Theoretical Computer Science 577 (2015) 43–73

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Bigraphs with sharing

Michele Sevegnani [*], Muffy Calder

School of Computing Science, Sir Alwyn Williams Building, Lilybank Gardens, University of Glasgow, G12 8RZ, Glasgow, United Kingdom

ARTICLE INFO

Article history:
Received 2 August 2013
Received in revised form 27 January 2015
Accepted 4 February 2015
Available online 19 February 2015
Communicated by U. Montanari

Keywords:
Bigraphs
Matching
Spatial models
Semantics
Rewriting

ABSTRACT

Bigraphical Reactive Systems (BRS) were designed by Milner as a universal formalism for modelling systems that evolve in time, locality, co-locality and connectivity. But the underlying model of location (the place graph) is a forest, which means there is no straightforward representation of locations that can overlap or intersect. This occurs in many domains, for example in wireless signalling, social interactions and audio communications. Here, we define *bigraphs with sharing*, which solves this problem by an extension of the basic formalism: we define the place graph as a directed acyclic graph, thus allowing a natural representation of overlapping or intersecting locations. We give a complete presentation of the theory of bigraphs with sharing, including a categorical semantics, algebraic properties, and several essential procedures for computation: bigraph with sharing matching, a SAT encoding of matching, and checking a fragment of the logic **BiLog**. We show that matching is an instance of the **NP**-complete sub-graph isomorphism problem and our approach based on a SAT encoding is also efficient for standard bigraphs. We give an overview of BigraphER (Bigraph Evaluator & Rewriting), an efficient implementation of bigraphs with sharing that provides manipulation, simulation and visualisation. The matching engine is based on the SAT encoding of the matching algorithm. Examples from the 802.11 CSMA/CA RTS/CTS protocol and a network management support system illustrate the applicability of the new theory.

1. Introduction

Bigraphical Reactive Systems [1] were designed by Milner as a universal formalism for modelling interacting systems that evolve in time and space. They provide intuitive representations for systems with evolving locality, co-locality, and connectivity. In the course of working with practical applications of bigraphs for modelling different aspects of wireless networking protocols and management, we discovered there was no straightforward way to represent overlapping or intersecting spatial locations in bigraphs. But, overlaps are fundamental to wireless signalling and to other domains such as social interactions and audio communications. Consider for instance, the wireless network drawn in Fig. 1a. A spatial model based on trees cannot represent the fact that machine A is currently occupying a space location covered by both A's signal and B's signal, but not by C's signal.

We have therefore extended standard bigraphs to *bigraphs with sharing*, which allow for overlapping locations. This requires defining the place graphs of BRS as directed acyclic graphs (DAGs), instead of forests. An example place graph with sharing representing the topology of the wireless network in Fig. 1a is given in Fig. 1b. The three stations are denoted by A, B, and C, while S indicates wireless signal ranges. We will describe in detail this notation in the following section.

[*] Corresponding author.
E-mail addresses: Michele.Sevegnani@glasgow.ac.uk (M. Sevegnani), Muffy.Calder@glasgow.ac.uk (M. Calder).

http://dx.doi.org/10.1016/j.tcs.2015.02.011

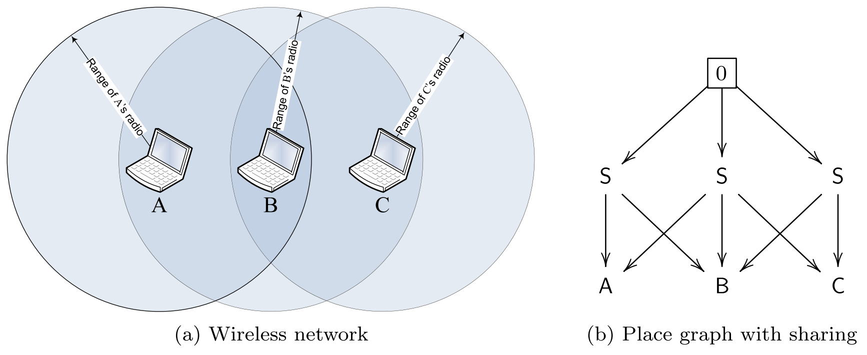


Fig. 1. Diagram of a wireless network of three stations (a) and a place graph with sharing representing its topology (b).

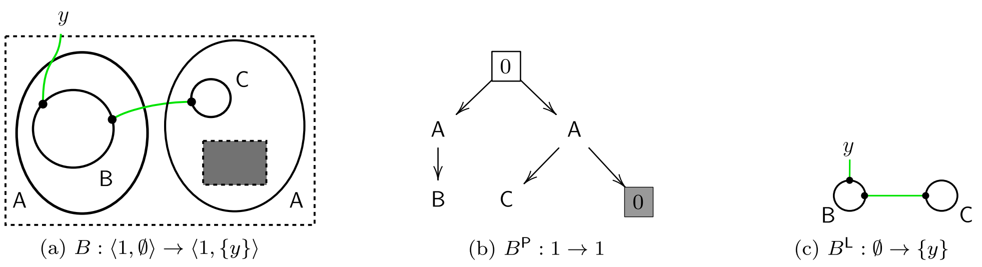

y
C
B
A
A
(a) $B : \langle 1, \emptyset\rangle \to \langle 1, \{y\}\rangle$
0
A
A
B
C
0
(b) $B^{\mathsf{P}} : 1 \to 1$
y
B
C
(c) $B^{\mathsf{L}} : \emptyset \to \{y\}$

Fig. 2. Example bigraph B (a) and its constituents: place graph B^{P} (b) and link graph B^{L} (c).

In this paper we give a complete presentation of the theory of bigraphs with sharing, including a categorical semantics, algebraic properties, and several essential procedures for computation: bigraph with sharing matching, a SAT encoding of matching, and checking a fragment of the logic **BiLog**. In each case we build on the existing results for (standard) bigraphs and our extensions are conservative in the sense that they still apply to standard bigraphs. We also give an overview of our implementation, called BigraphER, which provides an efficient implementation of computation, simulation, and visualisation for bigraphs with sharing.

When it is necessary to distinguish the different types of bigraphs, we refer to bigraphs as defined by Milner as *standard bigraphs*. We do not repeat the formal definitions here, but refer the reader to [1].

The paper is organised as follows. In the next Section we give an informal overview of bigraphs with sharing and discuss why an explicit representation of sharing is required. In Section 3 we show that the new composition and tensor product operators retain the same properties as the standard operators, and then we give a categorical semantics and results concerning algebraic forms and normalisation. In Section 4 we define a matching algorithm for bigraphs with sharing and give an overview of a SAT encoding. Our approach leads to a more efficient solution for standard bigraphs, compared with the standard approach that is based on inference. In Section 5 we show how checking a substantial fragment of the logic **BiLog** can be reduced to our matching, and in Section 6 we give an overview of BigraphER, our implementation of BRS with sharing. Finally, in Section 7 we discuss applications of bigraphs with sharing and in Sections 8 and 9 we discuss related work and our conclusions and plans for future work.

2. Bigraphs with sharing

2.1. Informal overview of standard bigraphs and bigraphs with sharing

The graphical form of an example standard bigraph B is drawn in Fig. 2a. Entities, real or virtual, are encoded by *nodes*, represented as ovals and circles. Their spatial placement is described by node nesting. Nodes are assigned a type, called *control*, denoted here by the labels A, B and C. A set of controls identifies a *signature*. Interactions between agents are represented by *links* like, for instance, the edge connecting the B-node and the C-node. Each node can have zero, one or many *ports*, indicated by bullets. Nodes of the same control have also the same number of ports. Dashed rectangles denote *regions*, also called *roots*. The rôle of a region is to specify adjacent parts of the system. Shaded squares are called *sites*. They encode parts of the system that have been abstracted away. Note that there is no significance in where a link crosses the boundary of a node in a bigraph. A bigraph can have *inner names* and *outer names*. In our example, y is an outer name. By convention in the graphical form, inner names and outer names are drawn below and above the bigraph, respectively. They encode links (or potential links) to other bigraphs representing the external environment or context. Elements forming a bigraph, namely nodes and edges, can be assigned unique identifiers, collectively called the *support* of a bigraph. When a bigraphical structure is assigned a support, it is called *concrete*.

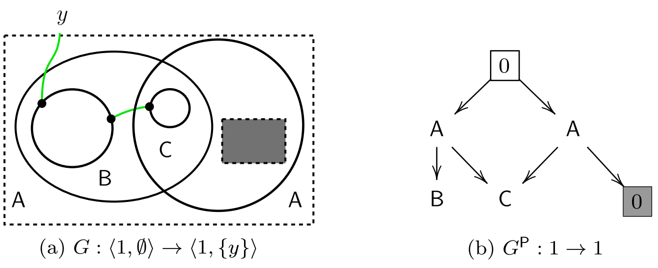

y
A
B
C
A
0
A
A
B
C
0
(a) $G : \langle 1, \emptyset \rangle \to \langle 1, \{y\} \rangle$
(b) $G^{\mathsf{P}} : 1 \to 1$

Fig. 3. Example bigraph with sharing G (a) and its place graph G^{P} (b).

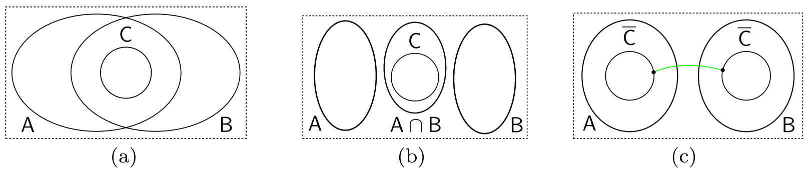


Fig. 4. An example bigraph with sharing (a) and two possible encodings (b) and (c).

Summarising, *locality* is represented by node placing while *connectivity* is encoded by the links of the bigraph. This is made explicit by defining bigraphs in terms of the constituent notions of *place graph* and *link graph*. A place graph is a forest whose roots are the regions of the corresponding bigraph and leaves are its sites and atomic nodes (i.e. nodes with no children). A link graph is a structure describing the linkage of a bigraph. It consists of a hypergraph whose vertices are the names and nodes of the corresponding bigraph and hyperedges are its links. Place graph B^{P} and link graph B^{L} of example bigraph B are drawn in Figs. 2b and 2c, respectively.

The capabilities of a bigraph to interact with the external environment are recorded in its *interface*. For example, we write $B : \langle 1, \emptyset \rangle \to \langle 1, \{y\} \rangle$ to indicate that B has one site, one region, no inner names and $\{y\}$ is its set of outer names. Informally, the *composition* of two bigraphs F (often called the *context*) and G, written $F \circ G$, is obtained by inserting G's regions in F's sites and by merging links on common names, i.e. inner names in F that are the same as outer names in G. The other fundamental operation on bigraphs is *tensor product*, denoted by $\otimes$. It is only defined over bigraphs with disjoint interfaces and it consists of putting two bigraphs side-by-side.

The dynamical theory of bigraphs is defined in terms of rewrite rules, called *reaction rules*. A reaction rule $R \longrightarrow R'$ consists of a pair of bigraphs that can be inserted into the same host bigraph. Left-hand side R (also called *redex*) specifies the pattern to be changed, while right-hand side (or *reactum*) R' specifies the changed pattern. A set of bigraphs and a set of reaction rules that define how the bigraphs can evolve over time form a *Bigraphical Reactive System (BRS)*. The *reaction relation* on bigraphs induced by the reaction rules of a BRS is written with an arrow thus: $\longrightarrow\!\triangleright$.

We have extended standard bigraphs to bigraphs *with sharing*, so that spatial locations can overlap. This induces a locality model based on directed acyclic graphs instead of forests. An example bigraph with sharing G is given in Fig. 3a. The only difference compared with standard bigraph B in Fig. 2a is that the node of control C is now placed within the intersection of the two A-nodes, i.e. C is *shared*. This can also be seen in the DAG for place graph G^{P} drawn in Fig. 3b in which two edges from A to C are present. The definition of link graph is unaffected by the introduction of sharing, thus $G^{\mathsf{L}} = B^{\mathsf{L}}$ (see Fig. 2c).

2.2. Why extend bigraphs to represent sharing explicitly

The explicit introduction of sharing is justified by considering the disadvantages of encoding sharing within standard bigraphs. We consider two possible encodings.

The first encoding consists of the introduction of dummy controls to represent intersections of nodes. For instance, if nodes A and B are overlapping, their intersection is represented as a separate node of control A ∩ B. A graphical representation is given in Fig. 4b. The immediate consequence of this approach is that place graphs are still representable by forests. However, a major disadvantage is that the number of dummy nodes to be added grows exponentially with the number of intersecting nodes. Moreover, this encoding is not complete because it cannot represent sharing when no nodes are involved. To prove this, consider the example of a C-node shared by two regions. It is possible to create a dummy region containing C, however we cannot assign a control to it. Hence, we lose track of who was sharing the node. This can be a limiting factor especially in the definition of reaction rules. Another shortcoming is that a node shared between A and B is placed inside the dummy node A ∩ B, thus both A and B appear as if they do not have a child.

The second encoding consists of keeping a copy of a shared node inside each of its parents and connecting the copies with a special link. For example, when a node C is shared between A and B, both A and B contain a node of control C̄ and the two C̄s are linked together. This is drawn in Fig. 4c. Note that control C̄ is defined exactly as C but with an extra port to handle the special link. However, this approach does not allow one to express sharing without nodes, e.g. two nodes sharing a site, and nodes or sites with no parents. Another disadvantage of this approach is that links are used to represent locality instead of connectivity, going against the spirit of bigraphs, in which the two notions are kept distinct.

Although other encodings are possible, they all present similar problems. In conclusion, our simple extension yields several advantages. First, its completeness allows for the representation of arbitrary place graphs with sharing. Second, the modelling phase is more natural and immediate, because no additional links, copies of nodes and controls have to be introduced. Third, the structure of place graphs with sharing appears to have many similarities with standard categorical notions as we will show in the next section. In particular, it makes place graphs self-dual.

2.3. Definitions

In the following we introduce formal definitions of bigraphs with sharing, their constituents and their operations. We establish some notational conventions below.

We frequently interpret a natural number as a finite ordinal, namely $m = \{0, \dots, m-1\}$. We write $S \uplus T$ to indicate the union of sets known or assumed to be disjoint. If a function f has domain S and $S' \subseteq S$, then $f \restriction S'$ denotes the restriction of f to S'. For two functions f and g with disjoint domains S and T we write $f \uplus g$ for the function with domain $S \uplus T$ such that $(f \uplus g) \restriction S = f$ and $(f \uplus g) \restriction T = g$. We write Id_S for the identity function on the set S. Given a binary relation $rel \subseteq A \times B$, we denote the domain restriction of rel over $S \subseteq A$ by $S \lhd rel$. Similarly, we write $rel \rhd S$ for the range restriction of rel over $S \subseteq B$. We also indicate set $\{b \mid (a,b) \in rel\}$, the transitive closure and the inverse of rel by writing $rel(a)$, rel^+ and rel^{-1}, respectively. In defining bigraphs we assume that names, node-identifiers, edge-identifiers and controls are drawn from four infinite sets, respectively $\mathcal{X}$, $\mathcal{V}$, $\mathcal{E}$ and $\mathcal{K}$, disjoint from each other. We denote identifiers by lower-case letter: x, y, z for names, v, u for nodes, and $e_0, e_1, \dots$ for edges. Upper-case letters $A, B, \dots$ are used to denote bigraphs and their constituents. We write $X, Y, \dots$ to indicate sets of names. We use ϵ as a shorthand for interface $\langle 0, \emptyset \rangle$.

2.3.1. Place graphs

Place graphs with sharing are DAG structures as in Fig. 3b. Their formal definition is a generalisation of the standard formulation based on forests given in [1].

Definition 1 *(Concrete place graph with sharing).* A *concrete place graph with sharing*

$$F = (V_F, ctrl_F, prnt_F) : m \to n$$

is a triple having an inner interface m and an outer interface n. These index the sites and regions of the place graph, respectively. F has a finite set $V_F \subset \mathcal{V}$ of nodes, a control map $ctrl_F : V_F \to \mathcal{K}$, and a *parent relation*

$$prnt_F \subseteq (m \uplus V_F) \times (V_F \uplus n)$$

that is acyclic i.e. $(v, v) \notin prnt_F^+$ for any $v \in V_F$.

The only difference with Milner's definition is that $prnt_F$ is now a binary relation instead of a function. This modification is sufficient to allow for sites and nodes to have zero or more parents. Special place graphs called *identities*, $\mathsf{id}_m : m \to m$, and *symmetries*, $\gamma_{m,n} : m + n \to n + m$, are defined as in standard bigraphs (see Fig. 7).

Composition and tensor product are the operations that allow combination of place graphs. They are defined as follows:

Definition 2 *(Composition for place graphs with sharing).* If $F : k \to m$ and $G : m \to n$ are two concrete place graphs with sharing with $V_F \cap V_G = \emptyset$, their composite

$$G \circ F = (V, ctrl, prnt) : k \to n$$

has nodes $V = V_F \uplus V_G$ and control map $ctrl = ctrl_F \uplus ctrl_G$. Its parent relation $prnt \subseteq (k \uplus V) \times (V \uplus n)$ is given by:

$$prnt \overset{\text{def}}{=} prnt_G^{\lhd} \uplus prnt_\circ \uplus prnt_F^{\rhd}$$

where

$$prnt_F^{\rhd} = prnt_F \rhd V_F \qquad prnt_G^{\lhd} = V_G \lhd prnt_G$$
$$prnt_\circ = (m \lhd prnt_G) \circ (prnt_F \rhd m) \ .$$

Definition 3 *(Tensor product for place graphs).* If $G_0 : m_0 \to n_0$ and $G_1 : m_1 \to n_1$ are two concrete place graphs with sharing with $V_F \cap V_G = \emptyset$, their tensor product

$$G_0 \otimes G_1 = (V, ctrl, prnt) : m_0 + m_1 \to n_0 + n_1$$

has nodes $V = V_{G_0} \uplus V_{G_1}$ and control map $ctrl \overset{\text{def}}{=} ctrl_{G_0} \uplus ctrl_{G_1}$. Its parent relation $prnt \subseteq ((m_0 + m_1) \uplus V) \times (V \uplus (n_0 + n_1))$ is defined as follows:

$$prnt_{G_0} \uplus prnt_{G_1}^{(m_0,n_0)}$$

where,

$$\begin{aligned} prnt_{G_1}^{(m_0,n_0)} = & \{(v, w) \mid (v, w) \in prnt_{G_1} \text{ and } v, w \in V_{G_1}\} \\ & \uplus \{(m_0 + i, w) \mid (i, w) \in prnt_{G_1}, w \in V_{G_1} \text{ and } i \in m_1\} \\ & \uplus \{(v, n_0 + i) \mid (v, i) \in prnt_{G_1}, v \in V_{G_1} \text{ and } i \in n_1\} \\ & \uplus \{(m_0 + i, n_0 + j) \mid (i, j) \in prnt_{G_1}, i \in m_1 \text{ and } j \in n_1\} \ . \end{aligned}$$

2.3.2. Link graphs

Here we recall the standard definition of link graphs given in [1]. The *arity* of a control K is the number of ports a K-node can have. It is indicated by $ar(\mathsf{K}) = n$.

