Electronic Communications of the EASST
Volume 29 (2010)

Proceedings of the
Ninth International Workshop on
Graph Transformation and
Visual Modeling Techniques
(GT-VMT 2010)

Graph Algebras for Bigraphs

Davide Grohmann, Marino Miculan

17 pages

Guest Editors: Jochen Küster, Emilio Tuosto
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Graph Algebras for Bigraphs[*]

Davide Grohmann[1], **Marino Miculan**[2]

[1] grohmann@dimi.uniud.it, [2] miculan@dimi.uniud.it
Department of Mathematics and Computer Science, University of Udine, Italy

Abstract: Binding bigraphs are a graphical formalism intended to be a meta-model for mobile, concurrent and communicating systems. In this paper we present an algebra of *typed graph terms* which correspond precisely to binding bigraphs over a given signature. As particular cases, pure bigraphs and local bigraphs are described by two sublanguages which can be given a simple syntactic characterization.

Moreover, we give a formal connection between these languages and *Synchronized Hyperedge Replacement* algebras and the hierarchical graphs used in *Architectural Design Rewriting*. This allows to transfer results and constructions among formalisms which have been developed independently, e.g., the systematic definition of congruent bisimulations for SHR graphs via the IPO construction.

Keywords: Bigraphs, graph grammars, types.

1 Introduction

Bigraphical Reactive Systems (BRSs) [Mil01] have been proposed as a promising meta-model for ubiquitous, mobile systems. The key feature of BRSs is that their states are *bigraphs*, semi-structured data which can represent at once both the (physical, logical) location and the connections of the components of a system. The dynamics of the system is given by a set of rewrite rules on this semi-structured data.

Bigraphs have been successfully used for representing many domain-specific calculi and models, from traditional programming languages, to process calculi for concurrency and mobility, from context-aware systems to web-service orchestration languages—see e.g. [JM03, LM06, BDE$^+$06, GM07, BGH$^+$08]. In fact, many variants of bigraphs have been proposed: the original *pure* bigraphs have been later generalized into *binding bigraphs*, allowing also for name scoping; other variants have been proposed, such as *local bigraphs* used for studying the λ-calculus.

In this paper, we propose a *general typed language* for binding bigraphs, which we recall in Section 2. More precisely, in Section 3 we define an algebra of typed graph terms, so that well-typed terms correspond to binding bigraphs, and congruence captures bigraphic equality; this interpretation and corresponding properties are exposed in Section 4. Moreover, as we will show in Section 5, the important subcategories of pure, local and prime bigraphs can be described by suitable sublanguages which can be given a simple and effective syntactic characterization.

Finally, we show how this language can be tailored to formalisms introduced in literature (for quite different purposes). In Section 6 we consider hypergraphs used in *Synchronized Hyperedge Rewriting* [FHL$^+$05] and the "designs" of *Architectural Design Rewriting* [BLMT07].

[*] Work funded by MIUR PRIN project "SisteR", prot. 20088HXMYN.
Dedicated to the memory of Robin Milner (13 January 1934 – 20 March 2010).

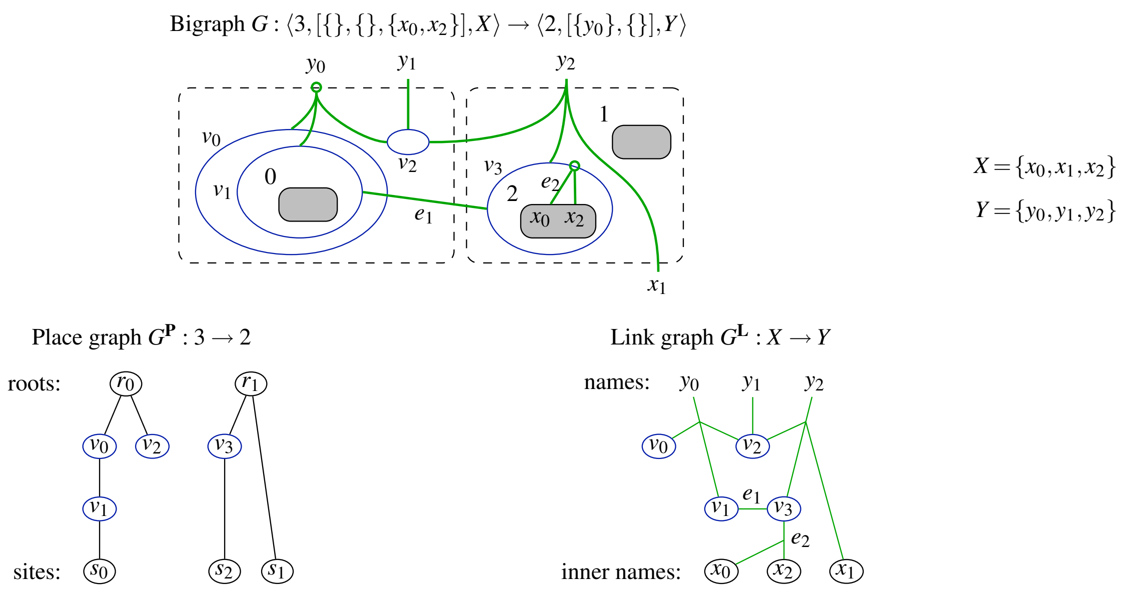

Bigraph $G : \langle 3, [\{\}, \{\}, \{x_0, x_2\}], X\rangle \to \langle 2, [\{y_0\}, \{\}], Y\rangle$
y_0 y_1 y_2
v_0 v_1 v_2 v_3
0 1 2
e_1 e_2
x_0 x_2 x_1
$X = \{x_0, x_1, x_2\}$
$Y = \{y_0, y_1, y_2\}$
Place graph $G^{\mathbf{P}} : 3 \to 2$
roots: r_0 r_1
v_0 v_2 v_3 v_1
sites: s_0 s_2 s_1
Link graph $G^{\mathbf{L}} : X \to Y$
names: y_0 y_1 y_2
v_0 v_2 v_1 e_1 v_3 e_2
inner names: x_0 x_2 x_1

Figure 1: A binding bigraph (picture taken from [BDGM07]).

These results are useful for several reasons. First, the typed algebra we propose can be used as a language for binding, pure, local and prime bigraphs, alternative to the bigraph algebra [JM03]. Moreover, we confirm once more that bigraphs are a quite expressive general framework of ubiquitous systems. These connections pave the way for transferring results and constructions from bigraphs to the SHR and ADR frameworks, and vice versa, as suggested in Section 7.

2 Binding Bigraphs

In this section we recall Milner's *binding bigraphs* [JM03, JM04]. Intuitively, a binding bigraph represents an open system, so it has an inner and an outer interface to "interact" with subsystems and the surrounding environment (Figure 1). The *width* of the outer interface describes the *roots*, that is, the various locations containing the system components; the width of the inner interface describes the *sites*, that is, the holes where other bigraphs can be inserted. On the other hand, the *names* in the interfaces describe free links, that is end points where links from the outside world can be pasted, creating new links among nodes. In particular, we consider binding bigraphs with (possibly) multiply localized names, as in [Mil04] and slightly generalizing [JM03, JM04].

More formally, let $\mathcal{K}$ be a *binding signature* of controls (i.e., node types), and $ar\colon \mathcal{K} \to \mathbb{N} \times \mathbb{N}$ be the arity function. The arity pair (h,k) (written $h \to k$) consists of the *binding arity* h and the *free arity* k, indexing respectively the binding and the free ports of a control.

Definition 1 (Interfaces) An *interface* is a pair $\langle m, X\rangle$ where m is a finite ordinal (called *width*), X is a finite set of names. A *binding interface* is a triple $\langle m, loc, X\rangle$, where $\langle m, X\rangle$ is an interface and $loc \subseteq m \times X$ is a *locality* map associating a subset of the names in X with sites in m. If $(s,x) \in loc$ then x is *located* at s, or *local* to s; x is *global* if, $\forall s$, $(s,x) \notin loc$.

Sometime, we shall represent the locality map as a vector $\vec{X} = (X_0, \ldots, X_{m-1})$ of subsets, where X_s is the set of names local to s; thus $X \setminus \vec{X} = X \setminus (X_0 \cup \cdots \cup X_{m-1})$ are the global names. We call an interface *local* (resp. *global*) if all its names are local (resp. global). We denote by $\uplus$ the union of already disjoint sets, i.e., $S \uplus T \triangleq S \cup T$ if $S \cap T = \emptyset$, otherwise it is undefined.

Definition 2 (Pure and binding bigraphs) A *(pure) bigraph* $G\colon \langle m, X\rangle \to \langle n, Y\rangle$ is composed by a *place graph* G^P and a *link graph* G^L describing node nesting and (hyper-)links among nodes:

$$\begin{aligned} G &= (V, E, ctrl, G^P, G^L)\colon \langle m, X\rangle \to \langle n, Y\rangle && \text{(pure bigraph)}\\ G^P &= (V, ctrl, prnt)\colon m \to n && \text{(place graph)}\\ G^L &= (V, E, ctrl, link)\colon X \to Y && \text{(link graph)} \end{aligned}$$

where V, E are the sets of nodes and edges respectively; $ctrl\colon V \to \mathscr{K}$ is the *control map,* which assigns a control to each node; $prnt\colon m \uplus V \to V \uplus n$ is the (acyclic) *parent map* (often written also as $<$); $link\colon X \uplus P \to E \uplus B \uplus Y$ is the *link map,* where $P = \sum_{v\in V} \pi_1(ar(ctrl(v)))$ is the set of ports and $B = \sum_{v\in V} \pi_2(ar(ctrl(v)))$ is the set of bindings (associated to all nodes). A link $l \in X \uplus P$ is *bound* if $link(l) \in B$; it is *free* if $link(l) \in Y \uplus E$.

A *binding bigraph* $G\colon \langle m, loc, X\rangle \to \langle n, loc', Y\rangle$ is a (pure) bigraph $G^u\colon \langle m, X\rangle \to \langle n, Y\rangle$ satisfying the following locality conditions:

1. if a link is bound, then the names and ports linked to it must lie within the node that binds it;
2. if a link is free, with outer name x, then x must be located in every region that contains any inner name or port of the link.

Definition 3 (Binding bigraph category) The category of binding bigraphs over a signature $\mathscr{K}$ (written $\mathbf{Bbg}(\mathscr{K})$) has local interfaces as objects, and binding bigraphs as morphisms.

Given two binding bigraphs $G\colon \langle m, loc, X\rangle \to \langle n, loc', Y\rangle$, $H\colon \langle n, loc', Y\rangle \to \langle k, loc'', Z\rangle$, the composition $H \circ G\colon \langle m, loc, X\rangle \to \langle k, loc'', Z\rangle$ is defined by composing their place and link graphs, whenever they have disjoint node and edge sets:

1. the composition of $G^P\colon m \to n$ and $H^P\colon n \to k$ is defined as
$$H^P \circ G^P = (V_G \uplus V_H, ctrl_G \uplus ctrl_H, (id_{V_G} \uplus prnt_H) \circ (prnt_G \uplus id_{V_H}))\colon n \to k;$$
2. the composition of $G^L\colon X \to Y$ and $H^L\colon Y \to Z$ is defined as
$$H^L \circ G^L = (V_G \uplus V_H, E_G \uplus E_H, ctrl_G \uplus ctrl_H, (id_{E_G} \uplus link_H) \circ (link_G \uplus id_{P_H}))\colon X \to Z.$$

Definition 4 (Pure, local and prime bigraphs) The category of *pure bigraphs* (**Big**) is the full subcategory of **Bbg** whose objects are of the form $\langle n, (\emptyset), X\rangle$ (often shorten as $\langle n, X\rangle$).

