(L2)} \quad & /y \circ y/x = /x \\
\text{(L3)} \quad & /y \circ y = \text{id}_\epsilon \\
\text{(L4)} \quad & z/(Y \uplus y)(\text{id}_Y \otimes y/X) = z/(Y \uplus X)
\end{aligned}$$

**Place axioms**

$$\begin{aligned}
\text{(P1)} \quad & \text{join}(1 \otimes \text{id}_1) = \text{id}_1 \\
\text{(P2)} \quad & \text{join}(\text{join} \otimes \text{id}_1) = \text{join}(\text{id}_1 \otimes \text{join}) \\
\text{(P3)} \quad & \text{join} \gamma_{1,1,(\emptyset,\emptyset)} = \text{join}
\end{aligned}$$

**Binding axioms**

$$\begin{aligned}
\text{(B1)} \quad & (\emptyset)P = P \\
\text{(B2)} \quad & (Y)^\ulcorner Y^\urcorner = \text{id}_{(Y)} \\
\text{(B3)} \quad & (\ulcorner X^\urcorner \otimes \text{id}_Y)(X)P = P \quad (P : I \rightarrow \langle 1, (Z), Z \uplus X \uplus Y \rangle) \\
\text{(B4)} \quad & (((Y)(P)) \otimes \text{id}_X)G = (Y)(P \otimes \text{id}_X)G \\
\text{(B5)} \quad & (X \uplus Y)P = (X)((Y)P)
\end{aligned}$$

**Ion axioms**

$$\begin{aligned}
\text{(N1)} \quad & (\text{id}_1 \otimes \alpha)K_{\vec{y}(\vec{X})} = K_{\alpha(\vec{y})(\vec{X})} \\
\text{(N2)} \quad & K_{\vec{y}(\vec{X})}\sigma^{\text{loc}} = K_{\vec{y}((\sigma^{\text{loc}})^{-1}(\vec{X}))}
\end{aligned}$$

TABLE I: Axioms for binding bigraphs

**Definition 5** (binding merge). *For all  $m \geq 0$  we define  $bmerge_{m, \vec{X}}$  recursively, by*

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

Binding join and merge behave similarly as their underlying place expressions when composed with permutations or themselves (refer the place graph axioms of Table I), 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.**

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

where in the last equation  $m = \sum_{i < k} m_i$  and  $\vec{X} = \vec{X}_0 \dots \vec{X}_{k-1}$ .

Using binding merge, we define and prove sufficient a normal form for **Place<sub>L</sub>** expressions.

**Lemma 4** (Normal form for **Place<sub>L</sub>** expressions). *For every **Place<sub>L</sub>** expression  $E$*

$$\vdash E = (bmerge_{m_0, \vec{X}_0} \otimes \dots \otimes bmerge_{m_{k-1}, \vec{X}_{k-1}}) \circ \pi$$

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

With the help of Lemma 3 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 **Place<sub>L</sub>** expressions follows with the help of the uniqueness properties stated in Theorem 1. These yield a number of equations which are provable within the theory.

**Proposition 4** (Completeness for **Place<sub>L</sub>** 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  $\vdash E = F$ .

### 4.3 $\mathbf{Link}_{\mathbf{G}}$ 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  $\mathbf{Link}_{\mathbf{G}}$ . Our term language for binding bigraphs has the same constructs for linking as the language used by Milner for pure bigraphs [12]. 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 5** (Link completeness). *The theory is complete for link expressions.*

### 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 6** (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 5.** *If  $E$  is linear expression, then  $\vdash E = E' \otimes \alpha$ , where  $E'$  is linear and has local innerface.*

**Lemma 6.** *If  $E : \langle m, \vec{U}, \{\vec{U}\} \rangle \rightarrow \langle n, \vec{Y}, \{\vec{Y}\} \uplus V \rangle$  is a linear expression with local innerface, then*

$$\vdash E \circ \bigotimes_{i < m} (\vec{u}_i) / (\vec{Z}_i) = \left( \left( \bigotimes_{i < n} (\vec{y}_i) / (\vec{X}_i) \right) \otimes \text{id}_V \right) \circ E',$$

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 1, item 4.

**Proposition 6** (Underlying linear expression). *For any expression  $G$  denoting a bigraph of outer width  $n$ , there exists a wiring  $\omega$ , a linear expression  $E$ , and a local renaming  $\bigotimes_{i < n} (\vec{y}_i) / (\vec{X}_i)$ , s.t.,*

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

The proof is by structural induction on  $G$ , using the lemmas above [5].

#### 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 7.** *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 8.** *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  $G^P \in \mathbf{Place}_{\mathbf{L}}$ .*

**Lemma 9.** *If  $E$  is linear and ion-free, then there exists concretions,  $E'$ , and a renaming  $\alpha$  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 10** (A normal form for ion-free expressions). *For all ion-free expressions  $G$  of width  $n$*

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

where  $G^P \in \mathbf{Place}_{\mathbf{L}}$ .

Completeness for ion-free expressions follows easily.

**Corollary 1** (The theory is complete for ion-free expressions).

#### 4.6 Syntactic Normal Form

Corresponding to the four classes of normal forms in Theorem 1 we define four classes of syntactic normal forms for binding bigraph expressions:

**Definition 7** (Syntactic binding discrete normal form (BDNF)).

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

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

We also need that DDNF expressions are closed under composition.

**Lemma 12** (DDNF is closed under composition). *For all composable DDNFs  $C, D$ , there exists a DDNF  $D'$ , s.t.  $\vdash D \circ C = D'$ .*

We now state the correspondence between our semantic normal form (Theorem 1) 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 7** (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 PDFN  $P$ .*
- (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 1 to yield a number of equations that are provable within our theory. We refer to the companion technical report [5] for more details on the proofs.

**Proposition 8** (Linear completeness). *If  $E$  and  $E'$  are linear expressions and  $E = E'$ , then  $\vdash E = E'$ .*

**Theorem 2** (Soundness and Completeness). *For all binding bigraph expressions  $E$  and  $F$ ,  $\vDash E = F$  iff  $\vdash E = F$ .*

## 5. Term Language and Normal Forms – by Example

We shall use our examples from Section 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  $((x)(\text{secret}_x \circ 1)) \otimes \text{pda}_z \circ 1$ , while  $B$  can be expressed for example as

$$\begin{aligned} & (\text{id}_2 \otimes / \{e0, e1\}) \circ ( \text{server}_{e0(\{x\})} ) \otimes \\ & (\text{id}_{\langle 1, e1 \rangle} \otimes / \{f0, f1\}) (\text{office} \circ 1 \otimes \text{id}_{\{e1, f0, f1\}}) \\ & (\text{merge}_3 \otimes \text{id}_{\{e1, f0, f1\}}) (\text{pc}_{e1} \circ 1 \otimes \text{pda}_{f0} \circ 1 \otimes \text{pda}_{f1} \circ 1) \end{aligned}$$

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  $B$ :

$$\begin{aligned}
 & (\text{id}_2 \otimes / \{e0, e1\} \otimes / \{f0, f1\}) \circ \\
 & ( (\emptyset)(\text{merge}_1 \otimes \text{id}_{e0})(\text{server}_{e0(\{x\})} \circ (x)(\text{merge}_1 \otimes \text{id}_x)(\ulcorner x \urcorner)) \otimes \\
 & (\emptyset)(\text{merge}_1 \otimes \text{id}_{\{e1, f0, f1\}})(\text{office} \circ \\
 & (\emptyset)(\text{merge}_3 \otimes \text{id}_{\{e1, f0, f1\}})(\text{pc}_{e1} \circ 1 \otimes \text{pda}_{f0} \circ 1 \otimes \text{pda}_{f1} \circ 1)) \\
 & \otimes \text{id}_\epsilon).
 \end{aligned}$$

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

## 6. Related and further work

Bigraphical reactive systems are related to graph transformation systems using the double pushout construction [7] and, recently, it has also been investigated how to derive bisimulation congruences in the double pushout approach to graph rewriting [8].

Recent work on spatial logics [4] for pure bigraphs utilizes the axiomatization of pure bigraphs by Milner [12]. 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 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 [9], 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 [3] work.

### Acknowledgements

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.

### Appendix A. Definition of Binding Bigraphs

We recall the definition of binding bigraphs [9].

**Definition 8** (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**  $\text{ar}_b(K) = h$  and the **free arity**  $\text{ar}_f(K) = k$ . We write  $\text{ar}(K) = \text{ar}_b(K) + \text{ar}_f(K)$ .

**Definition 9** (binding interface). A **binding interface**  $I = \langle m, \text{loc}, X \rangle$ , consists of a **width**  $m$ , a finite set of **names**  $X$ , and a **locality map**  $\text{loc} : X \rightarrow m \uplus \perp$ , which associates some of the names in  $X$  with a location in  $m$ ; if  $\text{loc}(x) = i \in m$ , we say  $x$  is **located** at  $i$  or **local** to  $i$ . When  $\text{loc}(x) = \perp$  we say  $x$  is **global**.

For an interface  $I = \langle m, \text{loc}, X \rangle$  we shall typically represent the locality map by a vector of disjoint subsets  $\vec{X} = (X_0, \dots, X_{m-1})$ , 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  $\epsilon$  to denote the interface  $\langle 0, (\cdot), \emptyset \rangle$ .

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 13).

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

**Definition 10** (place graph). A *(concrete) place graph* over signature  $\mathcal{K}$   $G = (V, ctrl, prnt) : m \rightarrow n$  has an **inner width**  $m$  and an **outer width**  $n$ , both finite ordinals; a finite set  $V$  of nodes with a control map  $ctrl : V \rightarrow \mathcal{K}$ ; and a **parent map**  $prnt : m \uplus V \rightarrow V \uplus n$ . 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  $G : m \rightarrow n$  index  $G$ 's **sites**  $0, \dots, m-1$  and **roots**  $0, \dots, n-1$ , respectively. We use  $\epsilon$  to denote the width 0. A place graph with inner width 0 is called an **agent**.

Place graphs are composed as follows. Let  $G_i = (V_i, ctrl_i, prnt_i) : m_i \rightarrow m_{i+1}$  ( $i \in \{0, 1\}$ ) 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  $prnt = (\text{id}_{V_0} \uplus prnt_1) \circ (prnt_0 \uplus \text{id}_{V_1})$ .

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

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  $F : k \rightarrow l$  and  $G : m \rightarrow n$  with disjoint node sets is  $F \otimes G : k+m \rightarrow l+n$ . It consists of placing the two forests side-by-side (see [9, Definition 7.5] for a formal definition). Note that  $\text{id}_\epsilon = \text{id}_0$  is the unit for  $\otimes$ , in the sense that  $F \otimes \text{id}_\epsilon = \text{id}_\epsilon \otimes F = F$ , for all place graphs  $F$ . Thus, an iterated tensor product  $F_0 \otimes \dots \otimes F_{k-1}$  equals  $\text{id}_\epsilon$  in case  $k=0$ .

Two concrete place graphs  $G_0$  and  $G_1$  are said to be **support equivalent**,  $G_0 \simeq G_1$ , if they differ only by a bijection between their node sets. An **abstract place graph** is an  $\simeq$ -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  $[F]_{\simeq} \otimes [G]_{\simeq} \stackrel{\text{def}}{=} [F \otimes G]_{\simeq}$ , makes it into a strict symmetric monoidal category.

**Definition 11** (link graph). A *(concrete) link graph*  $G$  over a signature  $\mathcal{K}$ , is a tuple  $(V, E, ctrl, link) : X \rightarrow Y$  with finite sets of nodes  $V$ , edges  $E$ , **inner names**  $X$ , and **outer names**  $Y$ . As place graphs it has a control map  $ctrl : V \rightarrow \mathcal{K}$ . The function  $link : X \uplus P \rightarrow E \uplus Y$  maps **points**, i.e., inner names  $X$  and ports  $P = \sum_{v \in V} \text{ar}(ctrl v)$  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  $G_i = (V_i, E_i, ctrl_i, link_i) : X_i \rightarrow X_{i+1}$  ( $i \in \{0, 1\}$ ) is defined when  $V_0 \cap V_1 = \emptyset$  and  $E_0 \cap E_1 = \emptyset$ ; and is then  $G_1 \circ G_0 \stackrel{\text{def}}{=} (V, E, ctrl, link) : X_0 \rightarrow X_2$ ; where  $V = V_0 \uplus V_1$ ,  $E = E_0 \uplus E_1$ ,  $ctrl = ctrl_0 \uplus ctrl_1$ , and  $link = (\text{id}_{E_0} \uplus link_1) \circ (link_0 \uplus \text{id}_{P_1})$ .

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

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

**Definition 12** (binding bigraph). A (*concrete*) **binding bigraph**  $G = (V, E, ctrl, G^{\mathbf{P}}, G^{\mathbf{L}}) : I \rightarrow J$  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  $ctrl : V \rightarrow \mathcal{K}$ , exactly as for link graphs.

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

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

**Definition 13** (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  $w' = prnt_{G^{\mathbf{P}}}^k(w)$ , 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  $G : I \rightarrow J$  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  $G = \langle G^{\mathbf{P}}, G^{\mathbf{L}} \rangle : I \rightarrow J$ . 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.,  $G = (V, E, ctrl, prnt, link) : I \rightarrow J$ .

