Fundamenta Informaticae 80 (2007) 1–20
IOS Press

Static BiLog: a Unifying Language for Spatial Structures *

Giovanni Conforti
DI, Università di Pisa, Italy

Damiano Macedonio
DSI, Università Ca' Foscari di Venezia, Italy

Vladimiro Sassone†
ECS, University of Southampton, UK
vs@ecs.soton.ac.uk

Abstract. Aiming at a unified view of the logics describing spatial structures, we introduce a general framework, BiLog, whose formulae characterise monoidal categories. As a first instance of the framework we consider bigraphs, which are emerging as a an interesting (meta-)model for spatial structures and distributed calculi. Since bigraphs are built orthogonally on two structures, a hierarchical place graph for locations and a link (hyper-)graph for connections, we obtain a logic that is a natural composition of other two instances of BiLog: a Place Graph Logic and a Link Graph Logic. We prove that these instances generalise the spatial logics for trees, for graphs and for tree contexts. We also explore the concepts of separation and sharing in these logics. We note that both the operator $*$ of Separation Logic and the operator $|$ of spatial logics do not completely separate the underlying structures. These two different forms of separation can be naturally derived as instances of BiLog by using the complete separation induced by the tensor product of monoidal categories along with some form of sharing.

Keywords: Bigraphs, Spatial Logics, Context Logic, Separation.

1. Introduction

To describe and reason about structured and distributed resources is one of the main goals of global computing research. Recently, many *spatial logics* have been studied to fulfill this aim [3, 4, 5, 6, 7, 8,

*Work partially supported by M.I.U.R (Italian Ministry of Education, University and Research) under contract n. 2005015785.
†Address for correspondence: ECS, University of Southampton, UK

15, 23, 25]. The term 'spatial,' as opposed to 'temporal,' refers to the use of operators inspecting the structure of the terms in the model, rather than a temporal behaviour. Spatial logics are usually equipped with a separation/composition connective that *splits* a term into two parts, in order to 'talk' about them separately. The notion of *separation* is interpreted differently in different logics.

In 'separation' logics [23], separation is used to reason about 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 spatial logics for trees [4] and graphs [6] the separation/composition operator is *structural*, as it induces separation in the spatial structure, but *weak* on names, as there is no constraint on terms, and names are usually shared between separated parts. In spatial logics which describe models with name restriction, like the Ambient Logic [8] or the Spatial Logic for π-calculus [3], separation is generally intended only for locations in space. Nevertheless, as a consequence of name restriction combined with name extrusion, the logical separation/composition operator separates on private/restricted names, and shares public names. Context Tree Logic [5] integrates Separation Logic with a spatial logic for trees. The result is a logic that describes tree-shaped structures (and contexts) with pointers.

All these logics have no a direct way of specifying the public names that can be shared among logically separated components, thus they cannot express *explicit sharing*. Here we introduce a new form of separation that subsumes the different kinds of separation discussed above, and define a form of explicit sharing that naturally subsumes unrestricted sharing (viz. parallel composition of spatial logics) when combined with a quantification on names à la Nominal Logic [24].

Bigraphs [17, 19] are an emerging model for structures in global computing, and they can be instantiated to model several well-known examples, including CCS [22], π-calculus [17], ambients [16] and Petri nets [20]. 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 channels). The two structures are orthogonal, so links between nodes can cross locality boundaries.

In this paper we introduce a logic for distributed resources as a natural composition of a Place Graph Logic, for tree contexts, and a Link Graph Logic, for name links. The main point is that a resource is associated both to a spatial structure and to a link structure. Suppose for instance to describe a tree-shaped distribution of resources in locations. We may use an atomic formula like $\mathsf{PC}(A)$ for a resource of 'type' PC (e.g. a personal computer) whose contents satisfy A, and a formula like $\mathsf{PC}_x(A)$ for 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 $\mathbf{T}$ – i.e., with anything whatsoever 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 by means of formulae such as

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

This formula describes two PCs with different names, a and b, 'uniquely' sharing a link on a distinct name c, which may model 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 (because of $\otimes$).

A bigraphical structure is, in general, a context with several holes and open links that can be filled by composition. Therefore, when instantiated to bigraphs, the logic 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 (cf. [9, 18]), and it can also be used as a general mechanism to describe contexts of bigraphical data structures (cf. [11]).

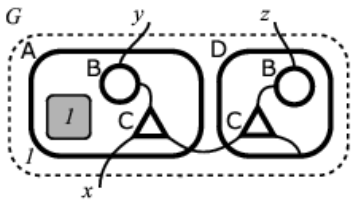
As bigraphs are establishing themselves as a truly general (meta)model of global systems, and appear to encompass several existing calculi and models (as shown in [16, 17, 20, 22]), our bigraph logic, *BiLog*, aims at achieving the same generality as a description language: as bigraphs specialise to 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 §4, and the positive preliminary results obtained for semistructured data [11] and CCS [9, 18]. Here, our main technical results are the encoding in BiLog fragments of the static spatial logics of [4, 5, 6].

BiLog was introduced in [12], this paper deepens some of the points covered there. In particular, we consider the static fragment of BiLog, we discuss more in detail about separation and we outline proofs. Further considerations and issues on BiLog are in [9, 11, 18]. Finally, we remind the reader to [21, 22] for a detailed background on bigraphs.

Structure of the paper: §2 recalls the basic background on bigraphs; §3 introduces the general framework and the model theory of BiLog; §4 instantiates the framework to obtain logics for place, link and bi-graphs; §5 focuses on separation and sharing concepts and §6 presents our conclusions.

2. Background

A bigraph consists of a set of *nodes*, which may be nested in a hierarchical tree structure expressing locality, the so-called *place graph*, and have *ports* that may be connected to each other by *links*, the so-called *link graph*. The two structures are completely orthogonal.