The category of *local bigraphs* (**Lbg**) is the full subcategory of binding bigraphs whose objects are of the form $\langle n, (\vec{X}), \bigcup \vec{X}\rangle$ (often shorten as $(\vec{X})$).

The category of *prime bigraphs* (**Pbg**) is the full subcategory of local bigraphs whose objects are of the form $\langle n, (\vec{X}), \bigcup \vec{X}\rangle$, with $n \in \{0, 1\}$, (often shorten as before: $(\vec{X})$).

Intuitively, in pure bigraphs all names are global, whilst in local bigraphs all names are local, finally prime bigraphs are all the local bigraphs with one root, and one or zero holes.

An important operation about (bi)graphs, is the *tensor product*. Intuitively, the tensor product puts "side by side" two bigraphs, given $G\colon \langle m,(\vec{X}),X\rangle \to \langle n,(\vec{Y}),Y\rangle$ and $H\colon \langle m',(\vec{X}'),X'\rangle \to \langle n',(\vec{Y}'),Y'\rangle$, their tensor product is a bigraph $G\otimes H\colon \langle m+m',(\vec{X}\vec{X}'),X\cup X'\rangle \to \langle n+n',(\vec{Y}\vec{Y}'),Y\cup Y'\rangle$ defined when global names in X,X' and Y,Y' are disjoint. Two useful variants of tensor product can be defined using tensor and composition: the *parallel product* $\|$, which merges shared names between two bigraphs, and the *prime product* $|$, that also merges all roots in a single one. As shown in [Mil04, DB06], all bigraphs can be constructed by composition and tensor product from a set of *elementary bigraphs:*

- $1 : \langle 0,(\emptyset),\emptyset\rangle \to \langle 1,(\emptyset),\emptyset\rangle$ is the barren (i.e., empty) root;
- $merge_n : \langle n,(\emptyset),\emptyset\rangle \to \langle 1,(\emptyset),\emptyset\rangle$ merges n roots into a single one;
- $\gamma_{m,n,(\vec{X},\vec{Y})} : \langle m+n,(\vec{X}\vec{Y}),(\bigcup\vec{X})\cup(\bigcup\vec{Y})\rangle \to \langle m+n,(\vec{Y}\vec{X}),(\bigcup\vec{X})\cup(\bigcup\vec{Y})\rangle$ permutes the first m roots having local names in $\vec{X}$ with the following n roots with local names in $\vec{Y}$.
- $/x : \langle 0,(\emptyset),\{x\}\rangle \to \langle 0,(\emptyset),\emptyset\rangle$ is a *closure*, that is it maps x to an edge;
- $y/X : \langle 0,(\emptyset),X\rangle \to \langle 0,(\emptyset),\{y\}\rangle$ substitutes the names in X with y, i.e., it maps the whole set X to y; as a shortcuts, we write $\vec{y}/\vec{X}$ to mean $y_0/X_0\otimes\ldots\otimes y_{n-1}/X_{n-1}$;
- $\ulcorner X\urcorner : \langle 1,(X),X\rangle \to \langle 1,(\emptyset),X\rangle$ means that names in X are switched from local to global
- conversely, $(X) : \langle 1,(\emptyset),X\rangle \to \langle 1,(X),X\rangle$ localizes the global names of X.
- Finally, $K_{\vec{x}(\vec{X})} : \langle 1,(X),\emptyset\rangle \to \langle 1,(\emptyset),\vec{x}\rangle$ is a control which may contain other graphs, and it has free ports linked to the name in $\vec{x}$, whilst the names $\vec{X}$ are connected to its binding ports.

We use the convention that local names are enclosed in parenthesises.

Bigraphs can be given always in *discrete normal form*: the idea of this normal form is to separate wirings (i.e., linkings) from discrete bigraphs (i.e., nesting of nodes). The following is an easy generalization of [DB06, Theorem 1] to the case of bigraphs with multiply located names.

Theorem 1 (Discrete Normal Form (DNF)) *1. Any binding bigraph $G : I \to \langle n,\vec{Y}_B,(\bigcup\vec{Y}_B)\uplus Y_F\rangle$ can be expressed as*

$$G = \Big(\bigotimes_{i<n}(\vec{y}_i)/(\vec{X}_i)\otimes\bigotimes_{i<|Y_F|} w_i/W_i\otimes\bigotimes_{i<|Z|}(/z_i\circ z_i/Z_i)\Big)\circ D$$

where $D : I \to \langle m,\vec{X},(\bigcup\vec{X})\uplus W\uplus Z\rangle$ is a name discrete.

2. *Any name discrete $D : I \to \langle m,\vec{X},(\bigcup\vec{X})\uplus W\uplus Z\rangle$ can be expressed as*

$$D = \alpha\otimes((P_0\otimes\ldots\otimes P_{n-1})\circ\pi)$$

where α is a renaming, and π a permutation.

3. *Any name-discrete prime $P : J \to \langle 1,(U_B),U\rangle$ can be expressed as*

$$(U_B)\circ(merge_{n+k}\otimes id_U)\circ(\ulcorner\alpha_0\urcorner\otimes\ldots\otimes\ulcorner\alpha_{n-1}\urcorner\otimes M_0\otimes\ldots\otimes M_{k-1})\circ\pi$$

where every $M_i : J_i \to \langle 1,(\emptyset),U_i^M\rangle$ is a free discrete molecule, and for renamings $\alpha_i : V_i \to U_i^C$, we have $U = (\biguplus_{i\in n} U_i^C)\uplus\biguplus_{i\in k} U_i^M$.

4. *Any free discrete molecule $M : K \to \langle 1,(\emptyset),\vec{x}\uplus V\rangle$ can be expressed as $M = (K_{\vec{x}(\vec{S})}\otimes id_V)\circ P$ where $P : H \to \langle 1,(\biguplus\vec{S}),(\biguplus\vec{S})\uplus V$ is a name discrete.*

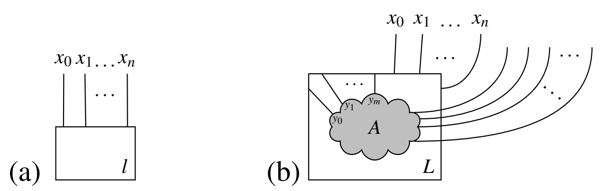
(a)  (b)

Figure 2: Example of an atomic label (a) and a non-atomic one (b).

3 Graph Grammar for Bigraphs

In this section we introduce a language for binding bigraphs. It is parametric over a ranked alphabet of labels $\mathscr{L} = (\mathscr{L}_a, \mathscr{L}_n, exit : \mathscr{L}_a \cup \mathscr{L}_n \to \mathbb{N}, in : \mathscr{L}_n \to \mathbb{N})$, where $\mathscr{L}_a$ are the *atomic* labels, ranged over by l, and $\mathscr{L}_n$ are the *non-atomic* labels, ranged over by L. Each label is given an *exit-rank*, $exit(l) \in \mathbb{N}$, enumerating the "exiting tentacles". Non-atomic labels have an *in-rank* $in(L) \in \mathbb{N}$, enumerating the label's "incoming tentacles". We often denote by $\mathscr{L}$ the set $\mathscr{L}_a \cup \mathscr{L}_n$.

One may think of a node with an atomic label l as an hyperedge with $exit(l)$ tentacles, as in Figure 2(a). A node labelled with L has $exit(L)$ tentacles, and may contain a subsystem whose exiting tentacles are either linked to the $in(L)$ ports of the node, or go "outside" the node, see Figure 2(b). More formally, the language of graphs is as follows.

Definition 5 (Agent graphs) Let $\mathscr{N}$ be an infinite set of names, $\mathscr{V}$ be an infinite set of variables, and $\mathscr{L}$ be a ranked alphabet of labels. An *agent-graph* A is a term generated as follows:

$$A ::= \varepsilon \mid \mathbf{0} \mid l(\vec{x}) \mid L(\vec{x})[A\backslash\vec{y}] \mid X \mid A|A \mid A \parallel A \mid \nu z.A \mid A[w \mapsto z] \mid A \downarrow z \mid A \uparrow z$$

where $\vec{x}, \vec{y} \subseteq \mathscr{N}^*$; $l \in \mathscr{L}_a$, $L \in \mathscr{L}_n$ with $exit(l) = exit(L) = |\vec{x}|$, $in(L) = |\vec{y}|$; $X \in \mathscr{V}$; and $w, z \in \mathscr{N}$. Moreover, in a term A, each X is used at most once.

Intuitively, ε represents the absence of any system, that is, no agents at all, while $\mathbf{0}$ represents an empty agent (i.e., an agent with no nodes). We denote by $l(\vec{x})$ atomic hyperedges whose tentacles are linked to the names in $\vec{x}$, whilst by $L(\vec{x})[A\backslash\vec{y}]$ non-atomic hyperedges having exiting tentacles linked to the names in $\vec{x}$, containing a subgraph A whose names $\vec{y}$ are linked on the edge itself. Graph variables X are needed for representing open systems, i.e., graphs with holes.

Two agent graphs A, B can be composed in parallel in two different ways: $A \mid B$ "merges" two graphs into a unique one (i.e., in the same location), while $A \parallel B$ puts the two systems side by side, i.e., they keep living in different locations.

As usual, $\nu y.A$ limits the scope of y to A, while $A[w \mapsto z]$ is the explicit substitution of name w with z. Notice that the agent-graph $A[w \mapsto z]$ exhibits the name z also when w does not appear in A; in this case, the operator $\_[w \mapsto z]$ "creates" unused (or idle) names.

Finally, $A \downarrow z$ *localizes* z to (the location of) A. This means that z can only be accessed by/linked to nodes lying in the location of A, that is, they must be inside or in parallel ($|$) to A. Dually, $A \uparrow z$ *globalizes* z, allowing a localized name to be used also by nodes in different locations.

