BiLog: Spatial Logics for Bigraphs [1]

Giovanni Conforti [a] Damiano Macedonio [b] Vladimiro Sassone [b]

[a]*University of Dortmund*

[b]*University of Sussex*

Abstract

Bigraphs are emerging as a (meta-)model for concurrent calculi, like CCS, ambients, π-calculus, and Petri nets. They are built orthogonally on two structures: a hierarchical place graph for locations and a link (hyper-)graph for connections. Aiming at describing bigraphical structures, we introduce a general framework, BiLog, whose formulae describe arrows in monoidal categories. We then instantiate the framework to bigraphical structures and we obtain a logic that is a natural composition of a place graph logic and a link graph logic. We explore the concepts of separation and sharing in these logics and we prove that they generalise well known spatial logics for trees, graphs and tree contexts. As an application, we show how XML data with links and web services can be modelled by bigraphs and described by BiLog. The framework can be extended by introducing dynamics in the model and a standard temporal modality in the logic. However, in some cases, temporal modalities can be already expressed in the static framework. To testify this, we show how to encode a minimal spatial logic for CCS in an instance of BiLog.

Key words: Concurrency, Bigraphs, Spatial Logics, Context Logic, Separation, XML

1 Introduction

To describe and reason about structured, distributed, and dynamic resources is one of the main goals of global computing research. Recently, many *spatial logics* have been studied to fulfill this aim. The term 'spatial,' as opposed to 'temporal,' refers to the use of modal operators inspecting the structure of the terms in the model,

[1] Research partially supported by the projects '**DisCo**: Semantic Foundations of Distributed Computation', EU IHP 'Marie Curie' HPMT-CT-2001-00290, '**MIKADO**: Mobile Calculi based on Domains', EU FET-GC IST-2001-32222, and '**MyThS**: Models and Types for Security in Mobile Distributed Systems', EU FET-GC IST-2001-32617.

rather than a temporal behaviour. Spatial logics are usually equipped with a separation/composition operator that *splits* a term into two parts, to ‘talk’ about them separately. The notion of *separation* is interpreted differently in different logics.

- In ‘separation’ logics [34], it is used to reason about dynamic update of heap-like structures, and it is *strong* as it forces names of resources in separated components to be disjoint. Consequently, term composition is usually partially defined.
- In static spatial logics, for instance for trees [6], graphs [10] and trees with hidden names [11], the separation/composition does not require any constraint on terms, and names are usually shared between separated parts.
- In dynamic spatial logics, too, the separation is intended only for locations in space (e.g. for ambients [13] or π-calculus [4]).

Context tree logic, introduced in [7], integrates the first approach above with a spatial logic for trees. The result is a logic able to express properties of tree-shaped structures (and contexts) with pointers, and it is used as an assertion language for Hoare-style program specifications in a tree memory model. Essentially, Spatial Logic founds its semantics on model structure.

Bigraphs [25,28] are an emerging model for structures in global computing, that can be instantiated to model several well-known examples, including λ-calculus [31], CCS [32], π-calculus [25], ambients [26] and Petri nets [29]. Bigraphs consist essentially of two graphs sharing the same nodes. The first graph, the *place graph*, is tree structured and expresses a hierarchical relationship on nodes (viz. locality in space and nesting of locations). The second graph, the *link graph*, is an hyper-graph and expresses a generic *“many-to-many”* relationship among nodes (e.g. data link, sharing of a channel). The two structures are orthogonal, so links between nodes can cross locality boundaries. Thus, clarify the difference between structural separation (i.e., separation in the place graph) and name separation (i.e., separation on the link graph).

In this paper we introduce a spatial logic for bigraphs as a natural composition of a place graph logic, for tree contexts, and a link graph logic, for name linkings. The main point is that a resource has a spatial structure as well as a link structure associated to it. Suppose for instance to be describing a tree-shaped distribution of resources in locations. We may use an atomic formula like $\mathsf{PC}(A)$ to describe a resource of ‘type’ PC (e.g. a personal computer) whose contents satisfy A, and a formula like $\mathsf{PC}_x(A)$ to describe the same resource at the location x. Note that the location type is orthogonal to the name. We can then write $\mathsf{PC}(\mathbf{T}) \otimes \mathsf{PC}(\mathbf{T})$ to characterise terms with two unnamed PC resources whose contents satisfy the tautological formula (i.e., with anything inside). Named locations, as e.g. in $\mathsf{PC}_a(\mathbf{T}) \otimes \mathsf{PC}_b(\mathbf{T})$, can express name separation, i.e., that names a and b are different (because separated by $\otimes$). Furthermore, link expressions can force name-sharing between resources with formulae like $\mathsf{PC}_a(\mathsf{in}_c \otimes \mathbf{T}) \overset{c}{\otimes} \mathsf{PC}_b(\mathsf{out}_c \otimes \mathbf{T})$. The formula describes two PC with different names, a and b, ‘uniquely’ sharing a link on a distinct name

c, which models, e.g. a communication channel. Name c is used as input (in) for the first PC and as an output (out) for the second PC. No other name is shared and c cannot be used elsewhere inside PCs.

A bigraphical structure is, in general, a context with several holes and open links that can be filled by composition. The logic therefore describes contexts for resources at no additional cost. We can then express formulae like $\mathsf{PC}_a(\mathbf{T} \otimes \mathsf{HD}(id_1))$, that describes a modular computer PC, where id_1 represents a 'plug-able' hole in the hard disc HD. Contextual resources have many important applications. In particular, the contextual nature of bigraphs is useful to characterise their dynamics, but it can also be used as a general mechanism to describe contexts of bigraphical data structures (cf. [19,23]).

As bigraphs are establishing themselves as a truly general (meta)model of global systems, and appear to encompass several existing calculi and models (see for instance [25,26,29,32]), our bigraph logic, *BiLog*, aims at achieving the same generality as a description language: as bigraphs specialise to particular models, we expect BiLog to specialise to powerful logics on these. In this sense, the contribution of this paper is to propose BiLog as a unifying language for the description of global resources. We will explore this path in future work, fortified by the embedding results for the static spatial logics presented in §5, and the positive preliminary results obtained for semistructured data (cf.§6) and CCS (cf.§7).

The paper is organised as follows: §2 provides a crash course on bigraphs; §3 introduces the general framework and model theory of BiLog; §4 shows how to derive some useful connectives, such as a temporal modality and assertions constraining the "type" of terms; §5 instantiates the framework to obtain logics for place, link and bi-graphs; §6 focus on the applications of BiLog to XML data; §7 studies how to deal with dynamic models. An abridged version of this work appears in the conference paper [20] and the application to XML was presented in [19]. Here a new embedding result for a dynamic logic based on CCS [5] is added to our main technical result, that is the embedding of the static spatial logics of [6], [10] and [7] by BiLog. In particular, CCS embedding is based on an structural way of expressing the 'next-step' modality by composition adjuncts and bigraphical contexts. Moreover we show proofs, examples and properties more in detail. Further examples and technical details can be found in [17,27].

2 An informal introduction to Bigraphs

Bigraphs formalise distributed systems by focusing on two of their main characteristics: locality and interconnections. A bigraph consists of a set of *nodes*, which may be nested in a hierarchical tree structure, the so-called *place graph*, and have

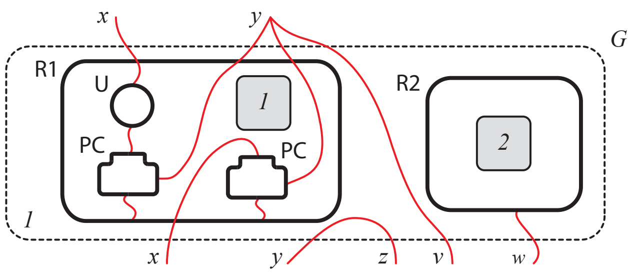

Fig. 1. A bigraph $G : \langle 2, \{x, y, z, v, w\}\rangle \rightarrow \langle 1, \{x, y\}\rangle$.

ports that may be connected to each other by *links*, the so-called *link graph*. Place graphs express locality, that is the physical arrangement of the nodes. Link graphs are hyper-graphs and formalise connections among nodes. The orthogonality of the two structures dictates that nestings impose no constrain upon interconnections.

The bigraph G of Fig. 1 represents a system where people and things interact. We imagine two offices with employees logged on PCs. Every entity is represented by a node, shown with bold outlines, and every node is associated with a *control* (either PC, U, R1, R2). Controls represent the kinds of nodes, and have fixed *arities* that determine their number of ports. Control PC marks nodes representing personal computers, and its arity is 3: in clockwise order, the ports represent a keyboard interacting with an employee U, a LAN connection interacting with another PC and open to the outside network, and the mains plug of the office R. The employee U may communicate with another one via the upper port in the picture. The nesting of nodes (place graph) is shown by the inclusion of nodes into each other; the connections (link graph) are drawn as lines.

At the top level of the nesting structure sit the *regions*. In Fig. 1 there is one sole region (the dotted box). Inside nodes there may be ‘context’ *holes*, drawn as shaded boxes, which are uniquely identified by ordinals. The hole marked by 1 represents the possibility for another user U to get into office R1 and sit in front of a PC. The hole marked by 2 represents the possibility to plug a subsystem inside office R2.

Place graphs can be seen as *arrows* over a symmetric monoidal category whose objects are finite ordinals. We write $P : m \rightarrow n$ to indicate a place graph P with m holes and n regions. In Fig. 1, the place graph of G has type $2 \rightarrow 1$. Given the place graphs P_1, P_2, their composition $P_1 \circ P_2$ is defined only if the holes of P_1 are as many as the regions of P_2, and amounts to *filling* holes with regions, according to the number each carries. The tensor product $P_1 \otimes P_2$ is not commutative, as it lays the two place graphs one next to the other (in order), thus obtaining a graph with more regions and holes, and it ‘renumbers’ regions and holes ‘from left to right’.

Link graphs are arrows of a partial monoidal category whose objects are (finite) sets of names. In particular, we assume a denumerable set Λ of names. A link graph is an arrow $X \rightarrow Y$, with X, Y finite subsets of Λ. The set X represents the *inner* names (drawn at the bottom of the bigraph) and Y represents the set of *outer* names

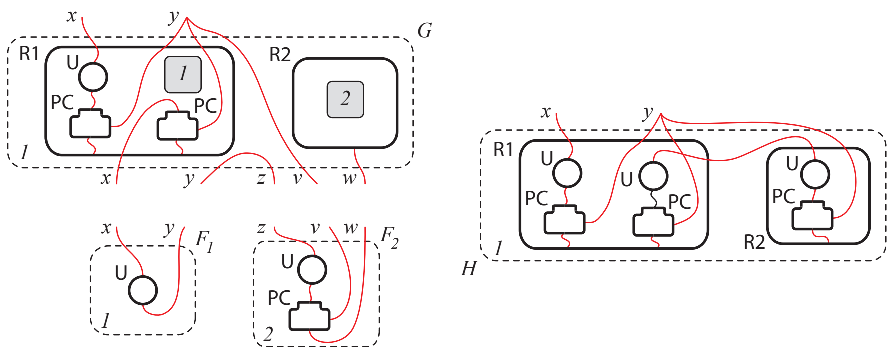

Fig. 2. Bigraphical composition, $H \equiv G \circ (F_1 \otimes F_2)$.

(drawn on the top). The link graph connects ports to names or to *edges* (represented in Fig. 1 by a line between nodes), in any finite number. A link to a name is *open*, i.e., it may be connected to other nodes as an effect of composition. A link to an edge is *closed*, as it cannot be further connected to ports. Thus, edges are *private*, or hidden, connections. The composition of link graphs $W \circ W'$ corresponds to *linking* the inner names of W with the corresponding outer names of W' and forgetting about their identities. As a consequence, the outer names of W' (resp. inner names of W) are not necessarily inner (resp. outer) names of $W \circ W'$. Thus link graphs can perform substitution and renaming, so the outer names in W' can disappear in the outer names of this means that either names may be renamed or edges may be added to the structure. As in [25], the tensor product of link graphs is defined in the obvious way only if their inner (resp. outer) names are disjoint.

By combining ordinals with names we obtain *interfaces*, i.e., couples $\langle m, X\rangle$ where m is an ordinal and X is a finite set of names. By combining the notion of place graph and link graphs on the same nodes we obtain the notion of bigraphs, i.e., arrows $G : \langle m, X\rangle \rightarrow \langle n, Y\rangle$.

Figure 2 represents a more complex situation. Its top left-hand side reports the system of Fig. 1, in its bottom left-hand side F_1 represents a user U ready to interact with a PC or with some other users, F_2 represents a user logged on its laptop, ready to communicate with other users. The system with F_1 and F_2 represents the tensor product $F = F_1 \otimes F_2$. The right-hand side of Fig. 2 represents the composition $G \circ F$. The idea is to insert F into the context G. The operation is partially defined, since it requires the inner names and the number of holes of G to match the outer names and the number of regions of F, respectively. Shared names create the new links between the two structures. Intuitively, composition *first* places every region of F in the proper hole of G (place composition) and *then* joins equal inner names of G and outer names of F (link composition). In the example, as a consequence of the composition the user U in the first region of F is logged on PC, the user U in the second region of F is in room R2. Moreover note the edge connecting the inner names y and z in G, its presence produces a link between the two users of F after the composition, imagine a phone call between the two users.

Table 3.1. *BiLog terms*

$G, G' ::=$	Ω	constructor (for $\Omega \in \Theta$)
	$G \circ G'$	vertical composition
	$G \otimes G'$	horizontal composition

3 BiLog: syntax and semantics

The final aim of the paper is to define a logic able to describe bigraphs and their substructures. Since bigraphs, place graphs, and link graphs are arrows of a (partial) monoidal category, we first introduce a meta-logical framework having monoidal categories as models; then we adapt it to model the orthogonal structures of place and link graphs. Finally, we specialise the logic to model the whole structure of (abstract) bigraphs.

Following the approach of spatial logics, we introduce connectives that reflect the structure of the model. In this case, models are monoidal categories and the logic describes spatially the structure of their *arrows*.

The meta-logical framework we propose is inspired by the bigraph axiomatisation presented in [30]. The model of the logic is composed by *terms* of a general language with *horizontal* and *vertical* compositions and a set of unary constructors. Terms are related by a *structural congruence* that satisfies the axioms of monoidal categories, and possibly more. The corresponding model theory is parameterised on basic constructors and structural congruence. To be as free as possible in choosing the level of intensionality, the logic is defined on a *transparency* predicate. Its role is to identify the terms allowing inspection of their content, *transparent* terms, and the ones that do not, *opaque* terms. We inspect the logical equivalence induced by the logic and we observe that it corresponds to the structural congruence when every term is transparent, and it becomes less discriminating with the introduction of *opaque terms*, cf. §3.2.

3.1 Terms

To evaluate formulae, we consider the terms freely generated from a set of constructors Θ, ranged over by Ω, by using the (partial) operators: composition ($\circ$) and tensor ($\otimes$). The order of binding precedence is $\circ$, $\otimes$. BiLog terms are defined in Tab. 3.1. When defined, these two operations must satisfy the *bifunctoriality property* of monoidal categories, thus we refer to these terms also as *bifunctorial terms*.

Terms represent structures built on a (partial) monoid $(M, \otimes, \epsilon)$ whose elements are dubbed *interfaces* and denoted by I, J. To model nominal resources, such as heaps or link graphs, we allow the monoid to be partial.

Table 3.2. *Typing rules*

$$\frac{type(\Omega) = I \to J}{\Omega : I \to J} \qquad \frac{G : I' \to J \quad F : I \to I'}{G \circ F : I \to J}$$

$$\frac{G : I_1 \to J_1 \quad F : I_2 \to J_2 \quad I = I_1 \otimes I_2 \quad J = J_1 \otimes J_2}{G \otimes F : I \to J}$$

Table 3.3. *Axioms*

Congruence Axioms:	
$G \equiv G$	Reflexivity
$G \equiv G'$ implies $G' \equiv G$	Symmetry
$G \equiv G'$ and $G' \equiv G''$ implies $G \equiv G''$	Transitivity
$G \equiv G'$ and $F \equiv F'$ implies $G \circ F \equiv G' \circ F'$	Congruence $\circ$
$G \equiv G'$ and $F \equiv F'$ implies $G \otimes F \equiv G' \otimes F'$	Congruence $\otimes$
Monoidal Category Axioms:	
$G \circ id_I \equiv G \equiv id_J \circ G$	Identity
$(G_1 \circ G_2) \circ G_3 \equiv G_1 \circ (G_2 \circ G_3)$	Associativity
$G \otimes id_\epsilon \equiv G \equiv id_\epsilon \otimes G$	Monoid Identity
$(G_1 \otimes G_2) \otimes G_3 \equiv G_1 \otimes (G_2 \otimes G_3)$	Monoid Associativity
$id_I \otimes id_J \equiv id_{I \otimes J}$	Interface Identity
$(G_1 \otimes F_1) \circ (G_2 \otimes F_2) \equiv (G_1 \circ G_2) \otimes (F_1 \circ F_2)$	Bifunctoriality

Intuitively, terms represent typed structures with a source and a target interface ($G : I \to J$). Structures can be placed one near to the other (horizontal composition) or one inside the other (vertical composition). Each Ω in Θ has a fixed type $type(\Omega) = I \to J$. For each interface I, we assume a distinguished construct $id_I : I \to I$. The types of constructors, together with the rules in Tab. 3.2, determine the type of each term. Terms of type $\epsilon \to J$ are called *ground*.

The term obtained by tensor is well typed when both corresponding tensors on source and target interface are defined, namely they are separated structures. On the other hand, composition is defined only when the two involved terms *share* a common interface. In the following, we consider only well typed terms.

Terms are defined up to the structural congruence $\equiv$ described in Tab. 3.3. It subsumes the axioms of the monoidal categories. All axioms are required to hold whenever both sides are well typed. Throughout the paper, when using $=$ or $\equiv$ we imply that both sides are defined; and when we need to remark that a bigraphical expression E is well given, we write $(E)\downarrow$. Later on, the congruence will be refined to model specialised structures, such as place graphs, link graphs or bigraphs.

The axioms correspond to those for (partial) monoidal categories. In particular we constrain the structural congruence to satisfy the bifunctoriality property between product and composition. Thus, we can interpret our terms as arrows of the free monoidal category on $(M, \otimes, \epsilon)$ generated by Θ. In this case the term congruence corresponds to the equality of the corresponding arrows.

Example 1 An intuitive example of bifunctorial terms is provided by located resources. Every location is represented by a *cell*; every cell can contain a resource. Horizontal composition represents the merging of cells, and vertical composition combines the resources included in the cells. This model will provide a semantics to the logical operators we are defining, and will show that BiLog, although inspired by bigraphs, is not only connected to the bigraphical framework (cf. Ex. 2).

The set of resources is a monoidal structure $(M, \lambda, \cdot)$ freely generated by a set Λ of resource generators. The resource monoid may possibly be partial. In this case, the monoid of interfaces is the commutative monoid of ordinals $(\mathbf{N}, 0, +)$, freely generated by $\{1\}$. We define the constructor $\boxed{\lambda} : 1 \to 1$ for the neutral element λ and a constructor $\boxed{a} : 1 \to 1$ for each element $a \in \Lambda$. Every element $\boxed{}$ represents a cell, the constructor $\boxed{a}$ represents a cell containing the resource generator a. Table 3.4 outlines the two composition operators. The vertical composition $\circ$ between two cells $\boxed{a_1}$ and $\boxed{a_2}$ corresponds to combine – when possible – the two generators contained in the cells, thus producing the cell $\boxed{a_1 \cdot a_2}$ containing the resource $a_1 \cdot a_2$. This operation produces a cell $\boxed{m}$ for every resource $m \in M$. The horizontal composition $\otimes$ consists of aligning two cells, thus producing lists of cells.

The terms generated by these settings are *resources vectors* Their inner and outer faces correspond to their size. The horizontal composition $\otimes$ is in general the juxtaposition of vectors. Given the vectors $\begin{array}{|c|c|c|}\hline m_1 & ... & m_n \\ \hline\end{array} : n \to n$, of size n, and $\begin{array}{|c|c|c|}\hline m'_1 & ... & m'_{n'} \\ \hline\end{array} : n' \to n'$, of size n', the composition $\otimes$ is formally defined as

$$\begin{array}{|c|c|c|}\hline m_1 & ... & m_n \\ \hline\end{array} \otimes \begin{array}{|c|c|c|}\hline m'_1 & ... & m'_{n'} \\ \hline\end{array} \quad \overset{def}{=} \quad \begin{array}{|c|c|c|c|c|c|}\hline m_1 & ... & m_n & m'_1 & ... & m'_{n'} \\ \hline\end{array} .$$

The resulting vector is typed by $(n + n') \to (n + n')$, and has size $n + n'$.

The vertical composition $\circ$ is defined only between vectors with equal size, and corresponds to combine the resources cell by cell, as follows:

$$\begin{array}{|c|c|c|}\hline m_1 & ... & m_n \\ \hline\end{array} \circ \begin{array}{|c|c|c|}\hline m'_1 & ... & m'_n \\ \hline\end{array} \quad \overset{def}{=} \quad \begin{array}{|c|c|c|}\hline m_1 \cdot m'_1 & ... & m_n \cdot m'_n \\ \hline\end{array} .$$

The two operations satisfy the bifunctiorial property, which represents here the possibility to chose either to concatenate the vectors first and then to combine the resources, or vice versa. For cells, the bifunctorial property says

$$\left(\boxed{m_1} \otimes \boxed{m_2} \right) \circ \left(\boxed{m_3} \otimes \boxed{m_4} \right) = \left(\boxed{m_1} \circ \boxed{m_3} \right) \otimes \left(\boxed{m_2} \circ \boxed{m_4} \right) .$$

The two terms above correspond to $\begin{array}{|c|c|}\hline m_1 \cdot m_3 & m_2 \cdot m_4 \\ \hline\end{array}$. The bifunctorial provides two possible normal forms: *(i)* the *horizontal outermost* $\left(\boxed{a_1} \circ \ldots \circ \boxed{a_n} \right) \otimes \ldots \otimes \left(\boxed{a^m_1} \circ \ldots \circ \boxed{a^m_{n^m}} \right)$, with $a^j_i \in \Lambda$, that first combines by $\circ$ and then by $\otimes$;

Table 3.4. *Cell Compositions*

$$\left.\begin{array}{ccccc} \boxed{a_1} & \otimes & \dots & \otimes & \boxed{a_n} \\ \circ & & & & \circ \\ \boxed{a'_1} & \otimes & \dots & \otimes & \boxed{a'_n} \end{array}\right\} \quad \boxed{a_1 \cdot a'_1}\boxed{\;\dots\;}\boxed{a_n \cdot a'_n}$$

and *(ii)* the *vertical outermost* $\left(\boxed{a_1} \otimes \dots \otimes \boxed{a_n}\right) \circ \dots \circ \left(\boxed{a^m_1} \otimes \dots \otimes \boxed{a^m_{n^m}}\right)$, where $a^j_i \in \Lambda \cup \{\lambda\}$ and $a^j_i = \lambda$ implies $a^{j+1}_i = \lambda$, that first combines by $\otimes$ and then by $\circ$. The congruence on resource vectors is represented by the equality on normal forms, and it satisfies all the axioms of Tab. 3.3. In Particular, id_0 represents the empty resource vector, id_1 corresponds to $\boxed{\lambda}$, and in general id_n is $\boxed{\lambda}\boxed{\dots}\boxed{\lambda} : n \to n$.

