Executing bigraphical reactive systems

Amal Gassara, Ismael Bouassida Rodriguez, Mohamed Jmaiel, Khalil Drira

Executing Bigraphical Reactive Systems

Amal Gassara

ReDCAD laboratory, University of Sfax, National School of Engineers of Sfax, B.P. 1173, 3038 Sfax, Tunisia
amal.gassara@redcad.org

Ismael Bouassida Rodriguez

ReDCAD laboratory, University of Sfax, National School of Engineers of Sfax, B.P. 1173, 3038 Sfax, Tunisia

Mohamed Jmaiel

Digital Research Center of Sfax, B.P. 275, Sakiet Ezzit, 3021 Sfax, Tunisia

Khalil Drira

CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Univ de Toulouse, LAAS, F-31400 Toulouse, France

September 23, 2018

Abstract

In order to enable experimentations and simulations of bigraphs, we need an implementation of their dynamic. In this paper, we tackle the matching issue of this task. We present a solution based on an investigation on graph matching. We propose to simulate a bigraphical reactive system (i.e., bigraphs with a set of reaction rules that allow their rewriting) with a graph transformation system. First, we translate a bigraph to a ranked graph. This translation is ensured by defining a faithful functor that allows to move from the bigraph category to the ranked graph category. Then, we show that reaction rules can be simulated with graph rules. Hence, we provide a formal basis allowing to execute bigraph transformations by simulating their translation aiming to use well-established and efficient graph transformation tools.

I. Introduction

The theory of Bigraphical Reactive Systems (BRSs) has been developed by Milner [1] as a formalism for describing and analyzing mobile computation and pervasive systems. BRS is a graphical model in which bigraphs can be reconfigured using reaction rules. The bigraph consists of a place graph that models the hierarchical structure of a system entities and the link graph that describes their connection structure.

BRSs have been successfully applied in encompassing existing theories for concurrency and mobility (e.g. CCS, Petri Nets and Pi-calculus, etc), as well as in capturing software architectures and modeling applications for context-aware systems and ubiquitous computing environments.

Therefore, it is very important to have an implementation of the dynamic of bigraphs to enable experimentations. The main challenge of implementing the dynamic of bigraphs is the matching problem. In fact, the latter is a computational task that determines for a given bigraph B and a reaction rule R whether and how the reaction rule can be applied to rewrite

the bigraph B (i.e., determining whether R occurs in B). However, it is not immediately clear how to implement a framework that executes such reaction rules.

In the literature, the first research activity addressing bigraph matching is presented in [2, 3]. This work proposes a matching system over bigraph terms for inferring legal matches of BRSs. This system is based on inference rules that are defined formally by an inductive characterisation of occurrence. Based on this matching system, a tool called BPL Tool [3, 4] has been implemented. However, they implemented normalisation, renaming, and regularisation steps before matching. These steps could increase the computation time.

Furthermore, the theory of BRS is closely related to Graph Transformation Systems (GTSs) [5, 6]. Considering the exhaustiveness of studies on graph transformations, it is natural to ask whether we could apply graph matching algorithms on bigraphs. As an alternative to implementing matching for bigraphs, we could try to formalize BRSs as GTSs. By this way, we can benefit from existing tools and techniques developed for graph transformations. This is very interesting especially that these tools are well mastered on the one hand and very efficient and powerful on the other. Consequentially, we initiate an investigation of how to simulate a BRS with a GTS. This would require us to establish a method for ensuring the validity of such simulation.

This paper presents our solution for executing BRSs that is based on an investigation of GTSs. We propose a formal basis allowing to translate bigraph transformations into graph transformations. First, we encode a bigraph into a ranked graph. This encoding is ensured by defining a faithful functor that allows to move from the bigraph category to the ranked graph category. Then, we show that reaction rules can be simulated by graph rules. Hence, we provide a formal basis allowing to execute bigraph transformations by simulating their encoding in order to use well-established and efficient graph transformation tools.

The rest of this paper is organized as follows. Section II introduces some research activities dealing with bigraph matching. In Section III, we present an overview of BRSs and GTSs. In Section IV, we explain our proposed approach for executing BRSs and we describe its steps. Then, Section V presents BiGMTE, our implementation of our proposed solution for executing BRSs. Finally, Section VI concludes this paper and gives some directions for future work.

II. Related work

We discuss in this section some research activities dealing with bigraph matching that consists in finding the occurrences of a *Redex* (guest) inside an agent (host). Computing bigraph matchings has been proved to be NP-complete, like the subgraph isomorphism problem [7]. In the literature, some algorithms were proposed. Sevegnani et al [8] presented a sound and complete algorithm for finding matchings in place graphs for bigraphs with sharing. It was shown how the matching problem can be mapped into a SAT (Satisfiability) instance since a SAT based solution could provide an efficient solution for such problems. However, this work deals with a special case of bigraphs (bigraph with sharing). Also, they presented only a procedure for the matching of place graphs. They do not yet propose an approach for link graphs. Miculan et al. [9] considered the embedding problem as a constraint satisfaction problem (CSP). This algorithm, that they proved to be sound and complete, was implemented in LibBig [10], a library for manipulating bigraphical reactive systems. Compared to our proposed tool, LibBig is slower than BiGMTE in the case of complex *Redex*. In fact, by increasing the complexity of the *Redex*, the number of constraints increases, too, and that slows down the resolution of the constraint satisfaction problem.

Differently from these two algorithms, Mansutti et al. [11] introduced an algorithm for computing bigraphical embeddings in distributed settings where bigraphs are spread

across several cooperating processes. This decentralized algorithm does not require a complete view of the host bigraph, but retains the fundamental property of computing every possible embedding for the given host. This hinders the scalability of BRS execution tools, especially on devices with low resources like embedded ones.

An implementation of bigraph matching [2, 3] was given in the BPL Tool [3, 4] that is based on a term-based representation of agents and rules, in the spirit of term rewriting systems. This work proposed a matching system over bigraph terms for inferring legal matches of BRSs. This system is based on inference rules that are defined formally by an inductive characterisation of occurrence. However, the implementation of matching is not very fast [4], due to the fact that it is derived directly from the inductive characterization of matching that is based on the binding discrete normal form. Authors implemented normalisation, renaming, regularisation and matching. These steps could increase the computation time.

Although the standard way to define a BRS matching is by algebraic manipulation of terms (using equational reasoning) and this is proven complete, it is obviously very costly. It was therefore thought that considering bigraphs as a special class of graphs would help solving the efficiency problem by considering BRS mathching as a special kind of graph embedding. Actually, BRSs are related to GTSs (See [6] for a comprehensive overview of GTSs). In fact, a GTS is based upon the double pushout (DPO) construction originated by Ehrig [5]. This approach relies upon the treatment of graphs as objects in a category whose arrows are embedding. This contrasts with bigraphical approach, where bigraphs are the arrows in a category whose objects are interfaces.

But, there are connections between these two approaches. Ehrig has investigated these links [12], after discussions with Milner. One of these connections is previously explored by Gadducci et al [13]. In fact, graphs as arrows are obtained via cospans $I \rightarrow G \leftarrow J$ of graphs as objects, where the interfaces I and J are discrete graphs. In our work, we rely on this connection in order to investigate translating a BRS into a GTS. A second connection goes on the other way. Graphs as objects are obtained in a coslice category of a category where graphs are arrows. This connection was proposed by Milner in [14], in the context of link graphs that are a constituent of bigraphs. It was shown that, for link graphs, the coslice category is isomorphic to the natural category of embeddings (as arrows) between link graphs.

In the direction of relating graphs to bigraphs, some promising steps were taken. Bruni et al [15] compared bigraphs with gs-graphs, i.e., provided as arrows of gs-monoidal theories. These theories are symmetric monoidal categories equipped with two transformations, satisfying a few properties. In this research activity, authors showed that gs-graphs can be proved essentially equivalent to bigraphs, with minor differences at the interface level. This work looks close to our proposal. However, it did not define a reference theory of rewriting bigraphs and gs-graphs.

Moreover, Damgaard [16] initiated an investigation of how to compile a BRS to a GTS and he is working on such an investigation. More specifically, he is investigating how one may faithfully encode a BRS into a typed, attributed GTS (translating a BRS simply by translating each of its constituents: an agent, a set of reaction rules and a signature). To the best of our knowledge, no further work on this investigation has been conducted. Along that line of research, our work investigates how BRS can be encoded as a particular type of graph transformation.

III. Preliminaries

To understand how bigraphs relate to graphs, we need, first, to understand these two graphical models. For this, we give, in this section, an overview of bigraphs, reaction rules, ranked graphs and graph rules. Readers fa-

miliar with these models can skip this section.

i. Bigraphs

The theory of bigraphs [1] was proposed by Milner and it is used to model the structure of a system. We use Figure 1(a) to introduce bigraphs informally. A bigraph consists of hyperedges and nodes that can be nested and that have ports. Each hyperedge can connect many ports on different nodes (for example, v_0 , v_1 and v_2 are joined by e_1). A bigraph combines two graphical structures -a *place graph* and a *link graph*- based on the same node set, hence the term bigraph. Figure 1(b) depicts the corresponding *place* and *link graphs* of the bigraph G.

Place graph: The *place graph* is a hierarchical tree that describes the locality of the nodes. In this graph, branches establish the parent (nesting) relationship of nodes in the bigraph. Trees are rooted by regions (or roots) represented by dashed rectangle in G. Within the *place graph*, in addition to nodes and regions, there can also be sites, represented as grey rectangles in G. A site is a hole that can host new nodes.

Link graph: The *link graph* is a hypergraph that describes the connectivity of nodes. Within this graph, there can be outer names like y_0 , y_1 and y_2 in G (cf. Figure 1(a)) and inner names like x_0 and x_1 in G (cf. Figure 1(a)) represented as open links. These names give bigraphs the possibility to be composed by joining the inner names of one bigraph with the corresponding outer names of another bigraph.

Control and signature: Each node in the bigraph is assigned a control. Controls (K and M in the case of G) indicate the node type and its ports' number through the arity. The set of controls forms the signature.

Interfaces: Bigraphs can be built through their interfaces. We distinguish two types of interface: inner interface and outer interface. The inner interface is defined by $I = \langle m, X \rangle$, where m is the number of sites in the bigraph and X the set of its inner names. The outer interface is defined by $J = \langle n, Y \rangle$, where n is the number of roots and Y is the set of outer names. In the example of Figure 1(a), the inner interface $I = \langle 2, \{x_0, x_1\} \rangle$ and the outer interface $J = \langle 2, \{y_0, y_1, y_2\} \rangle$.

