Spatial Logics for Bigraphs[⋆]

Giovanni Conforti[1,3], Damiano Macedonio[2,3], and Vladimiro Sassone[3]

[1] Università di Pisa
[2] Università Ca' Foscari di Venezia
[3] University of Sussex

Abstract. Bigraphs are emerging as an interesting model for concurrent calculi, like CCS, pi-calculus, and Petri nets. Bigraphs are built orthogonally on two structures: a hierarchical place graph for locations and a link (hyper-)graph for connections. With the aim of describing bigraphical structures, we introduce a general framework for logics whose terms represent arrows in monoidal categories. We then instantiate the framework to bigraphical structures and 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 some known spatial logics for trees, graphs and tree contexts.

1 Introduction

To describe and reason about structured, distributed, dynamic resources is one of the main goals of global computing research. Recently, many *spatial logics*, in different contexts, have been studied to fulfill this goal. The term 'spatial,' as opposed to 'temporal,' refers to the use of modal operators inspecting the structure of the terms in the considered model. Spatial logics are usually equipped with a separation/composition binary operator that *splits* a term into two parts, in order to 'talk' about them separately. Looking closely, we observe that the notion of *separation* is interpreted differently in different logics. In 'separation' logics [18], it is used to reason about dynamic update of heap-like structures, and it is *strong* in that it forces names of resources in separated components to be disjoint. As a consequence, term composition is usually partially defined. In static spatial logics (e.g., for, trees [2], graphs [4] or trees with hidden names [5]), the separation/composition does not require any constraint on terms, and names are usually shared between separated parts. Similarly in dynamic spatial logics (for, e.g., ambients [6] or π-calculus [1]), where the separation is intended only for location in space. Context tree logic, introduced in [3], 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.

[⋆] Research partially supported by the EU projects: IHP 'Marie Curie **DisCo**' HPMT-CT-2001-00290, FET-GC '**MIKADO**' IST-2001-32222, and FET-GC '**MyThS**' IST-2001-32617.

L. Caires et al. (Eds.): ICALP 2005, LNCS 3580, pp. 766–778, 2005.

Bigraphs [12, 14] are an emerging model for structures in global computing, which can be instantiated to model several well-known examples, including CCS [17], π-calculus [12], and Petri nets [16]. 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, bigraphs make clear 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 formulae like $\mathsf{PC}(A)$ and $\mathsf{PC}_x(A)$ to describe a resource in an unnamed location, respectively location x, of 'type' PC (e.g. a computer) whose contents satisfy A. 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). Using named locations, as e.g. in $\mathsf{PC}_a(\mathbf{T}) \otimes \mathsf{PC}_b(\mathbf{T})$, we are able to express name separation, i.e., that names a and b are different. Furthermore, using link expressions we can force name-sharing between resources with formulae like:

$$\mathsf{PC}_a(\mathsf{in}_c \otimes \mathbf{T}) \overset{c}{\otimes} \mathsf{PC}_b(\mathsf{out}_c \otimes \mathbf{T})$$

This describes two PC with different names, a and b, 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 names are shared and c cannot be used elsewhere inside the PCs.

A bigraphical structure is, in general, a context with several holes and open links that can be filled by composition. This means that the logic can describe 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 'pluggable' hole in the hard disc HD. Contextual resources have many important applications. In particular, the contextual nature of bigraphs is useful to specify reaction rules, but it can also be used as a general mechanism to describe contexts of bigraphical data structures (cf. [8, 10]).

As bigraphs are establishing themselves as a truly general (meta)model of global systems, 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 positive preliminary results obtained for semistructured data [8] and CCS [9].

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 ports that may be connected to each other (and to names) by *links* (the so-called *link graph*). Place graphs express locality, i.e., the physical arrangement of the nodes. Link graphs are hypergraphs 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 kinds of nodes, and have fixed *arities* that determine their number of ports. Control PC marks nodes representing computers, and its arity is 3: in clockwise order, these ports represent a keyboard interacting with an employee U, a LAN to an other PC and open to the outside network, and a plug connecting the computer to the electrical mains of office R. Employees U may communicate with each other 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 like 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. In figure 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 $P : m \to n$ to indicate a place graph P with m holes and n regions. In Fig. 1, the place graph of G is of type $2 \to 1$. Given 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 'renumbers' regions and holes 'from left to right'.

