An Implementation of Bigraph Matching

Arne John Glenstrup
Troels Christoffer Damgaard
Lars Birkedal
Espen Højsgaard

IT University Technical Report Series **TR-2010-135**

ISSN 1600–6100 **December 2010**

ISSN 1600–6100

ISBN 978–87–7949–228–8

Copies may be obtained by contacting:

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

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

An Implementation of Bigraph Matching

Arne John Glenstrup Troels Christoffer Damgaard Lars Birkedal
Espen Højsgaard

IT University of Copenhagen, Denmark

Abstract

We describe a provably sound and complete matching algorithm for bigraphical reactive systems. The algorithm has been implemented in our BPL Tool, a first implementation of bigraphical reactive systems. We describe the tool and present a concrete example of how it can be used to simulate a model of a mobile phone system in a bigraphical representation of the polyadic π calculus.

1 Introduction

The theory of bigraphical reactive systems [13] provides a general meta-model for describing and analyzing mobile and distributed ubiquituous systems. Bigraphical reactive systems form a graphical model of computation in which graphs embodying both locality and connectivity can be reconfigured using *reaction rules*. So far it has been shown how to use the theory for recovering behavioural theories for various process calculi [12, 13, 15] and how to use the theory for modelling context-aware systems [2].

In this paper we describe the core part of our BPL Tool, a first prototype implementation of bigraphical reactive systems, which can be used for experimenting with bigraphical models.

The main challenge of implementing the dynamics of bigraphical reactive systems is the *matching problem*, that is, to determine for a given bigraph and reaction rule whether and how the reaction rule can be applied to rewrite the bigraph. When studying the matching problem in detail, one finds that it is a surprisingly tricky problem (it is related to the NP-complete graph embedding problem). Therefore we decided early on to study the matching problem quite formally and base our prototype implementation on a provably correct specification. In previous work [1, 9], we gave a sound and complete inductive characterization of the matching problem for bigraphs. Our inductive characterization was based on normal form theorems for binding bigraphs [8].

In the present paper we extend the inductive characterization from graphs to a *term* representation of bigraphs. A single bigraph can be represented by several structurally congruent bigraph terms. Using an equational theory for bigraph terms [8], we essentially get a non-deterministic matching algorithm operating on bigraph terms. However, such an algorithm will be wildly non-deterministic and we thus provide an alternative, but still provably sound and complete, characterization of matching on terms, which is more suited for mechanically finding matching. In particular, it spells out how and where to make use of structural congruences.

We have implemented the resulting algorithm in our BPL Tool, which we briefly describe in Section 8. We also present an example of a bigraphical reactive system, an encoding of the polyadic π calculus, and show how it can be used to simulate a simple model of a mobile phone system.

Bigraphical reactive systems are related to general graph transformation systems; Ehrig et al. [10] provide a recent comprehensive overview of graph transformation systems. In particular, bigraph matching is related to the general graph pattern matching (GPM) problem, so general GPM algorithms might also be applicable to bigraphs [11, 14, 20, 21]. As an alternative to implementing matching for bigraphs, one

could try to formalize bigraphical reactive systems as graph transformation systems and then use an existing implementation of graph transformation systems. Some promising steps in this direction have been taken [19], but they have so far fallen short of capturing precisely all the aspects of binding bigraphs. For a more detailed account of related work, in particular on relations between BRSs, graph transformations, term rewriting and term graph rewriting, see the Thesis of Damgaard [7, Section 6].

The remainder of this paper is organized as follows. In Section 2 we give an informal presentation of bigraphical reactive systems and normalisation techniques needed for the implementation. In Section 3 we recall the graph-based inductive characterization, then in Section 4 we develop a term-based inductive characterization, forming the basis for our implementation of matching. Section 5 explains how we can restrict the kind of inferece trees the algorithm needs to consider, without sacrificing completeness; this is then used in Section 6, where we describe how to translate the inference system into a working algorithm. We discuss how to handle nondeterminism in Section 7, and in Section 8 we describe the BPL Tool and present an example use of it. Finally, we conclude and discuss future work in Section 9.

2 Bigraphs and Reactive Systems

In the following, we present bigraphs informally; for a formal definition, see the work by Jensen and Milner [13] and Damgaard and Birkedal [8].

2.1 Concrete Bigraphs

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

We usually illustrate the place graph by nesting nodes, as shown in the upper part of Figure 1 (ignore for now the interfaces denoted by "$:\cdot\to\cdot$"). A *link* is a hyper edge of the link graph, either an internal

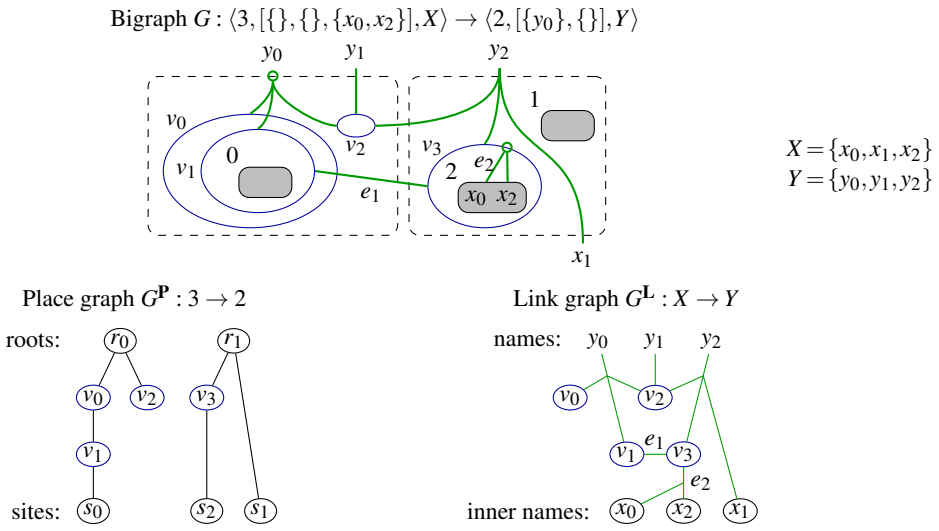


Figure 1: Example bigraph illustrated by nesting and as place and link graph.

edge e or a *name* y. Links that are names are called *open*, those that are edges are called *closed*. Names

and inner names can be *global* or *local*, the latter being located at a specific root or site, respectively. In Figure 1, y_0 is located at r_0, indicated by a small ring, and x_0 and x_2 are located at s_2, indicated by writing them within the site. Global names like y_1 and y_2 are drawn anywhere at the top, while global inner names like x_1 are drawn anywhere at the bottom. A link, including internal edges like e_2 in the figure, can be located with one *binder* (the ring), in which case it is a *bound link,* otherwise it is *free*. However, a bound link must satisfy the *scope rule*, a simple structural requirement that all points (cf. next paragraph) of the link lie within its location (in the place graph), except for the binder itself. This prevents y_2 and e_1 in the example from being bound.

2.2 Controls and Signatures

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

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

A collection of controls with their associated kinds and arities is referred to as a *signature*.

2.3 Abstract Bigraphs

While concrete bigraphs with named nodes and internal edges are the basis of bigraph theory [13], our prime interest is in *abstract bigraphs,* equivalence classes of concrete bigraphs that differ only in the names of nodes and internal edges[1]. Abstract bigraphs are illustrated with their node controls (see Figure 14 in Section 8). In what follows, "bigraph" will thus mean "abstract bigraph."

2.4 Interfaces

Every bigraph G has two *interfaces* I and J, written $G : I \to J$, where I is the *inner face* and J the *outer face*. An interface is a triple $\langle m, \vec{X}, X \rangle$, where m is the *width* (the number of sites or roots), X the entire set of local and global names, and $\vec{X}$ indicates the locations of each local name, cf. Figure 1. We let $\varepsilon = \langle 0, [], \{\} \rangle$; when $m = 1$ the interface is *prime*, and if all $x \in X$ are located by $\vec{X}$, the interface is *local*. As in the work by Milner [18] we write $G : \to J$ or $G : I \to$ for $G : I \to J$ when we are not concerned about about I or J, respectively.

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

2.5 Discrete and Regular Bigraphs

We say that a bigraph is *discrete* iff every free link is a name and has exactly one point. The virtue of discrete bigraphs is that any connectivity by internal edges must be bound, and node ports can be accessed individually by the names of the outer face. Further, a bigraph is *name-discrete* iff it is discrete and every

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

bound link is either an edge, or (if it is a name) has exactly one point. Note that name-discrete implies discrete.

A bigraph is *regular* if, for all nodes v and sites i, j, k with $i \leq j \leq k$, if i and k are descendants of v, then j is also a descendant of v. Further, for roots $r_{i'}$ and $r_{j'}$, and all sites i and j where i is a descendant of $r_{i'}$ and j of $r_{j'}$, if $i \leq j$ then $i' \leq j'$. 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.

2.6 Product and Composition

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

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

The *prime product* $G_1 \mid G_2$ is like the parallel product, except the result has just one root (also when G_1 and G_2 are wirings), produced by merging any roots of G_1 and G_2 into one.

We can *compose* bigraphs $G_2 : I \to I'$ and $G_1 : I' \to J$, yielding bigraph $G_1 \circ G_2 : I \to J$, by plugging the sites of G_1 with the roots of G_2, eliminating both, and connecting names of G_2 with inner names of G_1. In the following, we will omit the '$\circ$', and simply write $G_1 G_2$ for composition, letting it bind tighter than tensor product.

2.7 Notation, Basic Bigraphs, and Abstraction

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

Any bigraph can be constructed by applying composition, tensor product and abstraction to identities (on all interfaces) and a set of basic bigraphs, shown in Table 1 [8]. For permutations, when used in any context, $\pi_{\vec{X}} G$ or $G\pi_{\vec{X}}$, $\vec{X}$ is given entirely by the interface of G; in these cases we simply write $\pi_{\vec{X}}$ as π.

Given a prime P, the abstraction operation localises a subset of its outer names. Note that the scope rule is necessarily respected since the inner face of a prime P is required to be local, so all points of P are located within its root. The abstraction operator is denoted by $(\cdot)\cdot$ and reaches as far right as possible.

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

Note that $[]/[] = /\emptyset = \pi_0 = \mathsf{id}_\varepsilon$ and $merge_1 = \ulcorner \emptyset \urcorner = \pi_1 = \mathsf{id}_1$, where π_i is the nameless permutation of width i.

2.8 Bigraphical Reactive Systems

Bigraphs in themselves model two essential parts of context: locality and connectivity. To model also *dynamics,* we introduce *bigraphical reactive systems* (BRS) as a collection of *rules.* Each rule $R \xrightarrow{\rho} R'$ consists of a regular *redex* $R : I \to J$, a *reactum* $R' : I' \to J$, and an *instantiation* ρ, mapping each site of R' to a site of R, and mapping local names in I' to those of I, as illustrated in Figure 2. Interfaces $I = \langle m, \vec{X}, X \rangle$ and $I' = \langle m', \vec{X}', X' \rangle$ must be local, and are related by $X'_i = X_{\rho(i)}$, where ρ must be a bijection between X'_i and $X_{\rho(i)}$. We illustrate ρ by '$i := j$', whenever $\rho(i) = j \neq i$, or, alternatively, by listing $[\rho(0), \ldots, \rho(m'-1)]$. Given an instantiation ρ and a discrete bigraph $d = d_0 \otimes \cdots \otimes d_k$ with prime d_i's, we let $\rho(d) = d_{\rho(0)} \otimes \cdots \otimes d_{\rho(k)}$, allowing copying, discarding and reordering parts of d.

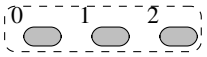
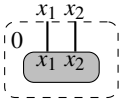
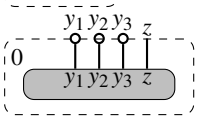
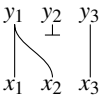
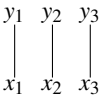
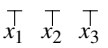
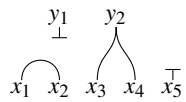
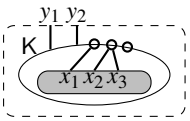
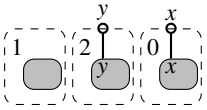
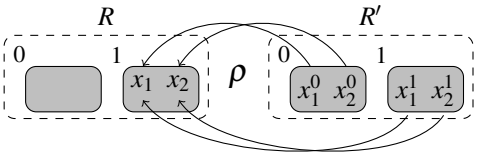
	Notation	*Example*	
Merge	$merge_n : n \to 1$	$merge_3 =$	
Concretion	$\ulcorner X \urcorner : (X) \to \langle X \rangle$	$\ulcorner \{x_1,x_2\} \urcorner =$	
Abstraction	$(Y)P : I \to \langle 1,[Y], Z \uplus Y \rangle$	$(\{y_1,y_2\})(\{y_3\})\ulcorner \{y_1,y_2,y_3,z\} \urcorner =$	
Substitution σ	$\vec{y}/\vec{X} : X \to Y$	$[y_1,y_2,y_3]/[\{x_1,x_2\},\{\},\{x_3\}] =$	
Renaming α, β	$\vec{y}/\vec{x} : X \to Y$	$[y_1,y_2,y_3]/[x_1,x_2,x_3] =$	
Closure	$/X : X \to \{\}$	$/\{x_1,x_2,x_3\} =$	
Wiring ω	$(\mathsf{id} \otimes /Z)\sigma : X \to Y$	$(\mathsf{id}_{\{y1,y2\}} \otimes /\{z_1,z_2\})$ $[y_1,z_1,y_2,z_2]/$ $[\{\},\{x_1,x_2\},\{x_3,x_4\},\{x_5\}] =$	
Ion	$K_{\vec{y}(\vec{X})} : (\{\vec{X}\}) \to \langle \{\vec{y}\} \rangle$	$\mathsf{K}_{[y_1,y_2]([\{x_1\},\{x_2,x_3\},\{\}])} =$	
Permutation $\pi_{\vec{X}}$	$\{i \mapsto j, \ldots\} : \langle m, \vec{X}, X \rangle \to \langle m, \pi(\vec{X}), X \rangle$	$\{0 \mapsto 2, 1 \mapsto 0, 2 \mapsto 1\}_{[\{x\},\emptyset,\{y\}]} =$	

Table 1: Basic bigraphs, the abstraction operation, and variables ranging over bigraphs.



$$\rho = [1 \& [x_1^0 \mapsto x_1, x_2^0 \mapsto x_2], \\ 1 \& [x_1^1 \mapsto x_1, x_2^1 \mapsto x_2]]$$

Figure 2: A reaction rule

Given an agent a, a *match* of redex R is a decomposition $a = C(\mathsf{id}_Z \otimes R)d$, with active context C and discrete parameter d with its global names Z. Dynamics is achieved by transforming a into a new agent $a' = C(\mathsf{id}_Z \otimes R')d'$, where $d' = \rho(d)$, cf. Figure 3. This definition of a match is as given by Jensen

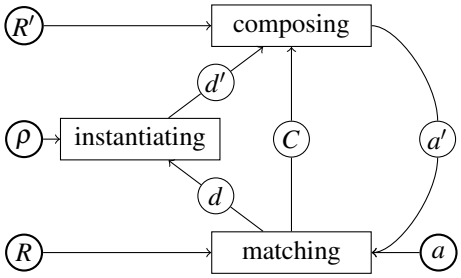


Figure 3: The reaction cycle

and Milner [13], except that we here also require R to be regular. This restriction to regular redexes R simplifies the inductive characterization of matching without limiting the set of possible reactions, as sites in R and R' can be renumbered to render R regular.