Bigraph composition: The composition of two bigraphs is defined by matching the inner interface of the first graph with the outer interface of the second (i.e., filling the sites (holes) of the first with regions of the second and merging the inner names of the first with the outer names of the second).

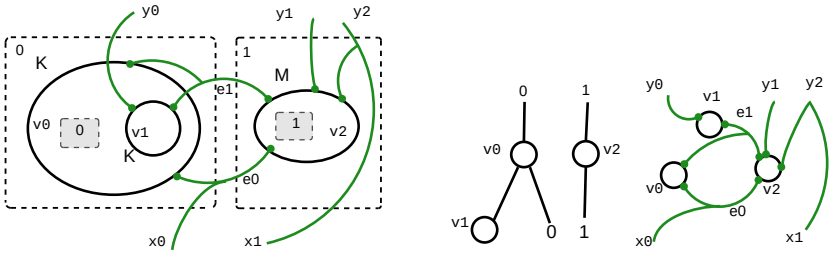
Definition III.1 (Bigraph)**.** Formally a bigraph is defined as a 5-tuple of the form:

$$(V, E, ctrl, prnt, link) : \langle m, X \rangle \rightarrow \langle n, Y \rangle \; where$$

- $V \in \mathcal{V}$ is a set of nodes ($\mathcal{V}$ node-identifiers)
- $E \in \mathcal{E}$ is a set of hyperedges ($\mathcal{E}$ hyperedge-identifiers)
- $ctrl : V \rightarrow \mathcal{K}$ is a control map that assigns controls to nodes ($\mathcal{K}$ is the signature)
- $prnt : m \uplus V \rightarrow V \uplus n$ is the parent map and defines the nested place structure (m is the number of sites and n is the number of the roots). The parent map is acyclic (i.e., $prnt^k(v) \neq v \; for \; all \; k > 0 \; and \; v \in V$)
- $link : X \uplus P \rightarrow E \uplus Y$ is the link map and defines the link structure. X is the set of inner names, Y is the set of outer names and P defines the set of ports of the bigraph and is formalized as $P = \{(v, i) \mid i \in \{0, 1, .., arity(ctrl(v))\}\}$.

ii. Reaction rule

Bigraphs are associated with reaction rules to form bigraphical reactive systems (BRSs) that can be applied to rewrite bigraphs. Each reaction rule consists of two bigraphs: a *Redex R* (the pattern to be changed) and a *Reactum R'* (the changed pattern). The application of the rule consists of identifying the image of R in a bigraph and replacing it by the corresponding R'.



(a) A Bigraph G (b) *Place* and *link* graphs

Figure 1: *A Bigraph and its graphs [1]*

iii. Ranked Graphs

In this paper, we deal with ranked graphs. They are graphs having interfaces through which they can be connected.

Definition III.2 (A ranked graph)**.** A ranked graph [13] is a triple of the form:

$$g = \langle r, d, v \rangle : i \to j \; where$$

- $d = \langle N, E, s, t \rangle$ is a concrete graph where N is a set of nodes, E is a set of edges, $s, t : E \to N$ are the source and the target functions. In this concrete graph d, nodes and edges have identifiers.
- i, j are natural numbers which represent the interfaces of g.
- $r : j \to N$ is a function called the root mapping.
- $v : i \to N$ is a function called the variable mapping.

We have enhanced ranked graphs with label functions $l_v : N \to L_N$ where L_N is fixed label alphabets for nodes. For a graph G, the components i and j represent discrete interfaces, through which graphs can be equipped with a compositional structure.

Ranked graphs composition: The composition of two ranked graphs is obtained by gluing the variables of the first one with the roots of the second one (i.e., matching them and then eliminating them). It is defined only if their number is equal.

iv. Graph rule

A graph rule (or production) p is essentially a pair of graphs, a left-hand side L and a right-hand side R. Rules are given with a common interface I (i.e., the intersection of L and R). Applying a rule on a graph G consists in finding L in G, removing $L \setminus I$ from G, and then adding $R \setminus I$, resulting in a graph H.

IV. Our proposed approach for executing BRSs

We propose a solution for executing BRSs based on an investigation of how to simulate a BRS with a GTS [17]. In other words, we simulate the application of a reaction rule on a bigraph with the application of a graph rule on a graph. Our solution follows three steps:

- Step 1: Encoding a bigraph into a graph.
- Step 2: Encoding a reaction rule into a graph rule.
- Step 3: Simulating the application of a reaction rule on a bigraph by applying the encoded graph rule on the encoded graph.
- Step 4: Encoding the resulted graph in the previous step into a bigraph.

This solution requires us to establish a formal basis for ensuring the validity of such encoding.

i. Step 1: Encoding Bigraphs to Ranked graphs

The main difference between bigraphs and ranked graphs lies in the nesting and the linking structure of bigraphs.

We define the nesting structure of bigraphs through the node identifiers of graphs. In fact, the hierarchy of places (roots and nodes) is built in the graph by exploiting graph node identifiers. For instance, in Figure 2, the parent of the node v_0 is the root 0 (v_0 is nested in 0). Its image in the graph G is the node having the identifier v_0:0.

Furthermore, the linking structure of bigraphs is represented in the graph by defining two types of nodes: *place nodes* that represent bigraph places, and *link nodes* that represent bigraph hyperedges.

Therefore, we can depict the relation between bigraphs and graphs by looking at Figure 2:

- Bigraph places and their hierarchy are represented in the graph by *place nodes* and their identifiers (black nodes).
- Bigraph hyperedges are represented in the graph by *link nodes* (the green nodes). For example, the hyperedge e_1 in the bigraph, connecting v_2 and v_3 nodes, is represented in the graph with the green node e_1 to which are connected v_2:0 and v_3:1 nodes.
- The inner and the outer interfaces of the bigraph correspond to numbers on the left and on the right, respectively, in the graph. The dashed lines represent which nodes are exported to the interfaces.

Categorically, bigraphs and their morphisms form an s-category (see Definition in Appendix A) denoted $\mathcal{BG}$ that has as objects inner and outer interfaces, and as arrows bigraphs (Definition in III.1). Similar to bigraphs, ranked graphs are presented as arrows between two interfaces i and j, forming a category denoted $\mathcal{DG}$ (Definition iii).

In order to ensure formally encoding bigraphs into ranked graphs, we shall construct a functor (Definition IV.1) that allows to move from one category to another, preserving its structure.

Definition IV.1 (functor [1])**.** A functor $F : \mathcal{C} \to \mathcal{D}$ between two categories $\mathcal{C}$ and $\mathcal{D}$ is a function taking objects to objects and arrows to arrows; it takes the arrow $f : A \to B$ in $\mathcal{C}$, to the arrow $F(f) : F(A) \to F(B)$ in $\mathcal{D}$, such that:

- F preserves identity: $F(Id_A) = Id_{F(A)}$ for every object A from $\mathcal{C}$,
- F preserves composition: $F(g \circ f) = F(g) \circ F(f)$ for all arrows $f : A \to B$ and $g : B \to C$

Since bigraphs represent the s-category $\mathcal{BG}$, we demonstrate that the $\mathcal{DG}$ category can be casted as an s-category (see the proof in **??**) in order to be able to construct an categorical functor that allows to move from the s-category $\mathcal{BG}$ of bigraphs to the s-category $\mathcal{DG}$ of ranked graphs.

We construct a functor named $F_{sim} : \mathcal{BG} \to \mathcal{DG}$ that allows to move from the s-category $\mathcal{BG}$ of bigraphs to the s-category $\mathcal{DG}$ of ranked graphs. This functor associates to each inner interface I from $\mathcal{BG}$, an interface i from $\mathcal{DG}$, it associates to each outer interface J from $\mathcal{BG}$, an interface j from $\mathcal{DG}$ and it associates to each morphism B (i.e., a bigraph) from $\mathcal{BG}$, a morphism G (i.e., a ranked graph) from $\mathcal{DG}$. Hence, for each Bigraph $B : I \to J$, F_{sim} associates a Graph $G : i \to j$ such that:

$$\begin{cases} F_{sim}(I) = i \\ F_{sim}(J) = j \\ F_{sim}(B) = G \end{cases}$$

We define the simulating functor F_{sim} on the objects and on the morphisms of the two categories $\mathcal{BG}$ and $\mathcal{DG}$.

i.1 Defining F_{sim} on objects

We define F_{sim} as an injective function between the objects (i.e., interfaces) of the two models. Actually, the main difference between them lies in the way of representing these interfaces.

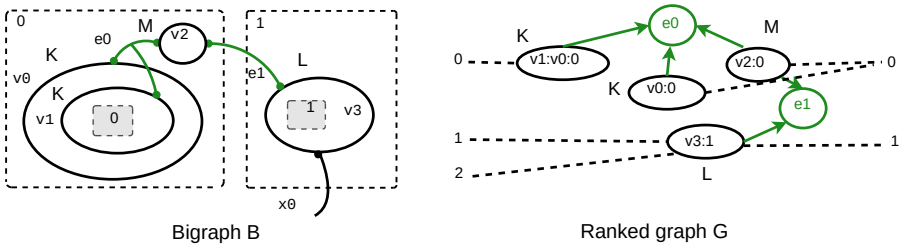


Figure 2: *Encoding a bigraph into a ranked graph*

For a bigraph, an interface is a pair composed of an ordinal, representing roots and sites, and by a set of names, representing inner and outer names, whereas, a graph interface is a list of ordered numbers.

Hence, given a bigraphical interface $\langle m, X\rangle$, F_{sim} associates a graph interface represented as a list of ordered numbers, starting from 0, with exactly $m + |X|$ elements. This list is regarded as a discrete graph. Every $x \in m$ (a root or a site) is encoded in the graph by a *place node* and every name in X (an inner name or an outer name) is encoded in the graph by a *link node*.

So, for each object $I = \langle m, X\rangle$ from $\mathcal{BG}$ (i.e., an inner interface or an outer interface), $F_{sim}(I) = m + |X| = i$ where i is the set $\{0, .., m-1; m, .., i-1\}$. This set has m elements in $\{0, ..., m-1\}$ that are *place nodes* and $|X|$ elements in $\{m, ..., i-1\}$ that are *link nodes*.

For example in Figure 2, the inner interface of B is $I = \langle 2, \{x_0\}\rangle$. Its corresponding image by $F_{sim}(I)$ is the interface $i = \{0, 1, 2\}$ of G where the nodes 0 and 1 are *place nodes* and the node 2 is a *link node*. The interface i is represented by the list of numbers on the left of the graph.

