

Matching of Bigraphs

Lars Birkedal¹ Troels Christoffer Damgaard¹
Arne John Glenstrup¹

IT University of Copenhagen, Denmark

Robin Milner²

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 [9,13,15]. 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 rules from which a labelled transition system may be derived in such a way

¹ Email: {birkedal,tcd,panic}@itu.dk

² Email: Robin.Milner@cl.cam.ac.uk

that an associated bisimulation relation is a congruence relation. The unification has recovered existing behavioural theories for the π -calculus [9], the ambient calculus [8], and has contributed to that for Petri nets [12]. 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 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 connections between folder and z (in d) and z and folder (in C), removing the

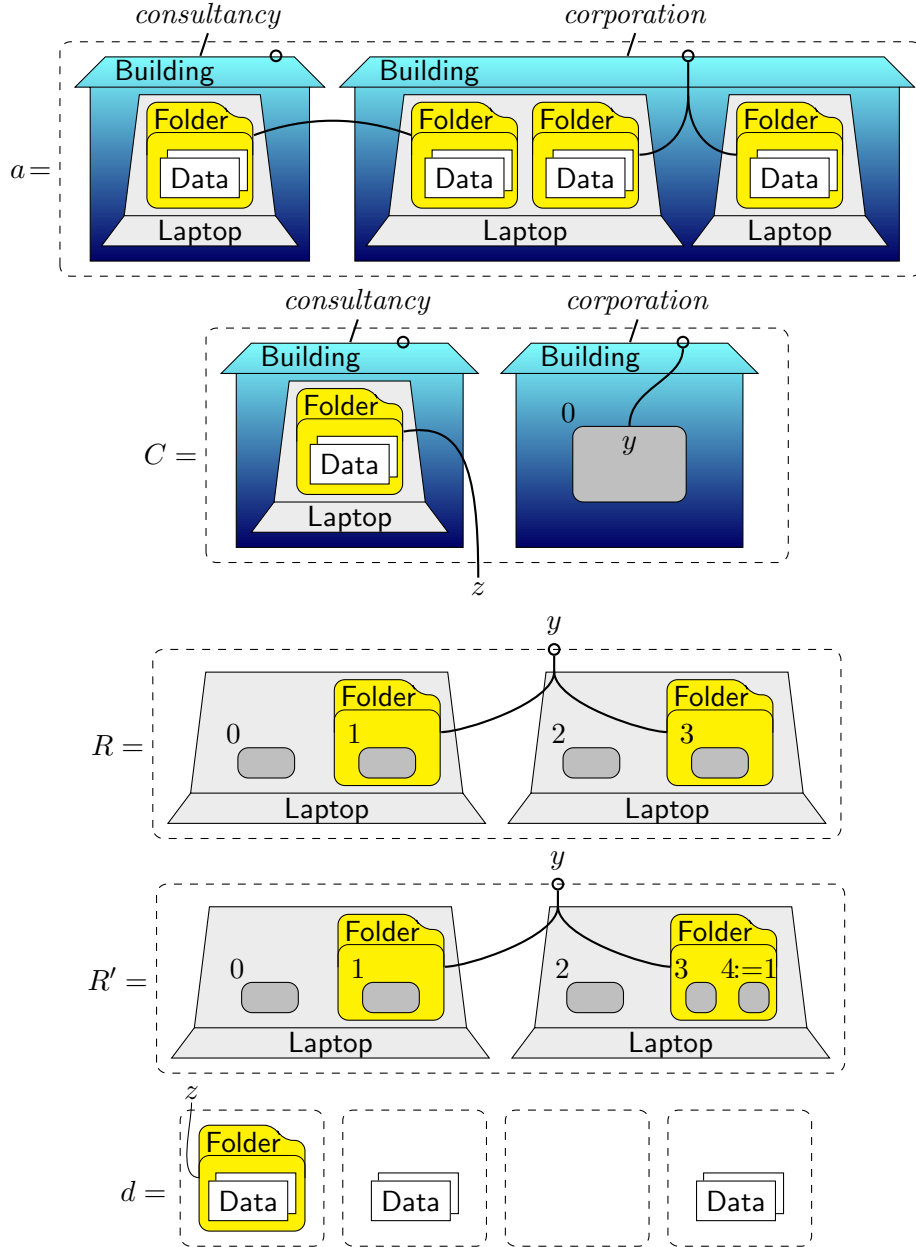


Fig. 1. Example of a ground agent $a = C \circ (\text{id}_z \otimes R) \circ d$. Reaction rule $R \rightarrow R'$ copies data between connected folders.

