Graph Grammars for Local Bigraphs

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

[a] *Department of Mathematics and Computer Science, University of Udine, Italy*

Abstract

We give a formal connection between the graph grammars of *Synchronized Hyperedge Replacement* and *Architectural Design Rewriting*, and *local bigraphs*. First, we define a category of SHR agents, which extend SHR graphs allowing for open systems (i.e., systems with "holes") and multiple locations. Then, we show that these agents correspond precisely to local bigraphs with atomic controls. Standard SHRs can be identified as the link graphs underlying these local bigraphs. Hence, SHR agents can be used as a language for describing local bigraphs with atomic controls.
Finally, we extend these results to ADR-like agents, where nesting of graphs within nodes is allowed. We show that these ADR agents correspond precisely to local bigraphs. Therefore, ADR agents can be used as a language for describing general local bigraphs, alternative to the (more complex) bigraphical algebra.

Keywords: Local bigraphs, Synchronized Hyperedge Replacement, concurrency models.

1 Introduction

Bigraphical Reactive Systems (BRSs) [11] 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 represented by a set of rewrite rules on this semi-structured data.

Bigraphs and BRSs are very flexible: they 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 [7,10,1,6,3].

However, a general foundational theory like this should be compared with other theories proposed for similar purposes. In this paper, we compare bigraphs and the graph grammars used in *Synchronized Hyperedge Replacement* (SHR) and *Architectural Design Rewriting* (ADR) frameworks [4,5,2].

More precisely, we show that (a slight generalization of) SHR graphs correspond precisely to *local bigraphs* (a variant of bigraphs which can deal with localized names,

[1] Email: grohmann@dimi.uniud.it
[2] Email: miculan@dimi.uniud.it

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

see [7] and Section 3) with atomic controls. Moreover, a similar generalization of ADR graphs correspond to local bigraphs.

To this end, we have to solve some technical issues. In particular, SHR and ADR graphs do not have a notion of "sub-graph (hierarchical) composition", and hence they do not yield immediately a category. We tackle this problem by extending these graph grammar with graph variables and substitutions, thus obtaining a symmetric monoidal category of *SHR (ADR) agents* (Section 2). An agent is a collection of SHR graphs, each representing a system on its own location, but possibly connected by hyperlinks; graph variables represent open parts of the agents. Then, the encoding is a functor from this category of agents into a corresponding category of local bigraphs (Section 4.1). The functor is monoidal, that is, it respects the parallel compositions of agents. Moreover, it has an inverse functor: any local bigraph with only atomic controls can be represented as an SHR agent (Section 4.2).

It is interesting to give also a characterization of the standard SHR graphs. In Section 5 we show that if we drop the place graph structure from local bigraphs (over atomic controls), we obtain a class of link graphs which correspond exactly to (possibly open) SHR graphs (not agents). Thus, given a local bigraph, its underlying link graph correspond to an SHR graph, while its place graph is simply an extra structure needed for enforcing name scoping.

In Section 6 we consider an extension of SHR agents in the spirit of ADR graphs, where a notion of nesting among edges/graphs is allowed. We generalize the previous results by showing that ADR agents correspond precisely to local bigraphs.

In our opinion, these results are important for several reasons. First, they show once more that bigraphs are a quite expressive general framework of ubiquitous systems. But more importantly, the correspondence points out that SHR and ADR graphs can be used as a language for local bigraphs, alternative to the bigraph algebra [7]. Finally, these constructions paves the way for the application of the rich theory of (local) bigraphs to the SHR and ADR frameworks, as suggested in the concluding Section 7.

2 Synchronized Hyperedge Replacement

Synchronized Hyperedge Replacement (SHR) is a hypergraph framework that allows graph 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 (w.r.t. some synchronization model). For more details see [4,5]. In this paper, we consider a graph grammar which extends the standard SHR grammar by adding graph variables. In this way we can define a composition mechanism among SHR graphs. To ensure that composition is always well-defined we introduce a type system for SHR graphs, which allows to replace variables with graphs having the same type.

A hyperedge is a labelled element from a ranked alphabet $\mathcal{L} = \{l_n\}_{n\in\mathbb{N}}$, each of which has many ordered tentacles as the rank of its label. A hypergraph is formed by a set of nodes and a set of edges, whose tentacles are connected to nodes. Moreover, a hypergraph has a set of external nodes, i.e., its interface with the environment.

Definition 2.1 Let $\mathcal{N}$ be a fixed infinite set of names, $\mathcal{V}$ be a fixed infinite set of variables, and $\mathcal{L}$ be a ranked finite alphabet of labels. A syntactic judgement is of the form $\Gamma \vdash G : \tau$ where:

(i) G is a term generated by the following grammar:

$$G ::= \mathbf{0} \mid l(\vec{x}) \mid X \mid G|G \mid \nu y.G \mid G[w \mapsto z] \tag{1}$$

where $\vec{x} \subseteq \mathcal{N}$, $l \in \mathcal{L}$ with $rank(l) = |\vec{x}|$, $X \in \mathcal{V}$, and $y, w, z \in \mathcal{N}$.

(ii) τ is a type, i.e., a finite set of names.

(iii) Γ is a list of typed variables, that is it is of the form $\Gamma = X_1 : \tau_1, \ldots, X_n : \tau_n$. $\langle\rangle$ is the empty list.

Intuitively, terms are defined inductively: the elementary graphs are: the empty graph ($\mathbf{0}$), a single hyperedge l, whose tentacles are linked to the names in $\vec{x}$ and a variable (X); complex graph are derived by parallel composing of two subgraphs ($G \mid G$), by restricting the scope of a name ($\nu y.G$) in a subgraph and by substituting a name with another ($G[w \mapsto z]$) in a subgraph.

Types (τ) describe the visible names of a graph from a context. Instead environments (Γ) describe the variables' names.