Definition 4 *(Concrete link graph).* A *concrete link graph*

$$F = (V_F, E_F, ctrl_F, link_F) : X \to Y$$

is a quadruple having an inner face X and an outer face Y, both finite subsets of $\mathcal{X}$, called respectively the inner and outer names of the link graph. F has finite sets $V_F \subset \mathcal{V}$ of nodes and $E_F \subset \mathcal{E}$ of edges, a control map $ctrl_F : V_F \to \mathcal{K}$ and a *link map*

$$link_F : X \uplus P_F \to E_F \uplus Y$$

where $P_F \overset{\text{def}}{=} \{(v, i) | i \in ar(ctrl_F(v))\}$ is the set of ports of F. Thus (v, i) is the ith port of node v. We shall call $X \uplus P_F$ the *points* of F, and $E_F \uplus Y$ its *links*.

The sets of points and the set of ports of a link l are defined by

$$points_F(l) \overset{\text{def}}{=} \{p \mid link_F(p) = l\} \qquad ports_F(l) \overset{\text{def}}{=} points_F(l) \setminus X \ .$$

An edge is *idle* if it has no points. Identities over name sets are defined by $\mathsf{id}_X = (\emptyset, \emptyset, \emptyset, \mathsf{Id}_X) : X \to X$.

2.3.3. Bigraphs with sharing

Like a standard bigraph, a bigraph with sharing is simply the pair of its constituents: a place graph with sharing and a link graph.

Definition 5 *(Concrete bigraph with sharing).* A concrete bigraph

$$F = (V_F, E_F, ctrl_F, prnt_F, link_F) : \langle k, X\rangle \to \langle m, Y\rangle$$

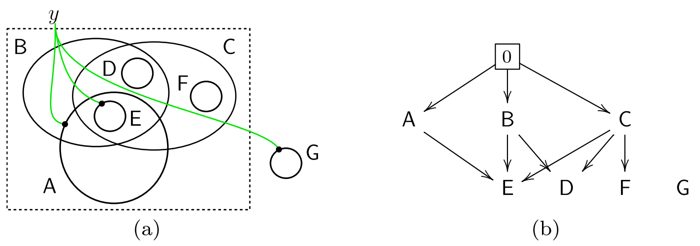
consists of a concrete place graph with sharing $F^{\mathsf{P}} = (V_F, ctrl_F, prnt_F) : k \to m$ and a concrete link graph $F^{\mathsf{L}} = (V_F, E_F, ctrl_F, link_F) : X \to Y$. We write the concrete bigraph with sharing as $F = \langle F^{\mathsf{P}}, F^{\mathsf{L}}\rangle$.

The composition of two concrete bigraphs with sharing A and B is obtained by separately composing their place graphs with sharing and link graphs. Formally, $A \circ B = \langle A^{\mathsf{P}} \circ B^{\mathsf{P}}, A^{\mathsf{L}} \circ B^{\mathsf{L}}\rangle$. The same holds for the definition of tensor product, i.e. $A \otimes B = \langle A^{\mathsf{P}} \otimes B^{\mathsf{P}}, A^{\mathsf{L}} \otimes B^{\mathsf{L}}\rangle$. The support of a concrete bigraph F, is $|F| = V_F \uplus E_F$. The identifiers of nodes and edges can be varied in a disciplined way by means of a *support translation*. Milner's definition also applies to bigraphs with sharing. We recall it here.

Definition 6 *(Support translation).* Given two bigraphs with sharing $F, G : \langle k, X\rangle \to \langle m, Y\rangle$, a support translation $\rho : |F| \to |G|$ from F to G consists of a pair of bijections $\rho_V : V_F \to V_G$ and $\rho_E : E_F \to E_G$ that respect structure as follows:

1. ρ preserves controls, i.e. $ctrl_G \circ \rho_V = ctrl_F$. It follows that ρ induces a bijection $\rho_P : P_F \to P_G$ on ports, defined by $\rho_P((v, i)) \overset{\text{def}}{=} (\rho_V(v), i)$.
2. ρ commutes with the structural maps as follows:

$$\begin{aligned} prnt_G \circ (\mathsf{Id}_m \uplus \rho_V) &= (\mathsf{Id}_n \uplus \rho_V) \circ prnt_F \\ link_G \circ (\mathsf{Id}_X \uplus \rho_P) &= (\mathsf{Id}_Y \uplus \rho_E) \circ link_F . \end{aligned}$$



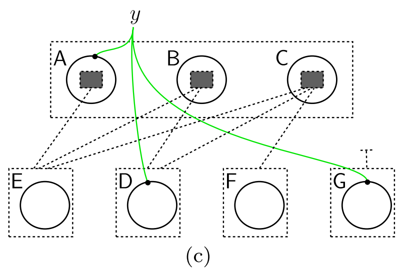


Fig. 5. Graphical forms for bigraph $B : \epsilon \to \langle 1, \{y\}\rangle$: nesting diagram (a) and stratified diagram (c). The corresponding place graph $B^{\mathsf{P}} : 0 \to 1$ is given in (b).

Given F and the bijection ρ, these conditions uniquely determine G. We therefore denote G by $\rho \bullet F$, and call it the *support translation of F by ρ*. We call F and G *support equivalent*, and we write $F \simeq G$, if such a translation exists. Support translation is defined similarly for place graphs and link graphs.

Since the definition of link graph is not affected by the introduction of sharing in the place graph, also the following standard definition applies to bigraphs with sharing:

Definition 7 *(Lean-support equivalence).* Two bigraphs with sharing F and G are *lean-support equivalent*, written $F \simeq G$, if they are support equivalent ignoring their idle edges.

An *abstract* bigraph with sharing can be viewed as a lean-support equivalence class of concrete bigraphs with sharing. Informally, a procedure to turn a concrete bigraph with sharing into an abstract bigraph with sharing, consists of discarding all its identifiers and idle edges.[1] It is also possible to do the reverse, namely from abstract to concrete bigraphs with sharing:

Definition 8 *(Concretion, abstraction).* If G is an abstract bigraph with sharing, then a concrete bigraph with sharing $\widetilde{G}$, called a *concretion* of G, is obtained by assigning to each node a unique identifier $v \in \mathcal{V}$ and to each edge a unique identifier $e \in \mathcal{E}$. Dually, bigraph G is called the *abstraction* of $\widetilde{G}$.

Observe that any two concretions $\widetilde{A}$, $\widetilde{B}$ of the same bigraph are always lean-support equivalent.

2.4. Graphical notation

Bigraphs with sharing can be represented graphically by using a modified version of the nesting diagrams for standard bigraphs in which intersecting nodes and regions are allowed.

Two examples are given in Fig. 3a and Fig. 5a. This notation is based on Venn diagrams and it is adequate to quickly specify simple models. However, it can be difficult to draw all the possible intersections when there are several (e.g. more than three) intersecting nodes. Moreover, there are other two major drawbacks:

- Empty intersections are meaningless. These often arise when there are more than two nodes overlapping. For instance, the part of the intersection between the A and C-nodes not in B (i.e. $(\mathsf{A} \cap \mathsf{C}) \setminus \mathsf{B}$) in Fig. 5a is empty.
- It is impossible to represent a node sharing a place with one of its ancestors.

To overcome these limitations, we introduce a new graphical form, called *stratified notation*. In this notation, we explicitly represent the underlying DAG structure of the placing graph by specifying how places are shared. An example stratified

[1] In our implementation, we slightly modified Milner's definition of composition for concrete link graphs in order to avoid the creation of idle edges. In this way, support equivalence is sufficient to represent abstract bigraphs.

representation of bigraph $B : \epsilon \to \langle 1, \{y\}\rangle$ is given in Fig. 5c. Note the similarities with the graphical representation of corresponding place graph $B^{\mathsf{P}} : 0 \to 1$ in Fig. 5b. In the stratified representation, nodes are organised in different layers. In the example, we have two layers. The bottom one contains the children (nodes and sites), while the top one contains their parents (nodes and regions). These are nodes D, E, F, G and A, B, C, respectively. Each node in the bottom layer is placed into a different region. This allows the nodes to be shared by different combinations of parents. Each parent node in the top layer contains a site. The nesting of the nodes is specified by the dashed edges connecting the two layers. In the example, the first region containing the D-node is connected to the sites in the B and C-nodes. This models the fact that D is shared by B and C. The orphan node G has a *place closure*, hence there are no edges connecting its region to the top layer. The representation of links is not changed. Note that with this notation meaningless intersections are not represented. In particular they do not arise because empty regions are not allowed to occur in the bottom layer. Note that when a bigraph has a more complex nesting structure, it is possible to use more than two layers.

The stratified notation is closely related to the normal form for bigraphs with sharing we will introduce in Proposition 15. Moreover, the automated drawing routines for bigraphs with sharing implemented in our tool BigraphER are based on this notation (refer to Section 6 for more details).

In the remainder of this document it is assumed that bigraphs allow sharing, unless stated otherwise. Therefore, we often write “bigraph” to mean “bigraph with sharing”.

3. Properties of bigraphs with sharing

In this section we list some properties of bigraphs with sharing. We begin by showing that composition and tensor product for place graphs with sharing enjoy the same properties of the corresponding operations for standard place graphs.

Lemma 1 *(Associativity of composition). If $A : m \to n$, $B : k \to m$, $C : h \to k$ are three concrete place graphs with sharing with disjoint node sets, then $A \circ (B \circ C) = (A \circ B) \circ C$.*

Lemma 2 *(Neutral elements for composition). For any concrete place graph with sharing $G : m \to n$, $G \circ \mathsf{id}_m = G = \mathsf{id}_n \circ G$.*

Lemma 3 *(Associativity of tensor product). If $A : m_0 \to n_0$, $B : m_1 \to n_1$, $C : m_2 \to n_2$ are three concrete place graphs with sharing with disjoint node sets, then $A \otimes (B \otimes C) = (A \otimes B) \otimes C$.*

Lemma 4 *(Neutral element for tensor product). For any concrete place graph with sharing $G : m \to n$ the following holds $G \otimes \mathsf{id}_0 = G = \mathsf{id}_0 \otimes G$.*

We now present some properties characterising composition of concrete place graphs. They will be useful for the definition of the matching algorithm for bigraphs with sharing in Section 4. The first property states that all the descendants of a node v in F are also in F.

Lemma 5. *Given two place graphs $G : m \to n$ and $F : k \to m$ such that*

$$B : k \to n = G \circ F \ .$$

For any node $v \in V_F$, if $(v', v) \in prnt_B^+$, then $v' \in (k \uplus V_F)$.

Another property specifies that in any valid composition $G \circ F$, when a node or site v in F has a parent set overlapping with G, then G provides a set of sites so that v’s parent set can be constructed in the composition. The dual property on the children set of a node or region in G also holds.

Lemma 6. *Given two place graphs $G : m \to n$ and $F : k \to m$ such that*

$$B : k \to n = G \circ F \ .$$

For any node or site $v \in (k \uplus V_F)$ such that $(prnt_B \rhd (V_G \uplus n))(v) = P$, with $P \neq \emptyset$, there exists a set $S \subseteq m$ such that

$$(prnt_F \rhd m)(v) = S \qquad \bigcup_{s' \in S} prnt_G(s') = P$$

Lemma 7. *Given two place graphs $G : m \to n$ and $F : k \to m$ such that*

$$B : k \to n = G \circ F \ .$$

For any node or region $v \in (V_G \uplus n)$ such that $(prnt_B^{-1} \rhd (k \uplus V_F))(v) = C$, with $C \neq \emptyset$, there exists a set $R \subseteq m$ such that

$$(prnt_G^{-1} \rhd m)(v) = R \qquad \bigcup_{r' \in R} prnt_F^{-1}(r') = C$$

3.1. Categorical semantics

Bigraphs with sharing and their operations can be defined in the general framework of category theory. Let us recall here the definitions of two non-standard kinds of categories introduced by Milner in [1] to define standard abstract and concrete bigraphs within a categorical semantics.

Definition 9 *(Spm category).* A partial monoidal category is *symmetric (spm)* if, whenever $X \otimes Y$ is defined, there is an arrow $\gamma_{X,Y} : X \otimes Y \to Y \otimes X$ called a *symmetry*, satisfying the following equations when the compositions and products are defined:

1. $\gamma_{X,\epsilon} = \mathsf{id}_X$
2. $\gamma_{Y,X} \circ \gamma_{X,Y} = \mathsf{id}_{X \otimes Y}$
3. $\gamma_{X_1,Y_1} \circ (f \otimes g) = (g \otimes f) \circ \gamma_{X_0,Y_0}$ for $f : X_0 \to X_1$ and $g : Y_0 \to Y_1$
4. $\gamma_{X \otimes Y, Z} = (\gamma_{X,Z} \otimes \mathsf{id}_Y) \circ (\mathsf{id}_X \otimes \gamma_{Y,Z})$

A functor between spm categories preserves unit, product and symmetries.

Definition 10 *(s-category).* An *s-category* $\widetilde{\mathbf{C}}$ is a precategory in which each arrow f is assigned a finite *support* $|f| \subset \mathcal{S}$. Further, $\widetilde{\mathbf{C}}$ possesses a partial tensor product, unit and symmetries, as in an spm category. The identities id_I and symmetries $\gamma_{I,J}$ are assigned empty support. In addition:

- For $f : I \to J$ and $g : J' \to K$, the composition $g \circ f$ is defined iff $J = J'$ and $|f| \cap |g| = \emptyset$; then $|g \circ f| = |g| \uplus |f|$.
- For $f : I_0 \to I_1$ and $g : J_0 \to J_1$, the tensor product $f \otimes g$ is defined iff $I_i \otimes J_i$ is defined $(i = 0, 1)$ and $|f| \cap |g| = \emptyset$; then $|f \otimes g| = |f| \uplus |g|$.

We adopt the following notational conventions. Given a signature $\mathcal{K}$, we write $\widetilde{\mathbf{Bg}}(\mathcal{K})$ and $\mathbf{Bg}(\mathcal{K})$ to indicate the precategory of concrete bigraphs and the category of abstract bigraphs, respectively. Similarly, $\widetilde{\mathbf{Pg}}(\mathcal{K})$ ($\widetilde{\mathbf{Lg}}(\mathcal{K})$) and $\mathbf{Pg}(\mathcal{K})$ ($\mathbf{Lg}(\mathcal{K})$) denote the precategory of concrete place (link) graphs and the category of abstract place (link) graphs, respectively.

We begin by showing that concrete place graphs with sharing form an s-category. To do so, we have to interpret them as a precategory.

Proposition 8 *(Precategory of concrete place graphs with sharing). Given a signature $\mathcal{K}$, $\widetilde{\mathbf{SPg}}(\mathcal{K})$ is the precategory whose arrows are concrete place graphs with sharing as given in Definition 1 and objects are finite ordinals. Identities are place graphs $\mathsf{id}_m : m \to m$ and composition is set out in Definition 2.*

Before presenting the main result, we prove bifunctoriality of tensor product for concrete place graphs with sharing.

Lemma 9 *(Bifunctoriality 1). If $A_0 : n_0 \to n_1$, $A_1 : n_1 \to n_2$, $B_0 : m_0 \to m_1$ and $B_1 : m_1 \to m_2$ are four concrete place graphs with sharing with disjoint node sets, then $(A_1 \otimes B_1) \circ (A_0 \otimes B_0) = (A_1 \circ A_0) \otimes (B_1 \circ B_0)$.*

Lemma 10 *(Bifunctoriality 2). If id_m and id_n are two place graphs with sharing then $\mathsf{id}_m \otimes \mathsf{id}_n = \mathsf{id}_{m \otimes n}$.*

We now can cast concrete place graphs with sharing to s-categories:

Proposition 11. *Given a signature $\mathcal{K}$, precategory $\widetilde{\mathbf{SPg}}(\mathcal{K})$ equipped with a finite support V_F for every arrow $F : m \to n$, a partial tensor product like in Definition 3 and symmetries $\gamma_{m,n}$, is an s-category.*

We proved that concrete place graphs with sharing belong to the same kind of category of $\widetilde{\mathbf{Pg}}(\mathcal{K})$. Therefore, $\widetilde{\mathbf{SPg}}(\mathcal{K})$ together with $\widetilde{\mathbf{Lg}}(\mathcal{K})$ can be used to form s-category $\widetilde{\mathbf{SBg}}(\mathcal{K})$ of concrete bigraphs with sharing. The spm category of abstract bigraphs with sharing is then obtained by applying a lean-support quotient functor $[\cdot] : \widetilde{\mathbf{SBg}}(\mathcal{K}) \to \mathbf{SBg}(\mathcal{K})$ defined as in standard bigraphs. Usual projection functors like $\mathfrak{F}^{\mathsf{P}} : \mathbf{SBg}(\mathcal{K}) \to \mathbf{SPg}(\mathcal{K})$ and $\mathfrak{F}^{\mathsf{L}} : \mathbf{SBg}(\mathcal{K}) \to \mathbf{Lg}(\mathcal{K})$ can also be defined by dropping the link graph and the place graph, respectively. Since functions are binary relations, $\mathbf{Pg}(\mathcal{K})$ is a sub-category of $\mathbf{SPg}(\mathcal{K})$. This relationship between the two categories is made explicit by inclusion functor $\mathfrak{I} : \mathbf{Pg}(\mathcal{K}) \to \mathbf{SPg}(\mathcal{K})$. Following the same argument, $\widetilde{\mathbf{Pg}}(\mathcal{K})$ is a sub-category of $\widetilde{\mathbf{SPg}}(\mathcal{K})$ with $\mathfrak{I}'$ acting as inclusion functor. The relationships between $\mathbf{SPg}(\mathcal{K})$ and other categories are summarised in Fig. 6. By an abuse of notation, we write $\mathfrak{I}$, $\mathfrak{I}'$ to indicate inclusion functors, $\mathfrak{F}^{\mathsf{P}}$, $\mathfrak{F}'^{\mathsf{P}}$ for functors projecting into place graphs, and $[\cdot]$ to denote lean-support quotient functors.