From the above intuition, it is clear that not all terms generated by this grammar are meaningful. For instance, what is the meaning of $A|B$ or $A \downarrow z$ when A is a system with more than one location? In order to rule out meaningless terms, we introduce a typing system for agent graphs.

$$\frac{}{\langle\rangle\,;\tau\vdash\varepsilon:(\langle\rangle,\tau)}\qquad\frac{}{\langle\rangle\,;\tau\vdash\mathbf{0}:(\emptyset,\tau)}\qquad\frac{}{\langle\rangle\,;\tau\vdash l(\vec{x}):(\emptyset,\vec{x}\cup\tau)}\qquad\frac{\Gamma;\tau\vdash A:(\sigma\cup\vec{y},\rho)\quad\vec{y}\cap\sigma=\emptyset}{\Gamma;\tau\vdash L(\vec{x})[A\backslash\vec{y}]:(\sigma,\rho\cup\vec{x})}$$

$$\frac{\Gamma;\tau\vdash A:(\sigma,\rho)\ \Gamma';\tau'\vdash A':(\sigma',\rho')\ \tau\cap\tau'=\emptyset}{\Gamma,\Gamma';\tau\cup\tau'\vdash A\mid A':(\sigma\cup\sigma',\rho\cup\rho')}\qquad\frac{\Gamma;\tau\vdash A:(\vec{\sigma},\rho)\ \Gamma';\tau'\vdash A':(\vec{\sigma}',\rho')\ \tau\cap\tau'=\emptyset}{\Gamma,\Gamma';\tau\cup\tau'\vdash A\parallel A':(\vec{\sigma}\vec{\sigma}',\rho\cup\rho')}$$

$$\frac{}{X:\sigma;\tau\vdash X:(\sigma,\tau)}\qquad\frac{\Gamma;\tau\vdash A:(\vec{\sigma},\rho)}{\Gamma;\tau\vdash\nu x.A:(\vec{\sigma}\setminus\{x\},\rho\setminus\{x\})}\qquad\frac{\Gamma;\tau\vdash A:(\vec{\sigma},\rho)\quad\pi\text{ permutation}}{\pi(\Gamma);\tau\vdash A:(\vec{\sigma},\rho)}$$

$$\frac{\Gamma;\tau\vdash A:(\vec{\sigma},\rho)\quad x\in\vec{\sigma}}{\Gamma;\tau\vdash A[x\mapsto y]:(\vec{\sigma}\{y/x\},\rho)}\qquad\frac{\Gamma;\tau\vdash A:(\vec{\sigma},\rho)\quad x\notin\vec{\sigma}}{\Gamma;\tau\vdash A[x\mapsto y]:(\vec{\sigma},(\rho\setminus\{x\})\cup\{y\})}$$

$$\frac{\Gamma;\tau\vdash A:(\sigma,\rho\cup\{x\})\quad x\notin\rho}{\Gamma;\tau\vdash A\downarrow x:(\sigma\cup\{x\},\rho)}\qquad\frac{\Gamma;\tau\vdash A:(\vec{\sigma},\rho)\quad x\in\vec{\sigma}}{\Gamma;\tau\vdash A\uparrow x:(\vec{\sigma}\setminus\{x\},\rho\cup\{x\})}$$

Figure 3: Type inference rules for agent-graphs.

Definition 6 (Type system for agent graphs) *Simple types* τ,σ,ρ are finite sets of names.

An *agent-type* $(\vec{\tau},\tau)$ is a pair formed by a list $\vec{\tau}=\tau_0\ldots\tau_{n-1}$ of simple types (where $\langle\rangle$ is the empty list), and a simple-type τ, such that $\tau\cap(\tau_0\cup\ldots\cup\tau_{n-1})=\emptyset$.

An *environment* is a pair $\Gamma;\tau$ formed by a list of typed variables ($\Gamma=\vec{X}:\vec{\tau}=X_0:\tau_0,\ldots,X_{n-1}:\tau_{n-1}$) and a simple-type ($\tau$), such that $(\vec{\tau},\tau)$ is an agent-type.

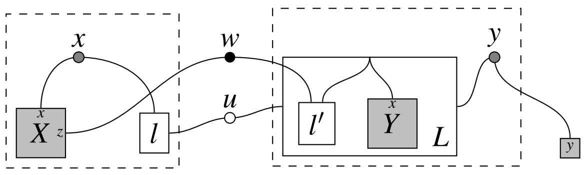
A typing judgement is of the form $\Gamma;\tau\vdash A:(\vec{\sigma},\rho)$, whose inference rules are in Figure 3.

Agent-types $(\vec{\tau},\tau)=(\tau_0\ldots\tau_{n-1},\tau)$ describe both the locations of a graph, and the names that the graph exposes to the environment. Names in τ are "global", and can be used from every location; instead, names in τ_i can be used only inside the i-th location of the system.

We are interested in open systems, that is systems with "holes". An environment $\Gamma;\tau=X_0:\tau_0,\ldots,X_{n-1}:\tau_{n-1}\,;\tau$ declares the "inner interface" of an agent: the names of the variables (X_i for $i\in n$), with their sets of local names (τ_i are the names local to X_i), and the set of incoming global names (τ), i.e., names that can be used from within any variable.

Notation. List concatenation is denoted simply by juxtaposition. We extend the operators $\in$, $\cup$ and $\setminus$ over set lists as follows: $x\in\vec{\tau}$ iff there exists $\bar{\tau}\in\vec{\tau}$ such that $x\in\bar{\tau}$; let S be a set, then $\tau_0\ldots\tau_{n-1}\cup S\triangleq(\tau_0\cup S)\ldots(\tau_{n-1}\cup S)$ and $\tau_0\ldots\tau_{n-1}\setminus S\triangleq(\tau_0\setminus S)\ldots(\tau_{n-1}\setminus S)$. Γ_1,Γ_2 is the concatenation of Γ_1 and Γ_2, defined when $\mathrm{dom}(\Gamma_1)\cap\mathrm{dom}(\Gamma_2)=\emptyset$. We introduce some useful shortcuts: $\nu\vec{x}.A=\nu x_{|\vec{x}-1|}.\ldots\nu x_0.A$; $A[\vec{x}\mapsto\vec{y}]=A[x_0\mapsto y_0]\ldots[x_{n-1}\mapsto y_{n-1}]$ when $|\vec{x}|=|\vec{y}|=n$; $A[X\mapsto x]=A[x_0\mapsto x]\ldots[x_{|X|}\mapsto x]$ if $X\neq\emptyset$, $A[\emptyset\mapsto x]=A[z\mapsto x]$ for some z fresh (i.e., z is not used by A); finally $A[\vec{X}\mapsto\vec{x}]=A[X_0\mapsto x_0]\ldots[X_{n-1}\mapsto x_{n-1}]$ when $|\vec{X}|=|\vec{x}|=n$.

Some intuitive explanation of the typing rules may be useful. Empty agents have only global names, as defined by the environment. Notice that $\mathbf{0}$ is the null process, which *is* an agent, while ε is no agent at all. An atomic hyperedge whose exit-tentacles are linked to the names $\vec{x}$ exposes those (global) names to a context, plus the ones added by the environment. As before, a non-atomic hyperedge shows names $\vec{x}$ that are linked to its exit-tentacles, plus the global ones defined in the environment. The difference is that it contains a graph term having $\vec{y}$ local names, that are linked to the hyperedge's input tentacles, and hence they are not visible from the context.



$$Y:\{x\},X:\{x,z\};\{y\}\vdash \nu u.((l(x,u)\mid X)\downarrow z[z\mapsto w]\parallel L(y,u)[(Y\mid l'(w,x))\backslash x]\downarrow w):(\{w\}\{w\},\{x,y\})$$

Figure 4: An example of an agent-graph.

The names exposed by a composition ($|$) of two subgraphs are the union of the names exposed by the two subgraphs. The rule for $\parallel$ is quite similar, but in this case the two graphs keep their different locations, and hence the names can be treated in a different way, so global names are the union of agents' global names, whilst local names remains unchanged, i.e, the two lists of local names are concatenated. If a variable has type σ in an environment Γ, then it exposes σ local names and the global names τ defined by the environment. The restriction deletes a name from the set of global or local exposed names. The next rule describes the possibility to reorder the variables in the environment; it will be important in the definition of a category for agent-graphs.

For the substitution $A[w\mapsto z]$ there are two cases: (1) if w is localized, it will be substituted by z; (2) if it is global the substitution (possibly) deletes w and adds z to the set of global names. Notice that if w,z are not used in A, then it effectively adds the name z to its interface.

An example of an agent-graph is given in Figure 4, where white nodes are closed (that is, nodes not accessible from the context); the other are the external nodes (which can be visible by a context): the grey nodes are global and the black ones are local.

Now, we can prove the following properties on our language.

Proposition 1 *If* $\Gamma;\tau\vdash A:(\vec{\sigma},\tau)$ *and* $\Gamma;\tau\vdash A:(\vec{\sigma}',\tau')$ *then* $\vec{\sigma}=\vec{\sigma}'$ *and* $\tau=\tau'$.

Lemma 1 (substitution lemma) *The following rule is admissible.*

$$\frac{\Gamma_i;\tau_i\vdash A_i:(\sigma_i,\rho_i)\quad 0\le i<n\quad \forall i\neq j.\,\tau_i\cap\tau_j=\emptyset\quad X_0:\sigma_0,\dots,X_{n-1}:\sigma_{n-1};\bigcup_{i=0}^{n-1}\rho_i\vdash A:(\vec{\eta},\zeta)}{\Gamma_0,\dots,\Gamma_{n-1};\bigcup_{i=0}^{n-1}\tau_i\vdash A\{A_0/X_0,\dots,A_{n-1}/X_{n-1}\}:(\vec{\eta};\zeta)}$$

As happens often with graph grammars, the same system may be denoted by many terms. Therefore, it is convenient to introduce a *structural congruence* over terms, capturing graph isomorphisms up-to free nodes. Congruence judgments are of the form $\Gamma;\tau\vdash A\equiv B$, for A,B terms of the language. This turns our language into a *graph algebra*, whose axioms are in Appendix A.

Proposition 2 *Let* $\Gamma;\tau\vdash A\equiv A'$, $\Gamma;\tau\vdash A:(\vec{\sigma},\rho)$ *if and only if* $\Gamma;\tau\vdash A':(\vec{\sigma},\rho)$.

4 Interpreting Agent Graphs as Binding Bigraphs

In this section we give an interpretation of agent graphs as binding bigraphs, showing that the language is sound and complete, and that the axiomatization captures bigraphical equivalence. In order to simplify the interpretations, we first introduce a category for agent-graphs.

Definition 7 The category $\mathbf{A}(\mathscr{L})$ of agent-graphs, over a ranked alphabet $\mathscr{L}$, has graph types $(\vec{\sigma},\rho)$ as objects, and judgments on agent-graphs as morphisms, that is, if $X_0 : \eta_0,\dots,X_{n-1} : \eta_{n-1}\,;\tau \vdash A : (\vec{\sigma},\rho)$ then $(X_0,\dots,X_{n-1},A) : (\vec{\eta},\tau) \to (\vec{\sigma},\rho)$ is a morphism. Composition is defined in virtue of Lemma 1.