This definition of F_{sim} on objects proves that it is an injective function.

i.2 Defining F_{sim} on morphisms

We define F_{sim} as a pair of functions (f_v, f_e) that preserves bigraphs structure as highlighted in the following.

Definition IV.2 (F_{sim} on morphisms)**.** Let $B = (V_B, E_B, ctrl_B, prnt_B, link_B)$ be a bigraph. We define $F_{sim}(B) = G$ where $G = \langle V_G, E_G, s, t\rangle$ is defined through a pair of functions (f_v, f_e) such that:

- $f_v(V_B) \subset V_G$. f_v associates for each node of the bigraph, a *place node* in the graph G. $f_v : V_B \to V_G$ is defined as follows: $\forall\ v \in V_B, \exists!\ place\ node\ x \in V_G$ *such that* $f_v(v) = x$ where

  $$id(x) = \begin{cases} id(v) & \text{if } v \text{ is a root} \\ id(v).id(f_v(prnt_B(v))) & \text{otherwise} \end{cases}$$

  Hence, the identifier of a node image is determined by concatenating the identifier of this node with the identifier of its parents. For example, in Figure 2, $f_v(v_1) = v_1 : v_0 : 0$
- $f_e(E_B) \subset V_G$. f_e associates for each hyperedge of the bigraph, a *link node* in the graph G. $f_e : E_B \to V_G$ is defined as follows:

  $\forall\ e \in E_B, \exists!\ link\ node\ v \in V_G$ such that $f_e(e) = v$

  For example, in Figure 2, $f_e(e_0) = e_0$ (the image of the hyperedge e_0 in the bigraph B is the node e_0 in the graph G).
- (f_v, f_e) respects the structure of bigraphs in the following sense:
  1. It preserves the controls
  2. It preserves the structural mapping *prnt*
  3. It preserves the structural mapping *link*

  In the following, we show that these three points hold.

1. F_{sim} preserves the controls since the control function $ctrl_B$ is defined in the graph by the labelling function l_v as follows:

$$\forall v \in V_B,\ l_v(f_v(v)) \stackrel{def}{=} ctrl_B(v)$$

2. The parent mapping $prnt_B$ is defined in the graph through a function called *getParent* that exploits the node identifiers. Since the *id* of a node in the graph is the concatenation of its parents *ids*, this function $getParent : L_V \to L_V$ allows to return the parent *id*.

   $\forall\ id = \{id_1 : id_2 : ... : id_n\}$, $getParent(id) = id_2 : ... : id_n$. By this definition, we can ensure that F_{sim} preserves the structural mapping *prnt*.

3. The link mapping $link_B$ is defined in the graph by the function target t as follows:

   Let v be a node in a bigraph that is connected via a port (v, i) to a hyperedge h. In the graph image, there is a *link node* nh that represents the image of the hyperedge h and an edge e that represents the image of the port (v, i) having as source the node v and as target the node nh. Indeed, it follows that F_{sim} induces a function $f_p : P_B \to E_G$ on ports of the bigraph nodes defined by:

$$\forall\ port\ (v, i) \in P_B, \exists!\ edge\ e \in E_G$$

$$\text{such that } f_p((v, i)) = e \text{ and } s(e) = v$$

   So, formally, $link_B$ is defined in the graph as follows:
   Let $(v, i) \in P_B$ be a port and $h \in E_B$ be a hyperedge such that $link_B(v, i) = h$ then:

$$\begin{cases} \exists!\ link\ node\ nh\ \in V_G \text{ such that} \\ f_e(h) = nh \\ \exists!\ edge\ e \in E_G \text{ such that} \\ f_p((v, i)) = e \text{ and } s(e) = v \\ t(e) = nh \end{cases}$$

   By this definition, F_{sim} preserves the structural mapping $link_B$.

Proposition IV.1. *F_{sim} is a faithful functor between $\mathcal{BG}$ (the s-category of bigraphs) and $\mathcal{DG}$ (the s-category of ranked graphs).*

Proof. Let demonstrate, first, that F_{sim} is a well defined functor by demonstrating that it preserves functor properties (i.e., preserves identity and composition).

1. F_{sim} preserves identity: $F_{sim}(Id_A) = Id_{F_{sim}(A)}$ for every object A from $\mathcal{BG}$.
   Let $I = \langle m, X\rangle$ be an object in $\mathcal{BG}$ (i.e., an outer interface or an inner interface).
   The identity bigraph at I is $Id_I = (\varnothing, \varnothing, \varnothing, Id_m, Id_X) : \langle m, X\rangle \to \langle m, X\rangle$
   By the definition of F_{sim}, $F_{sim}(Id_I)$ is a ranked graph having no nodes and, having as interfaces $F_{sim}(I) = m + |X|$. This graph represents the identity on $(i = m + |X|)$ where i is an object in $\mathcal{DG}$ (i.e., an interface).
   Hence, $F_{sim}(Id_I) = Id_i = Id_{F_{sim}(I)}$

2. F_{sim} preserves composition: $F_{sim}(H \circ B) = F_{sim}(H) \circ F_{sim}(B)$ for all morphisms B and H from $\mathcal{BG}$ (See the **??** for the proof).

Moreover, F_{sim} is a faithful functor (injective on arrows) since the morphisms f_v and f_e are injective functions. □

Proposition IV.1 ensures the validity of moving from the s-category $\mathcal{BG}$ to the s-category $\mathcal{DG}$ by the functor F_{sim}, ensuring in this way the validity of encoding a bigraph into a ranked graph.

ii. Step 2: Encoding a reaction rule into a graph rule

Bigraphs are associated with reaction rules to form BRSs. These rules can be applied to rewrite bigraphs. On the other hand, the double pushout approach, DPO approach, for graph transformations has been proposed by Ehrig et al. [5]. According to this approach, graphs are associated with rewrite rules.

In this section, we present how to encode a reaction rule into a DPO rule preserving direct bigraph transformations. Each direct bigraph transformation $B \to B'$ via a reaction

rule is simulated by a direct graph transformation $G \Rightarrow G'$ via the encoded DPO rule.

Our aim is to encode a reaction rule into a DPO rule through F_{sim} preserving direct bigraph transformations. Our proof performs two steps:

- Step 1: Preserving direct transformations. This step consists in verifying that our simulating functor F_{sim} preserves direct bigraph transformations as follows: if applying the reaction rule R on the bigraph B gives the bigraph B', then applying $F_{sim}(R)$ on $F_{sim}(B)$ will give $F_{sim}(B')$.
- Step 2: Encoding $F_{sim}(R)$ into a DPO rule. This step consists in proving that a reaction rule $F_{sim}(R)$ can be encoded into a DPO rule.

ii.1 Categorical framework of BRSs

A Bigraphical reactive system is an instance of a Reactive System (Definition IV.3).

Definition IV.3 (Reactive System [1])**.** A reactive system, written $\mathcal{A}(\mathcal{R})$, consists of a category $\mathcal{A}$ equipped with a set $\mathcal{R}$ of reaction rules. Each reaction rule consists of a pair $(R : \epsilon \to I,\ R' : \epsilon \to I)$ of ground arrows, a *Redex* and a *Reactum*. A ground arrow is an arrow with domain ϵ.

The reaction relation $\to$ over agents $(a, a' : \epsilon \to J)$ is defined (written $a \to a'$) if there exist a reaction rule (r, r') and a context $D : I \to J$ such that in $a = D \circ R$ and $a' = D \circ R'$ in $\mathcal{A}$ are defined as shown in the commutative diagram (cf. left part of Figure 3).

On the other hand, an adhesive grammar $\langle \mathcal{DG}, P \rangle$ where $\mathcal{DG}$ is a category and P is a set of rewrite rules, can be seen as a reactive system on the category Cospan($\mathcal{DG}$). Sassone et al. [18] presented a Lemma that shows that the DPO rewrite relation is exactly the reactive system reaction relation.

So, both BRSs and GTSs are reactive systems.

ii.2 Step 1: Preserving direct transformations

Categorically, we want to move from a given $BRS = \mathcal{BG}(\mathcal{R}_1)$ with an s-category $\mathcal{BG}$ and reactions $\mathcal{R}_1$ to a $GTS = \mathcal{DG}(\mathcal{R}_2)$ with an s-category $\mathcal{DG}$ and suitable reactions $\mathcal{R}_2$. This can be achieved by using the functor $F_{sim} : \mathcal{BG}(\mathcal{R}_1) \to \mathcal{DG}(\mathcal{R}_2)$ where $F_{sim} : \mathcal{BG} \to \mathcal{DG}$ and $F_{sim}(\mathcal{R}_1) = \mathcal{R}_2$.

Proposition IV.2. *F_{sim} preserves direct transformations, i.e., translates the applicability of reactions.*

Proof. Let $a \to a'$ be a reaction relation via (R, R') and D given by (1) and (2) as defined in Figure 3. Let F_{sim} associates an image to each morphism and object of the commutative diagram depicted in Figure 3.

In order to ensure that the reaction relation $a \to a'$ leads to a reaction relation $F_{sim}(a) \to F_{sim}(a')$ via $(F_{sim}(R), F_{sim}(R'))$ and $F_{sim}(D))$, we need to demonstrate that $F_{sim}(a) = F_{sim}(D) \circ F_{sim}(R)$ and $F_{sim}(a') = F_{sim}(D) \circ F_{sim}(R')$.

First, we have $F_{sim}(a) = F_{sim}(D \circ R)$. Since F_{sim} preserves composition, we obtain $F_{sim}(a') = F_{sim}(D) \circ F_{sim}(R')$. Then, we the same manner, we have $F_{sim}(a') = F_{sim}(D \circ R')$. Since F_{sim} preserves composition, we obtain $F_{sim}(a') = F_{sim}(D) \circ F_{sim}(R')$ □

ii.3 Step 2: Encoding $F_{sim}(R)$ into a DPO rule

The second step of our solution relies on the idea of Ehrig [12]. He showed that it is possible to use the cospan idea to construct from a reaction relation a corresponding DPO transformation $a \Rightarrow a'$ via (p, D) where p is constructed from the reaction rule (R, R'), depicted categorically in the left diagram of Figure 3.