Link graphs are arrows of a partial monoidal category whose objects are (finite) sets Λ of names, that we assume to be *denumerable*. A link graph is an arrow $X \to Y$,

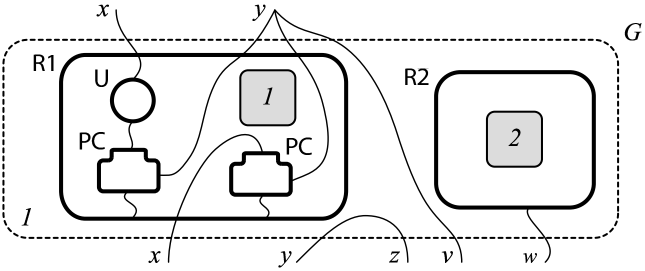

Fig. 1. A bigraph $G : \langle 2, \{x, y, z, v, w\}\rangle \to \langle 1, \{x, y\}\rangle$

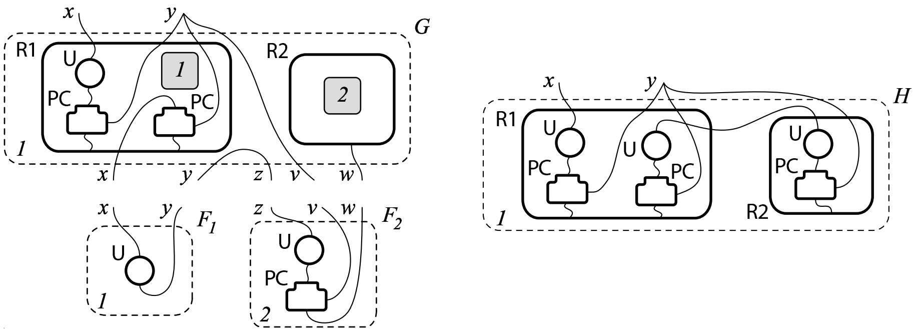

Fig. 2. Bigraphical composition, $H \equiv G \circ (F_1 \otimes F_2)$

with $X, Y \subseteq \Lambda$. The set X represents the *inner* names (drawn at the bottom of the bigraph) and Y represents the set of *outer* names (drawn on the top). The link graph connects ports to names or to *edges*, 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 (represented in Fig. 1 by a line between nodes) 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 $W \circ W'$, and the link graphs can perform substitution and renaming. The tensor product of link graphs is defined in the obvious way only if their inner (resp. outer) names are disjoint.

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

Fig. 2 represents a more complex situation. At the top left-hand side is the system of Fig. 1, At the 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.

3 BiLog: Syntax and Semantics

As place and link graphs are arrows of a (partial) monoidal category, we first introduce a logic having monoidal categories as models and then we adapt it to model the orthogonal structures of place and link graphs. Each of these is expressive enough to model and generalise (e.g. by means of contexts) well-known spatial logics. Finally we apply the logic to model the whole structure of abstract bigraphs.

Table 1. BiLog$(M, \otimes, \epsilon, \Theta, \equiv, \tau)$

$\boldsymbol{\Omega}$ ::=	$\mathbf{id}_I \mid \dots$	a constant formula for every Ω s.t. $\tau(\Omega)$		
A, B ::=	$\mathbf{F}$	false	$A \Rightarrow B$	implication
	$\mathbf{id}$	identity	$\boldsymbol{\Omega}$	Constant for a simple term
	$A \otimes B$	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