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  $G_i = \langle G_i^{\mathbf{P}}, G_i^{\mathbf{L}} \rangle : I_i \rightarrow J_i$  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,  $G_1 \circ G_0 \stackrel{\text{def}}{=} \langle G_1^{\mathbf{P}} \circ G_0^{\mathbf{P}}, G_1^{\mathbf{L}} \circ G_0^{\mathbf{L}} \rangle : I_0 \rightarrow J_1$  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 [9, 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  $I = \langle m, \text{loc}, X \rangle$  then  $\text{id}_I \stackrel{\text{def}}{=} \langle \text{id}_m, \text{id}_X \rangle : I \rightarrow I$ .

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  $\text{id}_\epsilon$ ; thus, an iterated tensor product  $P_0 \otimes \cdots \otimes P_{n-1}$  equals  $\text{id}_\epsilon$  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  $G_0 \simeq G_1$  iff they differ only by a bijection between their nodes and their non-idle edges; idle edges are disregarded entirely.

**Abstract binding bigraphs** are  $\simeq$ -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  $\epsilon$ . 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 14** (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 13.** *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  $B$ .*

## References

- [1] BIRKEDAL, L., DEBOIS, S., ELSBORG, E., HILDEBRANDT, T., AND NISS, H. 2006. Bigraphical Models of Context-aware Systems. In *In Proceedings of FOSSACS 2006*.
- [2] BIRKEDAL, LARS. 2004. Bigraphical Programming Languages—A LaCoMoCo Research Project. In *Second UK UbiNet Workshop, Cambridge*.
- [3] BIRKEDAL, LARS, DEBOIS, SØREN, ELSBORG, EBBE, HILDEBRANDT, THOMAS, AND NISS, HENNING. 2005. Bigraphical Models of Context-aware Systems. Tech. Report TR-2005-74, IT University of Copenhagen, Rued Langgards Vej 7, DK-2300 Copenhagen V.

- [4] CONFORTI, G., MACEDONIO, D., AND SASSONE, V. 2005. Spatial Logics for Bigraphs. In *Proc. of International Colloquium on Automata, Languages and Programming*.
- [5] DAMGAARD, T.C. AND BIRKEDAL, L. 2005. Axiomatization of Binding Bigraphs (Revised). Tech. Report TR-2005-71, IT University of Copenhagen.
- [6] DEBOIS, S. AND DAMGAARD, T. C. 2005. Bigraphs by Example. Tech. Report TR-2005-61, The IT University of Copenhagen.
- [7] EHRIG, H. 1979. Introduction to the theory of graph grammars. In *Graph Grammars and their application to Computer Science and Biology*, Number 73 in Lecture Notes in Computer Science. Springer-Verlag, 1-69.
- [8] EHRIG, H. AND KÖNIG, B. 2004. Deriving Bisimulation congruences in the DPO approach to graph rewriting. In *Foundations of Software Science and Computation Structures*, Volume 2987 of *Lecture Notes in Computer Science*. Springer-Verlag.
- [9] HØGH JENSEN, OLE AND MILNER, ROBIN. 2004. Bigraphs and mobile processes (revised). Tech. Report 580, University of Cambridge.
- [10] JENSEN, O.H. 2006. *Mobile Processes in Bigraphs (Forthcoming)*. PhD thesis, Univ. of Aalborg.
- [11] LEIFER, JAMES J. AND MILNER, ROBIN. 2004. Transition systems, link graphs and Petri nets. Tech. Report 598, University of Cambridge.
- [12] MILNER, ROBIN. 2005. Axioms for bigraphical structure. *Mathematical Structures in Computer Science* 15, 06 (December), 1005-1032.
- [13] MILNER, ROBIN. 2005. Pure bigraphs. Tech. Report 614, University of Cambridge.

**Errata for “Axiomatizing Binding Bigraphs”** In Table I, the following categorical axiom concerned with symmetries was mistakenly omitted:

$$\gamma_{I \otimes J, K} = (\gamma_{I, K} \otimes \text{id}_J) (\text{id}_I \otimes \gamma_{J, K}).$$

Also, in Lemma 2, replace  $P_{\pi(0)}$  with  $P_{\pi^{-1}(0)}$  and  $P_{\pi(n-1)}$  with  $P_{\pi^{-1}(n-1)}$ .

## Part III

# Matching of Bigraphs

This part contains an extended version of the paper “Matching of Bigraphs” [BDGM06] that I presented at the GT-VC workshop 2006. The paper is currently under publication, and this version is extended with an appendix with extensive details for the proof of the main theorem stating completeness of the characterization (Theorem 3.15).

# Matching of Bigraphs

(extended version)

Lars Birkedal<sup>1</sup> Troels Christoffer Damgaard<sup>1</sup>  
Arne John Glenstrup<sup>1</sup>

*IT University of Copenhagen, Denmark*

Robin Milner<sup>2</sup>

*University of Cambridge, UK*

---

## Abstract

We analyze the matching problem for bigraphs. In particular, we present a sound and complete inductive characterization of matching of binding bigraphs. Our results pave the way for a provably correct matching algorithm, as needed for an implementation of bigraphical reactive systems.

---

## 1 Introduction

Over the last decade, a theory of bigraphical reactive systems has been developed [9,13,14]. 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 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 (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 an abstract definition of reaction

---

<sup>1</sup> Email: {birkedal,tcd,panic}@itu.dk

<sup>2</sup> Email: Robin.Milner@cl.cam.ac.uk

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  $\pi$ -calculus [9], the ambient calculus [8], and has contributed to that for Petri nets [11]. Thus the evaluation of the second aim has so far been encouraging. In [3], 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 [9,1,3] 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. 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.

In Figure 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 which folders can be shared between the corporation and the consultancy group and inside the corporation. Thus the laptop shown in the middle is intended to belong to a consultant working for the corporation — the consultant has folders with data belonging to the consultancy group (the link shown to the left) and folders with data belonging to the corporation (the link shown to the right). The fact that folders belonging to the corporation should not leave the corporation is expressed by linking those folders to a so-called binding port on the corporation building, indicated by the circle.

The abstract semantic definition of matching, as defined in the theory of bigraphs [9], 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  $a = C \circ (R \otimes \text{id}_Z) \circ d$ , then it can be rewritten to  $C \circ (R' \otimes \text{id}_Z) \circ d$ . Here  $\circ$  denotes composition of bigraphs and  $Z$  is the set of global names of  $d$ . 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.

Consider again the example in Figure 1. There is a reaction rule expressed by the redex  $R$  and the reactum  $R'$ ; the intention of the reaction rule is to allow copying of data between connected folders in the same nesting hierarchy (note the link in  $R$  between the two folders and the so-called local name  $y$ ). The agent  $a$  can be written as a composition of  $C$ ,  $R$  and  $d$  — formally,  $a = C \circ (R \otimes \text{id}_z) \circ d$ . Composition works by (1) plugging the roots of  $R$  and  $d$  into the holes (aka sites) of  $C$  respectively  $R$ ; (2) fusing together the

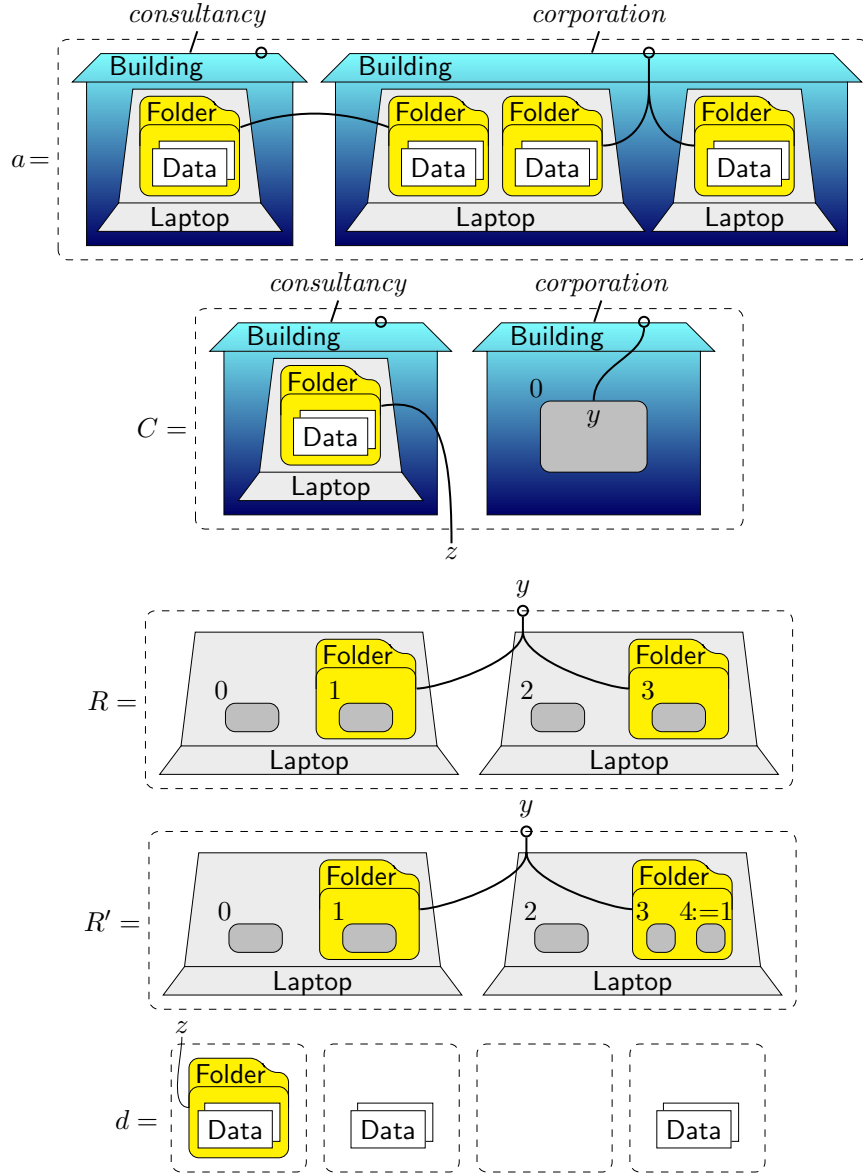


Fig. 1. Example of a ground agent  $a = C \circ (\text{id}_z \otimes R) \circ d$ . Reaction rule  $R \rightarrow R'$  copies data between connected folders.

connections between folder and  $z$  (in  $d$ ) and  $z$  and folder (in  $C$ ), removing the name  $z$  in the process; (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  $\text{id}_z$  in the composition  $a = C \circ (R \otimes \text{id}_z) \circ d$ ; it allows a name  $z$  from the parameter  $d$  to be “passed through” 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 1).

In the present paper we provide an *inductive characterization* of when  $a = C \circ (R \otimes \text{id}_Z) \circ d$  holds, by induction on  $a$  and  $R$  (the input to a matching algorithm). 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 paves the way for developing and *proving correct* an actual matching algorithm (which, given  $a$  and  $R$ , must find  $C$ ,  $d$ , and  $Z$  such that  $a = C \circ (R \otimes \text{id}_Z) \circ d$  holds). We further include a discussion of how one may derive matching algorithms from our inductive characterization. We will report on our work on an actual implementation of matching in a subsequent paper.

Our inductive characterization is based on normal form theorems for bigraphs [13,5], 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 so-called *discrete* decompositions of bigraphs. Discrete bigraphs are bigraphs with 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 [13,5]. One hopes that matching implementations will be efficient in practice since redices typically are small. Furthermore, sorting bigraphs [4] could be a source of early search elimination.

The remainder of this paper is organized as follows. In Section 2 we give an informal description of binding bigraphs. The main contribution of this paper is in Section 3, where we present our inductive characterization of matching. Section 4 discusses how the inductive characterization may ensure a correct and efficient algorithm for matching. In the final sections we discuss related work and conclude.

For lack of space, most proofs [2] have been omitted from this extended abstract.

## 2 Binding Bigraphs

Here we present bigraphs informally; for a formal definition, see [9,5].

### 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  $r_0, \dots, r_n$ , nodes  $v_0, \dots, v_k$ , and a number of special leaves  $s_0, \dots, s_m$  called

sites, while the link graph is a general graph over the node set  $v_0, \dots, v_k$  extended with *inner names*  $x_0, \dots, x_l$ , and equipped with hyper edges, indicating *connectivity*.

We usually illustrate the place graph by nesting nodes, as shown in the upper part of Figure 2 (ignore for now the interfaces denoted by “ $: \cdot \rightarrow \cdot$ ”). A *link* is a hyper edge of the link graph, either an internal *edge*  $e$  or a *name*

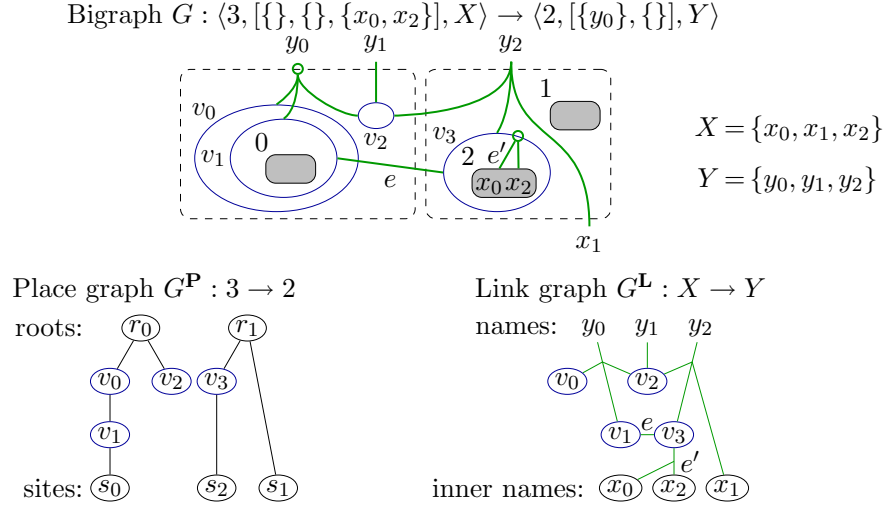


Fig. 2. Example bigraph illustrated by nesting and as place and link graph.