2.9 Bigraph Terms and Normal Forms

Expressing bigraphs as terms composed by product, composition and abstraction over basic bigraph terms, Damgaard and Birkedal [8] showed that bigraphs can be expressed on normal forms uniquely up to certain permutations and renamings. Further, they showed equivalence of term and bigraph equality, which will allow us in Section 4 to base our implementation on terms rather than graphs.

In this work, we use the normal forms shown in Figure 4, enabling us to express regular bigraphs simply by removing the permutations. These normal forms are unique up to permutation of S_i's and renaming of names not visible on the interfaces.

$$
\begin{array}{rll}
M & ::= (\mathsf{id}_Z \otimes K_{\vec{y}(\vec{X})})N & \textit{molecule} \\
S & ::= \ulcorner \alpha \urcorner \quad | \quad M & \textit{singular top-level node} \\
G & ::= (\mathsf{id}_Y \otimes merge_n)(\bigotimes_i^n S_i)\pi & \textit{global discrete prime} \\
N & ::= (X)G & \textit{name-discrete prime} \\
P,Q & ::= (\mathsf{id}_Z \otimes \widehat{\sigma})N & \textit{discrete prime} \\
D & ::= \alpha \otimes (\bigotimes_i^n P_i)\pi & \textit{discrete bigraph} \\
B & ::= (\omega \otimes \mathsf{id}_{(\vec{X})})D & \textit{binding bigraph}
\end{array}
$$

Figure 4: Normal forms for binding bigraphs

2.10 Normalising

For normalising an arbitrary bigraph t, we define a normalisation relation $t \downarrow_{\mathbf{B}} t'$ for bigraph terms (details are given in Figure 22 of Appendix A.1), with the following property:

Proposition 1 *For any bigraph terms t,t', if t represents a bigraph b and $t \downarrow_{\mathbf{B}} t'$, then t' represents b as well, and is on B-normal form given in Figure 4.*

The relation is straightforward, recursively normalising subterms and recombining the results; for tensor product, the rule stated is

$$\text{Bten}\frac{\begin{array}{c} t_i \downarrow_{\mathbf{B}} (\omega_i \otimes \mathsf{id}_{(\vec{Y}_i)})D_i \qquad D_i \equiv \alpha_i \otimes (\bigotimes_{j\in n_i} P_i^j)\pi_i : I_i \to \langle n_i, \vec{Y}_i, Y_i\rangle \\ \omega = \bigotimes_{i\in n} \omega_i \quad \alpha = \bigotimes_{i\in n} \alpha_i \quad \mathsf{id}_{(\vec{Y})} = \bigotimes_{i\in n} \mathsf{id}_{(\vec{Y}_i)} \quad \pi = \bigotimes_{i\in n} \pi_i \\ P = \bigotimes_{j\in n} \bigotimes_{i\in n_j} P_i^j \qquad D \equiv \alpha \otimes P\pi \end{array}}{\bigotimes_{i\in n} t_i \downarrow_{\mathbf{B}} (\omega \otimes \mathsf{id}_{(\vec{Y})})D}.$$

We find that the expression $\bigotimes_{j\in n} \bigotimes_{i\in n_j} P_i^j$ in general will lead to name clashes, because we can only assume that outer, not inner names, of the ω_i's are disjoint.

One solution could be to rename names on P_i^j's outer face in the Bten rule. However, as Bten is applied recursively at each level of tensor product, this would lead to multiple renamings of the same names, causing inefficiency. Instead, we precede normalisation by a renaming phase described in the following; it will prevent name clashes in normalisation.

2.11 Renaming

While renaming names used in a term might look trivial at first sight, it is in fact not entirely straightforward. First, inner and outer names of a term must not be renamed, or we would be representing a different bigraph. Second, we cannot even require of a renamed term that all internal names are unique, as a normalised subterm can contain several instances of the same name, due to the use of id_Y in the normal form.

Thus, we need to identify a more refined notion of internal horizontal uniqueness, where a name can be reused vertically in link compositions, but not horizontally in tensor products. To this end, given a term t, we conceptually replace all occurrences of $/X$ by $e_1/x_1 \otimes \cdots \otimes e_n/x_n$, and $K_{\vec{y}(\vec{X})}$ by $K_{\vec{y}(\vec{e}/\vec{X})}$, in effect naming uniquely each closed link. We then define a function *linknames*, mapping terms to link namers (details are given in Figure 23 of Appendix A.2). Using this function we define a predicate *normalisable*, which identifies terms whose tensor products and compositions do not produce subterms with name clashes, and is preserved by normalisation (details are given in Figure 24 of Appendix A.2):

Proposition 2 *For any bigraph term t, if normalisable(t), there exists a t' such that $t \downarrow_{\mathbf{B}} t'$ and normalisable(t').*

For the actual renaming, we define inductively a renaming judgment $U \vdash \alpha, t \downarrow_\beta t', \beta \dashv V$, where U is a set of used names and α renames t's inner names to those of t', while β renames t's outer names to those of t' and V extends U with names used in t' (details are given in Figure 25 of Appendix A.2).

We can show that renaming preserves the bigraph, and enables normalisation:

Proposition 3 *Given a term t representing a bigraph $b : \langle m, \vec{X}, X\rangle \to \langle n, \vec{Y}, Y\rangle$, we can derive $X \cup Y \vdash \mathsf{id}_X, t \downarrow_\beta t'', \beta \dashv V$ for some t'', β, V, and set $t' = ((\beta^{\mathbf{glob}})^{-1} \otimes \widehat{(\beta^{\mathbf{loc}})^{-1}})t''$; then t' represents b, and normalisable(t').*

2.12 Regularising

As a regular bigraph can be expressed as a term containing permutations, we must define *regularising* to represent it as a permutation-free term. This is done by splitting the permutations in the D- and G-normal forms, recursively pushing them into the subterms where they reorder the tensor product of S_i's.

While D's permutation π must be a tensor product of π_i's—otherwise the bigraph would not be regular—G's permutation, on the other hand, need not be so. However, as the bigraph is regular, it must be possible to split it into a major permutation $\underline{\pi}^{\vec{X}}$ and n minor permutations $\pi_i^{\vec{X}}$, based on the local

inner faces, $\vec{X}$, of the S_i's. Then $\underline{\pi}^{\vec{X}}$ is elided by permuting the S_i's, and each $\pi_i^{\vec{X}}$ permutation is handled recursively in its S_i (details are given in Figure 26 of Appendix A.3).

We can show that regularisation is correct:

Proposition 4 *Given a term t representing a regular bigraph b, we can infer $t \hookrightarrow t'$, for some t' where t' contains no nontrivial permutations, and t' represents b.*

2.13 Summary

A detailed illustration of the entire reaction cycle including the preceding transformation technologies can be seen in Figure 5.

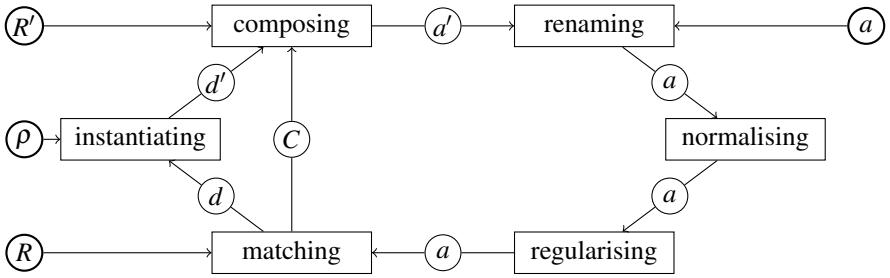


Figure 5: Details of the reaction cycle

3 Inferring Matches Using a Graph Representation

In this section we recap matching inference using a graph representation as developed in [9]; this representation is the basis for correctness proofs.

For simplicity, we will first consider just place graphs to explain the basic idea behind matching inference.

3.1 Matching place graphs

A place graph match is captured by a matching sentence:

Definition 5 (Matching Sentence for Place Graphs) *A* matching sentence *for place graphs is a 4-tuple of bigraphs $a, R \hookrightarrow C, d$, all are regular except C, with a and d ground. A sentence is* valid *iff $a = CRd$.*

We infer place graph matching sentences using the inference system given in Figure 6. Traversing an inference tree bottom-up, the agent is decomposed, while constructing the context, using the ION, MERGE and PAR rules. The PERM rule permutes redex parts to align tensor factors with corresponding agent factors.

At the point in the agent where a redex root should match, leaving a site in the context, the SWITCH rule is applied, switching the roles of the context and redex. This allows the remaining rules to be reused (above the switch rule) for checking that the redex matches the agent. When a site in the redex is reached, whatever is left of the agent should become (a part of) the parameter—this is captured by the PRIME-AXIOM rule.

For a match with a redex $R : m \to n$ consisting of n nontrivial (i.e., non-identity) primes, the inference tree will contain m applications of PRIME-AXIOM and n applications of SWITCH. Further, between any

$$\text{PRIME-AXIOM}\ \frac{}{p, \mathsf{id} \hookrightarrow \mathsf{id}, p} \qquad \text{ION}\ \frac{p, R \hookrightarrow P, d}{Kp, R \hookrightarrow KP, d} \qquad \text{SWITCH}\ \frac{p, \mathsf{id} \hookrightarrow P, d}{p, P \hookrightarrow \mathsf{id}_1, d}$$

$$\text{PAR}\ \frac{a, R \hookrightarrow C, d \qquad b, S \hookrightarrow D, e}{a \otimes b, R \otimes S \hookrightarrow C \otimes D, d \otimes e} \qquad \text{PERM}\ \frac{a, \bigotimes_i^n P_{\pi^{-1}(i)} \hookrightarrow C, \overline{\pi} d}{a, \bigotimes_i^n P_i \hookrightarrow C\pi, d}$$

$$\text{MERGE}\ \frac{a, R \hookrightarrow C, d}{merge\, a, R \hookrightarrow merge\, C, d}$$

Figure 6: Inference rules for deriving place graph matches

leaf and the root of the inference tree, SWITCH will be applied at most once. The structure of a matching inference tree will thus generally be as illustrated in Figure 7; rules applied above SWITCH match agent

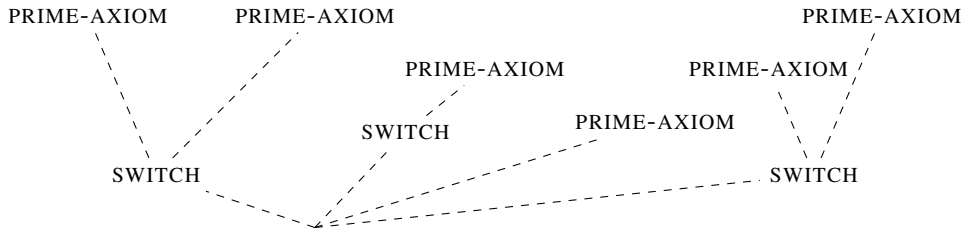


Figure 7: A sketch of the general structure of an inference tree for matching

and redex structure, while rules applied below match agent and context structure.

3.2 Matching binding bigraphs

Turning now to consider binding bigraphs, we extend the matching sentences to cater for links:

Definition 6 (Matching Sentence for Binding Bigraphs) *A* (binding bigraph) matching sentence *is a 7-tuple of bigraphs:* $\omega_{\mathbf{a}}, \omega_R, \omega_C \vdash a, R \hookrightarrow C, d$*, where* a, R, C *and* d *are discrete with local inner faces, all regular except* C*, with* a *and* d *ground. It is valid iff* $(\mathsf{id} \otimes \omega_{\mathbf{a}})a = (\mathsf{id} \otimes \omega_{\mathbf{C}})(\mathsf{id}_{Z \uplus V} \otimes C)(\mathsf{id}_Z \otimes (\mathsf{id} \otimes \omega_{\mathbf{R}})R)d$.

This definition separates the wirings, leaving local wiring in a, R, C and d, while keeping global wiring of agent, redex and context in $\omega_{\mathbf{a}}$, ω_R and ω_C, respectively; this is possible for any agent, redex and context [9]. The validity property shows how a valid matching sentence relates to a match, as illustrated in Figure 8.

To reach a system for inferring valid matching sentences for binding bigraphs, we simply augment the place graph rules with wirings as shown in Figure 9, and add three rules for dealing with purely wiring constructs, shown in Figure 10. A detailed explanation of the rules is available in the literature [9], along with proofs of soundness and completeness of the inference system.

4 From Graph Matching to Term Matching

In this section we transform the graph based inductive characterisation of matching to be based on a term representation in such a way that correctness and completeness is preserved.