$G \models \mathbf{F}$	$\stackrel{def}{=}$	never
$G \models A \Rightarrow B$	$\stackrel{def}{=}$	$G \models A$ implies $G \models B$
$G \models \boldsymbol{\Omega}$	$\stackrel{def}{=}$	$G \equiv \Omega$
$G \models \mathbf{id}$	$\stackrel{def}{=}$	$\exists I. G \equiv id_I$
$G \models A \otimes B$	$\stackrel{def}{=}$	$\exists G_1, G_2. G \equiv G_1 \otimes G_2$ and $G_1 \models A$ and $G_2 \models B$
$G \models A \circ B$	$\stackrel{def}{=}$	$\exists G_1, G_2. G \equiv G_1 \circ G_2$ and $\tau(G_1)$ and $G_1 \models A$ and $G_2 \models B$
$G \models A \circ\!\!- B$	$\stackrel{def}{=}$	$\forall G'. G' \models A$ and $\tau(G')$ and $(G' \circ G)\downarrow$ implies $G' \circ G \models B$
$G \models A -\!\!\circ B$	$\stackrel{def}{=}$	$\tau(G)$ implies $\forall G'. G' \models A$ and $(G \circ G')\downarrow$ implies $G \circ G' \models B$
$G \models A \otimes\!\!- B$	$\stackrel{def}{=}$	$\forall G'. G' \models A$ and $(G' \otimes G)\downarrow$ implies $G' \otimes G \models B$
$G \models A -\!\!\otimes B$	$\stackrel{def}{=}$	$\forall G'. G' \models A$ and $(G \otimes G')\downarrow$ implies $G \otimes G' \models B$

The models are categories built on a (possibly partial) monoid $(M, \otimes, \epsilon)$, whose elements are dubbed *interfaces* and denoted by I, J. The elements of a BiLog model are arrows on the corresponding (partial) monoid. Given a set of term constructors Θ, ranged over by Ω, the arrows are defined by the term language $G ::= G \otimes G' \mid G \circ G' \mid \Omega$. Each Ω in Θ has a 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 obvious rules [9] determine the type of each term. Terms of type $\epsilon \to J$ are called *ground*.

We consider terms up to a structural congruence $\equiv$, which subsumes the axioms of monoidal categories [9]. Later on, the congruence will be refined to model specialised structures, such as place graphs or bigraphs. All axioms are required to hold only when both sides are well typed. Throughout the paper, when using $=$ or $\equiv$ we imply that both sides are defined and we write $(G)\downarrow$ to say that G is defined.

BiLog internalises the term constructors in the style of the ambient logic [6]. Constructors are represented in the logic as constant formulae, while tensor product and composition are expressed by connectives. We thus have two binary spatial operators. This contrasts with other spatial logics, which have only one: ambient-like logics, with parallel composition $A \mid B$, Separation Logic [18], with separating conjunction $A * B$, and Context Tree Logic [3], with application $K(P)$. Our logic is parameterised on a transparency predicate τ, reflecting that not every term can be directly observed in the logic: some are opaque and do not allow inspection of their contents. We will see that when all terms are observable (i.e. $\tau(G)$ for all G), logical equivalence corresponds to $\equiv$. Otherwise, it can be less discriminating. We assume that id_I and ground terms are always transparent, and τ preserves $\equiv$, hence $\otimes$ and $\circ$, in particular.

The logic BiLog$(M, \otimes, \epsilon, \Theta, \equiv, \tau)$ is formally defined in Table 1 and the meaning of formulae is given in terms of a satisfaction relation. It features a logical constant

$\mathbf{\Omega}$ for each *transparent* construct Ω. 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 terms satisfying A and B respectively. The degree of separation enforced by $\otimes$ between terms plays a fundamental role in the various fragments of the logic (notably link graph and place graph). The *vertical decomposition* formula $A \circ B$ is satisfied by terms that can be seen as the composition of terms satisfying A and B. We shall see that both connectives correspond in some cases to well known spatial connectives. We define the *left* and *right adjuncts* for composition and tensor to express extensional properties. The left adjunct $A \mathrel{\circ\!\!-} B$ expresses the property of a term to satisfy B whenever inserted in a context satisfying A. Similarly, the right adjunct $A \multimap B$ expresses the property of a context to satisfy B whenever filled with a term satisfying A. A similar description for $\otimes\!-$ and $-\!\otimes$, the adjoints of $\otimes$. Observe that these collapse if the tensor is commutative in the model.

Derived Operators and Logical Properties. In Table 2 we outline some interesting operators that can be derived in BiLog. The operators constraining the interfaces are self-explanatory. The 'dual' operators have the following semantics: $A \ominus B$ is satisfied by terms G such that for every possible decomposition $G_1 \otimes G_2$ 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. 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 adjuncts work as usual. 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. The equality between interfaces $I = J$ is easily derivable using $\otimes$ and $\otimes\!-$.