DPO rule Categorically, a rewrite rule in the DPO approach is a rule $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ consists of two injective graph morphisms (l, r). It can be applied to a graph G, resulting in

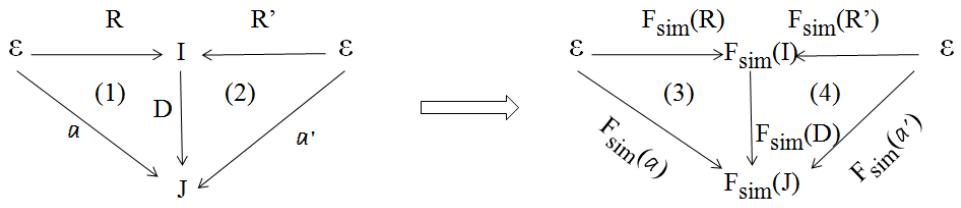


Figure 3: *Applying F_{sim} to a reaction relation*

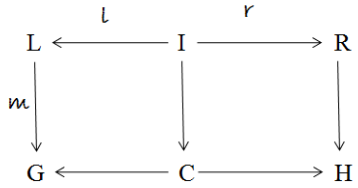


Figure 4: *Commutative diagram for a DPO transformation $G \Rightarrow H$ via (p, C)*

a graph H, if there is an injective match morphism $m : L \to G$ and if we can find a graph C and morphisms such that the two squares in the diagram of Figure 4 are both pushouts.

Now, let $(F_{sim}(R), F_{sim}(R') : \epsilon \to F_{sim}(I))$ be a reaction rule in $\mathcal{DG}$ and let's consider a reaction relation $F_{sim}(a) \to F_{sim}(a')$ via $(F_{sim}(R), F_{sim}(R')$ and $F_{sim}(D) : F_{sim}(I) \to F_{sim}(J)$ given by (3) and (4) in Figure 3, where $F_{sim}(a) = F_{sim}(D) \circ F_{sim}(R)$ and $F_{sim}(a') = F_{sim}(D) \circ F_{sim}(R')$.

For a reaction rule, morphisms of the square (3) and the triangle (4) are given as cospans in cospan $(\mathcal{G}raph)$, i.e., we have cospans $\epsilon \leftarrow F_{sim}(R) \to F_{sim}(I)$, $\epsilon \leftarrow F_{sim}(R') \to F_{sim}(I)$ and $F_{sim}(I) \leftarrow F_{sim}(D) \to F_{sim}(J)$ leading to cospans $\epsilon \leftarrow F_{sim}(a) \to F_{sim}(J)$ and $\epsilon \leftarrow F_{sim}(a') \to F_{sim}(J)$ by composition in $\mathcal{DG}$.

Since composition in $\mathcal{DG}$ is defined via pushouts, we obtain the diagram in Figure 5, where (5) and (6) is a double pushout leading to the DPO transformation $F_{sim}(a) \Rightarrow F_{sim}(a')$ via $(p, F_{sim}(D))$ where $p = (F_{sim}(R) \leftarrow F_{sim}(I) \to F_{sim}(R'))$.

iii. Step 4: Encoding a ranked graph into a bigraph

Ranked graphs with some additional information may be encoded into bigraphs. These information define the nesting and the linking structure of bigraphs. This encoding is the reverse process of encoding bigraphs into graphs presented in Section i. It is implemented as follows (cf. Figure 2):

- The graph nodes having the type *place nodes* are represented in the bigraph by nodes.
- The identifiers of a *place node* is exploited to represent the hierarchy of nodes in the bigraph. For example, the identifier of the node v_1:v_0:0 in the graph means that, in the bigraph, the node v_1 is nested in the node v_0 that is nested in turn in the root 0.
- The graph nodes having the type *link nodes* are represented in the bigraph by hyperedges. The source of edges connected to a *link node* are linked by the corresponding hyperedge. For example, the green *link node* e_1 in the graph to which are connected v_2:0 and v_3:1 nodes, is represented in the bigraph with the hyperedge e_1 connecting v_2 and v_3 nodes.
- The inner and the outer interfaces of the graph are represented by the inner and the outer interfaces of the bigraph.

Our main objective is to encode ranked graphs into bigraphs, preserving their structure. We shall achieve this by constructing a categorical functor that allows to move from the ranked graph s-category $\mathcal{DG}$ to the bigraph s-category $\mathcal{BG}$. We construct a functor

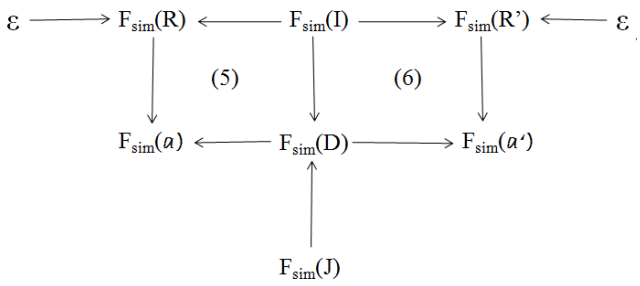


Figure 5: *A reaction relation as a DPO transformation*

named $F_{rev} : \mathcal{DG} \to \mathcal{BG}$ that associates to a graph $G : i \to j$, from $\mathcal{DG}$ a bigraph $B : I \to J$ from $\mathcal{BG}$ such that:

$$\begin{cases} F_{rev}(i) = I \\ F_{rev}(j) = J \\ F_{rev}(G) = B \end{cases}$$

We define the reversing functor F_{rev} on the objects and on the morphisms of the two categories $\mathcal{DG}$ and $\mathcal{BG}$.

iii.1 Defining F_{rev} on objects

We define F_{rev} as an injective function between the objects (i.e., interfaces) of the two models. Given a graph interface represented as a list of ordered numbers i, F_{rev} associates a bigraphical interface $\langle m, X \rangle$ where $m = |place\ node|$ and $|X| = |link\ node|$. Every $place\ node \in i$ is encoded in the bigraph by a root or a site and every $link\ node \in i$ is encoded in the bigraph by a name in X (an inner name or an outer name).

Hence, for each object $i = \{0, ..., i-1\}$ from $\mathcal{DG}$, $F_{rev}(i) = \langle m, X \rangle$ where $m = |place\ node|$ and $X = \{x_0, ..., x_i\}$ such that $i = |link\ node|$.

For example in Figure 2, the inner interface of G is $i = \{0, 1, 2\}$ where the nodes 0 and 1 are *place nodes* and the node 2 is a *link node*. Its corresponding image by $F_{rev}(i)$ is the interface I of B where $I = \langle 2, \{x_0\} \rangle$.

This definition of F_{rev} on objects proves that it is an injective function.

iii.2 Defining F_{rev} on morphisms

We define F_{rev} as a pair of functions (f_v, f_e) that preserves graph structure as highlighted in the following.

Definition IV.4 (F_{rev} on morphisms)**.** Let $G = (r, \langle V, E, s, t \rangle, v)$ be a graph. We define $F_{rev}(G) = B$. $B = (V_B, E_B, ctrl_B, prnt_B, link_B)$ is defined through a pair of functions (f_v, f_e) where:

- $f_v(V) = V_B \cup E_B$. f_v associates for each *link node* of the graph an hyperedge in the bigraph and it associates for each *place node* of the graph, a node in the bigraph B. $f_v : V_G \to V_B \cup E_B$ is defined as follows:
  $\forall$ *place node* $v \in V_G$, $\exists!$ *node* $x \in V_B$ *such that* $f_v(v) = x$ where

  $$id(x) = \begin{cases} id(v) & \text{if } v \text{ is a root} \\ split(id(v), :, 1) & \text{otherwise} \end{cases} \quad and$$

  $$prnt_B(x) = split(id(v), :, 2)$$

  Hence, the image of a node is determined by splitting the first item of its identifier and its parent is determined by splitting the second item. For example, in Figure 2, $f_v(v_1 : v_0 : 0) = v_1$ and $prnt_B(v_1) = v_0$.
- $f_e(E_G) = P_B$. f_e associates for each edge of the graph, a port for its source node in the graph G. $f_e : E_G \to P_B$ is defined as follows:

  $\forall e \in E_G$ such that $s(e) = v$ and $t(e) = h$,
  $\exists!$ port p of v such that $link_B(p) = f_v(h)$

- By definition, (f_v, f_e) respects the structure of bigraphs in the following sense:
  1. It preserves the source function s
  2. It preserves the target function t
  3. It preserves the label function L_v, i.e., $L_v \stackrel{def}{=} ctrl_B \circ f_v$

Proposition IV.3. *F_{rev} is a faithful functor between $\mathcal{DG}$ (the s-category of ranked graphs) and $\mathcal{BG}$ (the s-category of bigraphs).*

Proof. Let demonstrate, first, that F_{rev} is a well defined functor by demonstrating that it preserves functor properties (i.e., preserves identity and composition).

1. F_{rev} preserves identity: $F_{rev}(Id_A) = Id_{F_{rev}(A)}$ for every object A from $\mathcal{DG}$.
   Let i be an object in $\mathcal{DG}$ (i.e., an outer interface or an inner interface). $i = \{0, ..m-1, m, ...i-1\}$ having m places and $i-m$ names. Id_i is a ranked graph having $i-1$ nodes where the $k-th$ root is the $k-th$ variable, for all $k \in i$.
   $F_{rev}(Id_i)$ is the bigraph $(\varnothing, \varnothing, \varnothing, Id_m, Id_X) : \langle m, X\rangle \rightarrow \langle m, X\rangle$ where $|X| = i - m$. This bigraph represents the identity bigraph at the object $I = \langle m, X\rangle = F_{rev}(m + |X|) = F_{rev}(i)$ in $\mathcal{BG}$.
   Hence, $F_{rev}(Id_i) = Id_I = Id_{F_{rev}(i)}$

2. F_{rev} preserves composition: $F_{rev}(H \circ B) = F_{rev}(H) \circ F_{rev}(B)$ for all morphisms B and H from $\mathcal{DG}$ (See the **??** for the proof).

Moreover, F_{rev} is a faithful functor (injective on arrows) since the morphisms f_v and f_e are injective functions. □

Proposition IV.3 ensures the validity of moving from the category $\mathcal{DG}$ to the category $\mathcal{BG}$ by the functor F_{rev}, ensuring in this way the validity of encoding a ranked graph into a bigraph.