The properties of these particular terms depend strictly on the choice of the underlying resource monoid, which can be either non-commutative (whenever considering sequences of resources, or ordered trees), or commutative (whenever considering multisets of resources, or unordered trees), or partial (whenever dealing with heaps). This example is rather limited, in the sense that inner and outer faces are forced to be equals, an there are only two kinds of constructors. The full generality will be reached with bigraphs. The aim of this model is to hint that BiLog can characterise models not directly based on bigraphs, as Ex. 2 will show.

3.2 Transparency

In general not every structure of the model corresponds to an observable structure in a spatial logic. A classical example is ambient logic. Some mobile ambient constructors have their logical equivalent, e.g. ambients **a**[−] , and other ones are not directly mapped in the logic, e.g. the **in** and **out** capabilities. In this case the observability of the structure is distinguished from the observability of the computational terms: some terms are used to express behaviour and other to express structure. Moreover there are terms representing both notions since ambients can be opened.

The structure may be used not only to represent the distribution or the shape of resources but also to encode their behaviour. We may want to avoid a direct representation of some structures at logical level of BiLog. A natural solution is to define a notion of *transparency* over the structure. In such a way, entities representing the structure are *transparent*, while entities encoding behaviour are *opaque* and cannot be distinguished by the logical spatial connectives. Transparent terms allow the logic to see their entire structure while opaque terms block the inspection at some opacity point. A notion of transparency can also appear in models without temporal behaviour. In fact, consider a model with a variable access control policy determined by some structural characteristics. Thus, some terms may be either transparent or opaque, depending on the current policy, and the visibility in the logic, or in the query language, will be influenced by this.

When the model is dynamic, the reacting contexts, namely those with a possible temporal evolution, are specified with an activeness predicate. We may be tempted to identify transparency and activeness. Although these concepts collapse in some case, they are orthogonal in general. There may be transparent terms that are active, such as a public 'browse-able' directory; opaque terms that are active, such as an agent that hides its contents; passive transparent terms, such as a portable code; and passive opaque terms, such as controls encoding synchronisation.

More generally the transparency predicate prevents logical identification of terms. As an example, consider an XML document. We may want to restrict our attention to a particular set of nodes; we could, e.g., ignore data values when interested in the structure. In other situations, we may want a different logic focused on values, but not on node attributes.

Transparency is essentially a way to restrict the observational power of the structural logic Notice that in general such a restriction of the observational power in the static logic does not imply a restriction of observational power in the dynamic counterpart. In fact, a next step modality may induce a 'new' intensionalisation of the controls by observing how the model evolves, as shown in [5] and [36].

3.3 Formulae

BiLog internalises the constructors of bifunctorial terms in the style of the ambient logic [13]. Constructors appear in the logic as constant formulae, while tensor product and composition are expressed by connectives. Thus the logic presents two binary spatial operators. This contrasts with other spatial logics, with a single one: Spatial and Ambient Logics [4,13], with parallel composition $A \mid B$, Separation Logic [34], with separating conjunction $A * B$, and Context Tree Logic [7], with application $K(P)$. Both the operators inherit the monoidal structure and non-commutativity properties from the model.

The logic is parameterised by the transparency predicate $\tau(\,)$: as explained in the previous section, opaque terms do not allow inspection of their contents. We say that a term G is transparent, or observable, if $\tau(G)$ is verified. We will see that when all terms are observable the logical equivalence corresponds to $\equiv$. We assume that id_I and ground terms are always transparent, and τ preserves $\equiv$, hence $\otimes$ and $\circ$.

Given the monoid $(M, \otimes, \epsilon)$, the set of simple terms Θ, the transparency predicate τ and the structural congruence relation $\equiv$, the logic BiLog$(M, \otimes, \epsilon, \Theta, \equiv, \tau)$ is formally defined in Tab. 3.5. The satisfaction relation $\models$ gives the semantics. The logic features a constant $\mathbf{\Omega}$ for each transparent construct Ω. In particular it has the iden-

Table 3.5. *BiLog($M, \otimes, \epsilon, \Theta, \equiv, \tau$)*

$\mathbf{\Omega} ::= \mathbf{id}_I \mid \ldots$	a constant formula for every Ω s.t. $\tau(\Omega)$		
$A, B ::= \mathbf{F}$	false	$A \Rightarrow B$	implication
$\mathbf{id}$	identity	$\mathbf{\Omega}$	constant constructor
$A \otimes B$	tensor product	$A \circ B$	composition
$A \circ\!- B$	left comp. adjunct	$A -\!\circ B$	right comp. adjunct
$A \otimes\!- B$	left prod. adjunct	$A -\!\otimes B$	right prod. adjunct

$G \models \mathbf{F}$	iff	never
$G \models A \Rightarrow B$	iff	$G \models A$ implies $G \models B$
$G \models \mathbf{\Omega}$	iff	$G \equiv \Omega$
$G \models \mathbf{id}$	iff	exists I s.t. $G \equiv id_I$
$G \models A \otimes B$	iff	exists G_1, G_2 s.t. $G \equiv G_1 \otimes G_2$, with $G_1 \models A$ and $G_2 \models B$
$G \models A \circ B$	iff	exists G_1, G_2. s.t. $G \equiv G_1 \circ G_2$, with $\tau(G_1)$ and $G_1 \models A$ and $G_2 \models B$
$G \models A \circ\!- B$	iff	for all G', the fact that $G' \models A$ and $\tau(G')$ and $(G' \circ G)\!\downarrow$ implies $G' \circ G \models B$
$G \models A -\!\circ B$	iff	$\tau(G)$ implies that for all G', if $G' \models A$ and $(G \circ G')\!\downarrow$ then $G \circ G' \models B$
$G \models A \otimes\!- B$	iff	for all G', the fact that $G' \models A$ and $(G' \otimes G)\!\downarrow$ implies $G' \otimes G \models B$
$G \models A -\!\otimes B$	iff	for all G', the fact that $G' \models A$ and $(G \otimes G')\!\downarrow$ implies $G \otimes G' \models B$

tity $\mathbf{id}_I$ for each interface I. The satisfaction of logical constants is simply the congruence to the corresponding constructor. The *horizontal decomposition* formula $A \otimes B$ is satisfied by a term that can be decomposed as the tensor product of two terms satisfying A and B respectively. The degree of separation enforced by $\otimes$ between terms plays a fundamental role in the various instances of the logic, notably link graph and place graph. The *vertical decomposition* formula $A \circ B$ is satisfied by terms that can be the composition of terms satisfying A and B. We shall see that in some cases both connectives correspond to well known spatial ones. We define the *left* and *right adjuncts* for composition and tensor to express extensional properties. The left adjunct $A \circ\!- B$ expresses the property of a term to satisfy B whenever inserted in a context satisfying A. Similarly, the right adjunct $A -\!\circ B$ expresses the property of a context to satisfy B whenever filled with a term satisfying A. A similar description holds for $\otimes\!-$ and $-\!\otimes$, the adjoints of $\otimes$. Clearly these adjoints collapse whenever the tensor is commutative in the model.

Example 2 Consider the resource vectors defined in Ex. 1. When a BiLog formula is interpreted in that context, it represents a class of resource vectors. For sake of simplicity, we assume that all this terms are transparent. Thus, when instantiated on these terms, BiLog provides a formula $\boxed{\mathbf{a}}$ for each constructor $\boxed{a}$. The semantics of $\boxed{\mathbf{a}}$ represents the class of all the terms whose normal form is the constructor $\boxed{a}$. For instance, $\boxed{a \cdot \lambda} \otimes id_0 \models \boxed{\mathbf{a}}$. The formula $A \otimes B$ means that a resource vector can be horizontally divided into two resource vectors satisfying A and B respectively. For instance the formula $\boxed{\boldsymbol{a}} \otimes \mathbf{T}$ is satisfied by all the resource vectors having $\boxed{a}$ as first cell. On the other hand, the formula $\boxed{\boldsymbol{a}} \circ \mathbf{T}$ implicitly says that a resource vector is composed by a single cell containing a resource whose generators include a. In addition, if the resource monoid is not commutative, the previous formula says that the first element in the composition is actually a. The formula $\mathbf{T} \otimes A \otimes \mathbf{T}$ characterises resources vectors with a subvector satisfying A. In particular $\mathbf{T} \otimes (A \circ \mathbf{id}_1) \otimes \mathbf{T}$ means that one of the cells in the vector satisfied A. Finally, if we use $\mathbf{T} \otimes (\mathbf{T} \circ \boxed{a} \circ \mathbf{T}) \otimes \mathbf{T}$ says that the resource a appears somewhere in the resource vector. More generally the formula $\mathbf{id}_1 \circ \mathbf{T}$ means that the resource vector has size 1, then it is a simple sequence.

The formula $\boldsymbol{Cell} \stackrel{def}{=} \mathbf{id}_1 \circ (\neg\mathbf{id}_1 \wedge (\neg(\neg\mathbf{id}_1 \circ \neg\mathbf{id}_1))$ states that a resource vector is not empty and it is not composed by two not empty vectors, then it is a single cell. The *Cell* formula is useful to define two operators that correspond to the Kleene stars for the bigraphical combinators. Let $\boxed{\boldsymbol{a}}^{\otimes *} \stackrel{def}{=} \neg\Big(\mathbf{T} \otimes \Big(\boldsymbol{Cell} \wedge \neg\boxed{\boldsymbol{a}}\Big) \otimes \mathbf{T}\Big)$. This formula is satisfied by resource vectors that are not composed by cells different from $\boxed{a}$. Thus $\boxed{\boldsymbol{a}}^{\otimes *}$ characterises resource vectors of the kind $\boxed{a} \otimes \ldots \otimes \boxed{a}$, namely elements of the Kleene star generated by $\boxed{a}$ and the composition $\otimes$. This idea can be extended to a formula A:

$$A^{\otimes *} \stackrel{def}{=} \neg\,(\mathbf{T} \otimes (\boldsymbol{Cell} \wedge \neg A) \otimes \mathbf{T})\,;$$

$$A^{\circ *} \stackrel{def}{=} \neg\,(\mathbf{T} \otimes (\boldsymbol{Cell} \wedge \neg A) \otimes \mathbf{T})\,.$$

A vector of resources satisfies $A^{\otimes *}$ if it is composed only by cells satisfying A.

3.4 Properties

Here we show some basic results about BiLog. In particular, we observe that, in presence of trivial transparency, the induced logical equivalence coincides with the structural congruence of the terms. Such a property is fundamental to describe, query and reason about bigraphical data structures, as e.g. XML (cf. §6). In other terms, BiLog is *intensional* in the sense of [36], namely it can observe internal structures, as opposed to the extensional logics used to observe the behaviour of dynamic system. Inspired by [24], it would be possible to study a fragment of BiLog without the intensional operators $\otimes$, $\circ$, and constants.

The lemma below states that the relation $\models$ respects the congruence.

Lemma 1 (Congruence preservation) *For every couple of terms G and G', if $G \models A$ and $G \equiv G'$ then $G' \models A$.*

Proof. Induction on the structure of the formula, by recalling that the congruence is required to preserve the typing and the transparency. In detail

Case F. Nothing to prove.

Case $\mathbf{\Omega}$. By hypothesis $G \models \mathbf{\Omega}$ and $G \equiv G'$. By definition $G \equiv \Omega$ and by transitivity $G' \equiv \Omega$, thus $G' \models \mathbf{\Omega}$.

Case id. By hypothesis $G \models \mathbf{id}$ and $G \equiv G'$. Hence there exists an I such that $G' \equiv G \equiv id_I$ and so $G' \models \mathbf{id}$.

Case $A \Rightarrow B$**.** By hypothesis $G \models A \Rightarrow B$ and $G \equiv G'$. This means that if $G \models A$ then $G \models B$. By induction if $G' \models A$ then $G \models A$. Thus if $G' \models A$ then $G \models B$ and again by induction $G' \models B$.

Case $A \otimes B$**.** By hypothesis $G \models A \otimes B$ and $G \equiv G'$. Thus there exist G_1, G_2 such that $G' \equiv G \equiv G_1 \otimes G_2$ and $G_1 \models A$ and $G_2 \models B$. Hence $G' \models A \otimes B$.

Case $A \circ B$**.** By hypothesis $G \models A \circ B$ and $G \equiv G'$. Thus there exist G_1, G_2 such that $G' \equiv G \equiv G_1 \circ G_2$ and $\tau(G_1)$ and $G_1 \models A$ and $G_2 \models B$. Hence $G' \models A \circ B$.

Case $A \circ\!\!- B$**.** By hypothesis $G \models A \circ\!\!- B$ and $G \equiv G'$. Thus for every G'' such that $G'' \models A$ and $\tau(G'')$ and $(G'' \circ G)\!\downarrow$ it holds $G'' \circ G \models B$. Now $G \equiv G'$ implies $G'' \circ G \equiv G'' \circ G'$; moreover the congruence preserves typing, so $(G'' \circ G')\!\downarrow$. By induction $G'' \circ G' \models B$, then conclude $G' \models A \circ\!\!- B$.

Case $A -\!\!\circ B$**.** If $\tau(G')$ is not verified, then $G' \models A -\!\!\circ B$ trivially holds. Suppose $\tau(G')$ to be verified. As $G \equiv G'$ and transparency preserves congruence, $\tau(G)$ is verified as well. By hypothesis for each G'' satisfying A such that $(G \circ G'')\!\downarrow$ it holds $G \circ G'' \models B$, and by induction $G' \circ G'' \models B$, as $G \equiv G'$ and $(G \circ G'')\!\downarrow$ implies $(G' \circ G'')\!\downarrow$ and $G \circ G'' \equiv G' \circ G''$. This proves $G' \models A -\!\!\circ B$.

Case $A \otimes\!\!- B$ (and symmetrically $A -\!\!\otimes B$)**.** By hypothesis $G \models A \otimes\!\!- B$ and $G \equiv G'$. Thus for each G'' such that $G'' \models A$ and $(G'' \otimes G)\!\downarrow$ then $G'' \otimes G \models B$. Now $G \equiv G'$ implies $G'' \otimes G \equiv G'' \otimes G'$, again the congruence must preserve typing so $(G'' \otimes G')\!\downarrow$. Thus by induction $G'' \otimes G' \models B$. The generality of G'' implies $G' \models A \otimes\!\!- B$. □

BiLog induces a logical equivalence $=_L$ on terms in the usual sense. We say that $G_1 =_L G_2$ if for every formula A, $G_1 \models A$ implies $G_2 \models A$ and vice versa. It is easy to prove that the logical equivalence corresponds to the congruence in the model if the transparency predicate is true for every term.

Theorem 1 (Logical equivalence and congruence) *When the transparency predicate is always true, then $G =_L G'$ if and only if $G \equiv G'$ for every term G, G'.*

Proof. The forward direction is proved by defining the characteristic formula for terms, as every term can be expressed as a formula. In fact, the transparency predicate is total, hence every constant term corresponds to a constant formula. The converse is a direct consequence of Lemma 1. □

The logical equivalence is less discriminating in presence of opaque constructors. For instance, the logic cannot distinguish two opaque constructors of equal type.

The particular characterisation of the logical equivalence as the congruence in the case of trivial transparency can be generalised to a congruence 'up-to-transparency.'

Table 4.1. *Derived Operators*

$\mathbf{T}, \wedge, \vee, \Leftrightarrow, \Leftarrow, \neg$			Classical operators
A_I	$\overset{def}{=}$	$A \circ \mathbf{id}_I$	Constraining the source to be I
$A_{\to J}$	$\overset{def}{=}$	$\mathbf{id}_J \circ A$	Constraining the target to be J
$A_{I \to J}$	$\overset{def}{=}$	$(A_I)_{\to J}$	Constraining the type to be $I \to J$
$A \circ_I B$	$\overset{def}{=}$	$A \circ \mathbf{id}_I \circ B$	Composition with interface I
$A \multimapinv_J B$	$\overset{def}{=}$	$A_{\to J} \multimapinv B$	Contexts with J as target guarantee
$A \multimap_I B$	$\overset{def}{=}$	$A_I \multimap B$	Composing with terms having I as source
$A \ominus B$	$\overset{def}{=}$	$\neg(\neg A \otimes \neg B)$	Dual of tensor product
$A \bullet B$	$\overset{def}{=}$	$\neg(\neg A \circ \neg B)$	Dual of composition
$A \bullet\!\!- B$	$\overset{def}{=}$	$\neg(\neg A \multimapinv \neg B)$	Dual of composition left adjunct
$A -\!\!\bullet B$	$\overset{def}{=}$	$\neg(\neg A \multimap \neg B)$	Dual of composition right adjunct
$A^{\exists\otimes}$	$\overset{def}{=}$	$\mathbf{T} \otimes A \otimes \mathbf{T}$	Some horizontal term satisfies A
$A^{\forall\otimes}$	$\overset{def}{=}$	$\mathbf{F} \ominus A \ominus \mathbf{F}$	Every horizontal term satisfies A
$A^{\exists\circ}$	$\overset{def}{=}$	$\mathbf{T} \circ A \circ \mathbf{T}$	Some vertical term satisfies A
$A^{\forall\circ}$	$\overset{def}{=}$	$\mathbf{F} \bullet A \bullet \mathbf{F}$	Every vertical term satisfies A
$\diamond A$	$\overset{def}{=}$	$(\mathbf{T} \circ A)_\epsilon$	Somewhere modality (on ground terms)
$\square A$	$\overset{def}{=}$	$\neg\diamond\neg A$	Anywhere modality (on ground terms)

That means we can find an equivalence relation between trees that is 'tuned' by τ: the more τ covers, the less the equivalence distinguishes. This relation will be better understood when we instantiate the logic to particular terms. A possible definition of transparency will be provided in §5.6.

4 BiLog: derived operators

Table 4.1 outlines several operators that can be derived in BiLog. The classical operators and those constraining the interfaces are self-explanatory. The 'dual' operators are worth explaining. The formula $A \ominus B$ is satisfied by terms G such that for every possible decomposition $G \equiv G_1 \otimes G_2$ either $G_1 \models A$ or $G_2 \models B$. For instance, $A \ominus A$ describes terms where A is true in, at least, one part of each $\otimes$-decomposition. The formula $\mathbf{F} \ominus (\mathbf{T}_{\to I} \Rightarrow A) \ominus \mathbf{F}$ describes those terms where every component with outerface I satisfies A. Similarly, the composition $A \bullet B$ expresses structural properties universally quantified on every $\circ$-decomposition. Both these connectives are useful to specify security properties or types.

The adjunct dual $A \bullet\!\!- B$ describes terms that can be inserted into a particular context satisfying A to obtain a term satisfying B, it is a sort of existential quantification on contexts. For instance $(\mathbf{\Omega}_1 \vee \mathbf{\Omega}_2) \bullet\!\!- A$ describes the union between the class of two-region bigraphs (with no names in the outerface) whose merging satisfies A, and terms that can be inserted either in Ω_1 or Ω_2 resulting in a term satisfying A. Similarly the dual adjunct $A -\!\!\bullet B$ describes contextual terms G such that there exists a term satisfying A that inserted in G gives a term satisfying B.

The formulae $A^{\exists\otimes}$, $A^{\forall\otimes}$, $A^{\exists\circ}$, and $A^{\forall\circ}$ correspond to quantifications on the horizontal/vertical structure of terms. For instance $\mathbf{\Omega}^{\forall\circ}$ describes terms that are a finite (possibly empty) composition of simple terms Ω. Next section discusses spatial modalities ✧ and ⌑.

Following lemma states a first property involving the derived connectives, by proving that the interfaces for transparent terms can be observed.

Lemma 2 (Type observation) *For every term G, it holds: $G \models A_{I\to J}$ if and only if $G : I \to J$ and $G \models A$ and $\tau(G)$.*

Proof. For the forward direction, assume that $G \models A_{I\to J}$, then $G \equiv id_J \circ G' \circ id_I$ with $G' \models A$ and $\tau(G')$. Now, $id_J \circ G' \circ id_I : I \to J$. By Lemma 1: $G : I \to J$ and $G \models A$ and $\tau(G)$. The converse is a direct consequence of the semantics definition. $\square$

Thanks to the derived operators involving interfaces, the equality between interfaces, $I = J$, is derivable by $\otimes$ and $\otimes\!-$, as

$$\mathbf{T} \otimes (id_\epsilon \wedge (id_I \otimes\!- id_J)). \tag{1}$$

Whenever a bigraph satisfies such a formula, the interfaces I and J are equal. To gather the basic idea, assume the bigraph G satisfies (1). This means that $G \equiv G_1 \otimes G_2$ with $G_1 \models \mathbf{T}$ and $G_2 \models id_\epsilon \wedge (id_I \otimes\!- id_J)$. By definition, the latter is equivalent to $G_2 \equiv \epsilon$ and $G_2 \models id_I \otimes\!- id_J$. Then $G \equiv G_1$ and $\epsilon \models id_I \otimes\!- id_J$, by Lemma 1. Hence $\epsilon \otimes id_I \models id_J$, that entails $id_I \equiv id_J$. Clearly, the last equality holds only if $I = J$. By reversing the reasoning, it is easy to see that whenever $I = J$, every bigraph satisfies (1).

4.1 Somewhere modality

The idea of *sublocation*, $\sqsubseteq$ defined in [14], can be extended to the bigraphical terms. A sublocation corresponds to a subterm and it is formally defined on ground terms as follows. The definition of sublocation makes sense only for ground terms, as the structure of 'open' terms (i.e., with holes) is not known a priori. Formally it is defined as follows.