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 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 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 [10,5].

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 ex-

tended 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. A *link* is a hyper edge of the link graph, either an

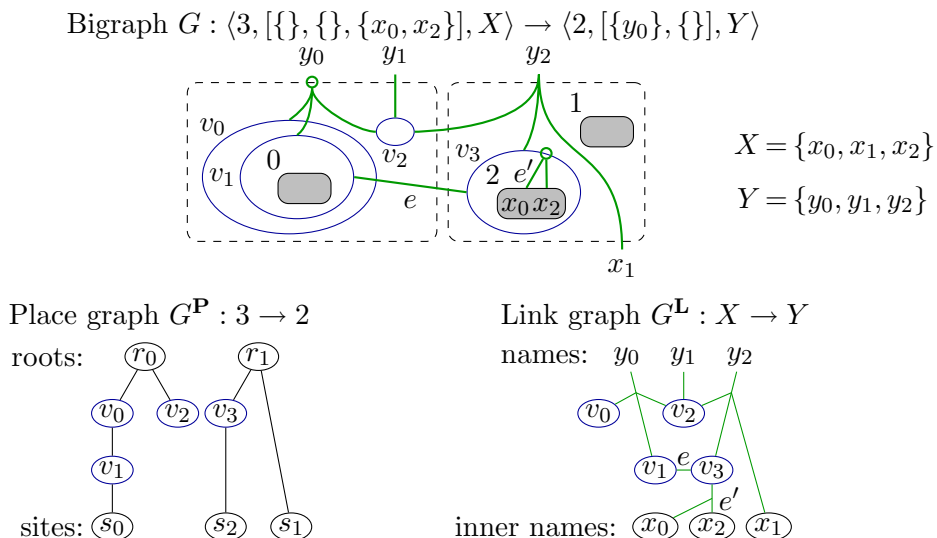


Fig. 2. Example bigraph illustrated by nesting and as place and link graph.

internal edge e or a *name* y . 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 [10], our prime interest is in *abstract bigraphs*, equivalence

classes of concrete bigraphs that differ only in the names of nodes and internal edges. Abstract bigraphs are illustrated with their node controls, as shown in Figure 1. 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*.

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 roots $r_{i'}$ and $r_{j'}$, and all sites s_i and s_j where s_i is a descendant of $r_{i'}$ and s_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 certain permutations can be avoided when composing them from basic bigraphs.

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 $G_0 \otimes \cdots \otimes G_{n-1}$.

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 in” the roots of G_2 into the sites of G_1 , 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* ϱ , mapping each site of R' to a site of R . Interfaces $I = \langle m, \vec{X}, X \rangle$ and $I' = \langle m', \vec{X}', X' \rangle$ must be local, and are related by $X'_i = X_{\varrho(i)}$. We illustrate ϱ by a ‘ $i := j$ ’, as shown in Figure 1, whenever $\varrho(i) = j \neq i$. Given an instantiation ϱ and a discrete bigraph $d = d_0 \otimes \cdots \otimes d_k$ with prime d_i ’s, we let $\varrho(d) = d_{\varrho(0)} \otimes \cdots \otimes d_{\varrho(k)}$, i.e., by 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 $Y_0 \uplus \cdots \uplus Y_{n-1}$ when $\vec{Y} = Y_0, \dots, Y_{n-1}$, and similarly $\{\vec{y}\}$ for $\{y_0, \dots, y_{n-1}\}$. For interfaces, we write 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 a set of basic bigraphs, shown in Table 1 [5]. Given a