We now show that bigraphs with sharing form a wide category, i.e. there exists a functor width which allows to interpret any bigraph with sharing as a function between finite ordinals. First, we define functor $\mathfrak{W} : \mathbf{SPg}(\mathcal{K}) \to \mathbf{Rel}$ to move from bigraphs with sharing to the category of binary relations between ordinals. It acts as the identity on objects and is defined on every arrow $G^{\mathsf{P}} : m \to n$ as follows:

$$\mathfrak{W}(G) = \{(i, j) \mid i \in m \wedge j \in n \wedge (i, j) \in prnt_G^{+}\} \ .$$

$$\begin{tikzcd}
\widetilde{\mathbf{Bg}}(\mathcal{K}) \arrow[rr, "{[\cdot]}"] \arrow[dr, "\mathfrak{F}'^{\mathsf{P}}"] \arrow[dd, "\mathfrak{I}'"'] & & \mathbf{Bg}(\mathcal{K}) \arrow[dr, "\mathfrak{F}^{\mathsf{P}}"] \arrow[dd, "\mathfrak{I}"] & & \\
& \widetilde{\mathbf{Pg}}(\mathcal{K}) \arrow[rr, "{[\cdot]}"] \arrow[dd, "\mathfrak{I}'"] & & \mathbf{Pg}(\mathcal{K}) \arrow[r, "\mathsf{width}"] \arrow[dd, "\mathfrak{I}"] & \mathbf{Finord} \\
\widetilde{\mathbf{SBg}}(\mathcal{K}) \arrow[rr, "{[\cdot]}"] \arrow[dr, "\mathfrak{F}'^{\mathsf{P}}"'] & & \mathbf{SBg}(\mathcal{K}) \arrow[dr, "\mathfrak{F}^{\mathsf{P}}"'] & & \\
& \widetilde{\mathbf{SPg}}(\mathcal{K}) \arrow[rr, "{[\cdot]}"'] & & \mathbf{SPg}(\mathcal{K}) \arrow[r, "\mathfrak{W}"'] \arrow[ur, "\mathsf{width}"] & \mathbf{Rel}
\end{tikzcd}$$

Fig. 6. Bigraphical categories.

Note that **Rel** corresponds to the class of node-free place graphs with sharing. Second, we define functor width to move directly to **Finord** by mapping every site of a bigraph with sharing to the set of its region ancestors. In more detail, for every arrow $G^{\mathsf{P}} : m \to n$ and for every site $i \in m$ we define

$$\mathsf{width}(G)(i) = j \text{ such that } j \in 2^n \text{ and } \forall x \in j.(i,x) \in prnt_G^+ \ .$$

A property enjoyed by $\mathbf{SPg}(\mathcal{K})$ is *self-duality*. This can be proved by showing that the inverse of any arrow is still a place graph with sharing. In more detail, for every arrow $G : m \to n$ we can construct its inverse arrow $G^{\mathsf{op}} : n \to m$ in which sites and regions are swapped, $prnt_{G^{\mathsf{op}}} = prnt_G^{-1}$, and G^{op} is itself an arrow of $\mathbf{SPg}(\mathcal{K})$. Note that place graphs without sharing are not self-dual. For example, if we try to construct the inverse of $join : 2 \to 1$, we obtain $prnt^{-1}(0) = 0$ and $prnt^{-1}(0) = 1$, which is not a function. Therefore $join^{\mathsf{op}}$ is not an arrow of $\mathbf{Pg}(\mathcal{K})$.

Finally, we characterise epimorphisms (epis) and monomorphisms (monos) in $\mathbf{SPg}(\mathcal{K})$. There are two considerations that help understanding their definitions. The first one is that epis are the inverse arrows of monos, and vice versa, due to self-duality of place graphs with sharing. Note that this is not the case in standard bigraphs. The second consideration is that epis (monos) in standard place graphs are also epis (monos) in place graphs with sharing, since $\mathbf{Pg}(\mathcal{K})$ is a sub-category of $\mathbf{SPg}(\mathcal{K})$.

Let us introduce some terminology. A region is *idle* if it has no children whereas a site is an *orphan* if it has no parents. Two sites with the same parents are called *siblings* and, dually, two regions with the same children are called *partners*.

Proposition 12 *(Epis for place graphs with sharing). A concrete place graph with sharing is epi iff no region is idle and no two regions are partners.*

Proposition 13 *(Monos for place graphs with sharing). A concrete place graph with sharing is mono iff no two sites are siblings and no site is an orphan.*

Epis and monos in $\mathbf{SBg}(\mathcal{K})$ are the arrows in which both the place graph and the link graph are epi and mono, respectively.

3.2. Algebra and normal form

We now show that bigraphs with sharing can be expressed by means of composition and tensor product starting from a minimal set of elementary building blocks. This results in an algebraic form similar to the one for standard bigraphs introduced by Milner in [1]. Also in bigraphs with sharing, nodes are introduced by only one kind of elementary bigraph called *ions*. For each control K with $ar(\mathsf{K}) = n$, an ion consists of a single K-node that is within one region, contains a site and has ports linked bijectively to n distinct names $\vec{x}$. Notation $\mathsf{K}_{\vec{x}} : 1 \to \langle 1, \{\vec{x}\}\rangle$ expresses this bigraph. The corresponding elementary place graph is indicated by $\mathsf{K} : 1 \to 1$. Two additional elementary place graphs are required: $0 : 1 \to 0$ and $split : 1 \to 2$. They are the dual bigraphs of $1 : 0 \to 1$ and $join : 2 \to 1$, respectively. They are essential to represent orphans and shared places. The entire set of elementary place graphs is summarised in Fig. 7.

Node-free place graphs with sharing, again called placings, are ranged over by $\phi, \psi, \ldots$. A placing with one region and n sites is denoted by $merge_n$, where $merge_0 = 1$, $merge_1 = \mathsf{id}_1$ and $merge_2 = join$. Dually, we indicate a placing with one site and n regions by $share_n$. Note that $share_0 = 0$, $share_1 = \mathsf{id}_1$ and $share_2 = split$.

In the following, B, G, ω, ψ, etc. are treated as expressions for bigraphs, but equality '=' is semantical, i.e. $B = B'$ means that B and B' denote the same bigraph and not that they are identical expressions. Formally, this is a shorthand for the more verbose $\models B = B'$.

Proposition 14. *Every place graph with sharing can be expressed as an expression containing only the elementary place graphs given in Fig. 7 as constants and composition and tensor product as operators.*

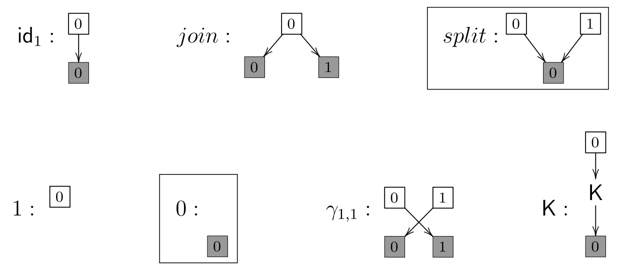


Fig. 7. Elementary place graphs. The additional elementary place graphs required to express sharing are *split* : $1 \to 2$ and $0 : 1 \to 0$.

Proof. We prove the proposition by induction on the number of nodes of the place graph with sharing. For the base case we show that node-free place graphs, i.e. placings, satisfy the proposition. A placing $\psi : m \to n$ may be expressed as

$$\psi = (merge_{n_0} \otimes \cdots \otimes merge_{n_{n-1}})\,\pi\,(share_{m_0} \otimes \cdots \otimes share_{m_{m-1}}) \tag{1}$$

where $\sum_i m_i = m$, $\sum_i n_i = n'$ and $\pi : n' \to n'$ is a permutation. Since a permutation may be generated by composition and tensor product of symmetries $\gamma_{m,n}$, the proposition holds.

Now, let $B : m \to n$ be a place graph with sharing with $k+1$ nodes. Then, it has a concretion $\widetilde{B} : m \to n$ with support V. Let $v \in V$ be a node in which none of its children are nodes (we call it a *leaf*). Such a node must exist by acyclicity of $prnt_B$. Note that v can still have $m' \leq m$ sites as children. Formally, $(u, v) \in prnt_B$ if and only if $u \in m'$. Without loss of generality we assume $ctrl(v) = \mathsf{K}$. Let $\widetilde{B}_1 : m+1 \to n$ be the place graph obtained from $\widetilde{B}$ by removing node v and substituting it with a site. Furthermore, let $\widetilde{B}_0 : m \to m+1$ be the place graph containing the sites in $\widetilde{B}$ and v. Then we have that $\widetilde{B} = \widetilde{B}_1 \circ \widetilde{B}_0$. By dropping the supports we can write $B = B_1 \circ B_0$. But B_0 can be defined in terms of elementary place graphs as follows:

$$B_0 = \psi \circ (\mathsf{id}_m \otimes \mathsf{K}) \circ \phi \quad \text{with} \quad \psi : m+1 \to m+1 \quad \phi : m \to m+1 \ .$$

Therefore, the statement follows by inductive hypothesis on B_1 since it has k nodes. □

The construction presented in the proof above can be adopted to define a normal form for place graphs with sharing. Intuitively, a place graph with sharing B can be expressed by iteratively applying the procedure for the construction of B_0, until all its nodes are consumed. The only difference is that all the leafs are removed in one go instead of removing only a single leaf at each step. We make this precise by introducing the notion of *normalised levels*.

Definition 11 *(Level).* A *level* $L : m \to n$ is a place graph with sharing that can be expressed uniquely, up to permutations, as

$$L = (\bigotimes_{i<n-l} \mathsf{K}_i \otimes \mathsf{id}_l) \circ \phi \ ,$$

with placing $\phi : m \to n$ expressed as in (1).

Definition 12 *(Normalised levels).* Take two levels $L_1 : m_1 \to m_2$ and $L_0 : m_0 \to m_1$ given by

$$L_1 = (K_1 \otimes \mathsf{K}' \otimes \mathsf{id}_{l_1}) \circ \phi_1 \qquad L_0 = (K_0 \otimes \mathsf{id}_{l_0}) \circ \phi_0$$
$$K_1 = \bigotimes_{i<m_2-l_1-1} \mathsf{K}_i \qquad K_0 = \bigotimes_{i<m_1-l_0} \mathsf{K}_i$$

with $\phi_0 : m_0 \to m_1$ and $\phi_1 : m_1 \to m_2$ placings. We say that L_1 and L_0 are *normalised* if

$$((K_1 \otimes \mathsf{id}_{l_1+1}) \circ \phi_1') \circ ((K_0 \otimes \mathsf{K}' \otimes \mathsf{id}_{l_0}) \circ \phi_0') \neq L_1 \circ L_0 \ ,$$

for any K' and any placings

$$\phi_0' : m_0 \to m_1 + 1 \qquad \phi_1' : m_1 + 1 \to m_2 \ .$$

In words, this means that K' is not a leaf. Thus, it cannot be pushed down a level and L_0 is already a maximal set of leafs in $L_1 \circ L_0$. When considering several levels $L_n \circ \cdots \circ L_0$, we say that the levels are normalised if every pair of adjacent levels $L_i \circ L_{i-1}$ is normalised. We can now define a *stratified normal form* (SNF) for place graphs with sharing.

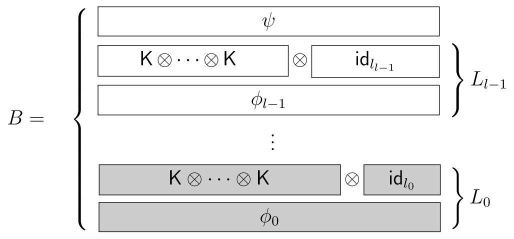


Fig. 8. Graphical representation of SNF for place graph with sharing B.

Proposition 15 *(SNF for place graphs with sharing). Every place graph with sharing $B : m \to n$ can be expressed uniquely, up to permutations, as*

$$B = \psi \circ L_{l-1} \circ \cdots \circ L_0$$

where ψ is a placing and each L_i is a normalised level.

A graphical explanation of the definition is shown in Fig. 8.

Example 1. We now use SNF to express place graph $B^{\mathsf{P}} : 0 \to 1$ given in Fig. 5b:

$$\begin{aligned}
B^{\mathsf{P}} &= merge_3 \circ L_1 \circ L_0 \\
L_1 &= (\mathsf{A} \otimes \mathsf{B} \otimes \mathsf{C} \otimes \mathsf{id}_0) \circ \phi \\
\phi &= (\mathsf{id}_1 \otimes merge_2 \otimes merge_3) \circ (\mathsf{id}_2 \otimes \gamma_{1,1} \otimes \mathsf{id}_2) \circ (share_3 \otimes share_2 \otimes \mathsf{id}_1 \otimes 0) \\
L_0 &= (\mathsf{E} \otimes \mathsf{D} \otimes \mathsf{F} \otimes \mathsf{G} \otimes \mathsf{id}_0) \circ (1 \otimes 1 \otimes 1 \otimes 1) \ .
\end{aligned}$$

This normal form can also be extended to represent bigraphs with sharing. Since nodes are the only structure in common between link graphs and place graphs, it suffices to replace the node generator described above with the ion $\mathsf{K}_{\vec{x}} : 1 \to \langle 1, \{\vec{x}\}\rangle$ defined for standard bigraphs. Hence, *extended levels* $L : m \to \langle n, X\rangle$ are defined as follows:

$$L = (\bigotimes_{i<n-l} \mathsf{K}_{\{\vec{x}_i\}} \otimes \mathsf{id}_l) \circ \phi \ \ ,$$

where $X = \biguplus_i \{\vec{x}_i\}$. Since normalisation is defined over place graphs, the definition of normalised levels is not affected by the introduction of ions.

Proposition 16 *(SNF for bigraphs with sharing). Every bigraph with sharing $G : \langle m, X\rangle \to \langle n, Y\rangle$ may be expressed, up to permutations and renaming, as*

$$\begin{aligned}
G &= (\psi \otimes \omega) \circ S_{l-1} \circ \cdots \circ S_0 \\
S_i &= L_i \otimes \mathsf{id}_{X_i}
\end{aligned}$$

where ψ is a placing with outer interface n, ω is a linking with outer interface Y, and each L_i is a normalised extended level. Also the inner face of L_0 is m and $X_0 = X$.

The algebraic operators defined for standard bigraphs (e.g. merge product $(\_ \parallel \_)$, parallel product $(\_ \mid \_)$ and nesting $(\_ . \_)$) can also be used to express bigraphs with sharing. However, an additional operator capable of expressing sharing is required. We define *share expressions* as

$$\text{share } F \text{ by } \phi \text{ in } G \overset{\text{def}}{=} G \circ (\phi \otimes \mathsf{id}_X) \circ F$$

where ϕ is a placing and all the operations are assumed defined.

3.2.1. Axioms for bigraphical equality

We now introduce a set of axioms that allow proof of every valid equation between bigraphical expressions. We write $\vdash B = B'$ to indicate equality inferred from the axioms. Recall that the existence of functor $\mathfrak{W} : \mathbf{SPg}(\mathcal{K}) \to \mathbf{Rel}$ assures that any placing $\phi : m \to n$ is also a relation between finite ordinals m and n. Thus, ϕ can also be viewed as an arrow in category **Rel**. In [2, Theorem 7], Mimram proves that a *complete axiomatisation* of **Rel** is given by the equational theory of qualitative bicommutative bialgebrae. This allows us to define a complete axiomatisation for placings with sharing as a superset of

Table 1
Axioms for (standard) place graph equality.

CATEGORICAL AXIOMS:	
$A \circ \mathsf{id} = A = \mathsf{id} \circ A$	(CAT1)
$A \circ (B \circ C) = (A \circ B) \circ C$	(CAT2)
$A \otimes \mathsf{id}_\epsilon = A = \mathsf{id}_\epsilon \otimes A$	(CAT3)
$A \otimes (B \otimes C) = (A \otimes B) \otimes C$	(CAT4)
$\mathsf{id}_I \otimes \mathsf{id}_J = \mathsf{id}_{I \otimes J}$	(CAT5)
$(A_1 \circ A_0) \otimes (B_1 \circ B_0) = (A_1 \otimes B_1) \circ (A_0 \otimes B_0)$	(CAT6)
$\gamma_{I,\epsilon} = \mathsf{id}_I$	(CAT7)
$\gamma_{J,I} \circ \gamma_{I,J} = \mathsf{id}_{I \otimes J}$	(CAT8)
$\gamma_{I,K} \circ (A \otimes B) = (B \otimes A) \circ \gamma_{H,J} \qquad (A : H \to I, B : J \to K)$	(CAT9)
PLACE AXIOMS:	
$join \circ (join \otimes \mathsf{id}_1) = join \circ (\mathsf{id}_1 \otimes join)$	(PLC1)
$join \circ (1 \otimes \mathsf{id}_1) = \mathsf{id}_1$	(PLC2)
$join \circ \gamma_{1,1} = join$	(PLC3)

Table 2
Additional axioms for place graphs with sharing equality.

CO-MONOID AXIOMS:	
$(split \otimes \mathsf{id}_1) \circ split = (\mathsf{id}_1 \otimes split) \circ split$	(CO-PLC1)
$(0 \otimes \mathsf{id}_1) \circ split = \mathsf{id}_1$	(CO-PLC1)
$\gamma_{1,1} \circ split = split$	(CO-PLC1)
BIALGEBRA AXIOMS:	
$split \circ join = (join \otimes join) \circ (\mathsf{id}_1 \otimes \gamma_{1,1} \otimes \mathsf{id}_1) \circ (split \otimes split)$	(BIA1)
$0 \circ join = 0 \otimes 0$	(BIA2)
$split \circ 1 = 1 \otimes 1$	(BIA3)
$0 \circ 1 = \mathsf{id}_0$	(BIA4)
$join \circ split = \mathsf{id}_1$	(BIA5)

the axioms for $\mathbf{Pg}(\mathcal{K})$ introduced by Milner. These are reported in Table 1. The additional axioms that are required to obtain completeness over place graphs with sharing form the qualitative bialgebra $(\{0\}, join, 1, split, 0, \gamma_{1,1})$. They are given in Table 2.