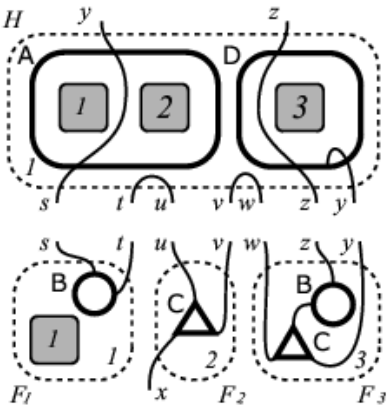
The picture on the right represents a bigraph G. Nodes, shown with bold outlines, are associated with a *control* (either A, B, C, D). Controls have fixed *arities* to determine the number of ports. For instance, B has arity 2, and C has arity 3. The nesting of nodes (place graph) is shown by the inclusion of nodes into each other; the connections (link graph) are drawn as lines. At the top level of the nesting structure sit the *regions*. The bigraph G has one sole region (the dashed box). Inside the nodes there may be 'context' *holes*, drawn as shaded boxes. Regions and holes are uniquely identified by finite ordinals.

Place graphs are *arrows* over a symmetric monoidal category whose objects are finite ordinals. The arrow $P : m \to n$ is a place graph P with m holes and n regions. The place graph of G has type $1 \to 1$. The composition of place graphs $P_1 \circ P_2$ 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 (left-to-right), thus obtaining a graph with more regions and holes, and it 'renumbers' regions and holes 'from left to right'.

Link graphs are arrows $X \to Y$ of a partial monoidal category whose objects are (finite) sets of names, belonging to a denumerable set. The set X represents the *inner* names (customarily drawn at

the bottom of the bigraph) and Y represents the set of *outer* names (drawn at the top). The link graph of G has type $\{x\} \to \{y, z\}$. The link graph connects ports to names or to *edges* (represented 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 $W \circ W'$. For instance, the fact that the outer names in W' disappear in the composition means that names may be renamed and that edges may be added to the structure. As in [17], 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 set of nodes we obtain the notion of bigraphs – i.e., arrows $G : \langle m, X\rangle \to \langle n, Y\rangle$. The bigraph G can be represented as the composition $H \circ F$ of the bigraphs depicted on the right. At the bottom of the picture, the system F_1, F_2 and F_3 represents the tensor product $F = F_1 \otimes F_2 \otimes F_3$. The idea of the composition is to insert F into the context H. The operation is partially defined, since it requires the inner names and the number of holes of H 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 H (place composition) and *then* joins equal inner names of H and outer names of F (link composition). Note the edge connecting the inner names t and u in H: it links two nodes of F after the composition.



3. The logic

This paper aims at defining a logic that describes bigraphs and their substructures. As bigraphs, place graphs, and link graphs are arrows of a (partial) monoidal category, we first introduce a meta-logical framework with monoidal categories as models; we then 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. The models are monoidal categories and the logic describes spatially the structure of their *arrows*. Our meta-logical framework is inspired by the bigraph axiomatisation of [21]. The model of the logic is composed of structures that can be placed one near to the other, via *horizontal composition*, or one inside the other, via *vertical composition*, and are generated by a set of *unary constructors*. These structures satisfy a *structural congruence* that conforms to the axioms of monoidal categories and possibly more. Thus the model theory is parametric both on term constructors and on structural congruence.

Table 3.1. *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 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

3.1. Models

To evaluate formulae, we consider the terms freely generated from a set of constructors Θ by using the vertical composition (the partial composition $\circ$) and the horizontal composition (the partial tensor $\otimes$). The order of binding precedence is $\circ$, $\otimes$. BiLog terms are defined as $G, G' ::= \Omega \mid G \circ G' \mid G \otimes G'$, where Ω ranges over Θ. We refer to these terms as *bifunctorial terms*, since the two operations must satisfy the *bifunctoriality property* of monoidal categories (the last rule in Table 3.1).

Terms are structures built on a monoid $(M, \otimes, \epsilon)$ whose elements are dubbed *interfaces* and denoted by I, J. To model nominal resources, e.g. heaps or link graphs, the monoid may be *partial*. Intuitively, terms represent typed structures with a source and a target interface ($G : I \to J$). Each constructor 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 obvious rules for composition and tensor [9, 18], determine the type of each term. Terms of type $\epsilon \to J$ are called *ground*. We consider only well typed terms.

Terms are defined up to a structural congruence $\equiv$ (see Tab. 3.1) which subsumes the axioms of (partial) 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 defined, we write $(E)\downarrow$. The congruence will be refined to model specialised structures: place graphs, link graphs and bigraphs.

3.2. Formulae

BiLog internalises the bifunctorial terms in the style of the Ambient Logic [8]. 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, that have a single operator: Spatial and Ambient Logics [3, 8], with the parallel composition $A \mid B$, Separation Logic [23], with the separating conjunction $A * B$, and Context Tree Logic [5], with the application $K(P)$. Both the operators inherit the monoidal structure and non-commutativity properties from the model.

Given the monoid $(M, \otimes, \epsilon)$, the set of simple terms Θ and the structural congruence relation $\equiv$, the logic BiLog$(M, \otimes, \epsilon, \Theta, \equiv)$ is formally defined in Tab. 3.2. The satisfaction relation $\models$ gives the

Table 3.2. *BiLog*$(M, \otimes, \epsilon, \Theta, \equiv)$

$\mathbf{\Omega}$ ::= $\mathbf{id}_I \mid \ldots$		a constant formula for every $\Omega \in \Theta$				
A, B ::= $\mathbf{F}$		false	$\mathbf{id}$	identity	$\mathbf{\Omega}$	constant
	$A \otimes B$	tensor product	$A \otimes\!- B$	left prod. adjunct	$A \mathrel{-\!\!\otimes} B$	right prod. adjunct
	$A \circ B$	composition	$A \ \ B$	left comp. adjunct	$A \multimap B$	right comp. adjunct
	$A \Rightarrow B$	implication				

$G \models \mathbf{F}$	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 $G_1 \models A$ and $G_2 \models B$
$G \models A \ \ B$	iff	for all G', the fact that $G' \models A$ and $(G' \circ G)\downarrow$ implies $G' \circ G \models B$
$G \models A \multimap B$	iff	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', if $G' \models A$ and $(G' \otimes G)\downarrow$ then $G' \otimes G \models B$
$G \models A \mathrel{-\!\!\otimes} B$	iff	for all G', if $G' \models A$ and $(G \otimes G')\downarrow$ then $G \otimes G' \models B$

semantics. The logic features a constant $\mathbf{\Omega}$ for each construct Ω and an identity $\mathbf{id}_I$ 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 $A \ \ B$ 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 holds for $\otimes\!-$ and $-\!\!\otimes$, the adjoints of $\otimes$. Clearly, these adjoints collapse whenever the tensor is commutative in the model.