We define the function $v(t)$, that gives the set of all variables in t. We use the notation $\Gamma, X : \tau$ to denote the append of $X : \tau$ to Γ when $X \notin v(\Gamma)$. Analogously, we write Γ_1, Γ_2 to mean the concatenation of Γ_1 and Γ_2 when $v(\Gamma_1) \cap v(\Gamma_2) = \emptyset$. We defined the functions fn, bn and n w.r.t. an environment Γ, as follows:

$$fn_\Gamma(\mathbf{0}) = \emptyset \qquad fn_\Gamma(l(\vec{x})) = \{x_1, \ldots, x_{|\vec{x}|}\} \qquad fn_\Gamma(X) = \tau \quad (\text{if } X : \tau \in \Gamma)$$
$$fn_\Gamma(G_1|G_2) = fn_\Gamma(G_1) \cup fn_\Gamma(G_2) \qquad fn_\Gamma(\nu y.G) = fn_\Gamma(G) \setminus \{y\}$$
$$fn_\Gamma(G[w \mapsto z]) = (fn_\Gamma(G) \setminus \{w\}) \cup \{z\}$$
$$bn_\Gamma(\mathbf{0}) = bn_\Gamma(l(\vec{x})) = bn_\Gamma(X) = \emptyset \qquad bn_\Gamma(G_1|G_2) = bn_\Gamma(G_1) \cup bn_\Gamma(G_2)$$
$$bn_\Gamma(\nu y.G) = bn_\Gamma(G) \cup \{y\} \qquad bn_\Gamma(G[w \mapsto z]) = bn_\Gamma(G)$$

Finally, $n_\Gamma(G) = fn_\Gamma(G) \cup bn_\Gamma(G)$. Notice that in the case $G[w \mapsto z]$ if $w \notin fn(G)$ $G[w \mapsto z]$ has z as free name anyway. The idea is that in a such a case the operator $[w \mapsto z]$ can "create" unused (or idle) free names, i.e., names linked to nothing. [3]

The judgments are taken up-to a structural congruence, that captures graph isomorphisms up-to free nodes, the full list of axioms is shown in Fig. 1.

The type inference rules for judgments on terms are listed below.

$$\frac{}{\langle\rangle \vdash \mathbf{0} : \emptyset} \qquad \frac{}{\langle\rangle \vdash l(\vec{x}) : n(\vec{x})} \qquad \frac{}{X : \tau \vdash X : \tau} \qquad \frac{\Gamma \vdash G : \tau \quad \Gamma' \vdash G' : \tau'}{\Gamma, \Gamma' \vdash G \mid G' : \tau \cup \tau'}$$

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

Intuitively, an empty graphs has no names. A hyperedge whose tentacles are linked to the names $\vec{x}$ exposes those names to a context. If a variable is typed by the set

[3] Idle closed names can be safely removed from the graph, see structural congruence in Fig. 1.

$$
\begin{aligned}
\Gamma \vdash G \mid \mathbf{0} &\equiv G \\
\Gamma \vdash G_1 \mid G_2 &\equiv G_2 \mid G_1 \\
\Gamma \vdash (G_1 \mid G_2) \mid G_3 &\equiv G_1 \mid (G_2 \mid G_3) \\
\Gamma \vdash \nu x.\mathbf{0} &\equiv \mathbf{0} \\
\Gamma \vdash \nu x.\nu y.G &\equiv \nu y.\nu x.G \\
\Gamma \vdash \nu x.(G_1 \mid G_2) &\equiv \nu x.G_1 \mid G_2 \text{ if } x \notin fn_\Gamma(G_2) \\
\Gamma \vdash \nu x.G &\equiv \nu y.(G\{y/x\}) \text{ if } y \notin fn_\Gamma(G) \\
\Gamma \vdash G[x \mapsto y] &\equiv (G\{z/x\})[z \mapsto y] \text{ if } z \notin fn_\Gamma(G) \\
\Gamma \vdash l(\vec{x})[y \mapsto z] &\equiv l(\vec{x})\{z/y\} \text{ if } \{y,z\} \cap \vec{y} \neq \emptyset \\
\Gamma \vdash G[x \mapsto y] &\equiv G \text{ if } x \notin fn_\Gamma(G) \text{ and } y \in fn_\Gamma(G) \\
\Gamma \vdash (G_1 \mid G_2)[x \mapsto y] &\equiv G_1[x \mapsto y] \mid G_2 \text{ if } x \notin fn_\Gamma(G_2) \\
\Gamma \vdash (G_1 \mid G_2)[x \mapsto y] &\equiv G_1[x \mapsto y] \mid G_2[x \mapsto y] \text{ if } x \in fn_\Gamma(G_1) \cap fn_\Gamma(G_2) \\
\Gamma \vdash G[x \mapsto y][w \mapsto z] &\equiv G[w \mapsto z][x \mapsto y] \text{ if } x \neq z, y \neq w \\
\Gamma \vdash \nu y.(G[x \mapsto y]) &\equiv \nu x.G \text{ if } y \notin fn_\Gamma(G) \\
\Gamma \vdash \nu z.(G[x \mapsto y]) &\equiv (\nu z.G)[x \mapsto y] \text{ if } z \notin \{x,y\}
\end{aligned}
$$

Fig. 1. Structural congruence for term judgments.

of names τ by an environment Γ, then it exposes such names. The names exposed by a composition of two subgraphs are the union of the names exposed by the two subgraphs. The restriction delete a name from the set of exposed names. Finally, the substitution $[w \mapsto z]$ (possibly) deletes the name w from the set of exposed names and adds z to this set.

Notice that in the last rule if $w \notin fn(G)$ then it effectively adds a new name z to G and to its type τ. Moreover, substitutions are important combined with variables: they should be used to rename variables' names.

Proposition 2.2 *Let $\Gamma \vdash G \equiv G'$, then $\Gamma \vdash G : \tau$ if and only if $\Gamma \vdash G' : \tau$.*

Now, we can introduce a notion of composition among term graphs.

Lemma 2.3 (substitution lemma) *The following rule is admissible.*

$$
\frac{\Gamma_1 \vdash G_1 : \tau_1 \ \ldots \ \Gamma_n \vdash G_n : \tau_n \quad X_1 : \tau_1, \ldots, X_n : \tau_n \vdash G : \tau}{\Gamma_1, \ldots, \Gamma_n \vdash G\{G_1/X_1, \ldots, G_n/X_n\} : \tau}
$$

In order to define a category for judgments we need a notion of tensor product, i.e., a way "to put n graphs side by side". Indeed, as shown in Lemma 2.3, if we want to substitute n variables in a graph G, we need a morphism representing G with "n holes" and another one that provides n graphs and it is composable with the former. To this end, we introduce a new operator ($\|$) and we extend the grammar (1) adding the definition of *agent-graphs* (A):

$$
\begin{aligned}
A &::= \epsilon \mid G \mid A \parallel A \mid \nu y.A \mid A[w \mapsto z] \\
G &::= \mathbf{0} \mid l(\vec{x}) \mid X \mid G|G
\end{aligned}
\tag{2}
$$