Proposition 17 *(Completeness). The set of axioms in Tables 1 and 2 is complete for expressions denoting place graphs with sharing. Formally, $B = B'$ implies $\vdash B = B'$ for all expressions B, B'.*

Proof (Outline). First, we show that $\vdash B = E$ and $\vdash B' = E'$, where E and E' are SNF expressions. Observe that Proposition 15 assures that such expressions exist for any place graph with sharing. Second, we show that the two normal forms can be proven equal, i.e. $\vdash E = E'$. This amounts to showing that the two expressions only differ by permutations of the ions in the levels:

$$\vdash \overbrace{\psi \circ L_{l-1} \circ \cdots \circ L_0}^{E} = \overbrace{\psi' \circ L'_{l-1} \circ \cdots \circ L'_0}^{E'}$$

$$\vdash \psi \circ \overbrace{(\bigotimes_i \mathsf{K}_i \otimes \mathsf{id}_l) \circ \phi}^{L_{l-1}} \circ \cdots \circ L_0 = \psi' \circ \overbrace{(\bigotimes_i \mathsf{K}_{\pi(i)} \otimes \mathsf{id}_l) \circ \phi'}^{L'_{l-1}} \circ \cdots \circ L'_0$$

Table 3
Axioms for link graph equality.

LINK AXIOMS:	
$^{x}/x = \mathsf{id}_x$	(LNK1)
$^{z}/(Y \uplus y) \circ (\mathsf{id}_Y \otimes {}^{y}/X) = {}^{z}/Y \uplus X$	(LNK2)
$/y \circ {}^{y}/x = /x$	(LNK3)
$/y \circ {}^{y}/ = \mathsf{id}_\epsilon$	(LNK4)
NODE AXIOM:	
$(\mathsf{id}_1 \otimes \alpha) \circ \mathsf{K}_{\vec{x}} = \mathsf{K}_{\alpha(\vec{x})}$	(NODE)

$$\vdash E = \overbrace{\psi' \circ (\pi \otimes \mathsf{id}_I)}^{\psi} \circ (\bigotimes_i \mathsf{K}_i \otimes \mathsf{id}_I) \circ (\pi \otimes \mathsf{id}_I) \circ \phi' \circ \cdots \circ L'_0$$

$$\vdots$$

where equality between expressions for placings (e.g. $\vdash \psi = \psi' \circ (\pi \otimes \mathsf{id}_I)$) can be proved by completeness of the axioms in Tables 1 and 2 for expressions of placings with sharing. □

Example 2. Consider the following expressions of placings with sharing:

$$\phi = (\mathsf{id}_1 \otimes join) \circ (split \otimes \mathsf{id}_1)$$
$$\phi' = (join \otimes join) \circ (share_3 \otimes \mathsf{id}_1) \ .$$

We now show how the additional axioms in Table 2 are required to prove equality between ϕ and ϕ':

$\vdash \phi = \phi'$	
$\vdash \phi = (join \otimes join) \circ (((split \otimes \mathsf{id}_1) \circ split) \otimes \mathsf{id}_1)$	by definition of $share_3$
$\vdash \phi = (join \otimes join) \circ (((split \otimes \mathsf{id}_1) \circ split) \otimes (\mathsf{id}_1 \circ \mathsf{id}_1))$	by (CAT1)
$\vdash \phi = (join \otimes join) \circ (split \otimes \mathsf{id}_1 \otimes \mathsf{id}_1) \circ (split \otimes \mathsf{id}_1)$	by (CAT6)
$\vdash \phi = (join \otimes join) \circ (split \otimes \mathsf{id}_2) \circ (split \otimes \mathsf{id}_1)$	by (CAT5)
$\vdash \phi = ((join \circ split) \otimes (join \circ \mathsf{id}_2)) \circ (split \otimes \mathsf{id}_1)$	by (CAT6)
$\vdash \phi = ((join \circ split) \otimes join) \circ (split \otimes \mathsf{id}_1)$	by (CAT1)

At this point, the only way to reduce the right-hand side is by invoking an additional axiom as follows:

$$\vdash \phi = (\mathsf{id}_1 \otimes join) \circ (split \otimes \mathsf{id}_1) \qquad \text{by (BIA5).}$$

This result can be extended to **SBg**($\mathcal{K}$) by including Milner's axioms for expressions over linkings given in Table 3. Recall that elementary *closures* and elementary *substitutions* are denoted by $/x : \{x\} \to \emptyset$ and ${}^{y}/X : X \to \{y\}$, respectively.

4. Matching of bigraphs with sharing

The bigraph matching problem is a computational task in which a bigraph P, called *pattern*, and a bigraph T, called *target*, are given as input, and one must determine whether P occurs in T. This problem plays a fundamental rôle in the definition of BRS as an instance of the problem arises every time a reaction rule is applied, in particular when the occurrences of a redex need to be computed. Moreover, we will show in Section 5 that it is possible to reduce the verification of predicates expressed in a fragment of **BiLog** to one or more instances of bigraph matching. Let us formally define an occurrence as follows:

Definition 13 *(Occurrence, matching).* A bigraph P *occurs* in a bigraph T if the equation $T = C \circ (P \otimes \mathsf{id}_I) \circ D$ holds for some interface $I = \langle m, X\rangle$ and bigraphs C and D. Two occurrences are defined to be the same if they differ only by a permutation on C's sites and D's roots or by a bijective renaming on the inner names of C and the outer names of D. We say that P *matches* T or P is a *match* in T when P occurs in T.

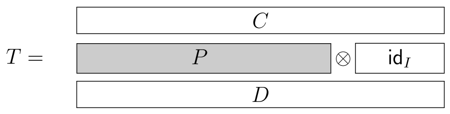


Fig. 9. Match $T = C \circ (P \otimes \mathsf{id}_I) \circ D$.

Identifying matches involving renamings and permutations on the interfaces is essential to count correctly the number of matches. Further, in a stochastic setting this allows for the computation of stochastic rates [3]. We call bigraphs C and D the *context* and the *parameter* (of P), respectively. The decomposition of a bigraph T induced by occurrence P can be described conveniently by means of diagrams as in Fig. 9. Observe that identity id_I is necessary to allow nodes in C to have children also in D, and to allow C and D to share links that do not involve P. This is the only difference with the definition of matching for standard bigraphs introduced by Milner, in which $I = \langle 0, X \rangle$.

When considering concrete bigraphs, the supports of $\widetilde{T}$ and $\widetilde{P}$ have to be compatible in order to find a valid occurrence.

Definition 14 *(Concrete occurrence).* Let $\widetilde{P}$ and $\widetilde{T}$ be two concrete bigraphs. We say there is a *concrete occurrence* of $\widetilde{P}$ in $\widetilde{T}$ if P occurs in T, with P and T the abstractions of $\widetilde{P}$ and $\widetilde{T}$, respectively.

The definition above implies that when $\widetilde{P}$ occurs in $\widetilde{T}$, there exists a support translation ρ such that $\widetilde{T} = \widetilde{C} \circ (\widetilde{P'} \otimes \mathsf{id}_I) \circ \widetilde{D}$ and $\widetilde{P'} = \rho \bullet \widetilde{P}$. Moreover, $\simeq$-equivalent occurrences are regarded as the same. The decomposition induced by an occurrence can be described by a pair (ι, η), in which the first element is a support translation from V_P to $V_{P'} \subseteq V_T$ (i.e. $\iota = \rho_V$) and η is a mapping from the links of P to the links of T. Note that $\rho_E \subseteq \eta$ because unlike ρ_E, η maps both outer names and edges.

4.1. Algorithm

We now present a graph-theoretic matching algorithm for concrete bigraphs with sharing, after introducing some definitions and notational conventions. Equivalent links are defined as follows:

Definition 15. Take two links l, k in a bigraph $F : \langle m, X \rangle \to \langle n, Y \rangle$. Equivalence relation $\doteq$ is defined as follows

$$l \doteq k \qquad \text{iff} \qquad \mathit{count}(l) = \mathit{count}(k) \text{ and } \tau(l) = \tau(k)$$

where

$$\tau(l) = \begin{cases} \mathsf{name} & \text{if } l \in Y \\ \mathsf{edge} & \text{if } l \in E_F \text{ and } \mathit{points}(l) \subseteq P_F \\ \mathsf{open\text{-}edge} & \text{otherwise} \end{cases}$$

and $\mathit{count}(l) \overset{\text{def}}{=} \{\{v \mid (v, i) \in \mathit{ports}(l)\}\}$ is a multiset.

We also define an explicit transformation from concrete place graphs to DAGs as follows.

Definition 16 *(Underlying graph).* Let $F = (V_F, \mathit{ctrl}_F, \mathit{prnt}_F) : m \to n$ be a concrete place graph. Directed graph $\mathcal{G}_F = (V, E)$ is called the *underlying graph* of F. The set of nodes is $V = V_F$ and the set of edges is $E = V_F, V_F \lhd \mathit{prnt}_F \rhd V_F$.

The transformation consists of dropping all the roots and sites from bigraph F. An important property is that any control-preserving graph isomorphism ι between two underlying graphs is a support translation ρ_V between the corresponding place graphs when sites and roots are ignored. Formally, $\iota : \mathcal{G}_F \to \mathcal{G}_G$ is a bijection such that $\mathit{ctrl}_F(u) = \mathit{ctrl}_G(\iota(u))$, $\mathit{ctrl}_F(v) = \mathit{ctrl}_G(\iota(v))$ and

$$(u, v) \in \mathit{prnt}_F \qquad \text{iff} \qquad (\iota(u), \iota(v)) \in \mathit{prnt}_G \ .$$

This is equivalent to writing

$$(V_G \lhd \mathit{prnt}_G \rhd V_G) \circ \iota = \iota \circ (V_F \lhd \mathit{prnt}_F \rhd V_F) \ .$$

Hence, ι satisfies Definition 6 when roots and sites are ignored. This correspondence between ι and ρ_V motivates the reduction of the matching problem to the sub-graph isomorphism problem.

Graph isomorphisms and maps between links are indicated by ι and η, respectively. We write $\widehat{\mathcal{G}_T}$ to denote the sub-graph of $\mathcal{G}_T$ that is isomorphic via ι to $\mathcal{G}_P$. Therefore, $\widehat{V_T} \subseteq V_T$ is the range of ι. When an isomorphism is applied to a port we write $\iota((v, i))$ to mean $(\iota(v), i)$. Two ports (v, i), (u, j) are isomorphic if and only if $\iota(v) = u$, i.e. port indexes i and j are

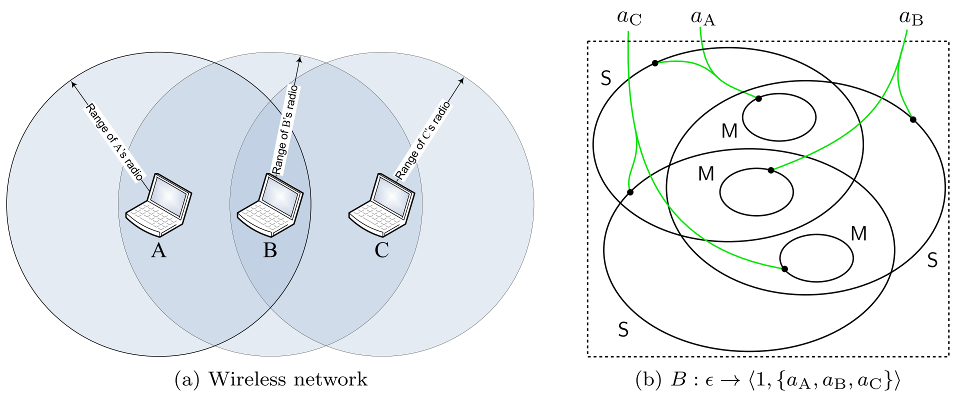


Fig. 10. Example wireless network of three stations (a) and its bigraphical representation (b).

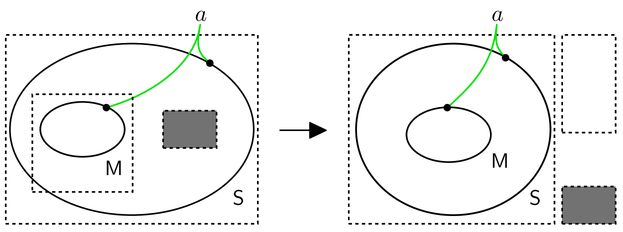


Fig. 11. Example reaction rule $R \longrightarrow R'$ modelling a station moving to a space location not covered by other signals.

not considered. In the following, we assume the pattern and the target are $P : \langle m, X\rangle \to \langle n, Y\rangle$ and $T : \langle m', X'\rangle \to \langle n', Y'\rangle$, respectively. Finally, the set of idle links of concrete bigraph P is defined by

$$P^{\mathsf{L}}_{idle} \overset{\text{def}}{=} \{l \in E_P \uplus Y \mid points_P(l) = \emptyset\} \ .$$

We are now ready to present the matching algorithm. We begin with an informal outline. The inputs of the algorithm are two concrete bigraphs: the first being the pattern while the second is the target. The output is the set of pairs (ι, η) specifying all the occurrences of the pattern in the target. The first phase of the algorithm consists of invoking a procedure that solves the sub-graph isomorphism problem. The underlying graphs of the pattern and the target are used as inputs. The output is a set of graph isomorphisms. In the next phase, the algorithm checks that every isomorphism obtained in the previous phase satisfies the following *compatibility* conditions:

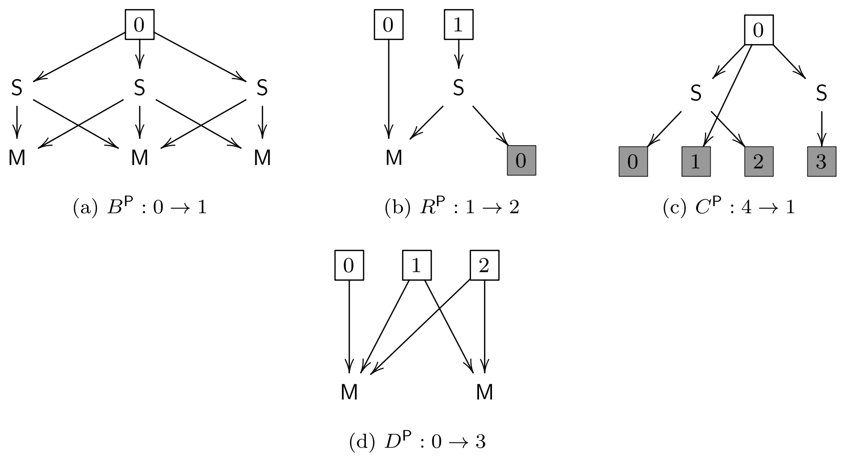
- Node controls are preserved.
- The pattern has sites and roots allowing for the construction of a context and a parameter by decomposing the target.
- No node in the context has an ancestor in the pattern.

All the isomorphisms failing to pass any of the checks above are discarded. Finally, output pairs are computed by associating every compatible isomorphism with a mapping from the pattern links to the target links. Each mapping ensures that the link graph of the pattern can be composed with the link graph of a context and the link graph of a parameter, both obtained by decomposing the target. If the procedure fails to build such a mapping, then the corresponding isomorphism is discarded.

Before presenting the full specification of the algorithm, we illustrate with a simple example how the algorithm computes the occurrences in a matching instance.

Example 3. Consider the wireless network given in Fig. 10a and its bigraphical encoding $B : \epsilon \to \langle 1, \{a_A, a_B, a_C\}\rangle$ described in Fig. 10b. In this model, nodes of control M and S encode stations and wireless signals, respectively. Moreover, each station is linked to its signal and an outer name encoding its unique network address. Now take reaction rule $R \longrightarrow R'$ described by the diagram in Fig. 11 which models a station moving to a space location not covered by other signals. On the left-hand side, the region containing the M-node represents the fact that the station may be within the range of zero or more signals (besides its own signal). On the right-hand side, the machine can only be in the range of its signal. In order to compute all the bigraphs that can be generated by applying rule $R \longrightarrow R'$ to B, we have to compute all the occurrences of R in B.

According to the algorithm outlined above, the first step consists of computing concretions $\widetilde{B}$ and $\widetilde{R}$. Then a set of graph isomorphisms between the underlying graphs of the pattern and the target are computed. In this case, there are seven possible isomorphisms and all of them satisfy the three compatibility conditions. Intuitively, each isomorphism associates the edge with form $\mathsf{S} \to \mathsf{M}$ in the place graph of the pattern given in Fig. 12b to all the seven edges with the same form in



(a) $B^{\mathsf{P}} : 0 \to 1$ (b) $R^{\mathsf{P}} : 1 \to 2$ (c) $C^{\mathsf{P}} : 4 \to 1$ (d) $D^{\mathsf{P}} : 0 \to 3$

Fig. 12. Place graphs of occurrence $B = C \circ (R \otimes \mathsf{id}_2) \circ D$.

```
1   function MATCH($T$, $P$)

2   $I := \text{SUB\_ISO}(\mathcal{G}_T, \mathcal{G}_P)$;
3   $I' := \emptyset$; $M := \emptyset$;
4   forall $\iota \in I$ do
5           $\widehat{V_P} := \iota(V_P)$;
6           if CTRL($\iota$, $ctrl_T$, $ctrl_P$) $\wedge$ SITES($\iota$, $T$, $P$) $\wedge$
7               ROOTS($\iota$, $T$, $P$) $\wedge$ TRANS($T$, $\widehat{V_P}$) then
8                   $I' := \{\iota\} \cup I'$;
9           end if;
10  forall $\iota \in I'$ do
11          $\eta := \text{BUILD\_LINK\_MAP}(\emptyset, (E_P \uplus Y) \setminus P^{\mathsf{L}}_{idle}, \iota, T, P)$;
12          if $\eta \neq \emptyset$ then
13                  $M := \{(\iota, \eta)\} \cup M$;
14          else if $P_P = \emptyset$ then
15                  $M := \{(\iota, \emptyset)\} \cup M$;
16          end if;
17  return $M$;
18  end function
```

Fig. 13. Pseudocode for the matching algorithm.

the place graph of the target given in Fig. 12a. This set of solutions is then reduced to three isomorphisms because pattern R specifies that an M-node and an S-node are allowed to be in the same match only if they are linked. When either station A or C is matched, the place graphs of context C and parameter D are as shown in Fig. 12c and Fig. 12d, respectively.

The algorithm is defined formally by the pseudocode given in Fig. 13. It takes the form of a function MATCH(_, _) in which the two arguments are concrete bigraphs. Since node-free patterns are a match for any target, we assume the second argument has a non-empty node set, i.e. $V_P \neq \emptyset$. Note that the pattern's port set can still be empty if all nodes in V_P have 0-arity controls.

We now describe in detail the operations described in the pseudocode. In line 2, the reduction to the sub-graph isomorphism problem is performed by invoking sub-routine SUB_ISO($\mathcal{G}_T, \mathcal{G}_P$). Here, we use it in a black-box fashion. Therefore, we assume an implementation is provided.[2] The results are stored in set I. Every isomorphism $\iota \in I$ takes the form $\iota : V_P \to \widehat{V_P}$.