3.3. Logical Equivalence

BiLog induces a logical equivalence $=_L$ on terms in the usual sense: we say that $G_1 =_L G_2$ when $G_1 \models A$ if and only if $G_2 \models A$ for every formula A. By induction on the structure of formulae, we can prove that the relation $=_L$ respects the congruence. We can prove that the logical equivalence coincides with the structural congruence, as every term admits a *characteristic formula*. This fact is fundamental to describe, query and reason about bigraphical data structures, as e.g. XML (cf. [11]). In other terms, BiLog is *intensional* in the sense of [25], namely it can observe internal structures, as opposed to the extensional logics used to observe the behaviour of dynamic system.

Theorem 3.1. (Logical equivalence is congruence)
$G =_L G'$ 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. The converse holds since $=_L$ respects the congruence. □

The logical equivalence may be less discriminating when there are constructors not directly represented by logical constants. The work in [12] show how the framework can be parameterised by a *transparency predicate* reflecting that not every term can be directly observed in the logic: some terms may be not visible to the logic or may be opaque without allowing inspection of their content. The particular characterisation of the logical equivalence given in Theorem 3.1 can be generalised to a congruence 'up-to-transparency': we can find an equivalence relation between trees that is 'tuned' by the transparency predicate – the more the predicate covers, the less the equivalence distinguishes [9, 18].

4. 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.

4.1. 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 the 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, ranged over by m, n. Ordinals represent the number of holes and regions of place graphs. Place graph terms are generated from the set

$$\Theta = \{1 : 0 \to 1,\ id_n : n \to n,\ join : 2 \to 1,\ \gamma_{m,n} : m + n \to n + m,\ \mathsf{K} : 1 \to 1 \text{ for } \mathsf{K} \in \mathcal{K}\}.$$

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. 4.1 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 $m \to 1$, namely it has a single region.

The Place Graph Logic PGL($\mathcal{K}$) is $BiLog(\omega, +, 0, \equiv, \mathcal{K} \cup \{1, join, \gamma_{m,n}\})$. Theorem 3.1 extends to PGL, thus the logic describes place graphs precisely. PGL resembles a propositional spatial tree logic as in [4], with the difference that PGL models contexts of trees and that the tensor product is not commutative, thus enabling the modelling of the order among regions. The logic can express a commutative separation by using $\mathbf{join}$ and $\otimes$, namely the *parallel composition* operator

$$A \mid B \stackrel{def}{=} \mathbf{join} \circ (\mathbf{id}_1 \circ A \otimes \mathbf{id}_1 \circ B).$$

At the term level, this separation, purely structural, corresponds to $join \circ (P_1 \otimes P_2)$, that is a total operation on all prime place graphs. More precisely, the semantics says that $P \models A \mid B$ if and only if there exist $P_1 : I_1 \to 1$ and $P_2 : I_2 \to 1$ such that: $P \equiv join \circ (P_1 \otimes P_2)$ and $P_1 \models A$ and $P_2 \models B$.

Table 4.1. *Additional Axioms for Place Graphs Structural Congruence*

Symmetry 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

Table 4.2. *Propositional Spatial Tree Logic*

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

Encoding STL. Not surprisingly, prime ground place graphs are isomorphic to the unordered trees that model the static fragment of Ambient Logic. Here we show that BiLog, when restricted to prime ground place graphs, is equivalent to the propositional Spatial Tree Logic of [4] (STL in the following). The logic STL expresses properties of unordered labelled trees, ranged over by T, T', T'' and constructed from the empty tree 0, the labelled node containing a tree $a[T]$, and the parallel composition of trees $T \mid T'$. Labels a are elements of a denumerable set. The obvious congruence $\equiv$ on trees makes the set of trees with $\mid$ and 0 a commutative monoid. STL is a static fragment of the Ambient Logic [8] 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 semantics of the logic is outlined in Tab. 4.2.

Table 4.3 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. We assume two auxiliary notations: $A \circ_1 B \stackrel{def}{=} A \circ \mathbf{id}_1 \circ B$ which forces the composition to the interface 1, and $A \ _1 \ B \stackrel{def}{=} (A \circ \mathbf{id}_1) \ \ B$, which guarantees terms with target type 1. The monoidal properties of parallel composition are guaranteed by the axioms of $join$ (symmetry and unit). 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 from trees 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 [21] as we shall see.

The theorem of discrete normal form in [21] implies that every ground place graph $g : 0 \to 1$ can be expressed, uniquely up to permutations, as $g = join_n \circ (M_0 \otimes \ldots \otimes M_{n-1})$, 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 $join_0 \stackrel{def}{=} 1$, and $join_{n+1} \stackrel{def}{=} join \circ (id_1 \otimes join_n)$.

Table 4.3. *Encoding STL in PGL over prime ground place graphs*

Trees into Prime Ground Place Graphs

$\llbracket 0 \rrbracket \stackrel{def}{=} \mathbf{1}$ $\quad \llbracket a[T] \rrbracket \stackrel{def}{=} \mathsf{K}(a) \circ \llbracket T \rrbracket$ $\quad \llbracket T \mid T' \rrbracket \stackrel{def}{=} join \circ (\llbracket T \rrbracket \otimes \llbracket T' \rrbracket)$

STL formulae into PGL formulae

$\llbracket \mathbf{0} \rrbracket \stackrel{def}{=} 1$ $\quad \llbracket \mathbf{F} \rrbracket \stackrel{def}{=} \mathbf{F}$ $\quad \llbracket a[A] \rrbracket \stackrel{def}{=} \mathsf{K}(a) \circ_1 \llbracket A \rrbracket$ $\quad \llbracket A \mid B \rrbracket \stackrel{def}{=} \llbracket A \rrbracket \mid \llbracket B \rrbracket$

$\llbracket A \Rightarrow B \rrbracket \stackrel{def}{=} \llbracket A \rrbracket \Rightarrow \llbracket B \rrbracket$ $\quad \llbracket A@a \rrbracket \stackrel{def}{=} \mathsf{K}(a) \ {}_1 \llbracket A \rrbracket$ $\quad \llbracket A \triangleright B \rrbracket \stackrel{def}{=} (\llbracket A \rrbracket \mid \mathbf{id}_1) \ {}_1 \llbracket B \rrbracket$

The bifunctoriality property implies that $join_n \circ (M_0 \otimes \ldots \otimes M_{n-1}) \equiv join \circ (M_0 \otimes \cdots \otimes (join \circ (M_{n-2} \otimes M_{n-1})))$. The inverse encoding $(\![\ ]\!)$ is defined on the discrete normal form of prime ground place graphs, and, along with $\llbracket\ \rrbracket$, it gives a bijection between trees and prime ground place graphs:

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

Theorem 4.1. (Encoding STL)
For each tree T and formula A of STL: $T \models_{\text{STL}} A$ if and only if $\llbracket T \rrbracket \models \llbracket A \rrbracket$.