V. BiGMTE: Bigraph Matching and Transformation Engine

We propose our BiGMTE tool [19, 20], an implementation of our proposed solution for executing BRSs based on graph rewriting.

i. BiGMTE architecture

BiGMTE is an integration of existing tools. It uses Big Red [21], a graphical editor allowing to create and edit bigraphs and reaction rules. Besides, it is based on GMTE [22], a tool for graph matching and transformation. Therofore, the matching and the transformation procedure performs five steps as highlighted in Figure 6:

- **BRS creating** Using Big Red the user creates a BRS (Bigraphs and reaction rules). The edited BRS is exported into an XML file.
- **BRS encoding** This XML file is parsed to be the input of our implemented algorithms. The first algorithm encodes the edited bigraph to a graph and the second one encodes the edited reaction rule to a DPO rule.
- **Graph matching** The obtained DPO rule is executed and applied on the obtained graph using GMTE.
- **Graph encoding** After the rule application, the resulting graphs generated by GMTE are encoded in turn to bigraphs.
- **Displaying** The obtained bigraphs are displayed by Big Red.

ii. The graphical interface of BiGMTE

In order to create a BRS with Big Red, the user should start by defining the signature, i.e., the set of the controls. Then, using the created signature, he creates the bigraphs and the reaction rules.

We illustrate the creation of a BRS using an example of the bigraphical reactive system “Built environment" given in [1]. A built environment, depicted in Figure 7, consists of

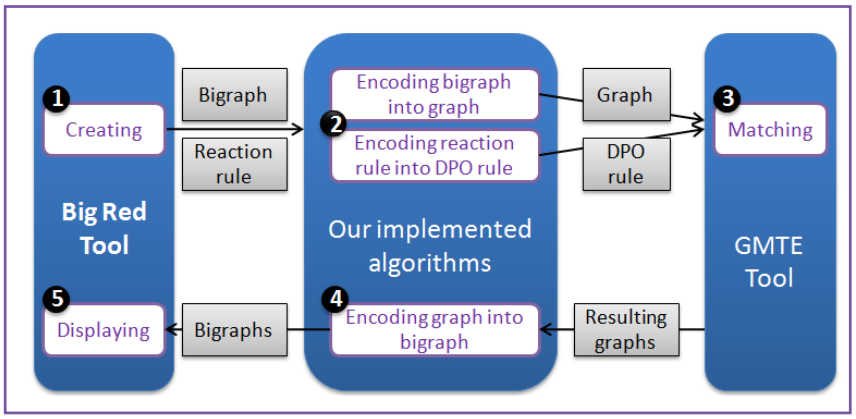


Figure 6: *BiGMTE architecture*

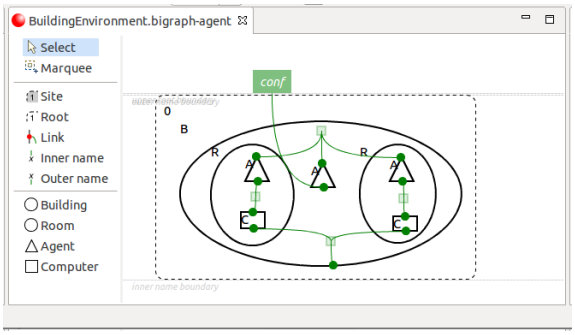


Figure 7: *Big Red's bigraph editor*

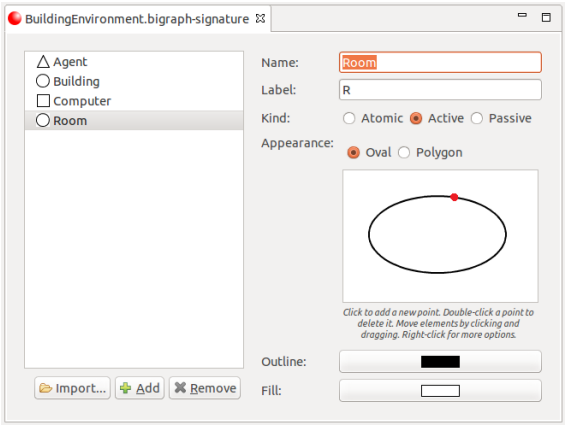


Figure 8: *Big Red's signature editor*

agents, buildings, rooms and computers. The state of the building may change because of the movement of agents, and perhaps other movements. Think of the agents as conducting a conference call (the long link). An agent in a room may also be logged in a computer in the room (the short links), and the computers in a building are linked to form a local area network.

ii.1 Creating the signature

The signature editor, shown in Figure 8, manages both the bigraphical and visual properties of controls. The list view on the left shows the controls of the signature, and allows controls to be added and removed, while the panel on the right allows controls to be modified.

The appearance and arity of a control are defined using the canvas in the centre-right of Figure 8. The black circles are the vertices of the control's oval, while ports are shown on the oval as red circles (in this case, only one is shown). All of these objects can be created, moved and removed using the canvas and its context menu.

ii.2 Creating the bigraph agent

Big Red's bigraph editor, shown in Figure 7, is essentially a structured vector graphics editor. The palette on the left provides the tools that can be used to modify the model, for example, the "Link" tool can be used to connect points and links, and the "Room" tool can be used to create new nodes whose control is Room,

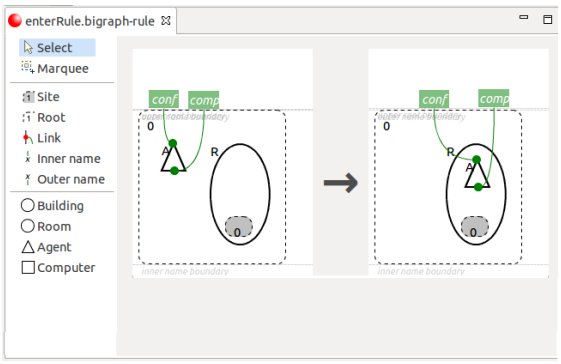


Figure 9: *Reaction rule*

while the view on the right shows the view of the model.

The editor enforces the structural rules and visual conventions of bigraphs; if the user attempts to modify the model in a way that would cause them to be violated (for example, by dragging node R out of root 0), an explanatory error message will be displayed.

ii.3 Creating a reaction rule

We create a reaction rule allowing an agent to enter a room, depicted in Figure 9. Likewise editing bigraphs, the palette on the left of the editor provides the tools that can be used to modify the model and the set of controls that can be used to create new nodes, while the view on the right shows the view of the reaction rule.

The rule requires the agent and the room to be in the same place (presumably a building). The site allows the room to contain other occupants and other agents. The matching discipline allows the ports of the agent to be already linked to a computer and other agents somewhere, perhaps in another room.

ii.4 Launching the execution of the BRS

In order to execute a BRS, the designer uses the interface depicted in Figure 10. He chooses, first, the signature. Then, he chooses the reaction rule to be applied and the model, i.e., the bigraph on which will be executed the

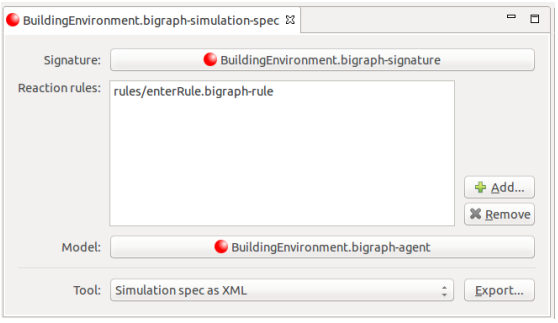


Figure 10: *Executing a BRS*

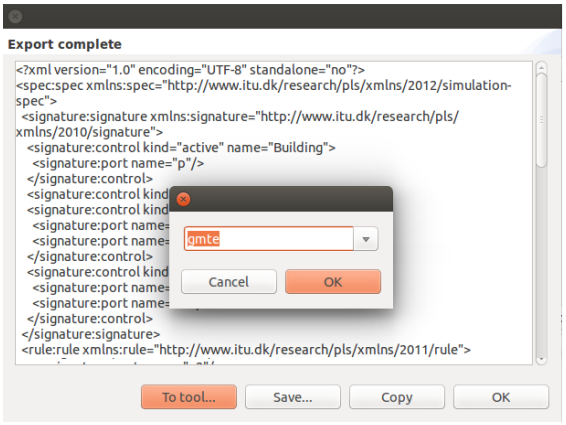


Figure 11: *Connecting to GMTE*

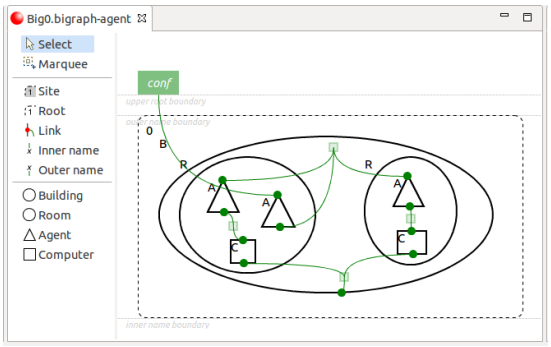


Figure 12: *The resulted Bigraph*

rule. By clicking on “export", Big Red generates an XML file describing the BRS as showed in Figure 11. Finally, the designer clicks on “To tool" in order to connect to GMTE. Hence, the encoding of the BRS into a GTS will be performed.

The edited bigraph is encoded into a graph. Likewise, the edited reaction rule is encoded into a graph rule. The obtained graph and graph rule are passed as input to GMTE. GMTE executes the application of the rule and generates a resulted graph. The resulted graph is encoded into a bigraph that will be then displayed through Big Red, as showed in Figure 12.

iii. Evaluation of BiGMTE

In order to evaluate BiGMTE, we compared its supported features and its performance with two existing tools: LibBig [10] and BPL Tool[4].

LibBig is a Java library for Bigraphs and BRSs. It is an implementation of the machinery for defining and manipulating BRSs. Matching is implemented as a constraint satisfaction problem (CSP). BPL Tool (Bigraphical Programming Languages) is a tool for experimenting with bigraphical models. It relies on an SML (Standard Meta Language) compiler with an interactive mode to provide a command line interface.

iii.1 Features evaluation

By comparing the features supported by each tool, we can note that:

- BiGMTE is able to execute a rule as many times as possible. That’s mean; it applies the rule recursively until it will be not applicable. This feature is not supported by LibBig and BPL.
- BiGMTE offers a GUI interface that helps the user to edit bigraphs and reaction rules. However, using LibBig and BPL, this task is a bit difficult since it is done through Java programs and SML programs, respectively.
- BiGMTE is able to apply a rule and execute all founded matching. It gives all resulted bigraphs. This feature is supported also by LibBig but not supported by BPL.
- Regarding the execution of parametric rules, this feature is supported by BPL but not supported by LibBig. The current version of BiGMTE is able to execute some parametric rules that allow copying parameters (deleting, moving and duplicating parameters are not yet supported).