$y$ . Links that are names are called *open*, otherwise they are *closed*. Names and inner names can be *global* or *local*, the latter being located at a specific root or site, respectively. In Figure 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  $e'$  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 of the link lie within its location (in the place graph), except for the binder itself. This prevents  $y_2$  and  $e$  in the example from being bound.

## 2.2 Controls

Every node  $v$  has a *control*  $K$  which determines a binding and free arity, indicated by  $v : K$ . In the example of Figure 2, we could have  $v_i : K_i, i = 0, 1, 2, 3$ , where  $K_0 : 0 \rightarrow 1$ ,  $K_1 : 0 \rightarrow 2$ ,  $K_2 : 0 \rightarrow 3$ ,  $K_3 : 1 \rightarrow 2$ . 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*.

### 2.3 Abstract Bigraphs

While concrete bigraphs with named nodes and internal edges are the basis of bigraph theory [9], our prime interest is in *abstract bigraphs*, equivalence classes of concrete bigraphs that differ only in the names of nodes and internal edges<sup>3</sup>. Abstract bigraphs are illustrated with their node controls, as shown in Figure 1 with **Building**, **Laptop**, etc. In what follows, “bigraph” will thus mean “abstract bigraph.”

### 2.4 Interfaces

Every bigraph  $G$  has two *interfaces*  $I$  and  $J$ , written  $G : I \rightarrow J$ , 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 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 [12] we write  $G : \rightarrow J$  or  $G : I \rightarrow$  for  $G : I \rightarrow J$  when we are not concerned about about  $I$  or  $J$ , respectively.

A bigraph  $G : I \rightarrow J$  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  $\text{id}_I : I \rightarrow I$  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  $x$ .

### 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 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 outer 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  $i' \leq j'$ . The bigraphs in the figures are all regular, the permutation in Table 1 is not. The virtue of regular bigraphs is that permutations can be avoided when composing them from basic bigraphs.

<sup>3</sup> Formally, we also disregard *idle* edges, those edges not connected to anything.

## 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 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  $\text{id}_e$ .

The *parallel product*  $G_1 \parallel G_2$  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  $G_1 \parallel G_2$ .

We can *compose* bigraphs  $G_2 : I \rightarrow I'$  and  $G_1 : I' \rightarrow J$ , yielding bigraph  $G_1 \circ G_2 : I \rightarrow J$ , 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 1, where  $a = C \circ (\text{id}_z \otimes R) \circ d$ . In the following, we will omit the ‘ $\circ$ ’, and simply write  $G_1 G_2$  for composition, letting it bind tighter than tensor product.

## 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 1 we could have **Data** : atomic( $0 \rightarrow 0$ ), **Folder** : passive( $0 \rightarrow 1$ ), **Laptop** : active( $0 \rightarrow 0$ ), **Building** : active( $1 \rightarrow 1$ ).

## 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  $R \rightarrow^e R'$  consists of a regular *redex*  $R : I \rightarrow J$ , a regular *reactum*  $R' : I' \rightarrow J$ , and an *instantiation*  $\varrho$ , 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 related by  $X'_i = X_{\varrho(i)}$ . We illustrate  $\varrho$  by a ‘ $i := j$ ’, as shown in Figure 1, whenever  $\varrho(i) = j \neq i$ . Given an instantiation  $\varrho$  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  $a = C(\text{id}_Z \otimes R)d$ , with active context  $C$ , discrete parameter  $d$ , and some set of names  $Z$ . Dynamics is achieved by transforming  $a$  into a new agent  $a' = C(\text{id}_Z \otimes R')d'$ , where  $d' = \varrho(d)$ —an example is shown in Figure 1. This definition of a match is as in [9], except that we here also require  $R$  to be regular. This restriction to regular redexes  $R$  (and to discrete parameters  $d$ ) does not limit the set of possible reactions. We restrict attention to regular  $R$ ’s because it

simplifies the inductive characterization of matching by allowing us to omit trivial permutations.

### 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  $Y_0 \uplus \dots \uplus Y_{n-1}$  when  $\vec{Y} = Y_0, \dots, Y_{n-1}$ , and similarly  $\{\vec{y}\}$  for  $\{y_0, \dots, y_{n-1}\}$ . For interfaces, we write  $n$  to mean  $\langle n, [\emptyset, \dots, \emptyset], \emptyset \rangle$ ,  $X$  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 1 [5]. For permutations, when used in any context,  $\pi_{\vec{X}}G$  or

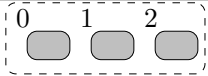
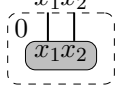
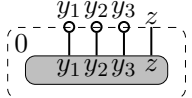
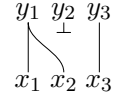
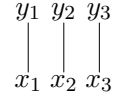
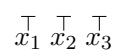
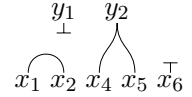
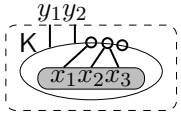
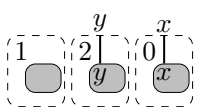
	Notation	Example
Merge	$merge_n : n \rightarrow 1$	$merge_3 =$ 
Concretion	$\lceil X \rceil : (X) \rightarrow \langle X \rangle$	$\lceil \{x_1, x_2\} \rceil =$ 
Abstraction	$(Y)P : I \rightarrow \langle 1, [Y], Z \uplus Y \rangle$	$(\{y_1, y_2\})(\{y_3\})\lceil \{y_1, y_2, y_3, z\} \rceil =$ 
Substitution	$\vec{y}/\vec{X} : X \rightarrow Y$ $\sigma$	$[y_1, y_2, y_3]/[\{x_1, x_2\}, \{\}, \{x_3\}] =$ 
Renaming	$\vec{y}/\vec{x} : X \rightarrow Y$ $\alpha, \beta$	$[y_1, y_2, y_3]/[x_1, x_2, x_3] =$ 
Closure	$/X : X \rightarrow \{\}$	$/\{x_1, x_2, x_3\} =$ 
Wiring	$(id \otimes /Z)\sigma : X \rightarrow Y$ $\omega$	$(id_{\{y_1, y_2\}} \otimes /\{z_1, z_2\}) [y_1, z_1, y_2, z_2] / [\{\}, \{x_1, x_2\}, \{x_4, x_5\}, \{x_6\}] =$ 
Ion	$K_{\vec{y}(\vec{X})} : (\{\vec{X}\}) \rightarrow \langle \{\vec{y}\} \rangle$	$K_{[y_1, y_2](\{\{x_1\}, \{x_2, x_3\}, \{\}\})} =$ 
Permutation	$\{i \mapsto j, \dots\} : \langle m, \vec{X}, X \rangle \rightarrow \langle m, \pi(\vec{X}), X \rangle$ $\pi_{\vec{X}}$	$\{0 \mapsto 2, 1 \mapsto 0, 2 \mapsto 1\}[\{x\}, \emptyset, \{y\}] =$ 

Table 1

Basic bigraphs, the abstraction operation, and variables ranging over bigraphs.

$G\pi_{\vec{X}}$ ,  $\vec{X}$  is given entirely by the interface of  $G$ ; in these cases we simply write  $\pi_{\vec{X}}$  as  $\pi$ .

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  $(\cdot)$  and reaches as far right as possible.

For a renaming  $\alpha : X \rightarrow Y$ , we write  $\ulcorner \alpha \urcorner$  to mean  $(\text{id}_1 \otimes \alpha) \ulcorner X \urcorner$ , and when  $\sigma : U \rightarrow Y$ , we let  $\hat{\sigma} = (Y)(\sigma \otimes \text{id}_1) \ulcorner U \urcorner$ . We write substitutions  $\vec{y}/[\emptyset, \dots, \emptyset] : \epsilon \rightarrow Y$  as  $Y$ .

Note that  $[\ ]/[ ] = / \emptyset = \pi_0 = \text{id}_\epsilon$  and  $\text{merge}_1 = \ulcorner \emptyset \urcorner = \pi_1 = \text{id}_1$ , where  $\pi_i$  is the nameless permutation of width  $i$ .

As an example, the bigraph of Figure 2 can be written

$$\begin{aligned} G &= (\omega \otimes ((\{y_0\})(y_0/Y \otimes \text{id}_1) \ulcorner Y \urcorner)) (((Y)P_1) \otimes P_2 \otimes y_2/x_1), \text{ where} \\ \omega &= (/e \otimes \text{id}_{\{y_1, y_2\}})[y_1, y_2, e]/[\{y_1\}, \{y_2, y'_2, y''_2\}, \{e, e'\}], \quad Y = \{y_0, y'_0, y''_0\} \\ P_1 &= (\text{id}_{\{y_0, y_1, y'_2, e\}} \otimes \text{merge}_2) ((\text{id}_{\{y_0, e\}} \otimes K_{0[y'_0]})K_{1[y_0, e]} \otimes K_{2[y'_0, y_1, y'_2]} \text{merge}_0) \\ P_2 &= (\text{id}_{\{e', y''_2\}} \otimes \text{merge}_2)(K_{3[e', y''_2]}(\{\{x_0, x_2\}\}) \otimes \ulcorner \emptyset \urcorner), \end{aligned}$$

and for Figure 1 we have  $a = (\text{id}_{\{\text{consultancy}, \text{corporation}\}} \otimes /z)(p_1 \parallel p_2)$ , where

$$\begin{aligned} p_1 &= (\text{id}_z \otimes \text{Building}_{[\text{consultancy}]}(\{\{\}\}) \text{Laptop}) \text{Folder}_{[z]} \text{Data} \text{merge}_0 \\ p_2 &= (\text{id}_z \otimes \text{Building}_{[\text{corporation}]}(\{\{y_1, y_2\}\})) (\{y_1, y_2\}) (\text{id}_{\{z, y_1, y_2\}} \otimes \text{merge}_2) (p'_2 \otimes p''_2) \\ p'_2 &= (\text{id}_{\{z, y_1\}} \otimes \text{Laptop} \text{merge}_2) (\text{Folder}_{[z]} \text{Data} \text{merge}_0 \otimes \text{Folder}_{[y_1]} \text{Data} \text{merge}_0) \\ p''_2 &= (\text{id}_{y_2} \otimes \text{Laptop}) \text{Folder}_{[y_2]} \text{Data} \text{merge}_0 \end{aligned}$$

Finally, a *molecule* is a prime with just one outermost node.

### 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 following presentation to include the active requirement).

#### 3.1 Preliminaries

In this subsection we introduce useful notation and establish some propositions about how one may decompose bigraphs. To simplify notation we shall simply write  $\text{id}$  for identity bigraphs, without a subscript showing the interface, when it is clear from the context what interface is intended.

The following propositions express how bigraphs may be decomposed into simpler constituent components. The proofs follow easily from the normal

form theorem in [5]. Note that  $\omega, \alpha, \sigma$  and  $\pi$  range over wirings, renamings, substitutions and permutations, cf. Table 1.

**Proposition 3.1** *Any bigraph  $G$  can be decomposed into a composition of the following form*

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

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

**Proposition 3.2** *Any discrete bigraph  $D$  of width  $n$  with local innerface can be decomposed such that*

$$D = \left( \bigotimes_i^n (\hat{\sigma}_i \otimes \text{id}) P_i \right) \pi,$$

where the  $P_i$ 's are name-discrete and prime. Any other decomposition on this form of  $D$  takes the form  $\left( \bigotimes_i^n (\hat{\sigma}'_i \otimes \text{id}) P'_i \right) \pi'$ , where, for some  $\hat{\alpha}_i, \rho_i$ , for all  $i$ ,  $P'_i = (\hat{\alpha}_i^{-1} \otimes \text{id}) P_i \rho_i$ ,  $\left( \bigotimes_i^n \rho_i \right) \pi' = \pi$ , and  $\hat{\sigma}'_i = \hat{\sigma}_i \hat{\alpha}_i$ .

For primes and molecules, the normal form can be found in *loc. cit.*

One can decompose binding ions  $K_{\vec{y}(\vec{x})}$  into  $K_{\vec{y}(\vec{x})} \bigotimes_i^n (u_i)/(X_i)$ . Such decompositions will be useful because of the following proposition, which is a corollary of Theorem 1, item 1, in [5] (specialized to free discrete ions).

**Proposition 3.3** *Any free discrete molecule  $M : I \rightarrow (\{\vec{y}\} \uplus Z)$  can be decomposed as*

$$M = (K_{\vec{y}(\vec{x})} \otimes \text{id}_Z) P,$$

where  $P$  is a discrete prime. Any other decomposition of  $M$  on this form, has the form  $(K_{\vec{y}(\vec{x})} \otimes \text{id}_Z) P'$ , where there exists a unique  $\hat{\alpha}$ , given by  $u_i \mapsto x_i$ , such that  $K_{\vec{y}(\vec{x})} \hat{\alpha} = K_{\vec{y}(\vec{x})}$  and  $P = (\hat{\alpha} \otimes \text{id}_Z) P'$ .