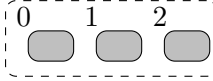
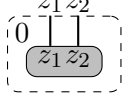
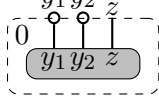
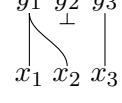
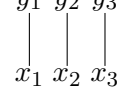
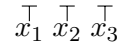
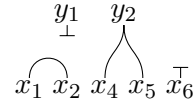
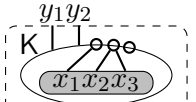
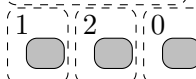
	<i>Notation</i>	<i>Example</i>
Merge	$merge_n : n \rightarrow 1$	$merge_3 =$ 
Concretion	$\lceil X \rceil : (X) \rightarrow \langle X \rangle$	$\lceil \{z_1, z_2\} \rceil =$ 
Abstraction	$(Y)P : I \rightarrow \langle 1, [Y], Z \uplus Y \rangle$	$(\{y_2\})(\{y_1\})\lceil \{y_1, y_2, z\} \rceil =$ 
Substitution	$\vec{y}/\vec{X} : X \rightarrow Y$	$[y_1, y_2, y_3]/[\{x_1, x_2\}, \{\}, \{x_3\}] =$ 
Renaming	$\vec{y}/\vec{x} : X \rightarrow Y$	$[y_1, y_2, y_3]/[x_1, x_2, x_3] =$ 
Closure	$/X : X \rightarrow \{\}$	$/\{x_1, x_2, x_3\} =$ 
Wiring	$(/Z \otimes \alpha)\sigma : X \rightarrow Y$	$(/\{z_2, z_4\} \otimes [y_1, y_2]/[z_1, z_3]) / [\{z_1, z_2, z_3, z_4\} / [\{\}, \{x_1, x_2\}, \{x_4, x_5\}, \{x_6\}]] =$ 
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	$\{i \mapsto j, \dots\} : m \rightarrow m$	$\{0 \mapsto 2, 1 \mapsto 0, 2 \mapsto 1\} =$ 

Table 1

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

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 $(\cdot)^\lceil$ that reaches as far right as possible.

For a renaming $\alpha : X \rightarrow Y$, we write $\lceil \alpha \rceil$ to mean $(\text{id}_1 \otimes \alpha)^\lceil X \rceil$, and when $\sigma : U \rightarrow Y$, we let $\widehat{\sigma} = (Y)(\sigma \otimes \text{id}_1)^\lceil U \rceil$.

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

$$\begin{aligned}
 G &= (\omega \otimes ((\{y_0\})y_0/Y^\lceil Y \rceil) \otimes \lceil \{\} \rceil) (((Y)P_1) \otimes P_2 \otimes y_2/x_1), \text{ where} \\
 \omega &= (/e \otimes \text{id}_{\{y_1, y_2\}})[y_1, y_2, e]/[\{y_1\}, \{y_2, y'_2, y''_2\}, \{e, e'\}], \quad Y = \{y_0, y'_0, y''_0\} \\
 P_1 &= (\text{id}_{\{y_0, y_1, y'_2, e\}} \otimes merge_2) ((\text{id}_{\{y_0, e\}} \otimes K_{0[y'_0]})K_{1[y_0, e]} \otimes K_{2[y'_0, y_1, y'_2]} merge_0) \\
 P_2 &= (\text{id}_{\{e', y'_2\}} \otimes merge_2)(K_{3[e', y'_2]}(\{\{x_0, x_2\}\}) \otimes \lceil \{\} \rceil),
 \end{aligned}$$

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

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

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 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 ω, α, σ and π 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 α .

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 $K_{\vec{y}(\vec{u})} \bigotimes_i^n (u_i)/(X_i)$. 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 $M : I \rightarrow (\{\bar{y}\} \uplus Z)$ can be decomposed as*

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

where P is a discrete prime. Any other decomposition of M on this form, has the form $(K_{\bar{y}(\bar{x})} \otimes \text{id}_Z)P'$, where there exists a unique $\hat{\alpha}$, given by $u_i \mapsto x_i$, such that $K_{\bar{y}(\bar{x})}\hat{\alpha} = K_{\bar{y}(\bar{x})}$ 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 $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \leftrightarrow C, d$, 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 $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \leftrightarrow C, d$, where $\omega_{\mathbf{R}} : U \rightarrow Y$, C has global outer names V , and d has global outer names Z , is *valid*, denoted $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vDash a, R \leftrightarrow C, d$, iff

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

Note that for a valid sentence $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \leftrightarrow C, d$, 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 $R' = (\omega_{\mathbf{R}} \otimes \text{id})R$, 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

In Figure 3 we present with a set of rules for inferring matching sentences. 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 π 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 ω if the two parts of the redex share a (necessarily global) name, cf. Figure 4.