Definition 1 (Sublocation) *Given two terms $G : \epsilon \to J$ and $G' : \epsilon \to J'$, term G' is defined to be a sublocation for G, and write $G' \sqsubseteq G$, inductively by:*

- *$G' \sqsubseteq G$, if $G' \equiv G$;*
- *$G' \sqsubseteq G$, if $G \equiv G_1 \otimes G_2$, with $G' \sqsubseteq G_1$ or $G' \sqsubseteq G_2$;*
- *$G' \sqsubseteq G$, if $G \equiv G_1 \circ G_2$, with $\tau(G_1)$ and $G' \sqsubseteq G_2$.*

This relation, introduce a *"somewhere"* modality in the logic. Intuitively, a term satisfies *"somewhere"*A whenever one of its sublocations satisfies A. Rephrasing the semantics given in [14], a term ground term G satisfies the formula *"somewhere"*A if and only if there exists $G' \sqsubseteq G$ such that $G' \models A$. Quite surprisingly, such a modality is expressible in the logic. In fact, in case of ground terms, the previous requirement is the semantics of the derived connective $\diamond$, defined in Tab. 4.1.

Proposition 1 *For every ground term G:*

$$G \models \diamond A \textit{ if and only if there exists } G' \sqsubseteq G \textit{ such that } G' \models A.$$

Proof. First prove a supporting property characterising the relation between a term and its sublocations.

Property 1 For every ground term G and G', it holds: $G' \sqsubseteq G$ if and only if there exists a term C such that $\tau(C)$ and $G \equiv C \circ G'$.

The direction from right to left is a simple application of Definition 1. The direction from left to right is proved by induction on Definition 1. For the *basic step*, the implication clearly holds if $G' \sqsubseteq G$ in case $G' \equiv G$. The *inductive step* distinguishes two cases.

If $G' \sqsubseteq G$ is due to the fact that $G \equiv G_1 \otimes G_2$, with $G' \sqsubseteq G_1$ or $G' \sqsubseteq G_2$. Without loss of generality, assume $G' \sqsubseteq G_1$. The induction says that there exists C such that $\tau(C)$ and $G_1 \equiv C \circ G'$. Hence, $G \equiv (C \circ G') \otimes G_2$. Now the typing is: $C : I_C \to J_C$; $G' : \epsilon \to I_C$; $G_2 : \epsilon \to J_2$; and $G : \epsilon \otimes \epsilon \to J_C \otimes J_2$. So $G \equiv (C \circ G') \otimes (G_2 \circ id_\epsilon)$. As the interface ϵ is the neutral element for the tensor product between interfaces, compose $C \otimes G_2 : I_C \otimes \epsilon \to J_C \otimes J_2$, and $G' \otimes id_\epsilon : \epsilon \otimes \epsilon \to I_C \otimes \epsilon$. Hence the term $(C \otimes G_2) \circ (G' \otimes id_\epsilon)$ is defined. Note that $\tau(C \otimes G_2)$ is true,as $\tau(G_2)$ is verified since $G_2 : \epsilon \to J_2$ and $\tau(C)$ is true by induction. Hence, by bifunctoriality property, conclude $G \equiv (C \otimes G_2) \circ G'$, with $\tau(C \otimes G_2)$, as aimed.

On the other hand, if $G' \sqsubseteq G$ is due to the fact that $G \equiv G_1 \circ G_2$, with $\tau(G_1)$ and $G' \sqsubseteq G_2$. The induction says that there exists C such that $\tau(C)$ and $G_2 \equiv C \circ G'$. Hence, $G \equiv G_1 \circ (C \circ G')$. Conclude $G \equiv (G_1 \circ C) \circ G'$, with $\tau(G_1 \circ C)$.

Suppose now that $G \models \diamond A$, this means that $G \models (\mathbf{T} \circ A)_\epsilon$. According to Tab. 4.5, this means that there exist C and G' such that $G' \models A$ and $\tau(C)$, and $G \equiv C \circ G'$. Finally, by Property 1, this means $G' \sqsubseteq G$ and $G' \models A$. $\square$

The *everywhere* modality ($\boxtimes$) is dual to $\diamond$. A term satisfies the formula $\boxtimes A$ if each of its sublocations satisfies A.

For every axiom of the model, the logic proves a corresponding property. In particular, the bifunctoriality property is expressed by formulae

$$(A_I \circ B_{\to I}) \otimes (A'_J \circ B'_{\to J}) \Leftrightarrow (A_I \otimes A'_J) \circ (B_{\to I} \otimes B'_{\to J})$$

valid when $(I \otimes J)\downarrow$.

In general, given two formulae A, B we say that A *yields* B, and we write $A \vdash B$, if for every term G it is the case that $G \models A$ implies $G \models B$. Moreover, we write $A \dashv\vdash B$ to say both $A \vdash B$ and $B \vdash A$.

Assume that I and J are two interfaces such that their tensor product $I \otimes J$ is defined. Then, the bifuctoriality property in the logic is expressed by

$$(A_I \circ B_{\to I}) \otimes (A'_J \circ B'_{\to J}) \dashv\vdash (A_I \otimes A'_J) \circ (B_{\to I} \otimes B'_{\to J}). \qquad (2)$$

Proposition 2 *Whenever $(I \otimes J)\downarrow$, the equation (2) holds in the logic.*

Proof. Prove separately the two way of the satisfaction. First prove $(A_I \circ B_{\to I}) \otimes (A'_J \circ B'_{\to J}) \vdash (A_I \otimes A'_J) \circ (B_{\to I} \otimes B'_{\to J})$. Assume that $G \models (A_I \circ B_{\to I}) \otimes (A'_J \circ B'_{\to J})$. This means that there exist $G' : I' \to I''$, $G'' : J' \to J''$ such that $I' \otimes J'$ and $I'' \otimes J''$ are defined, and $G \equiv G' \otimes G''$, with $G' \models A_I \circ B_{\to I}$ and $G'' \models A'_J \circ B'_{\to J}$. Now, $G' \models A_I \circ B_{\to I}$ means that there exist G_1 and G_2 such that *(i)* $G' \equiv G_1 \circ G_2$, *(ii)* $G_1 : I \to J'$, with $\tau(G_1)$ and $G_1 \models A$, and *(iii)* $G_2 : I' \to I$, with $G_2 \models B$. Similarly, $G'' \models A'_J \circ B'_{\to J}$ means *(i)* $G'' \equiv G'_1 \circ G'_2$ and *(ii)* $G'_1 : J \to J''$, with $\tau(G'_1)$ and $G'_1 \models A'$, and *(iii)* $G'_2 : I'' \to J$, with $G_2 \models B'$. In particular, conclude $G \equiv (G_1 \circ G_2) \otimes (G'_1 \circ G'_2)$. As $I \otimes J$ is defined, $(G_1 \otimes G'_1) \circ (G_2 \otimes G'_2)$ is an admissible composition. The bifunctoriality property implies $G \equiv (G_1 \otimes G'_1) \circ (G_2 \otimes G'_2)$. Moreover $\tau(G_1 \otimes G'_1)$, as $\tau(G_1)$ and $\tau(G'_1)$. Hence conclude that $G \models (A_I \otimes A'_J) \circ (B_{\to I} \otimes B'_{\to J})$, as required.

For the converse, prove $(A_I \otimes A'_J) \circ (B_{\to I} \otimes B'_{\to J}) \vdash (A_I \circ B_{\to I}) \otimes (A'_J \circ B'_{\to J})$. Assume that $G \models (A_I \otimes A'_J) \circ (B_{\to I} \otimes B'_{\to J})$. By following the same lines as before, deduce that $G \equiv (G_1 \otimes G'_1) \circ (G_2 \otimes G'_2)$, where *(i)* $\tau(G_1 \otimes G'_1)$, *(ii)* $G_1 : I \to J'$ such that $G_1 \models A$, *(iii)* $G'_1 : J \to J''$ such that $G'_1 \models A'$, *(iv)* $G_2 : I' \to I$ such that $G_2 \models B$, and *(v)* $G'_2 : I'' \to J$ such that $G_2 \models B'$. Also in this case, the tensor product of the required interfaces can be performed. Hence compose $(G_1 \circ G_2) \otimes (G'_1 \circ G'_2)$. Again, the bifunctoriality property implies $G \equiv (G_1 \circ G_2) \otimes (G'_1 \circ G'_2)$. Finally, by observing that $\tau(G_1 \otimes G'_1)$ implies $\tau(G_1)$ and $\tau(G'_1)$, deduce $G_1 \circ G_2 \models (A_I \circ B_{\to I})$ and $(G'_1 \circ G'_2) \models (A'_J \circ B'_{\to J})$. Then conclude $G \models (A_I \circ B_{\to I}) \otimes (A'_J \circ B'_{\to J})$. $\square$

Table 5.1. *Additional Axioms for Place Graphs Structural Congruence*

Symmetric Category Axioms:	
$\gamma_{m,0} \equiv id_m$	Symmetry Id
$\gamma_{m,n} \circ \gamma_{n,m} \equiv id_{m \otimes n}$	Symmetry Composition
$\gamma_{m',n'} \circ (G \otimes F) \equiv (F \otimes G) \circ \gamma_{m,n}$	Symmetry Monoid
Place Axioms:	
$join \circ (1 \otimes id_1) \equiv id_1$	Unit
$join \circ (join \otimes id_1) \equiv join \circ (id_1 \otimes join)$	Associativity
$join \circ \gamma_{1,1} \equiv join$	Commutativity

5 BiLog: instances and encodings

In this section BiLog is instantiated to describe place graphs, link graphs and bigraphs. A spatial logic for bigraphs is a natural composition of a place graph logic, for tree contexts, and a link graph logic, for name linkings. Each instance admits an embedding of a well known spatial logic.

5.1 Place Graph Logic

Place graphs are essentially ordered lists of regions hosting unordered labelled trees with holes, namely contexts for trees. Tree labels correspond to controls $\mathsf{K} : 1 \to 1$ belonging to a fixed signature $\mathcal{K}$. The monoid of interfaces is the monoid $(\omega, +, 0)$ of finite ordinals m, n. Ordinals represent the number of holes and regions of place graphs. Place graph terms are generated from the set

$$\Theta = \{1 : 0 \to 1, id_n : n \to n, join : 2 \to 1, \gamma_{m,n} : m + n \to n + m\} \cup \mathcal{K}$$

The only structured terms are the controls K, representing regions containing a single node with a hole inside. All the other constructors are *placings* and represent trees $m \to n$ with no nodes: the place identity id_n is neutral for composition; the constructor 1 represents a barren region; *join* is a mapping of two regions into one; $\gamma_{m,n}$ is a permutation that interchanges the first m regions with the following n. The structural congruence $\equiv$ for place graph terms is refined, in Tab. 5.1, by the usual axioms for symmetry of $\gamma_{m,n}$ and by the place axioms that essentially turn the operation $join \circ (\_ \otimes \_)$ in a commutative monoid with 1 as neutral element. In particular, the places generated by composition and tensor product from $\gamma_{m,n}$ are *permutations*. A place graph is *prime* if it has type $I \to 1$, namely it has a single region.

Example 3 The term

$$G \stackrel{def}{=} (service \circ (join \circ (name \otimes description))) \otimes (push \circ 1)$$

is a place graph of type $2 \to 2$, on a signature containing *service*, *name*, *description*, and *push*. It represents an ordered pair of trees. The first tree is labelled *service* and has *name* and *description* as (unordered) children, both children are actually contexts with a single hole. The second tree is ground as it has a single node without children. The term G is congruent to $(\mathit{service} \otimes \mathit{push}) \circ (\mathit{join} \otimes 1) \circ (\mathit{description} \otimes \mathit{name})$. Such a contextual pair of trees can be interpreted as semi-structured partial data (e.g. an XML message, a web service descriptor) that can be filled by composition. The order among holes is a major issue in the composition, for instance, $(\mathsf{K}_1 \otimes \mathsf{K}_2) \circ (\mathsf{K}_3 \otimes 1)$ is different from $(\mathsf{K}_1 \otimes \mathsf{K}_2) \circ (1 \otimes \mathsf{K}_3)$, as node K_3 plugs into K_1 in the first case, and inside K_2 in the second one.

Fixed the transparency predicate τ on each control in $\mathcal{K}$, the Place Graph Logic $\mathrm{PGL}(\mathcal{K}, \tau)$ is $\mathit{BiLog}(\omega, +, 0, \equiv, \mathcal{K} \cup \{1, \mathit{join}, \gamma_{m,n}\}, \tau)$. We assume the transparency predicate τ to hold for *join* and $\gamma_{m,n}$. Theorem 1 can be extended to PGL, thus such a logic can describe place graphs precisely. The logic resembles a propositional spatial tree logic, in the style of [6]. The main differences are that PGL models contexts of trees and that the tensor product is not commutative, unlike the parallel composition in [6], and it enables the modelling of the order among regions. The logic can express a commutative separation by using **join** and the tensor product, namely the *parallel composition* operator $A \mid B \overset{def}{=} \mathbf{join} \circ (A_{\to 1} \otimes B_{\to 1})$. At the term level, this separation, which is purely structural, corresponds to $\mathit{join} \circ (P_1 \otimes P_2)$, that is a total operation on all prime place graphs. More precisely, the semantics says that $P \models A \mid B$ means that there exist $P_1 : I_1 \to 1$ and $P_2 : I_2 \to 1$ such that: $P \equiv \mathit{join} \circ (P_1 \otimes P_2)$ and $P_1 \models A$ and $P_2 \models B$.

5.2 Encoding STL

Not surprisingly, prime ground place graphs are isomorphic to the unordered trees modelling the static fragment of ambient logic. Here we show that, when the transparency predicate is always verified, BiLog restricted to prime ground place graphs is equivalent to the propositional Spatial Tree Logic of [6] (STL in the following). The logic STL expresses properties of unordered labelled trees T constructed from the empty tree 0, the labelled node containing a tree $a[T]$, and the parallel composition of trees $T_1 \mid T_2$, as detailed in Tab. 5.2. Labels a are elements of a denumerable set Λ. STL is a static fragment of the ambient logic [13] and it is characterised by the usual classical propositional connectives, the spatial connectives 0, $a[A]$, $A \mid B$, and their adjuncts $A@a$, $A \triangleright B$. The language of the logic and its semantics is outlined in Tab. 5.3.

Table 5.4 encodes the tree model of STL into prime ground place graphs, and STL operators into PGL operators. We assume a bijective encoding between labels and controls, and we associate every label a with a distinct control $\mathsf{K}(a)$ of arity 0. As already said, we assume the transparency predicate to be verified on every control.

Table 5.2. *Information tree Terms (over Λ) and congruence*

$T, T' ::=$	0	empty tree consisting of a single root node
	$a[T]$	single edge tree labelled $l \in \Lambda$ leading to the subtree T
	$T \mid T'$	tree obtained by merging the roots of the trees T and T'
$T \mid 0 \equiv T$		neutral element
$T \mid T' \equiv T' \mid T$		commutativity
$(T \mid T') \mid T'' \equiv T \mid (T' \mid T'')$		associativity

Table 5.3. *Propositional Spatial Tree Logic*

$A, B ::= \mathbf{F}$	anything	$a[A]$	location
$\mathbf{0}$	empty tree	$A@a$	location adjunct
$A \Rightarrow B$	implication	$A \mid B$	composition
		$A \triangleright B$	composition adjunct

$T \models_{\text{STL}} \mathbf{F}$	iff	never
$T \models_{\text{STL}} \mathbf{0}$	iff	$F \equiv 0$
$T \models_{\text{STL}} A \Rightarrow B$	iff	$T \models_{\text{STL}} A$ implies $T \models_{\text{STL}} B$
$T \models_{\text{STL}} a[A]$	iff	there exists T' s.t. $T \equiv a[F']$ and $T' \models_{\text{STL}} A$
$T \models_{\text{STL}} A@a$	iff	$a[T] \models_{\text{STL}} A$
$T \models_{\text{STL}} A \mid B$	iff	there exists T_1, T_2 s.t. $T \equiv T_1 \mid T_2$ and $T_1 \models_{\text{STL}} A$ and $T_2 \models_{\text{STL}} B$
$T \models_{\text{STL}} A \triangleright B$	iff	for every T': if $T' \models_{\text{STL}} A$ *implies* $T \mid T' \models_{\text{STL}} B$

Table 5.4. *Encoding STL in PGL over prime ground place graphs*

Trees into Prime Ground Place Graphs		
$[\![0]\!] \stackrel{def}{=} \mathbf{1}$	$[\![a[T]]\!] \stackrel{def}{=} \mathsf{K}(a) \circ [\![T]\!]$	$[\![T_1 \mid T_2]\!] \stackrel{def}{=} join \circ ([\![T_1]\!] \otimes [\![T_2]\!])$
STL formulae into PGL formulae		
$[\![\mathbf{0}]\!] \stackrel{def}{=} 1$		$[\![a[A]]\!] \stackrel{def}{=} \mathsf{K}(a) \circ_1 [\![A]\!]$
$[\![\mathbf{F}]\!] \stackrel{def}{=} \mathbf{F}$		$[\![A@a]\!] \stackrel{def}{=} \mathsf{K}(a) \multimap_1 [\![A]\!]$
$[\![A \Rightarrow B]\!] \stackrel{def}{=} [\![A]\!] \Rightarrow [\![B]\!]$		$[\![A \mid B]\!] \stackrel{def}{=} [\![A]\!] \mid [\![B]\!]$
$[\![A \triangleright B]\!] \stackrel{def}{=} ([\![A]\!] \mid \mathbf{id}_1) \multimap_1 [\![B]\!]$		

The monoidal properties of parallel composition are guaranteed by the symmetry and unit axioms of *join*. The equations are self-explanatory once we remark that: *(i)* the parallel composition of STL is the structural commutative separation of PGL; *(ii)* tree labels can be represented by the corresponding controls of the place graph; *(iii)* location and composition adjuncts of STL are encoded by the left composition adjunct, as they add logically expressible contexts to the tree. This encoding is actually a bijection tree to prime ground place graphs. In fact, there is an *inverse encoding* $(\!|\ \ |\!)$ for prime ground place graphs in trees defined on the normal forms of [30].

The theorem of discrete normal form in [30] implies that every ground place graph $g : 0 \to 1$ can be expressed as

$$g = join_n \circ (M_0 \otimes \ldots \otimes M_{n-1}) \qquad (3)$$

where every M_j is a molecular prime ground place graph of the form $M = \mathsf{K}(a) \circ g$, with $ar(\mathsf{K}(a)) = 0$. As an auxiliary notation, $join_n$ is inductively defined as $join_0 \overset{def}{=} 1$, and $join_{n+1} \overset{def}{=} join \circ (id_1 \otimes join_n)$. The bifunctoriality property implies

$$join_n \circ (M_0 \otimes \ldots \otimes M_{n-1}) \equiv \\ \equiv join \circ (M_0 \otimes (join \circ (M_1 \otimes (join \circ (\ldots \otimes (join \circ (M_{n-2} \otimes M_{n-1})))))))).$$

The work in [30] says that the normal form in (3) is unique, up to permutations.

For every prime ground place graph, the inverse encoding $(\!\![\;\;]\!\!)$ considers its discrete normal form and it is inductively defined as follows

$$(\!\![\, join_0 \,]\!\!) \overset{def}{=} 0$$
$$(\!\![\, \mathsf{K}(a) \circ q \,]\!\!) \overset{def}{=} a[\, (\!\![\, q \,]\!\!) \,]$$
$$(\!\![\, join_s \circ (M_0 \otimes \ldots \otimes M_{s-1}) \,]\!\!) \overset{def}{=} (\!\![\, M_0 \,]\!\!) \mid \ldots \mid (\!\![\, M_{s-1} \,]\!\!)$$

The encodings $[\![\;\;]\!]$ and $(\!\![\;\;]\!\!)$ are one the inverse of the other, hence they give a bijection from trees to prime ground place graphs, which is fundamental in the proof of the following theorem.

Theorem 2 (Encoding STL) *For each tree T and formula A of STL:*

$$T \models_{\text{STL}} A \quad \textit{if and only if} \quad [\![\, T \,]\!] \models [\![\, A \,]\!].$$

Proof. The theorem is proved by structural induction on STL formulae. The transparency predicate is not considered here, as it holds on every control. The basic step deals with the constants **F** and **0**. Case **F** follows by definition. For the case **0**, $[\![\, T \,]\!] \models [\![\, \mathbf{0} \,]\!]$ means $[\![\, T \,]\!] \models 1$, that by definition is $[\![\, T \,]\!] \equiv 1$ and so $T \equiv (\!\![\, [\![\, T \,]\!] \,]\!\!) \equiv (\!\![\, 1 \,]\!\!) \overset{def}{=} 0$, namely $T \models_{\text{STL}} \mathbf{0}$.

The inductive steps deal with connectives and modalities.

Case $A \Rightarrow B$**.** Assuming $[\![\, T \,]\!] \models [\![\, A \Rightarrow B \,]\!]$ means $[\![\, T \,]\!] \models [\![\, A \,]\!] \Rightarrow [\![\, B \,]\!]$; by definition this says that $[\![\, T \,]\!] \models [\![\, A \,]\!]$ implies $[\![\, T \,]\!] \models [\![\, B \,]\!]$. By induction hypothesis, this is equivalent to say that $T \models_{\text{STL}} A$ implies $T \models_{\text{STL}} B$, namely $T \models_{\text{STL}} A \Rightarrow B$.

Case $a[A]$**.** Assuming $[\![\, T \,]\!] \models [\![\, a[A] \,]\!]$ means $[\![\, T \,]\!] \models \mathsf{K}(a) \circ_1 ([\![\, A \,]\!])$. This amount to say that there exist $G : 1 \to 1$ and $g : 0 \to 1$ such that $[\![\, T \,]\!] \equiv G \circ g$ and $G \models \mathsf{K}(a)$ and $g \models [\![\, A \,]\!]$, that is $[\![\, T \,]\!] \equiv \mathsf{K}(a) \circ g$ with $g \models [\![\, A \,]\!]$. Since the encoding is bijective, this is equivalent to $T \equiv (\!\![\, \mathsf{K}(a) \circ g \,]\!\!) \overset{def}{=} a[(\!\![\, g \,]\!\!)]$ with $g \models [\![\, A \,]\!]$. Since $g : 0 \to 1$, the induction hypothesis says that $(\!\![\, g \,]\!\!) \models A$. Hence it is the case that $T \models_{\text{STL}} a[A]$.