### 3.2 Matching Sentences

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

**Definition 3.4** A **matching sentence** is a 7-place relation among wirings and bigraphs, written  $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \hookrightarrow C, d$ , satisfying that  $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}}$  are wirings, and  $a, R, C, d$  are discrete bigraphs,  $R$  and  $C$  have local inner faces, and  $R$  is regular.

**Definition 3.5** 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

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

where unqualified identities are local and determined from the 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  $a' = (\text{id} \otimes \omega_{\mathbf{a}})a$ ,  $C' = (\text{id} \otimes \omega_{\mathbf{C}})(C \otimes \text{id}_Y \otimes \text{id}_Z)$ , and  $R' = (\text{id} \otimes \omega_{\mathbf{R}})R$ , then  $a' = C'(R' \otimes \text{id}_Z)d$ . Conversely, if, for general  $a', C', R', d$  we have a match  $a' = C'(R' \otimes \text{id}_Z)d$ , then by Proposition 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.

### 3.3 Rules for Matching

$$\begin{array}{c}
 \text{PERM} \frac{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, \bigotimes_i^m P_{\pi^{-1}(i)} \hookrightarrow C, (\bar{\pi} \otimes \text{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}} \parallel \omega \vdash a, R \hookrightarrow C, d \quad \omega_{\mathbf{b}}, \omega_{\mathbf{S}}, \omega_{\mathbf{D}} \parallel \omega \vdash b, S \hookrightarrow D, e}{\omega_{\mathbf{a}} \parallel \omega_{\mathbf{b}}, \omega_{\mathbf{R}} \parallel \omega_{\mathbf{S}}, \omega_{\mathbf{C}} \parallel \omega_{\mathbf{D}} \parallel \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 \quad \sigma_{\mathbf{a}} : Z \rightarrow W \quad \sigma_{\mathbf{C}} : U \rightarrow W}{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash (\widehat{\sigma}_{\mathbf{a}} \otimes \text{id})(Z)p, R \hookrightarrow (\widehat{\sigma}_{\mathbf{C}} \otimes \text{id})(U)P, d} \\
 \\
 \text{MERGE} \frac{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \hookrightarrow C, d \quad \text{a global}}{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash (\text{merge} \otimes \text{id})a, R \hookrightarrow (\text{merge} \otimes \text{id})C, d} \\
 \\
 \text{ION} \frac{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash ((\vec{v})/(\vec{X}) \otimes \text{id})p, R \hookrightarrow ((\vec{v})/(\vec{Z}) \otimes \text{id})P, d \quad \alpha = \vec{y}/\vec{u} \quad \sigma : \{\vec{y}\} \rightarrow}{\sigma \parallel \omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \sigma \alpha \parallel \omega_{\mathbf{C}} \vdash (K_{\vec{y}(\vec{X})} \otimes \text{id})p, R \hookrightarrow (K_{\vec{u}(\vec{Z})} \otimes \text{id})P, d} \\
 \\
 \text{SWITCH} \frac{\omega_{\mathbf{a}}, \text{id}_{\epsilon}, \omega_{\mathbf{C}}(\sigma \otimes \omega_{\mathbf{R}} \otimes \text{id}) \vdash p, \text{id} \hookrightarrow P, d \quad P := \langle W \uplus Y \rangle \quad \sigma : W \rightarrow U}{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash p, (\widehat{\sigma} \otimes \text{id})(W)P \hookrightarrow \ulcorner U \urcorner, d} \\
 \\
 \text{PRIME-AXIOM} \frac{\alpha : X \rightarrow \quad p : \langle X \uplus Z \rangle}{\omega, \text{id}_{\epsilon}, \omega(\alpha^{-1} \otimes \text{id}) \vdash p, \text{id} \hookrightarrow \ulcorner \alpha \urcorner, (X)p} \\
 \\
 \text{WIRING-AXIOM} \frac{}{y, Y, y/Y \vdash \text{id}_{\epsilon}, \text{id}_{\epsilon} \hookrightarrow \text{id}_{\epsilon}, \text{id}_{\epsilon}} \\
 \\
 \text{CLOSE} \frac{\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \text{id}_{Y_{\mathbf{R}}} \otimes \sigma_{\mathbf{C}} \vdash a, R \hookrightarrow C, d \quad \sigma_{\mathbf{C}} := Y \uplus Y_{\mathbf{C}} \quad \sigma_{\mathbf{R}} := U \uplus Y_{\mathbf{R}}}{(\text{id} \otimes / (Y_{\mathbf{R}} \uplus Y_{\mathbf{C}}))\sigma_{\mathbf{a}}, (\text{id} \otimes / Y_{\mathbf{R}})\sigma_{\mathbf{R}}, (\text{id} \otimes / Y_{\mathbf{C}})\sigma_{\mathbf{C}} \vdash a, R \hookrightarrow C, d}
 \end{array}$$

Fig. 3. Rules for matching binding bigraphs

In Figure 3 we present a set of rules 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 the rules.

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  $\bar{\pi}$ , depending on  $\pi$  and the innerface of the redex, as stated in the push-through lemma [5].

The PAR rule explains how to match a product, given two valid matches, which *share* some context wiring  $\omega$  if the two parts of the redex share a (necessarily global) name, cf. Figure 4.

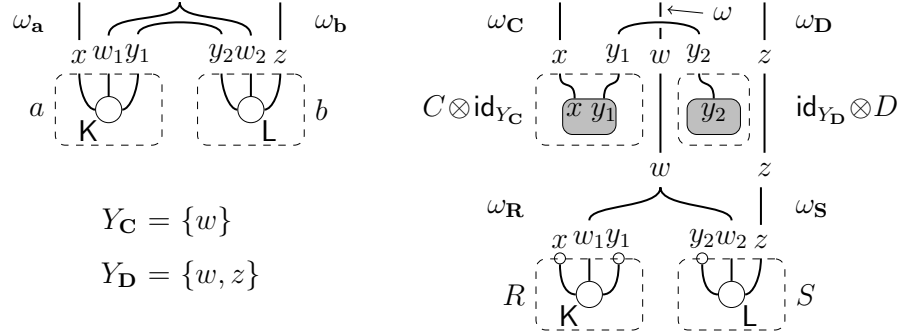


Fig. 4. Matching a product using the PAR rule

The LSUB rule allows us to match any discrete prime (c.f. Proposition 3.2) by matching an underlying *free* (name)discrete prime with the wiring of agent and context extended with the underlying global substitutions  $\sigma_a$  and  $\sigma_C$ . In other words, this rule expresses that we can match a bigraph with *local* names by matching the corresponding free bigraph (forgetting that the names are local) and then remember to make the correct names local again.

The MERGE rule simply states that to match bigraphs with an outer merge and a global id, we must be able to match the underlying bigraphs.

The ION rule works intuitively by splitting up a binding ion into a free, discrete ion and an underlying local substitution. 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 with isomorphic wiring on the outer names  $\vec{y}$  and  $\vec{u}$ ; stated in the rule by requiring that we extend with  $\sigma_y$  and  $\sigma_y \alpha$  (where  $\alpha = \vec{y}/\vec{u}$ ). For example, if we seek to match the agent  $a = (\text{id} \otimes K_{\vec{y}(\vec{X})})p$  yielding a context  $C = (\text{id} \otimes K_{\vec{u}(\vec{Z})})P$ , then it suffices to consider matching of  $a' = (\vec{v})/(\vec{X})p$  yielding a context  $C' = (\vec{v})/(\vec{Z})$ , as illustrated in Figure 5.

Given an agent and considering an inference tree operationally bottom up, the rules specify how to decompose the agent while *constructing* the corresponding context (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. Thus, when inferring a match, every rule except SWITCH can

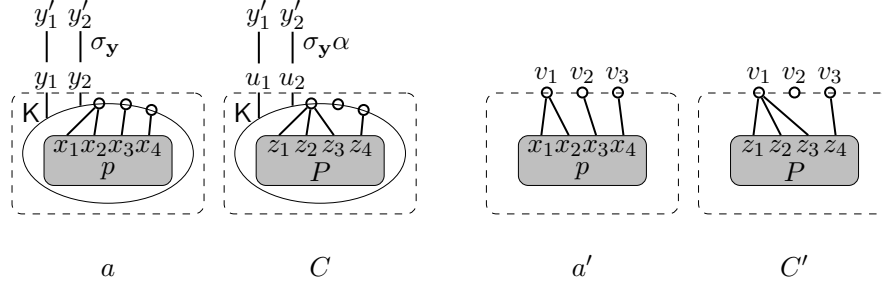


Fig. 5. Matching ion agent  $a$  yielding context  $C$  by matching  $a'$  yielding context  $C'$

be used in two modes: one where the agent and redex are given, resulting in a context and parameter; and one where the agent and context are given, resulting in a parameter.

The PRIME-AXIOM and WIRING-AXIOM axioms are our base cases and are intuitively clear (the latter is used to match bigraphs of zero width).

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 6. An internal edge in the agent need not have the same identity as its counterpart in redex or context, hence the  $\alpha$ .

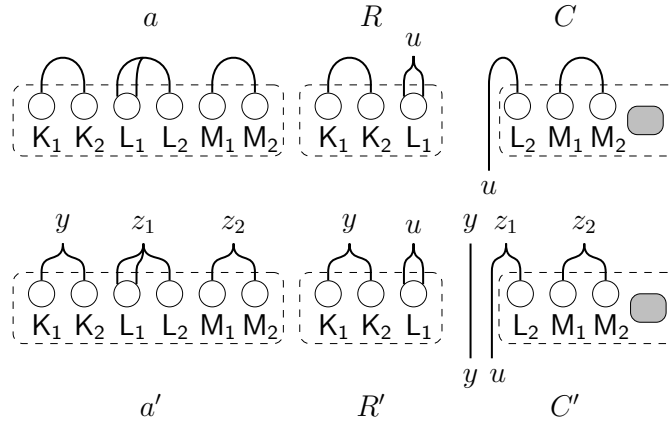


Fig. 6. Matching closed links within and between redex and context

**Theorem 3.6** *The rules for matching in Figure 3 are sound, that is, any matching sentence that can be derived is valid.*

**Proof.** Straightforward, but tedious, standard algebraic manipulations.  $\square$

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

**Definition 3.7** The size of a matching sentence  $\omega_a, \omega_R, \omega_C \vdash a, R \leftrightarrow C, d$  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 Propositions 3.1, 3.2, and 3.3, and the verifications proceed using centrally the unicity results for these normal forms and lemmas as found in [5].

**Lemma 3.8** *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 rule 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 3.9** *Every valid sentence  $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash a, R \hookrightarrow P \otimes \bigotimes_i^n P_i, d$ , with  $P$  and  $P_i$  prime and discrete, is provable using the PAR rule on valid sentences, of lesser or equal size, of the form  $\sigma_{\mathbf{a}}^P, \sigma_{\mathbf{R}}^P, \sigma_{\mathbf{C}}^P \parallel \sigma_{\mathbf{C}}^S \vDash p, S \hookrightarrow P, e$  and  $\sigma_{\mathbf{a}}^C, \sigma_{\mathbf{R}}^C, \sigma_{\mathbf{C}}^C \parallel \sigma_{\mathbf{C}}^S \vDash a', R' \hookrightarrow \bigotimes_i^n P_i, e'$ .*

**Lemma 3.10** *Every valid sentence  $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash a, R \hookrightarrow \text{id}_\epsilon, d$  is provable using PAR and WIRING-AXIOM.*

**Lemma 3.11** *Every valid sentence  $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash p, R \hookrightarrow P, d$ , 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  $\omega'_{\mathbf{a}}, \omega'_{\mathbf{R}}, \omega'_{\mathbf{C}} \vDash p', R \hookrightarrow P', d$ , where  $p'$  and  $P'$  are discrete free primes.*

**Lemma 3.12** *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 MERGE, PAR (iterated), and SWITCH rules on valid sentences, each of lesser or equal size, and each on one of two forms:*

- $\sigma'_{\mathbf{a}}, \sigma'_{\mathbf{R}}, \sigma'_{\mathbf{C}} \vDash p^N, \text{id} \hookrightarrow P^N, e$ , where  $p^n$  and  $P^N$  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.

**Lemma 3.13** *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.*

**Lemma 3.14** *Every valid sentence  $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash p, \text{id} \hookrightarrow P, e$ , 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  $\sigma'_{\mathbf{a}}, \sigma'_{\mathbf{r}}, \sigma'_{\mathbf{M}} \vDash m, R \hookrightarrow M, d$ .*

**Theorem 3.15** *The rules for matching in Figure 3 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 3.13, these can be derived from sentences of size less than  $n$ .  $\square$

## 4 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 (e.g., in Prolog) 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 characterise matching structurally and inductively in order to understand it (in particular to understand the relation to representations based on normal forms and to understand where exactly choices between different matches can be made during matching); second, to provide a point from which to begin the search for truly efficient matching algorithms, and to verify them. 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 3.8 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 choosing 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  $\parallel$  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 to also compare substitutions, allowing us to consider matching of discrete bigraphs up to renaming isos 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 [12] (where there is no global linkage but instead multilocal names) could be handled similarly, by recasting the rules to work on local links and just locating names at all roots where they occur.