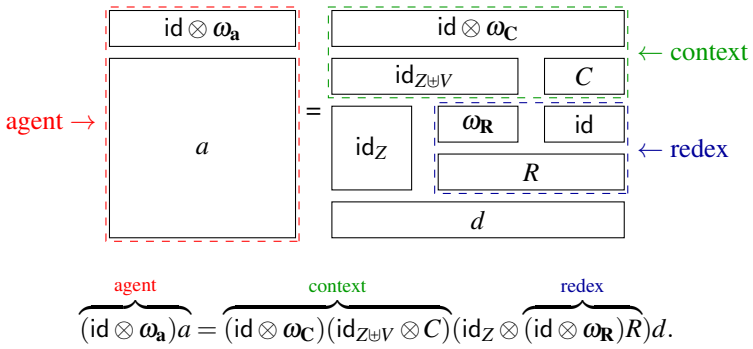


Figure 8: Decomposition of the bigraphs of a valid matching sentence

$$\text{PRIME-AXIOM}\ \frac{\sigma : W \uplus U \to \quad \beta : Z \to U \quad \alpha : V \to W \quad \tau : X \to V \quad p : \langle X \uplus Z \rangle}{\sigma(\beta \otimes \alpha\tau), \mathsf{id}_\varepsilon, \sigma \vdash p, \mathsf{id}_{(V)} \hookrightarrow \ulcorner \alpha \urcorner, (\beta \otimes \widehat{\tau})(X)p}$$

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

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

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

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

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

Figure 9: Place graph rules (shaded) augmented for deriving binding bigraph matches

$$\text{WIRING-AXIOM}\ \frac{}{y, X, y/X \vdash \mathsf{id}_\varepsilon, \mathsf{id}_\varepsilon \hookrightarrow \mathsf{id}_\varepsilon, \mathsf{id}_\varepsilon}$$

$$\text{ABSTR}\ \frac{\sigma_\mathbf{a} \otimes \omega_\mathbf{a}, \omega_\mathbf{R}, \sigma_\mathbf{C} \otimes \omega_\mathbf{C} \vdash p, R \hookrightarrow P, d \qquad \sigma_\mathbf{a} : Z \to W \quad p : \langle Z \uplus Y \rangle \quad \sigma_\mathbf{C} : U \to W \quad P : \to \langle U \uplus X \rangle}{\omega_\mathbf{a}, \omega_\mathbf{R}, \omega_\mathbf{C} \vdash (\widehat{\sigma_\mathbf{a}} \otimes \mathsf{id}_Y)(Z)p, R \hookrightarrow (\widehat{\sigma_\mathbf{C}} \otimes \mathsf{id}_X)(U)P, d}$$

$$\text{CLOSE}\ \frac{\sigma_\mathbf{a}, \sigma_\mathbf{R}, \mathsf{id}_{Y_\mathbf{R}} \otimes \sigma_\mathbf{C} \vdash a, R \hookrightarrow C, d \qquad \sigma_\mathbf{a} : \to U \uplus Y_R \qquad \sigma_\mathbf{R} : \to V \uplus Y_R \qquad \sigma_\mathbf{C} : \to W \uplus Y_C}{(\mathsf{id}_U \otimes /(Y_\mathbf{R} \uplus Y_\mathbf{C}))\sigma_\mathbf{a}, (\mathsf{id}_V \otimes /Y_\mathbf{R})\sigma_\mathbf{R}, (\mathsf{id}_W \otimes /Y_\mathbf{C})\sigma_\mathbf{C} \vdash a, R \hookrightarrow C, d}$$

Figure 10: Added inference rules for deriving binding bigraph matches

While the graph representation of matching sentences is useful for constructing a relatively simple inference system amenable to correctness proofs, it is not sufficient for an implementation based on syntax, that is, bigraph terms. One bigraph can be represented by several different bigraph terms that are structurally congruent by the axiom rules: $a = a \otimes \mathsf{id}_0 = merge_1\, a$, $a \otimes (b \otimes c) = (a \otimes b) \otimes c$ and $merge(a \otimes b) = merge(b \otimes a)$. If, for instance, we were to match agent $a = merge((\mathsf{K} \otimes \mathsf{L}) \otimes \mathsf{M})$ with redex $R = \mathsf{K}$, we would first need to apply the axioms to achieve $R = merge((\mathsf{K} \otimes \mathsf{id}_0) \otimes \mathsf{id}_0)$ before being able to apply the MERGE and PAR rules.

In the following, we recast the matching sentences to be tuples of 3 wirings and 4 bigraph *terms* $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_C \vdash a, R \rightsquigarrow C, d$, with the same restrictions and validity as before, interpreting the terms as the bigraphs they represent. Given this, adding just this one rule would be sufficient to achieve completeness of the inference system:

$$\textsc{struct}\ \frac{a \equiv a' \qquad R \equiv R' \qquad C \equiv C' \qquad h \equiv h' \qquad \omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a', R' \rightsquigarrow C', h'}{\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, R \rightsquigarrow C, h}$$

The STRUCT rule says that we can apply structural congruence to rewrite any term a, R, C or h to a term denoting the same bigraph. With the help of the equational theory for determining bigraph isomorphism on the term level [8], we have essentially a nondeterministic algorithm for matching bigraph terms—implementable in say, Prolog. A brief glance at the equational theory, shows us, though, that the associative and commutative properties of the basic operators of the language would yield a wildly nondeterminstic inference system, since we would need to apply structural congruence between every step to infer a match. This is reminiscent of the problems in implementing *rewriting logic*, that is, term rewriting modulo a set of static equivalences [5, 6, 16]. Consequentially, we abandon the fully general STRUCT rule. For the purposes of stating the completeness theorem below, we shall need to refer to sentences derived from the ruleset for bigraphs (i.e., from section 3.2) recast to terms with the help of the STRUCT rule above. We shall write such sentences $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, R \rightsquigarrow_{\mathrm{S}} C, h$ for wirings $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}}$ and terms a, R, C and h.

Definition 7 *For wirings $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}}$ and terms a, R, C and h, sentences $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, R \rightsquigarrow_{\mathrm{S}} C, h$ range over sentences derived from the rules of Figure 10—reading a, R, C and h as terms—extended with the* STRUCT *rule.*

5 Normal Inferences

Next, we look at how to restrict the term based inductive characterisation of matching as an enabling step for designing an algorithm. We define normal inferences, limiting the set of inferences we need to consider.

Normal inferences is a class of inferences that are complete in the sense that all valid matching sentences can be inferred, but suitably restricted, such that inferences can be built mechanically. In particular, normal inference definitions for term matching spell out *how* and *where* to apply structural congruence. As a main trick, we utilize a variant of the normal forms proven complete for binding bigraphs (cf. Section 2.9), lending us a set of uniform representations of classes of bigraphs based directly on terms for bigraphs; we define normal inferences that require each inference to start by rewriting the term to be on normal form.

Before giving the format for normal inferences, we incorporate structural congruence axioms into PRODUCT and MERGE rules. We derive rules for iterated tensor product and permutations under merge, arriving at the inference system shown in Figure 11. In this inference system, the terms in the conclusion of every rule except DNF is in some normal form as given by Figure 4, where e is a discrete prime (p) or global discrete prime (g). An expression $[\![t]\!]^G$ means term t expressed on G-normal form—for

$$\text{PAX}\ \frac{\sigma : W \uplus Z \to \quad \alpha : V \to W \quad \tau : X \to V \quad g : \langle X \uplus Z \rangle}{\sigma(\mathsf{id}_Z \otimes \alpha\tau), \mathsf{id}_\varepsilon, \sigma \vdash g, [\![\mathsf{id}_{(V)}]\!]^{\overline{P}} \rightsquigarrow [\![\ulcorner \alpha \urcorner]\!]^G, [\![(\mathsf{id}_Z \otimes \widehat{\tau})(X)g]\!]^{\overline{P}}}$$

$$\text{ION}\ \frac{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash (\mathsf{id} \otimes (\vec{v})/(\vec{X}))n, \overline{P} \rightsquigarrow (\mathsf{id} \otimes (\vec{v})/(\vec{Z}))N, \overline{q} \quad \alpha = \vec{y}/\vec{u} \quad \sigma : \{\vec{y}\} \to}{(\sigma \parallel \sigma^{\mathbf{a}}), \sigma^{\mathbf{R}}, (\sigma\alpha \parallel \sigma^{\mathbf{C}}) \vdash [\![(\mathsf{id}_U \otimes K_{\vec{y}(\vec{X})})n]\!]^G, \overline{P} \rightsquigarrow [\![(\mathsf{id}_W \otimes K_{\vec{u}(\vec{Z})})N]\!]^G, \overline{q}}$$

$$\text{SWX}\ \frac{\begin{array}{c}\sigma : W \to U \quad G : \to \langle W \uplus Y \rangle \quad \overline{q} : \langle n, \vec{X}, X \uplus Z \rangle \quad X = \{\vec{X}\} \\ \sigma^{\mathbf{a}}, \mathsf{id}_\varepsilon, \sigma^{\mathbf{C}}(\mathsf{id}_Z \otimes \sigma \otimes \sigma^{\mathbf{R}}) \vdash g, [\![\bigotimes_i^n \mathsf{id}_{(X_i)}]\!]^{\overline{P}} \rightsquigarrow G, \overline{q}\end{array}}{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash g, [\![(\mathsf{id}_Y \otimes \widehat{\sigma})(W)G]\!]^{\overline{P}} \rightsquigarrow [\![\ulcorner U \urcorner]\!]^G, \overline{q}}$$

$$\text{PAR}^{\mathscr{E}}_{\mathbf{n}}\ \frac{\sigma' : I_{\mathbf{R}} \to I_{\mathbf{a}} \quad (\forall i \in n)\ \sigma_i^{\mathbf{a}}, \sigma_i^{\mathbf{R}}, \sigma \parallel \sigma_i^{\mathbf{C}} \vdash e_i, \overline{P}_i \rightsquigarrow E_i, \overline{q}_i}{\big(I_{\mathbf{a}} \parallel \|_i^n \sigma_i^{\mathbf{a}}\big), \big(I_{\mathbf{R}} \parallel \|_i^n \sigma_i^{\mathbf{R}}\big), \big(\sigma' \parallel \sigma \parallel \|_i^n \sigma_i^{\mathbf{C}}\big) \vdash \bigotimes_i^n e_i, \bigotimes_i^n \overline{P}_i \rightsquigarrow \bigotimes_i^n E_i, \bigotimes_i^n \overline{q}_i}$$

$$\text{PAR}^{\mathscr{E}}_{\equiv}\ \frac{\begin{array}{c}P'_{ij} = P_{j+\sum_{r\in i} l_r} \quad q'_{ij} = q_{j+\sum_{r\in i} k_r} \quad P'_{ij} : \langle k_{ij}, \vec{X}_{ij} \rangle \to \quad k_i = \sum_{j\in l_i} k_{ij} \\ \sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash \bigotimes_i^n e_i, \bigotimes_i^n \bigotimes_j^{l_i} P'_{ij} \rightsquigarrow \bigotimes_i^n E_i, \bigotimes_i^n \bigotimes_j^{k_i} q'_{ij}\end{array}}{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash \bigotimes_i^n e_i, \bigotimes_i^m P_i \rightsquigarrow \bigotimes_i^n E_i, \bigotimes_i^{m'} q_i}$$

$$\text{PER}^{\mathscr{E}}\ \frac{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash \overline{e}, \bigotimes_i^n Q_{\pi^{-1}(i)} \rightsquigarrow \overline{E}, \bigotimes_i^m q_{\bar{\pi}^{-1}(i)}}{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash \overline{e}, \bigotimes_i^n Q_i \rightsquigarrow \overline{E}\pi, \bigotimes_i^m q_i}$$

$$\text{MER}\ \frac{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash \bigotimes_i^m (\mathsf{id} \otimes merge) \bigotimes_{j\in\rho_i, \rho\in\tilde{\rho}(n,m)} m_j, \overline{P} \rightsquigarrow (\bigotimes_i^m [\![S_{\pi^{-1}(i)}]\!]^G)\bar{\pi}, \overline{q}}{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash (\mathsf{id} \otimes merge) \bigotimes_i^n m_i, \overline{P} \rightsquigarrow (\mathsf{id} \otimes merge) \bigotimes_i^m S_i, \overline{q}}$$

$$\text{ABS}\ \frac{\sigma_L^{\mathbf{a}} \otimes \sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma_L^{\mathbf{C}} \otimes \sigma^{\mathbf{C}} \vdash g, \overline{P} \rightsquigarrow G, \overline{q} \quad \sigma_L^{\mathbf{a}} : Z \to W \quad \sigma_L^{\mathbf{C}} : U \to W \quad G : \to \langle U \uplus X \rangle}{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \sigma^{\mathbf{C}} \vdash (\mathsf{id} \otimes \widehat{\sigma_L^{\mathbf{a}}})(Z)g, \overline{P} \rightsquigarrow (\mathsf{id} \otimes \widehat{\sigma_L^{\mathbf{C}}})(U)G, \overline{q}}$$

$$\text{CLO}\ \frac{\sigma^{\mathbf{a}}, \sigma^{\mathbf{R}}, \mathsf{id}_{Y_{\mathbf{R}}} \otimes \sigma^{\mathbf{C}} \vdash \overline{p}, \overline{P} \rightsquigarrow \overline{Q}\pi, \overline{q}}{(\mathsf{id} \otimes /(Y_{\mathbf{R}} \uplus Y_{\mathbf{C}}))\sigma^{\mathbf{a}}, (\mathsf{id} \otimes /Y_{\mathbf{R}})\sigma^{\mathbf{R}}, (\mathsf{id} \otimes /Y_{\mathbf{C}})\sigma^{\mathbf{C}} \vdash \overline{p}, \overline{P} \rightsquigarrow \overline{Q}\pi, \overline{q}}$$

$$\text{DNF}\ \frac{\begin{array}{c}a \equiv \overline{p} \quad R \equiv \overline{P} \quad C \equiv \overline{Q}\pi \quad h \equiv \overline{q} \quad \overline{p}, \overline{P}, \overline{Q}, \overline{q} \text{ are on normal form} \quad R \text{ is regular} \\ \omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash \overline{p}, \overline{P} \rightsquigarrow \overline{Q}\pi, \overline{q}\end{array}}{\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, R \rightsquigarrow C, h}$$

Figure 11: Inference rules for binding bigraph terms

instance, $[\![\ulcorner \alpha \urcorner]\!]^G$ means $(\mathsf{id}_Y \otimes \mathit{merge}_1)(\bigotimes_i^1 \ulcorner \alpha \urcorner)$—and similarly for the remaining normal forms. The expression $\bar{\rho}(n,m)$ denotes the set of n-m-partitions. An n-m-partition ρ is a partition of $\{0,\ldots,n-1\}$ into m (possibly empty) subsets, and for $i \in m$, ρ_i is the ith subset. Given a metavariable $\mathscr{X}$, $\overline{\mathscr{X}}$ ranges over iterated tensor products of $\mathscr{X}$'es. As indicated by the superscript, rules $\text{PER}^{\mathscr{E}}$, $\text{PAR}_{\mathbf{n}}^{\mathscr{E}}$ and $\text{PAR}_{\equiv}^{\mathscr{E}}$ can be used either on discrete primes p and P or global discrete primes g and G.