Case $A@a$**.** Assuming $[\![\, T \,]\!] \models [\![\, A@a \,]\!]$ means $[\![\, T \,]\!] \models \mathsf{K}(a) \circ\!\!-_1 A$. This is equivalent to say that for every G such that $G \models \mathsf{K}(a)$, if $(G \circ [\![\, T \,]\!])\downarrow$ then $G \circ [\![\, T \,]\!] \models [\![\, A \,]\!]$. According to the definitions, this is $\mathsf{K}(a) \circ [\![\, T \,]\!] \models [\![\, A \,]\!]$, and so

$[\![a[T]]\!] \models [\![A]\!]$. By induction hypothesis, this is $a[T] \models_{\text{STL}} A$. Hence $T \models_{\text{STL}} A@a$ by definition.

Case $A \mid B$**.** Assuming that $[\![T]\!] \models [\![A \mid B]\!]$ means $[\![T]\!] \models [\![A]\!] \mid [\![B]\!]$. This is equivalent to say that $[\![T]\!] \models \mathbf{join} \circ ([\![A]\!]_{\to 1} \otimes [\![B]\!]_{\to 1})$, namely there exist $g_1, g_2 : 0 \to 1$ such that $[\![T]\!] \equiv join \circ (g_1 \otimes g_2)$ and $g_1 \models [\![A]\!]$ and $g_2 \models [\![B]\!]$. As the encoding is bijective this means that $T \equiv (\![g_1]\!) \mid (\![g_2]\!)$, and the induction hypothesis says that $(\![g_1]\!) \models A$ and $(\![g_2]\!) \models B$. By definition this is $T \models_{\text{STL}} A \mid B$.

Case $A \triangleright B$**.** Assuming that $[\![T]\!] \models [\![A \triangleright B]\!]$ means $[\![T]\!] \models \mathbf{join}([\![A]\!] \otimes \mathbf{id}_1)) \multimap_1 [\![B]\!]$, namely for every $G : 1 \to 1$ such that $G \models \mathbf{join}([\![A]\!] \otimes \mathbf{id}_1)$ it holds $G \circ [\![T]\!] \models [\![B]\!]$. Now, $G : 1 \to 1$ and $G \models \mathbf{join}([\![A]\!] \otimes \mathbf{id}_1)$ means that there exists $g : 0 \to 1$ such that $g \models [\![A]\!]$ and $G \equiv join(g \otimes id_1)$. Hence it is the case that for every $g : 0 \to 1$ such that $g \models [\![A]\!]$ it holds $join(g \otimes id_1) \circ [\![T]\!] \models [\![B]\!]$, that is $join(g \otimes [\![T]\!]) \models [\![B]\!]$ by bifunctoriality property. Since the encoding is a bijection, this is equivalent to say that for every tree T' such that $[\![T']\!] \models [\![A]\!]$ it holds $join([\![T']\!] \otimes [\![T]\!]) \models [\![B]\!]$, that is $[\![T' \mid T]\!] \models [\![B]\!]$. By induction hypothesis, for every T' such that $T' \models_{\text{STL}} A$ it holds $T' \mid T \models_{\text{STL}} B$, that is the semantics of $T \models_{\text{STL}} A \triangleright B$. $\square$

Differently from STL, PGL can also describe structures with several holes and regions. In §6 we show how PGL describes contexts of tree-shaped semistructured data. Consider, for instance, a function taking two trees and returning the tree obtained by merging their roots. Such a function is represented by the term *join*, which solely satisfies the formula **join**. Similarly, the function that takes a tree and encapsulates it inside a node *labelled* by K, is represented by the term K and captured by the formula K. Moreover, the formula $\mathbf{join} \circ (\mathsf{K} \otimes (\mathbf{T} \circ \mathbf{id}_1))$ expresses all contexts of form $2 \to 1$ that place their first argument inside a K node and their second one as a sibling of such node.

5.3 Link Graph Logic (LGL).

Fixed a denumerable set of names Λ, we consider the monoid $(\mathcal{P}_{fin}(\Lambda), \uplus, \emptyset)$, where $\mathcal{P}_{fin}(\_)$ is the finite powerset operator and $\uplus$ is the subset disjoint union. Link graphs are the structures arising from such a monoid. They can describe nominal resources, common in many areas: object identifiers, location names in memory structures, channel names, and ID attributes in XML documents. The fact that names cannot be implicitly shared does not mean that we can refer to them or link them explicitly (e.g. object references, location pointers, fusion in fusion calculi, and IDREF in XML files). Link graphs describe connections between resources performed by means of names, that are *references*.

Wiring terms are a structured way to map a set of inner names X into a set of outer names Y. They are generated by the constructors: $/a : \{a\} \to \emptyset$ and ${}^{a}/_{X} : X \to a$. The closure $/a$ hides the inner name a in the outer face. The substitution

Table 5.5. *Additional Axioms for Link Graph Structural Congruence*

Link Axioms:	
$^{a}/_{a} \equiv id_a$	Link Identity
$/a \circ {^{a}/_{b}} \equiv /b$	Closing renaming
$/a \circ a \equiv id_\epsilon$	Idle edge
$^{b}/_{(Y \uplus a)} \circ (id_Y \otimes {^{a}/_{X}}) \equiv {^{b}/_{Y \uplus X}}$	Composing substitutions
Link Node Axiom:	
$\alpha \circ \mathsf{K}_{\vec{a}} \equiv \mathsf{K}_{\alpha(\vec{a})}$	Renaming

$^{a}/_{X}$ associates all the names in the set X to the name a. We denote wirings by ω, substitutions by σ, τ, and bijective substitutions, dubbed *renamings*, by α, β. Substitution can be specialised in: $a \stackrel{def}{=} {^{a}/_{\emptyset}}$ and $a \leftarrow b \stackrel{def}{=} {^{a}/_{\{b\}}}$ and $a \Leftarrow b \stackrel{def}{=} {^{a}/_{\{a,b\}}}$. The constructor a represents the introduction of name a, the term $a \leftarrow b$ corresponds to rename b to a, and $a \Leftarrow b$ links, or fuses, a and b to name a.

Given a signature $\mathcal{K}$ of controls K with arity function $ar(\mathsf{K})$ we generate link graphs from wirings and the constructor $\mathsf{K}_{\vec{a}} : \emptyset \to \vec{a}$ with $\vec{a} = a_1, \ldots, a_k$, $\mathsf{K} \in \mathcal{K}$, and $k = ar(\mathsf{K})$. The control $\mathsf{K}_{\vec{a}}$ represents a resource of kind K with named ports $\vec{a}$. Any ports may be connected to other node ports via wiring compositions.

In this case, the structural congruence $\equiv$ is refined as outlined in Tab. 5.5 with obvious axioms for links, modelling α-conversion and extrusion of closed names. We assume the transparency predicate τ true on wiring constructors.

Fixed the transparency predicate τ for each control in $\mathcal{K}$, the Link Graph Logic LGL$(\mathcal{K}, \tau)$ is *BiLog*$(\mathcal{P}_{fin}(\Lambda), \uplus, \emptyset, \equiv, \mathcal{K} \cup \{/a, {^{a}/_{X}}\}, \tau)$. Theorem 1 extends to LGL: the logic describes the link graphs precisely. The logic expresses structural spatiality for resources and strong spatiality (separation) for names, and it can therefore be viewed as a generalisation of Separation Logic for contexts and multi-ports locations. On the other side, the logic can describe resources with local (hidden or private) names between resources, and in this sense the logic is a generalisation of Spatial Graph Logic [10]: it is sufficient to consider the edges as resources.

Moreover, if we consider identity as a constructor, it is possible to define

$$a \leftarrow b \stackrel{def}{=} (a \Leftarrow b) \circ (a \otimes id_b).$$

In LGL the formula $A \otimes B$ describes a decomposition into two *separate* link graphs, sharing neither resources, nor names, nor connections, that satisfy A and B respectively. Since it is defined only on link graphs with disjoint inner/outer sets of names, the tensor product is a kind a *spatial/separation* operator, in the sense that it separates the model into two distinct parts that cannot share names.

In this case, horizontal decomposition inherits the commutativity property from the monoidal tensor product. If we want a name a to be shared between separated resources, we need to make the sharing explicit, and the sole way to do that is through the link operation. We therefore need a way to first separate the names occurring in two wirings as to apply the tensor, and then link them back together.

As a shorthand, if $W : X \to Y$ and $W' : X' \to Y'$ with $Y \subset X'$, we write $[W']W$ for $(W' \otimes id_{X' \setminus Y}) \circ W$ and if $\vec{a} = a_1, \ldots, a_n$ and $\vec{b} = b_1, \ldots, b_n$, we write $\vec{a} \leftarrow \vec{b}$ for $a_1 \leftarrow b_1 \otimes \ldots \otimes a_n \leftarrow b_n$, similarly for $\vec{a} \leftleftarrows \vec{b}$. From the tensor product it is possible to derive a product with sharing on $\vec{a}$. Given $G : X \to Y$ and $G' : X' \to Y'$ with $X \cap X' = \emptyset$, we choose a list $\vec{b}$ (with the same length as $\vec{a}$) of fresh names. The composition with sharing $\vec{a}$ is

$$G \overset{\vec{a}}{\otimes} G' \overset{def}{=} [\vec{a} \leftleftarrows \vec{b}]([\vec{b} \leftarrow \vec{a}]G \otimes G').$$

In this case, the tensor product is well defined since all the common names $\vec{a}$ in W are renamed to fresh names, while the sharing is re-established afterwards by linking the $\vec{a}$ names with the $\vec{b}$ names.

By extending this sharing to all names we define the parallel composition $G \mid G'$ as a total operation. However, such an operator does not behave ‘well’ with respect to the composition, as shown in [30]. In addition a direct inclusion of a corresponding connective in the logic would impact the satisfaction relation by expanding the finite horizontal decompositions to the boundless possible name-sharing decompositions. (This may be the main reason why logics describing models with name closure and parallel composition are undecidable [18].) This is due to the fact that the set of names shared by a parallel composition is not known in advance, and therefore parallel composition can only be defined by using an existential quantification over the entire set of shared names.

Names can be internalised and effectively made private to a bigraph by the closure operator $/a$. The effect of composition with $/a$ is to add a new edge with no public name, and therefore to make a disappear from the outerface, and be completely hidden to the outside. Separation is still expressed by the tensor connective, which not only separates places, but also makes sure that no edge – whether visible or hidden – crosses the separating line.

As a matter of fact, without name quantification it is not possible to build formulae that explore a link, since the latter has the effect of hiding names. For this task, we employ the name variables $x_1, ..., x_n$ and the fresh name quantification $\text{И}.$ in the style of Nominal Logic [35]. The semantics is defined as

$$G \models \text{И}x_1 \ldots x_n. A \quad \textit{iff} \quad \begin{array}{l}\textit{there exist } a_1 \ldots a_n \notin \textit{fn}(G) \cup \textit{fn}(A) \\ \textit{such that } G \models A\{x_1 \ldots x_n \leftarrow a_1 \ldots a_n\},\end{array}$$

where $A\{x_1 \ldots x_n \leftarrow a_1 \ldots a_n\}$ is the usual variable substitution.

By fresh name quantification we define a notion of $\vec{a}$-linked name quantification for fresh names, whose purpose is to identify names linked to $\vec{a}$, as

$$\vec{a}\,\mathbf{L}\,\vec{x}.\,A \overset{def}{=} \text{И}\vec{x}.\,((\vec{a} \Leftarrow \vec{x}) \otimes \mathbf{id}) \circ A.$$

The formula above expresses that the variables in $\vec{x}$ denote in A names that are linked in the term to $\vec{a}$, and the role of $(\vec{a} \Leftarrow \vec{x})$ is to link the fresh names $\vec{x}$ with $\vec{a}$, while **id** deals with names not in $\vec{a}$. We also define a *separation-up-to* as the decomposition in two terms that are separated apart from the link on the specific names in $\vec{a}$, which crosses the separation line:

$$A \overset{\vec{a}}{\otimes} B \overset{def}{=} \vec{a}\,\mathbf{L}\,\vec{x}.\,(((\vec{x} \leftarrow \vec{a}) \otimes \mathbf{id}) \circ A) \otimes B. \tag{4}$$

The idea of the formula above is that the shared names $\vec{a}$ are renamed in fresh names $\vec{x}$, so that the product can be performed and finally $\vec{x}$ is linked to $\vec{a}$ to actually have the sharing.

The following lemma states that the two definition are consistent.

Lemma 3 (Separation-up-to) *If $g \models A \overset{\vec{x}}{\otimes} B$ with $g : \epsilon \to X$, and $\vec{x}$ is the vector of the elements in X, then there exist $g_1 : \epsilon \to X$ and $g_2 : \epsilon \to X$ such that $g \equiv g_1 \overset{\vec{x}}{\otimes} g_2$ and $g_1 \models A$ and $g_2 \models B$.*

Proof. Simply apply the definitions and observe that the identities must be necessarily id_ϵ, as the outer face of g is restricted to be X. □

The corresponding parallel composition operator is not directly definable by using the separation-up-to. In fact, in arbitrary decompositions the name shared are not all known a priori, hence we would not know the vector $\vec{x}$ in the operator sharing/separation operator $\overset{\vec{x}}{\otimes}$. However, next section shows that a careful encoding is possible for the parallel composition of spatial logics with nominal resources.

5.4 Encoding SGL

We show that LGL can be seen as a contextual (multi-edge) version of Spatial Graph Logic (SGL) [10]. The logic SGL expresses properties of directed graphs G with labelled edges. The notation $a(x, y)$ represents an edge from the node x to y and labelled by a. The graphs G are built from the empty graph *nil* and the edge $a(x, y)$ by using the parallel composition $G_1 \mid G_2$ and the binding for local names of nodes $(\nu x)G$. The syntax and the structural congruence for spatial graphs are outlined in Tab. 5.6.

The graph logic combines standard propositional logic with the structural connectives: composition and basic edge. Although we focus on its propositional fragment,

Table 5.6. *Spatial graph Terms (with local names) and congruence*

$G, G' ::=$	nil	empty graph
	$a(x,y)$	single edge graph labelled $a \in \Lambda$ connecting the nodes x, y
	$G \mid G'$	composing the graphs G, G', with sharing of nodes
	$(\nu x)G$	the node x is local in G

$G \mid nil \equiv G$	neutral element
$G \mid G' \equiv G' \mid G$	commutativity
$(G \mid G') \mid G'' \equiv G \mid (G' \mid G'')$	associativity
$y \notin fn(G)$ implies $(\nu x)G \equiv (\nu y)G\{x \leftarrow y\}$	renaming
$(\nu x)nil \equiv nil$	extrusion Zero
$x \notin fn(G)$ implies $G \mid (\nu x)G' \equiv (\nu x)(G \mid G')$	extrusion composition
$x \neq y, z$ implies $(\nu x)a(y,z) \equiv a(y,z)$	extrusion edge
$(\nu x)(\nu y)G \equiv (\nu y)(\nu x)G$	extrusion restriction

Table 5.7. *Propositional Spatial Graph Logic (SGL)*

$\varphi, \psi ::=$	$\mathbf{F}$	false	$a(x,y)$	an edge from x to y
	$\mathbf{nil}$	empty graph	$\varphi \mid \psi$	composition
	$\varphi \Rightarrow \psi$	implication		

$G \models_{\text{STL}} \mathbf{F}$	iff	never
$G \models_{\text{STL}} \mathbf{nil}$	iff	$G \equiv nil$
$G \models_{\text{STL}} \varphi \Rightarrow \psi$	iff	$G \models_{\text{STL}} \varphi$ implies $G \models_{\text{STL}} \psi$
$G \models_{\text{STL}} a(x,y)$	iff	$G \equiv a(x,y)$
$G \models_{\text{STL}} \varphi \mid \psi$	iff	there exist G_1, G_2 s.t. $G_1 \models_{\text{STL}} \varphi$ and $G_2 \models_{\text{STL}} \psi$ and $G \equiv G_1 \mid G_2$

the logics of [10] also includes edge label quantifier and recursion. In [10] SGL is used as a pattern matching mechanism of a query language for graphs. In addition, the logic is integrated with *transducers* to allow graph transformations. The applications of SGL include description and manipulation of semistructured data. Table 5.7 depicts the syntax and the semantics of the fragment we consider.

We consider a signature $\mathcal{K}$ with controls of arity 2, we assume a bijective function associating every label a to a distinct control $\mathsf{K}(a)$. The ports of the controls represent the starting and arrival node of the associated edge. The transparency predicate is defined to be verified on every control. The resulting link graphs are interpreted as contextual graphs with labelled edges, whereas the resulting class of ground link graphs is isomorphic to the graph model of SGL.

Table 5.8 encodes the graphs modelling SGL into ground link graphs and SGL formulae into LGL formulae. The encoding is parametric on a finite set X of names containing the free names of the graph under consideration. Observe that when we force the outer face of the graphs to be a fixed finite set X, the encoding of parallel composition is simply the separation-up-to $\vec{x}$, where $\vec{x}$ is a list of all the elements in X. Notice also how local names are encoded into name closures. Thanks to the Connected Normal Form of [30], it is easy to prove that ground link graphs featuring controls with exactly two ports are isomorphic to spatial graph models.

Table 5.8. *Encoding Propositional SGL in LGL over ground link graphs*

Spatial Graphs into Two-ported Ground Link Graphs	
$[\![\, nil \,]\!]_X \stackrel{def}{=} X$	
$[\![\, a(x,y) \,]\!]_X \stackrel{def}{=} \mathsf{K}(a)_{x,y} \otimes X \setminus \{x,y\}$	
$[\![\, (\nu x)G \,]\!]_X \stackrel{def}{=} ((/x \otimes id_{X \setminus \{x\}}) \circ [\![\, G \,]\!]_{\{x\} \cup X})) \otimes (\{x\} \cap X)$	
$[\![\, G \mid G' \,]\!]_X \stackrel{def}{=} [\![\, G \,]\!]_X \overset{\vec{X}}{\otimes} [\![\, G' \,]\!]_X$	
SGL formulae into LGL formulae	
$[\![\, \mathbf{nil} \,]\!]_X \stackrel{def}{=} X$	$[\![\, a(x,y) \,]\!]_X \stackrel{def}{=} \mathsf{K}(a)_{x,y} \otimes (X \setminus \{x,y\})$
$[\![\, \mathbf{F} \,]\!]_X \stackrel{def}{=} \mathbf{F}$	$[\![\, \varphi \Rightarrow \psi \,]\!]_X \stackrel{def}{=} [\![\, \varphi \,]\!]_X \Rightarrow [\![\, \psi \,]\!]_X$
$[\![\, \varphi \mid \psi \,]\!]_X \stackrel{def}{=} [\![\, \varphi \,]\!]_X \overset{\vec{X}}{\otimes} [\![\, \psi \,]\!]_X$	

As we impose a bijection between arrows labels and controls, the signature and the label set must have the same cardinality.

Lemma 4 (Isomorphism for spatial graphs) *There is a mapping $(\![\ ]\!)$ from two-ported ground bigraphs to spatial graphs, such that for every set X of names:*

(1) The mapping $(\![\ ]\!)$ is inverse to $[\![\ ]\!]_X$.
(2) For every ground link graph g with outer face X in the signature featuring a countable set of controls $\mathcal{K}$, all with arity 2, it holds $fn((\![\, g \,]\!)) = X$ and $[\![\, (\![\, g \,]\!) \,]\!]_X \equiv g$.
(3) For every spatial graph G with $fn(G) = X$ it holds $[\![\, G \,]\!]_X : \epsilon \to X$ and $(\![\, [\![\, G \,]\!]_X \,]\!) \equiv G$.

Proof. The idea is to interpret link graphs as bigraphs of type $\epsilon \to \langle 1, X \rangle$ without nested nodes. As proved in [30], bigraphs without nested nodes and $\langle 1, X \rangle$ as outerface have the following normal form (where $Z \subseteq X$):

$$\begin{aligned} G &::= (/Z \mid id_{\langle 1,X \rangle}) \circ (X \mid M_0 \mid \ldots \mid M_{k-1}) \\ M &::= \mathsf{K}_{x,y}(a) \circ 1 \end{aligned}$$

The inverse encoding is based on such a normal form:

$$\begin{aligned} (\![\, (/Z \mid id_{\langle 1,X \rangle}) \circ (X \mid M_0 \mid \ldots \mid M_{k-1}) \,]\!) &\stackrel{def}{=} (\nu Z) \, (nil \mid (\![\, M_0 \,]\!) \mid \ldots \mid (\![\, M_{k-1} \,]\!)) \\ (\![\, \mathsf{K}_{x,y}(a) \circ 1 \,]\!) &\stackrel{def}{=} a(x,y) \end{aligned}$$

Notice that the extrusion properties of local names correspond to node and link axioms. The encodings $[\![\ ]\!]$ and $(\![\ ]\!)$ provide a bijection, up to congruence, between graphs of SGL with free names X and ground link graphs with outer face X and built by controls of arity two. □

The previous lemma is fundamental in proving the soundness of the encoding for SGL in BiLog, stated in the following theorem.

Theorem 3 (Encoding SGL) *For every graph G, every finite set X that contains* *fn*(G)*, and every formula* φ *of the propositional fragment of SGL:*

$$G \models_{\text{SGL}} \varphi \quad \textit{if and only if} \quad [\![G]\!]_X \models [\![\varphi]\!]_X .$$

Proof. By induction on formulae of SGL. The transparency predicate is not considered here, as it is verified on every control. The basic step deals with the constants **F**, **nil** and $a(x, y)$. Case **F** follows by definition. For the case **nil**, $[\![G]\!]_X \models [\![\mathbf{nil}]\!]_X$ means $[\![G]\!]_X \models X$, that by definition is $[\![G]\!]_X \equiv X$ and so $G \equiv (\!| [\![G]\!]_X |\!) \equiv (\!| X |\!) \overset{def}{=} nil$, namely $G \models_{\text{SGL}} \mathbf{nil}$. For the case $a(x, y)$, to assume $[\![G]\!]_X \models [\![a(x, y)]\!]_X$ means $[\![G]\!]_X \models \mathsf{K}(a)_{x,y} \otimes X \setminus \{x, y\}$. So $G \equiv (\!| [\![G]\!]_X |\!) \equiv (\!| \mathsf{K}(a)_{x,y} \otimes X \setminus \{x, y\} |\!) \equiv a(x, y)$, that is $G \models_{\text{SGL}} a(x, y)$. The inductive steps deal with connectives.