Moreover, by manipulating BPL, we noticed that it is not able to execute a rule that allows linking more than three nodes.

iii.2 Performance evaluation

In order to evaluate the performance of BiGMTE, we compared it with LibBig by performing a set of tests. The generation of these tests is based on increasing the number and the hierarchical level of the nodes involved in both the agent and the *Redex*. In the first experimentation, we considered the size of the *Redex*. In the second and the third experimentations, the tests are generating by increasing the size of the agent. The last experimentation, the tests are generating by increasing the hierarchical level of nodes in the agent. In these experimentations, we considered, as a reference parameter, the time of executing a reaction rule on a bigraph and getting the new bigraph.

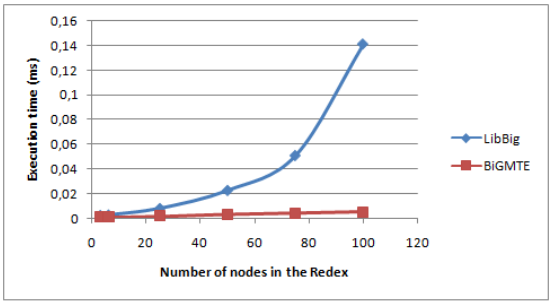


Figure 13: *Execution time compared to Redex size*

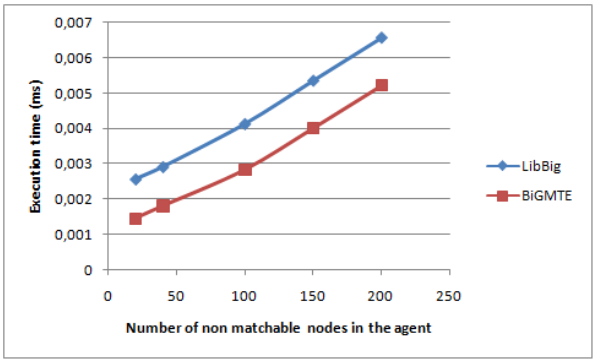


Figure 14: *Execution time while increasing the number of non matchable nodes*

Evaluation compared to the *Redex* size In this evaluation, we measured the execution time of a reaction rule while increasing the size of its *Redex*. To do so, we take, first, the example of a reaction rule that allows to find a pattern that contains one node. The agent (i.e., the bigraph on which the reaction rule will be applied) is a bigraph that is formed by a set of nodes with different controls and have the same parent. It contains one matching of the *Redex* pattern. Then, we carry on the generation of tests by modifying the reaction rule and increasing, each time, the number of nodes in the *Redex*. In each test, the added nodes have the same parent and have different controls.

Figure 13 depicts that the execution time increases exponentially with LibBig. However, it increases slightly with BiGMTE. Accordingly, BiGMTE is faster than LibBig in this case. This can be explained by the fact that LibBig is based on a CSP solver. In fact, by increasing the complexity of the *Redex*, the number of constraints increases, too, and that slows down the resolution of the constraint satisfaction problem.

Evaluation compared to the agent size: one matching In this evaluation, we measured the execution time of a reaction rule while increasing the size of the agent. To do so, we take, the example of a reaction rule that allows to find a pattern that contains one node. First, we apply this reaction rule on an agent that contains only the pattern to be matched. Then, we modify the agent by increasing the number of nodes. The application of the rule gives always one matching. As showed in Figure 14, BiGMTE is faster than LibBig.

iii.3 Evaluation compared to the agent size: several matching

In this evaluation, we take the same example of a reaction rule that allows to find a pattern that contains one node. However, we apply this reaction rule, first, on an agent that contains only the pattern to be matched (i.e., the application of the rule gives only one matching). Then, we modify the agent by increasing the number of nodes (nodes have the same types). Accordingly, the number of matching increases with increasing the number of nodes.

As depicted in Figure 15, we can note that the execution time using the two tools is very close when the number of matching is under 100. Whereas, when the number of matching is more than 100, LibBig responds better than BiGMTE.

iii.4 Evaluation compared to the agent depth

At first, we take the example of a reaction rule that allows to find a pattern that contains one node and an agent that contains only the pattern to be matched. Then, we modify the agent by increasing, each time, the nesting level of the *Redex* pattern to be found. The

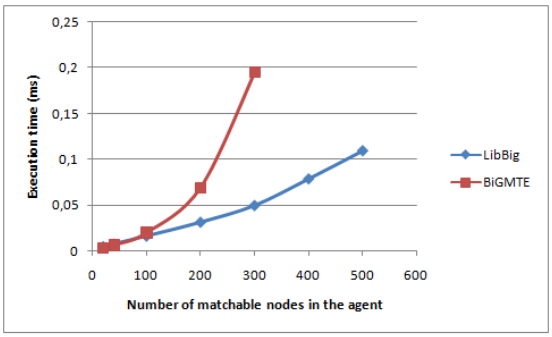


Figure 15: *Execution time while increasing the number of matchable nodes*

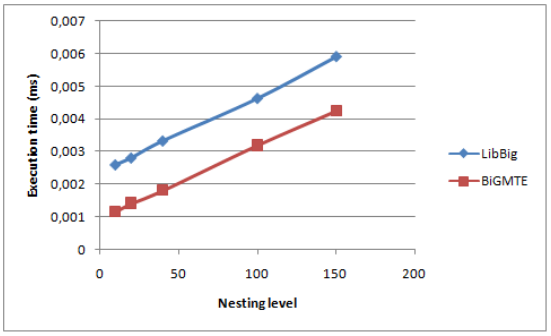


Figure 16: *Execution time while increasing the nesting level of nodes in the agent*

application of the rule gives always one matching. As showed in Figure 16, BiGMTE is faster than LibBig.

VI. Conclusions and Future Work

In this paper, we present a solution for executing BRSs based on an investigation on graph matching. We initiate this investigation following the closely relation between the theory of BRSs and GTSs and considering also the exhaustiveness of studies on graph transformations. As an alternative to implement matching for bigraphs, we propose to simulate a BRS with a GTS. First, we encode a bigraph into a ranked graph. The validity of this encoding is ensured by providing a formal proof. Categorically, both bigraphs and ranked graphs are presented as arrows between two interfaces, forming a category. So, we define a faithful functor called F_{sim} that allows to move from the bigraph category to the ranked graph category.

Then, we show that reaction rules can be simulated by graph rules. We encode a reaction rule into a graph rule through F_{sim} that preserves direct transformations, i.e., translates the applicability of reactions. Hence, we provide a formal basis allowing to execute bigraph transformations by simulating their encoding in order to use well-established and efficient graph transformation tools.

As an implementation of this solution, we presented BiGMTE, our tool for bigraph matching and transformation. It allows to execute the application of a reaction rule on a given bigraph to be rewritten. BiGMTE is an integration of existing tools. It uses Big Red, a graphical editor allowing to create and edit bigraphs and reaction rules. The edited bigraph and reaction rule are encoded into a graph and a graph rule, respectively. To do so, we implemented two algorithms allowing these encodings. BiGMTE is also based also on GMTE, a tool for graph matching and transformation, for executing the encoded rule on the encoded graph.

In order to improve the performance of our implemented BiGMTE tool, we are working on the issue of decreasing the execution time when having a large number of matchings. Moreover, we aim to extend BiGMTE to support the matching of labelled bigraphs. This feature can be easily extended since it is supported by GMTE. We only have to encode labelled bigraphs into labelled graphs.

References

[1] R. Milner, The Space and Motion of Communicating Agents, Cambridge University Press, 2009.

[2] T. C. Damgaard, A. J. Glenstrup, L. Birkedal, R. Milner, An inductive characterization of matching in binding bi-

graphs, Formal Aspects of Computing 25 (2) (2013) 257–288.

[3] A. J. Glenstrup, T. C. Damgaard, L. Birkedal, E. HÃÿjsgaard, An implementation of bigraph matching, Tech. Rep. TR-2010-135, IT University of Copenhagen (December 2010).

[4] E. HÃÿjsgaard, A. J. Glenstrup, The BPL Tool: A Tool for Experimenting with Bigraphical Reactive Systems, Tech. Rep. TR-2011-145, IT University of Copenhagen (October 2011).

[5] H. Ehrig, M. Pfender, H. J. Schneider, Graph-grammars: An algebraic approach, in: the IEEE Conference Record of 14th Annual Symposium on Switching and Automata Theory (SWAT), 1973, pp. 167–180.

[6] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer, Fundamentals of Algebraic Graph Transformation (Monographs in Theoretical Computer Science), Springer-Verlag, 2006.

[7] G. Bacci, M. Miculan, R. Rizzi, Finding a forest in a tree, in: Trustworthy Global Computing: 9th International Symposium, TGC, Springer Berlin Heidelberg, 2014, pp. 17–33.

[8] M. Sevegnani, C. Unsworth, M. Calder, A SAT based algorithm for the matching problem in bigraphs with sharing, Tech. rep., University of Glasgow, Department of Computing Science (2010).

[9] M. Miculan, M. Peressotti, A CSP implementation of the bigraph embedding problem, CoRR abs/1412.1042. arXiv:1412.1042.

[10] M. Miculan, M. Peressotti, Libbig: A library for bigraphs and bigraphical reactive systems, http://mads.uniud.it/wordpress/downloads/libbig/ (2014).

[11] A. Mansutti, M. Miculan, M. Peressotti, Distributed execution of bigraphical reactive systems, CoRR abs/1503.02434. arXiv:1503.02434.

[12] H. Ehrig, Bigraphs meet double pushouts, Bulletin of the EATCS 78 (2002) 72–85.

[13] F. Gadducci, R. Heckel, An inductive view of graph transformation, in: Recent Trends in Algebraic Development Techniques, Vol. 1376 of Lecture Notes in Computer Science, Springer, 1998, pp. 223–237.

[14] R. Milner, Embeddings and contexts for link graphs, in: Formal Methods in Software and Systems Modeling, Springer, 2005, pp. 343–351.

[15] R. Bruni, U. Montanari, G. Plotkin, D. Terreni, On Hierarchical Graphs: Reconciling Bigraphs, Gs-monoidal Theories and Gs-graphs, Fundamenta Informaticae 134 (3-4) (2014) 287–317.

[16] T. C. Damgaard, Syntactic theory for bigraphs, Tech. rep., IT University of Copenhagen (2006).