Proof:
Structural induction on STL formulae. The basic step involves the constants $\mathbf{F}$ and $\mathbf{0}$. For $\mathbf{F}$ apply the definition. For $\mathbf{0}$: $\llbracket T \rrbracket \models \llbracket \mathbf{0} \rrbracket$ means $\llbracket T \rrbracket \models 1$, that is $\llbracket T \rrbracket \equiv 1$ and so $T \equiv (\![\llbracket T \rrbracket]\!) \equiv (\![1]\!) \stackrel{def}{=} 0$, namely $T \models_{\text{STL}} \mathbf{0}$. The inductive steps deal with connectives and modalities.

Case $A \Rightarrow B$**.** To assume $\llbracket T \rrbracket \models \llbracket A \Rightarrow B \rrbracket$ means $\llbracket T \rrbracket \models \llbracket A \rrbracket \Rightarrow \llbracket B \rrbracket$; by definition this says that $\llbracket T \rrbracket \models \llbracket A \rrbracket$ implies $\llbracket T \rrbracket \models \llbracket B \rrbracket$. 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 $a[A]$**.** To Assume $\llbracket T \rrbracket \models \llbracket a[A] \rrbracket$ means $\llbracket T \rrbracket \models \mathsf{K}(a) \circ_1 (\llbracket A \rrbracket)$. Then there exist $G : 1 \to 1$ and $g : 0 \to 1$ such that $\llbracket T \rrbracket \equiv G \circ g$ and $G \models \mathsf{K}(a)$ and $g \models \llbracket A \rrbracket$, that is $\llbracket T \rrbracket \equiv \mathsf{K}(a) \circ g$ with $g \models \llbracket A \rrbracket$. Then $T \equiv (\![\mathsf{K}(a) \circ g]\!) \stackrel{def}{=} a[(\![g]\!)]$ with $g \models \llbracket A \rrbracket$, as the encoding is bijective. Since $g : 0 \to 1$, the induction says that $(\![g]\!) \models A$. Hence $T \models_{\text{STL}} a[A]$.

Case $A@a$**.** To assume $\llbracket T \rrbracket \models \llbracket A@a \rrbracket$ means $\llbracket T \rrbracket \models \mathsf{K}(a) \ {}_1 \ A$, which says that if $(G \circ \llbracket T \rrbracket)\!\downarrow$ then $G \circ \llbracket T \rrbracket \models \llbracket A \rrbracket$, for every G such that $G \models \mathsf{K}(a)$. By definition, this is $\mathsf{K}(a) \circ \llbracket T \rrbracket \models \llbracket A \rrbracket$, then $\llbracket a[T] \rrbracket \models \llbracket A \rrbracket$. By induction, this is $a[T] \models_{\text{STL}} A$. Hence $T \models_{\text{STL}} A@a$ by definition.

Case $A \mid B$**.** To assume $\llbracket T \rrbracket \models \llbracket A \mid B \rrbracket$ means $\llbracket T \rrbracket \models \llbracket A \rrbracket \mid \llbracket B \rrbracket$. This is equivalent to say that $\llbracket T \rrbracket \models \mathbf{join} \circ (\mathbf{id}_1 \circ \llbracket A \rrbracket \otimes \mathbf{id}_1 \circ \llbracket B \rrbracket)$, namely there exist $g_1, g_2 : 0 \to 1$ such that $\llbracket T \rrbracket \equiv join \circ (g_1 \otimes g_2)$ and $g_1 \models \llbracket A \rrbracket$ and $g_2 \models \llbracket B \rrbracket$. 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 $A \triangleright B$**.** To assume $\llbracket T \rrbracket \models \llbracket A \triangleright B \rrbracket$ means $\llbracket T \rrbracket \models \mathbf{join}(\llbracket A \rrbracket \otimes \mathbf{id}_1)) \ {}_1 \ \llbracket B \rrbracket$ – i.e., for every $G : 1 \to 1$ if $G \models \mathbf{join}(\llbracket A \rrbracket \otimes \mathbf{id}_1)$ then $G \circ \llbracket T \rrbracket \models \llbracket B \rrbracket$. Now, $G : 1 \to 1$ and $G \models \mathbf{join}(\llbracket A \rrbracket \otimes \mathbf{id}_1)$ means that there exists $g : 0 \to 1$ such that $g \models \llbracket A \rrbracket$ and $G \equiv join(g \otimes id_1)$. Hence for every $g : 0 \to 1$ such that $g \models \llbracket A \rrbracket$ it holds $join(g \otimes id_1) \circ \llbracket T \rrbracket \models \llbracket B \rrbracket$, that is $join(g \otimes \llbracket T \rrbracket) \models \llbracket B \rrbracket$ by bifunctoriality. As $\llbracket\ \rrbracket$ is bijective, for every T' such that $\llbracket T' \rrbracket \models \llbracket A \rrbracket$ it holds $join(\llbracket T' \rrbracket \otimes \llbracket T \rrbracket) \models \llbracket B \rrbracket$ – i.e., $\llbracket T' \mid T \rrbracket \models \llbracket B \rrbracket$. By induction, for every T' such that $T' \models_{\text{STL}} A$ it holds $T' \mid T \models_{\text{STL}} B$, then $T \models_{\text{STL}} A \triangleright B$. □

Table 4.4. *Additional Axioms for Link Graph Structural Congruence*

Link Node Axiom:	$\alpha \circ \mathsf{K}_{\vec{a}} \equiv \mathsf{K}_{\alpha(\vec{a})}$	Renaming
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

Differently from STL, PGL can also describe structures with several holes and regions. In [11] we show that PGL can describe 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 $\mathbf{join}$. Similarly, a 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 the node K and their second one as a sibling of such node.