The loop in lines 4–9 filters set I by checking that every element satisfies the compatibility conditions listed above. Each condition is checked in lines 6–7 by invoking a different sub-routine. The isomorphisms passing all the tests are stored in I'. Pseudocode for function CTRL(_, _, _) is given in Fig. 14. In line 2, there is a check that the input isomorphism preserves controls. Sites and roots are checked by invoking SITES(_, _, _) and ROOTS(_, _, _), respectively. The pseudocode for SITES(_, _, _) is in Fig. 15. The loop in lines 2–6 checks that P's sites allow the composition $(P \otimes \mathsf{id}_I) \circ D$. More precisely, it checks that for every node or site c in D having parents in $\widehat{V_P}$, there exists a set of sites S in P whose parent set is isomorphic to c's parent set. The pseudocode for ROOTS(_, _, _) is shown in Fig. 16. Again the inputs are an isomorphism, the

[2] Note that an implementation based on SAT will be described later in this section.

```
1  function CTRL($\iota$, $ctrl_T$, $ctrl_P$)

2  if exists $(v, u) \in \iota$ such that $ctrl_P(v) \neq ctrl_T(u)$ then
3          return false;
4  else
5          return true;
6  end if;
7  end function
```

Fig. 14. Pseudocode for sub-routine CTRL(_, _, _).

```
1  function SITES($\iota$, $T$, $P$)

2  forall $c \in \{v \in (m' \uplus V_T) \setminus \widehat{V_P} \mid prnt_T(v) \cap \widehat{V_P} \neq \emptyset\}$ do
3          if not exists $S \subseteq m$ such that
4              $\iota(\bigcup_{s \in S}(prnt_P \rhd V_P)(s)) = (prnt_T \rhd \widehat{V_P})(c)$ then
5                  return false;
6          end if;
7  return true;
8  end function
```

Fig. 15. Pseudocode for sub-routine SITES(_, _, _).

```
1  function ROOTS($\iota$, $T$, $P$)

2  forall $p \in \{v \in (V_T \uplus n') \setminus \widehat{V_P} \mid prnt_T^{-1}(v) \cap \widehat{V_P} \neq \emptyset\}$ do
3          if not exists $R \subseteq n$ such that
4              $\iota(\bigcup_{r \in R}(prnt_P^{-1} \rhd \widehat{V_P})(r)) = (prnt_T^{-1} \rhd \widehat{V_P})(p)$ then
5                  return false;
6          end if;
7  return true;
8  end function
```

Fig. 16. Pseudocode for sub-routine ROOTS(_, _, _).

```
1  function TRANS($T$, $\widehat{V_P}$)

2  if exists $u \in V_T \setminus \widehat{V_P}$ and $v, v' \in \widehat{V_P}$ such that
3      $u \in prnt_T(v)$ and $v' \in prnt_T^{+}(u)$ then
4          return false;
5  else
6          return true;
7  end if;
8  end function
```

Fig. 17. Pseudocode for sub-routine TRANS(_, _).

target and the pattern. The procedure is the dual of SITES(_, _, _). It checks that P's roots allow the composition $C \circ (P \otimes \mathsf{id}_I)$. Namely, the loop in lines 2–6 checks that for every node or root p in C having children in $\widehat{V_P}$, there exists a set of roots R in P whose set of children is isomorphic to p's set of children. The final condition on the isomorphisms is checked by invoking TRANS$(T, \widehat{V_P})$, pseudocode for this sub-routine is given in Fig. 17. In lines 2–3, it is checked that a node in $\widehat{V_P}$ does not have a parent in the context that has an ancestor in $\widehat{V_P}$. Observe that each $\iota \in I'$ satisfies all the compatibility conditions. Hence, it is a support translations for P with range $\widehat{V_P}$.

The second loop in MATCH(_, _) (lines 10–16 in Fig. 13) builds a mapping between P's and T's links. The solutions are stored in set M. If no mapping is returned and there are no ports in the pattern, then only the isomorphisms are returned (lines 14–16). A mapping for every ι found in the previous phases of the algorithm is obtained by the invocation of BUILD_LINK_MAP(_,_,_,_,_) in line 11. Since idle links can always occur in any pattern, only non-idle links of P are considered as shown by the second argument.

Pseudocode for sub-routine BUILD_LINK_MAP(_,_,_,_,_) is given in Fig. 18. The first argument η is a partial mapping. It is used by the procedure to recursively construct a full solution. The second argument L is a sub-set of T's link. Its elements are the links not in the range of η. The other three arguments are an isomorphism, the target and the pattern, in this order.

```
1    function BUILD_LINK_MAP($\eta, L, \iota, T, P$)

2    if $L \neq \emptyset$ then
3        choose $l \in L$;
4        $S := \iota(count_P(l))$;
5        if $\tau(l) = \mathsf{name}$ then
6            $K := \{k \mid S \subseteq count_T(k)\}$;
7        else if $\tau(l) = \mathsf{edge}$ then
8            $K := \{k \in E_T \mid S = count_T(k)\}$;
9        else
10           $K := \{k \in E_T \mid S \subseteq count_T(k) \wedge (\text{not exists } v \in \widehat{V_P},$
11                       $u \in (count_T(k) \setminus S) \text{ s.t. } (v, u) \in prnt_T^{+})\}$;
12       end if;
13       forall $k \in K$ do
14           $T' :=$ REM_PORTS($T, k, S$);
15           $res :=$ BUILD_LINK_MAP($\eta \cup \{(l, k)\}, L \setminus \{l\}, \iota, T', P$);
16           if $res \neq \emptyset$ then
17               return $res$;
18           end if;
19       return $\emptyset$;
20   else
21       return $\eta$;
22   end if;
23   end function
```

Fig. 18. Pseudocode for sub-routine BUILD_LINK_MAP(_, _, _, _, _).

When argument L is empty, this means that η contains a mapping for every non-idle link in the pattern. Therefore, η is accepted as a valid solution in line 21. On the other hand, when L is not empty, a link l is selected in line 3. Then, the operations in lines 5–12 construct a set K whose elements are all the links in the target that can be mapped to l. There are three cases:

- If l is an outer name (i.e. $\tau(l) = \mathsf{name}$), then l can be mapped to any link $k \in K$ such that l's ports are an isomorphic subset of k's ports. This corresponds to the conditional statement in lines 5–6.
- If l is an edge and all its points are ports (i.e. $\tau(l) = \mathsf{edge}$), then l can be mapped to any edge $k \in K$ such that l's ports are isomorphic to k's ports. This corresponds to the instructions in lines 7–8.
- If l is an edge having an inner name $x \in X$ as a point (i.e. $\tau(l) = \mathsf{open\text{-}edge}$), then l can be mapped to any edge $k \in K$ such that l's ports are an isomorphic subset of k's ports and the other k's ports belong to nodes who are not ancestors of any node in $\widehat{V_P}$. In other words, they must belong to nodes in the parameter. This corresponds to the pseudocode in lines 9–11.

Observe that port identifiers are ignored because all comparisons are performed on multisets of nodes generated by function *count*(_). The loop in lines 13–18 builds *res*, the output of the sub-routine. This is the result of one of the recursive invocations BUILD_ LINK_MAPS($\eta \cup \{(l, k)\}, L \setminus \{l\}, \iota, T', P$). By analysing the arguments of the sub-routine, we see that pair (l, k) is added to partial solution η and l is removed from L, the set of links yet to be checked. Moreover, target T is replaced by T' resulting from invocation REM_PORTS(T, k, S) in line 14, with multiset S defined in line 4. Intuitively, T' is defined exactly as T, but each port corresponding to a node in S is removed from the set of ports of link k. In this way, we ensure that k's ports used to match a link in the pattern are not used again to match a different link in a successive recursive invocation. If no mapping can be constructed, either because all recursive invocations return $\emptyset$ or because $K = \emptyset$, then BUILD_LINK_MAP(_,_,_,_,_) returns an empty mapping in line 19 and the partial solution η is discarded. This because only total mappings can be valid solutions.

Pseudocode for sub-routine REM_PORTS(_,_,_) is given in Fig. 19. The loop in lines 3–5 iterates over multiset S, which contains all the nodes v having ports already used in a match, and thus are to be removed. In line 4, a fresh port corresponding to v is chosen and is added to *res* in line 5. In line 6, the link map of T is updated by removing all the ports in *res* from the set of ports of k. Bigraph T' is returned in line 7.

4.2. Soundness and completeness

We now prove that the algorithm is sound and complete. Informally, soundness is proven by showing that any solution obtained as output of MATCH(T, P) identifies an occurrence P in T. This means that a context and a parameter can be obtained by decomposing T as specified by Definition 14. Completeness is proven by showing that when P occurs in T, then the algorithm produces a solution that identifies the decomposition of T induced by match P.

```
function REM_PORTS($T, k, S$)

res := $\emptyset$;
forall $v \in S$ do
    choose $i$ such that $(v, i) \in ports_T(k)$ and $(v, i) \notin res$;
    res := res $\cup \{(v, i)\}$;
$link'_T := link_T \setminus \{(p, k) \mid p \in res\}$;
return $\langle T^{\mathsf{P}}, (V_T, E_T, ctrl_T, link'_T)\rangle$;
end function
```

Fig. 19. Pseudocode for sub-routine REM_PORTS(_,_,_).

Proposition 18 *(Soundness). Let T and P be two concrete bigraphs, with $V_P \neq \emptyset$. If* MATCH(T, P) *returns a solution (ι, η), then (ι, η) identifies an occurrence of P in T.*

Proof. Recall that a decomposition takes the form $T = C \circ (P' \otimes \mathsf{id}_{\langle j, J\rangle}) \circ D$ with $P' = \rho \bullet P$. We begin by proving the decomposition over place graphs.

By construction $\iota = \rho_V$ and $\widehat{V_P} = V_{P'}$. Set V_C is defined to contain all the nodes in V_T having a descendant in $V_{P'}$. It follows that $V_D = V_T \setminus (V_C \uplus V_{P'})$. The interfaces of the context and the parameter are $C : n + j \to n'$ and $D : m' \to m + j$, respectively. By definition of composition, the parent relations are

$$prnt_D \rhd V_D = prnt_T \rhd V_D \qquad V_C \lhd prnt_C = V_C \lhd prnt_T \ .$$

Relations $prnt_D \rhd j$ and $j \lhd prnt_C$ contain the pairs (u, j_i) and (j_i, v), respectively, for every $(u, v) \in prnt_T$, with $u \in V_D$ and $v \in V_C$. Finally, $prnt_D \rhd m$ is the relation containing pairs (c, s) identified by procedure SITES(ι, T, P). Note that $c \in (m' \uplus V_D)$ and $s \in m$. In a similar fashion, relation $n \lhd prnt_C$ is formed by pairs (r, p) detected by ROOTS(ι, T, P), with $r \in n$ and $p \in (V_C \uplus n')$.

We now construct a decomposition for link graphs. By construction $E_P \lhd \eta = \rho_E$ and $E_{P'} = \rho_E(E_P)$. Set E_D contains all the edges in E_T that have a port set entirely over nodes in V_D. The edges of the context are $E_C = E_T \setminus (E_{P'} \uplus E_D)$. The interfaces in the link graphs of the decomposition are $C : Y \uplus J \to Y'$ and $D : X' \to X \uplus J$. Set J contains a name for each point in D being mapped to a link in C. Therefore, identity id_J is obtained by a construction similar to the one described for id_j: pairs (p, w), (w, l), with $w \in J$, are added to $link_D \rhd J$ and $J \lhd link_C$, respectively, for every $(p, l) \in link_T$. Therefore, the complete link map for C is given by

$$link_C = (J \lhd link_C) \uplus (Y \lhd \eta) \uplus (P_C \lhd link_T) \ .$$

Relation $link_D \rhd X$ contains the pairs (p, x), with $p \in (X' \uplus P_D)$ and $x \in X$, for every $(p, e) \in link_T$ and $(x, e) \in link_{P'}$. Finally, the complete link map for D is

$$link_D = (link_D \rhd J) \uplus (link_D \rhd X) \uplus (link_T \rhd E_D) \ .$$

This concludes the proof. □

Proposition 19 *(Completeness). Let T and P be two concrete bigraphs, with $V_P \neq \emptyset$. If P is a match in T, then* MATCH(T, P) *returns a solution (ι, η) for every occurrence of P in T.*

Proof. By Definition 14, the hypothesis is $T = C \circ (P' \otimes \mathsf{id}_I) \circ D$ with $P' = \rho \bullet P$. By Definitions 6 and 15, ρ_V is a control-preserving isomorphism from $\mathcal{G}_P$ to $\mathcal{G}_{P'}$. Since $V_{P'} \subseteq V_T$, ρ_V is an output of SUB_ISO$(\mathcal{G}_T, \mathcal{G}_P)$ and CTRL$(\rho_V, ctrl_T, ctrl_P)$ returns **true**. By Lemma 6 and by hypothesis $(P' \otimes \mathsf{id}_I) \circ D$, SITES$(\rho_V, T, P)$ returns **true**. Similarly, by Lemma 7 and by hypothesis $C \circ (P' \otimes \mathsf{id}_I)$, ROOTS$(\rho_V, T, P)$ returns **true**. The hypothesis and Lemma 5 also ensure that TRANS$(T, \widehat{V_P})$ returns **true**. We now show that the algorithm returns a valid link mapping η. By definition of composition for concrete link graphs, we know that when $\tau_{P'}(e) = \mathsf{edge}$ holds for an edge $e \in E_{P'}$, then $\tau_T(e) = \mathsf{edge}$ also holds. Therefore, $\rho_E \subset \eta$ because of lines 7–8 in BUILD_LINK_MAP(_, _, _, _, _). The other pairs in η are found by checks on lines 5–6 and 10–11. The existence of these pairs follows from the hypothesis and by Definition 4. This proves that (ι, η), with $\iota = \rho_V$, is a solution returned by MATCH(T, P) when the hypothesis holds. This concludes the proof. □

Another important property of the algorithm is the following:

Lemma 20. *Algorithm* MATCH(_, _) *supports non-sharing concrete bigraphs.*

4.3. Complexity

The matching problem for standard bigraphs is closely related to the sub-graph isomorphism problem. Specifically, the matching of place graphs without sharing can be reduced to the sub-forest isomorphism problem. A full match can then be computed by introducing extra constraints expressing the matching of link graphs. Both the sub-graph and sub-forest isomorphism problems[3] are proven to be **NP**-hard [4]. Therefore, algorithms that compute their solutions in polynomial time are not known. Traditionally, these problems are solved by using some type of backtracking search, as shown for instance in [5]. Observe that when the input bigraphs have only one root, the matching instance can be reduced to an instance of the sub-tree isomorphism, which can be solved in polynomial time.

The matching problem for bigraphs with sharing can be reduced to the sub-graph isomorphism problem. More precisely, it is reducible to the sub-DAG isomorphism problem, which is also **NP**-complete [6]. The time complexity of $\textsc{Match}(\_,\_)$ is dominated by the running time of sub-routine $\textsc{Sub\_iso}(\_,\_)$, which is $\mathcal{O}(|V_T|^{|V_P|})$. Note that the introduction of sharing does not involve an increase in complexity in the general case.

We have chosen to define our matching algorithm as a reduction to the sub-DAG isomorphism problem for several reasons. The first one is that algorithms defined by structural induction on the algebraic representation of the input bigraphs (e.g. the algorithm defined in [7]) are not suited for the DAG structure of place graphs with sharing since the amount of unnecessary blind search introduced during the inference process dramatically increases, thus negatively affecting the algorithm's performance. Second, a reduction to a standard problem tends to produce faster algorithms because it fully exploits established solvers which are very efficient due to the use of advanced heuristics. A typical example of this approach is the encoding of hard problems as instances of the boolean satisfiability problem (SAT) [8,9]. Finally, we want our algorithm to support concrete bigraphs and arbitrary patterns. Support for concrete bigraphs is required to allow the enumeration of all occurrences of the pattern in the target. This is necessary for any complete implementation of stochastic BRS [3]. Support for arbitrary patterns is necessary because we will use the matching algorithm not only to match a redex in an agent during the computation of the reaction relation in a BRS (see Section 6), but also for the verification of a class of **BiLog** formulae as we will describe in Section 5.

4.4. SAT encoding

We now describe the details of the algorithm implementation based on a SAT encoding of $\textsc{Match}(\_,\_)$. We note that while we have indicated that inference based procedures for matching bigraphs with sharing would not be efficient, a SAT encoding of matching is generally more efficient for standard bigraphs as well. An indication of example timings is given in Section 7.2.

In the following, we give insight into our approach by showing the encodings of sub-routines $\textsc{Sub\_iso}(\_,\_)$ and $\textsc{Ctrl}(\_,\_,\_)$. The complete encoding is given in [10].

The encoding is defined by a set of boolean formulae, called *constraints*, expressing an instance of the matching problem. Given a target T and a pattern P, invocation $\textsc{Sub\_iso}(\mathcal{G}_T, \mathcal{G}_P)$ is encoded by a matrix $\mathbf{X}$ of boolean variables and by four constraints on its structure. Each element $x_{i,j}$ of $\mathbf{X}$ represents a possible pair $(i,j) \in \iota$, with ι an isomorphism from P to a subset of T. Therefore, $\mathbf{X}$ has n rows and m columns, with $|V_P| = n$ and $|V_T| = m$. The constraints on $\mathbf{X}$ are formulae in conjunctive normal form (i.e. a conjunction of clauses). By definition of isomorphism, ι is total. Hence, every node of the pattern has to be mapped to at least one node in the target. This corresponds to having at least one variable in every row of $\mathbf{X}$ that is assigned **true**. Formally, the constraint has n clauses and is defined as follows

$$C_1 = \bigwedge_{0 \le i < n} \bigvee_{0 \le j < m} x_{i,j} \ .$$

The second constraint specifies that each node in P is mapped to at most one node in T. This corresponds to having at most one variable in every row of the matrix that is assigned **true**. Therefore, the encoding needs to check every pair $(x_{i,j}, x_{i,k})$, with $j < k$ in every row i of $\mathbf{X}$. The constraint is given by

$$C_2 = \bigwedge_{0 \le i < n} \bigwedge_{0 \le j < m-1} \bigwedge_{j < k < m} (\neg x_{i,j} \vee \neg x_{i,k}) \ .$$

The number of clauses is $n\binom{m}{2} = n\frac{m(m-1)}{2} = \mathcal{O}(nm^2)$. Since ι is a bijection, each node in T is mapped by at most one node in P. The corresponding constraint is defined similarly to the previous one, but the pairs are taken from the columns of $\mathbf{X}$. The definition is as follows:

$$C_3 = \bigwedge_{0 \le j < m} \bigwedge_{0 \le i < n-1} \bigwedge_{i < l < n} (\neg x_{i,j} \vee \neg x_{l,j}) \ .$$

[3] When expressed as decision problems.