#### 4.1 Representations of Graphs

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,  $\text{merge}(M_1 \otimes M_2) = \text{merge}(M_2 \otimes M_1)$ .) 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 bigraphical *expressions* (syntax), as defined in [13,5]. Doing so forces us to add an inference rule, which allows one to replace any expression in a matching sentence  $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \leftrightarrow C, d$ , say  $a$ , by another, say  $a'$ , that is provably equal via the axioms for equality in [5]. Doing so clearly yields a complete set of rules on bigraphical expressions. When defining normal inference trees for these, one seeks, of course, to restrict the application of the equality axioms. The definition of normal inference trees will then *formally explicate* all the possibilities that a matching algorithm needs to explore. We have worked out a definition of normal inference tree for matching of place graph *expressions* and proved it complete. Based on that experience, we believe it should not be too hard to work out a suitable definition of normal inference tree binding bigraph expressions and prove it complete.

## 5 Conclusion and Related Work

We have presented a sound and complete inductive characterization of matching for binding bigraphs. We are currently working toward an implementation of matching based upon the characterization.

Bigraphical reactive systems are related to graph transformations systems; see [6] 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 [17,7,10,20]. Due to the special structure of bigraphs, general GPM algorithms are expected to be inefficient, although some GPM tools [19] 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 [7] 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 [15,18,16] 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 the actual algorithm correct.

## 6 Acknowledgments

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).

## References

- [1] Birkedal, L., *Bigraphical Programming Languages—a LaCoMoCo research project*, in: *Second UK UbiNet Workshop, Cambridge*, 2004, position paper.
- [2] Birkedal, L., T. C. Damgaard, A. J. Glenstrup and R. Milner, *Matching of bigraphs — proofs of soundness and completeness*, available on request.
- [3] Birkedal, L., S. Debois, E. Elsborg, T. Hildebrandt and H. Niss, *Bigraphical models of context-aware systems*, in: L. Aceto and A. Ingólfssdóttir, editors, *FOSSACS '06: Proceedings of 9th International Conference on Foundations of Software Science and Computation Structures*, LNCS **3921** (2006), pp. 187–201.
- [4] Birkedal, L., S. Debois and T. Hildebrandt, *Sortings for reactive systems*, Technical Report 84, IT University of Copenhagen (2006).

- [5] Damgaard, T. C. and L. Birkedal, *Axiomatizing binding bigraphs*, Nordic Journal of Computing **13** (2006), pp. 58–77.
- [6] Ehrig, H., K. Ehrig, U. Prange and G. Taentzer, “Fundamentals of Algebraic Graph Transformation,” Monographs in Theoretical Computer Science. An EATCS Series, Springer, 2006.
- [7] Fu, J. J., *Directed graph pattern matching and topological embedding*, Journal of Algorithms **22** (1997), pp. 372–391.
- [8] Jensen, O. H., “Mobile Processes in Bigraphs,” Ph.D. thesis, Univ. of Cambridge (2006).
- [9] Jensen, O. H. and R. Milner, *Bigraphs and mobile processes (revised)*, Technical Report UCAM-CL-TR-580, University of Cambridge (2004).
- [10] Larrosa, J. and G. Valiente, *Constraint satisfaction algorithms for graph pattern matching*, Mathematical Structures in Computer Science **12** (2002), pp. 403–422.
- [11] Leifer, J. J. and R. Milner, *Transition systems, link graphs and Petri nets*, Technical Report UCAM-CL-TR-598, University of Cambridge (2004).
- [12] Milner, R., *Bigraphs whose names have multiple locality*, Technical Report UCAM-CL-TR-603, University of Cambridge, Computer Laboratory (2004).
- [13] Milner, R., *Axioms for bigraphical structure*, Mathematical Structures in Computer Science **15** (2005), pp. 1005–1032.
- [14] Milner, R., *Pure bigraphs: structure and dynamics*, Inf. Comput. **204** (2006), pp. 60–122.
- [15] Selkow, S. M., *The tree-to-tree editing problem*, Information Processing Letters **6** (1977), pp. 184–186.
- [16] Shamir, R. and D. Tsur, *Faster subtree isomorphism*, Journal of Algorithms **33** (1999), pp. 267–280.
- [17] Ullman, J. D., *An algorithm for subgraph isomorphism*, Journal of the ACM **23** (1976), pp. 31–42.
- [18] Valiente, G., “Algorithms on Trees and Graphs,” Springer, Berlin, 2002.
- [19] Varró, G., D. Varró and K. 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, 2005, pp. 191–205.
- [20] Zündorf, A., *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*, LNCS **1073** (1996), pp. 454–468.

## A Proofs of completeness

Below we give extensive details for the proof of Lemmas 3.8 through 3.14. To establish the proofs we shall need a number of preliminaries (extending the ones in Section 3.1), which we start by stating and proving.

### A.1 Preliminaries

First, Corollary A.1 is a simple restatement of Proposition 3.2, eliding the details of the representation of discrete primes.

**Corollary A.1** *Any discrete bigraph  $D$  with local innerface of width  $n$  can be decomposed such that*

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

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

We shall also need the following simple property of abstraction.

**Lemma A.2**

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

**Proof.**

( $\Leftarrow$ ) Immediate.

( $\Rightarrow$ ) Nearly immediate; algebraically, by composing with equal concretions:

$$P = (\ulcorner U^\top \otimes \text{id})(U)P = (\ulcorner U^\top \otimes \text{id})(U)Q = Q.$$

□

The **parallel product** of two bigraphs  $G : X \rightarrow Y \uplus Z$  and  $H : U \rightarrow V \uplus Z$ , 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.

We record a few properties of  $\parallel$ , which allows us a number of convenient algebraic manipulations. (These properties are easily proven by algebraic manipulations and with the help of the normal form for links proven in [13].) We start by giving some simple equivalent forms of the definition of the parallel product.

**Fact A.3 (Parallel product)**

$$\begin{aligned} G \parallel H &= \left( \left( \bigotimes_i^{|Z|} z_i / \{z_i, z'_i\} \right) \otimes \text{id}_Y \otimes \text{id}_V \right) (G \otimes \left( \left( \bigotimes_i^{|Z|} z'_i / z_i \right) \otimes \text{id}_V \right) H) \\ &= (\vec{z} / \vec{z}, \vec{z}' \otimes \text{id}_Y \otimes \text{id}_V) (G \otimes (\vec{z}' / \vec{z} \otimes \text{id}_V) H) \\ &= (\sigma \otimes \text{id}_Y \otimes \text{id}_V) (G \otimes (\alpha \otimes \text{id}_V) H), \end{aligned}$$

where in the second equation we introduce some shorthand notation and in the third equation  $\sigma$  and  $\alpha$  are given by the previous equations.

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

$$\omega = (\delta \otimes \text{id})(\sigma_0 \parallel \sigma_1) \quad \sigma_0 : U \rightarrow \quad \sigma_1 : V \rightarrow$$

respectively

$$\omega = \omega_0 \otimes \omega_1 \quad \omega_0 : \rightarrow X \quad \omega_1 : \rightarrow Y,$$

where  $\delta$ ,  $\sigma_0$ , and  $\sigma_1$ ; and  $\omega_0$  and  $\omega_1$  are uniquely given by  $\omega$ ,  $U$ ,  $V$ ,  $X$  and  $Y$ .

**Lemma A.4** *For all wirings  $\omega_a : \rightarrow Z \uplus Y_a$ ,  $\omega_b : \rightarrow Z \uplus Y_b$ ,  $\omega_c : \rightarrow X \uplus Y_c$ , and  $\omega_d : \rightarrow X \uplus Y_d$ , with all inner faces mutually disjoint and such that  $(Y_A \cup Y_D) \cap (Y_B \cup Y_C) = \emptyset$ , it holds that*

$$(\omega_a \parallel \omega_b) \otimes (\omega_c \parallel \omega_d) = (\omega_a \otimes \omega_c) \parallel (\omega_b \otimes \omega_d).$$

**Proof.** Algebraically,

$$\begin{aligned} (\omega_a \parallel \omega_b) \otimes (\omega_c \parallel \omega_d) &= (\sigma_Z \otimes \text{id})(\omega_a \otimes (\alpha_Z \otimes \text{id})\omega_b) \otimes (\sigma_X \otimes \text{id})(\omega_c \otimes (\alpha_X \otimes \text{id})\omega_d) \\ &= (\sigma_Z \otimes \text{id})(\omega_a \otimes (\alpha_Z \omega_b^Z \otimes \omega_b')) \otimes (\sigma_X \otimes \text{id})(\omega_c \otimes (\alpha_X \omega_d^X \otimes \omega_d')) \\ &= (\sigma_Z \otimes \sigma_X \otimes \text{id})(\omega_a \otimes \omega_c \otimes (\alpha_Z \omega_b^Z \otimes \alpha_X \omega_d^X \otimes \omega_d' \otimes \omega_b')) \\ &= (\omega_a \otimes \omega_c) \parallel (\omega_b \otimes \omega_d). \end{aligned}$$

— where the first and the last equality are instances of the third equation of Fact A.3, and  $\omega_b^Z$ ,  $\omega_b'$ ,  $\omega_d^X$ , and  $\omega_d'$  arises from splitting  $\omega_b$  and  $\omega_d$  by the outerface.  $\square$

**Lemma A.5** *For wirings  $\omega^i = X^i \rightarrow Y^i \uplus Z$  ( $i = 0, 1$ ),*

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

**Proof.** Algebraically,

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

— where we get the third equality by using the third equation of Fact A.3 splitting  $\omega^1$  by its outerface and rearranging. We split by  $Z$  and  $Y^1$  — producing another copy of the names  $Z$ ,  $Z'$ , and  $\sigma_Z$  (the substitution as defined in Fact A.3 that joins these, again). The fourth equality resolves the composition (equationally, by repeated application of axiom L4).  $\square$

**Lemma A.6** *Given  $\omega_c^i, \omega_a^i : X^i \rightarrow Y^i \uplus Z$ , ( $i = 0, 1$ ) we have that*

$$\omega_c^0 \parallel \omega_c^1 = \omega_a^0 \parallel \omega_a^1 \quad \text{iff} \quad \omega_c^0 = \omega_a^0 \quad \text{and} \quad \omega_c^1 = \omega_a^1.$$

**Proof.**

( $\Leftarrow$ ) Immediate.

( $\Rightarrow$ ) We have,

$$\begin{aligned} (/Y^0 \otimes \text{id}_{Y^1})(\omega_c^0 \parallel \omega_c^1)(X^0 \otimes \text{id}_{X^1}) &= (/Y^0 \otimes \text{id}_{Y^1})(Y^0 \otimes \omega_c^1) = \omega_c^1, \text{ and} \\ (/Y^0 \otimes \text{id}_{Y^1})(\omega_a^0 \parallel \omega_a^1)(X^0 \otimes \text{id}_{X^1}) &= (/Y^0 \otimes \text{id}_{Y^1})(Y^0 \otimes \omega_a^1) = \omega_a^1. \end{aligned}$$

— and analogously for  $\omega_c^1$  and  $\omega_a^1$ .  $\square$

We record a few further convenient properties of the interplay of substitutions with parallel product.

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

$$\sigma \parallel \sigma\alpha = \sigma(\text{id} \parallel \alpha), \quad (\text{A.1})$$

$$\sigma(\omega_0 \parallel \omega_1) = \sigma\omega_0 \parallel \sigma\omega_1, \quad (\text{A.2})$$

$$(\sigma_0 \parallel \sigma_1 \parallel \sigma_2)(\omega_0 \parallel \omega_1) = (\sigma_0 \parallel \sigma_2)\omega_0 \parallel (\sigma_1 \parallel \sigma_2)\omega_1. \quad (\text{A.3})$$

**Proof.**

(A.1) By definition of  $\parallel$  and normal form for  $\sigma$  (see, [5]).

(A.2) Follows easily from (A.1).

(A.3) Immediate from the earlier properties.  $\square$

We end these preliminaries by giving a number of convenient equivalent forms for Definition 3.5 of valid matching sentences. Both are simple algebraic 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 3.5.

**Fact A.8 (Valid matching sentence — with equivalent forms)** *A matching sentence  $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \hookrightarrow C, d$ , where  $\omega_{\mathbf{R}} : \rightarrow Y$ , for  $C$  with global outer names  $V$ , and  $d$  with global outer names  $Z$ , is **valid**, denoted  $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash a, R \hookrightarrow C, d$ , iff*

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

where unqualified identities are local and determined from the context.

## A.2 Proofs of Lemmas 3.8 through 3.14

In this section we give the proofs for the lemmas which support the main theorem on completeness, i.e., Theorem 3.15. For ease of reading we repeat the lemmas immediately before each proof.

**Note 1** In the following proofs, we proceed algebraically as allowed by the axiomatization of binding bigraphs [5].

In particular, in manipulating terms, we shall need to introduce quite a lot of  $\text{id}$ 's on interfaces determinable from the context. As usual, we leave out the interface, when they are determinable, and nonessential (being, e.g., eliminated in the next step). Sometimes we like to keep track of these identities, though (e.g., for allowing the reader to easier parse and check a term). In some cases, it is inconvenient to give names to all interfaces, and we shall adopt a convention of introducing an identity as  $\text{id}_G$ , when it is determined by an interface (inner- or outer-) of a term  $G$ . This is simply a naming convention, and has no bearing on the results.