Case $\varphi \Rightarrow \psi$**.** To assume $[\![G]\!]_X \models [\![\varphi \Rightarrow \psi]\!]_X$ means $[\![G]\!]_X \models [\![\varphi]\!]_X \Rightarrow [\![\psi]\!]_X$; by definition this says that $[\![G]\!]_X \models [\![\varphi]\!]_X$ implies $[\![G]\!]_X \models [\![\psi]\!]_X$. By induction hypothesis, this is equivalent to say that $G \models_{\text{SGL}} \varphi$ implies $G \models_{\text{SGL}} \psi$, namely $G \models_{\text{SGL}} \varphi \Rightarrow \psi$.

Case $\varphi \mid \psi$**.** To assume $[\![G]\!]_X \models [\![\varphi \mid \psi]\!]_X$ means $[\![G]\!]_X \models [\![\varphi]\!]_X \overset{\vec{x}}{\otimes} [\![\psi]\!]_X$. By Lemma 3 there exists g_1, g_2 such that $[\![G]\!]_X \equiv g_1 \overset{\vec{x}}{\otimes} g_2$ and $g_1 \models [\![\varphi]\!]_X$ and $g_2 \models [\![\psi]\!]_X$. Let $G_1 = (\!| g_1 |\!)$ and $G_2 = (\!| g_2 |\!)$, Lemma 4 says that $[\![G_1]\!]_X \equiv g_1$ and $[\![G_2]\!]_X \equiv g_2$, and by conservation of congruence, $[\![G_1]\!]_X \models [\![\varphi]\!]_X$ and $[\![G_2]\!]_X \models [\![\psi]\!]_X$. Hence the induction hypothesis says that $G_1 \models_{\text{SGL}} \varphi$ and $G_2 \models_{\text{SGL}} \psi$. In addition $[\![G_1 \mid G_2]\!]_X \equiv [\![G_1]\!]_X \overset{\vec{x}}{\otimes} [\![G_2]\!]_X \equiv g_1 \overset{\vec{x}}{\otimes} g_2 \equiv [\![G]\!]_X$. Conclude that G admits a parallel decomposition with parts satisfying A and B, thus $G \models_{\text{SGL}} \varphi \mid \psi$. $\square$

Also, LGL enables the encoding of Separation Logics on heaps: names used as identifiers of location are forcibly separated by tensor product, while names used for pointers are shared/linked. However we do not encode it explicitly since in §5.7 we will encode a more general logic: the Context Tree Logic [7].

5.5 Pure Bigraph Logic

By combining link graphs and place graphs we generate all the *(abstract pure) bigraphs* of [25]. In this case the underlying monoid is the product of link and place interfaces, namely $(\omega \times \mathcal{P}_{fin}(\Lambda), \otimes, \epsilon)$ where $\langle m, X \rangle \otimes \langle n, X \rangle \overset{def}{=} \langle m + n, X \uplus Y \rangle$ and $\epsilon \overset{def}{=} \langle 0, \emptyset \rangle$. As a short notation, we use X for $\langle 0, X \rangle$ and n for $\langle n, \emptyset \rangle$.

A set of constructors for bigraphical terms is obtained as the union of place and link graph constructors, except the controls which are subsumed by the new *discrete ion* constructors, denoted by $\mathsf{K}_{\vec{a}} : 1 \to \langle 1, \vec{a} \rangle$. It represents a prime bigraph containing a single node with ports named $\vec{a}$ and an hole inside. Bigraphical terms are thus defined in relation to a control signature $\mathcal{K}$ and a set of names Λ, as detailed in [30].

Table 5.9. *Additional axioms for Bigraph Structural Congruence*

Symmetric Category Axioms:	
$\gamma_{I,\epsilon} \equiv id_I$	Symmetry Id
$\gamma_{I,J} \circ \gamma_{J,I} \equiv id_{I \otimes J}$	Symmetry Composition
$\gamma_{I',J'} \circ (G \otimes F) \equiv (F \otimes G) \circ \gamma_{I,J}$	Symmetry Monoid
Place Axioms:	
$join \circ (1 \otimes id_1) \equiv id_1$	Unit
$join \circ (join \otimes id_1) \equiv join \circ (id_1 \otimes join)$	Associativity
$join \circ \gamma_{1,1} \equiv join$	Commutativity
Link Axioms:	
$^a/_a \equiv id_a$	Link Identity
$/a \circ {}^a/_b \equiv /b$	Closing renaming
$/a \circ a \equiv id_\epsilon$	Idle edge
$^b/_{(Y \uplus a)} \circ (id_Y \otimes {}^a/_X) \equiv {}^b/_{Y \uplus X}$	Composing substitutions
Node Axiom:	
$(id_1 \otimes \alpha) \circ \mathsf{K}_{\vec{a}} \equiv \mathsf{K}_{\alpha(\vec{a})}$	Renaming

The structural congruence for bigraphs corresponds to the sound and complete bigraph axiomatisation of [30]. The additional axioms are reported in Tab. 5.10: they are essentially a combination of the axioms for link and place graphs, with slight differences due to the interfaces monoid. In detail, we define the symmetry as $\gamma_{I,J} \overset{def}{=} \gamma_{m,n} \otimes id_{X \uplus Y}$ where $I = \langle m, X \rangle$ and $J = \langle n, Y \rangle$, and we restate the node axiom by taking care of the places.

PGL excels at expressing properties of *unnamed* resources, that are resources accessible only by following the structure of the term. On the other hand, LGL characterises names and their links to resources, but it has no notion of locality. A combination of them ought to be useful to model nominal spatial structures, either private or public.

BiLog promises to be a good (contextual) spatial logic for (semi-structured) resources with nominal links, thanks to bigraphs' orthogonal treatment of locality and connectivity. To testify this, §5.7 shows how recently proposed Context Logic for Trees (CTL) [7] can be encoded into bigraphs. The idea of the encoding is to extend the encoding of STL with (single-hole) contexts and identified nodes. First, §5.6 gives some details on the transparency predicate.

5.6 Transparency on bigraphs

In the logical framework we gave the minimal restrictions on the transparency predicate to prove our results. Here we show a way to define a transparency predicate. The most natural way is to make the transparent terms a sub-category of the more general category of terms. This essentially means to impose the product and the composition of two transparent terms to be transparent. Thus transparency on all terms can be derived from a transparency policy, i.e., a predicate $\tau_\Theta(\,)$ defined only on the constructors as follows.

Definition 2 (Transparency) *Given the monoid of interfaces $(M, \otimes, \epsilon)$, the set of constructors Θ, the congruence $\equiv$ and a transparency policy predicate τ_Θ defined on the constructors in Θ we define the transparency on terms as follows:*

$$\frac{G \equiv id_I}{\tau(G)} \qquad \frac{\exists I.G : \epsilon \to I}{\tau(G)} \qquad \frac{G \equiv \Omega \quad \tau_\Theta(\Omega)}{\tau(G)}$$

$$\frac{G \equiv G_1 \otimes G_2 \quad \tau(G_1) \quad \tau(G_2)}{\tau(G)} \qquad \frac{G \equiv G_1 \circ G_2 \quad \tau(G_1) \quad \tau(G_2)}{\tau(G)}$$

Next lemma proves that the conditions we required on the transparency predicate holds for this particular definition.

Lemma 5 (Transparency properties) *If G is ground or G is an identity then $\tau(G)$ is verified. Moreover, if $G \equiv G'$ then $\tau(G)$ is equivalent to $\tau(G')$.*

Proof. The former statement is verified by definition. The latter is proved by induction on the derivations. □

We assume every bigraphical constructor, which is not a control, to be transparent and the transparency policy to be defined only on the controls. The transparency the policy can be defined, for instance, by security requirements.

5.7 Encoding CTL

Paper [7] presents a spatial context logic to describe programs manipulating a tree structured memory. The model of the logic is the set of unordered labelled trees T and *linear contexts* C, which are trees with a unique hole. Every node has a name, so to identify memory locations. From the model, the logic is dubbed Context Tree Logic, CTL in the following. Given a denumerable set of labels and a denumerable set of identifiers, trees and contexts are defined in Tab. 5.10: a represents a label and x an identifier. The insertion of a tree T in a context C, denoted by $C(T)$, is defined in the standard way, and corresponds to fill the unique hole of C with the tree T. A *well formed tree* or *context* is one where the node identifiers are unique. The model of the logic is composed by trees and contexts that are well formed. In particular, composition, node formation and tree insertion are *partial* as they are restricted to well-formed trees. The structural congruence between trees is the smallest congruence that makes the parallel operator to be commutative, associative and with the empty tree as neutral element. Such a congruence is naturally extended to contexts.

Table 5.10. *Trees with pointers and Tree Contexts*

T, T'	$::=$	0	empty tree
		$a_x[T]$	a tree labelled a with identifier x and subtree T
		$T \mid T'$	partial parallel composition
C	$::=$	$-$	an hole (the identity context)
		$a_x[C]$	a tree context labelled a with identifier x and subtree C
		$T \mid C$	context right parallel composition
		$C \mid T$	context left parallel composition

Table 5.11. *Context Tree Logic (CTL)*

P, P'	$::=$	$false$	
		$\mathbf{0}$	empty tree formula
		$K(P)$	context application
		$K \triangleleft P$	context application adjunct
		$P \Rightarrow P'$	implication
K, K'	$::=$	$false$	
		$-$	identity context formula
		$a_x[K]$	node context formula
		$P \triangleright P'$	context application adjunct
		$P \mid K$	parallel context formula
		$K \Rightarrow K'$	implication

The logic exhibits two kinds of formulae: P, describing trees, and K, describing tree contexts. It has two spatial constants, the empty tree for P and the hole for K, and four spatial operators: the node formation $a_x[K]$, the application $K(P)$, and its two adjuncts $K \triangleright P$ and $P_1 \triangleleft P_2$. The formula $a_x[K]$ describes a context with a single root labelled by a and identified by x, whose content satisfies K. The formula $K \triangleright P$ represents a tree that satisfies P whenever inserted in a context satisfying K. Dually, $P_1 \triangleleft P_2$ represents contexts that composed with a tree satisfying P_1 produce a tree satisfying P_2. The complete syntax of the logic is outlined in Tab. 5.11, the semantics in 5.12.

CTL can be naturally embedded in an instance of BiLog. The complete structure of the Context Tree Logic has also link values. For sake of simplicity, we restrict our attention to the fragment without links. As already said, the terms giving a semantics to CTL do not to share identifiers: two nodes cannot have the same identifier, as it represents a precise location in the memory. This is easily obtained with bigraph terms by encoding the identifiers as names and the composition as tensor product, that separates them. We encode such a structure in BiLog by lifting the application to a particular kind of composition, and similarly for the two adjuncts.

The tensor product on bigraphs is both a spatial separation, like in the models for STL, and a partially-defined separation on names, like pointer composition for separation logic. Since we deal with both names and places, we define a formula $\mathbf{id}_{\langle m,\_\rangle}$ to represent identities on places by constraining the place part of the interface to

Table 5.12. *Semantics for CTL*

$T \models_{\mathcal{T}} false$	iff	never
$T \models_{\mathcal{T}} \mathbf{0}$	iff	$T \equiv 0$
$T \models_{\mathcal{T}} K(P)$	iff	there exist C, T' s.t. $C(T')$ well-formed, and $T \equiv C(T')$ and $C \models_{\mathcal{K}} K$ and $T' \models_{\mathcal{T}} P$
$T \models_{\mathcal{T}} K \triangleleft P$	iff	for every C: $C \models_{\mathcal{K}} K$ and $C(T)$ well-formed implies $C(T) \models_{\mathcal{T}} P$
$T \models_{\mathcal{T}} P \Rightarrow P'$	iff	$T \models_{\mathcal{T}} P$ implies $T \models_{\mathcal{T}} P'$
$C \models_{\mathcal{K}} false$	iff	never
$C \models_{\mathcal{K}} -$	iff	$C \equiv -$
$C \models_{\mathcal{K}} a_x[K]$	iff	there exists C' s.t. $a_x[C']$ well-formed, and $C \equiv a_x[C']$ and $C' \models_{\mathcal{K}} K$
$C \models_{\mathcal{K}} P \triangleright P'$	iff	for every T: $T \models_{\mathcal{T}} P$ and $C(T)$ well-formed implies $C(T) \models_{\mathcal{T}} P'$
$C \models_{\mathcal{K}} P \mid K$	iff	there exist C', T s.t. $T \mid C'$ well-formed, and $C \equiv T \mid C'$ and $T \models_{\mathcal{T}} P$ and $C' \models_{K} K$
$C \models_{\mathcal{K}} K \Rightarrow K'$	iff	$C \models_{\mathcal{K}} K$ implies $T \models_{\mathcal{T}} K'$

be fixed and leaving the name part to be free: $\mathbf{id}_{\langle m,\_\rangle} \overset{def}{=} \mathbf{id}_m \otimes (\mathbf{id} \wedge \neg(\mathbf{id}_1^{\exists\otimes}))$. The semantics says that $G \models id_{\langle m,-\rangle}$ means that there exits a set of names X such that $G \equiv id_m \otimes id_X$. By using such an identity formula we define the corresponding typed composition $\circ_{\langle m,\_\rangle}$ and the typed adjuncts $\circ\!-_{\langle m,\_\rangle}$, $-\!\circ_{\langle m,\_\rangle}$:

$$\begin{aligned} A \circ_{\langle m,\_\rangle} B &\overset{def}{=} A \circ \mathbf{id}_{\langle m,\_\rangle} \circ B \\ A \circ\!-_{\langle m,\_\rangle} B &\overset{def}{=} (\mathbf{id}_{\langle m,\_\rangle} \circ A) \circ\!- B \\ A -\!\circ_{\langle m,\_\rangle} B &\overset{def}{=} (A \circ \mathbf{id}_{\langle m,\_\rangle}) \circ\!- B \end{aligned}$$

We then define the operator $*$ for the parallel composition with separation operator $*$ as both a term constructor and a logical connective:

$$\begin{array}{ll} D * E \overset{def}{=} [join](D \otimes E) & \text{for } D \text{ and } E \text{ prime bigraphs} \\ A * B \overset{def}{=} (\mathbf{join} \otimes \mathbf{id}_{\langle 0,\_\rangle}) \circ (A_{\to\langle 1,\_\rangle} \otimes B_{\to\langle 1,\_\rangle}) & \text{for } A \text{ and } B \text{ formulae} \end{array}$$

The operator $*$ enables the encoding of trees and contexts to bigraphs. In particular, we consider a signature with controls of arity 1 and we define the transparency predicate to be verified on every control. Moreover we assume a bijective function from tags to controls: $a_x \longmapsto \mathsf{K}(a)_x$. The details are outlined in Tab. 5.13. The encodings of trees turn out to be *ground prime discrete bigraphs*: bigraphs with open links and type $0 \to \langle 1, X\rangle$. The result in [30] says that the normal form, up to permutations, for ground prime discrete bigraphs is:

$$g = (join_k \otimes id_X) \circ (M_1 \otimes \ldots \otimes M_k),$$

where M_i are called *discrete ground molecules* and are of the form $M = (\mathsf{K}(a)_x \otimes id_Y)g$. We can now define the reverse encoding $(\!|\ \ |\!)$ of $[\![\ \ ]\!]$, from ground prime

Table 5.13. *Encoding CTL in BiLog over prime discrete ground bigraphs*

Trees into prime ground	Contexts into unary discrete bigraphs
discrete bigraphs	$\llbracket - \rrbracket_C \stackrel{def}{=} id_1$
$\llbracket 0 \rrbracket \stackrel{def}{=} 1$	$\llbracket a_x[C] \rrbracket_C \stackrel{def}{=} (\mathsf{K}(a)_x \otimes id_{fn(C)}) \circ \llbracket C \rrbracket_C$
$\llbracket a_x[T] \rrbracket \stackrel{def}{=} (\mathsf{K}(a)_x \otimes id_{fn(T)}) \circ \llbracket T \rrbracket$	$\llbracket T \mid C \rrbracket_C \stackrel{def}{=} \llbracket T \rrbracket * \llbracket C \rrbracket_C$
$\llbracket T_1 \mid T_2 \rrbracket \stackrel{def}{=} \llbracket T_1 \rrbracket * \llbracket T_2 \rrbracket$	$\llbracket C \mid T \rrbracket_C \stackrel{def}{=} \llbracket C \rrbracket_C * \llbracket T \rrbracket$
TL formulae into PGL formulae	CTL formulae into PGL formulae
$\llbracket \mathit{false} \rrbracket_P \stackrel{def}{=} \mathbf{F}$	$\llbracket \mathit{false} \rrbracket_K \stackrel{def}{=} \mathbf{F}$
$\llbracket \mathbf{0} \rrbracket_P \stackrel{def}{=} \mathbf{1}$	$\llbracket - \rrbracket_K \stackrel{def}{=} \mathbf{id}_1$
$\llbracket K(P) \rrbracket_P \stackrel{def}{=} \llbracket K \rrbracket_K \circ_{\langle 1,\_\rangle} \llbracket P \rrbracket_P$	$\llbracket P \triangleright P' \rrbracket_K \stackrel{def}{=} \llbracket P \rrbracket_P \multimap_{\langle 1,\_\rangle} \llbracket P' \rrbracket_P$
$\llbracket K \triangleleft P \rrbracket_P \stackrel{def}{=} \llbracket K \rrbracket_K \multimapinv_{\langle 1,\_\rangle} \llbracket P \rrbracket_P$	$\llbracket a_x[K] \rrbracket_K \stackrel{def}{=} ((\mathsf{K}(a)_x) \otimes id_{\langle 0,\_\rangle}) \circ \llbracket K \rrbracket_K$
$\llbracket P \Rightarrow P' \rrbracket_P \stackrel{def}{=} \llbracket P \rrbracket_P \Rightarrow \llbracket P' \rrbracket_P$	$\llbracket P \mid K \rrbracket_K \stackrel{def}{=} \llbracket P \rrbracket_P * \llbracket K \rrbracket_K$
$\llbracket K \Rightarrow K' \rrbracket_K \stackrel{def}{=} \llbracket K \rrbracket_K \Rightarrow \llbracket K' \rrbracket_K$	

discrete bigraphs to trees, involving such a normal form:

$$\llparenthesis\, join_0 \,\rrparenthesis \stackrel{def}{=} 0$$

$$\llparenthesis\, (\mathsf{K}(a)_x \otimes id_Y) \circ g \,\rrparenthesis \stackrel{def}{=} a_x[\, \llparenthesis\, g \,\rrparenthesis\,]$$

$$\llparenthesis\, (join_k \otimes id_Y) \circ (M_1 \otimes \ldots \otimes M_k) \,\rrparenthesis \stackrel{def}{=} \llparenthesis\, M_1 \,\rrparenthesis * \ldots * \llparenthesis\, M_k \,\rrparenthesis$$

Moreover, the encodings of linear contexts turn out to be *unary discrete bigraphs* G: bigraphs with open links and type $\langle 1, X\rangle \to \langle 1, Y\rangle$. Again, the result in [30] implies that the normal form, up to permutations, for unary discrete bigraphs is:

$$G = (join_k \otimes id_Y) \circ (R \otimes M_1 \otimes \ldots \otimes M_{k-1})$$

where M_i are discrete ground molecules and R can be either id_1 or $(\mathsf{K}_{\vec{a}} \otimes id_Y) \circ Q$. Again, we can define the reverse encoding $\llparenthesis\ \ \rrparenthesis$ of $\llbracket\ \ \rrbracket$, from unary discrete bigraphs to linear contexts, involving such a normal form:

$$\llparenthesis\, id_1 \,\rrparenthesis \stackrel{def}{=} -$$

$$\llparenthesis\, (\mathsf{K}(a)_x \otimes id_Y) \circ Q \,\rrparenthesis \stackrel{def}{=} a_x[\llparenthesis\, Q \,\rrparenthesis]$$

$$\llparenthesis\, (join_k \otimes id_Y) \circ (R \otimes M_1 \otimes \ldots \otimes M_{k-1}) \,\rrparenthesis \stackrel{def}{=} \llparenthesis\, R \,\rrparenthesis \mid \llparenthesis\, M_1 \,\rrparenthesis \mid \ldots \mid \llparenthesis\, M_{k-1} \,\rrparenthesis$$

As the bigraphical model is specialised to context trees, so BiLog logic is specialised to the Context Tree Logic. The encodings of the connectives and the constants are in Tab. 5.13, and their soundness is shown in the next lemma.

Theorem 4 (Encoding Context Tree Logic) *For each tree T and formula P of CTL, $T \models_{\mathcal{T}} P$ if and only if $\llbracket T \rrbracket \models \llbracket P \rrbracket_P$. Also, for each context C and formula K of CTL, $C \models_{\mathcal{K}} K$ if and only if $\llbracket C \rrbracket_C \models \llbracket K \rrbracket_K$.*

Proof. Follow the lines of Theorem 2 and 3, by structural induction on CTL formulae and by exploiting the fact that the encoding of contexts trees into unary discrete bigraphs is bijective. □

The encoding shows that the models introduced in [7] are a particular kind of discrete bigraphs with one port for each node and a number of holes and roots limited to one. Hence, this shows how BiLog for discrete bigraphs is a generalisation of Context Tree Logic to contexts with several holes and regions. On the other hand, since STL is more general than separation logic, cf. [7], and it is used to characterise programs that manipulate tree structured memory model, BiLog can express separation logic as well.

6 BiLog for XML data and contexts

XML data are essentially tree-shaped resources. Starting from [8], where XML data were modelled by unordered labelled trees, much work on spatial logic for semistructured data and XML has been proposed [10,11,21]. A query language on semistructured data based on Ambient Logic was studied in [12]. Here we add links on resource names to that tree-shaped model, so as to obtain a more general framework for semistructured data and XML. A similar step was undertaken in [9]. As bigraphs naturally model XML contexts, here we improve on [9] by showing that BiLog is suitable to describe XML contexts, which can be interpreted as web services or XML transformations.

Here we focus on the applications of BiLog to XML data. In particular, we first show how XML data, contexts, and a class of web services can be interpreted as a bigraph. Then, equipped with a 'bigraphical' representation of XML data and contexts, we show how BiLog can describe and reason about XML.