$$
\begin{aligned}
\Gamma \vdash G \mid \mathbf{0} &\equiv G \\
\Gamma \vdash G_1 \mid G_2 &\equiv G_2 \mid G_1 \\
\Gamma \vdash (G_1 \mid G_2) \mid G_3 &\equiv G_1 \mid (G_2 \mid G_3) \\
\Gamma \vdash \nu x.\mathbf{0} &\equiv \mathbf{0} \\
\Gamma \vdash (G_1 \mid G_2)[x \mapsto y] &\equiv G_1[x \mapsto y] \mid G_2 \text{ if } x \notin fn_\Gamma(G_2) \\
\Gamma \vdash (G_1 \mid G_2)[x \mapsto y] &\equiv G_1[x \mapsto y] \mid G_2[x \mapsto y] \text{ if } x \in fn_\Gamma(G_1) \cap fn_\Gamma(G_2) \\
\Gamma \vdash l(\vec{x})[y \mapsto z] &\equiv l(\vec{x})\{z/y\} \text{ if } \{y,z\} \cap \vec{x} \neq \emptyset \\
\Gamma \vdash \Gamma \vdash A \parallel \epsilon &\equiv A \\
\Gamma \vdash \epsilon \parallel A &\equiv A \\
\Gamma \vdash (A_1 \parallel A_2) \parallel A_3 &\equiv A_1 \parallel (A_2 \parallel A_3) \\
\Gamma \vdash \nu x.\epsilon &\equiv \epsilon \\
\Gamma \vdash \nu x.\nu y.A &\equiv \nu y.\nu x.A \\
\Gamma \vdash \nu x.(A_1 \parallel A_2) &\equiv \nu x.A_1 \parallel A_2 \text{ if } x \notin fn_\Gamma(A_2) \\
\Gamma \vdash \nu x.A &\equiv \nu y.(A\{y/x\}) \text{ if } y \notin fn_\Gamma(A) \\
\Gamma \vdash A[x \mapsto y] &\equiv (A\{z/x\})[z \mapsto y] \text{ if } z \notin fn_\Gamma(A) \\
\Gamma \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 \vdash A[x \mapsto y] &\equiv A \text{ if } x \notin fn_\Gamma(A) \text{ and } y \in fn_\Gamma(A) \\
\Gamma \vdash A[x \mapsto x] &\equiv A \text{ if } x \in fn_\Gamma(A) \\
\Gamma \vdash (A_1 \parallel A_2)[x \mapsto y] &\equiv A_1[x \mapsto y] \parallel A_2 \text{ if } x \notin fn_\Gamma(A_2) \\
\Gamma \vdash (A_1 \parallel A_2)[x \mapsto y] &\equiv A_1 \parallel A_2[x \mapsto y] \text{ if } x \notin fn_\Gamma(A_1) \\
\Gamma \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(A_1) \cap fn_\Gamma(A_2) \\
\Gamma \vdash \nu y.(A[x \mapsto y]) &\equiv \nu x.A \text{ if } y \notin fn_\Gamma(A) \\
\Gamma \vdash \nu z.(A[x \mapsto y]) &\equiv (\nu z.A)[x \mapsto y] \text{ if } z \notin \{x,y\}
\end{aligned}
$$

Fig. 2. Structural congruence for agent-graph judgment.

where ϵ represents the empty agent-graph. All the functions defined on terms $(v, fn, \dots)$ can be smoothly lifted to agent-graphs. Similarly to the case of terms, we consider agent-graphs up-to a structural congruence capturing graph isomorphisms (all the axioms are shown in Fig. 2).

Now, we extend the definition of graph types over agent-graphs:

$$\sigma ::= \varepsilon \mid \tau \mid \sigma\sigma$$

where juxtaposition denotes concatenation. In practice the new types are sequences of name sets. We extend the operations $\cup$ and $\setminus$ over sequences as follows:

$$(S \text{ is a set}) \qquad \tau_1 \dots \tau_n \cup S \stackrel{\triangle}{=} (\tau_1 \cup S) \dots (\tau_n \cup S) \qquad \tau_1 \dots \tau_n \setminus S \stackrel{\triangle}{=} (\tau_1 \setminus S) \dots (\tau_n \setminus S)$$

We extend the previous type inference rules for judgment on agent-graphs as

shown below.

$$\frac{}{\langle\rangle \vdash \mathbf{0} : \emptyset} \qquad \frac{}{\langle\rangle \vdash l(\vec{x}) : n(\vec{x})} \qquad \frac{}{X : \tau \vdash X : \tau} \qquad \frac{\Gamma \vdash G : \tau \quad \Gamma' \vdash G' : \tau'}{\Gamma, \Gamma' \vdash G \mid G' : \tau \cup \tau'}$$

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

$$\frac{}{\langle\rangle \vdash \epsilon : \varepsilon} \qquad \frac{\Gamma \vdash A : \sigma \quad \Gamma' \vdash A' : \sigma'}{\Gamma, \Gamma' \vdash A \parallel A' : \sigma\sigma'} \qquad \frac{\Gamma \vdash A : \sigma \quad \pi \text{ permutation}}{\pi(\Gamma) \vdash A : \sigma}$$

Intuitively, the empty agent-graph represent the empty agent and it is typed by the empty list. Notice that there is a difference between $\mathbf{0}$ and ϵ: $\mathbf{0}$ is the null process, and it is a non-empty agent. The rule for $\parallel$ are quite similar to the one for $|$, but in this case the two graphs lives into two difference locations, and hence the names can be treated in a different way, so the new type is defined by juxtaposing the two subtypes. Finally, last rule describes the ability to reorder the variables in the environment, it will be important in the definition of a category for agent-graphs.

Proposition 2.4 *Let $\Gamma \vdash A \equiv A'$, $\Gamma \vdash A : \sigma$ if and only if $\Gamma \vdash A' : \sigma$.*

Now, we are able to define a category for typed SHR graphs.

Definition 2.5 The category $\mathbf{H}$ of SHR graphs has graph types (σ) as objects, and judgments on agent-graphs as morphisms, that is, if $X_1 : \tau_1, \ldots, X_n : \tau_n \vdash A : \sigma$ then $(X_1, \ldots, X_n, A) : \tau_1 \ldots \tau_n \to \sigma$ is a morphism. Composition is defined in virtue of Lemma 2.3:

$$\frac{\Gamma \vdash G_1 \parallel \ldots \parallel G_n : \tau_1 \ldots \tau_n \quad X_1 : \tau_1, \ldots, X_n : \tau_n \vdash A : \sigma}{\Gamma \vdash A\{G_1/X_1, \ldots, G_n/X_n\} : \sigma}$$

Proposition 2.6 $(\mathbf{H}, \parallel, \epsilon)$ *is a strict symmetric monoidal category.*

3 Local Bigraphs

In this section we recall Milner's *local bigraphs*, a subclass of binding bigraphs.

Intuitively, a (local) bigraph represents an open system, so it has an inner and an outer interface to "interact" with subsystems and the surrounding environment. An example of a local bigraph is shown in Fig. 3. The *ordinals* of the interfaces describe the *roots* in the outer interface (that is, the various locations where the nodes live) and the *sites* in the inner interface (that is, the gray holes where other bigraphs can be inserted). On the other hand, the *names* in the interfaces describe the free links, that is end points where links from the outside world can be pasted, creating new links among nodes. We refer the reader to [7,8,12,13] for more details.