Proposition 3 $(\mathbf{A}(\mathscr{L}), \|, \langle\rangle\,;\emptyset \vdash \varepsilon : (\langle\rangle,\emptyset))$ *is a strict symmetric monoidal category.*

4.1 Interpretation of agent-graphs as binding bigraphs

Let $\mathscr{L}$ be a ranked alphabet of labels; we define a functor from the agent-graph category $\mathbf{A}(\mathscr{L})$ to the binding bigraph category $\mathbf{BBg}(\mathscr{K}_{\mathscr{L}})$, for a suitable bigraphical signature $\mathscr{K}_{\mathscr{L}}$. The idea is to map agent-graph hyperedges into nodes, and nodes (or names) into links (i.e., outer names and edges); hence, the bigraphical signature corresponds to the alphabet of labels. Formally:

$$\mathscr{K}_{\mathscr{L}} \triangleq \{l : 0 \to exit(l) \mid l \in \mathscr{L}_a\} \cup \{L : in(L) \to exit(L) \mid L \in \mathscr{L}_n\}\,.$$

We can now define the functor $[\![-]\!] : \mathbf{A}(\mathscr{L}) \to \mathbf{BBg}(\mathscr{K}_{\mathscr{L}})$ by induction on the typing judgments:

Objects:
$$[\![(\vec{\sigma},\tau)]\!] = \langle|\vec{\sigma}|,\vec{\sigma},\tau\rangle$$

Morphisms:
$$\begin{aligned}
[\![\langle\rangle\,;\tau \vdash \varepsilon : (\langle\rangle,\tau)]\!] &= id_\tau\\
[\![\langle\rangle\,;\tau \vdash \mathbf{0} : (\emptyset,\tau)]\!] &= 1 \parallel id_\tau\\
[\![\langle\rangle\,;\tau \vdash l(\vec{x}) : (\emptyset,\vec{x}\cup\tau)]\!] &= l_{\vec{x}} \parallel id_\tau\\
[\![\Gamma;\tau \vdash L(\vec{x})[A\backslash\vec{y}] : (\sigma,\rho\cup\vec{x})]\!] &= L_{\vec{x},(\vec{y})} \circ [\![\Gamma;\tau \vdash A : (\sigma\cup\vec{y},\rho)]\!]\\
[\![X : \sigma;\tau \vdash X : (\sigma,\tau)]\!] &= id_{(\sigma)} \parallel id_\tau\\
[\![\Gamma,\Gamma';\tau\cup\tau' \vdash A \mid A' : (\sigma\cup\sigma',\rho\cup\rho')]\!] &= [\![\Gamma;\tau \vdash A : (\sigma,\rho)]\!] \mid [\![\Gamma';\tau' \vdash A' : (\sigma',\rho')]\!]\\
[\![\Gamma,\Gamma';\tau\cup\tau' \vdash A \parallel A' : (\vec{\sigma}\vec{\sigma}',\rho\cup\rho')]\!] &= [\![\Gamma;\tau \vdash A : (\vec{\sigma},\rho)]\!] \parallel [\![\Gamma';\tau' \vdash A' : (\vec{\sigma}',\rho')]\!]\\
[\![\Gamma;\tau \vdash \nu x.A : (\vec{\sigma}\setminus\{x\},\rho)]\!] &= /(x) \circ [\![\Gamma;\tau \vdash A : (\vec{\sigma},\rho)]\!] \qquad (\text{if } x \in \vec{\sigma})\\
[\![\Gamma;\tau \vdash \nu x.A : (\vec{\sigma},\rho\setminus\{x\})]\!] &= /x \circ ([\![\Gamma;\tau \vdash A : (\vec{\sigma},\rho)]\!] \parallel \{x\}) \qquad (\text{if } x \notin \vec{\sigma})\\
[\![\Gamma;\tau \vdash A[x \mapsto y] : (\vec{\sigma}\{y/x\},\rho)]\!] &= (y)/(x) \circ [\![\Gamma;\tau \vdash A : (\vec{\sigma},\rho)]\!] \qquad (\text{if } x \in \vec{\sigma})\\
[\![\Gamma;\tau \vdash A[x \mapsto y] : (\vec{\sigma},(\rho\setminus\{x\})\cup\{y\})]\!] &= y/x \circ ([\![\Gamma;\tau \vdash A : (\vec{\sigma},\rho)]\!] \parallel \{x\}) \qquad (\text{if } x \notin \vec{\sigma})\\
[\![\Gamma;\tau \vdash A \downarrow x : (\sigma\cup\{x\},\rho)]\!] &= (x) \circ [\![\Gamma;\tau \vdash A : (\sigma,\rho\cup\{x\})]\!]\\
[\![\Gamma;\tau \vdash A \uparrow x : (\vec{\sigma}\setminus\{x\},\rho\cup\{x\})]\!] &= \ulcorner x \urcorner \circ [\![\Gamma;\tau \vdash A : (\vec{\sigma},\rho)]\!] \qquad (\text{if } x \in \vec{\sigma})\\
[\![\pi(\Gamma);\tau \vdash A : (\vec{\sigma},\rho)]\!] &= [\![\Gamma;\tau \vdash A : (\vec{\sigma},\rho)]\!] \circ \pi\,.
\end{aligned}$$

Basically, each variable of type σ is encoded as a site having σ local names; therefore, variable permutation is site permutation. Restricted names are represented by bigraph edges, not accessible from the context. The graph $\mathbf{0}$ is represented by the empty root 1. An example is in Figure 5.

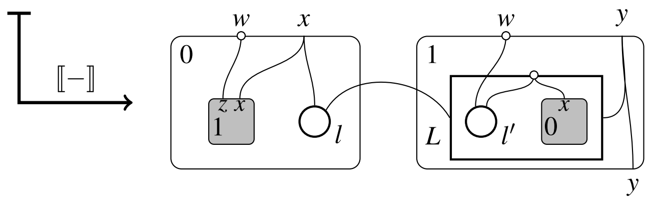
Now we can prove that $[\![-]\!]$ respects the structure of the two categories:

Proposition 4 $[\![-]\!] : (\mathbf{A}(\mathscr{L}), \|, \langle\rangle\,;\emptyset \vdash \varepsilon : (\langle\rangle,\emptyset)) \to (\mathbf{Bbg}(\mathscr{K}), \|, id_{(0,\emptyset,\emptyset)})$ *is strict monoidal.*

Moreover, the axiomatization of the graph algebra given in Appendix A is sound and complete with respect to bigraph equivalence.

Proposition 5 *Let A,A' be two agent-graphs; then, for every environment $\Gamma;\tau$: $\Gamma;\tau \vdash A \equiv A'$ if and only if $[\![\Gamma;\tau \vdash A : (\vec{\sigma},\rho)]\!] = [\![\Gamma;\tau \vdash A' : (\vec{\sigma},\rho)]\!]$.*

$$Y:\{x\},X:\{x,z\};\{y\}\vdash \nu u.((l(x,u)\mid X)\downarrow z[z\mapsto w]\parallel L(y,u)[(Y\mid l'(w,x))\backslash x]\downarrow w):(\{w\}\{w\},\{x,y\})$$



$$G:\langle 2,(\{z,x\},\{x\}),\{z,x,y\}\rangle\to\langle 2,(\{w\},\{w\}),\{x,y,w\}\rangle$$

Figure 5: An example of encoding an agent-graph into a binding bigraph.

4.2 Representing binding bigraphs with agent-graphs

In this section we show that our language is expressive enough to cover all binding bigraphs, over a given signature $\mathscr{K}$. To this end, we define a translation from binding bigraphs to agent-graphs of a language whose ranked labels are defined by means of the bigraphical signature.

$$\mathscr{L}_{\mathscr{K}}\triangleq(\{k\mid k:0\to n\in\mathscr{K},atomic\},\{k\mid k:m\to n\in\mathscr{K},non\ atomic\},exit,in)$$

$$exit(k)\triangleq n\quad\text{for } k:m\to n\in\mathscr{K}\qquad\qquad in(k)\triangleq m\quad\text{for } k:m\to n\in\mathscr{K}\ non\ atomic$$

The representation function $(\!|-|\!)$ maps objects of the category $\mathbf{Bbg}(\mathscr{K})$ to agent-types, as $(\!|\langle n,(\vec{X}),(\bigcup\vec{X})\uplus X\rangle|\!)\triangleq(\vec{X},X)$. In order to simplify the translation of bigraphs, in virtue of Theorem 1 we can suppose w.l.o.g. that all binding bigraphs are in discrete normal form. Let $G:\langle m,\vec{X}_B,(\bigcup\vec{X}_B)\uplus X_F\rangle\to\langle n,\vec{Y}_B,(\bigcup\vec{Y}_B)\uplus Y_F\rangle$, be in discrete normal form as follows

$$G=\Big(\bigotimes_{i<n}(\vec{y}_i)/(\vec{X}_i)\otimes\bigotimes_{i<|Y_F|}w_i/W_i\otimes\bigotimes_{i<|Z|}(/z_i\circ z_i/Z_i)\Big)\circ(\vec{a}/\vec{b}\otimes((P_0\otimes\ldots\otimes P_{n-1})\circ\pi))$$

then, for $\vec{Q}=Q_0,\ldots,Q_{m-1}$ a list of m variables, we define

$$\begin{aligned}(\!|G|\!)_{\vec{Q}}=\nu z_{|Z|-1}.\ldots.\nu z_0.(((\!|p_0|\!)_{\vec{Q}}\parallel\ldots\parallel(\!|p_{n-1}|\!)_{\vec{Q}})\\ [\vec{b}\mapsto\vec{a}][W_0\mapsto w_0]\ldots[W_{|Y_F|-1}\mapsto w_{|Y_F|-1}][\vec{X}_0\mapsto\vec{y}_0]\ldots[\vec{X}_{n-1}\mapsto\vec{y}_{n-1}])\end{aligned}$$

where $p_0\otimes\ldots\otimes p_{n-1}=(P_0\otimes\ldots\otimes P_{n-1})\circ\pi\circ(v^0_{(X_0)}\otimes\ldots\otimes v^{m-1}_{(X_{m-1})})$.