6.1 Modelling XML Contexts as Bigraphs

The importance of the underlying hierarchical structure in XML, as well as the fact that links are used only to model relations between nodes, suggests bigraphs as good models for XML documents. Ground bigraphs represent XML documents, while those with holes represent XML contexts. The interpretation is trivial when nominal constraints (such as `ID` and `IDREF` attributes and namespaces) are not considered. Without nominal attributes there is in fact no link between nodes, and XML tree structures can be mapped to place graphs by associating tags and values to bigraphical controls with arity zero. This yields an ambient-like formalism [8].

To model nominal resources and links, controls must be enriched by identification and pointer ports, connected to each other by the link graph. The model so obtained is similar to the one in [9], where *trees with dangling pointers* are considered. In

Table 6.1. *XML documents as ground bigraphs*

$(\!\| v \|\!) \overset{def}{=} K_{val}(v)$	value
$(\!\| v \|\!)_a \overset{def}{=} K_{val}(v)_a$	value linked to an attribute name a
$(\!\| \vec{v} \|\!)_{\vec{b}} \overset{def}{=} (\!\| v_1 \|\!)_{b_1} \otimes \ldots \otimes (\!\| v_n \|\!)_{b_n}$	with $\vec{v} = v_1 \ldots v_n$ and $\vec{b} = b_1 \ldots b_n$
$(\!\| \emptyset \|\!) \overset{def}{=} 1$	empty tree
$(\!\| T \|\!) \overset{def}{=} /\vec{a} \circ \sigma \circ K_{tag}(t)_{u,\vec{u},\vec{b}} \circ join_{n+k}((\!\| \vec{v} \|\!)_{\vec{b}} \otimes \alpha_1 \circ (\!\| T_1 \|\!) \otimes \ldots \otimes \alpha_n \circ (\!\| T_n \|\!))$	
with $T = \langle t, \mathtt{ID} = u, \vec{a} = \vec{u}, \vec{b} = \vec{v} \rangle T_1, ..., T_n \langle /t \rangle$	XML tree
$\vec{a} = a_1 \ldots a_k$	link attributes
$\vec{u} = u_1 \ldots u_k$	names
$\vec{b} = b_1 \ldots b_p$	value attributes
$\vec{v} = v_1 \ldots v_k$	values
α_i	renaming the names of T_i into fresh names
$\sigma = \alpha_1^{-1} \cup \ldots \cup \alpha_n^{-1}$	inverse renaming
$/\vec{a} \overset{def}{=} /a_1 \otimes \ldots \otimes /a_p$	closure of the names in $\vec{a}$
$join_{n+k}$	merging among $n + k$ bigraphs (definable from $join$)

addition, link graphs model local names, and so also unnamed connections.

As seen in §5.5, the main constituents of a bigraph are the discrete ions $\mathsf{K}_{\vec{a}}$, whose ports are linked to the names in $\vec{a}$. In XML settings, a ion represents a *tag* with some *attributes*. Since ports are unambiguously identified, they can be associated to attributes. The first port of a ion is associated to a (unique) name, which identifies, as an `ID` attribute, the element represented by the ion. Other ports are linked either to other nodes' `ID`s, so acting effectively as `IDREF`s, or to internal edges connected to internal nodes, so representing general attributes. Example 4 will clarify the idea. Embedding a ion into the hole of another ion, represents the inclusion of the corresponding elements.

XML data are encoded as ground bigraphs as outlined in Tab. 6.1. Without attributes, XML data are completely modelled by the place graph, since the arity is zero for every bigraphical control. When dealing with attributes, names and edges represent XML attributes and XML links between elements, respectively. We consider the `ID`s used in XML data as names and we assume two functions for values:

$K_{val}(v)$ maps the value v to a single node with no outer names, no nodes and no holes inside, and it is actually used to encode the value v by bigraphs.

$K_{val}(v)_a$ maps the value v a single node with outer name a, no nodes and no holes inside, and it is auxiliary to encode values linked to attributes.

We assume a class $\mathcal{K}_{tag}$ of controls. Let t be an XML tag, and Att the list of attributes for t. Being finite and ordered, the list Att can be associated to an ordinal $\#Att$. In particular, every attribute can be identified by the position. So the tag t is associated to $K_{tag}(t,)_{\vec{u}}$, which represents the ion with control $K_{tag}(t) \in \mathcal{K}_{tag}$ and arity $\#Att$. The vector $\vec{u}$ indicates the names connected to the control. These names correspond to the `ID`s associated to the attributes in Att. A value attribute is encoded as a value inside the node and connected to the port whose position marks the corresponding attribute. Identifiers (`ID`) and links (`IDREF`) attributes become *names* of the tag and can be connected with other names to model references. The connection is performed by link graph constructors: $a \leftleftarrows b$, to create a reference, and $/a$, to create a closed connection for attributes.

In Tab. 6.1 the term 1 corresponds to the empty tree. The core of the translation is the encoding of (non empty) trees. Here, the role of *join* is to group together the (encodings of the) set of children of T and the (encodings of the) values linked to attributes. The renamings α_i guarantee that the product is defined and they are obtained by choosing fresh names, not appearing in the encoded tree, and by combining operators $a \leftarrow b$. The bigraph obtained by *join* is single-rooted, thus it fits in the ion associated to the tag t. After the composition with the ion, names are renamed in order to actualise all the references, finally the links between the root and the values linked to attributes are closed. The renaming is obtained by considering the inverse of α_i (definable by using the operators $a \leftarrow b$ and $a \leftleftarrows b$), and the closure is obtained by combining the closures of the names associated to attributes.

Example 4 Consider a database that stores scientific papers and information about their authors, and focus on the fragment quoted in the document below.

```
<authors>
 <author n="ID1" name="Conf" coauth="ID4">
   <add n="ID2">"..."</add>   <phon n="ID3">"..."</phon>
 </author>
 <author n="ID4" name="Mace" coauth="ID7">
   <add n="ID5">"..."</add>   <phon n="ID6">"..."</phon>
 </author>
 <author n="ID7" name="Sass" coauth="ID10">
   <add n="ID8">"..."</add>   <phon n="ID9">"..."</phon>
 </author>
</authors>
```

Tag `author` has the following attributes: an identifier `IDn`, a link to another author `coauth`, that is an `IDREF`, and a general attribute `name`. In the corresponding bigraphical encoding (see Fig. 6.3) every tag `author` is associated to a control of arity three. Exploiting the order of the ports, we identify a port with the corresponding XML attribute unambiguously. In the picture we assume the ports ordered clockwise. The first port corresponds to the identifier, `ID`, and is connected to an outer name. The second one corresponds to the general attribute `name`, and is connected by a closed link to a value. The final attribute corresponds to the reference, `coauth`, and it is connected to a name that corresponds to another `author` tag.

This encoding does not capture the order among children of a node, so they represent lists of unordered trees connected through links. This model can be used

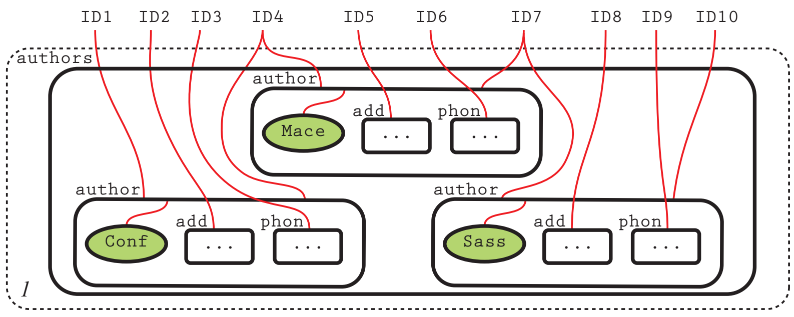

Fig. 3. XML encoding

for XML data whose document order is not relevant, as, for instance, for XML encodings of relational databases [2], or for distributed XML documents in a P2P computing, or *semantics web* where attaching meaning to denote order is undesirable. Sorting disciplines may provide an encoding that respects the order.

More generally, a bigraph represents a context for unordered XML data, just because there can be holes in it. So in Ex. 6.4 we can imagine holes in place of some nodes. This yields a contextual XML document, representing a function, or *web service*, that takes a list of XML files and returns their composition in the context, by fitting every file in the relative position. In this way, besides plain XML documents, we can model web services.

6.2 *BiLog for XML Contexts*

This section informally discusses how BiLog can be used for describing, querying and reasoning about XML. We analyse three possible cases: *(i)* PGL to model XML data trees and tree contexts, without nominal resources; *(ii)* logics for *discrete bigraphs* to model XML data trees with identified nodes; *(iii)* BiLog to model XML data trees with soft-link connections, that are implemented with nominal resources.

XML without IDs As said in §6.1, without nominal resources XML amounts to unordered labelled tree. In [8] the author outlines the similarities between such a model and ambient calculus. Then Ambient Logic is used in [12] to introduce a query language for semistructured. In §5.2 and § 5.7 we show that PGL extends the static fragment of ambient logic and models general contexts of tree-shaped resources. Hence it can describe XML contexts, without attributes.

The models of PGL are *positive* functions $m \to n$, which produce a list of n XML contexts from a list of m XML contexts. The adjective 'positive' means that the functions can only *add* structure to the parameters, without removing or replace any part of XML data. In this sense, XML contexts are viewed as positive XML web services that take XML documents and return XML documents. This is similar to Positive Active XML [1], but presents a remarkable difference, as the bigraphical model does not handle ordered trees. We use a *list* of parameters and a *list* of resulting contexts. For instance, consider a web service *wb* that satisfies the formula $K_1(id_1) \mid K_2(id_2)$. This web service takes two trees and puts the first inside a node labelled by K_1, then it puts the second inside a node labelled by K_2, and finally it performs a parallel composition between the two resulting trees. The ordered parameters are required to fix the exact correspondence between holes and roots. The web service *wb* is characterised by the formula above, but it satisfies also the formula $\mathsf{K}_1(id_1) \mid \mathbf{T}$. The formula characterises web services which have at least one hole and are the composition of a node with arity one labelled by K_1 in parallel with something else. In this sense a notion of *type* for web services arises: we can use PGL to formalise web service types and constraints.

Since XML active documents are contexts, PGL actually describes active XML documents and web service in an unique framework. In addition, an approach similar to TQL [12] can be used to query Active XML documents and web service. PGL may be eventually used to type web service in order to avoid useless invocations.

XML Contexts with identified nodes A simple tree structure does not allow logic and model to directly identify the resources, which are accessed only through navigation. When XML documents have nominal resources in addition to the tree structure, names can refer to locations, hence the resulting model can be seen as an extension of a heap memory model. In particular, names are intrinsically separated by the tensor product. Trees with names correspond to discrete bigraphs, namely place graphs with named resources but no name sharing between different resources. PGL extended by named controls K_x and renamings $x \leftarrow y$ is suitable to describe these models. In detail, K_x denotes a node labelled by K, with name identifier x, and an hole inside. The rename $x \leftarrow y$ is suitable to map names of different sources to different identifiers. The tensor product constraints two models to be separated both in locality and in names. In fact, a models satisfies $A \otimes B$ if it has two sub-models satisfying A and B respectively and with disjoint sets of identifiers, i.e., disjoint outer faces. Such a PGL extension characterises (contexts of) resources which can be accessed either by navigation through the tree structure or by using name controls as pointers.

XML Contexts with Connections For XML data models, nodes which are not related by a parent-child relationship can be connected either explicitly by `ID` and `IDREF` attributes or implicitly by namespaces. BiLog's notion of sharing can model connections between resources to treat structures with pointers. Sharing is obtained through links between names of resources. In Tab. 1, identifiers are encoded as tag names and `IDREF`s as pointers to names in the same document. The connection between `ID` and `IDREF` is expressed in BiLog by closed names. Moreover the 'separation-up-to' operator, defined in (4), can express properties like "The author of paper X has a relationship with the author of paper Y," which express separation on resources, since there are different authors for different papers, but sharing on linked names. BiLog can also express XML contexts with links. For instance a alteration to a namespace can be represented by a link composed to an identity, and unnamed resources can be represented by closed names.

7 Towards dynamics

A main feature of a distributed system is mobility, or dynamics in general. In dealing with communicating and nomadic processes, the interest is to describe not only their internal structure, but also their behaviour. So far, it has been shown how BiLog can describe structures, this section is intended to study how to express evolving systems. BiLog is able to deal with the dynamic behaviour of models. Essentially, this is due to its the contextual nature, suitable to characterise structural parametric reaction rules that model dynamics.

The usual way to express dynamics with a logic is to introduce a *next step* modality ($\Diamond$), that hints how the system develops in the future. In general, a process satisfies the formula $\Diamond A$ if it may evolve into a process satisfying A.

In process algebras, dynamics is often presented by *reaction* (or rewriting) rules of the form $r \longrightarrow r'$, meaning that the term r (the *redex*) is replaced by r' (the *reactum*) in *suitable* contexts, named *active*. The 'activeness' is defined on the structure of contexts by a predicate δ.

In general, a *bigraphical reactive system* is a bigraphical system provided with a set of parametric reaction rules, namely a set S of pairs[2] $(R, R' : I \to J)$, where R and R' are the redex and the reactum of a parametric reaction. We consider only ground bigraphs, as they identifies processes, contrary to non-ground bigraphs that are open and identifies contexts. The active bigraphs are identified by the predicate δ, closed for compositions and *id*s. A ground bigraph g reacts to g' (written $g \longrightarrow g'$) if there is a couple $(R, R') \in S$, a set of names Y, a bigraph D (usually not ground) with $\delta(D)$ true, and a ground bigraph d, such that:

$$g \equiv D \circ (R \otimes id_Y) \circ d \quad \text{and} \quad g' \equiv D \circ (R' \otimes id_Y) \circ d.$$

When the model is enriched with a dynamical framework, the usual way to introduce the modality $\Diamond$ is to extend the relation $\models$ by defining '*$g \models \Diamond A$ iff $g \longrightarrow g'$ and*

[2] This is a simplification to capture the case of CCS presented in this section. In general, bigraphical theory does not require R and R' to have the same inner face.

$g' \models A$.' According to the formulation of the reduction given above, we obtain

$$\begin{aligned} g \models \Diamond A \quad \textit{iff} \quad & \textit{there exist}(R, R') \in S\textit{, } id_Y\textit{, } D \textit{ active, and } d \textit{ ground} \\ & \textit{such that } g \equiv D \circ (R \otimes id_Y) \circ d \textit{ and } D \circ (R' \otimes id_Y) \circ d \models A. \end{aligned} \tag{5}$$

One may wonder whether the modality $\Diamond$ is the only way to express a temporal evolution in BiLog. It turns out that BiLog has a built in notion of dynamics. There are several cases in which BiLog itself is sufficient to express the computation. One of them is the encoding of CCS, shown in the following.

We focus on the fairly small fragment of CCS considered in [5], consisting of prefix and parallel composition only; P, Q will range over CCS *processes*; a, b, c over *actions*, chosen in the enumerable set *Acts*; and $\overline{a}, \overline{b}, \overline{c}$ over *coactions*. Process syntax is defined by the following grammar:

$$\begin{array}{lcl} P & ::= & \mathbf{0} \;\mid\; \lambda.P \;\mid\; P \mid P \\ \lambda & ::= & a \;\mid\; \overline{a} \end{array}$$

As operator ν is not included, all the actions appearing in a process are not bound; this fact yields the encoding to produce bigraphs with open links. Moreover, as *Acts* will actually be the set of names for the bigraphs used to encode CCS processes, we will refer to its elements as names. In particular, the 'names' of a CCS process are all the elements of *Acts* appearing in its syntax, both as actions and as coactions. For instance, the names in the process $a.\overline{c}.b.\overline{a}.\mathbf{0}$ are a, b, c.

The *structural congruence* $\equiv$ is defined as the least congruence on processes such that $P \mid \mathbf{0} \equiv P$, $P \mid Q \equiv Q \mid P$ and $P \mid (Q \mid R) \equiv (P \mid Q) \mid R$. Finally, the usual *reduction operational semantics* gives dynamics:

$$\frac{}{a.P \mid \overline{a}.Q \to P \mid Q} \qquad \frac{P \to Q}{P \mid R \to Q \mid R} \qquad \frac{P \equiv P' \quad P' \to Q' \quad Q' \equiv Q}{P \to Q} \tag{6}$$

The work [32] presents a bigraphical encoding for this CCS. The bigraphs suitable to encode CCS are built by two controls with arity 1: act for actions and coact for coactions. As mentioned above, every action $a \in Acts$ is treated as a name in the bigraphical model. The corresponding constructors assume the form act_a and coact_a. Reactions are intuitively expressed as

$$\mathsf{act}_a\square_1 \mid \mathsf{coact}_a\square_2 \longrightarrow a \mid \square_1 \mid \square_2. \tag{7}$$

Rules are parametric, in the sense that the two holes, $\square_1$ and $\square_2$, can be filled up by any process, and the link a is introduced to maintain the same interface between redex and reactum. By definition, redex can be replaced by the reactum in any bigraphical active context. As the active contexts are identified by the predicate δ, in this particular case such a predicate has to project CCS's active contexts into bigraphs. It is easy to see that rules in (6) imply that active CCS contexts have the form '$P \mid \Box$,' whose corresponding bigraphical context is '$[\![P]\!] \mid \Box$,' where $[\![P]\!]$ is the bigraphical encoding for P. Since Lemma 7 will prove that the encoding introduced in this section is bijective on bigraphs that are ground, *prime* (i.e., with a single root, as for the definition on place graphs) and with open links, the formal definition for an active bigraphical context is

$$g \mid \Box, \tag{8}$$

for $g : \epsilon \to \langle 1, Z \rangle$ ground, prime and with open links. Moreover, controls act and coact are declared to be *passive*, i.e., no reaction can occur inside them. It is straightforward to conclude that the most general context ready to react has the form '$\Box_0 \mid \mathsf{act}_a \Box_1 \mid \mathsf{coact}_a \Box_2$' and the most general reaction is

$$\Box_0 \mid \mathsf{act}_a \Box_1 \mid \mathsf{coact}_a \Box_2 \longrightarrow \Box_0 \mid a \mid \Box_1 \mid \Box_2, \tag{9}$$

where holes $\Box_0$, $\Box_1$ and $\Box_2$ has to be filled in by prime ground bigraphs with open links. Such a reduction turns out to be compositional with parallel operator.

The encoding maps CCS processes into ground, prime open linked bigraphs, and it is denoted by $[\![\ ]\!]_X$. Such an encoding is parameterised by a *finite* subset $X \subseteq Acts$; it yields ground bigraphs with outer face $\langle 1, X \rangle$ and open links. The value $[\![P]\!]_X$ is defined only the names in P belong to X:

$$\begin{aligned}
[\![\mathbf{0}]\!]_X &\overset{def}{=} 1 \otimes X \\
[\![a.P]\!]_X &\overset{def}{=} (\mathsf{act}_a \overset{a}{\otimes} id_X) \circ [\![P]\!]_X \\
[\![\bar{a}.P]\!]_X &\overset{def}{=} (\mathsf{coact}_a \overset{a}{\otimes} id_X) \circ [\![P]\!]_X \\
[\![P \mid Q]\!]_X &\overset{def}{=} (join \otimes id_X) \circ ([\![P]\!]_X \overset{X}{\otimes} [\![Q]\!]_X)
\end{aligned}$$

where $a \in X$, and the sharing/separation operator $\overset{X}{\otimes}$ stands for $\overset{\vec{a}}{\otimes}$ where $\vec{a}$ is any array of all the elements in X.

Note, in particular, that the sharing tensor '$\_ \overset{a}{\otimes} id_X$' enables the definition to be compositional, as the outer face is $\langle 1, X \rangle$ for every encoding. Moreover, such a sharing tensor allows the process filling the hole in act_a (and coact_a) to perform other a actions. In fact, consider the simple CCS process $a.\bar{a}.\mathbf{0}$, then $[\![a.\bar{a}.\mathbf{0}]\!]_{\{a\}}$ is $(\mathsf{act}_a \overset{a}{\otimes} id_{\{a\}}) \circ (\mathsf{coact}_a \overset{a}{\otimes} id_{\{a\}}) \circ (1 \otimes a)$. Clearly, the composition is granted by the sharing operator.

In the encoding for parallel, operator *join* makes tensor commutative. There is a straight correspondence between parallel operators in the two calculi, as $[\![P \mid Q]\!]_X$ corresponds to $[\![P]\!]_X \mid [\![Q]\!]_X$, that is the parallel operator on bigraphs, defined in [30]. The result stated in Lemma 7 says that the encoding is bijective on prime ground bigraphs with open links. First, Lemma 6 provides a general result on bigraphs and parallel composition. It says that to add names that already appear in a bigraph dos not alter the bigraph itself.

Lemma 6 (Adding Names) *If x is in the outer names of G, then $G \mid x \equiv G$.*

Proof. Express the parallel in terms of renamings, linkings and tensor product as in [30], and use axioms of [30]. Assume $G : \langle m, X\rangle \to \langle n, \{x\} \cup Y\rangle$, with $y \notin \{x\} \cup Y$. Then $G \mid x$ corresponds to $(id_{\langle n,Y\rangle} \otimes (x \leftleftarrows y)) \circ (G \otimes ((y \leftarrow x) \circ x))$, that is $(id_{\langle n,Y\rangle} \otimes (x \leftleftarrows y)) \circ (G \otimes y)$ by the third link axiom. By bifunctoriality property, this is congruent to $(id_{\langle n,Y\rangle} \otimes (x \leftleftarrows y)) \circ (id_{\langle n,Y\rangle} \otimes id_x \otimes y) \circ (G \otimes id_\epsilon)$, and again to $((id_{\langle n,Y\rangle} \circ id_{\langle n,Y\rangle}) \otimes ((x \leftleftarrows y) \circ (id_x \otimes y))) \circ G$. The latter is congruent to $(id_{\langle n,Y\rangle} \otimes id_x) \circ G$, by the second link axiom. Since $(id_{\langle n,Y\rangle} \otimes id_x) \circ G \equiv G$, conclude the thesis. $\square$

Lemma 6 is useful to prove that the encoding is bijective on ground prime bigraphs with open links.

Lemma 7 (Bijective Translation) *For every finite subset $X \subseteq Acts$:*

(1) The translation $[\![\cdot]\!]_X$ is surjective on prime ground bigraphs with outerface $\langle 1, X\rangle$ and open links.
(2) For every couple of processes P, Q and for every finite subset $X \subseteq Acts$ containing all the names in P and Q, it holds: $P \equiv Q$ iff $[\![P]\!]_X \equiv [\![Q]\!]_X$.