[17] A. Gassara, I. B. Rodriguez, M. Jmaiel, K. Drira, Encoding bigraphical reactive systems into graph transformation systems, Electronic Notes in Discrete Mathematics 55 (2016) 207–210.

[18] V. Sassone, P. Sobocinski, Reactive Systems over Cospans, in: the 20th IEEE Symposium on Logic in Computer Science (LICS), 2005, pp. 311–320.

[19] A. Gassara, I. B. Rodriguez, M. Jmaiel, A tool for modeling sos architectures using bigraphs, in: Proceedings of the Symposium on Applied Computing, 2017, pp. 1787–1792.

[20] A. Gassara, I. Bouassida Rodriguez, BiGMTE, http://www.redcad.tn/projects/bigmte/ (2017).

[21] A. J. Faithfull, G. Perrone, T. T. Hildebrandt, Big red: A development environment for bigraphs, Electronic Communications of the EASST, 2013, 61.

[22] M. A. Hannachi, I. Bouassida Rodriguez, K. Drira, S. E. Pomares Hernandez, GMTE: A tool for graph transformation and exact/inexact graph matching, in: Graph-Based Representations in Pattern Recognition, Springer, 2013, pp. 71–80.

Appendix A. Mathematical framework

Bigraphs may be defined as arrows in certain kinds of category. In our work, we are concerned with concrete bigraphs, in s-categories, where nodes and edges have identifiers. In this section, we present some definitions related to the theory of category.

Definition Appendix A.1 (category [20])**.** A category $\mathcal{C}$ has a set of objects and a set of arrows. We shall often denote objects by I, J, K and arrows by f, g, h. Each arrow f has a domain and a codomain, both objects; if these are I and J then we write $f : I \to J$, $I = dom(f)$ and $J = cod(f)$.

For each object I there is an identity arrow $id_I : I \to I$. The composition $g \circ f$ of f and g satisfies the following:
(C1) $g \circ f$ is defined iff $cod(f) = dom(g)$
(C2) $h \circ (g \circ f) = (h \circ g) \circ f$ when either are defined
(C3) $id \circ f = f$ and $f = f \circ id$.

In s-categories, each arrow is associated with a set called its support. This association will be arbitrary, subject to simple constraints detailed in the following definition. In a bigraph, the support will include its nodes, and this immediately allows to handle occurrences and the sharing of nodes.

Definition Appendix A.2 (s-category [20])**.** An s-category is a category in which each arrow f is assigned a finite support $|f|$ and composition of f and g may be undefined even when $cod(f) = dom(g)$.

Furthermore, $g \circ f$ is defined iff $cod(f) = dom(g)$ and $|f| \cap |g| = \varnothing$, and in that case $|g \circ f| = |f| \uplus |g|$.

Appendix B. Casting the $\mathcal{DG}$ category as an s-category

The obvious way to cast a category (Definition in Appendix A.1) into an s-category (Definition in Appendix A.2) is to give all morphisms a support and to verify that the composition of two morphisms g and f is defined iff $cod(f) = dom(g)$ and $|f| \cap |g| = \varnothing$.

Let $f = \langle r, d, v \rangle$ and $g = \langle r', d', v' \rangle$ be two ranked graphs (having supports since d and d' are concrete graphs, i.e., nodes have identifiers). We demonstrate that $g \circ f$ is defined iff $cod(f) = dom(g)$ and $|f| \cap |g| = \varnothing$.

Proof. ($\Leftarrow$) Evident since $\mathcal{DG}$ is a category.

($\Rightarrow$) $\mathcal{DG}$ is a category. So, $g \circ f$ is defined iff $cod(f) = dom(g)$. Let demonstrate that $|f| \cap |g| = \varnothing$.

We assume that the statement is false (i.e., $|f| \cap |g| \neq \varnothing$). So, $\exists$ a node $x \in f$ and a node $y \in g$ such that $id(x) = id(y)$ (having the same identifier). By definition, the composition $f \circ g = \langle r, d'', v' \rangle$ where d'' is the disjoint union modulo the equivalence on nodes induced by $v(x) \approx r'(x)$. So, $\{x, y\} \in f \circ g$.

Since in a concrete graph each node has an unique identifier, $id(x)$ must be different to $id(y)$ and this is in a contradiction with the assumption. □

Appendix C. F_{sim} preserves composition

We demonstrate here that $F_{sim}(H \circ B) = F_{sim}(H) \circ F_{sim}(B)$ for all morphisms in $\mathcal{BG}$. To do so, we give, first, the definition of bigraph composition and ranked graph composition.

Bigraph composition

The composition of two bigraphs is defined by matching the inner interface of the first graph with the outer interface of the second (i.e., filling the sites (holes) of the first with regions of the second and merging the inner names of the first with the outer names of the second).

Definition Appendix C.1 (Bigraph composition [20])**.** Let $B = (V_B, E_B, ctrl_B, prnt_B, link_B) : \langle m, X\rangle \to \langle n, Y\rangle$ and $H = (V_H, E_H, ctrl_H, prnt_H, link_H) : \langle n, Y\rangle \to \langle k, Z\rangle$ be two bigraphs. Their composite

$$H \circ B = (V, E, ctrl, prnt, link) : \langle m, X\rangle \to \langle k, Z\rangle$$

has $V = V_B \uplus V_H, E = E_B \uplus E_H, ctrl = ctrl_B \uplus ctrl_H, prnt$ is defined as follows:
If $w \in m \uplus V_B \uplus V_H$ is a site or node of $H \circ B$ then

$$prnt(w) = \begin{cases} prnt_B(w) & if \quad w \in V_B \uplus m \text{ and } prnt_B(w) \in V_B \\ prnt_H(j) & if \quad w \in V_B \uplus m \text{ and } prnt_B(w) = j \in n \\ prnt_H(w) & if \quad w \in V_H \end{cases}$$

and link is defined as follows:
If q $\in P_B \uplus P_H \uplus X$ is a point (a port or an innername) of $H \circ B$ then

$$link(q) = \begin{cases} link_B(q) & if \quad q \in P_B \uplus X \text{ and } link_B(q) \in E_B \\ link_H(y) & if \quad q \in P_B \uplus X \text{ and } link_B(q) = y \in Y \\ link_H(q) & if \quad q \in P_H \end{cases}$$

Ranked graphs composition

The composition of two ranked graphs is obtained by gluing the variables of the first one with the roots of the second one (i.e., matching them and then eliminating them). It is defined only if their number is equal.

Definition Appendix C.2 (Ranked graph composition [9])**.** Let $G_1 = [\langle r_1, d_1, v_1\rangle]$ and $G_2 = [\langle r_2, d_2, v_2\rangle]$ be two ranked graphs. Their composition is the ranked graph $H = G_1; G_2$ defined as $H = [\langle in_{d2} \circ r_2, d''; in_{d1} \circ r_1\rangle]$, where $\langle d'', in_{d2}, in_{d1}\rangle$ is a pushout of : $\langle v_2 : j \to d_2, r_1 : j \to d_1\rangle$ in $\mathcal{DG}$ (category of ranked graphs). d'' is the disjoint union, modulo the equivalence on nodes induced by $v_2(i) \approx r_1(i)$.

Let $B = (V_B, E_B, ctrl_B, prnt_B, link_B) : I = \langle m, X\rangle \to J = \langle n, Y\rangle$,
$H = (V_H, E_H, ctrl_H, prnt_H, link_H) : J = \langle n, Y\rangle \to K = \langle k, Z\rangle$ and
$H \circ B : I \to K$. In order to demonstrate that $F_{sim}(H \circ B) = F_{sim}(H) \circ F_{sim}(B)$, we start by giving the expression of $F_{sim}(H \circ B)$ then the expression of $F_{sim}(H) \circ F_{sim}(B)$. For sake of simplicity, we use the notation F instead of F_{sim}.

The expression of $F(H \circ B)$

We give the expression of $F(H \circ B)$ based on the Definition Appendix C.1 of the composition $H \circ B$.
$F(H \circ B) : F(I) \to F(K)$ is a ranked graph that has:

- $V = F(V_B \uplus V_H) = V_{F(B)} \uplus V_{F(H)}$
- $VE = F(E_B \uplus E_H) = VE_{F(B)} \uplus VE_{F(H)}$
- $l_v = F(ctrl_B \uplus ctrl_H) = ctrl_{F(B)} \uplus ctrl_{F(H)}$
- The identifiers of nodes are defined as follows:
  If $w \in V_B \uplus V_H$ is a node of $H \circ B$ then $\exists!\ f_v(w) \in F(H \circ B)$ such that:

$$id(f_v(w)) = \begin{cases} id(w) & \text{if } w = i, i \in k \\ id(w).id(f_v(prnt_H(w))) & \text{if } w \in V_H \\ id(w).id(f_v(prnt_B(w))) & \text{if } w \in V_B \\ id(w).id(f_v(prnt_H(j))) & \text{if } w \in V_B \text{ and } prnt_B(w) = j, j \in n \end{cases}$$

- t defined as follows: If q $\in P_B \uplus P_H \uplus X$ is a point of $H \circ B$ then $\exists!$ *edge* $e = f_p(q) \in F(H \circ B)$ such that:

$$t(f_p(q)) = \begin{cases} f_e(link_B(q)) & \text{if } q \in P_B \uplus X \text{ and } link_B(q) \in E_B \\ f_e(link_H(y)) & \text{if } q \in P_B \uplus X \text{ and } link_B(q) = y \in Y \\ f_e(link_H(q)) & \text{if } q \in P_H \end{cases}$$

The expression of $F(H) \circ F(B)$

Let $F(B) = \langle r_1, (V_{F(B)}, VE_{F(B)}, t_{F(B)}), v_1 \rangle : F(I) \to F(J)$ and $F(H) = \langle r_2, (V_{F(H)}, VE_{F(H)}, t_{F(H)}), v_2 \rangle : F(J) \to F(K)$ where $F(I) = i$, $F(J) = j$ and $F(K) = k$.
$F(H) \circ F(B)$ is the disjoint union modulo the equivalence on nodes induced by $v_2(i) \approx r_1(i)$, $i \in j$. So, $F(H) \circ F(B)$ is a ranked graph which has:

- $V = V_{F(H)} \uplus V_{F(B)}$
- $VE = VE_{F(H)} \uplus VE_{F(B)}$
- The identifiers of nodes are determined as follows: For $v \in V$

$$id(v) = \begin{cases} id(v) & \text{if } v = r_2(i), i \in k \\ id(v) & \text{if } v \in V_{F(H)} \\ id(v) & \text{if } v \in V_{F(B)} \\ id(v).id(v_2(i)) & \text{if } v \in V_{F(B)} \text{ and } v = r_1(i), i \in j \end{cases}$$