The number of clauses is $m\binom{n}{2} = \mathcal{O}(mn^2)$ clauses. Finally, the edges of $\mathcal{G}_P$ have to correspond to the edges of $\mathcal{G}_T$ and vice versa. This is expressed by the following constraint:

$$C_4 = \bigwedge_{(i,l,j,k)\in\mathcal{C}} \neg x_{i,j} \vee \neg x_{l,k}$$

with $\mathcal{C} = \{(i,l,j,k) \mid (i,l) \in prnt_P \neq (j,k) \in prnt_T\}$. In the worst case, the constraint has n^2m^2 clauses. The formula encoding Sub_iso$(\mathcal{G}_T, \mathcal{G}_P)$ is the conjunction of the four constraints. If an isomorphism $\iota : \mathcal{G}_P \to \mathcal{G}_T$ exists, then the SAT solver returns a truth assignment $\mathcal{A}$ such that

$$\mathcal{A} \models C_1 \wedge C_2 \wedge C_3 \wedge C_4 \qquad \text{and} \qquad (i,j) \in \iota \Longleftrightarrow \mathcal{A}(x_{i,j}) = \mathbf{true} \ .$$

It is possible to encode Ctrl$(\iota, ctrl_T, ctrl_P)$ with a similar constraint on $\mathbf{X}$:

$$C_5 = \bigwedge_{(i,j)\in\mathcal{C}'} \neg x_{i,j} \qquad \text{with} \qquad \mathcal{C}' = \{(i,j) \mid ctrl_P(i) \neq ctrl_T(j)\} \ .$$

It has nm clauses in the worst case. Note that $\bigwedge_{0<i<6} C_i$ can be used to obtain directly a control-preserving isomorphism. In this way, isomorphisms that are not solutions can be detected and discarded earlier in the computation. This is more efficient that computing all the possible isomorphisms with Sub_iso(_, _) and filtering them afterwards with Ctrl(_, _, _), as specified in the definition of Match(_, _). The total number of clauses required by the encoding is $\mathcal{O}(n^2m^2)$.

Note that the constraints described so far only allow to compute one isomorphism. When the computation of all distinct occurrences is necessary (e.g. in stochastic BRS), a new SAT instance is generated by adding to the set of constraints a clause corresponding to the negation of a solution $\mathcal{A}$ already found:

$$\mathcal{A}' \models (\bigwedge_{0<i<6} C_i) \wedge (\bigvee_{x\in\mathcal{C}''} \neg x) \qquad \text{with} \qquad \mathcal{C}'' = \{x \mid \mathcal{A}(x) = \mathbf{true}\}$$

where $\mathcal{A}'$ corresponds to a different isomorphism. This operation is iterated until the SAT instance we obtain is unsatisfiable. This assures that all the isomorphisms from P to a subset of T are found.

The other phases of the algorithm are expressed as constraints in a similar fashion. The conjunction of all constraints is the SAT instance encoding the matching algorithm. Note that a different matrix is needed for link matching.

4.5. Bigraph equality

The bigraph equality problem is closely related to the bigraph matching problem. It is defined as the computational task in which two bigraphs A and B are given as input, and one must determine whether $A \simeq B$. This corresponds to checking for the existence of a support translation ρ such that $\widetilde{A} = \rho \cdot \widetilde{B}$. Bigraph equality is fundamental to check **BiLog** predicates and also to detect if a state (i.e. an abstract bigraph) was already discovered when computing the transition system of a BRS. An algorithm for bigraph equality can also be used to identify all the automorphisms of a bigraph by repeatedly running it against itself. This is necessary to count distinct occurrences in a stochastic BRS.

We define an equality algorithm by reduction to the graph isomorphism problem and we encode it in SAT. Equality over place graphs is defined by the five constraints $C_1, \dots, C_5$ over boolean matrix $\mathbf{X}$ that form the encoding of place graph matching described above. Here they are used to find isomorphism $\iota : \mathcal{G}_A \to \mathcal{G}_B$. Two additional constraints are required in order to check the correspondence between edges involving roots and sites:

$$C_6 = \bigwedge_{(i,j)\in\mathcal{S}} \neg x_{i,j} \quad \text{with} \quad \mathcal{S} = \{(i,j) \mid (s,i) \in prnt_A \neq (s,j) \in prnt_B\}$$

$$C_7 = \bigwedge_{(i,j)\in\mathcal{R}} \neg x_{i,j} \quad \text{with} \quad \mathcal{R} = \{(i,j) \mid (i,r) \in prnt_A \neq (j,r) \in prnt_B\}$$

where s and r indicate a site and a root in A and B, respectively. Note that any ι satisfying C_6 and C_7 is also a support translation ρ_V. Therefore, if the SAT solver returns a truth assignment $\mathcal{A}$ such that

$$\mathcal{A} \models \bigwedge_{1\leq i\leq 7} C_i \qquad \text{and} \qquad (i,j) \in \iota \Longleftrightarrow \mathcal{A}(x_{i,j}) = \mathbf{true}$$

then $A^{\mathsf{P}} \simeq B^{\mathsf{P}}$. Constraints for link graph equality are similar. In this case η is constraint to be a bijection.

The bigraph equality problem is in the same complexity class of the graph isomorphism problem which belongs to **NP**, but it is not known to be **NP**-complete or in **P**.

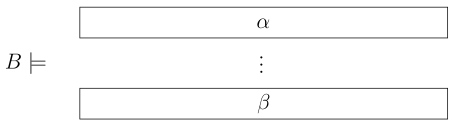


Fig. 20. Diagram for $B \models \alpha \circ \top \circ \beta$.

5. Checking BiLog predicates by bigraph matching

The structure of bigraphs can be described by **BiLog**, a spatial logic whose modal operators are capable of expressing the placing and linking structure of bigraphical terms, in a fashion common to logics for process calculi. In **BiLog**, we can express formulae like $\mathbf{A} \circ \varphi$ to describe a node of control A acting as context for a bigraph satisfying predicate φ and $(\mathbf{A}_a \circ \top) \otimes (\mathbf{A}_b \circ \top)$ to describe two ions in two different regions next to each other, with different names that can contain any sub-term, indicated by tautology $\top$. More advanced operators called *adjuncts* are also defined. For example, the left adjunct $\varphi \circ\!\!\!-\!\!\!- \psi$ expresses the property of a term satisfying ψ whenever inserted in a context satisfying φ. Refer to [11] for a complete account on **BiLog**.

In the following, we present how a class of predicates can be checked by reduction to bigraph matching. The **BiLog** fragment we consider contains the formulae solely formed by operators $\circ$ and $\otimes$ and elements of $\Omega \cup \top$, where Ω indicates the set of formulae denoting elementary bigraphs. Example formulae in Ω are $\mathbf{id}_I$, which is satisfied by every identity over interface I, and $\mathbf{K}_x$, which holds for ion K_x. Intuitively, any formula in this form contains one or more sub-formulae that can be used as a pattern in an instance of bigraph matching. Formally, matching instances correspond to the formulae generated by the following grammar:

$$\psi ::= \Omega \;\Big|\; \psi \circ \psi \;\Big|\; \psi \otimes \psi \ .$$

Note that boolean operators and logical adjuncts are not allowed. Formulae in the fragment are indicated by φ. They are generated by

$$\begin{aligned}
\varphi &::= \psi \;\Big|\; \zeta \\
\zeta &::= \top \;\Big|\; \top \circ \kappa \;\Big|\; \top \otimes \kappa \;\Big|\; \kappa \circ \zeta \;\Big|\; \zeta \circ \kappa \;\Big|\; \kappa \otimes \zeta \;\Big|\; \zeta \otimes \kappa \\
\kappa &::= \psi \;\Big|\; \psi \circ \zeta \;\Big|\; \psi \otimes \zeta \ .
\end{aligned}$$

When a formula is parsed, it is assumed that symbol ψ has the highest precedence, composition binds more tightly than tensor product, and parentheses are used to disambiguate.

We now illustrate with an example how φ-formulae can be checked by combining the SAT instances that encode the matching of their ψ-sub-terms. Consider the following satisfiability statement: $B \models \alpha \circ \top \circ \beta$, where B is an abstract bigraph and α, β are ψ-formulae. A description of the statement by means of diagrams is drawn in Fig. 20. Two concrete bigraphs P_α and P_β can be constructed starting from α and β, respectively. They act as patterns in matching instances $\textsc{Match}(\widetilde{B}, P_\alpha)$ and $\textsc{Match}(\widetilde{B}, P_\beta)$. The first instance is encoded by boolean matrix $\mathbf{A}$ with dimensions of $n \times m$ (with $|V_{P_\alpha}| = n$, $|V_B| = m$) and by instantiations $C_i^{\mathbf{A}}$ $(0 < i < 6)$ over matrix $\mathbf{A}$ of constraints C_i described in Section 4 (Note, in Section 4 this matrix is called $\mathbf{X}$). For example, the instantiation of constraint C_1

$$C_1^{\mathbf{A}} = \bigwedge_{0 \le i < n} \bigvee_{0 \le j < m} a_{i,j} \ .$$

The second instance is encoded by boolean matrix $\mathbf{B}$ with dimensions of $n' \times m$ (with $|V_{P_\beta}| = n'$) and by constraints $C_i^{\mathbf{B}}$ $(0 < i < 6)$. In order to encode formula $\alpha \circ \top \circ \beta$, five additional constraints specifying the relative positions of α and β in the parse tree have to be included. The first constraint specifies that the sub-graphs of $\widetilde{B}$ matched to P_α and P_β cannot overlap. The definition is as follows:

$$C_8 = \bigwedge_{0 \le j < m} \bigwedge_{0 \le i < n} \bigwedge_{0 \le l < n'} (\neg a_{i,j} \vee \neg b_{l,j}) \ .$$

The second constraint specifies that any node in $\widetilde{B}$ matched to a node in P_α cannot have an ancestor matched by a node in P_β, i.e. β is not above α in the parse tree. The constraint is given by

$$C_9 = \bigwedge_{(j,k) \in prnt_B^+} \bigwedge_{0 \le i < n} \bigwedge_{0 \le l < n'} (\neg a_{i,j} \vee \neg b_{l,k}) \ .$$

The third constraint specifies that α and β are not juxtapposed. Therefore, any node in $\widetilde{B}$ matched to a node in P_β has at least one ancestor matched by a node in P_α. This is expressed as follows:

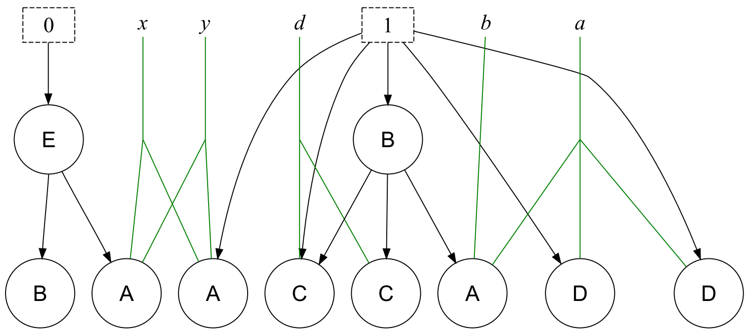


Fig. 21. Example of automatically-generated visualisation.

$$C_{10} = \bigwedge_{(j,k)\in prnt_B^+} \bigwedge_{0\le l<n'} (\neg b_{l,j} \vee \bigvee_{0\le i<n} a_{i,k}) \ .$$

Note that constraints C_9 and C_{10} together specify that α is above β in the parse tree. Finally, the outer and inner interfaces of B have to correspond to the outer interface of P_α and the inner interface of P_β, respectively. This is expressed by constraints $C_6^{\mathbf{B}}$ and $C_7^{\mathbf{A}}$, i.e. the instantiations over **A** and **B** of the constraints introduced in Section 4 to encode bigraph equality. Therefore, the complete encoding is

$$\mathcal{A} \models C_1^{\mathbf{A}} \wedge \cdots \wedge C_5^{\mathbf{A}} \wedge C_1^{\mathbf{B}} \wedge \cdots \wedge C_5^{\mathbf{B}} \wedge C_8 \wedge C_9 \wedge C_{10} \wedge C_6^{\mathbf{B}} \wedge C_7^{\mathbf{A}} \ .$$

If the SAT solver returns a valid truth assignment $\mathcal{A}$, then $B \models \alpha \circ \top \circ \beta$. Constraints to handle link graphs are similar.

The example above shows how to check a φ-formula by reduction to an instance of SAT. Summarising, each ψ-sub-formula specifies the constraints encoding an instance of bigraph matching, while the remaining constraints are obtained by analyzing where these occurrences appear in the parse tree. The final step is to add constraints over interfaces.

6. BigrapER: bigraph rewriting engine

We now describe BigraphER [12], an implementation of BRS and stochastic BRS that supports place graphs with sharing. The tool consists of an OCaml library and a command-line tool that provides efficient manipulation and simulation of BRS and stochastic BRS. It is composed of three distinct modules: the *compiler*, the *matching engine* and the *rewriting engine*.

The compiler is the component that translates an input source file containing the specification of a bigraphical model into its run-time representation. The language used, called BigraphER *specification language*, closely resembles the algebraic form of bigraphs described in Section 3. Observe that although the specification language defines abstract BRS, the compiler stores in memory the corresponding concretions. This facilitates the creation and manipulation (i.e. composition, tensor product and matching) of the data structures employed internally by the system to represent bigraphs. The compiler is used by the rewriting engine to retrieve the data structures corresponding to the initial state of the BRS and to the reaction rules of the model. The compiler can also return a graphical representation of every bigraph specified in the input file. This is generated by the automatic graph-layout generator Graphviz [13]. An example output is shown in Fig. 21.

The matching engine implementation is based on the SAT encoding of the graph theoretic matching algorithm introduced in Section 4. It consists of two OCaml modules. The first encodes the data structures representing a target and a pattern into appropriate boolean matrices. Then, it computes the SAT constraints over the matrices and passes the resulting SAT instance to the APIs provided by the MiniSat [14] solver. This module can also be used as a stand-alone command-line application. The second provides bindings to the other modules of the BigraphER system. In particular, the matching engine is used by the rewriting engine to apply reaction rules to a state and to check equality of states.

The rewriting engine is the component of the system that computes the dynamic evolution of BRS. This is implemented by building a graph that has bigraphs as states and corresponds to the transition system of the input BRS. A Continuous Time Markov Chain (CTMC) is built when the input is a stochastic BRS. The graph is constructed by iteratively applying the reaction rules of the BRS to each state and then storing the resulting states, until a fixed point is reached. This occurs when all the bigraphs obtained by the application of the reaction rules are already present in the graph. Note that the model is assumed to have a finite state space. Therefore, no controls are performed to assure that the loop terminates. In order to avoid an abrupt termination when the program runs out of memory, a switch specifying the number of iterations can be used when BigraphER is launched. The reconfigurations of a state are computed by constructing its decompositions starting from the occurrences of a redex, i.e. the output of the matching engine. This procedure is defined in constructive Proof 18. Predicates belonging to the fragment of **BiLog** defined in Section 5 are checked during the generation of the transition system. In more detail, every time a state is added to the graph, all the predicates specified by the input model are checked against it. A labelling function is implemented by a hash table binding a state to a set whose elements are the identifiers of the predicates that are satisfied by the state. Therefore, the graph together with the hash table can be interpreted as

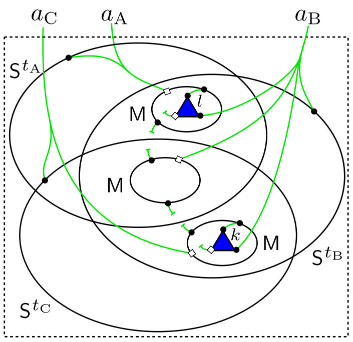


Fig. 22. Bigraphical representation of WLAN of three stations.

a Kripke structure or a labelled CTMC and temporal properties of the model can be checked. The rewriting engine also supports *rule priorities*.

The BigraphER OCaml library provides programming interfaces for the data structures used internally by the command-line tool. For instance, it is equipped with functions to compute the composition of two bigraphs and the result of a reaction application.

7. Applications of bigraphs with sharing

While many applications of bigraphs have been to meta-modelling, e.g. to modelling the π-calculus, λ-calculus, and CCS (Calculus of Communicating Systems), we have found bigraphs with sharing to be particularly useful for modelling and reasoning about spatial and temporal behaviours in distributed computing systems, especially those involving wireless communication. We give an overview of two such applications.

7.1. A communication protocol for wireless interference

In [15] we used bigraphs with sharing to model and reason about the four way handshaking RTS/CTS reservation scheme in the 802.11 CSMA/CA RTS/CTS protocol. The protocol is designed to address the hidden node problem, which occurs in wireless local area networks (WLANs) when two stations can communicate with a third station, but not with each other. An example of such a network topology, with overlapping network signals, is given in Figs. 1a and 10, where stations A and C can communicate with B, B can communicate with A and C, but A cannot communicate with C and C cannot communicate with A. To model all the functionality of the protocol, we encoded the networks as bigraphs as follows.

As in the example model for wireless networks described in Example 3, Section 4, nodes represent the main entities: stations, wireless signals and packets. Their relationships are expressed by links between them and by node sharing. In particular, when there are overlapping signals, there is a node representing a station in the intersection of the two signals (i.e. the nodes representing the signals). Control M is used to represent nodes encoding stations, and control S is used to represent wireless signals, the latter are parameterised by the size of the contention widow associated with that signal. Packets are either CTS, RTS, or data, and represented graphically by a triangle: red triangle for RTS and purple triangle for CTS. For example, the (graphical) bigraph given in Fig. 22 represents the example WLAN from Fig. 10a, with S^{t_A}, S^{t_B}, and S^{t_C} as signals of stations A, B, and C, respectively. Contention window sizes for A, B and C are indicted by t_A, t_B, and t_C, respectively. The blue triangles indicate that A and C are waiting to transmit a packet to B. Note that each station sits within its signal node.

The protocol is modelled by introducing additional controls to represent station states (e.g. idle, locked, backoff, deferred, error), and defining reaction rules (with priorities) that encode the stages of the protocol. There are 12 rules, 6 of which are *stochastic*, meaning there is a rate associated with the rule (the details of stochastic reaction rules are beyond the scope of this paper). In Fig. 23 we give one example rule: the graphical form of the rule that describes the transmission of a CTS packet from the sender to the receiver. In this rule, M_L denotes a *locked* station, M_D a *deferred* station, and S^t_L a *locked* signal. The reaction can be triggered only when the sender and the receiver sense each other and the receiver is available for a transmission. These two preconditions are encoded in the left-hand side by two nodes of control M_L and M_D being in the same intersection of signals and by the node of control $\mathsf{S}^{t'}$ containing the receiver, respectively. Moreover, M_D can be identified as the receiver because it is linked to the sender's RTS triangle. Note also that both the receiver and the sender can be in the range of other stations. On the right-hand side the receiver is locked (nodes of control $\mathsf{S}^{t'}_L$ and M_L) and a link between the two machines is established. The reaction rate is $\rho_2 = (\lambda_S + \lambda_C)^{-1}$, where λ_S is the short inter-frame space time period and λ_C is the time taken to transmit a CTS packet (at 1 Mbps).