4.2. 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 disjoint union of subsets. Link graphs are the structures arising from such a monoid. They can describe nominal resources, which are common in many areas: object identifiers, location names in memory structures, channel names, and ID attributes in XML documents. Due to the disjoint union, names cannot be shared implicitly; anyway, they can be referred to or linked explicitly (e.g. as 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 ${}^{a}/_{X}$ 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 $a \stackrel{def}{=} {}^{a}/_{\emptyset}$, which introduces the name a, in $a \leftarrow b \stackrel{def}{=} {}^{a}/_{\{b\}}$, which renames b to a, and in $a \leftleftarrows b \stackrel{def}{=} {}^{a}/_{\{a,b\}}$, which links a and b to a.

Another class of constructors for link graphs are the controls K in the signature $\mathcal{K}$. They represent nodes associated to names, and they have fixed *arities* to determine the number of ports – i.e., the number of the outer names associated to the control. Arity is given by the arity function $ar : \mathcal{K} \to \mathbb{N}$. Link graphs are generated from wirings and the constructors $\mathsf{K}_{\vec{a}} : \emptyset \to \vec{a}$ 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. The structural congruence $\equiv$ is refined as outlined in Tab. 4.4 with the obvious axioms for links, which model α-conversion and extrusion of closed names. In this case, the horizontal decomposition inherits the commutativity property from the monoidal tensor product.

The Link Graph Logic LGL($\mathcal{K}$) is $BiLog(\mathcal{P}_{fin}(\Lambda), \uplus, \emptyset, \equiv, \mathcal{K} \cup \{/a, {}^{a}/_{X}\})$. Theorem 3.1 extends to LGL, thus the logic describes the link graphs precisely. On the one hand, the logic expresses structural spatiality for resources and strong spatiality (separation) for names, therefore it can be seen as a generalisation of a separation logic for contexts and multi-ports locations. On the other hand, the logic describes

resources with local (hidden or private) names, and in this sense it is a generalisation of Spatial Graph Logic [6]. To see this it is sufficient to consider the edges as resources.

The formula $A \otimes B$ describes a decomposition into two *separate* link graphs, which share neither resources, nor names, nor connections, and which 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.

If we want a name a to be shared between separated resources, we need to make the sharing explicit, and we can do that only 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 shown next.

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 on $\vec{a}$ is

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

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

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 [21]. 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 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 $/a$. In fact, the effect of the 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. This connective not only separates places, but also ensures 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 closure has the effect of hiding names. For this task, we employ the name variables $x_1, ..., x_n$ and the fresh name quantification N in the style of Nominal Logic [24]. The semantics is defined as

$$G \models \mathsf{N}x_1 \ldots x_n.\, A \textit{ iff there exist } a_1 \ldots a_n \notin \mathit{fn}(G) \cup \mathit{fn}(A) \textit{ such that } G \models A\{^{x_1 \ldots x_n}/_{a_1 \ldots a_n}\},$$

where $A\{^{x_1 \ldots x_n}/_{a_1 \ldots a_n}\}$ 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

$$\vec{a}\,\mathbf{L}\,\vec{x}.\, A \stackrel{def}{=} \mathsf{N}\vec{x}.\, ((\vec{a} \leftleftarrows \vec{x}) \otimes \mathbf{id}) \circ A.$$

This formula expresses that the variables $\vec{x}$ in A denote names that are linked in the term to $\vec{a}$, and the role of $(\vec{a} \leftleftarrows \vec{x})$ is to link the fresh names $\vec{x}$ to $\vec{a}$, while $\mathbf{id}$ deals with names not in $\vec{a}$. We also define a

Table 4.5. *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

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:

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

The idea expressed by this formula 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. It is straightforward to prove that the two definitions are consistent.

Lemma 4.1. (Separation-up-to)
If $g \models A \otimes^{\vec{X}} B$ 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 $g \equiv g_1 \otimes^{\vec{X}} g_2$ with $g_1 \models A$ and $g_2 \models B$.

Proof:
Apply the definitions and observe that the identities forcing $\mathbf{id}$ must be id_ϵ, as the outer face of g is X. □

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

Encoding SGL. Here we show that LGL can be seen as a contextual (multi-edge) version of Spatial Graph Logic (SGL) [6], which expresses properties of directed graphs G with labelled edges. The syntax of graphs outlined in Tab. 4.5. 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 Spatial Graph Logic combines the standard propositional logic with two structural connectives: composition and basic edge. Although we focus on its propositional fragment, the logic of [6] also includes edge label quantifier and recursion. In [6] SGL is used as a pattern matching mechanism of a query

Table 4.6. *Propositional Spatial Graph Logic (SGL)*

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

$G \models_{\text{STL}} \mathbf{F}$ 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 $G \equiv G_1 \mid G_2$

Table 4.7. *Encoding Propositional SGL in LGL over ground link graphs*

Spatial Graphs into Two-ported Ground Link Graphs

$[\![nil]\!]_X \stackrel{def}{=} X$ $\qquad$ $[\![(\nu x)G]\!]_X \stackrel{def}{=} ((/x \otimes id_{X\setminus\{x\}}) \circ [\![G]\!]_{\{x\}\cup X})) \otimes (\{x\} \cap X)$

$[\![a(x,y)]\!]_X \stackrel{def}{=} \mathsf{K}(a)_{x,y} \otimes X \setminus \{x,y\}$ $\qquad$ $[\![G \mid G']\!]_X \stackrel{def}{=} [\![G]\!]_X \otimes^{\vec{X}} [\![G']\!]_X$

SGL formulae into LGL formulae

$[\![\mathbf{nil}]\!]_X \stackrel{def}{=} X$ $\qquad$ $[\![a(x,y)]\!]_X \stackrel{def}{=} \mathsf{K}(a)_{x,y} \otimes (X \setminus \{x,y\})$ $\qquad$ $[\![\varphi \mid \psi]\!]_X \stackrel{def}{=} [\![\varphi]\!]_X \otimes^{\vec{X}} [\![\psi]\!]_X$

$[\![\mathbf{F}]\!]_X \stackrel{def}{=} \mathbf{F}$ $\qquad$ $[\![\varphi \Rightarrow \psi]\!]_X \stackrel{def}{=} [\![\varphi]\!]_X \Rightarrow [\![\psi]\!]_X$

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 4.6 depicts the syntax and the semantics of the fragment we consider.