Let $\mathcal{K}$ be a *binding signature* of controls, 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 3.1 A *local interface* is a list $(X_0, \ldots, X_{n-1})$, where n is a finite ordinal (called *width*) and X_is are sets of names. X_i represents the names located at i.

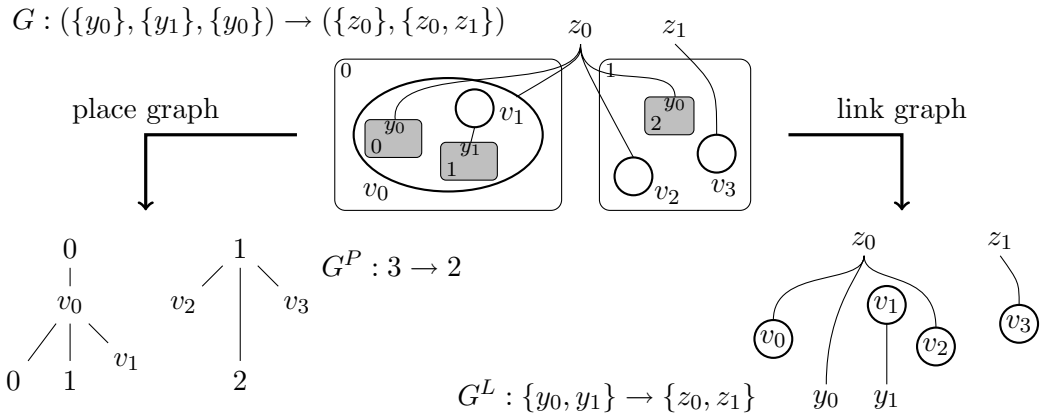

Fig. 3. An example of a local bigraph.

Notice that it is not necessary that the X_is are disjoint. We say that a name has multiple locations in I if it belongs to more than one name set of I.

Definition 3.2 A *local bigraph* $G\colon (\vec{X}) \to (\vec{Y})$ is defined as a (pure) bigraph $G^u\colon \langle |\vec{X}|, \bigcup \vec{X}\rangle \to \langle |\vec{Y}|, \bigcup \vec{Y}\rangle$ satisfying certain locality conditions.

G^u is defined by composing a *place graph* G^P, describing the nesting of nodes, and a *link graph* G^L, describing the (hyper)links among nodes.

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

where V, E are the sets of nodes and edges respectively, $ctrl\colon V \to \mathcal{K}$ 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,* $P = \sum_{v\in V} \pi_1(ar(ctrl(v)))$ 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), and $link\colon X \uplus P \to E \uplus B \uplus Y$ is the *link map.*

The locality conditions are the following:

(i) if a link is bound, then its inner names and ports must lie within the node that binds it;

(ii) 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 3.3 The category of local bigraphs over a signature $\mathcal{K}$ (**Lbg**($\mathcal{K}$)) has local interfaces as objects, and local bigraphs as morphisms.

Given two local bigraphs $G\colon (\vec{X}) \to (\vec{Y})$, $H\colon (\vec{Y}) \to (\vec{Z})$, the composition $H \circ G\colon (\vec{X}) \to (\vec{Z})$ is defined by composing their place and link graphs:

(i) the composition of $G^P\colon |\vec{X}| \to |\vec{Y}|$ and $H^P\colon |\vec{Y}| \to |\vec{Y}|$ is defined as

$$H^P \circ G^P = (V_G \uplus V_H, ctrl_G \uplus ctrl_H, (id_{V_G} \uplus prnt_H) \circ (prnt_G \uplus id_{V_H}))\colon |\vec{X}| \to |\vec{Y}|;$$

(ii) the composition of $G^L \colon \bigcup \vec{X} \to \bigcup \vec{Y}$ and $H^L \colon \bigcup \vec{Y} \to \bigcup \vec{Z}$ is defined as

$$H^L \circ G^L = \\ (V_G \uplus V_H, E_G \uplus E_H, ctrl_G \uplus ctrl_H, (id_{E_G} \uplus link_H) \circ (link_G \uplus id_{P_H})) \colon \bigcup \vec{X} \to \bigcup \vec{Z}.$$

Plg and **Lnk** will denote the categories of place and link graphs, respectively.

Intuitively, composition is performed in two steps. First, the place graph are composed by putting the roots of the "lower" bigraph inside the sites (i.e., holes) of the "upper" one, respecting the order given by the ordinal in the interface; then, the links are connected by sticking together the end parts of connections in the two link graphs, labelled with the same name in the common interface.

An important operation about (bi)graphs, is the *tensor product*. Intuitively, the tensor product of two bigraphs $G \colon (\vec{X}) \to (\vec{Y})$ and $H \colon (\vec{X'}) \to (\vec{Y'})$, is a bigraph $G \otimes H \colon (\vec{X}\vec{X'}) \to (\vec{Y}\vec{Y'})$ is defined when $X \cap X' = Y \cap Y' = \emptyset$ and it obtained by putting "side by side" G and H, without merging any root nor any name in the interfaces. Two useful variant of tensor product can be defined using tensor and composition: the *parallel product* $\|$, which merges shared names between two bigraphs or link graphs, and the *prime product* $|$, that moreover merges all roots in a single one. Due to lack of space, we cannot give a formal definition of these operations; we refer the reader to [13].

It is easy to check that composition and tensor preserve the locality conditions.

4 From SHR graphs to Local Bigraphs, and back

4.1 Encoding SHR graphs into Local Bigraphs

In this section we introduce an encoding functor from the SHR graph category **H** to the local bigraph category **Lbg**.

The first step is notice that the SHR ranked alphabet of labels ($\mathcal{L}$) and the bigraphical signature ($\mathcal{K}$) are quite similar. So, the idea is to choose as bigraphical signature exactly the alphabet of labels. In other words, our encoding translates the SHR hyperedges into nodes, and nodes (or names) into links (i.e., outer names and edges). Finally, note that every $l \in \mathcal{L}$ has only "exiting tentacles", hence the corresponding l in the bigraphical signature has no binding ports. Formally:

$$\mathcal{K}_{\mathcal{L}} \stackrel{\triangle}{=} \{l : 0 \to n \mid l_n \in \mathcal{L}\}.$$

Now we can define our encoding functor $[\![-]\!] : \mathbf{H}(\mathcal{L}) \to \mathbf{Lbg}(\mathcal{K}_{\mathcal{L}})$ as follows.