Given $p=(U_B)\circ(merge_{h+k}\otimes id_U)\circ(\ulcorner\vec{a}_0/\vec{b}_0\urcorner\otimes\ldots\otimes\ulcorner\vec{a}_{h-1}/\vec{b}_{h-1}\urcorner\otimes m_0\otimes\ldots\otimes m_{k-1})$, then

$$\begin{aligned}(\!|p|\!)_{\vec{Q}}=((\!|K^0_{\vec{x}^0,(\vec{S}^0)}\circ p^0|\!)_{\vec{Q}}\mid\ldots\mid(\!|K^{k_i-1}_{\vec{x}^{k_i-1},(\vec{S}^{k_i-1})}\circ p^{k_i-1}|\!)_{\vec{Q}}\mid\\ (\!|v^{j_0}_{(X_{j_0})}|\!)_{\vec{Q}}[\vec{b}_0\mapsto\vec{a}_0]\uparrow\vec{a}_0\mid\ldots\mid(\!|v^{j_{h_i-1}}_{(X_{j_{h_i-1}})}|\!)_{\vec{Q}}[\vec{b}_{h_{i-1}}\mapsto\vec{a}_{h_{i-1}}]\uparrow\vec{a}_{h_{i-1}})\downarrow U_B\end{aligned}$$

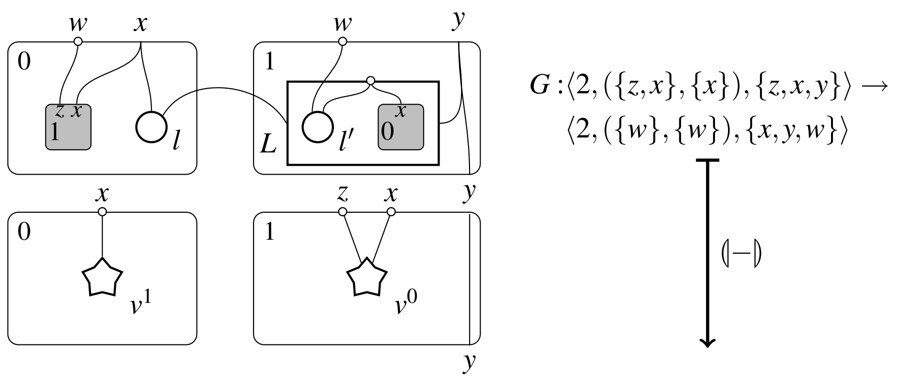
$$(\!|v^i_{(X_i)}|\!)_{\vec{Q}}=Q_i$$

$$(\!|K_{\vec{x},(\emptyset)}\circ 1|\!)_{\vec{Q}}=K(\vec{x})\qquad\text{where}\quad K\text{ atomic}$$

$$(\!|K_{\vec{x},(\vec{S})}\circ p|\!)_{\vec{Q}}=K(\vec{x})[(\!|p|\!)_{\vec{Q}}[\vec{S}\mapsto\vec{s}]\backslash\vec{s}]\qquad\text{where}\quad K\text{ non-atomic, and }\vec{s}\text{ fresh}$$

where the nodes $v^0,\ldots,v^{m-1}$ have special controls not present in $\mathscr{K}$, and they are used only to simplify the translation. In practice, these special nodes give a "name" to each hole of the bigraphs, i.e., the node v^i represents the i-hole of the bigraphs. Notice that, the hole sequence may not follow necessarily the numeration of holes, as shown in the bigraph in Figure 6.

The following result, states the expressivity of our language.


w
x
0
z x
1
l
w
y
1
L
l'
x
0
$G : \langle 2, (\{z,x\},\{x\}), \{z,x,y\}\rangle \to \langle 2, (\{w\},\{w\}), \{x,y,w\}\rangle$
$\llparenthesis - \rrparenthesis$
x
0
v^1
z
x
y
1
v^0
y

$$Q_0 : \{x\}, Q_1 : \{x,z\}; \{y\} \vdash$$
$$\nu z.((((l(x,z) \mid Q_1) \parallel \varepsilon) \downarrow z \parallel (L(y,z)[(Q_0 \mid l'(w,x))[x \mapsto s]\backslash s] \parallel \varepsilon) \downarrow w)[z \mapsto w]) : (\{w\}\{w\}, \{x,y\})$$

Figure 6: An example of encoding a binding bigraph into an agent-graph.

Proposition 6 *Let $G : \langle m, \vec{X}_B, (\bigcup \vec{X}_B) \uplus X_F \rangle \to \langle n, \vec{Y}_B, (\bigcup \vec{Y}_B) \uplus Y_F \rangle$ be a binding bigraph. Then, $\llparenthesis G \rrparenthesis_{\vec{Q}}$ is an agent-graph s.t. $\vec{Q} : \vec{X}_B; X_F \vdash \llparenthesis G \rrparenthesis_{\vec{Q}} : (\vec{Y}_B, Y_F)$, and $[\![\vec{Q} : \vec{X}_B; X_F \vdash \llparenthesis G \rrparenthesis_{\vec{Q}} : (\vec{Y}_B, Y_F)]\!] = G$.*

We can also establish nice connections between the axiomatizations of the two categories.

Proposition 7 *Let $G, G' : \langle m, \vec{X}_B, (\bigcup \vec{X}_B) \uplus X_F \rangle \to \langle n, \vec{Y}_B, (\bigcup \vec{Y}_B) \uplus Y_F \rangle$ be two binding bigraphs over a given signature. Then, $G = G'$ if and only if $\vec{Q} : \vec{X}_B; X_F \vdash \llparenthesis G \rrparenthesis_{\vec{Q}} \equiv \llparenthesis G' \rrparenthesis_{\vec{Q}}$.*

Proposition 8 *For $\Gamma; \tau \vdash A : (\vec{\sigma}, \rho)$ a typing judgment: $\Gamma; \tau \vdash \llparenthesis [\![\Gamma; \tau \vdash A : (\vec{\sigma}, \rho)]\!] \rrparenthesis_{\mathrm{dom}(\Gamma)} \equiv A$.*

These results induces a *normal form* for agent-graphs inspired to the discrete normal form of binding bigraphs. This normal form tries to separate the notions of nesting and linking:

$$A \equiv \nu \vec{z}. \left((\bar{A}_0 \parallel \ldots \parallel \bar{A}_{n-1})[\vec{X} \mapsto \vec{x}] \right)$$
$$\bar{A} \equiv \big(L_0(\vec{x}_0)[\bar{A}^0[\vec{Y}_0 \mapsto \vec{y}_0]\backslash \vec{y}_0] \mid \ldots \mid L_{m-1}(\vec{x}_{m-1})[\bar{A}^{m-1}[\vec{Y}_{m-1} \mapsto \vec{y}_{m-1}]\backslash \vec{y}_{m-1}] \mid$$
$$l_0(\vec{z}_0) \mid \ldots \mid l_{k-1}(\vec{z}_{k-1}) \mid X_0[\vec{Z}_0 \mapsto \vec{z}_0] \uparrow \vec{z}_0 \mid \cdots \mid X_{h-1}[\vec{Z}_{h-1} \mapsto \vec{z}_{h-1}] \uparrow \vec{z}_{h-1} \big) \downarrow W\,.$$

Proposition 9 *Every agent-graph is structural equivalent to an agent-graph in normal form.*

Finally, notice that the mapping $\llparenthesis - \rrparenthesis : \mathbf{Bbg}(\mathscr{K}) \to \mathbf{A}(\mathscr{L}_{\mathscr{K}})$ is not a functor, because the composition of wirings in binding bigraphs is not respected by the graph composition defined in virtue of Lemma 1. Therefore, $\mathbf{Bbg}(\mathscr{K})$ and $\mathbf{A}(\mathscr{L}_{\mathscr{K}})$ are not isomorphic. However, as we will see next, composition is respected in the important subcategories of pure and local bigraphs.

5 Characterizing pure, local and prime bigraphs

In this section we show that pure, local and prime bigraphs can be captured by simple syntactic conditions on the language and types of the typed language presented in Section 3. Indeed, these

$$\begin{tikzcd}
(\mathbf{P}, \|, \langle\rangle\,;\emptyset \vdash \varepsilon : \emptyset) \arrow[r, tail] \arrow[d, leftrightarrow, "\llbracket - \rrbracket"', "(\!| - |\!)"] & (\mathbf{A}, \|, \langle\rangle\,;\emptyset \vdash \varepsilon : (\langle\rangle, \emptyset)) \arrow[d, leftrightarrow, "\llbracket - \rrbracket"', "(\!| - |\!)"] & (\mathbf{L}, \|, \langle\rangle \vdash \varepsilon : \langle\rangle) \arrow[l, tail] \arrow[r, "\Pi"'] \arrow[d, leftrightarrow, "\llbracket - \rrbracket"', "(\!| - |\!)"] & (\mathbf{H}, |, \vdash \varepsilon : \emptyset) \arrow[d, leftrightarrow, "\llbracket - \rrbracket"', "(\!| - |\!)"] \\
(\mathbf{Big}, \|, id_{\langle 0,\emptyset\rangle}) \arrow[r, tail] & (\mathbf{Bbg}, \|, id_{\langle 0,\emptyset,\emptyset\rangle}) & (\mathbf{Lbg}, \|, id_{()}) \arrow[l, tail] \arrow[r, "\mathscr{U}"] & (\mathbf{Pbg}, |, id_{\emptyset})
\end{tikzcd}$$

Figure 7: Relations among the categories under investigation.

$$\dfrac{}{\langle\rangle\,;\tau \vdash \varepsilon : (0,\tau)} \qquad \dfrac{}{\langle\rangle\,;\tau \vdash \mathbf{0} : (1,\tau)} \qquad \dfrac{}{\langle\rangle\,;\tau \vdash l(\vec{x}) : (1, \vec{x} \cup \tau)}$$

$$\dfrac{\Gamma;\tau \vdash A : (1,\rho)}{\Gamma;\tau \vdash L(\vec{x})[A \backslash \langle\rangle] : (1, \rho \cup \vec{x})} \qquad \dfrac{}{X\,;\tau \vdash X : (1,\tau)}$$

$$\dfrac{\Gamma;\tau \vdash A : (1,\rho) \;\; \Gamma';\tau' \vdash A' : (1,\rho') \;\; \tau \cap \tau' = \emptyset}{\Gamma,\Gamma';\tau \cup \tau' \vdash A \,|\, A' : (1, \rho \cup \rho')} \qquad \dfrac{\Gamma;\tau \vdash A : (n,\rho) \;\; \Gamma';\tau' \vdash A' : (n',\rho') \;\; \tau \cap \tau' = \emptyset}{\Gamma,\Gamma';\tau \cup \tau' \vdash A \parallel A' : (n+n', \rho \cup \rho')}$$

$$\dfrac{\Gamma;\tau \vdash A : (n,\rho)}{\Gamma;\tau \vdash \nu x.A : (n, \rho \setminus \{x\})} \qquad \dfrac{\Gamma;\tau \vdash A : (n,\rho) \;\; \pi \text{ permutation}}{\pi(\Gamma)\,;\tau \vdash A : (n,\rho)} \qquad \dfrac{\Gamma;\tau \vdash A : (n,\rho)}{\Gamma;\tau \vdash A[x \mapsto y] : (n, (\rho \setminus \{x\}) \cup \{y\})}$$

Figure 8: Typing rules for restricted agent-graphs where all names are global.

subcategories are covered by the same sublanguage, obtained by removing $\downarrow$ and $\uparrow$:

$$A ::= \varepsilon \mid \mathbf{0} \mid l(\vec{x}) \mid L(\vec{x})[A \backslash \vec{y}] \mid X \mid A|A \mid A \parallel A \mid \nu z.A \mid A[w \mapsto z]\,. \tag{1}$$

Despite we use the same (sub)language, and essentially the same typing rules of Figure 3, we are able to describe both pure and local bigraphs, just by restricting the form of types and typing environment. Figure 7 summarizes the correspondences among the categories under investigation.

5.1 Pure bigraphs

In pure bigraphs all names are global, hence, variables and agents cannot have localized names. Therefore, a typing system for pure bigraphs is derivable from the system in Figure 3 by simply restricting to types of the form $(\vec{\emptyset}, \tau)$, while the variables in the environment can have only $\emptyset$ as type. The only function of $\vec{\emptyset}$ is to count the locations of the system. Therefore, taking $n = |\vec{\emptyset}|$, a typing judgement is simply of the form $\Gamma;\tau \vdash A : (n,\rho)$ where A is a term as per (1). We can hence define *global type* (n,ρ) is a pair where $n \in \mathbb{N}$ and ρ is a simple types; an environment $\Gamma;\tau$ is a list of variables $\Gamma = \vec{X} = X_0,\dots,X_{n-1}$, together with a simple-type τ.