We start by proving two sublemmas, of which Lemma 3.8 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 A.9** *Every valid sentence  $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash a, R \hookrightarrow C, d$  is a consequence by PERM on a valid sentence, of equal size, of the form  $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash a, S \hookrightarrow \bigotimes_i^n Q_i, e$ , where each  $Q_i$  is prime (and discrete).*

**Proof.** By Fact A.8 and Corollary A.1,  $C$  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 [5] we find by standard manipulations

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

Choosing  $S = \bigotimes_i^m P_{\pi^{-1}(i)}$  and  $e = (\bar{\pi} \otimes \text{id}_Z)d$  by Fact A.8 we have a valid sentence  $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash a, S \hookrightarrow \bigotimes_i^n Q_i, e$ , which taken as the premise in the PERM yields the required sentence as conclusion.  $\square$

**Lemma A.10** *Every valid sentence  $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash a, R \hookrightarrow C, d$  is a consequence by CLOSE on a valid sentence, of equal size, of the form  $\sigma'_{\mathbf{a}}, \sigma'_{\mathbf{R}}, \sigma'_{\mathbf{C}} \vDash a, R \hookrightarrow C, d$ .*

**Proof.** We may write  $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}$  and  $\omega_{\mathbf{C}}$  as (by the normal form for linkage (see [13]))

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

Given a valid sentence as above, by Proposition 3.1 and Fact A.8, there exists a renaming  $\beta$ , s.t.,

$$a = (\beta^{-1} \otimes \text{id})((C \otimes \text{id}_{\mathbf{R}})R \otimes \text{id}_{\mathbf{d}})d, \quad (\text{A.4})$$

$$\begin{aligned} \text{and } (\text{id} \otimes /Y_{\mathbf{a}})\sigma_{\mathbf{a}} &= (\text{id} \otimes /Y_{\mathbf{C}})\sigma_{\mathbf{C}}((\text{id} \otimes /Y_{\mathbf{R}})\sigma_{\mathbf{R}} \otimes \text{id}_{C \otimes d})\beta \\ &= (\text{id} \otimes /Y_{\mathbf{C}} \otimes /Y_{\mathbf{R}})(\sigma_{\mathbf{C}} \otimes \text{id}_{Y_{\mathbf{R}}})(\sigma_{\mathbf{R}} \otimes \text{id}_{C \otimes d})\beta. \end{aligned} \quad (\text{A.5})$$

Further, we have  $|Y_{\mathbf{a}}| = |Y_{\mathbf{R}}| + |Y_{\mathbf{C}}|$ ; as, in particular, the number of free edges in agent and context composed with redex and parameter, must be equal —  $d$ ,  $C$  and  $R$  are discrete, the free edges must be created entirely by  $\omega_{\mathbf{R}}$  and  $\omega_{\mathbf{C}}$ . Hence, working from (A.5) we see easily, that there exists an isomorphism  $\alpha : Y_{\mathbf{a}} \rightarrow Y_{\mathbf{R}} \uplus Y_{\mathbf{C}}$ , s.t.

$$(\text{id} \otimes \alpha)\sigma_{\mathbf{a}} = (\text{id} \otimes \text{id}_{Y_{\mathbf{C}}} \otimes \text{id}_{Y_{\mathbf{R}}})(\sigma_{\mathbf{C}} \otimes \text{id}_{Y_{\mathbf{R}}})(\sigma_{\mathbf{R}} \otimes \text{id}_{C \otimes d})\beta. \quad (\text{A.6})$$

Combining (A.4) and (A.6), we find

$$(\text{id}(\text{id} \otimes \alpha)\sigma_{\mathbf{a}})a = (\text{id} \otimes (\sigma_{\mathbf{C}} \otimes \text{id}_{Y_{\mathbf{R}}})(\sigma_{\mathbf{R}} \otimes \text{id}_{C \otimes d}))((C \otimes \text{id}_{\mathbf{R}})R \otimes \text{id}_{\mathbf{d}})d,$$

which by Fact A.8 means that the sentence

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

is valid. This sentence is on the required form, and, checking, we see that applying CLOSE to this sentence, we arrive at (dissolving  $\alpha$ )

$$(\text{id} \otimes /Y_{\mathbf{a}})\sigma_{\mathbf{a}}, (\text{id} \otimes /Y_{\mathbf{R}})\sigma_{\mathbf{R}}, (\text{id} \otimes /Y_{\mathbf{C}})\sigma_{\mathbf{C}} \vDash a, R \hookrightarrow C, d,$$

i.e., the original sentence, with the linkage in normal form.  $\square$

**Lemma (3.8)** *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 rule 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 A.10 and Lemma A.9.  $\square$

**Lemma (3.9)** *Every valid sentence  $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash a, R \hookrightarrow P \otimes \bigotimes_i^n P_i, d$ , with  $P$  and  $P_i$  prime and discrete, is provable using the PAR rule on valid sentences, of lesser or equal size, of the form  $\sigma_{\mathbf{a}}^P, \sigma_{\mathbf{R}}^P, \sigma_{\mathbf{C}}^P \parallel \sigma_{\mathbf{C}}^S \vDash p, S \hookrightarrow P, e$  and  $\sigma_{\mathbf{a}}^C, \sigma_{\mathbf{R}}^C, \sigma_{\mathbf{C}}^C \parallel \sigma_{\mathbf{C}}^S \vDash a', R' \hookrightarrow \bigotimes_i^n P_i, e'$ .*

**Proof.** By Fact A.8 and Corollary A.1,  $a$  can be decomposed as  $\bigotimes_i^n p_i$  (with discrete and prime  $p_i$ ); we immediately utilize that by validity of the original sentence, in particular, the width of  $a$  and  $\bigotimes_i^n P_i$  must be equal.

Hence, given the validity of the original sentence, we have (introducing  $C = \bigotimes_{i=1}^n P_i$ )

$$(\text{id} \otimes \sigma_{\mathbf{a}}) \bigotimes_i^n p_i = (\text{id} \otimes \sigma_{\mathbf{C}}(\sigma_{\mathbf{R}} \otimes \text{id}_{\mathbf{d}} \otimes \text{id}_{\mathbf{P}_0} \otimes \text{id}_{\mathbf{C}}))((P_0 \otimes C \otimes \text{id}_{\mathbf{R}})R \otimes \text{id}_{\mathbf{d}})d.$$

By Proposition 3.1 there exists a renaming  $\alpha$ , s.t.,

$$\sigma_{\mathbf{a}} = \sigma_{\mathbf{C}}(\sigma_{\mathbf{R}} \otimes \text{id}_{\mathbf{d}} \otimes \text{id}_{\mathbf{P}_0} \otimes \text{id}_{\mathbf{C}})\alpha, \quad (\text{A.7})$$

$$\text{and } \bigotimes_i^n p_i = (\alpha^{-1} \otimes \text{id})((P_0 \otimes C \otimes \text{id}_{\mathbf{R}})R \otimes \text{id}_{\mathbf{d}})d. \quad (\text{A.8})$$

By the normal form for discrete bigraphs, we also have, for discrete primes  $R'_i$  and  $d'_i$ ;  $R = \bigotimes_i^{m'} R'_i$  and  $d = \bigotimes_i^{l'} d'_i$ . Hence, we can partition the discrete primes of the redex and parameter according to the innerfaces of  $P_0$  and  $\bigotimes_i^n P_i$ . Introducing a few more convenient metavariables, let  $R = \bigotimes_i^m R_i = R_0 \otimes R'$  and  $d = \bigotimes_i^l d_i = d_0 \otimes d'$  (where each  $R_i$  and  $d_i$  is *not* (in general) prime).

Further, we also split  $\alpha$  and  $\sigma_{\mathbf{R}}$  by the outerface of  $P_i$ 's,  $R_i$ 's, and  $d_i$ 's, to get from (A.7):

$$\sigma_{\mathbf{a}} = \sigma_{\mathbf{C}}((\sigma_{\mathbf{R}_0} \alpha_{\mathbf{R}_0} \parallel \sigma_{\mathbf{R}'} \alpha_{\mathbf{R}'}) \otimes \alpha_{\mathbf{d}_0} \otimes \alpha_{\mathbf{d}'} \otimes \alpha_{\mathbf{P}_0} \otimes \alpha_{\mathbf{C}}), \quad (\text{A.9})$$

and from (A.8), by standard manipulations we derive

$$\begin{aligned} \bigotimes_i^n p_i &= (\alpha^{-1} \otimes \text{id})(((P_0 \otimes \text{id}_{\mathbf{R}_0})R_0 \otimes \text{id}_{\mathbf{d}_0})d_0 \otimes ((C \otimes \text{id}_{\mathbf{R}'})R' \otimes \text{id}_{\mathbf{d}'})d') \\ &= ((\alpha_{\mathbf{P}_0}^{-1} P_0 \otimes \alpha_{\mathbf{R}_0}^{-1})R_0 \otimes \alpha_{\mathbf{d}_0}^{-1})d_0 \otimes \\ &\quad \left[ \left( \left( \left( \bigotimes_{i=1}^n \alpha_{i,\mathbf{C}}^{-1} P_i \right) \otimes \text{id}_{\mathbf{R}'} \otimes \text{id}_{\mathbf{D}'} \right) \left( \bigotimes_{i=1}^m \alpha_{i,\mathbf{R}}^{-1} R_i \right) \otimes \text{id}_{\mathbf{d}'} \right) \left( \bigotimes_{i=1}^l \alpha_{i,\mathbf{d}'}^{-1} d_i \right) \right] \\ &= ((\alpha_{\mathbf{P}_0}^{-1} P_0 \otimes \alpha_{\mathbf{R}_0}^{-1})R_0 \otimes \alpha_{\mathbf{d}_0}^{-1})d_0 \otimes \\ &\quad \bigotimes_{i=1}^n \left[ (\alpha_{i,\mathbf{C}}^{-1} \otimes \text{id}) \left( \left( \bigotimes_j^{m^i} S_j^i \right) \otimes \text{id} \right) \bigotimes_j^{l^i} e_j^i \right], \end{aligned}$$

introducing  $\bigotimes_{i=1}^m \alpha_{\mathbf{R}_0}^{-1} R_i = \bigotimes_{i=1}^n \bigotimes_j^{m^i} S_j^i$  and  $\bigotimes \alpha_{i,\mathbf{d}'}^{-1} d_i = \bigotimes_{i=1}^n \bigotimes_j^{l^i} e_j^i$ .

By Corollary A.1, as the bigraphs are ground (i.e., there are no permutations to consider), we find

$$p_0 = ((\alpha_{\mathbf{P}_0}^{-1} P_0 \otimes \alpha_{\mathbf{R}_0}^{-1})R_0 \otimes \alpha_{\mathbf{d}_0}^{-1})d_0 \quad (\text{A.10})$$

$$\begin{aligned} \text{and } \bigotimes_{i=1}^n p_i &= \bigotimes_{i=1}^n \left[ (\alpha_{i,\mathbf{C}}^{-1} \otimes \text{id}) \left( \left( \bigotimes_j^{m^i} S_j^i \right) \otimes \text{id} \right) \bigotimes_j^{l^i} e_j^i \right] \\ &= (\alpha_{\mathbf{C}}^{-1} \otimes \alpha_{\mathbf{d}'}^{-1} \otimes \alpha_{\mathbf{R}'}^{-1} \otimes \text{id}) \left( \bigotimes_{i=1}^n \text{id}_{\mathbf{R}'} \right) R \otimes \text{id}_{\mathbf{d}'} d', \quad (\text{A.11}) \end{aligned}$$

introducing  $\alpha_{\mathbf{C}}^{-1}, \alpha_{\mathbf{d}'}^{-1}$  and  $\alpha_{\mathbf{R}'}^{-1}$  as metavariables for the (equally named) products of renamings used above.

Now we split  $\sigma_a$  and  $\sigma_C$  by their innerfaces according to their underlying components; we define

$$\begin{aligned} \sigma_a &= \sigma_a^{\mathbf{P}_0} \parallel \sigma_a^{\mathbf{R}} \parallel \sigma_a^{\mathbf{d}_0} \parallel \sigma_a^{\mathbf{C}} \parallel \sigma_a^{\mathbf{d}'}, \\ \text{and } \sigma_C &= \sigma_C^{\mathbf{P}_0} \parallel \sigma_C^{\mathbf{R}} \parallel \sigma_C^{\mathbf{d}_0} \parallel \sigma_C^{\mathbf{C}} \parallel \sigma_C^{\mathbf{d}'}. \end{aligned}$$

By (A.9) we have:

$$\begin{aligned} \sigma_a^{\mathbf{P}_0} \parallel \sigma_a^{\mathbf{R}} \parallel \sigma_a^{\mathbf{d}_0} \parallel \sigma_a^{\mathbf{C}} \parallel \sigma_a^{\mathbf{d}'} &= (\sigma_C^{\mathbf{P}_0} \parallel \sigma_C^{\mathbf{R}} \parallel \sigma_C^{\mathbf{d}_0} \parallel \sigma_C^{\mathbf{C}} \parallel \sigma_C^{\mathbf{d}'}) \circ \\ &\quad ((\sigma_{\mathbf{R}_0} \alpha_{\mathbf{R}_0} \parallel \sigma_{\mathbf{R}'} \alpha_{\mathbf{R}'}) \otimes \alpha_{\mathbf{d}_0} \otimes \alpha_{\mathbf{d}'} \otimes \alpha_{\mathbf{P}_0} \otimes \alpha_{\mathbf{C}}) \\ &= \sigma_C^{\mathbf{P}_0} \alpha_{\mathbf{P}_0} \parallel \sigma_C^{\mathbf{R}} (\sigma_{\mathbf{R}_0} \alpha_{\mathbf{R}_0} \parallel \sigma_{\mathbf{R}'} \alpha_{\mathbf{R}'}) \parallel \sigma_C^{\mathbf{d}_0} \alpha_{\mathbf{d}_0} \parallel \sigma_C^{\mathbf{C}} \alpha_{\mathbf{C}} \parallel \sigma_C^{\mathbf{d}'} \alpha_{\mathbf{d}'}. \end{aligned}$$

We note that by Lemma A.6, as each pair of wirings have equal interfaces, they are equal.