For the encoding, we consider a signature $\mathcal{K}$ with controls of arity 2, and we assume a bijective function associating every label a to a distinct control $\mathsf{K}(a)$. The ports represent the starting and arrival node of the associated edge. 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 4.7 encodes the graphs that model SGL into ground link graphs and SGL formulae into LGL formulae. The encoding is parametric on a finite set X of names which contains the free names of the graph. As we force the outer face of the graphs to be a fixed set X, the encoding of parallel composition is simply a separation-up-to the elements $\vec{X}$ of X. Local names are encoded into name closures. The Connected Normal Form of [21] helps in proving that ground link graphs featuring controls with exactly two ports are isomorphic to spatial graph models.

Lemma 4.2. (Isomorphism for spatial graphs)
There is a mapping $(\![\ ]\!)$ from two-ported ground bigraphs to spatial graphs, such that for every set X of names: *(i)* the mapping $(\![\ ]\!)$ is inverse to $[\![\ ]\!]_X$; *(ii)* for every ground link graph g with outer face X featuring a countable set of controls K, all with arity 2, it holds $fn((\![g]\!)) = X$ and $[\![(\![g]\!)]\!]_X \equiv g$; and *(iii)* for every spatial graph G with $fn(G) = X$ it holds $[\![G]\!]_X : \epsilon \to X$ and $(\![[\![G]\!]_X]\!) \equiv G$.

Proof:
Represent link graphs as bigraphs of type $\epsilon \to \langle 1, X \rangle$ without nested nodes, which, as proved in [21], have the normal form $G = (/Z \mid id_{\langle 1,X\rangle}) \circ (X \mid M_0 \mid \ldots \mid M_{k-1})$, where $Z \subseteq X$ and $M ::=$

$\mathsf{K}_{x,y}(a) \circ 1$. The inverse encoding is based on this normal form:

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

The encodings $[\![\ ]\!]$ and $(\![\ ]\!)$ provide a bijection between graphs with free names X and ground link graphs with outer face X and controls of arity two. □

Theorem 4.2. (Encoding SGL)
For every graph G, every finite set X that contains $fn(G)$, and every formula φ of the propositional fragment of SGL: $G \models_{\mathrm{SGL}} \varphi$ if and only if $[\![\, G \,]\!]_X \models [\![\, \varphi \,]\!]_X$.

Proof:
By induction on formulae of SGL. The basic steps deal with the constants $\mathbf{F}$, $\mathbf{nil}$ and $a(x,y)$.The case $\mathbf{F}$ follows by definition. For the case $\mathbf{nil}$, $[\![\, G \,]\!]_X \models [\![\, \mathbf{nil} \,]\!]_X$ means $[\![\, G \,]\!]_X \models X$, that by definition is $[\![\, G \,]\!]_X \equiv X$ and so $G \equiv (\![\, [\![\, G \,]\!]_X \,]\!) \equiv (\![\, X \,]\!) \stackrel{def}{=} nil$, namely $G \models_{\mathrm{SGL}} \mathbf{nil}$. 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 $G \models_{\mathrm{SGL}} a(x,y)$. The inductive steps deal with connectives.

Case $\varphi \Rightarrow \psi$**.** $[\![\, G \,]\!]_X \models [\![\, \varphi \Rightarrow \psi \,]\!]_X$ means $[\![\, G \,]\!]_X \models [\![\, \varphi \,]\!]_X \Rightarrow [\![\, \psi \,]\!]_X$ that is: $[\![\, G \,]\!]_X \models [\![\, \varphi \,]\!]_X$ implies $[\![\, G \,]\!]_X \models [\![\, \psi \,]\!]_X$. By induction, this means that $G \models_{\mathrm{SGL}} \varphi$ implies $G \models_{\mathrm{SGL}} \psi$ – i.e., $G \models_{\mathrm{SGL}} \varphi \Rightarrow \psi$.

Case $\varphi \mid \psi$**.** $[\![\, G \,]\!]_X \models [\![\, \varphi \mid \psi \,]\!]_X$ means $[\![\, G \,]\!]_X \models [\![\, \varphi \,]\!]_X \otimes^{\vec{X}} [\![\, \psi \,]\!]_X$. By Lemma 4.1 there exists g_1, g_2 such that $[\![\, G \,]\!]_X \equiv g_1 \otimes^{\vec{X}} g_2$ and $g_1 \models [\![\, \varphi \,]\!]_X$ and $g_2 \models [\![\, \psi \,]\!]_X$. Let $G_1 = (\![\, g_1 \,]\!)$ and $G_2 = (\![\, g_2 \,]\!)$, then $[\![\, G_1 \,]\!]_X \equiv g_1$ and $[\![\, G_2 \,]\!]_X \equiv g_2$. Hence $[\![\, G_1 \,]\!]_X \models [\![\, \varphi \,]\!]_X$ and $[\![\, G_2 \,]\!]_X \models [\![\, \psi \,]\!]_X$. By induction: $G_1 \models_{\mathrm{SGL}} \varphi$ and $G_2 \models_{\mathrm{SGL}} \psi$. Now, $[\![\, G_1 \mid G_2 \,]\!]_X \equiv [\![\, G_1 \,]\!]_X \otimes^{\vec{X}} [\![\, G_2 \,]\!]_X \equiv g_1 \otimes^{\vec{X}} g_2 \equiv [\![\, G \,]\!]_X$, thus $G \models_{\mathrm{SGL}} \varphi \mid \psi$. □

LGL enables the encoding of Separation Logic for 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 §4.3 we will encode a more general logic: the Context Tree Logic [5].