Notice that for L non-atomic, it must be $in(L) = 0$, because there are no local names which can be linked to an in-tentacle. This is enforced by the typing system, which is given Figure 8. These rules are essentially the same of Figure 3, just with the restricted form of types and environments.

Definition 8 The category $\mathbf{P}(\mathscr{L})$ of agent-graphs, over a ranked alphabet $\mathscr{L}$, has types (m,ρ) as objects, and judgments as morphisms, i.e., if $X_0,\dots,X_{n-1}\,;\tau \vdash A : (m,\rho)$ then $(X_0,\dots,X_{n-1},A) : (n,\tau) \to (m,\rho)$ is a morphism. Composition is defined in virtue of Lemma 1.

Proposition 10 $(\mathbf{P}(\mathscr{L}), \|, \langle\rangle\,;\emptyset \vdash \varepsilon : \emptyset)$ *is a strict symmetric monoidal category.*

$$\frac{}{\langle\rangle \vdash \varepsilon : \langle\rangle} \qquad \frac{}{\langle\rangle \vdash \mathbf{0} : \emptyset} \qquad \frac{}{\langle\rangle \vdash l(\vec{x}) : \vec{x}} \qquad \frac{\Gamma \vdash A : \sigma \cup \vec{y} \quad \vec{y} \cap \sigma = \emptyset}{\Gamma \vdash L(\vec{x})[A\backslash\vec{y}] : \sigma \cup \vec{x}}$$

$$\frac{}{X : \sigma \vdash X : \sigma} \qquad \frac{\Gamma \vdash A : \sigma \quad \Gamma' \vdash A' : \sigma'}{\Gamma, \Gamma' \vdash A \mid A' : \sigma \cup \sigma'} \qquad \frac{\Gamma \vdash A : \vec{\sigma} \quad \Gamma' \vdash A' : \vec{\sigma}'}{\Gamma, \Gamma' \vdash A \parallel A' : \vec{\sigma}\vec{\sigma}'}$$

$$\frac{\Gamma \vdash A : \vec{\sigma}}{\Gamma \vdash \nu x.A : \vec{\sigma} \setminus \{x\}} \qquad \frac{\Gamma \vdash A : \vec{\sigma} \quad \pi \text{ permutation}}{\pi(\Gamma) \vdash A : \vec{\sigma}} \qquad \frac{\Gamma \vdash A : \vec{\sigma}}{\Gamma \vdash A[x \mapsto y] : (\vec{\sigma} \setminus \{x\}) \cup \{y\}}$$

Figure 9: Typing rules for restricted agent-graphs where all names are local.

The encoding functor $[\![-]\!] : \mathbf{Big}(\mathscr{K}) \to \mathbf{L}(\mathscr{P}_{\mathscr{K}})$ and the representation function $(\!|-|\!) : \mathbf{P}(\mathscr{K}) \to \mathbf{Big}(\mathscr{K}_{\mathscr{L}})$ are particular cases of the ones for binding bigraphs. Again the two maps establish a bijection between the two categories.

5.2 Local bigraphs

In local bigraphs all names are localized, hence there are no global names, and variables can have only their own names. So, the typing is obtained again from the system in Figure 3 by simply restricting to types of the form $(\vec{\sigma}, \emptyset)$, while in the environment the set of the global names is always $\emptyset$. More formally, a typing judgement is of the form $\Gamma \vdash A : \vec{\sigma}$ where A is a term generated by the grammar (1), a *local type* $\vec{\tau} = \tau_0 \dots \tau_{n-1}$ is a list of simple types, and an *environment* Γ is a list of typed variables ($\Gamma = \vec{X} : \vec{\tau} = X_0 : \tau_0, \dots, X_{n-1} : \tau_{n-1}$). The type inference rules are in Figure 9. Notice that, in local bigraphs, non-atomic hyperedges can have non-zero in-rank.

Definition 9 The category $\mathbf{L}(\mathscr{L})$ of agent-graphs, over a ranked alphabet $\mathscr{L}$, has local types $\vec{\sigma}$ as objects, and judgments as morphisms, that is, if $X_0 : \tau_0, \dots, X_{n-1} : \tau_{n-1} \vdash A : \vec{\sigma}$ then $(X_0, \dots, X_{n-1}, A) : \vec{\tau} \to \vec{\sigma}$ is a morphism. Composition is defined in virtue of Lemma 1.

Proposition 11 $(\mathbf{L}(\mathscr{L}), \|, \langle\rangle \vdash \varepsilon : \langle\rangle)$ *is a strict symmetric monoidal category.*

The two encoding functors $[\![-]\!] : \mathbf{Lbg}(\mathscr{K}) \to \mathbf{L}(\mathscr{L}_{\mathscr{K}})$, and $(\!|-|\!) : \mathbf{L}(\mathscr{K}) \to \mathbf{Lbg}(\mathscr{K}_{\mathscr{L}})$ are particular cases of the ones for binding bigraphs. Notice that, in this particular case, $(\!|-|\!)$ is actually a functor; and, as before, the two functors establish a bijection between the two categories.

5.3 Prime bigraphs

Following the idea of the functor $[\![-]\!]$ from agent-graph to bigraphs, we can identify a subcategory of $\mathbf{A}$, where all agents have zero or one variable. These are *prime bigraph*, that is bigraphs with at most one hole. One may think of these bigraphs as single-located (open) systems.

We can characterize pure prime bigraphs by a restriction on agent types. A typing judgement is of the form $\Gamma \vdash A : \sigma$ where A is a one-variable term generated by $A ::= \mathbf{0} \mid l(\vec{x}) \mid X \mid A|A \mid \nu z.A \mid A[w \mapsto z]$. A *prime type* σ is a simple type, and an environment Γ is a list of typed variables of at most length one, i.e., $\Gamma ::= \langle\rangle \mid X : \rho$. The induced type inference rules are in Figure 10.

Definition 10 The category $\mathbf{H}$ has simple types (τ) as objects, and judgments as morphisms,

$$\frac{}{\langle\rangle \vdash \mathbf{0} : \emptyset} \qquad \frac{}{\langle\rangle \vdash l(\vec{x}) : \vec{x}} \qquad \frac{\Gamma \vdash A : \sigma \quad \Gamma' \vdash A' : \sigma' \quad |\Gamma,\Gamma'| < 2}{\Gamma,\Gamma' \vdash A \mid A' : \sigma \cup \sigma'}$$

$$\frac{}{X : \sigma \vdash X : \sigma} \qquad \frac{\Gamma \vdash A : \sigma}{\Gamma \vdash \nu x.A : \sigma \setminus \{x\}} \qquad \frac{\Gamma \vdash A : \sigma}{\Gamma \vdash A[x \mapsto y] : (\sigma \setminus \{x\}) \cup \{y\}}$$

Figure 10: Typing rules for restricted agent-graphs with one locality.

i.e., if $X : \tau \vdash A : \rho$ then $(X,A) : \tau \to \rho$ is a morphism or if $\langle\rangle \vdash A : \rho$ then $(\langle\rangle,A) : \emptyset \to \rho$ is a morphism. Composition is defined as follows:

$$\frac{\Gamma \vdash A : \tau \quad X : \rho \vdash A' : \rho}{\Gamma \vdash A'\{A/X\} : \rho}$$

Proposition 12 $(\mathbf{H}, |, \langle\rangle \vdash \mathbf{0} : \emptyset)$ *is a strict symmetric monoidal category.*

The two encoding functors $[\![-]\!] : \mathbf{Pbg}(\mathscr{K}) \to \mathbf{H}(\mathscr{L}_{\mathscr{K}})$ and $(\!|-|\!) : \mathbf{H}(\mathscr{L}) \to \mathbf{Pbg}(\mathscr{L}_{\mathscr{K}})$ can be defined as a "simplification" of the ones for local bigraphs.

Proposition 13 *Let A, A' be terms; then, for every environment $X : \tau$: $X : \tau \vdash A \equiv A'$ if and only if $[\![X : \tau \vdash A : \tau']\!] = [\![X : \tau \vdash A' : \tau']\!]$.*

Proposition 14 *Let $H, H' : (\langle\rangle) \to (Y)$ be two prime graphs; then, $H = H'$ iff $\langle\rangle \vdash (\!|H|\!)_{\langle\rangle} \equiv (\!|H'|\!)_{\langle\rangle}$. Instead, if $H, H' : (X) \to (Y)$, then $H = H'$ iff $Q : X \vdash (\!|H|\!)_Q \equiv (\!|H'|\!)_Q$.*

Forgetting localities. Let us consider only atomic signatures $\mathscr{K}_a$, that is, where all controls are atomic, and hence there is no nesting of nodes. In this case, we can define a functor $\mathscr{U} : \mathbf{Lbg}(\mathscr{K}_a) \to \mathbf{Pbg}(\mathscr{K}_a)$ which "forgets" the localities of a local bigraph, merging all roots into a single one and all sites (holes) into a single one. Formally:
Objects: $\mathscr{U}((X_0 \ldots X_{n-1})) = X_0 + \cdots + X_{n-1}$.
Morphisms: $\mathscr{U}((V,E,ctrl,prnt,link)) = (V,E,ctrl,prnt',link)$ where $prnt'(v) = 0$, for all v.

The previous functors $[\![-]\!], (\!|-|\!)$ and the forgetful functor $\mathscr{U}$ induce a forgetful functor $\Pi : \mathbf{L}(\mathscr{L}_{\mathscr{K}_a}) \to \mathbf{H}(\mathscr{L}_{\mathscr{K}_a})$, defined as follows:
Objects: $\Pi(\sigma_0 \ldots \sigma_{n-1}) = \sigma_0 + \cdots + \sigma_{n-1}$.
Morphisms: given a graph in normal form $A \equiv \nu\vec{z}.\Big((\bar{A}_0 \parallel \ldots \parallel \bar{A}_{n-1})[\vec{X} \mapsto \vec{x}]\Big)$, where every subgraph $\bar{A}_i \equiv \Big(l^i_0(\vec{z}^i_0) \mid \ldots \mid l^i_{k_i-1}(\vec{z}^i_{k_i-1}) \mid X^i_0[\vec{Z}^i_0 \mapsto \vec{z}^i_0] \mid \cdots \mid X^i_{h_i-1}[\vec{Z}^i_{h_i-1} \mapsto \vec{z}^i_{h_i-1}]\Big)$, then

$$\begin{aligned}\Pi(A) = \nu\vec{z}.\big((l^0_0(\vec{z}^0_0) \mid \ldots \mid l^0_{k_0-1}(\vec{z}^0_{k_0-1})) \mid \ldots \mid (l^{n-1}_0(\vec{z}^{n-1}_0) \mid \ldots \mid l^{n-1}_{k_{n-1}-1}(\vec{z}^{n-1}_{k_{n-1}-1})) \mid \\ X[\vec{Z}^0_0 \mapsto \vec{z}^0_0] \ldots [\vec{Z}^0_{h_0-1} \mapsto \vec{z}^0_{h_0-1}] \ldots [\vec{Z}^{n-1}_0 \mapsto \vec{z}^{n-1}_0] \ldots [\vec{Z}^{n-1}_{h_{n-1}-1} \mapsto \vec{z}^{n-1}_{h_{n-1}-1}]\big)\end{aligned}$$

In practice the above functor merges all the separate agent-graphs into a single-located graph. It translates a $\parallel$ operator with the $\mid$ one and unifies all variables into a single one.