We now consider the substitution working on the redex stemming from the context,  $\sigma_C^{\mathbf{R}}$ , which needs a little extra care.

We have, by the arguments above, in particular,

$$\sigma_a^{\mathbf{R}} = \sigma_C^{\mathbf{R}} (\sigma_{\mathbf{R}_0} \alpha_{\mathbf{R}_0} \parallel \sigma_{\mathbf{R}'} \alpha_{\mathbf{R}'}).$$

Splitting  $\sigma_C^{\mathbf{R}}$  by the underlying wirings, we get (as, in general,  $\sigma_{\mathbf{R}_0}$  and  $\sigma_{\mathbf{R}'}$  can share names),

$$\sigma_a^{\mathbf{R}} = (\sigma_C^{\mathbf{R}_0} \parallel \sigma_C^{\mathbf{R}'}) (\sigma_{\mathbf{R}_0} \alpha_{\mathbf{R}_0} \parallel \sigma_{\mathbf{R}'} \alpha_{\mathbf{R}'}).$$

For ease of notation, we break our metavariable conventions for substitutions temporarily, and introduce  $\phi_C^{\mathbf{R}} = \sigma_C^{\mathbf{R}_0} \parallel \sigma_C^{\mathbf{R}'}$  and  $\psi_C^{\mathbf{R}} = \sigma_C^{\mathbf{R}'}$ .

Then, applying Lemma A.7(A.2), we have

$$\sigma_a^{\mathbf{R}} = (\phi_C^{\mathbf{R}} \sigma_{\mathbf{R}_0} \alpha_{\mathbf{R}_0} \parallel \psi_C^{\mathbf{R}} \sigma_{\mathbf{R}'} \alpha_{\mathbf{R}'}). \quad (\text{A.12})$$

Splitting  $\sigma_a^{\mathbf{R}}$  (accordingly) by its innerface, we get (again applying Lemma A.6),

$$\sigma_a^n R z = \phi_C^{\mathbf{R}} \sigma_{\mathbf{R}_0} \alpha_{\mathbf{R}_0}, \quad (\text{A.13})$$

$$\sigma_a^n R x = \psi_C^{\mathbf{R}} \sigma_{\mathbf{R}'} \alpha_{\mathbf{R}'}. \quad (\text{A.14})$$

Finally, we are set to utilize what we have learnt from these somewhat tedious symbol manipulations; from A.10 and (A.13):

$$\begin{aligned} (\sigma_a^{\mathbf{P}_0} \parallel \sigma_a^{\mathbf{R}_0} \parallel \sigma_a^{\mathbf{d}_0} \otimes \text{id}) p_0 &= ((\sigma_C^{\mathbf{P}_0} \alpha_{\mathbf{P}_0} \parallel \phi_C^{\mathbf{R}} \sigma_{\mathbf{R}_0} \alpha_{\mathbf{R}_0} \parallel \sigma_C^{\mathbf{d}_0} \alpha_{\mathbf{d}_0}) \otimes \text{id}) \circ \\ &\quad ((\alpha_{\mathbf{P}_0}^{-1} P_0 \otimes \alpha_{\mathbf{R}_0}^{-1}) R_0 \otimes \alpha_{\mathbf{d}_0}^{-1}) d_0 \\ &= ((\sigma_C^{\mathbf{P}_0} \parallel \phi_C^{\mathbf{R}} \parallel \sigma_C^{\mathbf{d}_0}) (\sigma_{\mathbf{R}_0} \text{id}_{\mathbf{P}_0} \text{id}_{\mathbf{d}_0}) \otimes \text{id}) \circ \\ &\quad ((P_0 \otimes \text{id}_{\mathbf{R}_0}) R_0 \otimes \text{id}_{\mathbf{d}_0}) d_0 \end{aligned}$$

Analogous manipulations from (A.11) and (A.14) ensures us that we have valid sentences

$$\sigma_C^{\mathbf{P}_0} \parallel \sigma_a^{\mathbf{R}_0} \parallel \sigma_a^{\mathbf{d}_0}, \sigma_{\mathbf{R}_0}, \sigma_C^{\mathbf{P}_0} \parallel \sigma_C^{\mathbf{d}_0} \parallel \sigma_C^{\mathbf{R}_0} \parallel \sigma_C^{\mathbf{R}'_0} \models p_0, R_0 \hookrightarrow P_0, d_0,$$

and

$$\sigma_a^{\mathbf{C}} \parallel \sigma_a^{\mathbf{R}'} \parallel \sigma_a^{\mathbf{d}'}, \sigma_{\mathbf{R}'}, \sigma_C^{\mathbf{C}} \parallel \sigma_C^{\mathbf{d}'} \parallel \sigma_C^{\mathbf{R}'} \parallel \sigma_C^{\mathbf{R}'_0} \models \bigotimes_{i=1}^n p_i, R' \hookrightarrow \bigotimes_{i=1}^n Q_i, d',$$

which by PAR yields the original sentence.  $\square$

**Lemma (3.10)** *Every valid sentence  $\sigma_a, \sigma_{\mathbf{R}}, \sigma_C \models a, R \hookrightarrow \text{id}_e, d$  is provable using PAR and WIRING-AXIOM.*

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

*Agent* By Fact A.8, the agent is expressible as  $(\sigma_{\mathbf{a}} \otimes \text{id}_\epsilon)\text{id}_\epsilon$ , hence  $\sigma_{\mathbf{a}} = Y$  for some  $Y$ .

*Context* Equally, by Fact A.8, the context is  $(\sigma_{\mathbf{C}} \otimes \text{id}_\epsilon)(\text{id}_\epsilon \otimes \text{id}_U)$  (for some names  $U$  stemming from the redex (wiring)). Hence,  $\sigma_{\mathbf{C}} : U \rightarrow Y$ .

*Redex* And finally, also by Fact A.8, redex is  $(\sigma_{\mathbf{R}} \otimes \text{id}_\epsilon)\text{id}_\epsilon$ , hence  $\sigma_{\mathbf{R}} : \epsilon \rightarrow U = U$ .

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

$$Y, U, \sigma_{\mathbf{C}} \vDash \text{id}_\epsilon, \text{id}_\epsilon \hookrightarrow \text{id}_\epsilon, \text{id}_\epsilon \quad (\sigma_{\mathbf{C}} : U \rightarrow Y).$$

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.  $\square$

**Note 2** Iterating Lemma 3.9 allows us to break any product of discrete primes in context and agent into prime parts, resulting in sentences of the form

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

for (discrete) primes  $p, P$ . Lemma 3.10 handles the base case, where there are no primes in the context.

**Lemma (3.11)** *Every valid sentence  $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash p, R \hookrightarrow P, d$ , 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  $\omega'_a, \omega'_R, \omega'_C \vDash p', R \hookrightarrow P', d$ , where  $p'$  and  $P'$  are discrete free primes.*

**Proof.** We have (by Fact A.8 and Proposition 3.2),

$$p = (\widehat{\sigma}_{\mathbf{q}} \otimes \text{id}_Y)(Z)q, \quad \text{and} \quad P = (\widehat{\sigma}_{\mathbf{Q}} \otimes \text{id}_V)(X)Q,$$

where  $q$  and  $Q$  are free primes.

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

$$\begin{aligned} (\omega_{\mathbf{a}} \otimes \text{id})(\widehat{\sigma}_{\mathbf{q}} \otimes \text{id}_Y)(Z)p &= (\omega_{\mathbf{C}} \otimes \text{id})(((\widehat{\sigma}_{\mathbf{Q}} \otimes \text{id}_V)(X)Q) \otimes \omega_{\mathbf{R}})R \otimes \text{id}_Z)d \\ \iff (\omega_{\mathbf{a}} \otimes \text{id})(U)(\sigma_{\mathbf{q}} \otimes \text{id}_Y \otimes \text{id}_1)p &= (\omega_{\mathbf{C}} \otimes \text{id}) \circ \\ &\quad (((U)(\sigma_{\mathbf{Q}} \otimes \text{id}_V \otimes \text{id}_1)Q) \otimes \omega_{\mathbf{R}})R \otimes \text{id}_Z)d \\ \iff (U)(\omega_{\mathbf{a}} \otimes \sigma_{\mathbf{q}} \otimes \text{id}_1)p &= (U)(\omega_{\mathbf{C}} \otimes \sigma_{\mathbf{Q}} \otimes \text{id}_1)((Q \otimes \omega_{\mathbf{R}})R \otimes \text{id}_Z)d, \end{aligned}$$

assuming  $\widehat{\sigma}_{\mathbf{q}}$  (hence, necessarily also  $\widehat{\sigma}_{\mathbf{Q}}$ ) has outerface  $(U)$ .

Now, by Lemma A.2, we also have

$$(\omega_{\mathbf{a}} \otimes \sigma_{\mathbf{q}} \otimes \text{id}_1)p = (\omega_{\mathbf{C}} \otimes \sigma_{\mathbf{Q}} \otimes \text{id}_1)((Q \otimes \omega_{\mathbf{R}})R \otimes \text{id}_Z)d.$$

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

**Lemma (3.12)** *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 MERGE, PAR (iterated), and SWITCH rules on valid sentences, each of lesser or equal size, and each on one of two forms:*

- $\sigma'_a, \sigma'_R, \sigma'_C \vDash p^N, \text{id} \hookrightarrow P^N, e$ , where  $p^n$  and  $P^N$  are free discrete primes,
- $\sigma'_a, \sigma'_R, \sigma'_C \vDash m, S \hookrightarrow M, e$ , where  $m$  and  $M$  are free discrete molecules.

**Proof.** By Fact A.8, and the propositions on normal forms (for named discrete primes, [5, Theorem 1(2)] and for discrete, Proposition 3.2), we see that,

$$p = (\text{merge} \otimes \text{id}) \bigotimes_i^k m_i,$$

$$Q = (\text{merge} \otimes) \left( \left( \left( \bigotimes_i^n \alpha_i^\neg \right) \otimes \bigotimes_i^m M_i \right) \pi, \right.$$

$$\text{and } R = \bigotimes_i^l P_i,$$

as  $R$  is regular.

Then by Fact A.8 and Proposition 3.1, there exists a renaming  $\beta$ , s.t.,

$$\sigma_{\mathbf{a}} = \sigma_{\mathbf{C}}(\sigma_{\mathbf{R}} \otimes \text{id}_{\mathbf{C}} \otimes \text{id}_{\mathbf{d}})\beta^{-1}, \quad (\text{A.15})$$

$$(\text{merge} \otimes \text{id}) \bigotimes_i^k m_i = (\beta \otimes \text{id}) \circ \quad (\text{A.16})$$

$$(\text{merge} \otimes \text{id}) \left( \left( \left( \left( \left( \bigotimes_i^n \alpha_i^\neg \right) \otimes \bigotimes_i^m M_i \right) \pi \otimes \text{id}_{\mathbf{R}} \right) R \otimes \text{id}_{\mathbf{d}} \right) d. \right.$$

Splitting  $\beta$  by  $\ulcorner \alpha_i^\neg$  and  $M_i$  we get from (A.16) (applying the push-through-lemma [5]),

$$\begin{aligned} & (\text{merge} \otimes \text{id}) \bigotimes_i^k m_i \\ &= (\text{merge} \otimes \text{id}) \left( \left( \left( \left( \left( \bigotimes_i^n \beta_i^{\mathbf{C}} \alpha_i^\neg \right) \otimes \bigotimes_i^m (\beta_i^{\mathbf{M}} \otimes \text{id}_1) M_i \right) \bigotimes_i^k P_{\pi^{-1}(i)} \otimes \text{id}_{\mathbf{d}} \right) \bar{\pi} d \right. \\ &= (\text{merge} \otimes \text{id}) \left( \left( \left( \bigotimes_i^n \beta_i^{\mathbf{C}} \alpha_i^\neg P_{\pi^{-1}(i)} \right) \otimes \left( \bigotimes_i^m (\beta_i^{\mathbf{M}} \otimes \text{id}_1) M_i S_i \right) \otimes \text{id}_{\mathbf{d}} \right) \bar{\pi} d \right. \\ &= (\text{merge} \otimes \text{id}) \left( \left( \bigotimes_i^n (\ulcorner \beta_i^{\mathbf{C}} \alpha_i^\neg P_{\pi^{-1}(i)} \otimes \text{id}) e_i \right) \otimes \left( \bigotimes_i^m (\beta_i^{\mathbf{M}} \otimes \text{id}_1) M_i S_i \otimes \text{id} \right) f_i, \right. \end{aligned} \quad (\text{A.17})$$

letting in the second step,

$$\bigotimes_i^m S_i = \bigotimes_{i=n}^k P_{\pi^{-1}(i)},$$

and in the third step,

$$\left( \bigotimes_i^n e_i \right) \otimes \bigotimes_i^m f_i = \bar{\pi} d.$$

Each  $e_i$ ,  $S_i$  and  $f_i$  are determined by the innerfaces of the corresponding  $P_{\pi^{-1}(i)}$ 's,  $M_i$ 's and  $S_i$ 's, respectively.