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

$$\begin{array}{c}
 \text{PERM} \frac{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, \bigotimes_i^m P_{\pi(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}{\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 (\bigotimes_i^n (v_i)/(X_i) \otimes \text{id})p, R \hookrightarrow (\bigotimes_i^n (v_i)/(Z_i) \otimes \text{id})P, d \quad \alpha = \vec{y}/\vec{u} \quad \sigma_{\mathbf{y}} : \{\vec{y}\} \rightarrow \{\vec{u}\}}{\sigma_{\mathbf{y}} \parallel \omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \sigma_{\mathbf{y}}\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_{\mathbf{R}} \otimes \omega_{\mathbf{R}} \otimes \text{id}) \vdash p, \text{id} \hookrightarrow P, d \quad \sigma_{\mathbf{R}} : W \rightarrow U}{\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash p, (\widehat{\sigma}_{\mathbf{R}} \otimes \text{id})(W)P \hookrightarrow \ulcorner U \urcorner, d} \\
 \\
 \text{PRIME-AXIOM} \frac{}{\omega, \text{id}_{\epsilon}, \omega(\alpha^{-1} \otimes \text{id}) \vdash p, \text{id} \hookrightarrow \ulcorner \alpha \urcorner, p} \\
 \\
 \text{WIRING-AXIOM} \frac{}{y, X_{\mathbf{R}}/\emptyset, y/(X_{\mathbf{R}} \uplus X_{\mathbf{d}}) \vdash \text{id}_{\epsilon}, \text{id}_{\epsilon} \hookrightarrow \text{id}_{\epsilon}, X_{\mathbf{d}}/\emptyset} \\
 \\
 \text{CLOSE} \frac{\{\vec{y}_i\} = Y \quad \{\vec{Z}_i\} = Z \quad \sigma_{\mathbf{R}}^{\mathbf{Z}} : U \rightarrow Z \quad (\bigotimes_i^m y_i/X_i) \otimes (\bigotimes_i^l z_i/X_{i+m}) \otimes \sigma_{\mathbf{a}}, \quad (\bigotimes_i^m y_i/Y_i) \otimes \text{id}_U \otimes \sigma_{\mathbf{R}}^{\mathbf{C}}, (\bigotimes_i^l z_i/Z_i)\sigma_{\mathbf{R}}^{\mathbf{Z}} \otimes \sigma_{\mathbf{C}} \otimes \text{id}_Y \vdash a, R \hookrightarrow C, d}{(\bigotimes_i^m /y_i y_i/X_i) \otimes (\bigotimes_i^l /z_i z_i/X_{i+m}) \otimes \sigma_{\mathbf{a}}, \quad (\bigotimes_i^m /y_i y_i/Y_i) \otimes \sigma_{\mathbf{R}}^{\mathbf{Z}} \otimes \sigma_{\mathbf{R}}^{\mathbf{C}}, (\bigotimes_i^l /z_i z_i/Z_i) \otimes \sigma_{\mathbf{C}} \vdash a, R \hookrightarrow C, d}
 \end{array}$$

Fig. 3. Rules for matching binding bigraphs

and context extended with the underlying global substitutions $\sigma_{\mathbf{a}}$ and $\sigma_{\mathbf{C}}$. In other words, this rule expresses that we can match a 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.