As a consequence of the definitions of the functors defined above, we can prove the following results. Notice that the lists $\vec{X}$, $\vec{Q}$ and $\vec{\tau}$ are either empty or just singletons.

Proposition 15 *Let* $\vec{X} : \vec{\tau} \vdash A : \rho = \Pi(\Gamma \vdash B : \vec{\sigma})$*; then,* $\vec{X} : \vec{\tau} \vdash (\!|\mathscr{U}([\![\Gamma \vdash B : \vec{\sigma}]\!])|\!)_{\vec{X}} \equiv A$.

Proposition 16 *1. Let* $P : (\vec{X}) \to (Y)$ *be a prime bigraph, then* $[\![\vec{Q} : \vec{X} \vdash (\!|P|\!)_{\vec{Q}}]\!] = P$.

2. Let $\vec{X} : \vec{\tau} \vdash A : \sigma$ *be a term, then* $\vec{X} : \vec{\tau} \vdash (\!|[\![\vec{X} : \vec{\tau} \vdash A : \sigma]\!]|\!)_{\vec{X}} \equiv A$.

6 Comparing with SHR hypergraphs and ADR designs

Our language for binding bigraphs can be used for capturing formalisms introduced in literature, often for quite different purposes. Here we consider the hypergraphs used in *Synchronized Hyperedge Rewriting* (SHR) [FHL+05] and the "designs" of *Architectural Design Rewriting* (ADR) [BLMT07]. Both are derived from the algebra of graphs introduced first in [CMR94].

SHR hypergraphs. SHR is a framework that allows hypergraph transformations by means of local productions replacing a single hyperedge by a generic hypergraph, possibly with constraints given by the surrounding nodes. The global rewriting is obtained by combining different local production whose conditions are compatible (with respect to some synchronization model).

In this paper, we are interested only in SHR hypergraphs, which are inductively defined as:

$$G ::= \mathbf{0} \mid l(\vec{x}) \mid G|G \mid \nu x.G$$

where $\mathbf{0}$ is the empty graph, the hyperedge l is linked to the nodes in $\vec{x}$, and ν binds x in G.

Clearly, the SHR grammar is a particular case of the one for prime bigraphs (Section 5.3), and specifically when the variable and substitutions are dropped.

ADR designs. ADR graphs (called *designs*) resemble SHR graphs, but they have a notion of graph nesting, as some hyperedges can contain other graphs. Such nesting is used for *incremental modelling*, that is, edges can be refined into graphs or vice versa graphs collapse into edges. The ADR designs are inductively defined as:

$$D ::= L[\lambda\vec{x}.G] \qquad\qquad G ::= \mathbf{0} \mid x \mid l(\vec{x}) \mid G|G \mid \nu x.G \mid D(\vec{x})$$

where $\mathbf{0}$ is the empty graph, the hyperedge l is linked to the nodes in $\vec{x}$, ν binds x in G, $D(\vec{x})$ is a design generated by attaching design D to nodes $\vec{x}$, and finally $L[\lambda\vec{x}.G]$ represent a design L, with "body graph" G and exposing the names $\vec{x}$ in its interface.

The grammar of designs recalls the one defined for local bigraphs, when the $\|$ composition is omitted. In such a case, we deal with graphs residing in only one location. A formal translation of the ADR design grammar into the grammar in (1) can be defined as follow:

$$\begin{array}{lll} T(\mathbf{0}) = \mathbf{0} & T(x) = \mathbf{0}[z \mapsto x] \quad (z \text{ fresh}) & T(G_1 \mid G_2) = T(G_1) \mid T(G_2) \\ T(l(\vec{x})) = l(\vec{x}) & T(\nu x.G) = \nu x.T(G) & T(L[\lambda\vec{x}.G](\vec{y})) = L(\vec{y})[G\backslash\vec{x}] \end{array}$$

By means of these translations of SHR hypergraphs as prime bigraphs, and ADR designs as local bigraphs, we can transfer results and constructions among formalisms developed independently. As examples, it is possible to extend the SHR semantics allowing for not only replacing single hyperedges, but more complex graphs; moreover, we can also define congruent bisimulations for SHR systems via the so-called IPO construction over bigraphical reactive systems.

7 Conclusion

In this paper we have first defined an algebra of typed term graphs which corresponds precisely to binding bigraphs, on a given signature. Secondly, we have shown that particular sublanguages of our main language properly characterize interesting subclasses of bigraphs, more precisely: pure and local bigraphs. Moreover, on this last kind of bigraphs we also give a reduced language for dealing with one-location (bi)graphs, named prime bigraphs. So, those languages can be used in place of the more complex bigraph algebra already present in literature. A family of bigraphical calculi has been introduced in [DK08]; however, these calculi has been suitably restricted for modelling biological systems and do not cover all possible bigraphs over a given signature.

Finally, it turns out that these languages are strictly connected with two well-know formalisms: *Synchronized Hyperedge Replacement* hypergraphs, which can be represented as a sublanguage of the algebra for prime bigraphs (over atomic signatures), and *Architectural Design Rewriting* designs, which are a sub-case of the local bigraphs' language.

A possible future work is to take advantage of the rich theory provided by bigraphical reactive systems [JM03], in order to obtain interesting results about SHR and ADR. In particular, we hope to generalize the transitions allowed in SHR graphs, which only rewrites a single hyperedge, to more general ones dealing with (sub)graphs. Moreover, bigraphs allow to synthesise labelled transition systems out of rewriting rules, via the so-called *idem-pushout* construction [LM00]; it is important to notice that the bisimilarity induced by this labelled transitions system (LTS) is always a congruence. Therefore, given a reactive system over SHR (ADR) graphs, we can derive the labelled transition system in bigraphs, and remap it on SHR (ADR) graphs. Then, the inductive definition of SHR (ADR) agents can be useful for defining an SOS-like presentation of the LTS derived in this way.

Acknowledgments. We thank Emilio Tuosto and Ivan Lanese for useful discussions about SHR.

Bibliography

[BDE+06] L. Birkedal, S. Debois, E. Elsborg, T. Hildebrandt, H. Niss. Bigraphical Models of Context-Aware Systems. In Aceto and Ingólfsdóttir (eds.), *Proc. FoSSaCS*. Lecture Notes in Computer Science 3921, pp. 187–201. Springer, 2006.

[BDGM07] L. Birkedal, T. C. Damgaard, A. J. Glenstrup, R. Milner. Matching of Bigraphs. *Electr. Notes Theor. Comput. Sci.* 175(4):3–19, 2007.

[BGH+08] M. Bundgaard, A. J. Glenstrup, T. T. Hildebrandt, E. Højsgaard, H. Niss. Formalizing Higher-Order Mobile Embedded Business Processes with Binding Bigraphs. In Lea and Zavattaro (eds.), *COORDINATION*. Lecture Notes in Computer Science 5052, pp. 83–99. Springer, 2008.

[BLMT07] R. Bruni, A. Lluch-Lafuente, U. Montanari, E. Tuosto. Service Oriented Architectural Design. In Barthe and Fournet (eds.), *Proc. TGC*. Lecture Notes in Computer Science 4912, pp. 186–203. Springer, 2007.

[CMR94] A. Corradini, U. Montanari, F. Rossi. An Abstract Machine for Concurrent Modular Systems: CHARM. *Theor. Comput. Sci.* 122(1&2):165–200, 1994.

[DB06] T. C. Damgaard, L. Birkedal. Axiomatizing Binding Bigraphs. *Nord. J. Comput.* 13(1-2):58–77, 2006.

[DK08] T. C. Damgaard, J. Krivine. A Generic Language for Biological Systems based on Bigraphs. Technical report TR-2008-115, IT University of Copenhagen, Dec. 2008.

[FHL+05] G. L. Ferrari, D. Hirsch, I. Lanese, U. Montanari, E. Tuosto. Synchronised Hyperedge Replacement as a Model for Service Oriented Computing. In Boer et al. (eds.), *Proc. FMCO*. Lecture Notes in Computer Science 4111, pp. 22–43. Springer, 2005.

[GM07] D. Grohmann, M. Miculan. Reactive Systems over Directed Bigraphs. In Caires and Vasconcelos (eds.), *Proc. CONCUR 2007*. Lecture Notes in Computer Science 4703, pp. 380–394. Springer, 2007.

[JM03] O. H. Jensen, R. Milner. Bigraphs and transitions. In *Proc. POPL*. Pp. 38–49. 2003.

[JM04] O. H. Jensen, R. Milner. Bigraphs and mobile processes (revised). Technical report UCAM-CL-TR-580, Computer Laboratory, University of Cambridge, 2004.

[LM00] J. J. Leifer, R. Milner. Deriving Bisimulation Congruences for Reactive Systems. In Palamidessi (ed.), *Proc. CONCUR*. Lecture Notes in Computer Science 1877, pp. 243–258. Springer, 2000.

[LM06] J. J. Leifer, R. Milner. Transition systems, link graphs and Petri nets. *Mathematical Structures in Computer Science* 16(6):989–1047, 2006.

[Mil01] R. Milner. Bigraphical Reactive Systems. In Larsen and Nielsen (eds.), *Proc. 12th CONCUR*. Lecture Notes in Computer Science 2154, pp. 16–35. Springer, 2001.

[Mil04] R. Milner. Bigraphs whose names have multiple locality. Technical report 603, University of Cambridge, CL, Sept. 2004.