  Since $v \in V_{F(B)}$ (resp. $\in V_{F(H)}$) then $\exists!\ w \in V_B$ (resp. $\in V_H$) such that $f_v(w) = v$.
  Hence, for $f_v(w) \in V$

$$id(f_v(w)) = \begin{cases} id(f_v(w)) & \text{if } f_v(w) = r_2(i), i \in k \\ id(f_v(w)) & \text{if } f_v(w) \in V_{F(H)} \\ id(f_v(w)) & \text{if } f_v(w) \in V_{F(B)} \\ id(f_v(w)).id(v_2(i)) & \text{if } f_v(w) \in V_{F(B)} \text{ and } f_v(w) = r_1(i), i \in j \end{cases}$$

$$id(f_v(w)) = \begin{cases} id(f_v(w)) & \text{if } w = i \in k \\ id(f_v(w)) & \text{if } w \in V_H \\ id(f_v(w)) & \text{if } w \in V_B \\ id(f_v(w)).id(v_2(i)) & \text{if } w \in V_B \text{ and } prnt_B(w) = j, j \in n \end{cases}$$

According to the definition of F_{sim}, we obtain the following expression:

$$id(f_v(w)) = \begin{cases} id(w) & \text{if } w = i \in k \\ id(w).id(f_v(prnt_H(w))) & \text{if } w \in V_H \\ id(w).id(f_v(prnt_B(w))) & \text{if } w \in V_B \\ id(w).id(f_v(prnt_H(w))) & \text{if } w \in V_B \text{ and } prnt_B(w) = j, j \in n \end{cases}$$

- t is determined by:

$$t(e) = \begin{cases} t_{F(B)}(e) & \text{if } e \in E_{F(B)} \text{ and } t_{F(B)}(e) \neq r_1(i), i \in j \\ t_{F(H)}(i) & \text{if } e \in E_{F(B)} \text{ and } t_{F(B)}(e) = r_1(i), i \in j \\ t_{F(H)}(e) & \text{if } e \in E_{F(H)} \end{cases}$$

Since $e \in E_{F(B)}$ (resp. $\in E_{F(H)}$) then $\exists!\ q \in P_B \uplus X$ (resp. $\in P_H$) such that $f_p(q) = e$.
Hence, for $f_p(q) \in VE$

$$t(f_p(q)) = \begin{cases} t_{F(B)}(f_p(q)) & \text{if } q \in P_B \uplus X \text{and } link_B(q) \in E_B \\ t_{F(H)}(f_p(q)) & \text{if } q \in P_B \uplus X \text{and } link_B(q) = y \in Y \\ t_{F(H)}(f_p(q)) & \text{if } q \in P_H \end{cases}$$

Since F preserves $link$ mapping (i.e., $f_e \circ link_B = t_{F(B)} \circ f_p$ and $f_e \circ link_H = t_{F(H)} \circ f_p$), then:

$$t(f_p(q)) = \begin{cases} f_e(link_B(q)) & \text{if } q \in P_B \uplus X \text{and } link_B(q) \in E_B \\ f_e(link_H(y)) & \text{if } q \in P_B \uplus X \text{and } link_B(q) = y \in Y \\ f_e(link_H(q)) & \text{if } q \in P_H \end{cases}$$

By looking at the expression of $F(H \circ B)$ and $F(H) \circ F(B)$, we can notice that $F(H \circ B) = F(H) \circ F(B)$. Hence, F_{sim} preserves composition.

Appendix D. F_{rev} preserves composition

Let $B = \langle r, d, v \rangle : i \to j$ where $d = \langle N_B, E_B, s_B, t_B \rangle$, $H = \langle r', d', v' \rangle : j \to k$ where $d' = \langle N_H, E_H, s_H, t_H \rangle$ and $H \circ B : i \to k$.

In order to demonstrate that $F_{rev}(H \circ B) = F_{rev}(H) \circ F_{rev}(B)$, we start by giving the expression of $F_{rev}(H \circ B)$ then the expression of $F_{rev}(H) \circ F_{rev}(B)$. For sake of simplicity, we use the notation F instead of F_{rev}.

The expression of $F(H \circ B)$

$H \circ B$ is a ranked graph that represents the disjoint union modulo the equivalence on nodes induced by $v(i) \approx r'(i)$, $i \in j$. So, $F(H \circ B)$ is a bigraph which has:

- $V = F(V_H \uplus V_B) = V_{F(H)} \uplus V_{F(B)}$
- $E = F(VE_H \uplus VE_B) = E_{F(H)} \uplus E_{F(B)}$
- $ctrl = F(lv_H \uplus lv_B) = ctrl_{F(H)} \uplus ctrl_{F(B)}$
- $prnt$ is determined as follows: If $w \in V_B \uplus V_H$ is a node of $H \circ B$ then $\exists!\ f_v(w) \in F(H \circ B)$ such that:

$$prnt(f_v(w)) = \begin{cases} f_v(prnt_B(w)) & \text{if } w \in V_B \text{ and } prnt_B(w) \neq r'(i), i \in j \\ f_v(prnt_H(w)) & \text{if } w \in V_H \\ f_v(v_H(i)) = f_v(prnt_H(i)) & \text{if } w \in V_B \text{ and } prnt_B(w) = r'(i), i \in j \end{cases}$$

- $link$ defined as follows: If e $\in E_B \uplus E_H$ is an edge of $H \circ B$ such that $s(e) = v$ then $\exists!$ *port* $p = f_e(e) \in F(H \circ B)$ such that:

$$link(f_e(e)) = \begin{cases} f_v(t_B(e)) & \text{if } e \in E_B \text{ and } t_B(e) \neq r'(i), i \in j \\ f_v(v_H(i)) & \text{if } e \in E_B \text{ and } t_B(e) = r'(i), i \in j \\ f_v(t_H(e)) & \text{if } e \in E_H \end{cases}$$

The expression of $F(H) \circ F(B)$

Let
$F(B) = (V_{F(B)}, E_{F(B)}, ctrl_{F(B)}, prnt_{F(B)}, link_{F(B)}) : F(i) = I \to F(j) = J$
and
$F(H) = (V_{F(H)}, E_{F(H)}, ctrl_{F(H)}, prnt_{F(H)}, link_{F(H)}) : F(j) = J \to F(k) = K$
$F(H) \circ F(B)$ is a bigraph which has:

- $V = V_{F(H)} \uplus V_{F(B)}$
- $E = E_{F(H)} \uplus E_{F(B)}$
- $ctrl = ctrl_{F(H)} \uplus ctrl_{F(B)}$
- $prnt$ is determined as follows: For $v \in V$

$$prnt(v) = \begin{cases} prnt_{F(B)}(v) & \text{if } v \in V_{F(B)} \text{ and } prnt_{F(B)}(v) \in V_{F(B)} \\ prnt_{F(H)}(v) & \text{if } v \in V_{F(H)} \\ v_{F(H)}(i) = prnt_{F(H)}(i) & \text{if } v \in V_{F(B)} \text{ and } prnt_{F(B)}(v) = j \in n \end{cases}$$

Since $v \in V_{F(B)}$ (resp. $\in V_{F(H)}$) then $\exists!\ w \in V_B$ (resp. $\in V_H$) such that $f_v(w) = v$.
Hence, for $f_v(w) \in F(H) \circ F(B)$

$$prnt(f_v(w)) = \begin{cases} prnt_{F(B)}(f_v(w)) & \text{if } f_v(w) \in V_{F(B)} \text{ and } f_v(prnt_B(w)) \in V_{F(B)} \\ prnt_{F(H)}(f_v(w)) & \text{if } f_v(w) \in V_{F(H)} \\ prnt_{F(H)}(f_v(i)) & \text{if } f_v(w) \in V_{F(B)} \text{ and } f_v(prnt_B(w)) = i \in n \end{cases}$$

Since F preserves $prnt$ mapping (i.e., $f_v \circ prnt_B = prnt_{F(B)} \circ f_v$ and $f_v \circ prnt_H = prnt_{F(H)} \circ f_v$), then:

$$prnt(f_v(w)) = \begin{cases} f_v(prnt_B(w)) & \text{if } w \in V_B \text{ and } prnt_B(w) \neq r'(i), i \in j \\ f_v(prnt_H(w)) & \text{if } w \in V_H \in V_{F(H)} \\ f_v(prnt_B(i)) & \text{if } w \in V_B \in V_{F(B)} \text{ and } prnt_B(w) = r'(i), i \in j \end{cases}$$

- $link$ is determined by: For $q \in P_{F(B)} \uplus P_{F(H)}$

$$link(p) = \begin{cases} link_{F(B)}(p) & \text{if } p \in P_{F(B)} \text{ and } link_{F(B)}(p) \in E_{F(B)} \\ link_{F(H)}(p) & \text{if } p \in P_{F(B)} \text{ and } link_{F(B)}(p) = y \in Y \\ link_{F(H)}(p) & \text{if } p \in P_{F(H)} \end{cases}$$

Since $p \in P_{F(B)}$ (resp. $\in P_{F(H)}$) then $\exists!\ e \in E_B \uplus X$ (resp. $\in E_H$) such that $f_e(e) = p$.
Hence, for $f_e(e) \in F(H) \circ F(B)$

$$link(f_e(e)) = \begin{cases} link_{F(B)}(f_e(e)) & \text{if } f_e(e) \in P_{F(B)} \text{ and } link_{F(B)}(f_e(e)) \in E_{F(B)} \\ link_{F(H)}(f_e(e)) & \text{if } f_e(e) \in P_{F(B)} \text{ and } link_{F(B)}(f_e(e)) = y \in Y \\ link_{F(H)}(f_e(e)) & \text{if } f_e(e) \in P_{F(H)} \\ \\ \end{cases}$$

Since F preserves $target$ mapping (i.e., $f_v \circ t_B = link_{F(B)} \circ f_e$ and $f_v \circ t_H = link_{F(H)} \circ f_e$), then:

$$link(f_e(e)) = \begin{cases} f_v(t_B(e)) & \text{if } e \in E_B \text{ and } t_B(e) \neq r'(i), i \in j \\ f_v(v_H(i)) & \text{if } e \in E_B \text{ and } t_B(e) = r'(i), i \in j \\ f_v(t_H(e)) & \text{if } e \in E_H \end{cases}$$