Objects:

$$[\![\varepsilon]\!] = ()$$
$$[\![\tau_1 \dots \tau_n]\!] = (\tau_1, \dots, \tau_n)$$

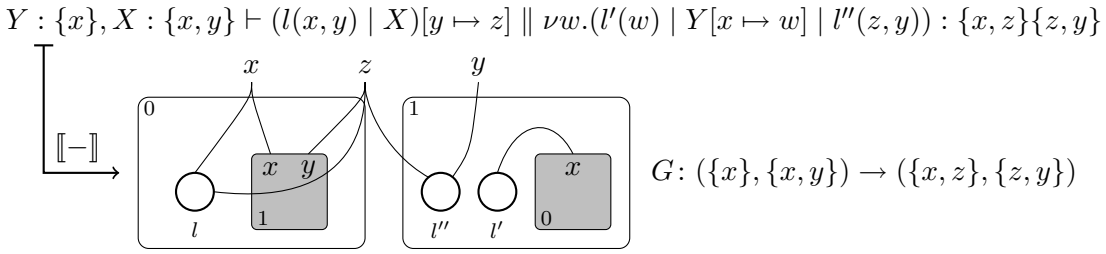

Fig. 4. An example of encoding an agent-graph into a local bigraph.

Morphisms:

$$
\begin{aligned}
[\![\langle\rangle \vdash \epsilon : \varepsilon]\!] &= id_{()} \\
[\![\Gamma, \Gamma' \vdash A \parallel A' : \sigma\sigma']\!] &= [\![\Gamma \vdash A : \sigma]\!] \parallel [\![\Gamma' \vdash A' : \sigma']\!] \\
[\![\pi(\Gamma) \vdash A : \sigma]\!] &= [\![\Gamma \vdash A : \sigma]\!] \circ \pi \\
[\![\Gamma \vdash \nu x.A : \sigma \setminus \{x\}]\!] &= /(x) \circ ([\![\Gamma \vdash A : \sigma]\!] \parallel \{x\}) \\
[\![\Gamma \vdash A[x \mapsto y] : (\sigma \setminus \{x\}) \cup \{y\}]\!] &= (y)/(x) \circ ([\![\Gamma \vdash A : \sigma]\!] \parallel \{x\}) \\
[\![\langle\rangle \vdash \mathbf{0} : \emptyset]\!] &= 1 \\
[\![\langle\rangle \vdash l(\vec{x}) : n(\vec{x})]\!] &= l_{\vec{x}} \\
[\![X : \tau \vdash X : \tau]\!] &= id_{(\tau)} \\
[\![\Gamma, \Gamma' \vdash G \mid G' : \tau \cup \tau']\!] &= [\![\Gamma \vdash G : \tau]\!] \mid [\![\Gamma' \vdash G' : \tau']\!]
\end{aligned}
$$

An example of encoding is given in Fig. 4. Basically, each variable of type τ is encoded as a site having τ local names; therefore, permutation of variables is permutation of sites. Restricted names are represented by bigraph edges, not accessible from the context. The empty graph $\mathbf{0}$ is represented by the empty root 1.

We can now prove the following result about the adequacy of the encoding.

Proposition 4.1 *Let A, A' be agent-graphs; then, for every Γ: $\Gamma \vdash A \equiv A'$ if and only if $[\![\Gamma \vdash A : \sigma]\!] = [\![\Gamma \vdash A' : \sigma]\!]$.*

4.2 Encoding Local Bigraphs into SHR graphs

In this section we provide a translation of local bigraphs into SHR graphs, supposing that every control in $\mathcal{K}$ is atomic (since SHR graphs do not have allow nestings). We take as alphabet of ranked labels the bigraphical signature. Formally:

$$\mathcal{L}_{\mathcal{K}} \stackrel{\triangle}{=} \{k_n \mid k : 0 \to n \in \mathcal{K}\}.$$

In order to simplify the translation procedure, we suppose that all local bigraphs are provided in a normal form, i.e., the *connected normal form* [8, Proposition 10.3]. The idea of this normal form is pushing all wirings inwards as far as we can: by using $\parallel$ and $\mid$ in place of $\otimes$, and also pushing closure inwards wherever possible.

Definition 4.2 Each local bigraph G, prime P and molecule M can be expressed

by an equation of the following form, named *connected normal form*:

$$\begin{aligned} G &= (/(Z) \parallel id_{(\vec{Y})}) \circ (P_0 \parallel \cdots \parallel P_{n-1}) \circ \pi \\ P &= (/(Z) \mid id_{(X)}) \circ (\delta \mid M_0 \mid \cdots \mid M_{k-1}) \circ \pi \\ M &= (/(Z) \mid id_{(W)}) \circ c \end{aligned}$$

where δ is a local substitution, π is a local permutation, and c is an atom.

Now we can define our encoding functor $(\!|-|\!) : \mathbf{Lbg}(\mathcal{K}) \to \mathbf{H}(\mathcal{L}_{\mathcal{K}})$ as follows.

Objects:

$$\begin{aligned} (\!|()|\!) &= \varepsilon \\ (\!|(X_1, \ldots, X_n)|\!) &= X_1 \ldots X_n \end{aligned}$$

Morphisms: for $G = (/(Z) \parallel id_{(\vec{Y})}) \circ (P_0 \parallel \cdots \parallel P_{n-1}) \circ \pi$, where $Z = \{z_1, \ldots, z_n\}$:

$$\begin{aligned} (\!|G : (\vec{X}) \to (\vec{Y})|\!) &= \vec{W} : (\!|\vec{X}|\!) \vdash \nu z_1 \ldots \nu z_n.((\!|p_0|\!) \parallel \cdots \parallel (\!|p_{|\vec{Y}|-1}|\!)) : (\!|\vec{Y}|\!) \\ \text{where } p_0 \parallel \ldots \parallel p_{|\vec{Y}|-1} &= (P_0 \parallel \cdots \parallel P_{|\vec{Y}|-1}) \circ \pi \circ (v^0_{X_0} \parallel \cdots \parallel v^{|\vec{X}|-1}_{X_{|\vec{X}|-1}}) \\ (\!|id_{()}|\!) &= \epsilon \\ (\!|(w)/(\{w_1, \ldots, w_n\}) \mid \delta|\!) &= [w_1 \mapsto w] \ldots [w_m \mapsto w](\!|\delta|\!) \\ (\!|p_i|\!) &= \nu z_1 \ldots \nu z_n.((\!|m_0|\!) \mid \cdots \mid (\!|m_{k_i}|\!))(\!|\delta|\!)) \quad 0 \le i < |\vec{Y}| \\ (\!|m_j|\!) &= \nu z_1 \ldots \nu z_n.(\!|c|\!) \quad 0 \le j \le k_i \\ (\!|v^i_{X_i}|\!) &= W_i \quad 0 \le i < |\vec{X}| \\ (\!|K_{\vec{x}}|\!) &= K(\vec{x}) \end{aligned}$$

where the nodes $v^0, \ldots, v^{|\vec{X}|-1}$ have special controls not present in $\mathcal{K}$, and they are used only to simplify the translation procedure. In practice, these variables 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 of the example encoding in Fig. 5.

Now we can state and prove the following result on the adequacy of the encoding.