A Structural congruence

The free name function fn is defined as follows with respect to an environment (Γ, τ).

$$\begin{array}{rlrl}
fn_{\Gamma;\tau}(\varepsilon) = \tau & & fn_{\Gamma;\tau}(L(\vec{x})[A \backslash \vec{y}]) = (fn_{\Gamma;\tau}(A) \setminus \vec{y}) \cup \vec{x} \cup \tau & \\
fn_{\Gamma;\tau}(\mathbf{0}) = \tau & & fn_{\Gamma;\tau}(A_1 \mid A_2) = fn_{\Gamma;\tau}(A_1) \cup fn_{\Gamma;\tau}(A_2) & \\
fn_{\Gamma;\tau}(l(\vec{x})) = \vec{x} \cup \tau & & fn_{\Gamma;\tau}(A_1 \parallel A_2) = fn_{\Gamma;\tau}(A_1) \cup fn_{\Gamma;\tau}(A_2) & \\
fn_{\Gamma;\tau}(\nu y.A) = fn_{\Gamma;\tau}(A) \setminus \{y\} & & fn_{\Gamma;\tau}(A[w \mapsto z]) = (fn_{\Gamma;\tau}(A) \setminus \{w\}) \cup \{z\} & \\
fn_{\Gamma;\tau}(A \downarrow x) = fn_{\Gamma;\tau}(A) & & fn_{\Gamma;\tau}(A \uparrow x) = fn_{\Gamma;\tau}(A) & \\
fn_{\Gamma;\tau}(X_i) = \tau_i \cup \tau \quad \text{if } X_i{:}\tau_i \in \Gamma & & &
\end{array}$$

In the following table, the structural congruence for agent-graph is defined with respect to some environment $\Gamma;\tau$.

$$\Gamma;\tau \vdash A \mid \mathbf{0} \equiv A$$
$$\Gamma;\tau \vdash A_1 \mid A_2 \equiv A_2 \mid A_1$$
$$\Gamma;\tau \vdash (A_1 \mid A_2) \mid A_3 \equiv A_1 \mid (A_2 \mid A_3)$$
$$\Gamma;\tau \vdash A \parallel \varepsilon \equiv A$$
$$\Gamma;\tau \vdash \varepsilon \parallel A \equiv A$$
$$\Gamma;\tau \vdash (A_1 \parallel A_2) \parallel A_3 \equiv A_1 \parallel (A_2 \parallel A_3)$$
$$\Gamma;\tau \vdash \nu x.\mathbf{0} \equiv \mathbf{0} \text{ if } x \notin fn_{\Gamma;\tau}(\mathbf{0})$$
$$\Gamma;\tau \vdash \nu x.\varepsilon \equiv \varepsilon \text{ if } x \notin fn_{\Gamma;\tau}(\varepsilon)$$
$$\Gamma;\tau \vdash \nu x.\nu y.A \equiv \nu y.\nu x.A$$
$$\Gamma;\tau \vdash \nu x.A \equiv \nu y.(A\{y/x\}) \text{ if } y \notin fn_{\Gamma;\tau}(A)$$
$$\Gamma;\tau \vdash A[x \mapsto y] \equiv (A\{z/x\})[z \mapsto y] \text{ if } z \notin fn_{\Gamma;\tau}(A)$$
$$\Gamma;\tau \vdash A[x \mapsto x] \equiv A \text{ if } x \in fn_{\Gamma;\tau}(A)$$
$$\Gamma;\tau \vdash A[x \mapsto y] \equiv A \text{ if } x \notin fn_{\Gamma;\tau}(A) \wedge y \in fn_{\Gamma;\tau}(A)$$
$$\Gamma;\tau \vdash A[x \mapsto y][w \mapsto z] \equiv A[w \mapsto z][x \mapsto y] \text{ if } x \neq z, y \neq w$$
$$\Gamma;\tau \vdash A[x \mapsto y][y \mapsto z] \equiv A[x \mapsto z] \text{ if } y \notin fn_{\Gamma;\tau}(A)$$
$$\Gamma;\tau \vdash \nu y.(A[x \mapsto y]) \equiv \nu x.A \text{ if } y \notin fn_{\Gamma;\tau}(A)$$
$$\Gamma;\tau \vdash \nu z.(A[x \mapsto y]) \equiv (\nu z.A)[x \mapsto y] \text{ if } z \notin \{x,y\}$$
$$\Gamma;\tau \vdash A \downarrow x \uparrow x \equiv A \text{ if } x \in fn_{\Gamma;\tau}(A)$$
$$\Gamma;\tau \vdash A \uparrow x \downarrow x \equiv A \text{ if } x \in fn_{\Gamma;\tau}(A)$$
$$\Gamma;\tau \vdash A \downarrow x[x \mapsto y] \equiv A[x \mapsto y] \downarrow y$$
$$\Gamma;\tau \vdash A \downarrow x[y \mapsto z] \equiv A[y \mapsto z] \downarrow x \text{ if } y \neq x \wedge z \neq x$$
$$\Gamma;\tau \vdash A \uparrow x[x \mapsto y] \equiv A[x \mapsto y] \uparrow y$$
$$\Gamma;\tau \vdash A \uparrow x[y \mapsto z] \equiv A[y \mapsto z] \uparrow x \text{ if } y \neq x \wedge z \neq x$$
$$\Gamma;\tau \vdash \nu x.(A \downarrow x) \equiv \nu x.A \text{ if } x \in fn_{\Gamma;\tau}(A)$$
$$\Gamma;\tau \vdash \nu y.(A \downarrow x) \equiv (\nu y.A) \uparrow x \text{ if } x \neq y$$
$$\Gamma;\tau \vdash \nu x.(A \uparrow x) \equiv \nu x.A \text{ if } x \in fn_{\Gamma;\tau}(A)$$
$$\Gamma;\tau \vdash \nu y.(A \uparrow x) \equiv (\nu y.A) \uparrow x \text{ if } x \neq y$$

$$\Gamma;\tau \vdash \nu x.(A_1 \mid A_2) \equiv \nu x.A_1 \mid A_2 \text{ if } x \notin fn_{\Gamma;\tau}(A_2)$$
$$\Gamma;\tau \vdash \nu x.(A_1 \parallel A_2) \equiv \nu x.A_1 \parallel A_2 \text{ if } x \notin fn_{\Gamma;\tau}(A_2)$$
$$\Gamma;\tau \vdash \nu x.(A_1 \parallel A_2) \equiv A_1 \parallel \nu x.A_2 \text{ if } x \notin fn_{\Gamma;\tau}(A_1)$$
$$\Gamma;\tau \vdash (A_1 \mid A_2)[x \mapsto y] \equiv A_1[x \mapsto y] \mid A_2 \text{ if } x \notin fn_{\Gamma;\tau}(A_2)$$
$$\Gamma;\tau \vdash (A_1 \mid A_2)[x \mapsto y] \equiv A_1[x \mapsto y] \mid A_2[x \mapsto y]$$
$$\Gamma;\tau \vdash (A_1 \parallel A_2)[x \mapsto y] \equiv A_1[x \mapsto y] \parallel A_2 \text{ if } x \notin fn_{\Gamma;\tau}(A_2) \wedge x \in fn_{\Gamma;\tau}(A_1)$$
$$\Gamma;\tau \vdash (A_1 \parallel A_2)[x \mapsto y] \equiv A_1 \parallel A_2[x \mapsto y] \text{ if } x \notin fn_{\Gamma;\tau}(A_1) \wedge x \in fn_{\Gamma;\tau}(A_2)$$
$$\Gamma;\tau \vdash (A_1 \parallel A_2)[x \mapsto y] \equiv A_1[x \mapsto y] \parallel A_2[x \mapsto y] \text{ if } x \in fn_{\Gamma;\tau}(A_1) \cap fn_{\Gamma;\tau}(A_2)$$
$$\Gamma;\tau \vdash (A_1 \parallel A_2)[x \mapsto y] \equiv A_1[x \mapsto y] \parallel A_2[x \mapsto y] \text{ if } x \notin fn_{\Gamma;\tau}(A_1) \cup fn_{\Gamma;\tau}(A_2)$$
$$\Gamma;\tau \vdash (A_1 \mid A_2) \downarrow x \equiv A_1 \downarrow x \mid A_2 \text{ if } x \in fn_{\Gamma;\tau}(A_1) \wedge x \notin fn_{\Gamma;\tau}(A_2)$$
$$\Gamma;\tau \vdash (A_1 \mid A_2) \downarrow x \equiv A_1 \downarrow x \mid A_2 \downarrow x \text{ if } x \in fn_{\Gamma;\tau}(A_1) \cap fn_{\Gamma;\tau}(A_2)$$
$$\Gamma;\tau \vdash (A_1 \mid A_2) \uparrow x \equiv A_1 \uparrow x \mid A_2 \text{ if } x \in fn_{\Gamma;\tau}(A_1) \wedge x \notin fn_{\Gamma;\tau}(A_2)$$
$$\Gamma;\tau \vdash (A_1 \mid A_2) \uparrow x \equiv A_1 \uparrow x \mid A_2 \uparrow x \text{ if } x \in fn_{\Gamma;\tau}(A_1) \cap fn_{\Gamma;\tau}(A_2)$$
$$\Gamma;\tau \vdash (A_1 \parallel A_2) \uparrow x \equiv A_1 \uparrow x \parallel A_2 \text{ if } x \in fn_{\Gamma;\tau}(A_1) \wedge x \notin fn_{\Gamma;\tau}(A_2)$$
$$\Gamma;\tau \vdash (A_1 \parallel A_2) \uparrow x \equiv A_1 \parallel A_2 \uparrow x \text{ if } x \notin fn_{\Gamma;\tau}(A_1) \wedge x \in fn_{\Gamma;\tau}(A_2)$$
$$\Gamma;\tau \vdash (A_1 \parallel A_2) \uparrow x \equiv A_1 \uparrow x \parallel A_2 \uparrow x \text{ if } x \in fn_{\Gamma;\tau}(A_1) \cap fn_{\Gamma;\tau}(A_2)$$
$$\Gamma;\tau \vdash l(\vec{x})[x \mapsto z] \equiv l(\vec{x})\{z/x\} \text{ if } x \in \vec{x}$$
$$\Gamma;\tau \vdash L(\vec{x})[A\backslash\vec{y}] \equiv L(\vec{x})[(A\{z/y\})\backslash(\vec{y}\{z/y\})] \text{ if } x \in \vec{y} \wedge z \notin \vec{y} \cup fn_{\Gamma}(A)$$
$$\Gamma;\tau \vdash \nu z.L(\vec{x})[A\backslash\vec{y}] \equiv L(\vec{x})[(\nu z.A)\backslash\vec{y}] \text{ if } z \notin \vec{x} \cup \vec{y}$$
$$\Gamma;\tau \vdash L(\vec{x})[A\backslash\vec{Y}][x \mapsto w] \equiv L(\vec{x}\{w/x\})[A\backslash\vec{y}] \text{ if } x \in \vec{x} \wedge w \notin (fn_{\Gamma}(A) \setminus \vec{y})$$
$$\Gamma;\tau \vdash L(\vec{x})[A\backslash\vec{y}][w \mapsto z] \equiv L(\vec{x})[(A[w \mapsto z])\backslash\vec{y}] \text{ if } w \notin \vec{x} \cup \vec{y} \wedge z \notin \vec{y}$$
$$\Gamma;\tau \vdash L(\vec{x})[A[w \mapsto y]\backslash\vec{y}] \equiv L(\vec{x})[A\backslash(\vec{y}\{w/y\}] \text{ if } y \in \vec{y} \wedge w \notin fn_{\Gamma;\tau}(A)$$
$$\Gamma;\tau \vdash L(\vec{x})[A\backslash\vec{y}] \downarrow z \equiv L(\vec{x})[(A \downarrow z)\backslash\vec{y}] \text{ if } z \notin \vec{x} \cup \vec{y}$$
$$\Gamma;\tau \vdash L(\vec{x})[A\backslash\vec{y}] \uparrow z \equiv L(\vec{x})[(A \uparrow z)\backslash\vec{y}] \text{ if } z \notin \vec{x} \cup \vec{y}$$