The rules are organised into *priority classes* in order to avoid the introduction of parameterised reaction rules, and to allow for the specification of reaction rules with fixed size, regardless of the underlying communication topology. This leads to matching instances that are solvable in polynomial time because the number of nodes in the patterns is always bounded

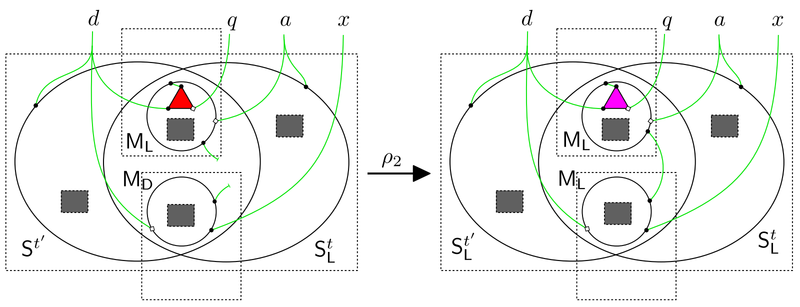


Fig. 23. Example reaction rule. On the left-hand side, the sender has sent an RTS; on the right-hand side, the sender and receiver are locked and linked and CTS is sent.

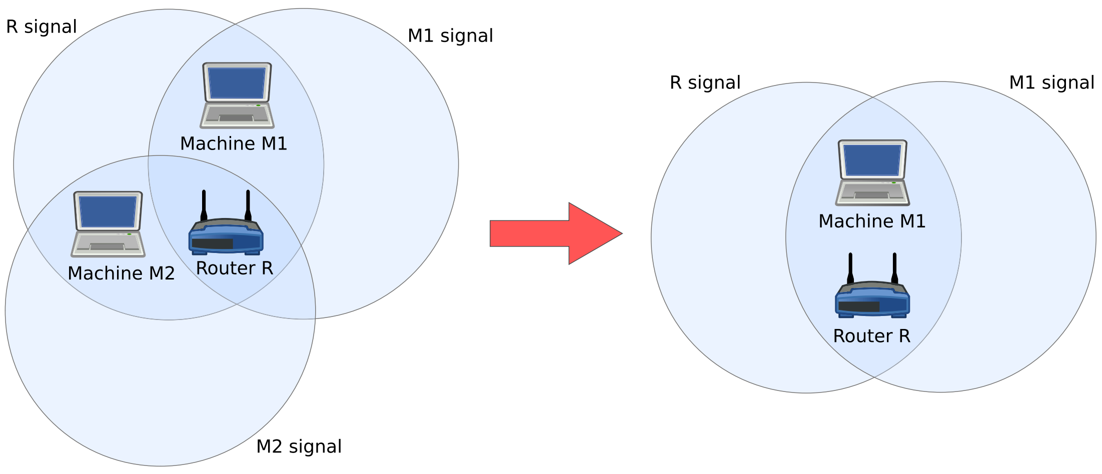


Fig. 24. Evolution of a WLAN: machine M_2 leaves the network.

by a constant. The model enables for the automatic generation of a CTMC capturing the behaviour of any wireless network governed by this protocol. This in turn allows for the analysis of quantitative properties such as the probability of collision and the average number of transmissions before a data packet is sent successfully. Quantitative analysis was carried out using the probabilistic model checker PRISM [16]. Properties are expressed by combining **CSL** and **BiLog**, i.e. **BiLog** predicates are used as atomic propositions in **CSL** (Continuous Stochastic Logic). Example properties include the probability that, from a given state, eventually a station is in an error state, a collision occurs, or a station successfully transmits its packet.

We now turn our attention to another application involving WLANs, with a focus on providing user-oriented support in a network management system.

7.2. Domestic wireless network management

Our aim in this application [17] was to model network interactions and user-initiated network policies for access control, in order to define a real-time, online framework for network management. The management system is based on the generation and analysis of models from actual network and user-initiated events. Analysis is carried out in real-time by a bespoke bigraph reasoning system based on the BigraphER rewriting engine, implemented on an experimental system router.

A brief overview of the bigraph encoding is the following. Similar to the example above, we use nodes to encode signals and stations, except now we distinguish between machines and routers. These are represented by controls S, M, and R respectively. Fig. 24 depicts a simple example evolution of a WLAN with one machine leaving the network; Fig. 25 gives the corresponding bigraphs.

Individual machines are identified and have associated properties, such as which policies have been invoked (or revoked). Individual machines are distinguished by a link to their MAC address, which lies within a property box and may, for example, be linked to the Internet via a pair of communication channels.

We have defined over 20 rules encoding network and user-initiated events. We give one example in Fig. 26, which models the appearance of a new machine in the signal range of the router. On the left-hand side, the router is in the range of its signal and possibly other signals. On the right-hand side, a new machine is in the range of the router's signal. The router senses the new machine's signal and possibly other signals. This is expressed by nodes R and M being in the intersection of the two S-nodes. A property box is also linked to M. Note that the only configuration setting specified at this stage is the MAC address of the new machine M, which is the new node placed inside Properties. Observe that this reaction

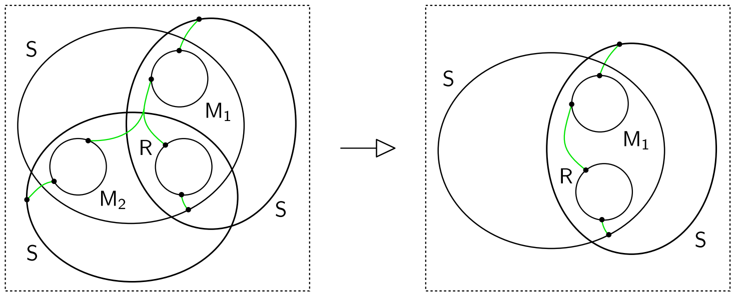


Fig. 25. Bigraphical evolution of a WLAN: machine M_2 leaves the network.

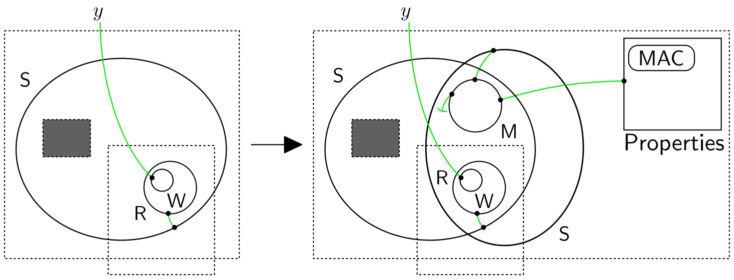


Fig. 26. Example reaction rule: a new machine appears in the router's signal range.

rule forces all M-nodes to be shared by only two S-nodes. This means our model does not capture any interference between the signals of the machines in the system: the model is based solely on information provided by the router. In other words, we have modelled exactly what the router senses.

More generally, events include moving out of the router's range, granting and revoking of DHCP leases, policy invocations and revocations such as blocking TCP traffic from a given site. The model generation process is event-driven, i.e. instead of computing the whole state space of the BRS, sequences of reaction rules are applied in real-time to update the current model of the network according to the events captured by the router of the domestic network. The analysis process consists of verifying system properties expressed as **BiLog** predicates, for example, detecting configurations that violate user-invoked access control policies. In order to verify predicates involving universal quantifiers or negated existential quantifier, a method based on tagging/untagging reaction rules is introduced, which also permits tracking of nodes through a reaction by using a class of controls as unique node identifiers. In essence, we generate a real-time simulation trace, where states are bigraph representations of the live system: at each step, the bigraph state is checked for invariants and violations are reported to the user (and the router). The system has been implemented on a Linux Eee PC laptop acting as network router, with trials on synthetic and actual network data indicating the slowest update to the bigraph model requires just under 0.10 s. Observe that in this application, an update involves a sequence of bigraphical rewriting steps. Moreover, additional instances of matching are introduced to check invariants at run-time.

8. Related work

Several extensions and refinements to Milner's definition of bigraphs have been proposed since their first introduction.

In one of the earliest extensions, the full independence of placing and linking is relaxed by assigning a locality to some links in a bigraph. Therefore, in this new setting, it matters whether a link crosses the boundaries of a place. For example, an edge bound to a certain node can only link ports that lie within that node. The structures defined by this extension are called *binding bigraphs* and formalised in [18]. Their characteristics have been proven adequate to encode calculi with name binding (e.g. π-calculus) and to model protocols with security features that enforce communications to take place only within certain boundaries (e.g. private channels). To define binding of links uniformly, names are also allowed to be located at roots and sites.

Another extension, called *directed bigraphs*, is presented in [19]. It consists of a new definition of link graphs in which edges are assigned a direction. By doing so, edges can be naturally interpreted as resources and names as requests of resources. Directed bigraph have been used to encode the fusion calculus.

A stochastic extension of BRS is introduced in [3] by attaching a stochastic rate to reaction rules. Thus, the state space generated by a stochastic BRS can be interpreted as a CTMC. Since a reaction may occur with different underlying reaction rules, the contribution of each rule has to be taken into account. Therefore, the reaction rate for a rule is obtained by the product of the rate and the number of distinct occurrences of the rule.

A categorical semantics of Milner's bigraphs based on 2-categories instead of s-categories and spm categories is formalised in [20]. A similar approach in which bigraphical structures are formulated in terms of presheaf categories is described in [21].

A general abstract theory for computation involving shared resources is presented in [22]. In particular, the author generalises recursive computation using *cyclic* sharing graphs.

In [23], Pereira et al. investigate the use of the *BigActor Model* as a formalism for modelling and controlling systems of networked mobile robotic systems with dynamic structure, such as Unmanned Air Vehicles, Autonomous Surface Vehicles and Autonomous Underwater Vehicles. The model combines the Actor Model and standard bigraphs, the result is BigActors are distributed concurrent computation entities that interact with a dynamic structure of the world modelled as a BRS. Implementation of the system and binding to a robotics middleware are currently underway.

The earliest formalisation of a matching algorithm for standard bigraphs was introduced in [7]. This algorithm is based on inference rules that are defined starting from an inductive characterisation of occurrence, i.e. a pattern P can be proven to occur in a target T by structural induction on the normal forms of the algebraic representations of P and T. Since only abstract bigraphs are supported, distinct occurrences cannot be counted. The algorithm is implemented in the rewrite engine of the BPL Tool, a prototype implementation of BRS described in [24]. A similar approach has been taken for the implementation of DBtk [25], a tool-kit for directed bigraphs. An implementation based on graph embeddings is described in [26].

9. Conclusions and future work

We have given a complete presentation of the theory of bigraphs with sharing, including a categorical semantics, algebraic properties, and several essential procedures for computation over bigraphs with sharing: matching, a SAT encoding of matching, and checking a fragment of the logic **BiLog**. Where possible we have extended the existing results for (standard) bigraphs and all our results are conservative in the sense that they still apply to standard bigraphs. Additionally, our approach to matching leads to a more efficient solution for standard bigraphs, compared with the existing approach that is based on inference.

We have given an overview of our implementation BigraphER, which provides an efficient implementation of computation, simulation, and visualisation for bigraphs with sharing. We have given an overview of two substantial, practical applications where we used bigraphs with sharing and components of BigraphER: in modelling and analysis of the IEEE 802.11 CSMA/CA protocol with RTS/CTS exchange, and in modelling network and policy management in domestic wireless networks. In the former application, overlapping wireless signals are expressed by shared nodes; locality determines collision probability and the reaction rules define network evolution during protocol execution. Quantitative analysis depends on predicates from the **BiLog** fragment. In the latter application, overlapping wireless signals are also expressed by shared nodes but reaction rules encode events such as moving in and out of router range and policy activations. Sequences of reaction rules are applied in real-time to update a model of the current network, which is then analysed according to predicates defined in the **BiLog** fragment. Both applications exhibit a fundamental requirement for sharing to express overlapping wireless signals, and they could not have been modelled in standard bigraphs (without substantial obfuscation and overheads).

Future work includes investigating further applications outside the wireless networking domain, extending BigraphER (e.g. graphical editor for bigraphs and support for sortings), optimisations to the SAT encoding of matching by a more efficient CNF encoding that introduces auxiliary variables (e.g. commander-variable encoding [27] and Tseitin transformation [28]), and extending the **BiLog** fragment. For example, in the last, we could add support for conjunctions and disjunctions over atomic ions by modifying constraint C_5.

Acknowledgements

The authors thank Simon Gay, Peter Sewell, and the anonymous reviewers of this paper for their helpful comments. This work was supported by EPSRC project Homework (EP/F064225/1) and MS is funded by an EPSRC Doctoral Prize Research Fellowship.

Appendix A. Proofs

Proof of Lemma 1. Let us define $A \circ (B \circ C) = G_0$ and $(A \circ B) \circ C = G_1$. Since the node sets V_A, V_B, V_C are all disjoint and the domains are all compatible, then by Definition 2 all the composition are defined. We have to prove that $G_0 = G_1$. Again by Definition 2, $G_0, G_1 : h \to n$,

$$V_{G_0} = V_{G_1} = V_A \uplus V_B \uplus V_C$$
$$ctrl_{G_0} = ctrl_{G_1} = ctrl_A \uplus ctrl_B \uplus ctrl_C \quad .$$

It remains to prove that $prnt_{G_0} = prnt_{G_1}$. Since both relations are sub-sets of $(h \uplus V_{G_0}) \times (V_{G_0} \uplus n)$, we have to show that $(v, w) \in prnt_{G_0}$ if and only if $(v, w) \in prnt_{G_1}$ for every element (v, w). The parent relations are defined as

$$prnt_{G_0} = prnt_A^{\lhd} \uplus prnt_{\circ} \uplus prnt_{B\circ C}^{\rhd} \tag{A.1}$$
$$prnt_{G_1} = prnt_{A\circ B}^{\lhd} \uplus prnt'_{\circ} \uplus prnt_C^{\rhd} \quad . \tag{A.2}$$

To analyse the single components we also compute the parent relations for the compositions $B \circ C$ and $A \circ B$:

$$prnt_{B\circ C} = prnt_B^{\lhd} \uplus prnt_{\circ}^{BC} \uplus prnt_C^{\rhd}$$
$$prnt_{A\circ B} = prnt_A^{\lhd} \uplus prnt_{\circ}^{AB} \uplus prnt_B^{\rhd} \quad .$$

Therefore,

$$\begin{aligned} prnt_{B\circ C}^{\rhd} &= prnt_{B\circ C} \rhd (V_B \uplus V_C) \\ &= (prnt_B^{\lhd} \uplus prnt_{\circ}^{BC} \uplus prnt_C^{\rhd}) \rhd (V_B \uplus V_C) \\ &= (prnt_B^{\lhd} \rhd (V_B \uplus V_C)) \uplus (prnt_{\circ}^{BC} \rhd (V_B \uplus V_C)) \uplus (prnt_C^{\rhd} \rhd (V_B \uplus V_C)) \\ &= (V_B \lhd prnt_B \rhd V_B) \uplus (prnt_{\circ}^{BC} \rhd V_B) \uplus prnt_C^{\rhd} \end{aligned}$$

and similarly

$$prnt_{A\circ B}^{\lhd} = prnt_A^{\lhd} \uplus (V_B \lhd prnt_{\circ}^{AB}) \uplus (V_B \lhd prnt_B \rhd V_B) \quad .$$

Then (A.1) and (A.2) can be rewritten as

$$prnt_{G_0} = prnt_A^{\lhd} \uplus prnt_{\circ} \uplus (V_B \lhd prnt_B \rhd V_B) \uplus (prnt_{\circ}^{BC} \rhd V_B) \uplus prnt_C^{\rhd}$$
$$prnt_{G_1} = prnt_A^{\lhd} \uplus (V_B \lhd prnt_{\circ}^{AB}) \uplus (V_B \lhd prnt_B \rhd V_B) \uplus prnt'_{\circ} \uplus prnt_C^{\rhd} \quad .$$

Hence, to prove that $prnt_{G_0} = prnt_{G_1}$ we have to show that

$$prnt_{\circ} \uplus (prnt_{\circ}^{BC} \rhd V_B) = (V_B \lhd prnt_{\circ}^{AB}) \uplus prnt'_{\circ}$$

holds. We start by proving $\Rightarrow$. We have the following cases:

1. If $(v, w) \in prnt_{\circ}$ then there exists a $w' \in m$ such that $(v, w') \in prnt_{B\circ C}$ and $(w', w) \in prnt_A$, where $v \in h \uplus V_C \uplus V_B$ and $w \in V_A \uplus n$. There are two sub-cases:
   (a) If $v \in V_B$ then $(v, w') \in prnt_B$. Therefore, $(v, w) \in (V_B \lhd prnt_{\circ}^{AB})$.
   (b) If $v \in h \uplus V_C$ then there exists a $w'' \in k$ such that $(v, w'') \in prnt_C$ and $(w'', w') \in prnt_B$. But then $(w'', w) \in prnt_{A\circ B}$. It follows that $(v, w) \in prnt'_{\circ}$.
2. If $(v, w) \in (prnt_{\circ}^{BC} \rhd V_B)$ then there exists a $w' \in k$ such that $(v, w') \in prnt_C$ and $(w', w) \in prnt_B$, where $v \in h \uplus V_C$ and $w \in V_B$. But we also have that $(w', w) \in prnt_{A\circ B}$. Hence, $(v, w) \in prnt'_{\circ}$.