4.3. Pure Bigraph Logic

By combining link graphs and place graphs we generate all the *(abstract pure) bigraphs* of [17]. In this case the underlying monoid is the product of link and place interfaces, namely $(\omega \times \mathcal{P}_{fin}(\Lambda), \otimes, \epsilon)$ where $\langle m, X\rangle \otimes \langle n, X\rangle \stackrel{def}{=} \langle m+n, X \uplus Y\rangle$ and $\epsilon \stackrel{def}{=} \langle 0, \emptyset\rangle$. 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. Only controls are subsumed by the new *discrete ion* constructors, which are denoted by $\mathsf{K}_{\vec{a}} : 1 \to \langle 1, \vec{a}\rangle$ and represent prime bigraphs 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 [21].

The structural congruence for bigraphs corresponds to the sound and complete bigraph axiomatisation of [21]. The additional axioms are reported in Tab. 4.9: they are essentially a combination of the axioms for link and place graphs, with slight differences due to the monoid of interfaces.

Table 4.8. *Additional axioms for Bigraph Structural Congruence*

Symmetry 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

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 this two logics is 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, next section shows how the Context Logic for Trees (CTL) [5] can be encoded into bigraphs.

Encoding CTL. In [5] the authors propose 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, T', T'' and *linear contexts* C, C', C'' which are trees with a unique hole. Every node has a name, so to identify memory locations. 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. 4.9: 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 by filling the unique hole of C with T. A *well formed tree* or *context* is one where the node identifiers are unique. The model of the logic is composed by well formed trees and contexts. Composition, node formation and tree insertion are *partial* as they are restricted to well-formed trees. The structural congruence between trees and contexts is the smallest congruence that makes the parallel operator to be commutative, associative and with the empty tree as neutral element.

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. 4.10, the semantics in Tab. 4.11.

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 consider its fragment without links. As already said, the terms giving a semantics to CTL do not to share identifiers, as an identifier represents a precise

Table 4.9. *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
$T \mid 0 \equiv T$			neutral element
$T \mid T' \equiv T' \mid T$			commutativity
$(T \mid T') \mid T'' \equiv T \mid (T' \mid T'')$			associativity
$(T \mid T')\downarrow$			if and only if identifiers of T and T' are disjoint
similarly for contexts			

Table 4.10. *Context Tree Logic (CTL)*

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

location in the memory. This property is obtained with bigraphs by encoding the identifiers as names and the composition as tensor product. This structure is then encoded in BiLog by lifting the application between contexts and trees to a particular kind of composition.

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 be fixed and leaving the name part to be free: $\mathbf{id}_{\langle m,\_\rangle} \stackrel{def}{=} \mathbf{id}_m \otimes (\mathbf{id} \wedge \neg(\mathbf{T} \otimes \mathbf{id}_1 \otimes \mathbf{T}))$. The semantics says that $G \models \mathbf{id}_{\langle m,-\rangle}$ if and only if 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 $A \circ_{\langle m,\_\rangle} B \stackrel{def}{=} A \circ \mathbf{id}_{\langle m,\_\rangle} \circ B$ and the typed adjuncts: $A \ {}_{\langle m,\_\rangle} B \stackrel{def}{=} (\mathbf{id}_{\langle m,\_\rangle} \circ A) \ B$ and $A \multimap_{\langle m,\_\rangle} B \stackrel{def}{=} (A \circ \mathbf{id}_{\langle m,\_\rangle}) \ B$.

We then encode the operator for the parallel composition by the separation operator $*$ defined as both a term constructor and a logical connective: $G * G' \stackrel{def}{=} [join](G \otimes G')$ on prime bigraphs, and

$$A * B \stackrel{def}{=} (\mathbf{join} \otimes \mathbf{id}_{\langle 0,\_\rangle}) \circ (\mathbf{id}_{\langle 1,\_\rangle} \circ A \otimes \mathbf{id}_{\langle 1,\_\rangle} \circ B)$$

on formulae. The operator $*$ enables the encoding of trees and contexts into bigraphs. Finally, we consider a signature with controls of arity 1 and we assume a bijective function from tags to controls: $a_x \longmapsto \mathsf{K}(a)_x$. The details of the encoding are outlined in Tab. 4.12. The encodings of trees are the *ground prime discrete bigraphs* – i.e., bigraphs with open links and type $0 \to \langle 1, X\rangle$.

Table 4.11. *Semantics for CTL*

$T \models_{\mathcal{T}} \mathit{false}$	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, $T \equiv C(T'), C \models_{\mathcal{K}} K$ 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 imply $C(T) \models_{\mathcal{T}} P$
$T \models_{\mathcal{T}} P \Rightarrow P'$	iff	$T \models_{\mathcal{T}} P$ implies $T \models_{\mathcal{T}} P'$
$C \models_{\mathcal{K}} \mathit{false}$	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, $C \equiv a_x[C']$ 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 imply $C(T) \models_{\mathcal{T}} P'$
$C \models_{\mathcal{K}} P \mid K$	iff	there exist C', T s.t. $T \mid C'$ well-formed, $C \equiv T \mid C', T \models_{\mathcal{T}} P$ and $C' \models_K K$
$C \models_{\mathcal{K}} K \Rightarrow K'$	iff	$C \models_{\mathcal{K}} K$ implies $C \models_{\mathcal{K}} K'$

Paper [21] shows 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 $M = (\mathsf{K}(a)_x \otimes id_Y)g$. Thanks to this result, we can define the reverse encoding $(\![\ ]\!)$ of $[\![\ ]\!]$, from ground prime discrete bigraphs to trees, just by considering such a normal form:

$$\begin{array}{rcl} (\![\, join_0 \,]\!) & \stackrel{def}{=} & 0 \\ (\![\, (\mathsf{K}(a)_x \otimes id_Y) \circ g \,]\!) & \stackrel{def}{=} & a_x[\,(\![\, g \,]\!)\,] \\ (\![\, (join_k \otimes id_Y) \circ (M_1 \otimes \ldots \otimes M_k) \,]\!) & \stackrel{def}{=} & (\![\, M_1 \,]\!) * \ldots * (\![\, M_k \,]\!) \end{array}$$