Proposition 4.3 *Let G, G' be local bigraphs over an atomic signature; then, $G = G'$ if and only if there exists Γ such that $\Gamma \vdash (\!|G|\!) \equiv (\!|G'|\!)$.*

Moreover, we can establish nice connections between the two categories.

Proposition 4.4 (i) *Let G be a local bigraph, then $[\![(\!|G|\!)]\!] = G$.*
(ii) *Let $\Gamma \vdash A : \sigma$ be a graph judgment, then $\Gamma \vdash (\!|[\![\Gamma \vdash A : \sigma]\!]|\!) \equiv A$.*

5 SHR graphs as link graphs

In this section, we give a characterization of (possibly open) standard SHR graphs, as the link graphs underlying the local bigraphs over atomic signatures. This cat-

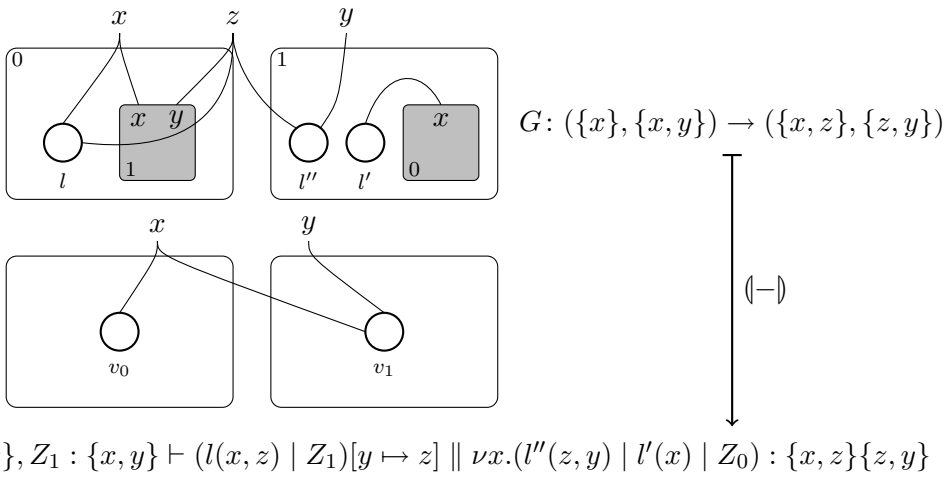

Fig. 5. An example of encoding a local bigraph into an agent-graph.

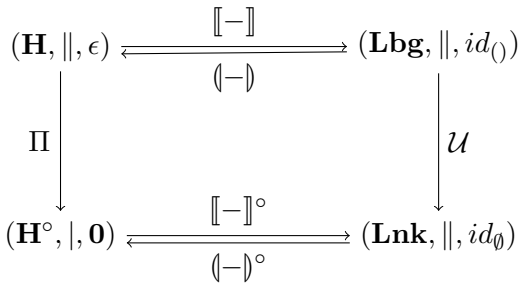

Fig. 6. Functors among the categories under consideration.

egory of link graphs can be characterized by the forgetful functor from bigraphs to link graphs. A summary diagram of the functors we are dealing with is in Fig. 6.

Definition 5.1 The forgetful functor $\mathcal{U} : \mathbf{Lbg}(\mathcal{K}) \to \mathbf{Lnk}(\mathcal{K})$ is defined as follows:

Objects: $\mathcal{U}((X_0, \ldots, X_{n-1})) = \bigcup_{i=0}^{n-1} X_i$

Morphisms: $\mathcal{U}((V, E, ctrl, prnt, link)) = (V, E, ctrl, link)$.

Following the encoding idea of the functor $(\!|-|\!)$ from bigraphs to SHR, we can map the link graphs inside $\mathbf{H}$. The idea is to consider a link graph as one-locality bigraphs, i.e., local bigraphs having only one root and zero or one holes. Clearly, the vice versa does not hold. We can recover the (almost) one to one correspondence by defining a sub-category of $\mathbf{H}$.

Definition 5.2 The category $\mathbf{H}^\circ$ has types (τ) as objects, and one variable judgments on terms as morphisms, i.e., if $X : \tau \vdash G : \tau'$ then $(X, G) : \tau \to \tau'$ is a morphism. Composition is defined as follows:

$$\frac{X : \tau \vdash G : \tau' \quad Y : \tau' \vdash G' : \tau''}{X : \tau \vdash G'\{G/Y\} : \tau''}$$

Proposition 5.3 $(\mathbf{H}^\circ, |, \mathbf{0})$ *is a strict symmetric monoidal category.*

For completeness we list below the definition of the two new functors. Their definition is a "simplification" of the more general functor seen before.

$[\![-]\!]^\circ : \mathbf{Lnk}(\mathcal{K}) \to \mathbf{H}^\circ(\mathcal{K})$:

Objects: $[\![\tau]\!]^\circ = \tau$

Morphisms:

$$\begin{aligned}
[\![\langle\rangle \vdash \mathbf{0} : \emptyset]\!]^\circ &= id_\emptyset \\
[\![\Gamma \vdash \nu x.A : \sigma \setminus \{x\}]\!]^\circ &= /(x) \circ ([\![\Gamma \vdash A : \sigma]\!]^\circ \parallel \{x\}) \\
[\![\Gamma \vdash A[x \mapsto y] : (\sigma \setminus \{x\}) \cup \{y\}]\!]^\circ &= (y)/(x) \circ ([\![\Gamma \vdash A : \sigma]\!]^\circ \parallel \{x\}) \\
[\![\langle\rangle \vdash l(\vec{x}) : n(\vec{x})]\!]^\circ &= l_{\vec{x}} \\
[\![X : \tau \vdash X : \tau]\!]^\circ &= id_\tau \\
[\![\Gamma, \Gamma' \vdash G \mid G' : \tau \cup \tau']\!]^\circ &= [\![\Gamma \vdash G : \tau]\!]^\circ \mid [\![\Gamma' \vdash G' : \tau']\!]^\circ
\end{aligned}$$

$(\!|-|\!)^\circ : \mathbf{H}^\circ(\mathcal{L} \to \mathbf{Lnk}(\mathcal{L}))$:

Objects: $(\!|X|\!)^\circ = X$

Morphisms:

$$\begin{aligned}
(\!|G : X \to Y|\!)^\circ &= Z : X \vdash (\!|/(Z)|\!)^\circ (\!|P \circ \pi \circ v_X|\!)^\circ : Y \\
(\!|id_\emptyset|\!)^\circ &= \mathbf{0} \\
(\!|/(\{z_1, \ldots, z_n\})|\!)^\circ &= \nu z_1. \cdots \nu z_n. \\
(\!|(w)/(\{w_1, \ldots, w_n\}) \mid \delta|\!)^\circ &= [w_1 \mapsto w] \ldots [w_m \mapsto w] (\!|\delta|\!)^\circ \\
(\!|p|\!)^\circ &= (\!|/(Z)|\!)^\circ ((\!|m_0|\!)^\circ \mid \cdots \mid (\!|m_k|\!)^\circ)(\!|\delta|\!)^\circ) \\
(\!|m_j|\!)^\circ &= (\!|/(Z)|\!)^\circ (\!|c|\!)^\circ \quad j \in \{0, \ldots, k\} \\
(\!|v_X|\!)^\circ &= Z \quad i \in \{0, \ldots, |\vec{X}| - 1\} \\
(\!|K_{\vec{x}}|\!)^\circ &= K(\vec{x})
\end{aligned}$$