We can extend the idea of *sublocation* ($\sqsubseteq$) defined in [7] to our terms. The inductive definition of $\sqsubseteq$ specifies that $G \sqsubseteq G$, and $G' \sqsubseteq G$ if either $G \equiv G_1 \otimes G_2$, with $G' \sqsubseteq G_1$ (and symmetrically $G' \sqsubseteq G_2$) or $G \equiv G_1 \circ G_2$, with $\tau(G_1)$ and $G' \sqsubseteq G_2$. Exploiting this relation between ground terms, we define a *somewhere* modality. Intuitively, we say that a term satisfies ✧A whenever one of its sublocations satisfies A. Quite surprisingly, ✧A is expressible in the logic, as described in [9].

The lemma below states that the relation $\models$ is well defined w.r.t. the congruence and that the interfaces for transparent terms can be observed.

Lemma 3.1 (Type and Congruence preservation).
For every couple of term G, G', it holds: $G \models A$ and $G \equiv G'$ implies $G' \models A$.
For every term G, it holds: $G \models A_{I\to J}$ if and only if $G : I \to J$, $G \models A$, and $\tau(G)$.

BiLog induces a logical equivalence $=_L$ on terms in the usual sense, that is $G_1 =_L G_2$ if $G_1 \models A$ implies $G_2 \models A$ and vice versa, for every formula A.

Theorem 3.2 (Logical equivalence and congruence). *If the transparency predicate is always true, then for every term G, G', it holds: $G =_L G'$ if and only if $G \equiv G'$.*

Place Graph Logic (PGL). Place graphs are essentially ordered lists of regions hosting unordered labelled trees with holes. The labels of the trees correspond to controls K belonging to the fixed signature $\mathcal{K}$. We consider the monoid $(\omega, +, 0)$ of finite ordinals

Table 2. Derived Operators

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

m, n. Interfaces here represent the number of holes and regions of place graphs. Place graph terms are generated from the set $\Theta = \{1 : 0 \to 1, id_n : n \to n, join : 2 \to 1, \gamma_{m,n} : m + n \to n + m, K : 1 \to 1 \text{ for } K \in \mathcal{K}\}$. The main structural term is K, that represents a region containing a single node with a hole inside. Other simple terms are *placings*, representing trees $m \to n$ with no nodes; 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 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 neutral element 1. Hence, the places generated by composition and tensor product from $\gamma_{m,n}$ are *permutations*. A place graph is *prime* if it has type $I \to 1$ (i.e., with a single region).

Defined the transparency predicate τ on each control in $\mathcal{K}$, the Place Graph Logic PGL$(\mathcal{K}, \tau)$ is $BiLog(\omega, +, 0, \equiv, \mathcal{K} \cup \{1, join, \gamma_{m,n}\}, \tau)$. We assume the τ to be true for *join* and $\gamma_{m,n}$. It follows from Theorem 3.2 that PGL can describe place graphs precisely. The logic resembles a propositional spatial tree logic, like [2]. The main differences are that PGL models contexts of trees and that the tensor product is not commutative, allowing us to model the order of regions. We can define a commutative separation using **join** and the tensor product, the *parallel composition* $A \mid B \stackrel{def}{=} \mathbf{join} \circ (A_{\to 1} \otimes B_{\to 1})$. This separation is purely structural, and corresponds at term level to $join \circ (P \otimes P')$ that is a total operation on all prime place graphs.

We show that BiLog restricted to prime ground place graphs (with the always-true transparency predicate) is equivalent to the propositional spatial tree logic of [2] (STL in the following). The logic STL expresses properties of unordered labelled trees T

Table 3. Encoding STL in PGL over prime ground place graphs

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

constructed from the empty tree 0, the labelled node containing a tree $a[T]$, and the parallel composition of trees $T_1 \mid T_2$. It is a static fragment of the ambient logic [6] characterised by propositional connectives, spatial connectives (i.e., 0, $a[A]$, $A \mid B$), and their adjuncts (i.e., $A@a$, $A \triangleright B$).

In Table 3 we encode 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 associate every label a with a distinct control $K(a)$. 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; and *(iii)* location and composition adjuncts of STL are encoded in terms of the left composition adjunct, as they add logically expressible contexts to the tree. This encoding allows us to prove the following.