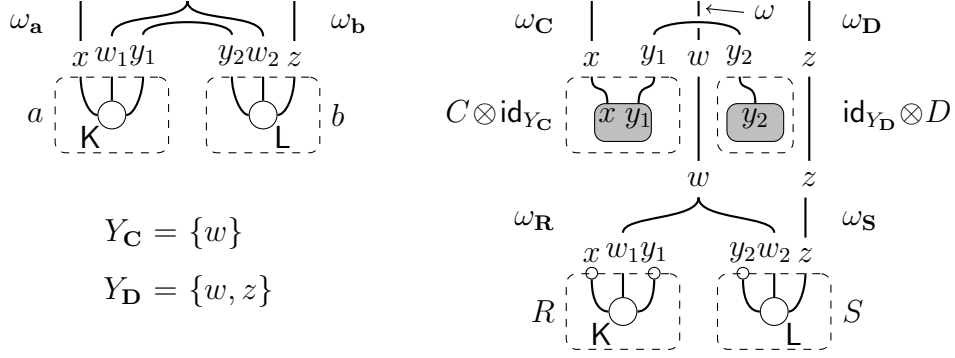
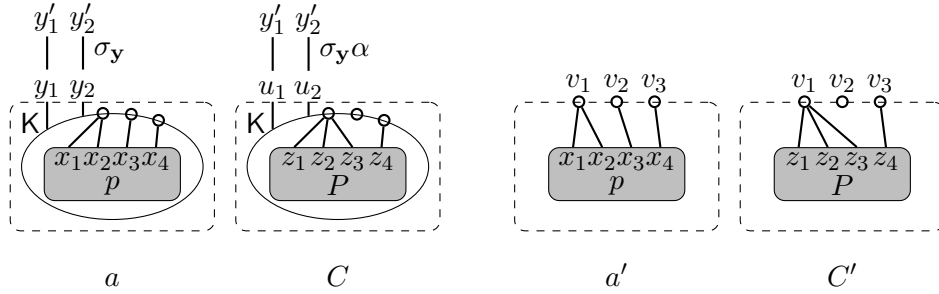


Fig. 4. Matching a product using the PAR rule

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 $K_{\vec{n}(\vec{z})}$, if we extend the wirings of agents and contexts with isomorphic wiring on the outer names \vec{y} and \vec{n} ; stated in the rule by requiring that we extend with σ_y and $\sigma_y\alpha$ (where $\alpha = \vec{y}/\vec{n}$). 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.


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

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 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 *open* bigraphs and “close” this match by replacing names in wirings with edges, taking care to split multi-closures appropriately. For example, the agent a in Figure 6 is matched by

matching agent a' and then closing the names y, z_1 and z_2 . So internal agent

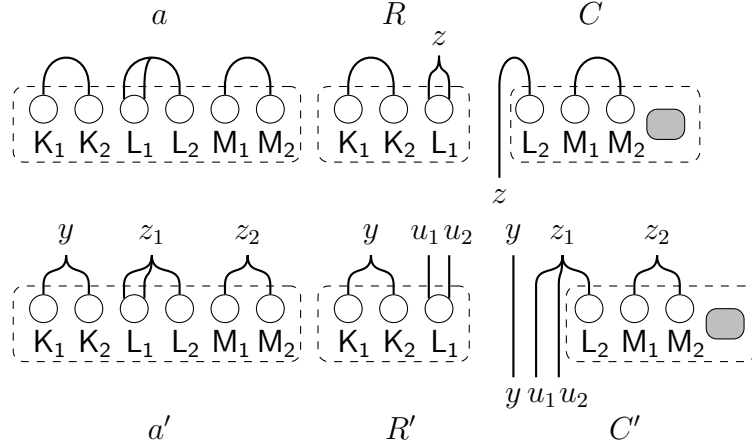


Fig. 6. Matching closed links within and between redex and context

edges matched by internal redex edges are named y_i , and edges matched by internal context edges are named z_i .

Theorem 3.6 *The rules for matching in Figure 3 are sound, i.e., 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 $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \hookrightarrow C, d$ is the number of ions in a .

The following lemmas express how a valid sentence may be derived by applications of inference rules to valid sentences of lesser or equal size. The proofs proceed by first decomposing the components of the given valid sentence, then defining the components of the valid sentence(s) claimed to exist and, finally, verifying that (1) the sentences claimed to exist really are valid and (2) that the given sentence can indeed be derived as claimed. The decompositions are obtained via Propositions 3.1, 3.2, and 3.3, and the verifications proceed using lemmas found in [5] (in particular, the “push-through-lemma,” which expresses how we can push a permutation “through” a product of primes, permuting the order in which they appear in the product, and producing a permutation that reorders the sites in the primes to preserve the inner face).