The main differences from the preceding inference system is that we have replaced the binary PAR rule by two iterative PAR rules, $\text{PAR}_{\mathbf{n}}^{\mathscr{E}}$ and $\text{PAR}_{\equiv}^{\mathscr{E}}$, and specialised the MERGE rule into a rule, MER, that makes the partitioning of children in an agent node explicit. The $\text{PAR}_{\equiv}^{\mathscr{E}}$ rule splits up an iterated tensor product into a number of products matching agent factors, while $\text{PAR}_{\mathbf{n}}^{\mathscr{E}}$ performs the actual inductive inference on each of the factors. (Note, by the way, that $\text{PAR}_{\equiv}^{\mathscr{E}}$ and $\text{MER}_{\equiv}^{\mathscr{E}}$ correspond just to particular instances of the STRUCT-rule, that we abandoned above.)

Furthermore, note that the usage of the previous WIRING-AXIOM-rule for introducing idle linkage has been inlined to a side-condition on a slightly generalized PAR-rule (i.e., the $\text{PAR}_{\mathbf{n}}^{\mathscr{E}}$-rule). The σ' in that rule allows us to introduce idle linkage in redex and agent, and link them in context; as previously allowed by the WIRING-AXIOM-rule. Hence, $\text{PAR}_{\mathbf{n}}^{\mathscr{E}}$ also serves as an axiom, introducing 0-ary products of id_ε's on G- and P-normal forms.

While this inference system is more explicit about partitioning tensor products (in the MER and $\text{PAR}_{\equiv}^{\mathscr{E}}$ rules), there is still a lot of nondeterministic choice left in the *order* in which the rules can be applied. To limit this, we define normal inferences based, essentially, on the order rules were used in the proof of completeness [9]. We derive a sufficient order that still preserves completeness:

Definition 8 (Normal Inference) *A* normal inference *is a derivation using the term matching rules of Figure 11 in the order specified by the context free grammar given in Figure 12.*

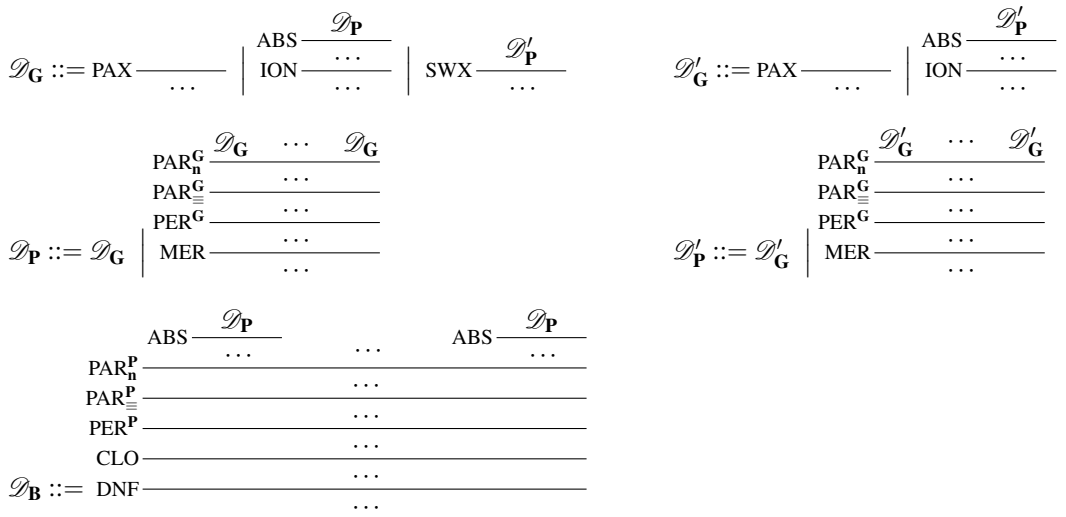


Figure 12: Grammar (BNF) for normal inferences for binding bigraphs with start symbol $\mathscr{D}_{\mathbf{B}}$

Now we can give the main theorem stating that normal inferences are sufficient for finding all valid matches. The following theorem states formally for every sentence derivable with the ruleset for bigraphs recast to bigraph terms by extending with STRUCT, that such a sentence is also derivable as a normal inference.

Theorem 9 (Normal inferences are sound and complete) *For wirings $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}}$ and terms a,R,C,d, we can infer $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a,R \rightsquigarrow_{\mathrm{S}} C,d$ iff we can infer $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a,R \rightsquigarrow C,d$ using a normal inference.*

Proof. (*Sketch*) By induction over the structure of the derivation of the sentence $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, R \rightsquigarrow_{\mathrm{S}} C, d$. We case on the last rule used to conclude this sentence. By the induction hypothesis (IH), we can conclude a normal derivation of the sentence used for concluding $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, R \rightsquigarrow_{\mathrm{S}} C, d$.

STRUCT: By IH, we can construct a normal derivation of $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a', R' \rightsquigarrow C', d'$, with $a = a'$, $R' = R$, $C' = C$ and $d' = d$. This normal derivation can be used directly to conclude also $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a', R' \rightsquigarrow C', d'$.

PRIME-AXIOM: We produce the needed normal inference by starting with an application of PAX, which introduces the needed prime bigraphs and wiring—that is, each term being equal up to structural congruence to the sentence concluded with PRIME-AXIOM. Now we proceed to build the needed normal inference by a building first a $\mathscr{D}_{\mathbf{P}}$ and then a $\mathscr{D}_{\mathbf{B}}$-inference. All steps add only term structure to match a particular normal form, while not changing the denotation of the terms.

ION: By IH, we can construct a normal derivation of $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash ((\vec{v})/(\vec{X}) \otimes \mathsf{id}_U)p, R \rightsquigarrow ((\vec{v})/(\vec{Z}) \otimes \mathsf{id}_W)P, d$. For this case, we have to unroll that normal derivation up across the $\mathscr{D}_{\mathbf{B}}$ production except for the last ABS-step, concluding with a $\mathrm{PAR}_1^{\mathbf{P}}$ step (since we know p and P are prime). We now have a $\mathscr{D}_{\mathbf{P}}$ normal inference with an added ABS-step, which we can use for concluding an ION-step introducing our needed ion. Referring to the grammar in Figure 12, we see that this produces a $\mathscr{D}_{\mathbf{G}}$-inference, which we have to lead through two series of PAR-PER-MER steps (and one ABS-step), to produce a full normal inference.

SWITCH: This case needs a little extra care. First, we point out two properties of normal derivations: (i) any $\mathscr{D}_{\mathbf{G}}$ and $\mathscr{D}_{\mathbf{P}}$ inference without SWX is also a $\mathscr{D}'_{\mathbf{G}}$ or $\mathscr{D}'_{\mathbf{P}}$ inference, respectively; and, (ii) any sentence, $\omega_{\mathbf{a}}, \mathsf{id}_\varepsilon, \omega_{\mathbf{C}} \vdash a, \mathsf{id} \rightsquigarrow C, h$ has a normal derivation with no SWX-steps. Both are easily verified.

Now, by the IH we can construct a normal derivation of a sentence $\omega_{\mathbf{a}}, \mathsf{id}_\varepsilon, \omega_{\mathbf{C}}(\sigma \otimes \omega_{\mathbf{R}} \otimes \mathsf{id}_Z) \vdash p, \mathsf{id} \rightsquigarrow P, d$ for global P. By property (ii), we can assume that this normal derivation does not contain any applications of SWX. We unroll this normal derivation up across the whole $\mathscr{D}_{\mathbf{B}}$ production, . This leaves us with a $\mathscr{D}_{\mathbf{P}}$-type normal derivation, which by property (i), we can use also as $\mathscr{D}'_{\mathbf{P}}$ derivation. Hence, we can apply SWX to obtain a $\mathscr{D}_{\mathbf{G}}$ derivation. We proceed to build first a $\mathscr{D}_{\mathbf{P}}$ type inference, and then a $\mathscr{D}_{\mathbf{B}}$ type inference, in particular applying again ABS to introduce local linkage in p.

PAR: By IH, we can construct normal derivations of $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \parallel \omega \vdash a, R \rightsquigarrow C, d$ and $\omega_{\mathbf{b}}, \omega_{\mathbf{S}}, \omega_{\mathbf{D}} \parallel \omega \vdash b, S \rightsquigarrow D, e$. Each of these normal derivations we can unroll up to the last application of $\mathrm{PAR}_{\mathbf{n}}^{\mathbf{P}}$ $\mathscr{D}_i$ and $\mathscr{E}_j$, applied for concluding these $\mathrm{PAR}_{\mathbf{n}}^{\mathbf{P}}$ steps. To construct the required normal inference we simply let instead a single $\mathrm{PAR}_{\mathbf{n}}^{\mathbf{P}}$ step utilize all of the normal inferences $\mathscr{D}_i$ and $\mathscr{E}_j$.

PERM: By IH, we can construct a normal derivation of $\omega^{\mathbf{a}}, \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash a, \bigotimes_i^m P_{\pi(i)} \rightsquigarrow C, (\overline{\pi} \otimes \mathsf{id}_Z)d$. Unrolling this normal derivation up through the applications of DNF, CLO, and $\mathrm{PER}^{\mathbf{P}}$, we can edit the $\mathrm{PER}^{\mathbf{P}}$-step to also move the permutation π to the context.

MERGE: By IH, we can construct a normal derivation of $\omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \omega_{\mathbf{C}} \vdash a, R \rightsquigarrow C, d$ for global a and C. We unroll this derivation up across the $\mathscr{D}_{\mathbf{B}}$ production to obtain n $\mathscr{D}_{\mathbf{P}}$-derivations (for a and C of width n). We may consider these as $\mathscr{D}_{\mathbf{G}}$-derivations, also. We combine these in a single application of $\mathrm{PAR}_{\mathbf{n}}^{\mathbf{G}}$, and, after a $\mathrm{PAR}_{\equiv}^{\mathbf{P}}$ and a PER-step, we apply MER to merge the roots as required by the case. We conclude by adding term structure to the terms of this $\mathscr{D}_{\mathbf{P}}$-inference as required by the normal form and lead it through the steps to produce a $\mathscr{D}_{\mathbf{B}}$-derivation.

WIRING-AXIOM: As sketched in the text above, introduction of idle names is now handled by $\mathrm{PAR}_n^{\mathbf{P}}$. For this case, we simply start with a $\mathrm{PAR}_0^{\mathbf{P}}$-step and proceed through the grammar for $\mathscr{D}_{\mathbf{B}}$ to produce a normal inference as needed.

ABSTR: By IH, we can construct a normal derivation of $\sigma_{\mathbf{a}} \otimes \omega_{\mathbf{a}}, \omega_{\mathbf{R}}, \sigma_{\mathbf{C}} \otimes \omega_{\mathbf{C}} \vdash p, R \rightsquigarrow P, d$. We unroll this normal derivation up across the entire $\mathscr{D}_{\mathbf{B}}$-inference to obtain a $\mathscr{D}_{\mathbf{P}}$ type inference. (We know there is only one $\mathscr{D}_{\mathbf{P}}$-inference, as p and P are prime.) We construct the required $\mathscr{D}_{\mathbf{B}}$ inference by starting with a modified ABS-step, where we introduce the required abstractions and local substitutions.

CLOSE: By IH, we can construct a normal inference for a sentence with only substitutions (i.e., with no closed links). We simply unroll this normal inference up across the CLO-step, and instead, to produce

the needed normal inference, close the needed names in a new CLO-step. □

Normal inferences are sufficiently restricted such that we can base our prototype implementation on mechanically constructing them.

6 Bigraph Matching Algorithm

In this section, we show how to interpret the inference rules as functions—achieving an implementation proven correct in great detail.

In general, an inference tree can be divided into two parts: the twigs of the tree, consisting of all the rule applications above a SWX rule application, and the base of the tree, consisting of the remaining rule applications, cf. Figure 13. The base and the SWX rule applications determine the locations in the agent

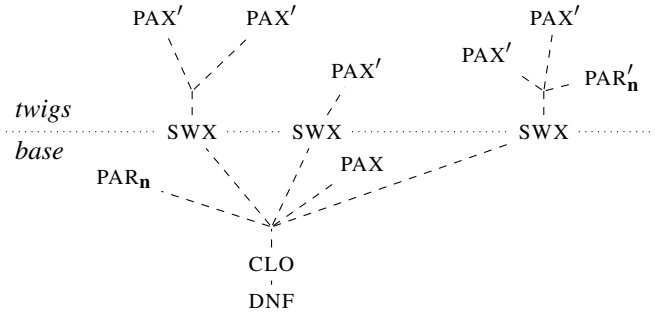


Figure 13: A sketch of the general structure of a normal inference tree for matching; dashed lines represent rules omitted for brevity

at which the roots of the redex are matched, and each twig determines how the place graph subtree below a redex root is matched to the corresponding subtree in the agent. Each PAX leaf determines a component of the parameter, and each $\mathrm{PAR}_{\mathbf{n}}$ leaf corresponds to a leaf node in the agent.

We turn the declarative matching specification of the preceding section into a matching algorithm by considering each rule operationally. By implementing the inference system faitfully rule by rule, we ensure that the proof of completeness is valid also for the implemented algorithm.

All rules of Figure 11 except SWX, CLO and DNF can be applied in two flavours: below and above a SWX rule. We distinguish these applications using primes ($'$) for rules applied above a SWX rule.

The DNF rule is concerned with normalising a and R, which is done using the algorithm described in Section 2.10, so for the remaining rules of the base of the tree we have as input to the algorithm wirings $\omega^{\mathbf{a}}$, $\omega^{\mathbf{R}}$ and discrete bigraphs $\bar{p}$ (agent) and $\bar{P}$ (redex). The goal is to find context wiring $\omega^{\mathbf{C}}$, context term $\bar{Q}\pi$ and parameter term $\bar{q}$.

In general, there are zero or more matches, so in the implementation the application of each rule returns a lazy list of matches, each containing a context wiring, along with context and parameter bigraphs.

As the matching inference starts at the root of the inference tree, we will do the same, considering each rule of Figure 11 in turn, and giving an operational interpretation of it:

6.1 Rule Applications in the Base of the Tree

CLO: Operationally, CLO opens $\omega^{\mathbf{a}}$ and $\omega^{\mathbf{R}}$, producing $\sigma^{\mathbf{a}}$ and $\sigma^{\mathbf{R}}$ and assigning fresh names $Y_{\mathbf{R}}$ to edges in $\omega^{\mathbf{R}}$.

[Aside: We cannot generally conclude there are no matches if the number of edges in $\omega^{\mathbf{a}}$ is smaller than the number of edges in $\omega^{\mathbf{R}}$! But the number of port-connected edges in $\omega^{\mathbf{R}}$ must be greater than or equal to the number of edges in $\omega^{\mathbf{a}}$. Unfortunately, port-connectedness is expensive to determine in the term representation.]