Theorem 3.3 (Encoding STL). *For each tree T and formula A of STL we have that $T \models_{SL} A$ if and only if $\llbracket T \rrbracket \models (\llbracket A \rrbracket)_{0\to 1}$.*

Differently from STL, PGL can also describe structures with several holes and regions. In [8] we show how PGL describes contexts of tree-shaped semistructured data. In particular multi-contexts can be useful to specify properties of web-services. Consider for instance a function taking two trees and returning the tree obtained by merging their roots. Such 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 $\mathbf{join} \circ (\mathbf{K} \otimes (\mathbf{T} \circ \mathbf{id}_1))$ 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.

Link Graph Logic (LGL). For Λ a denumerable set of names, we consider the monoid of interfaces $(\mathcal{P}_{fin}(\Lambda), \uplus, \emptyset)$, where $\mathcal{P}_{fin}(\_)$ is the finite powerset operator and $\uplus$ is the union on disjoint pairs of sets and undefined otherwise. The structures that arise from such a monoid are the link graphs discussed in §2. They can describe nominal resources common in many areas, such as object identifiers, location names in memory structures, channel names, and ID attributes in XML documents.

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 ${}^{a}/_{X}$ associates all the names in the set X to the name a. We denote wirings by ω, substitutions by σ, τ, and *renamings* (i.e., bijective substitutions) by α, β. Substitutions can be specialised in:

$$a \overset{def}{=} {}^{a}/_{\emptyset}; \qquad a \leftarrow b \overset{def}{=} {}^{a}/_{\{b\}}; \qquad a \leftleftarrows b \overset{def}{=} {}^{a}/_{\{a,b\}}.$$

Constructor a represents the introduction of a name a, term $a \leftarrow b$ the renaming of b to a, and finally $a \leftleftarrows b$ links (or fuse) a and b in the name a.

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

The structural congruence $\equiv$ for link graphs is refined with obvious axioms for links, modelling α-conversion and extrusion of closed names, cf. [9]. We assume the transparency predicate τ to be true for wiring constructors.

Given the transparency τ for each control in $\mathcal{K}$, the Link Graph Logic LGL($\mathcal{K}, \tau$) is $BiLog(\mathcal{P}_{fin}(\Lambda), \uplus, \emptyset, \equiv, \mathcal{K} \cup \{/a, {}^{a}/_{X}\}, \tau)$. By Theorem 3.2, LGL 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/private) names between resources, and in this sense the logic is a generalisation of Spatial Graph Logic [4], considering the edges as resources.

In LGL the formula $A \otimes B$ describes a decomposition into two *separate* link graphs (i.e., sharing no resources, names, nor connections) satisfying respectively A and B. Observe that 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 the sharing to be made 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 in order 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 $a_1 \leftarrow b_1 \otimes \ldots \otimes a_n \leftarrow b_n$ (and similarly for $\vec{a} \leftleftarrows \vec{b}$). It is possible to derive from the tensor product 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:

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

By extending this sharing to all names we can 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 [15]. 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.

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 a fresh name quantification in the style of Nominal Logic [19].

$$G \models \text{И}x_1,\ldots,x_n.A \stackrel{def}{=} \exists a_1\ldots a_n \notin fn(G)\cup fn(A).\, G \models A\{x_1,\ldots\; x_n \leftarrow a_1\ldots a_n\}$$

Using fresh name quantification we can define a notion of $\vec{a}$-linked name quantification for fresh names, whose purpose is to identify names that are linked to $\vec{a}$:

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

The formula above expresses that 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-upto*, namely the decomposition in two terms that are separated apart from the link on the specific names in $\vec{a}$, which crosses the separation line.

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

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 $\vec{a}$ in order to actually have the sharing. The corresponding parallel composition operator is not directly definable using separation-upto, since we do not know a priori the name shared in arbitrary decompositions. However, we will show that a careful encoding is possible for the parallel composition of spatial logics with nominal resources.