In general, for ground bigraphs  $g$  and  $h$ ,  $(\text{merge} \otimes \text{id})g = (\text{merge} \otimes \text{id})h$  iff  $g = \pi'h$  (for some permutation  $\pi'$ ), iff  $g$  and  $h$  can be written as tensor products  $\bigotimes_i^{n_g} g_i$  and  $\bigotimes_i^{n_h} h_i$ , where each  $g_i$  and  $h_i$  are equal molecules. (This property follows from the normal form theorem of *loc.cit.*)

Since both sides of (A.17) are ground the property applies. But, we do not want to break the redex and parameter entirely into molecules in this step, so we rewrite on the lefthand side by a permutation, and define,

$$(\rho \otimes \text{id}) \bigotimes_i^k m_i = \left( \bigotimes_i^n p_i \right) \otimes \bigotimes_i^m n_i, \quad (\text{A.18})$$

for some permutation  $\rho$ , s.t.,  $p_i = \text{merge}(\bigotimes_j^{n'} m_j)$  (for some  $n'$ ), and  $n_i = m_{k-m+i}$ .

Combining (A.17) and (A.18), we find,

$$\begin{aligned} (\forall i \in n) p_i &= (\ulcorner \beta_i^{\mathbf{C}} \alpha_i \urcorner P_{\pi^{-1}(i)} \otimes \text{id}) e_i \\ &= (\ulcorner \beta_i^{\mathbf{C}} \alpha_i \urcorner \otimes \text{id}_1) ((\ulcorner U_i \urcorner \otimes \text{id}) (\widehat{\alpha}_i \otimes \text{id}) P_{\pi^{-1}(i)} \otimes \text{id}) e_i, \end{aligned} \quad (\text{A.19})$$

$$\text{and } (\forall i \in m) n_i = (\beta_i^{\mathbf{M}} \otimes \text{id}) (M_i S_i \otimes \text{id}) f_i \quad (\text{A.20})$$

Now we concern ourselves with the wirings; we split  $\sigma_{\mathbf{a}}$  and  $\sigma_{\mathbf{R}}$  into  $n + m$  substitutions in parallel, by the outerfaces of the underlying components, as partitioned above,

$$\begin{aligned} \sigma_{\mathbf{a}} &= \left( \prod_i^n \sigma_{i,\mathbf{a}}^{\mathbf{C}} \right) \parallel \left( \prod_i^m \sigma_{i,\mathbf{a}}^{\mathbf{M}} \right) \\ \sigma_{\mathbf{R}} &= \left( \prod_i^n \sigma_{i,\mathbf{R}}^{\mathbf{C}} \right) \parallel \left( \prod_i^m \sigma_{i,\mathbf{R}}^{\mathbf{M}} \right) \end{aligned}$$

Hence, from (A.15),

$$\begin{aligned} \left( \prod_i^n \sigma_{i,\mathbf{a}}^{\mathbf{C}} \right) \parallel \left( \prod_i^m \sigma_{i,\mathbf{a}}^{\mathbf{M}} \right) &= \sigma_{\mathbf{C}} \left( \left( \prod_i^n \sigma_{i,\mathbf{R}}^{\mathbf{C}} \right) \parallel \left( \prod_i^m \sigma_{i,\mathbf{R}}^{\mathbf{M}} \right) \otimes \text{id}_{\mathbf{C}} \otimes \text{id}_{\mathbf{d}} \right) \beta^{-1} \\ &= \left( \left( \prod_i^n \sigma_{i,\mathbf{C}}^{\mathbf{C}} \sigma_{i,\mathbf{R}}^{\mathbf{C}} \right) \parallel \left( \prod_i^m \sigma_{i,\mathbf{C}}^{\mathbf{M}} \sigma_{i,\mathbf{R}}^{\mathbf{M}} \right) \otimes \text{id}_{\mathbf{C}} \otimes \text{id}_{\mathbf{d}} \right) \beta^{-1} \end{aligned} \quad (\text{A.21})$$

where we use Lemma A.7(A.3) (iterated) to split shared wiring in  $\sigma_{\mathbf{C}}$  into  $n+m$  substitutions according to  $\sigma_{i,\mathbf{R}}^{\mathbf{C}}$  and  $\sigma_{i,\mathbf{R}}^{\mathbf{M}}$ .

We split  $\beta^{-1}$  into the inverses of the  $\beta_i^{\mathbf{M}}$ 's and  $\beta_i^{\mathbf{C}}$ 's, and infer from (A.19), (A.20), and (A.21), that,

$$\begin{aligned} (\forall i \in n) (\sigma_{i,\mathbf{a}}^{\mathbf{C}} \otimes \text{id}_1) p_i &= (\sigma_{i,\mathbf{C}}^{\mathbf{C}} (\sigma_{i,\mathbf{R}}^{\mathbf{C}} \otimes \text{id}) \otimes \text{id}_1) ((\ulcorner U_i \urcorner \otimes \text{id}) (\widehat{\alpha}_i \otimes \text{id}) P_{\pi^{-1}(i)} \otimes \text{id}) e_i, \\ \text{and } (\forall i \in m) (\sigma_{i,\mathbf{a}}^{\mathbf{M}} \otimes \text{id}_1) n_i &= (\sigma_{i,\mathbf{C}}^{\mathbf{M}} (\sigma_{i,\mathbf{R}}^{\mathbf{M}} \otimes \text{id}) \otimes \text{id}_1) (M_i S_i \otimes \text{id}) f_i. \end{aligned}$$

Hence, 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 resulting from the first  $n$  sentences,

$$\sigma_{i,\mathbf{a}}^{\mathbf{C}}, \sigma_{i,\mathbf{R}}^{\mathbf{C}}, \sigma_{i,\mathbf{C}}^{\mathbf{C}} \vDash p_i, (\widehat{\alpha}_i \otimes \text{id}) P_{\pi^{-1}(i)} \hookrightarrow \ulcorner U_i \urcorner, e_i,$$

is a consequence by SWITCH of the sentence,

$$\sigma_{i,\mathbf{a}}^{\mathbf{C}}, \text{id}_{\epsilon}, \sigma_{i,\mathbf{C}}^{\mathbf{C}} (\alpha_i \sigma_{\pi^{-1}(i)} \otimes \sigma_{i,\mathbf{R}}^{\mathbf{C}} \otimes \text{id}) \vDash p_i, \text{id} \hookrightarrow P_{\pi^{-1}(i)}^{\mathbf{N}}, e_i$$

for  $P_i = (\widehat{\sigma}_i \otimes \text{id}) (Y_i) P_i^{\mathbf{N}}$ ; where  $P_{\pi^{-1}(i)}^{\mathbf{N}}$  is discrete, free and prime.  $\square$

**Lemma (3.13)** *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  $\sigma'_a, \sigma'_R, \sigma'_C \vDash p, R \hookrightarrow P, d$ , of lesser size, where  $p$  and  $P$  are discrete primes.*

**Proof.** By Fact A.8, and Proposition 3.3 on the normal form for molecules, we have for  $m$

$$m = \left( K_{\vec{y}(\vec{x})} \otimes \text{id} \right) q,$$

while for  $M$  (using immediately that, in particular, the control  $K$  must be equal),

$$M = \left( K_{\vec{u}(\vec{v})} \otimes \text{id} \right) Q,$$

where  $q$  and  $Q$  are namediscrete and prime.

Assuming validity of the original sentence, we have,

$$(\sigma_{\mathbf{a}} \otimes \text{id}) \left( K_{\vec{y}(\vec{x})} \otimes \text{id} \right) q = (\text{id}_1 \otimes \sigma_{\mathbf{C}}(\sigma_{\mathbf{R}} \otimes \text{id}))((M \otimes \text{id})R \otimes \text{id})d.$$

Hence, by Proposition 3.1, there exists a renaming  $\beta$ , s.t.,

$$\sigma_{\mathbf{a}} = \sigma_{\mathbf{C}}(\sigma_{\mathbf{R}} \otimes \text{id})\beta^{-1}. \quad (\text{A.22})$$

$$\text{and } \left( K_{\vec{y}(\vec{x})} \otimes \text{id} \right) q = (\beta \otimes \text{id})((M \otimes \text{id}_{\mathbf{R}})R \otimes \text{id}_{\mathbf{D}})d. \quad (\text{A.23})$$

We split the linkage of  $\beta$  by the underlying components  $K$ ,  $Q$ ,  $R$ , and  $d$ , and get  $\beta = \beta_K \otimes \beta_Q \otimes \beta_R \otimes \beta_d$ . Further, let  $\hat{\sigma}_{(\vec{x})}$  and  $\hat{\phi}_{(\vec{v})}$  be shorthand for local substitutions underlying the ions of  $q$  and  $Q$ , respectively (as detailed in the comments for Proposition 3.3). We overload our notational conventions for local substitutions slightly, and define  $\hat{\sigma}_{(\vec{x})} = \bigotimes_i^{|\vec{x}|} (z_i)/(X_i)$  and  $\hat{\phi}_{(\vec{v})} = \bigotimes_i^{|\vec{v}|} (e_i)/(V_i)$ . Finally, let  $\vec{w} = \beta_K(\vec{u})$ .

Now from (A.23) we get, by standard manipulations,

$$\begin{aligned} \left( K_{\vec{y}(\vec{x})} \otimes \text{id} \right) \left( \hat{\sigma}_{(\vec{x})} \otimes \text{id} \right) q &= \left( \left( \left( K_{\vec{w}(\vec{v})} \otimes \text{id} \right) (\text{id} \otimes \beta_Q) Q \otimes \beta_R \right) R \otimes \beta_d \right) d \\ &= \left( K_{\vec{w}(\vec{v})} \otimes \text{id} \right) \left( \left( \hat{\phi}_{(\vec{v})} \otimes \beta_Q \right) Q \otimes \beta_R \right) R \otimes \beta_d \end{aligned} \quad (\text{A.24})$$

From (A.24), by Proposition 3.3,  $\beta_K = \vec{u}/\vec{y}$ , and there exists a (unique)  $\hat{\alpha}$ , s.t.  $\hat{\alpha}\hat{\phi}_{(\vec{v})} = \hat{\sigma}_{(\vec{v})}$  (mapping the set  $V_i$  to  $z_i$  iff  $\hat{\sigma}_{(\vec{x})}$  maps  $X_i$  to  $z_i$ ).

Hence, we have

$$\begin{aligned} \left( \hat{\sigma}_{(\vec{x})} \otimes \text{id} \right) q &= \left( \left( \left( \hat{\sigma}_{(\vec{v})} \otimes \beta_Q \right) Q \otimes \beta_R \right) R \otimes \beta_d \right) d \\ &= (\text{id} \otimes \beta_Q \otimes \beta_R \otimes \beta_d) \left[ \left( \left( \hat{\sigma}_{(\vec{v})} \otimes \text{id} \right) (Q \otimes \text{id}) R \otimes \text{id} \right) d \right] \\ &= (\text{id} \otimes \beta_{QRd}) P, \end{aligned} \quad (\text{A.25})$$

in the last line introducing metavariables  $\beta_{QRd}$ , and  $P$ .

By (A.22), we have

$$\begin{aligned} \sigma_{\mathbf{a}} &= \sigma_{\mathbf{a}}^K \parallel \sigma_{\mathbf{a}}^{QRd} \\ \text{and } \sigma_{\mathbf{C}}(\sigma_{\mathbf{R}} \otimes \text{id}) \left( \beta_K^{-1} \otimes \beta_{QRd}^{-1} \right) &= \sigma_{\mathbf{C}}^K \beta_K^{-1} \parallel \sigma_{\mathbf{C}}^{QRd} (\sigma_{\mathbf{R}} \otimes \text{id}) \beta_{QRd}^{-1}, \end{aligned}$$

splitting  $\sigma_{\mathbf{a}}$  and  $\sigma_{\mathbf{C}}$  by their innerfaces according to the underlying components. By Lemma A.6, then also — as the interfaces of the corresponding substitutions match,

$$\begin{aligned} \sigma_{\mathbf{a}}^K &= \sigma_{\mathbf{C}}^K \beta_K^{-1} \\ \text{and } \sigma_{\mathbf{a}}^{QRd} &= \sigma_{\mathbf{C}}^{QRd} (\sigma_{\mathbf{R}} \otimes \text{id}) \beta_{QRd}^{-1}. \end{aligned}$$

Combining this with (A.25), we find,

$$(\text{id} \otimes \sigma_{\mathbf{a}}^{QRd}) \left( \widehat{\sigma}_{(\vec{X})} \otimes \text{id} \right) q = \sigma_{\mathbf{C}}^{QRd} (\sigma_{\mathbf{R}} \otimes \text{id}) P.$$

Choosing  $\sigma'_a = \sigma_{\mathbf{a}}^{QRd}$ ,  $\sigma'_R = \sigma_{\mathbf{R}}$ ,  $\sigma'_C = \sigma_{\mathbf{C}}^{QRd}$ ,  $p = \left( \widehat{\sigma}_{(\vec{X})} \otimes \text{id} \right) q$ , and  $P = P$ , we have a valid sentence, which by ION yields the original sentence.  $\square$

**Lemma (3.14)** *Every valid sentence  $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash p, \text{id} \leftrightarrow P, e$ , 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  $\sigma'_a, \sigma'_r, \sigma'_M \vDash m, R \leftrightarrow M, d$ .*

**Proof.** (Omitted) Analogous to the proof of Lemma 3.12, but simpler as redex is id. Further, for the concretions in  $P$ , instead of arriving at sentences which are derivable by SWITCH, each such sentence is an instance of PRIME-AXIOM.  $\square$