Lemma 3.8 *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 $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \vDash a, R \hookrightarrow Q \otimes \bigotimes_i^n P_i, d$, 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 $\sigma_{\mathbf{a}}^P, \sigma_{\mathbf{R}}^P, \sigma_{\mathbf{C}}^P \parallel \sigma_{\mathbf{C}}^S \models p, S \hookrightarrow P, e$ and $\sigma_{\mathbf{a}}^C, \sigma_{\mathbf{R}}^C, \sigma_{\mathbf{C}}^C \parallel \sigma_{\mathbf{C}}^S \models a', R' \hookrightarrow \bigotimes_i^n P_i, e'$.

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

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

Lemma 3.12 *Every valid sentence $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \models p, R \hookrightarrow P, d$, with p and P 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:*

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

Lemma 3.13 *Every valid sentence $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \models m, R \hookrightarrow M, d$, with m and M free discrete molecules, is provable using the ION rule on a valid sentence $\sigma'_{\mathbf{a}}, \sigma'_{\mathbf{R}}, \sigma'_{\mathbf{C}} \models p, R \hookrightarrow P, d$, of lesser size, where p and P are discrete primes.*

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

Theorem 3.15 *The rules for matching in Figure 3 are complete, i.e., 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 $\sigma_{\mathbf{a}}, \sigma_{\mathbf{R}}, \sigma_{\mathbf{C}} \models m, R \hookrightarrow M, d$, 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; 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, exactly as matching is the workhorse of evaluation in functional languages such as Standard ML.

In practice, one is, of course, interested in minimizing unnecessary blind search, and thus, e.g., 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 of 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 [14] (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 $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \mapsto C, d$, 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 need 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 [18,7,11,21]. Due to the special structure of bigraphs, general GPM algorithms are expected to be inefficient, although some GPM tools [20] 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 [16,19,17] 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 implemen-

tations 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: *FOSSACS '06: Proceedings of 9th International Conference on Foundations of Software Science and Computation Structures*, LNCS **3921** (2006).
- [4] Birkedal, L., S. Debois and T. Hildebrandt, *Sortings for reactive systems*, Technical Report 84, IT University of Copenhagen (2006), iISBN 87-7949-124-3.
- [5] Damgaard, T. C. and L. Birkedal, *Axiomatizing binding bigraphs (revised)*, Technical Report TR-2005-71, IT University of Copenhagen (2005).
- [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 (2005), forthcoming.
- [9] Jensen, O. H. and R. Milner, *Bigraphs and mobile processes (revised)*, Technical Report 580, University of Cambridge (2004).
- [10] Jensen, O. H. and R. Milner, *Bigraphs and mobile processes (revised)*, Technical Report UCAM-CL-TR-580, University of Cambridge – Computer Laboratory (2004), iISSN 1476-2986.
- [11] Larrosa, J. and G. Valiente, *Constraint satisfaction algorithms for graph pattern matching*, Mathematical Structures in Computer Science **12** (2002), pp. 403–422.

- [12] Leifer, J. J. and R. Milner, *Transition systems, link graphs and Petri nets*, Technical Report 598, University of Cambridge (2004).
- [13] Milner, R., *Axioms for bigraphical structure*, Technical Report 581, University of Cambridge (2004).
- [14] Milner, R., *Bigraphs whose names have multiple locality*, Technical Report UCAM-CL-TR-603, University of Cambridge, Computer Laboratory (2004).
URL <http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-603.pdf>
- [15] Milner, R., *Pure bigraphs*, Technical Report 614, University of Cambridge (2005).
- [16] Selkow, S. M., *The tree-to-tree editing problem*, Information Processing Letters **6** (1977), pp. 184–186.
- [17] Shamir, R. and D. Tsur, *Faster subtree isomorphism*, Journal of Algorithms **33** (1999), pp. 267–280.
- [18] Ullman, J. D., *An algorithm for subgraph isomorphism*, Journal of the ACM **23** (1976), pp. 31–42.
- [19] Valiente, G., “Algorithms on Trees and Graphs,” Springer, Berlin, 2002.
- [20] 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.
URL http://www.inf.mit.bme.hu/FTSRG/Publications/varro/2005/gramot05_vvf.pdf
- [21] 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.