In the implementation, we represent $\sigma^{\mathbf{R}}$ as $\sigma^{\mathbf{R}}_{\mathbf{e}} \otimes \sigma^{\mathbf{R}}_{\mathbf{n}}$, where $\sigma^{\mathbf{R}}_{\mathbf{n}}$ contains all the outer names of $\omega^{\mathbf{R}}$, and $\sigma^{\mathbf{R}}_{\mathbf{e}} : X_{\mathbf{e}} \to Y_{\mathbf{R}}$ contains the edges of $\omega^{\mathbf{R}}$ for some $X_{\mathbf{e}}$.

As we do not yet know which edges in $\omega^{\mathbf{a}} : \to W$ match which names in $Y_{\mathbf{R}}$, we represent $\sigma^{\mathbf{a}}$ as $\alpha\sigma^{\mathbf{a}}_{\mathbf{e}} \otimes \sigma^{\mathbf{a}}_{\mathbf{n}}$, where $\sigma^{\mathbf{a}}_{\mathbf{n}}$ contains all the open links of $\omega^{\mathbf{a}}$, and α is to be constructed during (i.e., returned by) the rest of the inference.

Inference must be done under the following condition:

- links in $\sigma^{\mathbf{R}}_{\mathbf{e}}$ must only be matched with links in $\alpha\sigma^{\mathbf{a}}_{\mathbf{e}}$

because redex edges can only match agent *edges,* not open links.

When the premise has been inferred, yielding $\mathsf{id}_{Y_{\mathbf{R}}} \otimes \sigma^{\mathbf{C}}$, we determine $\sigma^{\mathbf{C}} : \to W \uplus Y_{\mathbf{C}}$ (as $Y_{\mathbf{R}}$ is known), and then $Y_{\mathbf{C}} = (W \uplus Y_{\mathbf{C}}) \setminus W$. Finally, $\omega^{\mathbf{C}} = (\mathsf{id} \otimes /_{Y_{\mathbf{C}}})\sigma^{\mathbf{C}}$.

ABS: During the inference, ABS adds the links $\sigma^{\mathbf{a}}_{\mathbf{L}}$ to $\sigma^{\mathbf{a}}_{\mathbf{n}}$; the inner names of $\sigma^{\mathbf{a}}_{\mathbf{L}}$ are collected in a set L.

As $\sigma^{\mathbf{a}}_{\mathbf{L}}$ contains links bound in the agent, inference must be done under the following condition:

- links in $\sigma^{\mathbf{a}}_{\mathbf{n}}$ whose inner names are in L must not be matched via $\sigma^{\mathbf{R}}$, but must be matched via σ (the local outer names of the redex) in the SWX rule.

This enforces the scoping rule for the resulting context bigraph.

When the premise of ABS has been inferred, $\sigma^{\mathbf{C}}_{\mathbf{L}}$ is computed by restricting the outer face of the context wiring to W, the outer names of $\sigma^{\mathbf{a}}_{\mathbf{L}}$.

MER: Taking r be the outer width of redex $\bar{P}$, we let $m = r+1$, and compute all partitions of width m. Setting $m > r$ allows parts of the agent to not be matched by the redex ($l_i = 0$ in $\mathrm{PAR}^{\mathscr{E}}_{\mathbf{n}}$), that is, to become part of the context. For each partition, the premise is inferred, and if the permutation $\bar{\pi}$ in the returned context really is a pushthrough of some permutation π, the factors S_i of the tensor product are permuted before they are returned.

PER: For each n-permutation π, $\bar{\pi}$ is computed by pushing π through $\bigotimes^i_n Q_i$. After the premise has been inferred on the permuted redex primes, $\bar{\pi}$ is used to permute the resulting parameter primes accordingly before they are returned.

$\mathrm{PAR}^{\mathscr{E}}_{=}$: For each split of m into n parts, $\bigotimes^n_i \bigotimes^{l_i}_j P'_{ij}$ is computed, and after the premise has been inferred, the factors of the resulting parameter tensor product are concatenated into one tensor product before returning context and parameter.

$\mathrm{PAR}^{\mathscr{E}}_{\mathbf{n}}$: For each i, the global outer names of e_i is used to compute $\sigma^{\mathbf{a}}_{\mathbf{e},i}, \sigma^{\mathbf{a}}_{\mathbf{n},i}$ by restriction, and similarly for $\bar{P}_i$ and $\sigma^{\mathbf{R}}_{\mathbf{e},i}, \sigma^{\mathbf{R}}_{\mathbf{n},i}$. The context wirings resulting from inferring the premise are combined using parallel product, but allowing inner name clashes as long as each operand maps the inner names to the same outer name.

ION: Letting $Y = \{\vec{y}\}$, we split $\sigma^{\mathbf{a}}_{\mathbf{e}} = \sigma^{Y}_{\mathbf{e}} \parallel \sigma^{\mathbf{a}'}_{\mathbf{e}}$ according to whether the inner names are in Y or not; and similarly for $\sigma^{\mathbf{a}}_{\mathbf{n}} = \sigma^{Y}_{\mathbf{n}} \parallel \sigma^{\mathbf{a}'}_{\mathbf{n}}$. Fresh names $\vec{v}$ are created

6.2 Application of the SWX Rule

As the SWX rule preserves $\sigma^{\mathbf{a}}$, its representation is not changed above the SWX rule. As CLO requires the resulting $\sigma^{\mathbf{C}}$ to be of the form $\mathsf{id}_{Y_{\mathbf{R}}} \otimes \sigma^{\mathbf{C}}$, where $\sigma^{\mathbf{R}}_{\mathbf{e}} : \to Y_{\mathbf{R}}$, the third wiring above SWX is $(\mathsf{id}_{Y_{\mathbf{R}}} \otimes \sigma^{\mathbf{C}})(\mathsf{id}_Z \otimes \sigma \otimes \sigma^{\mathbf{R}}_{\mathbf{e}} \otimes \sigma^{\mathbf{R}}_{\mathbf{n}}) = \sigma^{\mathbf{R}}_{\mathbf{e}} \otimes \sigma^{\mathbf{C}}(\mathsf{id}_Z \otimes \sigma \otimes \sigma^{\mathbf{R}}_{\mathbf{n}})$. Letting $\sigma^{\mathbf{C}}_{\mathbf{n}} = \sigma \otimes \sigma^{\mathbf{R}}_{\mathbf{n}}$, we thus represent the context wiring above the rule by $\sigma^{\mathbf{C}}_{\mathbf{e}} \otimes \sigma^{\mathbf{C}}(\mathsf{id}_Z \otimes \sigma^{\mathbf{C}}_{\mathbf{n}})$.

In the twigs we are thus given wirings $\sigma^{\mathbf{a}}_{\mathbf{e}}, \sigma^{\mathbf{a}}_{\mathbf{n}}, \sigma^{\mathbf{R}}_{\mathbf{e}}, \sigma^{\mathbf{R}}_{\mathbf{n}}, \sigma^{\mathbf{C}}_{\mathbf{e}}, \sigma^{\mathbf{C}}_{\mathbf{n}}$ and terms $\bar{p}$ (agent) and $\bar{P}$ (redex). The goal is to check that $\bar{p}$ matches $\bar{P}$, and find $\sigma^{\mathbf{C}}, Z, \alpha$ and $\bar{q}$, as we want $\omega^{\mathbf{a}} = \alpha\sigma^{\mathbf{a}}_{\mathbf{e}} \otimes \sigma^{\mathbf{a}}_{\mathbf{n}}$, $\omega^{\mathbf{R}} = \sigma^{\mathbf{R}}_{\mathbf{e}} \otimes \sigma^{\mathbf{R}}_{\mathbf{n}}$, and $\omega^{\mathbf{R}} = \sigma^{\mathbf{C}}_{\mathbf{e}} \otimes \sigma^{\mathbf{C}}(\mathsf{id}_Z \otimes \sigma^{\mathbf{C}}_{\mathbf{n}})$ in the judgment $\omega^{\mathbf{a}},\ \omega^{\mathbf{R}}, \omega^{\mathbf{C}} \vdash \bar{p}, \mathsf{id} \rightsquigarrow \bar{P}, \bar{q}$ above the SWX rule.

6.3 Rule Applications in the Twigs

ABS′: During the inference, the ABS′ rule adds links to $\sigma^{\mathbf{C}}$. By adding them to $\sigma^{\mathbf{C}}_{\mathbf{e}}$, not $\sigma^{\mathbf{C}}_{\mathbf{n}}$, they are treated like internal edges in R, and thus not linked to the parameter via id_Z.

MER′: For each m-permutation π, $\bar{\pi}$ is computed; for each partition $\rho \in \bar{\rho}(n,m)$ of n into m (possibly empty) subsets, the tensor product of m_j's are computed, and then the premise is inferred, returning $\sigma^{\mathbf{C}}, \alpha$ and $\bar{q}$.

PER′: The premise is inferred, and the resulting $\bar{q}$ are permuted using $\bar{\pi}$ before they are returned.

PAR′$_{\equiv}$: The premise is inferred, and the resulting list of n tensor products are concatenated and returned as one product $\otimes_i^m q_i$.

PAR′$_{\mathbf{n}}$: Taking PAR′ literally, $\sigma^{\mathbf{a}}$ and $\sigma^{\mathbf{C}}$ must be split when performing the subinferences. However, as the inner face of $\sigma^{\mathbf{a}}$ must always match the global outer face of a, explicit splitting of $\sigma^{\mathbf{a}}_{\mathbf{e}}, \sigma^{\mathbf{a}}_{\mathbf{n}}, \sigma^{\mathbf{C}}_{\mathbf{e}}, \sigma^{\mathbf{C}}_{\mathbf{n}}$ can be avoided. This also implies that (1) need only be solved for links mapping outer names of g (i.e., $X \uplus Z$).

ION′: Deconstructing agent and context ions, their controls are checked for equality; for each $u_i \in \mathrm{dom}(\sigma^{\mathbf{C}}_{\mathbf{e}})$ we update α' so that $\alpha'\sigma^{\mathbf{a}}_{\mathbf{e}}(y_i) = \sigma^{\mathbf{C}}_{\mathbf{e}}(u_i)$. Using a fresh $\vec{v}$, the premise is inferred, and α' and $\sigma^{\mathbf{C}\prime}$ are updated for each $u_i \notin \mathrm{dom}(\sigma^{\mathbf{C}}_{\mathbf{e}})$ so that $\sigma^{\mathbf{C}\prime}(\sigma^{\mathbf{C}}_{\mathbf{n}}(u_i))$ is equal to $\alpha'\sigma^{\mathbf{a}}_{\mathbf{e}}(y_i)$ or $\sigma^{\mathbf{a}}_{\mathbf{n}}(y_i)$, depending on whether $y_i \in \mathrm{dom}(\sigma^{\mathbf{a}}_{\mathbf{e}})$ or not.

PAX′: At the PAX′ rule, we are given $V, (X \uplus Z)$ and α, and $\sigma^{\mathbf{a}}$ as $\alpha'\sigma^{\mathbf{a}}_{\mathbf{e}} \otimes \sigma^{\mathbf{a}}_{\mathbf{n}}$, and $\sigma^{\mathbf{C}}$ as $\sigma^{\mathbf{C}}_{\mathbf{e}} \otimes \sigma^{\mathbf{C}\prime}(\mathsf{id}_Z \otimes \sigma^{\mathbf{C}}_{\mathbf{n}})$. We must now solve the equation $\sigma^{\mathbf{a}} = \sigma^{\mathbf{C}}(\mathsf{id}_z \otimes \alpha\tau)$, i.e.

$$\alpha'\sigma^{\mathbf{a}}_{\mathbf{e}} \otimes \sigma^{\mathbf{a}}_{\mathbf{n}} = (\sigma^{\mathbf{C}}_{\mathbf{e}} \otimes \sigma^{\mathbf{C}\prime}(\mathsf{id}_Z \otimes \sigma^{\mathbf{C}}_{\mathbf{n}}))(\mathsf{id}_Z \otimes \alpha\tau) \tag{1}$$

for α', $\sigma^{\mathbf{C}\prime}$, Z and τ, where $X_{\mathbf{e}} \cap Z = \emptyset$ (recall $\sigma^{\mathbf{C}}_{\mathbf{e}} : X_{\mathbf{e}} \to$).

7 Nondeterminism

Given these term-based rules and the normal inference grammar, proven correct matching has been expressed in an operational, that is, implementable, form. However, there is still a fair amount of nondeterminism left, but fortunately we can clearly identify where it occurs:

Grammar selection: Which branches to select for $\mathscr{D}_{\mathbf{G}}$, $\mathscr{D}_{\mathbf{P}}$, $\mathscr{D}'_{\mathbf{G}}$ and $\mathscr{D}'_{\mathbf{P}}$.

Tensor grouping: How to group the tensor product in PAR$_{\equiv}$.

Children partitioning: How to partition molecules in MER.

Prime permutation: How to permute redex primes in PER.

Context-redex-parameter wiring: How to choose Z, α and τ in PAX.

Mapping closed links: How to find an appropriate decomposition of agent wiring in CLO such that closed agent links are matched correctly with closed redex links (i.e., determining $\sigma^{\mathbf{a}}$ and $Y_{\mathbf{R}}$).

When implementing matching, the challenge is to develop a heuristic that will handle typical cases well. In general, an agent-redex pair can lead to many different matches, so in our implementation we return for every inference rule a lazy list of possible matches.

To handle nondeterminism, we return possible matches as follows, bearing in mind that operationally speaking, rules applied below SWX are given agent and redex, while rules above SWX are given agent (, redex) and context:

Grammar selection: For $\mathscr{D}_{\mathbf{G}}$ and $\mathscr{D}_{\mathbf{P}}$, we concatenate the returned lazy lists returned from matching each branch in turn. However, if PAX succeeds, there is no reason to attempt a SWX match, as no new matches will result.

For $\mathscr{D}_{\mathbf{G}}{}'$ and $\mathscr{D}_{\mathbf{P}}{}'$, we try each branch in turn, returning the first branch that succeeds, as later branches will not find any new matches.

Tensor grouping: For given m and n in $\text{PAR}_{\equiv}^{\mathscr{E}}$, we compute all the ways of splitting $[0,\ldots,m-1]$ into n (possibly empty) subsequences, trying out matching for each split. Note that this need only be done for applications of $\text{PAR}_{\equiv}^{\mathscr{E}}$ below the SWX rule.