All previous functors suggest a forgetful functor from the SHR graph category to the SHR graph sub-category, $\Pi : \mathbf{H} \to \mathbf{H}^\circ$, defined as follows:

Objects: $\Pi(\tau_1 \ldots \tau_n) = \bigcup_{i=1}^n \tau_i$

Morphisms:

$$\begin{aligned}
\Pi(\langle\rangle \vdash \epsilon : \varepsilon) &= id_\emptyset \\
\Pi(\Gamma, \Gamma' \vdash A \parallel A' : \sigma\sigma') &= \Pi(\Gamma \vdash A : \sigma) \mid \Pi(\Gamma' \vdash A' : \sigma') \\
\Pi(\pi(\Gamma) \vdash A : \sigma) &= \Pi(\Gamma \vdash A : \sigma) \\
\Pi(\Gamma \vdash \nu x.A : \sigma \setminus \{x\}) &= /(x) \circ (\Pi(\Gamma \vdash A : \sigma) \mid \{x\}) \\
\Pi(\Gamma \vdash A[x \mapsto y] : (\sigma \setminus \{x\}) \cup \{y\}) &= (y)/(x) \circ (\Pi(\Gamma \vdash A : \sigma) \mid \{x\}) \\
\Pi(\langle\rangle \vdash \mathbf{0} : \emptyset) &= id_\emptyset \\
\Pi(\langle\rangle \vdash l(\vec{x}) : n(\vec{x})) &= l_{\vec{x}} \\
\Pi(X : \tau \vdash X : \tau) &= id_\tau \\
\Pi(\Gamma, \Gamma' \vdash G \mid G' : \tau \cup \tau) &= \Pi(\Gamma \vdash G : \tau) \mid \Pi(\Gamma' \vdash G' : \tau')
\end{aligned}$$

In practice the above functor merges all the separate graph-agents graphs into a unique bigger graphs. In other words, it translates a $\parallel$ operator with the $\mid$ one.

As a consequence of the definitions of the functors $[\![-]\!]$, $\mathcal{U}$ and $(\!|-|\!)^\circ$, we can prove the following result.

Proposition 5.4 *Let* $\Gamma \vdash A : \sigma$ *be a graph judgment; then,* $\Gamma \vdash (\!|\mathcal{U}([\![\Gamma \vdash A : \sigma]\!])|\!)^\circ \equiv \Pi(\Gamma \vdash A : \sigma)$.

Proposition 5.5 (i) *Let* H *be a link graph, then* $[\![(\!|H|\!)^\circ]\!]^\circ = H$.

(ii) *Let* $X : \tau \vdash G : \tau'$ *be a term judgment, then* $X : \tau \vdash (\!|[\![X : \tau \vdash G : \tau']\!]^\circ|\!)^\circ \equiv G$.

6 From ADR graphs to local bigraphs, and back

In this section we generalize further the syntax of SHR agents to get a language capable to express any local bigraph, not only those on atomic controls. To this end, we have to allow nesting of graphs within edge labels. Such kind of graph grammars have been already analyzed in the case of *Architectural Design Rewriting* (ADR) graphs [2], where a notion of nesting among edges/graphs is considered.

The idea is to add to $\mathcal{L}$ a new set of egde labels, which can contain graphs. We call the old labels *atomic* (written l_n, where n is the exit-rank of l), and the new ones *non-atomic* (written $L_n(m)$, where n is the exit-rank and m is the in-rank of L). Then, the syntax in (2) can be extended as follows to define layered-graphs:

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

where the graph $L(\vec{y})[G\backslash\vec{x}]$ describes a non-atomic edge L having exiting tentacles linked to the names in $\vec{y}$, and also has a subgraph G inside itself. Finally, $\backslash\vec{x}$ means that the names $\vec{x}$ coming from the graph G are stopped by the edge itself by linking them to "incoming" tentacles of L.

All the functions defined on terms and agent-graphs $(v, fn, \dots)$ can be smoothly lifted to the layered-graphs. Similarly to the case of terms and agent-graphs, we consider those graphs up-to a structural congruence capturing graph isomorphisms; to this end, we extend the rules of Figure 2 with the followings:

$$\Gamma \vdash L(\vec{y})[G\backslash\vec{x}] \equiv L(\vec{y})[(G\{z/x\})\backslash(\vec{x}\{z/x\})] \text{ if } x \in \vec{x} \text{ and } z \notin \vec{x} \cup fn_\Gamma(G)$$
$$\Gamma \vdash L(\vec{y})[G\backslash\vec{x}][y \mapsto z] \equiv L(\vec{y}\{z/y\})[G[y \mapsto z]\backslash\vec{x}] \text{ if } \{y, z\} \cap \vec{y} \neq \emptyset \text{ and } y, z \notin \vec{x}$$
$$\Gamma \vdash \nu z.L(\vec{y})[G\backslash\vec{x}] \equiv L(\vec{y})[(\nu z.G)\backslash\vec{x}] \text{ if } z \notin \vec{y} \cup \vec{x}$$

The set of types considered here is the same of agent-graphs, i.e., types are sequences of name sets. Now we must extend the set of type inference rules defined for agent-graphs to be used for layered-graphs by adding the following new rule:

$$\frac{\Gamma \vdash G : \tau \quad n(\vec{x}) \subseteq \tau}{\Gamma \vdash L(\vec{y})[G\backslash\vec{x}] : (\tau \setminus n(\vec{x})) \cup n(\vec{y})}$$

This new rule allows to insert a graphs inside a non-atomic edge; the unique requirement is that the graph must correspond to a graph term, i.e., its type is a single set of names. The new type is constructed by deleting the names $\vec{x}$ of G stopped by L and by adding the names $\vec{y}$ defining the exiting tentacles of L.

Notice that, composition remains unchanged as well as the definition of the category. We call this new category *layered-graph category* (**L**).

Proposition 6.1 $(\mathbf{L}, \|, \epsilon)$ *is a strict symmetric monoidal category.*

The translations from ADR graphs to local bigraphs and vice versa is quite similar to the one shown in sections 4.1 and 4.2, more precisely they are a straightforward extension of them.