We show that LGL can be seen as a contextual (and multi-edge) version of Spatial Graph Logic (SGL) [4]. The logic SGL expresses properties of directed edge labelled graphs G built from the empty graph *nil*, the edge labelled a from x to y nodes $a(x,y)$, the parallel composition of graphs $G_1 \mid G_2$, and the binding for local names of nodes $(\nu x)G$. We consider a $\mathcal{K}$ such that: there is a bijective function associating every edge label a to a distinct control $K(a)$ and the arity of every control is 2 (the ports represent the starting and arrival node respectively). The resulting link graphs can be interpreted as contextual edge labelled graphs and the resulting class of ground link graphs is isomorphic to the graph model of SGL. In Table 4 we encode 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-upto $\vec{a}$, where $\vec{a}$ is a list of all the elements in X. Notice also how local names are encoded into name closures (and identity).

Theorem 3.4 (Encoding SGL). *For each graph G, finite set X containing $fn(G)$, and formula ϕ of the propositional fragment of SGL, we have that $G \models_{GL} \phi$ if and only if $[\![\, G \,]\!]_X \models ([\![\, \phi \,]\!]_X)_{\emptyset\to X}$.*

In LGL is also possible to encode the Separation Logics on heaps: names used as identifiers of location will be forcibly separated by tensor product, while names used for pointers will be shared/linked.

Bigraphs as a Model for BiLog. We combine the structures of link graphs and place graphs to generate all *(abstract pure) bigraphs* of [12]. We take as monoid the product of link and place interfaces, i.e. $(\omega \times \mathcal{P}_{fin}(\Lambda), \otimes, \epsilon)$ where $\langle m,X\rangle \otimes \langle n,X\rangle \stackrel{def}{=} \langle m+n, X \uplus Y\rangle$

Table 4. Encoding Propositional SGL in LGL over two ported ground link graphs

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

and $\epsilon \overset{def}{=} \langle 0, \emptyset \rangle$. We will use X for $\langle 0, X \rangle$ and n for $\langle n, \emptyset \rangle$. As constructors for bigraphical terms we have the union of place and link graph constructors apart from the controls $K : 1 \to 1$ and $K_{\vec{a}} : \emptyset \to \vec{a}$, which are replaced by the new *discrete ion* constructor, which we note $K_{\vec{a}} : 1 \to \langle 1, \vec{a} \rangle$; this is a prime bigraph containing a single node with ports named $\vec{a}$ and an hole inside. Bigraphical terms thus are defined w.r.t. a control signature $\mathcal{K}$ and a set of names Λ, cf. [15] for details.

PGL excels at expressing properties of *unnamed* resources, i.e., 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 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 witness this we have proved in [9] that also the recently proposed Context Logic for Trees [3] can be encoded into bigraphs. The idea of the encoding is to extend the one of STL with contexts and identified nodes. Essentially, in [9] we show that the model of [3] is a particular class of prime bigraphs with one port for each node and a number of holes and regions limited to one. Since [3] is more general than separation logic, and is used to reason about programs that manipulate tree structured memories, it is possible to generalise separation logic as well.

4 Conclusion and Future Work

In this paper we moved 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 CCS [17], pi-calculus [12] and Petri-nets [13, 16]. We introduced BiLog, a logic for bigraphs (and more generally for monoidal categories), with two main spatial connectives: composition and tensor product. Our main technical results are the embedding and comparison with other spatial logics previously studied. Moreover, we have shown that BiLog is expressive enough to internalise the somewhere modality. In §3 we observed that the induced logical equivalence can be forced to coincide with the structural congruence of terms. This property is fundamental in order to describe, query and reason about bigraphical data structures. For a more detailed discussion we refer to [8].

In [9] we study how BiLog can deal with dynamics. A natural solution is to add a temporal *next step* modality basically describing bigraphs that can compute (*react*) according to a Bigraphical Reactive System [12]. When the transparency predicate τ enables the inspection of 'dynamic' controls, BiLog is '*intensional*' in the sense of [11], namely it can observe internal structures. In several cases, notably the bigraphical system describing CCS [17], this can be to the extent that the next step modality can be expressed directly by using the static fragment of BiLog. Notice that τ specifies what structure the logic can directly observe, while the next step modality, along with the spatial connectives, allows to deduce the structure by observing the behaviour. It would be interesting to investigate how the transparency predicate influences the expressiveness and intentionality of significant fragments of the dynamic logic.