Children partitioning: For given m and n in MER, we compute all the ways of partitioning $\{0,\ldots,m-1\}$ into n (possibly empty) sets, trying out matching for each partitioning.

Prime permutation: For given n in $\text{PER}^{\mathscr{E}}$, we compute all n-permutations, trying out matching for each permutation. This is done for applications of $\text{PER}^{\mathscr{E}}$ below the SWX rule; above, similar permutations are computed in the MER rule.

Context-redex-parameter wiring: Given global agent wiring, we compute the ways of decomposing it into $\sigma(\mathsf{id}_Z \otimes \alpha\tau)$, returning a match for each decomposition.

Mapping closed links: We split agent wiring into named and closed links, and postpone the actual mapping of each closed link to redex or context links until some constraint, given by ION or PAX produces it.

Note that even after limiting nondeterminism in this way, we can still in general find several instances of the same match, reached by different inference trees, as we are computing abstract bigraph matches using concrete representations. For instance, matching redex $R = \mathsf{K1}$ in agent $a = \mathit{merge}(\mathsf{K1} \otimes \mathsf{K1})$ produces matches with context $C_1 = \mathit{merge}(\mathsf{id}_1 \otimes \mathsf{K1})$ and context $C_2 = \mathit{merge}(\mathsf{K1} \otimes \mathsf{id}_1)$.

8 Tool Implementation and Example Modelling

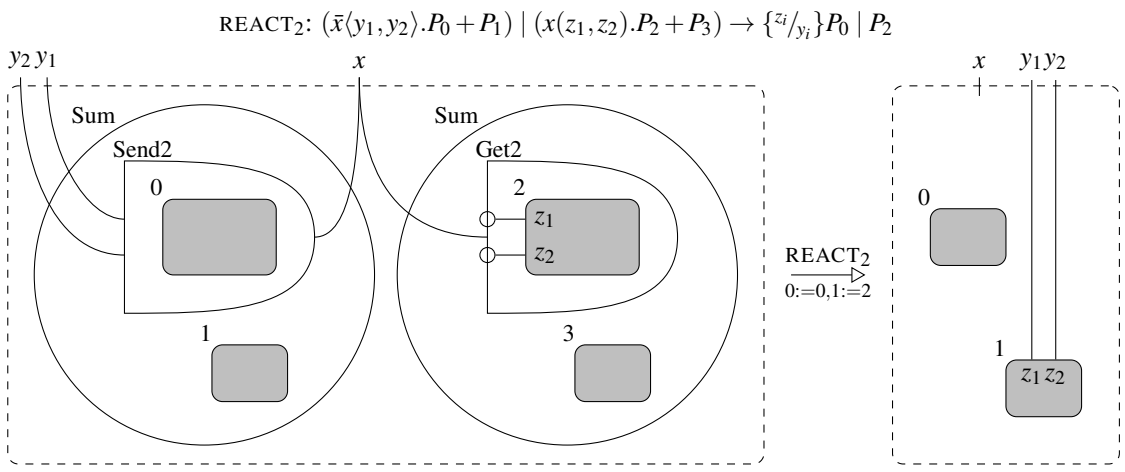
We have implemented a BPL Tool as a reference implementation of binding bigraph matching, and as a toolbox for experimenting with bigraphs. It is written in SML, consists of parser, normalisation and matching kernel, and includes web and command line user interfaces [4].

To ensure correctness, we have implemented normalisation, renaming, regularisation and matching faithfully by implementing one SML function for every inference rule—in the case of matching, two: one for applications above and one for below the SWX rule.

The BPL Tool handles normalisation, regularisation, matching and reaction for the full set of binding bigraphs, and allows construction of simple tactics for prescribing the order in which reaction rules should be applied. The following example output is taken verbatim from the command line interface, which is based on the SMLNJ interactive system; omitted details are indicated by "`[...]`".

As an example, we model the polyadic π calculus, running the mobile phone system introduced in Milner's π book [17]. The calculus can be modeled by a family of reaction rules $\{\text{REACT}_i \mid i = 0,1,\ldots\}$, one for each number of names that are to be communicated in a reaction [13]; REACT_2 is shown in Figure 14.

The signature for the nodes modelling the calculus and the mobile phone system is constructed using `passive` and `atomic` functions as shown in Figure 15. For this system, we only need `Send` and `Get`




```
val REACT2 = "REACT2" :::
  Sum o (Send2[x,y1,y2] `|` idp(1)) `|` Sum o (Get2[x][[z1],[z2]] `|` idp(1))
  --[0 |-> 0, 1 |-> 2]--|>
  (y1/z1 * y2/z2 * x//[] * idp(1)) o (idp(1) `|` `[z1, z2]`);
```

Figure 14: π calculus reaction rule shown as bigraphs and BPL value.

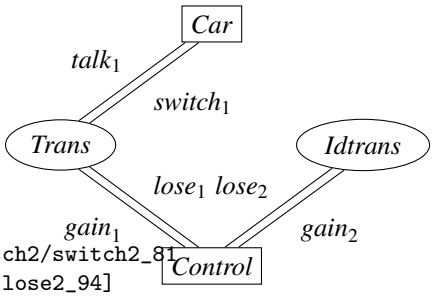
```
(* Pi calculus nodes *)
val Sum   = passive0 ("Sum")
val Send0 = passive ("Send0" -: 0 + 1)
val Get0  = passive ("Get0"  =: 0 --> 1)
val Send2 = passive ("Send2" -: 2 + 1)
val Get2  = passive ("Get2"  =: 2 --> 1)
```

```
(* Mobile phone system nodes *)
val Car      = atomic ("Car" -: 2)
val Trans    = atomic ("Trans" -: 4)
val Idtrans  = atomic ("Idtrans" -: 2)
val Control  = atomic ("Control" -: 8)
```

Figure 15: Signature for π calculus and mobile phone system nodes.

nodes for REACT_0 and REACT_2. Note that all reaction rule nodes are passive, preventing reaction within a guarded expression.

The system consists of a car, one active and one idle transmitter, and a control centre, as shown in Figure 16. Internally, a prime product constructed using the ‘|‘ operator is represented by a wiring and




```
- val System1 = simplify (
      Car[talk1,switch1]
  ‘|‘ Trans[talk1,switch1,gain1,lose1]
  ‘|‘ Idtrans[gain2,lose2]
  ‘|‘ Control[lose1,talk2,switch2,gain2,
              lose2,talk1,switch1,gain1]);
val System1 =
  (lose1//[lose1_83, lose1_98] * talk2/talk2_82 * switch2/switch2_81
    * gain2//[gain2_80, gain2_95] * lose2//[lose2_7f, lose2_94]
    * talk1//[talk1_7e, talk1_9b, talk1_a5]
    * switch1//[switch1_7d, switch1_9a, switch1_a4]
    * gain1//[gain1_7c, gain1_99]) o merge(4) o
  (Car[talk1_a5, switch1_a4] *
   Trans[talk1_9b, switch1_9a, gain1_99, lose1_98] *
   Idtrans[gain2_95, lose2_94] *
   Control[lose1_83, talk2_82, switch2_81, gain2_80, lose2_7f, talk1_7e,
           switch1_7d, gain1_7c])
  : 0 -> <{lose1, talk2, switch2, gain2, lose2, talk1, switch1, gain1}> : bgval
-
```

Figure 16: Definition of the mobile phone system, $System_1$

$merge_2$ composed with a binary tensor product. The function `simplify` applies various heuristics for producing human-readable bigraph terms, in this case for a prime product of four factors.

The definition of these nodes and connections, shown in Figure 17, allows the control centre to switch *Car* communication between the two transmitters (supposedly when the car gets closer to the ilde than the active transmitter), and allows the car to talk with the active transmitter. Note that in the BPL tool, we define a node by a rule that unfolds an atomic node into a bigraph corresponding to the defining π calculus expression.

Our BPL definition of the initial system in Figure 16, $System_1$, is the folded version; as BPL matching is complete, querying the tool reveals the four possible unfolding matches, illustrated in Figure 18. Here `mkrules` constructs the internal representation of a rule set, and `print_mv` prettyprints a lazy list of matches, produced by the `matches` function.

Using `react_rule` that simply applies a named reaction rule, and ++ that runs its arguments sequentially, we construct a tactic, `TAC_unfold`, for unfolding all four nodes once, shown in Figure 19. Applying this tactic using function `run`, we get an unfolded version of the system.

Querying the BPL Tool for all possible matches in the unfolded system reveals exactly the switch and talk actions, initiated by REACT_2 and REACT_0 rules, respectively, cf. Figure 20. Applying the π calculus reaction rules for switching, we arrive at $System_2$, where *Car* communication has been switched to the other transmitter, as witnessed by the outer names to which *Car* ports link, as well as the order of names to which *Control* ports link.

This concludes our description of the example highlighting how we can use the BPL Tool to experiment with bigraphical reactive systems.

Defining equation	*BPL definition*

$\underline{Car}(talk, switch) \stackrel{\text{def}}{=}$
$\overline{talk}.\,Car\langle talk, switch\rangle$
$\quad + switch(t,s).\,Car\langle t,s\rangle$
	`val DEF_Car = "DEF_Car" :::`
	`  Car[talk,switch]`
	`  ----|>`
	`  Sum o (Send0[talk] o Car[talk,switch]`
	`  '|' Get2[switch][[t],[s]]`
	`  o (<[t,s]> Car[t,s]))`

$Trans(talk, switch, gain, lose) \stackrel{\text{def}}{=}$
$talk.\,Trans\langle talk, switch, gain, lose\rangle$
$\quad + lose(t,s).\overline{switch}\langle t,s\rangle$
$\quad\quad .Idtrans\langle gain, lose\rangle$
	`val DEF_Trans = "DEF_Trans" :::`
	`  Trans[talk,switch,gain,lose]`
	`  ----|>`
	`  Sum o (Get0[talk][] o Trans[talk,switch,gain,lose]`
	`  '|' Get2[lose][[t],[s]]`
	`  o (<[t,s]> Sum o Send2[switch,t,s]`
	`  o Idtrans[gain,lose]))`

$Idtrans(gain, lose) \stackrel{\text{def}}{=}$
$gain(t,s).\,Trans\langle t,s,gain,lose\rangle$
	`val DEF_Idtrans = "DEF_Idtrans" :::`
	`  Idtrans[gain, lose]`
	`  ----|>`
	`  Sum o Get2[gain][[t],[s]]`
	`  o (<[t,s]> Trans[t,s,gain,lose])`

$Control(lose_1, talk_2, switch_2, gain_2,$
$\quad lose_2, talk_1, switch_1, gain_1) \stackrel{\text{def}}{=}$
$\overline{lose_1}\langle talk_2, switch_2\rangle.\overline{gain_2}\langle talk_2, switch_2\rangle$
$\quad .Control\langle lose_2, talk_1, switch_1, gain_1,$
$\quad\quad lose_1, talk_2, switch_2, gain_2\rangle$
	`val DEF_Control = "DEF_Control" :::`
	`  Control[lose1,talk2,switch2,gain2,`
	`  lose2,talk1,switch1,gain1]`
	`  ----|>`
	`  Sum o Send2[lose1,talk2,switch2]`
	`  o Sum o Send2[gain2,talk2,switch2]`
	`  o Control[lose2,talk1,switch1,gain1,`
	`  lose1,talk2,switch2,gain2]`

Figure 17: Definitions of Car, Trans, Idtrans and Control nodes.

```
- val rules = mkrules [REACT0, REACT2, DEF_Car, DEF_Trans, DEF_Idtrans, DEF_Control];
[...]
- print_mv (matches rules System1);
[{rule = "DEF_Car",
 context
  = (lose1//[lose1_d3, lose1_d6] * talk2/talk2_d2 * switch2/switch2_d1
      * gain2//[gain2_d0, gain2_d5] * lose2//[lose2_cf, lose2_d4]
      * talk1//[talk, talk1_ce, talk1_d9]
      * switch1//[switch, switch1_cd, switch1_d8]
      * gain1//[gain1_cc, gain1_d7]) o
    (merge(4) o
     (Trans[talk1_d9, switch1_d8, gain1_d7, lose1_d6] *
      Idtrans[gain2_d5, lose2_d4] *
      Control[lose1_d3, talk2_d2, switch2_d1, gain2_d0, lose2_cf,
              talk1_ce, switch1_cd, gain1_cc])),
 parameter = idx0},
{rule = "DEF_Control", [...] },
{rule = "DEF_Idtrans", [...] },
{rule = "DEF_Trans", [...] }]
```

Figure 18: Determining which rules match $System_1$.

```
- val TAC_unfold =
  react_rule "DEF_Car"     ++ react_rule "DEF_Trans"   ++
  react_rule "DEF_Idtrans" ++ react_rule "DEF_Control";
[...]
- val System1_unfolded = run rules TAC_unfold System1;
val System1_unfolded =
  (lose1//[lose1_3f9, lose1_419, lose_441, lose_459, lose_45d]
    * talk2//[talk2_3f8, talk2_40f, talk2_418]
    * switch2//[switch2_3f7, switch2_40e, switch2_417]
    * gain2//[gain2_3f6, gain2_410, gain_431, gain_438]
    * lose2//[lose2_3fd, lose_430]
    * talk1//[talk1_3fc, talk_460, talk_465, talk_482, talk_485]
    * switch1//[switch1_3fb, switch_447, switch_45f, switch_480, switch_481]
    * gain1//[gain1_3fa, gain_442, gain_45e]) o merge(4) o
  (Sum o merge(2) o
   (Send0[talk_485] o Car[talk_482, switch_481] *
    Get2[switch_480][[t_47d], [s_47c]] o
    (<[s_47c, t_47d]> Car[t_47d, s_47c])) *
   Sum o merge(2) o
   (Get0[talk_465] o Trans[talk_460, switch_45f, gain_45e, lose_45d] *
    Get2[lose_459][[t_446], [s_445]] o
    (<[s_445, t_446]>
      Sum o (Send2[switch_447, t_446, s_445] o Idtrans[gain_442, lose_441]))) *
   Sum o Get2[gain_438][[t_433], [s_432]] o
   (<[s_432, t_433]> Trans[t_433, s_432, gain_431, lose_430]) *
   Sum o
   (Send2[lose1_419, talk2_418, switch2_417] o
    (Sum o
     (Send2[gain2_410, talk2_40f, switch2_40e] o
      Control[lose2_3fd, talk1_3fc, switch1_3fb, gain1_3fa, lose1_3f9,
              talk2_3f8, switch2_3f7, gain2_3f6]))))
  : 0 -> <{lose1, talk2, switch2, gain2, lose2, talk1, switch1, gain1}> : agent
```

Figure 19: Unfolding $System_1$, using the TAC_unfold tactic.

```
- print_mv (matches rules System1_unfolded);
[{rule = "REACT0", [...] }, {rule = "REACT2", [...] }]
[...]
- val TAC_switch =
  react_rule "REACT2"      ++ (* Control tells Trans to lose.  *)
  react_rule "REACT2"      ++ (* Control tells Idtrans to gain. *)
  react_rule "REACT2";        (* Trans tells Car to switch.     *)
[...]
- val System2 = run rules TAC_switch System1_unfolded;
val System2 =
  (lose1//[lose1_86a, lose_8c0] * talk2//[t_858, talk2_869, t_8bf]
    * switch2//[s_857, switch2_868, s_8be] * gain2//[gain_856, gain2_867]
    * lose2//[lose_855, lose2_86e] * talk1/talk1_86d * switch1/switch1_86c
    * gain1//[gain1_86b, gain_8c1]) o merge(4) o
  (Idtrans[gain_8c1, lose_8c0] * Car[t_8bf, s_8be] *
   Control[lose2_86e, talk1_86d, switch1_86c, gain1_86b, lose1_86a, talk2_869,
           switch2_868, gain2_867] * Trans[t_858, s_857, gain_856, lose_855])
  : 0 -> <{lose1, talk2, switch2, gain2, lose2, talk1, switch1, gain1}> : agent
-
```

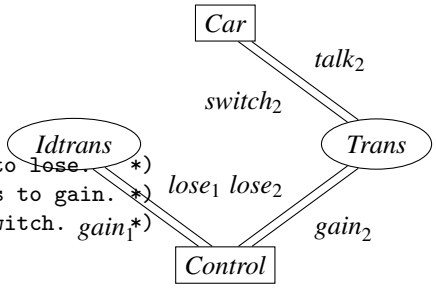


Figure 20: Checking possible matches, then switching to $System_2$, using the TAC_switch tactic.