Proof. Prove point (1) by showing that every prime ground bigraph with outerface $\langle 1, X\rangle$ has at least one pre-image for the translation $[\![\cdot]\!]_X$. Proceed by induction on the number of nodes in bigraphs. The Connected Normal Form (CNF) for bigraphs presented in [30] simplifies the proof. According to [30], every prime ground bigraph G with outerface $\langle 1, X\rangle$ and open links has the following connected normal form: $G ::= X \mid F$, where $F ::= M_1 \mid \dots \mid M_k$, with $M ::= (\mathsf{K}_a \mid id_Y) \circ F$ for $a \in Acts$ and $\mathsf{K}_a \in \{\mathsf{act}_a, \mathsf{coact}_a\}$. In particular, a term M is a *ground molecule*.

The base of induction is X, intended as a bigraph, and clearly $[\![\mathbf{0}]\!]_X = X$. For the inductive step, consider a bigraph G with at least one node. This means $G = X \mid ((\mathsf{K}_a \mid id_Y) \circ F) \mid G'$. Without losing generality, assume $\mathsf{K}_a = \mathsf{act}_a$, so $G = ((\mathsf{act}_a \mid id_X) \circ (X \mid F)) \mid (X \mid G')$ by Lemma 6. Now, the induction says that there exist P and Q such that $[\![P]\!]_X = X \mid F$ and $[\![Q]\!]_X = X \mid G'$, hence conclude $[\![a.P \mid Q]\!]_X = G$.

The forward implication of point (2) is proved by showing that the translation is sound with respect to the rules of congruence in CCS. This has been already proved in [30], where the parallel operator between bigraphs is shown to be commutative and associative, and to have 1 as a unit. Moreover, by Lemma 6, the bigraph $1 \otimes X$ is the unit for the parallel operator on prime ground bigraphs with outerface $\langle 1, X\rangle$.

The following claim, stated in [32], is the crucial step in proving the reverse implication of point (2). Its proof considers the connected normal form for bigraphs.

Claim 1 *If G_i ($i = 1 \dots m$) and F_j ($j = 1 \dots n$) are ground molecules and $G_1 \mid \dots \mid G_m \equiv F_1 \mid \dots \mid F_n$, then $m = n$ and $G_i \equiv F_{\pi(i)}$ for some permutation π on m.*

The proof of the reverse implication of point (2) proceeds by induction on the structure of the CCS process P. The base of induction is $P = \mathbf{0}$, in this case the statement is verified since $[\![Q]\!]_X \equiv [\![\mathbf{0}]\!]_X = X$ implies $Q \equiv \mathbf{0} \mid \dots \mid \mathbf{0}$. For the inductive step, let $P \equiv a_1.P_1 \mid \dots \mid a_m.P_m$ for any $m \geq 1$, and assume $[\![Q]\!] \equiv [\![P]\!]$. Furthermore we have $Q \equiv b_1.Q_1 \mid \dots \mid b_n.Q_n$, then

$$\begin{aligned}
[\![P]\!]_X &= (\mathsf{act}_{a_1} \overset{a_1}{\otimes} id_X) \circ [\![P_1]\!]_X \mid \dots \mid (\mathsf{act}_{a_m} \overset{a_m}{\otimes} id_X) \circ [\![P_m]\!]_X \\
[\![Q]\!]_X &= (\mathsf{act}_{b_1} \overset{b_1}{\otimes} id_X) \circ [\![Q_1]\!]_X \mid \dots \mid (\mathsf{act}_{b_m} \overset{b_m}{\otimes} id_X) \circ [\![Q_m]\!]_X
\end{aligned}$$

Since the two translations are both a parallel compositions of ground molecules, the previous claim says that $m = n$, and there exists a permutation π on m such that $a_i \equiv a_{\pi(i)}$ and $[\![Q_i]\!] \equiv [\![P_{\pi(i)}]\!]$. By induction $Q_i \equiv P_{\pi(i)}$, hence $Q \equiv P$. $\square$

Paper [32] proves that the translation preserves and reflects the reactions, namely: $P \longrightarrow P'$ *if and only if* $[\![P]\!]_X \longrightarrow [\![P']\!]_X$. A similar result is obtained in this case.

In the current bigraphical system, reaction rules are defined as $(\mathsf{act}_a \mid id_{Y_1}) \mid (\mathsf{coact}_a \mid id_{Y_2}) \longrightarrow a \mid id_{\langle 1, Y_1 \rangle} \mid id_{\langle 1, Y_2 \rangle}$. It is easy to see that this can be mildly sugared to obtain the rule introduced in (7). Moreover, the active contexts introduced in (8) can be specialised as $g \mid (id_1 \otimes id_Y)$, for $g : \epsilon \to \langle 1, Z \rangle$ ground, prime and with open links. Moreover, Y, Y_1 and Y_2 must be finite sets of names, viz., the outer names of the term that can fill the contexts. Finally, the general reaction (9) is specialised as

$$(id_1 \otimes id_Y) \mid (\mathsf{act}_a \mid id_{Y_1}) \mid (\mathsf{coact}_a \mid id_{Y_2}) \longrightarrow (id_1 \otimes id_Y) \mid a \mid id_{Y_1} \mid id_{Y_2}. \qquad (10)$$

When a reacting (ground) bigraph is a CCS encoding, such as $[\![P]\!]_X$, it can actually be decomposed into a redex, essentially the one in the left-hand side of (10), and a ground bigraph with a well defined structure, essentially with three regions. The composition of such a bigraph with the corresponding reactum, essentially the one in the right-hand side of (10), gives the result of the reaction. Lemma8 expresses such a characterisation. Redex and Reactum are formally outlined in Tab. 7.1. They complex structure is due to the fact that tensor product is defined only disjoint names, and this is guaranteed by renamings. To better understand the table, it is worth to reintroduce some syntactic sugar, as in (9). According to such a notation, $Redex_a^{y_1, y_2, Y_1, Y_2}$ and $React_a^{Y_1, Y_2}$ are simply $\square_0 \mid \mathsf{act}_a \square_1 \mid \mathsf{coact}_a \square_2$ and $\square_0 \mid \square_1 \mid \square_2$, where the sets of names X, Y_1, Y_2 are respectively associated to the holes $\square_0, \square_1, \square_2$ and they must be disjoint to allow the tensor product. Names y_1 and y_2 are useful to join the action with the corresponding coaction, they must be disjoint with X, Y_1 and Y_2. Wirings W, W' and *join* operators assure that the outerfaces are $\langle 1, X \rangle$.

Table 7.1. *Reacting Contexts for CCS encodings*

Bigraphs:

$$Redex_a^{y_1,y_2,Y_1,Y_2} \stackrel{def}{=} W \circ (id_Y \otimes join) \circ (id_Y \otimes join \otimes id_1) \circ \{((y_1 \leftarrow a) \otimes id_1) \circ$$
$$\circ\ \mathsf{act}_a \otimes id_{Y_1} \otimes ((y_2 \leftarrow a) \otimes id_1) \circ \mathsf{coact}_a \otimes id_{Y_2} \otimes id_{\langle 1,X\rangle}\}$$
$$React_a^{Y_1,Y_2} \stackrel{def}{=} W' \circ (id_{Y'} \otimes join) \circ (id_{Y'} \otimes join \otimes id_1)$$

Wirings:

$$W \stackrel{def}{=} ((X \Lleftarrow Y_1) \otimes id_1) \circ (id_{Y_1} \otimes (X \Lleftarrow Y_2) \otimes id_1) \circ (id_{Y_1} \otimes id_{Y_2} \otimes id_{X\setminus\{a\}} \otimes$$
$$\otimes (a \Lleftarrow y_1) \otimes id_1) \circ (id_{Y_1} \otimes id_{Y_2} \otimes id_{X\setminus\{a\}} \otimes id_{\{y_1\}} \otimes (a \Lleftarrow y_2) \otimes id_1)$$
$$W' \stackrel{def}{=} ((X \Lleftarrow Y_1) \otimes id_1) \circ (id_{Y_1} \otimes (X \Lleftarrow Y_2) \otimes id_1)$$

Supporting Sets:

$$Y \stackrel{def}{=} \{y_1, y_2\} \cup Y_1 \cup Y_2 \cup X$$
$$Y' \stackrel{def}{=} Y_1 \cup Y_2 \cup X$$

Lemma 8 (Reducibility) *For every CCS process P, the following are equivalent.*

(1) The translation $[\![P]\!]_X$ can perform the reduction $[\![P]\!]_X \longrightarrow G$.
(2) There exist bigraphs $G_1, G_2, G_3 : \epsilon \to \langle 1, X\rangle$ and name $a \in X$, such that $[\![P]\!]_X \equiv ((\mathsf{act}_a \mid id_X) \circ G_1) \mid ((\mathsf{coact}_a \mid id_X) \circ G_2) \mid G_3$ and $G \equiv G_1 \mid G_2 \mid G_3$.
(3) There exist actions $a \in X$ and $y_1, y_2 \notin X$, and two mutually disjoint subsets $Y_1, Y_2 \subseteq Acts$ with the same cardinality as X, but disjoint with X, y_1, y_2, and there exist the bigraphs $H_1 : \epsilon \to \langle 1, Y_1\rangle$, $H_2 : \epsilon \to \langle 1, Y_2\rangle$, and $H_3 : \epsilon \to \langle 1, X\rangle$ with open links, such that $[\![P]\!]_X \equiv Redex_a^{y_1,y_2,Y_1,Y_2} \circ (H_1 \otimes H_2 \otimes H_3)$ and $G \equiv React_a^{Y_1,Y_2} \circ (H_1 \otimes H_2 \otimes H_3)$, where $Redex_a^{y_1,y_2,Y_1,Y_2}$, $React_a^{Y_1,Y_2}$ are defined in Tab. 7.1.

Proof. First prove that points (1) and (2) are equivalent. Assume that the bigraph $[\![P]\!]_X$ can perform a reaction. This means that $[\![P]\!]_X \equiv ((\mathsf{act}_a \mid id_{Y_1}) \circ G'_1) \mid ((\mathsf{coact}_a \mid id_{Y_2}) \circ G'_2) \mid G'_3$ and that $G \equiv a \mid G'_1 \mid G'_2 \mid G'_3$ for some suitable ground bigraphs G'_1, G'_2 and G'_3 and action $a \in X$. Since the type of both $[\![P]\!]_X$ and G is $\epsilon \to \langle 1, X\rangle$, Lemma 6 says that $G \equiv (X \mid G'_1) \mid (X \mid G'_2) \mid (X \mid G'_3)$ and $[\![P]\!]_X \equiv ((\mathsf{act}_a \mid id_X) \circ (X \mid G'_1)) \mid ((\mathsf{coact}_a \mid id_X) \circ (X \mid G'_2)) \mid (X \mid G'_3)$. Then define G_i to be $X \mid G'_i$ for $i = 1, 2, 3$, and conclude that $G \equiv G_1 \mid G_2 \mid G_3$ and $[\![P]\!]_X \equiv ((\mathsf{act}_a \mid id_X) \circ G_1) \mid ((\mathsf{coact}_a \mid id_X) \circ G_2) \mid G_3$.

Then prove that point (2) implies point (3). Assume that $[\![P]\!]_X \equiv ((\mathsf{act}_a \mid id_X) \circ G_1) \mid ((\mathsf{coact}_a \mid id_X) \circ G_2) \mid G_3$ and $G \equiv G_1 \mid G_2 \mid G_3$, with $G_1, G_2, G_3 : \epsilon \to \langle 1, X\rangle$. Chose two actions $y_1, y_2 \notin X$ and two mutually disjoint subsets $Y_1, Y_2 \subseteq Acts$ with the same cardinality as X, but disjoint with X, y_1, y_2, and follow the definition of parallel operator in [30] to obtain

$$[\![P]\!]_X \equiv W \circ (id_Y \otimes join) \circ (id_Y \otimes join \otimes id_1) \circ \{((y_1 \leftarrow a) \otimes$$
$$\otimes\ id_{\langle 1,Y_1\rangle}) \circ (\mathsf{act}_a \otimes id_{Y_1}) \circ ((Y_1 \leftarrow X) \otimes id_{\langle 1,Y_2\rangle}) \circ G_1 \otimes ((y_2 \leftarrow a) \otimes$$
$$\otimes\ id_1) \circ (\mathsf{coact}_a \otimes id_{Y_2}) \circ ((Y_2 \leftarrow X) \otimes id_1) \circ G_2 \otimes G_3\}$$

and

$$G \equiv W' \circ (id_{Y'} \otimes join) \circ (id_{Y'} \otimes join \otimes id_1) \circ \\ \circ \{((Y_1 \leftarrow X) \otimes id_{\langle 1,Y_2 \rangle}) \circ G_1 \otimes ((Y_2 \leftarrow X) \otimes id_1) \circ G_2 \otimes G_3\}$$

where $Y = \{y_1\} \cup Y_1 \cup \{y_2\} \cup Y_2 \cup X$ and $Y' = Y_1 \cup Y_2 \cup X$. The bigraphs W and W' are defined in Tab. 7.1, they both link the subsets Y_1 and Y_2 with X, and moreover W links y_1 and y_2 with a. By bifunctoriality property, $[\![P]\!]_X$ is rewritten as

$$W \circ (id_Y \otimes join) \circ (id_Y \otimes join \otimes id_1) \circ \{((y_1 \leftarrow a) \otimes id_1) \circ \\ \circ \mathsf{act}_a \otimes id_{Y_1} \otimes ((y_2 \leftarrow a) \otimes id_1) \circ \mathsf{coact}_a \otimes id_{Y_2} \otimes G_3 \} \circ \\ \circ \{ ((Y_1 \leftarrow X) \otimes id_1) \circ G_1 \otimes ((Y_2 \leftarrow X) \otimes id_1) \circ G_2 \},$$

and, again by bifunctoriality property, as

$$W \circ (id_Y \otimes join) \circ (id_Y \otimes join \otimes id_1) \circ \{((y_1 \leftarrow a) \otimes id_1) \circ \\ \circ \mathsf{act}_a \otimes id_{Y_1} \otimes ((y_2 \leftarrow a) \otimes id_1) \circ \mathsf{coact}_a \otimes id_{Y_2} \otimes id_{\langle 1,X \rangle} \} \circ \\ \circ \{ ((Y_1 \leftarrow X) \otimes id_1) \circ G_1 \otimes ((Y_2 \leftarrow X) \otimes id_1) \circ G_2 \otimes G_3 \}.$$

Point (3) follows by defining $H_i' = ((Y_i \leftarrow X) \otimes id_1) \circ G_i$ for $i = 1, 2$, and $H_3 = G_3$. Note that the three bigraphs G_i and H_i have open links as so does $[\![P]\!]_X$. Finally, point (3) implies point (2), by inverting previous reasoning. □

By following the ideas of [32] it is easy to demonstrate that there is an exact match between the reactions generated in CCS and in the bigraphical system. This a consequence of the fact that CCS reacting contexts are clearly identified and easily transferred in bigraphical settings.

Proposition 3 (Matching Reactions) *For every finite set X, that contains all the names appearing in P and Q, it holds: $P \to Q$ if and only if $[\![P]\!]_X \longrightarrow [\![Q]\!]_X$.*

Proof. For the forward direction, proceed by induction on the number of the rules applied in the derivation for $P \to Q$ in CCS. The base of the induction is the only rule without premixes, meaning that P is $a.P_1 \mid \overline{a}.P_2$ and Q is $P_1 \mid P_2$. The translation is sound as regards this rule, since the reactive system says

$$((\mathsf{act}_a \mid id_X) \circ [\![P_1]\!]_X) \mid ((\mathsf{coact}_a \mid id_X) \circ [\![P_2]\!]_X) \longrightarrow X \mid [\![P_1]\!]_X \mid [\![P_2]\!]_X.$$

The induction step considers two cases. First, assume that $P \to Q$ is derived from $P' \to Q'$, where P is $P' \mid R$ and Q is $Q' \mid R$. Then the induction hypothesis says that $[\![P']\!]_X \longrightarrow [\![Q']\!]_X$, hence $[\![P']\!]_X \mid [\![R]\!]_X \longrightarrow [\![Q']\!]_X \mid [\![R]\!]_X$. Conclude that $[\![P]\!]_X \longrightarrow [\![Q]\!]_X$, as $[\![P]\!]_X$ is $[\![P']\!]_X \mid [\![R]\!]_X$ and $[\![Q]\!]_X$ is $[\![Q']\!]_X \mid [\![R]\!]_X$. Second, assume that $P \to Q$ is derived from the congruences $P \equiv P'$ and $Q' \equiv Q$, and from the transition $P' \to Q'$. By Lemma 7, $[\![P]\!]_X \equiv [\![P']\!]_X$ and $[\![Q']\!]_X \equiv [\![Q]\!]_X$, and, by induction hypothesis, $[\![P']\!]_X \longrightarrow [\![Q']\!]_X$. Conclude $[\![P]\!]_X \longrightarrow [\![Q]\!]_X$, since the reduction is defined up to congruence.

Table 7.2. *Semantics of formulae $\mathcal{L}_{spat}$ in CCS*

$P \models_{spat}$	0	if $P \equiv \mathbf{0}$
$P \models_{spat}$	$\neg A$	if not $P \models_{spat} A$
$P \models_{spat}$	$A \wedge B$	if $P \models_{spat} A$ and $P \models_{spat} B$
$P \models_{spat}$	$A \mid B$	if there exist R, Q, s.t. $P \equiv R \mid Q$, $R \models_{spat} A$ and $Q \models B_{spat}$
$P \models_{spat}$	$A \triangleright B$	if for every Q, $Q \models_{spat} A$ implies $P \mid Q \models_{spat} B$
$P \models_{spat}$	$\Diamond A$	if there exist P' s.t. $P \longrightarrow P'$ and $P' \models_{spat} A$

For the reverse implication, assume $[\![P]\!]_X \longrightarrow [\![Q]\!]_X$. Lemma 8 says that there exist the bigraphs $G_1, G_2, G_3 : \epsilon \to \langle 1, X\rangle$ and the name $a \in X$ such that $[\![P]\!]_X \equiv ((\mathsf{act}_a \mid id_X) \circ G_1) \mid ((\mathsf{coact}_a \mid id_X) \circ G_1) \mid G_3$ and $G \equiv G_1 \otimes G_2 \otimes G_3$. Now, Lemma 7 says that for every $i = 1, 2, 3$ there exists a CCS process P_i such that $[\![P_i]\!]$ corresponds to G_i, hence $[\![P]\!] \equiv [\![a.P_1 \mid \overline{a}.P_2 \mid P_3]\!]$ and $[\![Q]\!] \equiv [\![P_1 \mid P_2 \mid P_3]\!]$. Again, Lemma 7 says that $P \equiv a.P_1 \mid \overline{a}.P_2 \mid P_3$ and $Q \equiv P_1 \mid P_2 \mid P_3$, then $P \to Q$. □

Tanks to Lemma 7, the previous result can be further specialised: whenever a bigraphical encoding reacts, so does the corresponding CCS process.

Proposition 4 (Conservative Reaction) *If $[\![P]\!]_X \longrightarrow G$ for a CCS process P, then there exists a CCS process Q such that $[\![Q]\!]_X = G$ and $P \to Q$.*

Proof. Assume $[\![P]\!]_X \longrightarrow G$, then point (2) of Lemma 8 says that G has type $\epsilon \to \langle 1, X\rangle$ and open links, as so does $[\![P]\!]_X$. Lemma 7 says that there exists a process Q such that $[\![Q]\!]_X \equiv G$. Conclude $P \to Q$ by Lemma 3. □

Paper [5] introduces $\mathcal{L}_{spat}$, a spatial logic suitable to describe structure and behaviour of CCS processes. Such a logic is based on the language $A, B ::= 0 \mid A \wedge B \mid A|B \mid \neg A \mid A \triangleright B \mid \Diamond A$. It includes the void constant 0 and the basic spatial operators: composition $|$, and its adjunct $\triangleright$. It presents also a temporal operator, next step modality $\Diamond$, to capture process dynamics. Table. 7.2 outlines the semantics of $\mathcal{L}_{spat}$ in term of CCS processes, as defines in [5]. In particular, parallel connective describes processes that are the parallel composition between two processes that satisfies the corresponding formulae. A process satisfies $A \triangleleft B$ if it satisfies the formula B whenever put in parallel with any process satisfying A. Finally, next step $\Diamond A$ is satisfied by a process that can evolve into a process satisfying A.

The logic $\mathcal{L}_{spat}$ can be encoded in a suitable instantiation of BiLog, without using the modality defined in (5), but exploiting BiLog expressivity, suitable to characterise reacting contexts. It is sufficient to instantiate the logic BiLog$(M, \otimes, \epsilon, \Theta, \equiv, \tau)$ to obtain the bigraphical encoding of CCS. We define Θ to be composed by the standard constructor for a bigraphical system with $\mathcal{K} = \{\mathsf{act}, \mathsf{coact}\}$. Moreover, transparency predicate τ must be always true. This fact is determinant for the soundness of the logical encoding, as it enables BiLog to fully describe any bigraphical term, and, therefore, to detect all reacting contexts by simply analysing their ‘spatial’

Table 7.3. *Encoding of $\mathcal{L}_{spat}$ into BiLog*

Encodings:

$$\llbracket 0 \rrbracket_X \stackrel{def}{=} X \otimes \mathbf{1}$$
$$\llbracket \neg A \rrbracket_X \stackrel{def}{=} \neg \llbracket A \rrbracket_X$$
$$\llbracket A \wedge B \rrbracket_X \stackrel{def}{=} \llbracket A \rrbracket_X \wedge \llbracket B \rrbracket_X$$
$$\llbracket A \mid B \rrbracket_X \stackrel{def}{=} \mathbf{join} \circ (\llbracket A \rrbracket_X \overset{X}{\otimes} \llbracket B \rrbracket_X)$$
$$\llbracket A \triangleright B \rrbracket_X \stackrel{def}{=} \text{И}Y. (((Y \leftarrow X) \otimes \mathbf{id}_1) \circ \mathbf{A}_X) \multimap\!\otimes (\mathbf{join} \circ ((X \Lleftarrow Y) \otimes \mathbf{id}_1) \circ\!- \llbracket B \rrbracket_X)$$
$$\llbracket \Diamond A \rrbracket_X \stackrel{def}{=} \bigvee_{a \in X} \text{И}y_1.y_2.Y_1.Y_2.\ \mathbf{Redex}_a^{y_1,y_2,Y_1,Y_2} \circ [(\mathbf{React}_a^{Y_1,Y_2} \circ\!- \llbracket A \rrbracket_X) \wedge \mathbf{Triple}]$$

Supporting Formulae:

$$\mathbf{Open} \stackrel{def}{=} \neg \text{И}x. \diamond(/x \circ \mathbf{T})$$
$$\mathbf{A}_X \stackrel{def}{=} \llbracket A \rrbracket_X \wedge \mathbf{T}_{\epsilon \to \langle 1, Y_2 \rangle} \wedge \mathbf{Open}$$
$$\mathbf{Triple} \stackrel{def}{=} \mathbf{T}_{\epsilon \to \langle 1, Y_1 \rangle} \otimes \mathbf{T}_{\epsilon \to \langle 1, Y_2 \rangle} \otimes \mathbf{T}_{\epsilon \to \langle 1, X \rangle}$$

structure.