From ADR graphs to local bigraphs. The signature $\mathcal{K}^{\Box}_{\mathcal{L}}$ for the category of local bigraphs is defined from the ranked set of labels $\mathcal{L}$ as follows:

$$\mathcal{K}^{\Box}_{\mathcal{L}} \triangleq \{l : 0 \to n \mid l_n \in \mathcal{L}\} \cup \{L : m \to n \mid L_n(m) \in \mathcal{L}\}$$

where the ls are atomics and the Ls are non-atomic [4] .

We define our encoding functor $[\![-]\!]^{\Box} : \mathbf{L}(\mathcal{L}) \to \mathbf{Lbg}(\mathcal{K}^{\Box}_{\mathcal{L}})$ as the functor $[\![-]\!]$ by adding the following case.

$$[\![\Gamma \vdash L(\vec{y})[G\backslash\vec{x}] : (\tau \setminus n(\vec{x})) \cup n(\vec{y})]\!]^{\Box} = L_{\vec{y}(\vec{x})} \circ [\![\Gamma \vdash G : \tau]\!]^{\Box}$$

From local bigraphs to ADR graphs. The ranked set of labels $\mathcal{L}^{\Box}_{\mathcal{K}}$ for the layered graphs is defined from the signature $\mathcal{K}$ of local bigraphs as follows:

$$\mathcal{L}^{\Box}_{\mathcal{K}} \triangleq \{k_n \mid k : 0 \to n \in \mathcal{K}\} \cup \{K_n(m) \mid K : m \to n \in \mathcal{K}\}$$

where the ks are atomics and the Ks are non-atomic.

We define our encoding functor $(\!|-|\!)^{\Box} : \mathbf{Lbg}(\mathcal{K}) \to \mathbf{L}(\mathcal{L}^{\Box}_{\mathcal{K}})$ as the functor $(\!|-|\!)$ by adding the following case.

$$(\!|K_{\vec{y}(\vec{x})} \circ p|\!)^{\Box} = K(\vec{y})[(\!|p|\!)^{\Box}\backslash\vec{x}]$$

Now we can state and prove the following result on the encodings.

Proposition 6.2 *Let G, G' be local bigraphs; then, $G = G'$ if and only if there exists Γ such that $\Gamma \vdash (\!|G|\!)^{\Box} \equiv (\!|G'|\!)^{\Box}$.*

Proposition 6.3 (i) *Let G be a local bigraph, then $[\![(\!|G|\!)^{\Box}]\!]^{\Box} = G$.*

(ii) *Let $\Gamma \vdash A : \sigma$ be a graph judgment, then $\Gamma \vdash (\!|[\![\Gamma \vdash A : \sigma]\!]^{\Box}|\!)^{\Box} \equiv A$.*

7 Conclusions

In this paper we have first investigated the connection between *Synchronized Hyperedge Replacement* graphs and *(local) bigraphs*, two different graphical frameworks for concurrency. We have given a mild generalization of SHR graph grammar, allowing for "holes" and localities, which turns out to correspond exactly to local bigraphs over atomic controls. Moreover, the link graphs underlying these local bigraphs correspond to usual SHR graphs. Then, we have extended this approach to *Architectural Design Rewriting* graphs, which turn out to correspond to general local bigraphs. Therefore, SHR graph grammar is a language for local bigraphs over

[4] We can further distinguish the non-atomic controls in *active* of *passive*, but in the present case it is not necessary, since we do not deal with the bigraph semantics.

atomic signatures, and ADR graph grammar is a language for local bigraphs; these languages can be used in place of the more complex bigraph algebra.

A possible future work is to take advantage of the rich theory provided by bigraphical reactive systems [7], in order to obtain interesting results about SHR and ADR. In particular, local bigraphs allow to synthesise labelled transition systems out of rewriting rules, via the so-called *idem-pushout* construction [9]; 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 local 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. The authors thank Emilio Tuosto and Ivan Lanese for useful discussions about SHR.

References

[1] Lars Birkedal, Søren Debois, Ebbe Elsborg, Thomas Hildebrandt, and Henning Niss. Bigraphical models of context-aware systems. In Luca Aceto and Anna Ingólfsdóttir, editors, *Proc. FoSSaCS*, volume 3921 of *Lecture Notes in Computer Science*, pages 187–201. Springer, 2006.

[2] Roberto Bruni, Alberto Lluch-Lafuente, Ugo Montanari, and Emilio Tuosto. Service oriented architectural design. In Gilles Barthe and Cédric Fournet, editors, *Proc. TGC*, volume 4912 of *Lecture Notes in Computer Science*, pages 186–203. Springer, 2007.

[3] Mikkel Bundgaard, Arne J. Glenstrup, Thomas T. Hildebrandt, Espen Højsgaard, and Henning Niss. Formalizing higher-order mobile embedded business processes with binding bigraphs. In Doug Lea and Gianluigi Zavattaro, editors, *COORDINATION*, volume 5052 of *Lecture Notes in Computer Science*, pages 83–99. Springer, 2008.

[4] Pierpaolo Degano and Ugo Montanari. A model for distributed systems based on graph rewriting. *J. ACM*, 34(2):411–449, 1987.

[5] Gian Luigi Ferrari, Ugo Montanari, and Emilio Tuosto. A lts semantics of ambients via graph synchronization with mobility. In Antonio Restivo, Simona Ronchi Della Rocca, and Luca Roversi, editors, *Proc. ICTCS*, volume 2202 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2001.

[6] Davide Grohmann and Marino Miculan. Reactive systems over directed bigraphs. In L. Caires and V.T. Vasconcelos, editors, *Proc. CONCUR 2007*, volume 4703 of *Lecture Notes in Computer Science*, pages 380–394. Springer-Verlag, 2007.

[7] Ole Høgh Jensen and Robin Milner. Bigraphs and transitions. In *Proc. POPL*, pages 38–49, 2003.

[8] Ole Høgh Jensen and Robin Milner. Bigraphs and mobile processes (revised). Technical report UCAM-CL-TR-580, Computer Laboratory, University of Cambridge, 2004.

[9] James J. Leifer and Robin Milner. Deriving bisimulation congruences for reactive systems. In Catuscia Palamidessi, editor, *Proc. CONCUR*, volume 1877 of *Lecture Notes in Computer Science*, pages 243–258. Springer, 2000.

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

[11] Robin Milner. Bigraphical reactive systems. In Kim Guldstrand Larsen and Mogens Nielsen, editors, *Proc. 12th CONCUR*, volume 2154 of *Lecture Notes in Computer Science*, pages 16–35. Springer, 2001.

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

[13] Robin Milner. Local bigraphs and confluence: Two conjectures. In *Proc. EXPRESS 2006*, volume 175(3) of *Electronic Notes in Theoretical Computer Science*, pages 65–73. Elsevier, 2007.