9 Conclusion and Future Work

We have developed a provably sound and complete inference system over bigraph terms for inferring legal matches of bigraphical reactive systems. Moreover, we have implemented our BPL Tool, the first implementation of bigraphical reactive systems. We have demonstrated a simple, but concrete, example of how the tool can be used to simulate bigraphical models. We have found it very useful to base this first implementation of bigraphical reactive systems so closely on the developed theory—this has naturally given us greater confidence in the implementation, but the implementation work has also helped to debug the developed theory.

There are lots of interesting avenues for future work. While the current implementation of BPL Tool is efficient enough to experiment with small examples, we will try to make it more efficient by using a number of different techniques: we plan to investigate how to prune off invalid matches quickly, for instance by making use of sorting information [3]. Moreover, we will investigate to what extent we can capture the link graph matching via a constraint-based algorithm.

We also plan to investigate smarter ways of combining matching and rewriting. As a starting point, we have made it possible for users to combine *tactics* to inform the tool in which order it should attempt to apply reaction rules.

Jean Krivine and Robin Milner are currently investigating stochastic bigraphs, which will be particularly important for simulation of real systems. We hope that our detailed analysis of matching for binding bigraphs will make it reaonably straightforward to extend it to stochastic bigraphs.

Acknowledgements

The authors would like to thank Mikkel Bundgaard and anonymous referees for detailed comments to earlier versions of the paper.

References

[1] Lars Birkedal, Troels Christoffer 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.

[2] Lars Birkedal, Søren Debois, Ebbe Elsborg, Thomas Troels Hildebrandt, and Henning Niss. Bigraphical models of context-aware systems. In Luca Aceto and Anna Ingólfsdóttir, editors, *Proceedings of the 9th International Conference on Foundations of Software Science and Computation Structure*, volume 3921 of *Lecture Notes in Computer Science*, pages 187–201. Springer-Verlag, March 2006. ISBN 3-540-33045-3.

[3] Lars Birkedal, Søren Debois, and Thomas Troels Hildebrandt. Sortings for reactive systems. In Christel Baier and Holger Hermanns, editors, *Proceedings of the 17th International Conference on Concurrency Theory*, volume 4137 of *Lecture Notes in Computer Science*, pages 248–262. Springer-Verlag, August 2006.

[4] The BPL Group. BPLweb—the BPL tool web demo, 2007. URL `http://tiger.itu.dk:8080/bplweb/`. IT University of Copenhagen, Denmark. Prototype.

[5] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José Francisco Quesada. Maude: Specification and programming in rewriting logic. *Theoretical Computer Science*, 2001.

[6] 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.

[7] Troels Christoffer Damgaard. Syntactic theory for bigraphs. Master's thesis, IT University of Copenhagen, Rued Langgaards Vej 7, DK-2300 Copenhagen V, December 2006.

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

[9] Troels Christoffer Damgaard, Arne John Glenstrup, Lars Birkedal, and Robin Milner. An inductive characterization of matching in binding bigraphs. *to appear*, 2011.

[10] Hartmut Ehrig, Karsten Ehrig, Ulrike Prange, and Gabriele Taentzer. *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[11] James Jianghai Fu. Directed graph pattern matching and topological embedding. *Journal of Algorithms*, 22(2):372–391, 1997.

[12] Ole Høgh Jensen. *Mobile Processes in Bigraphs*. PhD thesis, Univ. of Cambridge, 2008. Forthcoming.

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

[14] Javier Larrosa and Gabriel Valiente. Constraint satisfaction algorithms for graph pattern matching. *Journal of Mathematical Structures in Computer Science*, 12:403–422, 2002.

[15] James Judi Leifer and Robin Milner. Transition systems, link graphs and Petri nets. Technical Report UCAM-CL-TR-598, University of Cambridge, August 2004.

[16] The Maude Team. The Maude system, 2007. `http://maude.cs.uiuc.edu/`.

[17] Robin Milner. *Communicating and Mobile Systems: The π-Calculus*. Cambridge University Press, 1999.

[18] Robin Milner. Bigraphs whose names have multiple locality. Technical Report UCAM-CL-TR-603, University of Cambridge, September 2004.

[19] Vladimiro Sassone and Paweł Sobociński. Reactive systems over cospans. In *Proceedings of Logic in Computer Science (LICS'05)*, pages 311–320. IEEE Press, 2005.

[20] Jeffrey D. Ullman. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, 1976.

[21] Albert Zündorf. Graph pattern matching in PROGRES. In Janice E. Cuny, Hartmut Ehrig, Gregor Engels, and Grzegorz Rozenberg, editors, *TAGT*, volume 1073 of *Lecture Notes in Computer Science*, pages 454–468. Springer-Verlag, 1994. ISBN 3-540-61228-9.

A Auxiliary Technologies Details

A.1 Normalising

We define a normalisation relation $t \downarrow_{\mathbf{B}} t'$ for elementary bigraphs: $merge_n, \ulcorner X \urcorner, \vec{y}/\vec{X}, K_{\vec{y}(\vec{X})}$ and π as shown in Figure 21, and inductively for operations: abstraction $(X)P$, product $\bigotimes_i^n B_i$ and composition $B_1 B_2$ as shown in Figure 22, where the notation $\sigma\lfloor^Y$ means $\{X \mapsto y \in \sigma \mid y \in Y\}$.

$$\text{Bmer}\ \frac{N \equiv (\emptyset)(\mathsf{id}_\emptyset \otimes merge_n)\bigotimes_{i\in n} \ulcorner \mathsf{id}_\emptyset \urcorner \qquad P \equiv (\mathsf{id}_\emptyset \otimes {}^{([])}/_{([])})N \qquad D \equiv \mathsf{id}_\emptyset \otimes (\bigotimes_{i\in 1} P)\mathsf{id}_n}{merge_n \downarrow_{\mathbf{B}} (\mathsf{id}_\emptyset \otimes \mathsf{id}_{[\emptyset]})D}$$

$$\text{Bcon}\ \frac{\begin{array}{c} N \equiv (\emptyset)(\mathsf{id}_X \otimes merge_1)\bigotimes_{i\in 1}(\mathsf{id}_X \otimes \mathsf{id}_1)\ulcorner X \urcorner \\ P \equiv (\mathsf{id}_X \otimes {}^{([])}/_{([])})N \qquad D \equiv \mathsf{id}_\emptyset \otimes (\bigotimes_{i\in 1} P)\mathsf{id}_{(X)} \end{array}}{\ulcorner X \urcorner \downarrow_{\mathbf{B}} (\mathsf{id}_X \otimes \mathsf{id}_{[\emptyset]})D} \qquad \text{Bwir}\ \frac{}{\vec{y}/\vec{X} \downarrow_{\mathbf{B}} (\vec{y}/\vec{X} \otimes \mathsf{id}_{[]})(\mathsf{id}_X \otimes \mathsf{id}_0 \mathsf{id}_0)}$$

$$\text{Bion}\ \frac{\begin{array}{c} X = \{\vec{X}\} \qquad Y = \{\vec{y}\} \qquad M \equiv (\mathsf{id}_\emptyset \otimes K_{\vec{y}(\vec{X})})(X)(\mathsf{id}_X \otimes merge_1)\bigotimes_{i\in 1}(\mathsf{id}_X \otimes \mathsf{id}_1)\ulcorner X \urcorner \\ N \equiv (\emptyset)(\mathsf{id}_Y \otimes merge_1)\bigotimes_{i\in 1} M \qquad P \equiv (\mathsf{id}_Y \otimes {}^{([])}/_{([])})N \qquad D \equiv \mathsf{id}_\emptyset \otimes (\bigotimes_{i\in 1} P)\mathsf{id}_{(X)} \end{array}}{K_{\vec{y}(\vec{X})} \downarrow_{\mathbf{B}} (\mathsf{id}_Y \otimes \mathsf{id}_{[\emptyset]})D}$$

$$\text{Bper}\ \frac{\begin{array}{c} Y_i = \{\vec{y}_i\} \qquad N_i \equiv (Y_i)(\mathsf{id}_{Y_i} \otimes merge_1)\bigotimes_{j\in 1}(\mathsf{id}_{Y_i} \otimes \mathsf{id}_1)\ulcorner Y_i \urcorner \\ P_i \equiv (\mathsf{id}_\emptyset \otimes \widehat{\vec{y}_i/\vec{y}_i})N_i \qquad\qquad D \equiv \mathsf{id}_\emptyset \otimes (\bigotimes_{i\in m} P_i)\pi \end{array}}{\pi : \langle m, \vec{X}, X\rangle \to \langle m, \vec{Y}, X\rangle \downarrow_{\mathbf{B}} (\mathsf{id}_\emptyset \otimes \mathsf{id}_{\vec{Y}})D}$$

Figure 21: Inference rules for normalising elementary bigraph expressions

$$\text{Babs}\ \frac{\begin{array}{c} b \downarrow_{\mathbf{B}} (z/W \otimes \mathsf{id}_{([Y])})(\mathsf{id}_\emptyset \otimes (\bigotimes_{i\in 1}(\mathsf{id}_Z \otimes \widehat{\vec{y}/\vec{X}})(W)G)\mathsf{id}_I) \\ \vec{z}^X = [z_j \leftarrow \vec{z} \mid z_j \in X] \qquad \vec{z}^{\bar{X}} = [z_j \leftarrow \vec{z} \mid z_j \notin X] \\ \vec{W}^X = [W_j \leftarrow \vec{W} \mid z_j \in X] \qquad \vec{W}^{\bar{X}} = [W_j \leftarrow \vec{W} \mid z_j \notin X] \\ W^X = \{\vec{W}^X\} \quad W^{\bar{X}} = \{\vec{W}^{\bar{X}}\} \quad U = \{\vec{y}\vec{z}^X\} \quad N \equiv (W^X \cup W)G \quad P \equiv (\mathsf{id}_{W^{\bar{X}}} \otimes \widehat{\vec{y}\vec{z}^X/\vec{X}\vec{W}^X})N \end{array}}{(X)b \downarrow_{\mathbf{B}} (z^{\bar{X}}/W^{\bar{X}} \otimes \mathsf{id}_{([U])})(\mathsf{id}_\emptyset \otimes (\bigotimes_{i\in 1} P)\mathsf{id}_I)}$$

$$\text{Bten}\ \frac{\begin{array}{c} b_i \downarrow_{\mathbf{B}} (\omega_i \otimes \mathsf{id}_{(\vec{Y}_i)})D_i \qquad D_i \equiv \alpha_i \otimes (\bigotimes_{j\in n_i} P_i^j)\pi_i : I_i \to \langle n_i, \vec{Y}_i, Y_i\rangle \\ \omega = \bigotimes_{i\in n}\omega_i \quad \alpha = \bigotimes_{i\in n}\alpha_i \quad \mathsf{id}_{(\vec{Y})} = \bigotimes_{i\in n}\mathsf{id}_{(\vec{Y}_i)} \quad \pi = \bigotimes_{i\in n}\pi_i \\ P = \bigotimes_{j\in n}\bigotimes_{i\in n_j} P_i^j \qquad D \equiv \alpha \otimes P\pi \end{array}}{\bigotimes_{i\in n} b_i \downarrow_{\mathbf{B}} (\omega \otimes \mathsf{id}_{(\vec{Y})})D}$$

$$\text{Ccom}\ \frac{\sigma = (\mathsf{id}_Z \otimes \alpha)(\mathsf{id}_Z \otimes y/X)}{(\mathsf{id}_Z \otimes (\alpha \otimes \mathsf{id}_1)^{\ulcorner} Y^{\urcorner}) \bigotimes_{i\in 1}(\mathsf{id}_Z \otimes \widehat{\vec{y}/\vec{X}})(X)(\mathsf{id}_U \otimes merge_n)\bar{S} \downarrow_{\bar{\mathbf{S}}} \sigma, \bar{S}}$$

$$\text{Mcom}\ \frac{\begin{array}{c} (\mathsf{id}_Z \otimes N)\bar{P} \downarrow_{\mathbf{N}} \sigma, N' \quad \vec{X}' = \sigma^{-1}(\vec{X}) \quad Z' = \sigma^{-1}(Z) \quad Y' = \sigma^{-1}(Y) \quad \sigma' = \mathsf{id}_{\{\vec{y}\}} \otimes \sigma\lfloor^{Z\uplus Y} \end{array}}{(\mathsf{id}_Z \otimes (\mathsf{id}_Y \otimes K_{\vec{y}(\vec{X})})N)\bar{P} \downarrow_{\bar{\mathbf{S}}} \sigma', \bigotimes_{i\in 1}(\mathsf{id}_{Z'\uplus Y'} \otimes K_{\vec{y}(\vec{X}')})N'}$$