Lemma 8 is informally rephrased by saying that reactions for encoded CCS processes are determined by couples of the form ($Redex_a, Reactum_a$), cf. Tab. 7.1, and every reacting process is characterised by

$$\llbracket P \rrbracket_X \longrightarrow\!\!\triangleright \llbracket Q \rrbracket_X \textit{ iff there exists a bigraph } g \textit{ and } a \in X \textit{ such that}$$
$$\llbracket P \rrbracket_X \equiv Redex_a \circ g \textit{ and } \llbracket Q \rrbracket_X \equiv Reactum_a \circ g.$$

Since τ is always true, it is possible to define a characteristic formula for every redex and reactum, simply by rewriting every bigraphical constructor and operator with the correspondent logical constant in their bigraphical encodings. For the new names y_1, y_2, and the new subsets Y_1, Y_2, denote with $\mathbf{Redex}_a^{y_1,y_2,Y_1,Y_2}$ and $\mathbf{React}_a^{Y_1,Y_2}$ the characteristic formulae for $Redex_a^{y_1,y_2,Y_1,Y_2}$ and $React_a^{Y_1,Y_2}$, respectively. Clearly, $G \models \mathbf{Redex}_a^{y_1,y_2,Y_1,Y_2}$ if and only if $G \equiv Redex_a^{y_1,y_2,Y_1,Y_2}$, and the same for reactum. This has a prominent role in defining the encoding of the temporal modality in BiLog.

Table 7.3 formally defines logical encoding, that is parameterised on the set X of names, as so does the process encoding. The encodings for logical connectives and spatial composition are self-explanatory. In particular, spatial composition requires the sharing of all the names in X: it corresponds to the logical parallel operator when the set of bigraph names is fixed and finite, as happens for processes encoded by $\llbracket\ \rrbracket_X$. The encoding for $\triangleright$ introduces an auxiliary notation. Intuitively, formula $\mathbf{A}_X$ is defined to constrain a bigraph to be the encoding of a CCS process and to satisfy $\llbracket A \rrbracket_X$. In fact, $G \models \mathbf{A}_X$ means that G satisfies $\llbracket A \rrbracket_X$, it has type $\epsilon \to \langle 1, X \rangle$ and its links are open, as a bigraph satisfies **Open** only if no closure appears in any of its decompositions. Proposition 5 will show that a bigraph satisfies $\llbracket P \rrbracket_X \models \llbracket A \triangleright B \rrbracket_X$ if it satisfies $\llbracket B \rrbracket_X$ whenever connected in parallel with any encoding of a CCS process satisfying $\llbracket A \rrbracket_X$.

In the encoding for the temporal modality $\Diamond$, the supporting formula **Triple** is satisfied by processes that are the composition of three single-rooted ground bigraphs whose outerfaces have the same number of names as X. Proposition 5 will show that a process satisfies $[\![\Diamond A]\!]_X$ if and only if it is the combination between a particular redex and a bigraph that satisfies the requirement of Lemma 8, and moreover that the corresponding reactum satisfies $[\![A]\!]_X$.

Proposition 5 formalises the main result of the. It expresses the semantical equivalence between $\mathcal{L}_{spat}$ and its encoding in BiLog, note, in particular, the requirement for a finite set of actions performable by the CCS processes. Such a limitation is not due to the presence of the next step operator. Indeed, inspecting the proof, one can see that the induction step for the temporal operator still holds in the case of a not-finite set of actions. The limitation, in fact, is due to the adjoint operator $\triangleright$: the number of names shared between the processes must be bound. This happens because of the different choice for the logical product operator in BiLog. On one hand, spatial logic has parallel operator built in. This means that the logic does not care about the names that are actually shared between the processes. On the other hand, BiLog has a strong control on the names shared between two processes, and they must be known with accuracy.

Proposition 5 *If the set of names in every CCS process is bounded to be a finite set X, then $P \models_{spat} A$ if and only if $[\![P]\!]_X \models [\![A]\!]_X$.*

Proof. Proceed by induction on formula structure. Base of induction is formula 0. To assume $[\![P]\!]_X \models [\![0]\!]_X$ means $[\![P]\!]_X \equiv X \otimes 1$, that correspond to $P \equiv \mathbf{0}$, hence $P \models_{spat} 0$ by definition.

Inductive step deals with connectives. Treatments of $\neg$, $\wedge$ and $|$ are similar; hence focus on parallel operator.

Case $A \mid B$. To say $[\![P]\!]_X \models [\![A \mid B]\!]_X$ means that there are two bigraphs g_1, g_2, with $g_1 \models [\![A]\!]_X$ and $g_1 \models [\![B]\!]_X$, such that $[\![P]\!]_X \equiv join \circ (g_1 \overset{X}{\otimes} g_2)$. The bigraphs g_1, g_2 must have type $\epsilon \to \langle 1, X \rangle$ and open links, as so does $[\![P]\!]_X$. By Lemma 7, there are two processes Q_1 and Q_2 such that $[\![Q_1]\!]_X$ and $[\![Q_2]\!]_X$ are g_1 and g_2, respectively. Then conclude $[\![P]\!]_X \equiv join \circ ([\![Q_1]\!]_X \overset{X}{\otimes} [\![Q_2]\!]_X)$, that means $P \equiv Q_1 \mid Q_2$, again by Lemma 7. Moreover, induction hypothesis says that $Q_1 \models A$ and $Q_2 \models B$, hence $P \models_{spat} A \mid B$.

Case $A \triangleright B$. Assume $[\![P]\!]_X \models [\![A \triangleright B]\!]_X$, then by definition there exists a fresh set Y of actions such that for every G satisfying $(((Y \leftarrow X) \otimes \mathbf{id}_1) \circ \mathbf{A}_X)$ it holds $[\![P]\!]_X \otimes G \models \mathbf{join} \circ ((X \Leftarrow Y) \otimes \mathbf{id}_1) \multimap [\![B]\!]_X$, that is

$$join \circ ((X \Leftarrow Y) \otimes id_1) \circ ([\![P]\!]_X \otimes G) \models [\![B]\!]_X \tag{11}$$

Now $G \models (((Y \leftarrow X) \otimes \mathbf{id}_1) \circ \mathbf{A}_X)$ means that there is $g \models \mathbf{A}_X$ such that $G \equiv$

$((Y \leftarrow X) \otimes \mathbf{id}_1) \circ g$. As previously discussed (cf. the introduction to the current proposition) $g \models \mathbf{A}_X$ says that $g \models [\![A]\!]_X$ and that g is a bigraph with open link and type $\epsilon \to \langle 1, X\rangle$. By Lemma 7, g is $[\![Q]\!]_X$ for some CCS process Q whose actions are in X.

Hence, as the set of actions *Acts* corresponds to X, (11) is rephrased by saying that for *every* CCS process Q such that $[\![Q]\!]_X \models [\![A]\!]_X$ it holds

$$join \circ ((X \leftleftarrows Y) \otimes id_1) \circ ([\![P]\!]_X \otimes ((Y \leftarrow X) \otimes id_1) \circ [\![Q]\!]_X) \models [\![B]\!]_X$$

that is $[\![P \mid Q]\!]_X \models [\![B]\!]_X$. Then, the induction hypothesis says that for every Q, if $Q \models_{spat} A$ then $P \mid Q \models_{spat} B$, namely $P \models_{spat} A \triangleright B$.

Case $\Diamond A$. to assume $[\![P]\!]_X \models [\![\Diamond A]\!]_X$ signifies that there exists an action $a \in X$ such that

$$[\![P]\!]_X \equiv Redex_a^{y_1,y_2,Y_1,Y_2} \circ H \tag{12}$$

where y_1, y_2 are fresh names, Y_1, Y_2 are fresh subsets with the same cardinality as X, and H is a bigraph satisfying

$$H \models (\mathbf{React}_a^{Y_1,Y_2} \multimap [\![A]\!]_X) \wedge \mathbf{Triple}. \tag{13}$$

In particular, Property (13) amounts to assert the two following points.

(1) $H \models \mathbf{React}_a^{Y_1,Y_2} \multimap [\![A]\!]_X$, that means

$$React_a^{Y_1,Y_2} \circ H \models [\![A]\!]_X. \tag{14}$$

(2) $H \models \mathbf{T}_{\epsilon\to\langle 1,Y_1\rangle} \otimes \mathbf{T}_{\epsilon\to\langle 1,Y_2\rangle} \otimes \mathbf{T}_{\epsilon\to\langle 1,X\rangle}$, that means

$$H \equiv H_1 \otimes H_2 \otimes H_3 \tag{15}$$

with $H_i : \epsilon \to \langle 1, Y_i\rangle$, for $i = 1, 2$, and $H_3 : \epsilon \to \langle 1, X\rangle$.

Now $[\![P]\!]_X \equiv Redex^{y_1,y_2,Y_1,Y_2} \circ (H_1 \otimes H_2 \otimes H_3)$, by (12) and (15). This means $[\![P]\!]_X \longrightarrow React_a^{Y_1,Y_2} \circ (H_1 \otimes H_2 \otimes H_3)$, by Lemma 8. Furthermore, the bigraphs H_1, H_2, H_3 have open links, as so does $[\![P]\!]_X$. Hence Lemma 7 says that there exists the CCS process Q such that $[\![Q]\!]_X$ corresponds to $React_a^{Y_1,Y_2} \circ (H_1 \otimes H_2 \otimes H_3)$, hence $P \to Q$ by Proposition 3. Finally, (14) says that $[\![Q]\!]_X \models [\![A]\!]_X$, and this means $Q \models_{spat} A$ by induction hypothesis. Conclude that $[\![P]\!]_X \models [\![\Diamond A]\!]_X$ is equivalent to $P \to Q$ with $Q \models_{spat} A$, namely $P \models_{spat} \Diamond A$. □

The main steps in encoding CCS spatial logic into BiLog have been to encode the underlying calculus into bigraphical settings, to find the right reaction rules and, and then to characterise the corresponding reactive contexts by BiLog formulae. This hints how it may be possible to extend such a result to other calculi, such as π and ambients by employing their encodings, already provided in [25,26].

8 Conclusions and future work

This paper moves a first step towards describing global resources by focusing on bigraphs. Our final objective is to design a general dynamic logic able to cope uniformly with all the models bigraphs have been proved useful for, as of today these include λ-calculus [31], Petri-nets [29], CCS [32], pi-calculus [25], ambient calculus [26], and context-aware systems [3]. We introduced BiLog, a logic founded on bigraphs, whose formulae describe arrows in monoidal categories.

BiLog may at first appear complex and over-provided of connectives. On the contrary, the backbone of the logic is relatively simple, consisting of two operators ($\otimes$ and $\otimes$) regulated by elementary monoidal and interchange laws. Such a structure gives then rise to many – occasionally complex – derived connectives. This is a fundamental expressiveness property that does not put us off: BiLog is in fact meant to be a comprehensive meta-level framework in which several different logics can be isolated, understood and compared.

In particular, here we have seen how the 'separation' plays in various fragments of the logic. For instance, in the case of *Place Graph Logic*, where models are bigraphs without names, the separation is purely structural and coincides with the notion of parallel composition in Spatial Tree Logic. Dually, as the models for *Link Graph Logic* are bigraphs with no location, the separation in such a logic is disjointness of nominal resources. Finally, for *Bigraph Logic*, where nodes of the model are associated with names, the separation is not only structural, but also nominal, since the constraints on composition force port identifiers to be disjoint. In this sense, it can be seen as the separation in memory structures with pointers, like Separation Logic's heap structures [34], and trees with either pointers [7] or hidden names [11].

In §6 we sketched the application of BiLog to describe XML data, and we plan to extend the logic to more sophisticated semistructured data models. The similarities between XML and bigraphs have been pointed out independently also in [23] where XML is proposed as a language to codify bigraphs. In §6 we have focused on the other way around, by considering 'bigraphs as models for XML'.

In §7 we showed how BiLog can deal with dynamics. A natural solution is adding a temporal modality basically describing bigraphs that can compute according to a Bigraphical Reactive System [25]. When the transparency predicate enables the inspection of 'dynamic' controls, BiLog is '*intensional*' in the sense of [36], as it can observe internal structures. In the case of the bigraphical system describing CCS [32], BiLog can be so intensional that its static fragment directly expresses a temporal modality. A transparency predicate specifies which structures can be directly observed by the logic, while a temporal modality, along with the spatial connectives, allows to deduce the structure by observing the behaviour. It would be interesting to isolate some fragments of the logic and investigate how the transparency predicate influences their expressivity and intensionality, as done in [24].

The existential/universal quantifiers are omitted as they imply an undecidable satisfaction relation (cf. [16]), while we aim at a decidable logic. The decidability of BiLog is an open question. We plan to extend the result of [6] to isolate decidable fragments of BiLog. We introduced the freshness quantifier as it is useful to express hiding and it preserves decidability in spatial logics [18].

We have not addressed a logic for tree with hidden names. As a matter of fact, we have such a logic. More precisely we can encode abstract trees into bigraphs by controls ambs with arity one. The name assigned to this control will actually be the name of the ambient. Extrusion and renaming of abstract trees have their correspondence with closure and substitution of bigraphical terms. At the logical level we may encode operators of tree logic with hidden names as follows:

$$\begin{aligned}
©\, a &\stackrel{def}{=} ((a \leftarrow a) \otimes \mathbf{id}) \circ \mathbf{T} \\
\mathbf{C}x.\, A &\stackrel{def}{=} И x.\, (/x \otimes \mathbf{id}) \circ A \\
a \circledR A &\stackrel{def}{=} (\neg ©\, a \wedge A) \vee (/a \otimes \mathbf{id}) \circ A \\
\mathbf{H}x.\, A &\stackrel{def}{=} И x.\, x \circledR A
\end{aligned}$$

The operator $©\, a$ says that the name a appears in the outer face of the bigraphs. The new quantifier $\mathbf{C}x.\, A$ expresses the fact that in a process satisfying A a name has been closed. The revelation $\circledR$ says that A can be asserted by revealing the restricted name a, which may be hidden in the model as it must either to be closed by an edge or not to appear in the model. The hiding quantification $\mathbf{H}$ is derived as in [15]. We are currently studying the expressivity and decidability of this logical framework. Moreover, to obtain a robust logical setting, we are developing a proof theory, a sequent calculus in particular, that will be useful to compare BiLog with other spatial logics, not only with respect to the model theory, but also from a proof theoretical point of view.

Several important questions remain: as bigraphs have an interesting dynamics, specified using reactions rules, we plan to extend BiLog to such a framework. Building on the encodings of ambient and π calculi into bigraphical reactive systems, we expect a dynamic BiLog to be able to express both ambient logic [13] and spatial logics for π-calculus [4]. Finally, more recent works that suggest applications and possible extensions for BiLog are [22,33].

Acknowledgment. We thank Philippe Bidinger, Annalisa Bossi, Rohit Chadha, Murdoch Gabbay, Giorgio Ghelli, Robin Milner, and Peter O'Hearn for useful comments and discussions.

References

[1] S. Abiteboul, O. Benjelloun, and T.Milo. Positive active XML. In *Proc. of Symposium on Principles of Database Systems (PODS)*, 2004.

[2] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: from relations to semistructured data*. Morgan Kaufmann, 1999.

[3] L. Birkedal, S. Debois, E. Elsborg, T. Hildebrandt, and H. Niss. Bigraphical models of context-aware systems. In *Proc. of Foundations of Software Science and Computation Structures (FOSSACS)*, 2006. To appear.

[4] L. Caires and L. Cardelli. A spatial logic for concurrency (Part I). In *Proc. of International Symposium on Theoretical Aspects of Computer Software (TACS)*, volume 2215 of *LNCS*, pages 1–37. Springer-Verlag, 2001.

[5] L. Caires and E. Lozes. Elimination of quantifiers and undecidability in spatial logics for concurrency. In *Proc. of International Conference on Concurrency Theory (CONCUR)*, volume 3170 of *LNCS*, pages 240–257. Springer-Verlag, 2004.

[6] C. Calcagno, L. Cardelli, and A. D. Gordon. Deciding validity in a spatial logic for trees. In *Proc. of ACM SIGPLAN Workshop on Types in Language Design and Implementation (TLDI)*, pages 62 – 73. ACM Press, 2003.

[7] C. Calcagno, P. Gardner, and U. Zarfaty. A context logic for tree update. In *Proc. of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 271–282. ACM Press, 2005.

[8] L. Cardelli. Describing semistructured data. *SIGMOD Record, Database Principles Column*, 30(4), 2001.

[9] L. Cardelli, P. Gardner, and G. Ghelli. Querying trees with pointers. Manuscript.

[10] L. Cardelli, P. Gardner, and G. Ghelli. A spatial logic for querying graphs. In *Proc. of International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2380 of *LNCS*, pages 597 – 610. Springer-Verlag, 2002.

[11] L. Cardelli, P. Gardner, and G. Ghelli. Manipulating trees with hidden labels. In *Proc. of International Conference on Foundations of Software Science and Computational Structures (FOSSACS)*, volume 2620 of *LNCS*, pages 216–232. Springer-Verlag, 2003.

[12] L. Cardelli and G. Ghelli. TQL: A query language for semistructured data based on the ambient logic. *Mathematical Structures in Computer Science*, 14:285–327, 2004.

[13] L. Cardelli and A. D. Gordon. Ambient logic. *Mathematical Structures in Computer Science*. To appear.

[14] L. Cardelli and A. D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Proc. of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*. ACM Press, 2000.

[15] L. Cardelli and A. D. Gordon. Logical properties of name restriction. In *Proc. of International Conference on Typed Lambda Calculi and Applications (TCLA)*, volume 2044 of *LNCS*, pages 46–60. Springer-Verlag, 2001.

[16] W. Charatonik and J.M. Talbot. The decidability of model checking mobile ambients. In *Proc. of Workshop on Computer Science Logic (CSL)*, volume 2142 of *LNCS*, pages 339 – 354. Springer-Verlag, 2001.

[17] G. Conforti. *Spatial Logics for Semistructured Resources*. PhD Thesis, Informatics Department, University of Pisa, 2005.

[18] G. Conforti and G. Ghelli. Decidability of freshness, undecidability of revelation. In *Proc. of International Conference on Foundations of Software Science and Computational Structures (FOSSACS)*, volume 2987 of *LNCS*, pages 105–120. Springer-Verlag, 2004.

[19] G. Conforti, D. Macedonio, and V. Sassone. Bigraphical logics for XML. In *Proc. of Italian Symposium on Advanced Database Systems (SEBD)*, pages 392 – 399, 2005.

[20] G. Conforti, D. Macedonio, and V. Sassone. Spatial logics for bigraphs. In *Proc. of International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3580 of *LNCS*, pages 766 – 778. Springer-Verlag, 2005.

[21] S. Dal Zilio and D. Lugiez. A logic you can count on. In *Proc. of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, 2004.

[22] T. C. Damgaard and L. Birkedal. Axiomatizing binding bigraphs (revised). Technical Report TR-2005-71, IT University of Copenhagen, 2005.

[23] T. Hildebrandt and J.W. Winther. Bigraphs and (Reactive) XML, an XML-centric model of computation. Technical Report TR-2005-26, University of Copenhagen, February 2005.

[24] D. Hirschkoff. An extensional spatial logic for mobile processes. In *Proc. of International Conference on Concurrency Theory (CONCUR)*, volume 3170 of *LNCS*, pages 325–339. Springer-Verlag, 2004.

[25] O. H. Jensen and R. Milner. Bigraphs and mobile processes (revised). Technical Report UCAM-CL-TR-580, University of Cambridge, February 2004.

[26] O.H. Jensen. Forthcoming PhD Thesis. Aalborg University, 2004.

[27] D. Macedonio. *Logics for Distributed Resources*. PhD Thesis TD-2006-2, Informatics Department, University Ca' Foscari of Venice, 2006.

[28] R. Milner. Bigraphical reactive systems. In *Proc. of International Conference on Concurrency Theory (CONCUR)*, volume 2154 of *LNCS*, pages 16–35. Springer-Verlag, 2001.

[29] R. Milner. Bigraphs for petri nets. In *Lectures on Concurrency and Petri Nets: Advances in Petri Nets*, volume 3098 of *LNCS*, pages 686–701. Springer-Verlag, 2004.

[30] R. Milner. Axioms for bigraphical structure. *Mathematical Structures in Computer Science*, 15(6):1005–1032, 2005.

[31] R. Milner. Bigraphs whose names have multiple locality. Technical Report UCAM-CL-TR-603, University of Cambridge, January 2005.

[32] R. Milner. Pure bigraphs: Structure and dynamics. *Information and Computation*, 204(1):60–122, 2006.

[33] S. O'Conchuir. Kind bigraphs - static theory. Thecnical Report TCD-CS-2005-36, Trinity College Dublin, Computer Science Department, 2005.

[34] Peter O'Hearn, John C. Reynolds, and Hongseok Yang. Local reasoning about programs that alter data structures. In *Proc. of International Workshop on Computer Science Logic (CSL)*, volume 2142 of *LNCS*, pages 1–19. Springer-Verlag, 2001.

[35] A. M. Pitts. Nominal logic: a first order theory of names and binding. In *Proc. of International Symposium on Theoretical Aspects of Computer Software (TACS)*, volume 2215 of *LNCS*, pages 219–242. Springer-Verlag, 2001.

[36] D. Sangiorgi. Extensionality and intensionality of the ambient logic. In *Proc. of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 4–13. ACM Press, 2001.