The proof for $\Leftarrow$ is symmetric. This concludes the proof. □

Proof of Lemma 2. All the compositions are defined and $V_{\mathsf{id}_m} = V_{\mathsf{id}_n} = \emptyset$. The composite $G \circ \mathsf{id}_m = (V, ctrl, prnt)$ is defined according to 2. In particular, we have $V = V_G \uplus \emptyset$, $ctrl = ctrl_G$ and

$$prnt = prnt_G^{\lhd} \uplus prnt_{\circ} \uplus (\mathsf{Id}_m \rhd \emptyset) \quad .$$

But, $\mathsf{Id}_\mathsf{m} \rhd \emptyset = \emptyset$ and

$$prnt_{\circ} = (m \lhd prnt_G) \circ (\mathsf{Id} \rhd m) = m \lhd prnt_G \quad .$$

It follows that $prnt = prnt_G$ and then $G \circ \mathsf{id}_m = G$. The proof for $\mathsf{id}_n \circ G = G$ is similar. □

Proof of Lemma 3. Let us define $A \otimes (B \otimes C) = G_0$ and $(A \otimes B) \otimes C = G_1$. Since the node sets are all disjoint, by Definition 3, all the products are defined. We have to prove that $G_0 = G_1$. Associativity of $\uplus$ and $+$ assures that

$$\begin{aligned} V_{G_0} = V_{G_1} &= V_A \uplus V_B \uplus V_C \\ ctrl_{G_0} = ctrl_{G_1} &= ctrl_A \uplus ctrl_B \uplus ctrl_C \\ m = m_0 + (m_1 + m_2) &= (m_0 + m_1) + m_2 \\ n = n_0 + (n_1 + n_2) &= (n_0 + n_1) + n_2 \quad . \end{aligned}$$

It remains to prove that $prnt_{G_0} = prnt_{G_1}$. By construction the following equalities hold:

$$prnt_{G_0} = prnt_A \uplus prnt_{B\otimes C}^{(m_0,n_0)} \qquad prnt_{G_1} = prnt_{A\otimes B} \uplus prnt_C^{(m_0+m_1,n_0+n_1)}$$

with

$$prnt_{B\otimes C} = prnt_B \uplus prnt_C^{(m_1,n_1)} \qquad prnt_{A\otimes B} = prnt_A \uplus prnt_B^{(m_0,n_0)} \quad .$$

Hence, we have

$$prnt_{G_0} = prnt_A \uplus prnt_B^{(m_0,n_0)} \uplus prnt_C^{(m_0+m_1,n_0+n_1)} = prnt_{G_1} \quad . \qquad \square$$

Proof of Lemma 4. Immediate from $prnt_{\mathsf{id}_0} = \emptyset$. $\square$

Proof of Lemma 5. Assume the hypothesis holds but $v' \notin (k \uplus V_F)$. Then $v' \in V_G$. Since $v \in V_F$ and $(v', v) \in prnt_B^+$, we have

$$(v', v_0), \dots, (v_i, m_i) \in prnt_G \qquad (m_i, v_{i+1}), \dots, (v_{i+j}, v) \in prnt_F$$

for some $m_i \in m$. However, by Definition 2, $prnt_G \subseteq (m \uplus V_G) \times (V_G \uplus n)$. Therefore, $(v_i, m_i) \notin prnt_G$ because $m_i \notin (V_G \uplus n)$. Similarly, $(m_i, v_{i+1}) \notin prnt_F$ because $m_i \notin (k \uplus V_F)$. This contradicts the hypothesis. Hence, $v' \in (k \uplus V_F)$. $\square$

Proof of Lemma 6. Immediate by Definition 2. Assume S does not exist. Then, it means that $prnt_B(v) \subseteq V_F$. But $(prnt_B \rhd (V_G \uplus n))(v) = \emptyset$, so we have a contradiction. $\square$

Proof of Lemma 7. Dual of Lemma 6. $\square$

Proof of Proposition 8. By Definition 2, composition is a partial operation. Moreover, given $G : m \to n$ and $F : k \to m'$, when $G \circ F$ is defined then $m = m'$. Additionally, lemmata 1 and 2 prove that composition is associative and identities are its neutral elements, respectively. $\square$

Proof of Lemma 9. Define $(A_1 \otimes B_1) \circ (A_0 \otimes B_0) = G_0$ and $(A_1 \circ A_0) \otimes (B_1 \circ B_0) = G_1$. By Definitions 2 and 3, we have

$$\begin{aligned} V_{G_0} = V_{G_1} &= V_{A_0} \uplus V_{A_1} \uplus V_{B_0} \uplus V_{B_1} \\ ctrl_{G_0} = ctrl_{G_1} &= ctrl_{A_0} \uplus ctrl_{A_1} \uplus ctrl_{B_0} \uplus ctrl_{B_1} \quad . \end{aligned}$$

It remains to prove that $prnt_{G_0} = prnt_{G_1}$. By construction we have

$$prnt_{G_0} = prnt^{\lhd}_{A_1\otimes B_1} \uplus prnt_{\circ} \uplus prnt^{\rhd}_{A_0\otimes B_0} \tag{A.3}$$

where the sub-sets are given by

$$prnt^{\lhd}_{A_1\otimes B_1} = prnt^{\lhd}_{A_1} \uplus prnt_{B_1}^{(\_,n_2)\lhd} \qquad prnt^{\rhd}_{A_0\otimes B_0} = prnt^{\rhd}_{A_0} \uplus prnt_{B_0}^{(n_0,\_)\rhd}$$

and

$$prnt_{G_1} = prnt_{A_1\circ A_0} \uplus prnt_{B_1\circ B_0}^{(n_0,n_2)} \tag{A.4}$$

with

$$\begin{aligned} prnt_{A_1\circ A_0} &= prnt^{\lhd}_{A_1} \uplus prnt^{l}_{\circ} \uplus prnt^{\rhd}_{A_0} \\ prnt_{B_1\circ B_0}^{(n_0,n_2)} &= prnt_{B_1}^{(\_,n_2)\lhd} \uplus prnt_{\circ}^{(n_0,n_2)r} \uplus prnt_{B_0}^{(n_0,\_)\rhd} \quad . \end{aligned}$$

We write $prnt_{B_1}^{(\_,n_2)\lhd}$ and $prnt_{B_0}^{(n_0,\_)\rhd}$ to highlight the fact that these sets do not contain pairs with sites and regions to be shifted to the right. By equating the left-hand sides of (A.3) and (A.4), we obtain

$$prnt_{\circ} = prnt^{l}_{\circ} \uplus prnt_{\circ}^{(n_0,n_2)r} \tag{A.5}$$

By expanding the left-hand side, the equation can be rewritten as follows

$$\begin{aligned} prnt_\circ &= ((n_1+m_1) \lhd prnt_{A_1 \otimes B_1}) \circ (prnt_{A_0 \otimes B_0} \rhd (n_1+m_1)) \\ &= ((n_1+m_1) \lhd (prnt_{A_1} \uplus prnt_{B_1}^{(n_1,n_2)})) \circ ((prnt_{A_0} \uplus prnt_{B_0}^{(n_0,n_1)}) \rhd (n_1+m_1)) \\ &= ((n_1 \lhd prnt_{A_1}) \uplus ((n_1+m_1) \lhd prnt_{B_1}^{(n_1,n_2)})) \circ ((prnt_{A_0} \rhd n_1) \uplus (prnt_{B_0}^{(n_0,n_1)} \rhd (n_1+m_1))) \\ &= ((n_1 \lhd prnt_{A_1}) \circ (prnt_{A_0} \rhd n_1)) \uplus (((n_1+m_1) \lhd prnt_{B_1}^{(n_1,n_2)}) \circ (prnt_{B_0}^{(n_0,n_1)} \rhd (n_1+m_1))) \\ &= prnt_\circ^l \uplus prnt_\circ^{(n_0,n_2)r} \ . \end{aligned}$$

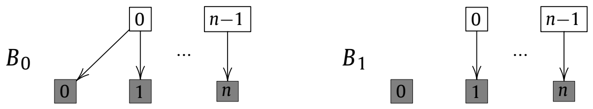
This concludes the proof. □

Proof of Lemma 10. The parent relations are Id_{m+n} and $\mathsf{Id}_{m \otimes n}$. Since $m \otimes n = m+n$ we have equality. □

Proof of Proposition 11. Immediate. We already proved bifunctoriality of tensor product. Since symmetries are defined like in $\widetilde{\mathbf{Pg}}(\mathcal{K})$, it is routine to prove that the same properties hold. □

Proof of Proposition 12. Recall that $B : m \to n$ is epi if $B_0 \circ B = B_1 \circ B$ implies $B_0 = B_1$ for any $B_0, B_1 : n \to h$. We start by proving $\Rightarrow$. Assume B is epi; we prove that it has no idle regions (i) and no two regions are partners (ii).

(i) Suppose there is an idle region. Without loss of generality we set region $0 \in n$ idle. Pick $B_0, B_1 : n \to n-1$ as below



Then $B_0 \neq B_1$ but $B_0 \circ B = B_1 \circ B$, contradicting B is epi.

(ii) Suppose regions $0, 1 \in n$ are partners. Pick B_0, B_1 as in the previous case. Then $B_0 \neq B_1$ but $B_0 \circ B = B_1 \circ B$, contradicting B is epi.

Now we prove $\Leftarrow$. Assume B has no idle regions and no two regions are partners; we prove it to be epi. Let $B_0 \circ B = B_1 \circ B$. Then B_0 and B_1 have the same nodes and control map; so to prove $B_0 = B_1$ we have to show that $prnt_{B_0} = prnt_{B_1}$. By definition of composition for place graphs with sharing, equality $prnt_{B_0 \circ B} = prnt_{B_1 \circ B}$ leads to

$$prnt_{B_0}^{\lhd} \uplus prnt_\circ = prnt_{B_1}^{\lhd} \uplus prnt'_\circ \ . \tag{A.6}$$

Assume $prnt_{B_0}^{\lhd} \neq prnt_{B_1}^{\lhd}$. Then we have two cases:

1. There exists an element $(v, w) \in prnt_{B_0}^{\lhd}$ (i.e. with $v \in V_{B_0}$) such that $(v, w) \notin prnt_{B_1}^{\lhd}$. Then by (A.6), $(v, w) \in prnt'_\circ$. But by construction of $prnt'_\circ$, $v \in V_B \uplus m$. Hence, $v \notin V_{B_0}$. This is a contradiction.
2. There is an element $(v, w) \in prnt_{B_1}^{\lhd}$ such that $(v, w) \notin prnt_{B_0}^{\lhd}$. By contradiction as in the previous case.

This proves $prnt_{B_0}^{\lhd} = prnt_{B_1}^{\lhd}$. Together with (A.6), it implies $prnt_\circ = prnt'_\circ$. By expanding the two terms, we can write

$$(n \lhd prnt_{B_0}) \circ (prnt_B \rhd n) = (n \lhd prnt_{B_1}) \circ (prnt_B \rhd n) \ .$$

By hypothesis, B has no idle regions and no two regions are partners. Therefore $prnt_B \rhd n$ is a surjective partial function. Thus, it is epi in **Rel**. It follows that $(n \lhd prnt_{B_0}) = (n \lhd prnt_{B_1})$. Since

$$(n \lhd prnt_{B_i}) \uplus prnt_{B_i}^{\lhd} = prnt_{B_i} \qquad i = 0, 1 \ ,$$

we proved $prnt_{B_0} = prnt_{B_1}$. This concludes the proof. □

Proof of Proposition 13. Immediate by invoking Proposition 12 and self-duality of $\mathbf{SPg}(\mathcal{K})$. Observe that orphan sites and siblings correspond to idle regions and partners, respectively. □

Proof of Proposition 15. Immediate by Definition 12 and by Proposition 14. □

Proof of Proposition 16. Immediate by Proposition 15 and Milner's discrete normal form (DNF) [1] for link graphs. □

Proof of Lemma 20. We need to prove that whenever T and P are non-sharing, then context C and parameter D in *any* decomposition constructed by MATCH(T, P) are also non-sharing. We begin by proving that D is non-sharing.

Assume D is a sharing bigraph. Then by Definition 1 its place graph can contain i) orphans and ii) shared places. We prove the two cases separately.

i) Let c be an orphan in D. Since c is also a place in T, then by Proposition 18

$$prnt_D(c) = prnt_T(c) = \emptyset \ .$$

But T is non-sharing, therefore $|prnt_T(c)| = 1$. This is a contradiction.

ii) Without loss of generality, let $prnt_D(c) = \{p, p'\}$ with $\{p, p'\} \subseteq (V_D \uplus j)$. There are six cases:

1) If $\{p, p'\} \subseteq V_D$, then by Proposition 18 $prnt_D \triangleright V_D = prnt_T \triangleright V_D$. Therefore, $prnt_T(c) = \{p, p'\}$. But this is a contradiction because T is non-sharing by hypothesis.
2) If $p \in V_D$ and $p' \in m$, then by Proposition 18 and by the fact that P is non-sharing, there exists one $p'' \in V_{P'}$ such that $(c, p'') \in prnt_T$. However, Proposition 18 also implies that $(c, p) \in prnt_T$, which contradicts the hypothesis.
3) If $p \in V_D$ and $p' \in j$, then by Proposition 18 there exists one $p'' \in V_C$ such that $(c, p'') \in prnt_T$. As in the previous case, this contradicts the hypothesis.
4) If $\{p, p'\} \subseteq m$ then by Proposition 18 and by the fact that P is non-sharing, we have that $|prnt_T(c)| = 2$. But this is a contradiction because T is non-sharing by hypothesis.[4]
5) If $\{p, p'\} \subseteq j$ then c has a parent in C. The construction of id_j described in Proposition 18 ensures the minimality of j. Therefore, there is only one site in id_j connecting c with its parent in C. This implies that $(c, p') \notin prnt_D$. Contradiction.
6) If $p \in m$ and $p' \in j$, then by Proposition 18 and by the fact that P does not have any orphan sites, we have that $\{(c, v), (c, p'')\} \in prnt_T$, with $v \in V_{P'}$ and p'' in C. This is a contradiction because T is non-sharing by hypothesis.

The proof for C is similar and thus omitted. □

References

[1] R. Milner, The Space and Motion of Communicating Agents, Cambridge University Press, 2009.
[2] S. Mimram, The structure of first-order causality, in: Proceedings of the 2009 24th Annual IEEE Symposium on Logic In Computer Science, LICS '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 212–221.
[3] J. Krivine, R. Milner, A. Troina, Stochastic bigraphs, Electron. Notes Theor. Comput. Sci. 218 (2008) 73–96.
[4] R. Shamir, D. Tsur, Faster subtree isomorphism, J. Algorithms 33 (1999) 267–280.
[5] J. Ullmann, An algorithm for subgraph isomorphism, J. ACM 23 (1976) 31–42.
[6] T. Werth, Design and Implementation of a DAG-Miner, Diploma thesis, Friedrich-Alexander-Universität, Erlangen-Nürnberg, 2007.
[7] L. Birkedal, T. Damgaard, A. Glenstrup, R. Milner, Matching of bigraphs, Electron. Notes Theor. Comput. Sci. 175 (2007) 3–19.
[8] H. Kautz, D. McAllester, B. Selman, Encoding plans in propositional logic, in: Proc. International Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann Publishers, 1996, pp. 374–385.
[9] K. Claessen, N. Een, M. Sheeran, N. Sorensson, SAT-solving in practice, in: 9th International Workshop on Discrete Event Systems, WODES, 2008, IEEE, 2008, pp. 61–67.
[10] M. Sevegnani, C. Unsworth, M. Calder, A SAT based algorithm for the matching problem in bigraphs with sharing, Technical Report TR-2010-311, University of Glasgow, 2010.
[11] G. Conforti, D. Macedonio, V. Sassone, Spatial logics for bigraphs, in: L. Caires, G.F. Italiano, L. Monteiro, C. Palamidessi, M. Yung (Eds.), Proceedings of the 32th International Colloquium on Automata, Languages and Programming, ICALP'05, in: Lecture Notes in Computer Science, vol. 3580, Springer-Verlag, 2005, pp. 766–778.
[12] M. Sevegnani, BigraphER: Bigraph Evaluator and Rewriting, http://www.dcs.gla.ac.uk/~michele/bigrapher.html, 2013.
[13] Graphviz, Graphviz – graph visualization software, http://www.graphviz.org/, 2012.
[14] N. Eén, N. Sörensson, MiniSat: a minimalistic and high-performance SAT solver, http://minisat.se/, 2012.
[15] M. Calder, M. Sevegnani, Modelling IEEE 802.11 CSMA/CA RTS/CTS with stochastic bigraphs with sharing, Form. Asp. Comput. 26 (2014) 537–561.
[16] M. Kwiatkowska, G. Norman, D. Parker, PRISM 4.0: verification of probabilistic real-time systems, in: G. Gopalakrishnan, S. Qadeer (Eds.), Proc. 23rd International Conference on Computer Aided Verification, CAV'11, in: Lecture Notes in Computer Science, vol. 6806, Springer, 2011, pp. 585–591.
[17] M. Calder, A. Koliousis, M. Sevegnani, J. Sventek, Real-time verification of wireless home networks using bigraphs with sharing, Sci. Comput. Program. 80 (Part B) (2014) 288–310.
[18] T. Damgaard, L. Birkedal, Axiomatizing binding bigraphs, Nordic J. Comput. 13 (2006) 58–77.
[19] D. Grohmann, M. Miculan, Directed bigraphs, Electron. Notes Theor. Comput. Sci. 173 (2007) 121–137.
[20] F. Bonchi, U. Montanari, A coalgebraic theory of reactive systems, Electron. Notes Theor. Comput. Sci. 209 (2008) 201–215.
[21] M. Miculan, M. Peressotti, Bigraphs reloaded, Technical Report UDMI/01/2013/RR, Dept. of Mathematics and Computer Science, Univ. of Udine, 2013.
[22] M. Hasegawa, Models of sharing graphs, Ph.D. thesis, School of Informatics, The University of Edinburgh, 1997.
[23] E. Pereira, C. Kirsch, R. Sengupta, J. Sousa, BigActors – a model for structure-aware computation, in: 4th International Conference on Cyber-Physical Systems, ACM/IEEE, 2013, pp. 199–208.
[24] A. Glenstrup, T. Damgaard, L. Birkedal, E. Højsgaard, An implementation of bigraph matching, Technical Report TR-2010-135, IT University of Copenhagen, 2010.
[25] G. Bacci, D. Grohmann, M. Miculan, DBtk: a toolkit for directed bigraphs, in: Proceedings of the 3rd International Conference on Algebra and Coalgebra in Computer Science, CALCO'09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 413–422.
[26] G. Perrone, Domain-specific modelling languages in bigraphs, Ph.D. thesis, IT University of Copenhagen, 2013.
[27] W. Klieber, G. Kwon, Efficient CNF encoding for selecting 1 from N objects, in: Proc. International Workshop on Constraints in Formal Verification, 2007.
[28] G.S. Tseitin, On the complexity of derivation in propositional calculus, in: Studies in Constructive Mathematics and Mathematical Logic, vol. 2, 1968, pp. 10–13.

[4] Observe that by definition of procedure SITES(_, _, _), it is impossible to have $\{(p, v), (p', v)\} \subseteq prnt_P$ if $prnt_D(c) = \{p, p'\}$.