$$\text{Ncom}\ \frac{\begin{array}{c} P_i : \langle m_i, \vec{X}_i, X_i\rangle \to \langle 1, (Y_i), Y_i \uplus W_i\rangle \\ \bigotimes_{i\in n}\bar{P}_i = \bigotimes_{i\in k} P_i \qquad \bar{P}_i : I_i \to \langle n_i, \vec{Y}_i, \{\vec{Y}_i\} \uplus Z_i\rangle \qquad (\mathsf{id}_{Z_i} \otimes S_i)\bar{P}_i \downarrow_{\bar{\mathbf{S}}} \sigma_i, \bar{S}_i \\ \bar{S} = \bigotimes_{i\in n}\bar{S}_i : I \to \langle n', Z' \uplus Y'\rangle \quad \sigma = \bigotimes_{i\in n}\sigma_i \quad X' = \sigma^{-1}(X) \quad Z' = \sigma^{-1}(Z) \quad Y' = \sigma^{-1}(Y) \end{array}}{(\mathsf{id}_Z \otimes (X)(\mathsf{id}_Y \otimes merge_n)\bigotimes_{i\in n} S_i)\bigotimes_{i\in k} P_i \downarrow_{\mathbf{N}} \sigma, (X')(\mathsf{id}_{Z'\uplus Y'} \otimes merge_{n'})\bar{S}}$$

$$\text{Pcom}\ \frac{(\mathsf{id}_Z \otimes N)\bar{P} \downarrow_{\mathbf{N}} \sigma, N' \quad W = \sigma^{-1}(Z \uplus Z') \quad \vec{X}' = \sigma^{-1}(\vec{X}) \quad \sigma' = \sigma\lfloor^{Z\uplus Z'}}{(\mathsf{id}_Z \otimes (\mathsf{id}_{Z'} \otimes \widehat{\vec{y}/\vec{X}})N)\bar{P} \downarrow_{\mathbf{P}} \sigma', (\mathsf{id}_W \otimes \widehat{\vec{y}/\vec{X}'})N'}$$

$$\text{Bcom}\ \frac{\begin{array}{c} b_1 \downarrow_{\mathbf{B}} \Big(\omega_1 \otimes \mathsf{id}_{(\vec{U}^1)}\Big)D_1 : \langle m', \vec{X}', X' \uplus Z\rangle \to \langle n, \vec{U}^1, U^1 \uplus W\rangle \\ b_2 \downarrow_{\mathbf{B}} \Big(\omega_2 \otimes \mathsf{id}_{(\vec{U}^2)}\Big)D_2 : \langle m, \vec{X}, X \uplus U\rangle \to \langle m', \vec{U}^2, U^2 \uplus Z\rangle \\ D_1 \equiv \alpha_1 \otimes \Big(\bigotimes_{i\in n} P_i^1\Big)\pi_1 : \langle m', \vec{X}', X' \uplus Z\rangle \to \langle n, \vec{U}^1, U^1 \uplus V^1 \uplus W^1\rangle \\ D_2 \equiv \alpha_2 \otimes \Big(\bigotimes_{i\in m'} P_i^2\Big)\pi_2 : \langle m, \vec{X}, X \uplus U\rangle \to \langle m', \vec{U}^2, U^2 \uplus V^2 \uplus W^2\rangle \\ P_i^1 : \langle m_i', \vec{X}_i', X_i'\rangle \to \langle (U_i^1), U_i^1 \uplus V_i^1\rangle \qquad P_i^2 : \langle m_i'', \vec{X}_i'', X_i''\rangle \to \langle (U_i^2), U_i^2 \uplus V_i^2\rangle \\ \omega_1 : V^1 \uplus W^1 \to W \qquad \omega_2 : V^2 \uplus W^2 \to Z \qquad \alpha_1 : Z \to W^1 \qquad \alpha_2 : U \to W^2 \\ V^2 = \uplus_{i\in m'} V_i^2 \qquad \bigotimes_{i\in m'} P^2_{\pi_1^{-1}(i)} = \bigotimes_{i\in n}\bar{P}_i \qquad \bar{P}_i : I_i' \to \langle m_i', \vec{X}_i', X_i' \uplus Z_i'\rangle \\ (\mathsf{id}_{Z_i'} \otimes P_i^1)\bar{P}_i \downarrow_{\mathbf{P}} \sigma_i, P_i \quad \sigma = \mathsf{id}_U \otimes \bigotimes_{i\in n}\sigma_i \quad \omega = \omega_1(\alpha_1\omega_2(\alpha_2 \otimes \mathsf{id}_{V^2}) \otimes \mathsf{id}_{V^1})\sigma \\ \pi = \overline{\pi_1}_{\vec{X}''}\pi_2 \qquad D \equiv \mathsf{id}_U \otimes \big(\bigotimes_{i\in n} P_i\big)\pi \end{array}}{b_1 b_2 \downarrow_{\mathbf{B}} \Big(\omega \otimes \mathsf{id}_{(\vec{U}^1)}\Big)D}$$

Figure 22: Inference rules for normalising bigraph abstraction, product and composition expressions

A.2 Renaming

Let a *link namer* be a map μ mapping every link l (outer name or edge) in its domain to a pair (E,X), where E is a set of names used internally to compose the link, and X are the inner names linking to l. We let $\mathscr{V}_i(Y,\mu) = \bigcup_{y\in Y, y\mapsto(X_1,X_2)\in\mu} X_i$ and define link namer composition by

$$\begin{array}{rcl} \mu_1 \circ \mu_2 & = & \{y_1 \mapsto (E_1 \cup X_1 \cup V_1, V_2) \mid y_1 \mapsto (E_1,X_1) \in \mu_1 \wedge V_i = \mathscr{V}_i(X_1,\mu_2)\} \\ & \cup & \{y_2 \mapsto (E_2,X_2) \in \mu_2 \mid \forall y_1 \mapsto (E_1,X_1) \in \mu_1 : y_2 \notin X_1\}, \end{array}$$

essentially composing links of μ_1 with those of μ_2, and adding closed links from μ_2.

We then define a function *linknames*, mapping terms to link namers, by the equations given in Figure 23. By using the link namers of immediate subterms, we can determine whether a term can be

$$\begin{array}{lcl} \mathit{linknames}(\mathit{merge}_n) & = & \{\} \\ \mathit{linknames}(\ulcorner X \urcorner) & = & \{x \mapsto (\{\},\{x\}) \mid x \in X\} \\ \mathit{linknames}(\vec{y}/\vec{X}) & = & \{y_i \mapsto (\{\},X_i) \mid i \in |\vec{y}|\} \\ \mathit{linknames}(K_{\vec{y}(\vec{e}/\vec{X})}) & = & \{y_i \mapsto (\{\},\{\}) \mid i \in |\vec{y}|\} \cup \{e_i \mapsto (\{\},X_i) \mid i \in |\vec{X}|\} \\ \mathit{linknames}(\pi : \to \langle m,\vec{X},X\rangle) & = & \{x \mapsto (\{\},\{x\}) \mid x \in X\} \\ \mathit{linknames}((Y)P) & = & \mathit{linknames}(P) \\ \mathit{linknames}(\bigotimes_i t_i) & = & \bigcup_i \mathit{linknames}(t_i) \\ \mathit{linknames}(t_1 t_2) & = & \mathit{linknames}(t_1) \circ \mathit{linknames}(t_2) \end{array}$$

Figure 23: Function for determining which names are used internally to compose a link

normalised without name clashes. To this end, we define a predicate *normalisable* by the equations given in Figure 24. We basically just require, that at no level in the term does two different links share any internal names.

$$\begin{array}{lcl} \mathit{normalisable}(\mathit{merge}_n) & = & \mathit{true} \\ \mathit{normalisable}(\ulcorner X \urcorner) & = & \mathit{true} \\ \mathit{normalisable}(\vec{y}/\vec{X}) & = & \mathit{true} \\ \mathit{normalisable}(K_{\vec{y}(\vec{e}/\vec{X})}) & = & \mathit{true} \\ \mathit{normalisable}(\pi : \to \langle m,\vec{X},X\rangle) & = & \mathit{true} \\ \mathit{normalisable}((Y)P) & = & \mathit{normalisable}(P) \\ \mathit{normalisable}(\bigotimes_i t_i) & = & \bigwedge_i \mathit{normalisable}(t_i) \\ & & \wedge(\forall i \neq j : E_i \cap E_j = \emptyset) \\ & & \text{where } \mu_i = \mathit{linknames}(t_i) \\ & & \quad\; E_i = \bigcup_{y\mapsto(E,X)\in\mu_i} E \\ \mathit{normalisable}(t_1 t_2) & = & \mathit{normalisable}(t_1) \wedge \mathit{normalisable}(t_2) \\ & & \wedge(\forall l_1 \neq l_2 : \mu_{\mathbf{E}}(l_1) \cap \mu_{\mathbf{E}}(l_2) = \emptyset) \\ & & \text{where } \mu_i = \mathit{linknames}(t_i) \\ & & \quad\; \mu = \mu_1 \circ \mu_2 \\ & & \quad\; \mu_{\mathbf{E}}(l) = E, \text{ if } \mu(l) = (E,X) \end{array}$$

Figure 24: Function for determining whether a (well-formed) term is normalisable

Renaming is achieved by the judgment $U \vdash \alpha, t \downarrow_\beta t', \beta \dashv V$, where U is a set of used names and α renames t's inner names to those of t', while β renames t's outer names to those of t' and V extends U with names used in t'. The system of rules for inferring this judgment is given in Figure 25.

$$\text{Rmer}\ \frac{}{U \vdash \mathsf{id}_\emptyset, merge_n \downarrow_\beta merge_n, \mathsf{id}_\emptyset \dashv U} \qquad \text{Rcon}\ \frac{X' = \alpha(X)}{U \vdash \alpha, \ulcorner X \urcorner \downarrow_\beta \ulcorner X' \urcorner, \alpha \dashv U}$$

$$\text{Rwir}\ \frac{\begin{array}{c} Z = \{\vec{z}\} \quad Z \cap U = \emptyset \quad |Z| = |\vec{z}| = |\vec{y}| \\ \vec{X}' = \alpha(\vec{X}) \qquad \beta = \{y_i \mapsto z_i\} \end{array}}{U \vdash \alpha, \vec{y}/\vec{X} \downarrow_\beta \vec{z}/\vec{X}', \beta \dashv U \cup Z} \qquad \text{Rion}\ \frac{\begin{array}{c} Z = \{\vec{z}\} \quad Z \cap U = \emptyset \quad |Z| = |\vec{z}| = |\vec{y}| \\ \vec{X}' = \alpha(\vec{X}) \qquad \beta = \{y_i \mapsto z_i\} \end{array}}{U \vdash \alpha, K_{\vec{y}(\vec{X})} \downarrow_\beta K_{\vec{z}(\vec{X}')}, \beta \dashv U \cup Z}$$

$$\text{Rper}\ \frac{X' = \alpha(X) \quad \vec{X}' = \alpha(\vec{X}) \quad \vec{Y}' = \alpha(\vec{Y})}{U \vdash \alpha, \pi : \langle m, \vec{X}, X \rangle \to \langle m, \vec{Y}, X \rangle \downarrow_\beta \pi : \langle m, \vec{X}', X' \rangle \to \langle m, \vec{Y}', X' \rangle, \alpha \dashv U}$$

$$\text{Rabs}\ \frac{U \vdash \alpha, t \downarrow_\beta t', \beta \dashv V \quad X' = \beta(X)}{U \vdash \alpha, (X)t \downarrow_\beta (X')t', \beta \dashv V}$$

$$\text{Rten}\ \frac{t_i : \langle m_i, \vec{X}_i, X_i \rangle \to J_i \quad \alpha_i = \alpha \restriction_{X_i} \quad U_i \vdash \alpha_i, t_i \downarrow_\beta t_i', \beta_i \dashv U_{i+1} \quad \beta = \bigotimes_{i \in n} \beta_i}{U_0 \vdash \alpha, \bigotimes_{i \in n} t_i \downarrow_\beta \bigotimes_{i \in n} t_i', \beta \dashv U_n}$$

$$\text{Rcom}\ \frac{U_1 \vdash \alpha_1, t_2 \downarrow_\beta t_2', \beta_1 \dashv U_2 \quad U_2 \vdash \beta_1, t_1 \downarrow_\beta t_1', \beta_2 \dashv V_2}{U_1 \vdash \alpha_1, t_1 t_2 \downarrow_\beta t_1' t_2', \beta_2 \dashv V_2}$$

Figure 25: Renaming rules

A.3 Regularising

The system of rules for inferring a permutation-free term representing a regular bigraph is given in Figure 26.

$$\alpha\ \frac{}{\ulcorner \alpha \urcorner \mathsf{id}_{(X)} \hookrightarrow \ulcorner \alpha \urcorner} \qquad \text{M}\ \frac{N\pi \hookrightarrow N'}{(\mathsf{id}_Z \otimes K_{\vec{y}(\vec{X})})N\pi \hookrightarrow (\mathsf{id}_Z \otimes K_{\vec{y}(\vec{X})})N'}$$

$$\text{N}\ \frac{S_i : \langle m_i, \vec{X}_i \rangle \to J_i \quad \pi = \underline{\pi'}^{\vec{X}} \quad S_i \pi_i'^{\vec{X}} \hookrightarrow S_i'}{((X)(\mathsf{id}_Y \otimes merge_n) \bigotimes_{i \in n} S_i)\pi' \hookrightarrow (X)(\mathsf{id}_Y \otimes merge_n) \bigotimes_{i \in n} S'_{\pi(i)}}$$

$$\text{D}\ \frac{\pi = \bigotimes_{i \in n} \pi_i \quad \pi_i : I_i' \to I_i \quad N_i : I_i \to J_i \quad N_i \pi_i \hookrightarrow N_i'}{\alpha \otimes (\bigotimes_{i \in n} (\mathsf{id}_{Z_i} \otimes \widehat{\vec{y}_i/\vec{X}_i}) N_i)\pi \hookrightarrow \alpha \otimes \bigotimes_{i \in n} (\mathsf{id}_{Z_i} \otimes \widehat{\vec{y}_i/\vec{X}_i}) N_i'} \qquad \text{B}\ \frac{D \hookrightarrow D'}{(\omega \otimes \mathsf{id}_{(\vec{X})})D \hookrightarrow (\omega \otimes \mathsf{id}_{(\vec{X})})D'}$$

Figure 26: Removing nontrivial permutations from regular bigraphs.