The 'separation' plays differently in various fragments of the logic. For instance, in the case of *Place Graph Logic*, where the model is the class of bigraphs without names, the separation is purely structural and coincides with the notion of parallel composition in Spatial Tree Logic. The separation in the *Link Graph Logic* is disjointness of nominal resources. Finally, for *Bigraph Logic* it is a combination that can be seen as separation in a structured term with nominal resources (e.g. the trees with pointers of [3] and trees with hidden names [5]). In the paper we have not addressed logical operators for hidden names (e.g. ®, **H**, © of ambient logic). We can encode them easily using, in particular, И and $/a$. The decidability of BiLog is an open question, we are working on extending the results of [2], and we are isolating decidable fragments of BiLog.

We are currently developing a proof theory for Bilog in order to complete the robust logical setting provided by the model theory presented here. Besides aiming at a generalise existing proof systems, this will allow direct comparisons between BiLog and other spatial logics also from the proof-theoretic point of view.

Acknowledgment. We would like to thank Philippe Bidinger, Annalisa Bossi, Rohit Chadha, Murdoch Gabbay, Giorgio Ghelli, Robin Milner, Peter O'Hearn and all the anonymous referees for useful comments and discussions.

References

1. L. Caires and L. Cardelli. A spatial logic for concurrency (Part I). In *Proc. of TACS*, volume 2215 of *LNCS*, pages 1–37. Springer-Verlag, 2001.
2. C. Calcagno, L. Cardelli, and A. D. Gordon. Deciding validity in a spatial logic for trees. In *Proc. of TLDI*, 2003.
3. C. Calcagno, P. Gardner, and U. Zarfaty. A context logic for tree update. In *Proc. of LRPP*, 2004; revised version to appear in *POPL*, 2005.
4. L. Cardelli, P. Gardner, and G. Ghelli. A spatial logic for querying graphs. In *Proc. of ICALP*, volume 2380 of *LNCS*, page 597. Springer-Verlag, 2002.
5. L. Cardelli, P. Gardner, and G. Ghelli. Manipulating trees with hidden labels. In *Proc. of FOSSACS*, volume 2620 of *LNCS*, pages 216–232. Springer-Verlag, 2003.
6. L. Cardelli and A. D. Gordon. Ambient logic. To appear in Mathematical Structures in Computer Science.
7. L. Cardelli and A. D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Proc. of POPL*. ACM Press, 2000.

8. G. Conforti, D. Macedonio, and V. Sassone. Bigraphical logics for XML. In *Proc. of SEBD*, 2005. To appear.
9. G. Conforti, D. Macedonio, and V. Sassone. BiLog: spatial logics for bigraphs. Computer Science Report 2005:02, University of Sussex, 2005.
10. T. Hildebrandt and J.W. Winther. Bigraphs and (Reactive) XML, an XML-centric model of computation. IT University of Copenhagen Technical Report TR-2005-26, 2005.
11. D. Hirschkoff. An extensional spatial logic for mobile processes. In *Proc. of CONCUR*, volume 3170 of *LNCS*, pages 325–339. Springer-Verlag, 2004.
12. O. H. Jensen and R. Milner. Bigraphs and mobile processes (revised). Technical Report UCAM-CL-TR-580. University of Cambridge, 2004.
13. J. J. Leifer and R. Milner. Transition systems, link graphs and petri nets. Technical Report UCAM-CL-TR-598. University of Cambridge, 2004.
14. R. Milner. Bigraphical reactive systems. In *Proc. of CONCUR*, volume 2154 of *LNCS*, pages 16–35. Springer-Verlag, 2001.
15. R. Milner. Axioms for bigraphical structure. Technical Report UCAM-CL-TR-581. University of Cambridge, 2004.
16. R. Milner. Bigraphs for petri-nets. In *Lectures on Concurrency and Petri Nets: Advances in Petri Nets*, pages 686–701. Springer-Verlag, 2004.
17. R. Milner. Pure bigraphs. Technical Report UCAM-CL-TR-614. University of Cambridge, 2005.
18. P. O'Hearn, J. C. Reynolds, and H. Yang. Local reasoning about programs that alter data structures. In *Proc. of CSL*, volume 2142 of *LNCS*, pages 1–19. Springer-Verlag, 2001.
19. A. M. Pitts. Nominal logic: A first order theory of names and binding. In *Proc. of TACS*, volume 2215 of *LNCS*, pages 219–242. Springer-Verlag, 2001.