Moreover, the encodings of linear contexts are the *unary discrete bigraphs* G – i.e., bigraphs with open links and type $\langle 1, X\rangle \to \langle 1, Y\rangle$. Again, [21] says 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 $(\![\ ]\!)$ of $[\![\ ]\!]$, from unary discrete bigraphs to linear contexts, just by considering such a normal form:

$$\begin{array}{rcl} (\![\, id_1 \,]\!) & \stackrel{def}{=} & - \\ (\![\, (\mathsf{K}(a)_x \otimes id_Y) \circ Q \,]\!) & \stackrel{def}{=} & a_x[(\![\, Q \,]\!)] \\ (\![\, (join_k \otimes id_Y) \circ (R \otimes M_1 \otimes \ldots \otimes M_{k-1}) \,]\!) & \stackrel{def}{=} & (\![\, R \,]\!) \mid (\![\, M_1 \,]\!) \mid \ldots \mid (\![\, M_{k-1} \,]\!) \end{array}$$

As the model is specialised to context trees, so BiLog is specialised to the Context Tree Logic. The encoding of the logic is in Tab. 4.12, and the proof of soundness mimes Theorems 4.1 and 4.2.

Theorem 4.3. (Encoding Context Tree Logic)
For each tree T and formula P of CTL, $T \models_{\mathcal{T}} P$ if and only if $[\![\, T \,]\!] \models [\![\, P \,]\!]_P$. Moreover, for each context C and formula K of CTL, $C \models_{\mathcal{K}} K$ if and only if $[\![\, C \,]\!]_C \models [\![\, K \,]\!]_K$.

The encoding shows that the models in [5] 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. [5], and it is used to characterise programs that manipulate tree structured memory model, BiLog may express Separation Logic as well.

Table 4.12. *Encoding CTL in BiLog over prime discrete ground bigraphs*

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

5. BiLog and separation

The notion of separation in BiLog is not fixed a priori, but it relies on the projection of the tensor product to the logical level. The operators of the model, along with their logical projections, inherit the characteristics and the behaviour from the underlying resource monoid. This fact is fundamental to express the notions of separation/composition of other logics. When the resource monoid is partial, then the tensor product of terms is partial as well, and thus its logical projection describes only well-defined structures. Take for instance the heap memory addresses formalised by sets of names and consider their disjoint union. In this case we find the operator $*$ of Separation Logic, which says nothing about the structures that do not have disjoint set of names, because they are not well-defined heaps. When the resource monoid is commutative, then also the tensor product is commutative, its logical projection is commutative and its two adjuncts collapse.

When we consider ordinals (as for place graphs) the tensor product is total, as it corresponds to placing two structures one next to the other, and the separation is total and purely structural, like for the Spatial Tree Logic. Notice that a commutative monoid combined with bifunctoriality has a peculiar behaviour with respect to composition: for instance, given the controls K_1 and K_2, commutativity and bifunctoriality would say that $(\mathsf{K}_1 \circ \mathsf{K}_1) \otimes (\mathsf{K}_2 \circ \mathsf{K}_2)$ and $(\mathsf{K}_1 \circ \mathsf{K}_2) \otimes (\mathsf{K}_2 \circ \mathsf{K}_1)$ are congruent as both are congruent to $(\mathsf{K}_1 \otimes \mathsf{K}_2) \circ (\mathsf{K}_1 \otimes \mathsf{K}_2)$. Thus commutativity cannot hold when controls are intended as places. While the parallel composition for ground structures is commutative, the tensor product for placing contexts is not commutative. Consequently, the left and right tensor adjuncts have different semantics in place graphs.

When ordinals and names are combined, the logic inherits their orthogonality. The resulting monoid can be seen as the product of two monoids, the one for ordinals (e.g. place graph) and the other for names (e.g. link graph). The constructive character of this approach says that, independently of the complexity of the underlying algebra and the resource model, we can split the model into 'smaller' parts and combine the corresponding logics to obtain a spatial logic for the desired model in a compositional way.

We do not explicitly encode Separation Logic and pointers of Context Logics in BiLog as we focus on pure bigraphs. As a matter of fact, it would be sufficient to include a two sorts policy [2] that differentiates input ports (the addresses) from output ports (the pointers/values) to obtain a partial monoid for

heaps, and thus to have a natural encoding of Separation Logic. The separation induced by the tensor product would be more strict than heap separation, as it forces also values to be separated. The heap composition can be achieved by forcing the sharing of pointers and values and by preserving the separation of addresses. The derived logical operators would behave as the tensor product on the names in the first sort (the addresses) and as a (kind of) parallel composition for the names in the second sort (the values/pointers). We do not necessarily need closure or restriction to model heaps. Open link graphs are sufficient. Similarly, open discrete bigraphs are sufficient for trees with pointers.

6. Conclusions and related 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 Petri-nets [20], CCS [22], π-calculus [17], ambient calculus [16], and context-aware systems [1]. Here we introduce the static fragment of 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 connectives regulated by elementary monoidal and interchange laws. This structure gives rise to many – occasionally complex – derived connectives. This is a fundamental expressiveness property that does not put us off, as BiLog is meant to be a comprehensive meta-level framework in which several different logics can be isolated, understood and compared.

In particular, here show 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 LGL 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 [23], and trees with either pointers [5] or hidden names [7].

The work in [10] proves that unrestricted sharing combined with name restriction makes the logic undecidable. The sharing/separation operator provided here hints that the real cause of undecidability is the quantification on the set of names and suggests that the decidability result of [4] can be extended to logics with explicit sharing and name revelation. We leave the this issue for future work.

In §3 we observed that the induced logical equivalence coincides with the structural congruence of terms. This property is fundamental to describe, query and reason about bigraphical data structures. For a more detailed discussion we refer to [11], where we sketch the application of BiLog to XML data.

To be as free as possible in choosing the level of intensionality and to 'tune' the power of the logical equivalence, the general definition for BiLog (c.f. [12]) is parameterised on a *transparency* predicate, whose role is to identify the terms allowing inspection of their content, *transparent* terms, and the ones that do not, *opaque* terms. Theorem 3.1 says that the logical equivalence is the structural congruence if every term is transparent, but [9, 18] show how BiLog becomes less discriminating with opaque terms.

Moreover, in [9, 18] we show how BiLog can deal with dynamics. A natural solution is to add a temporal modality basically describing bigraphs that can evolve (*react*) according to a Bigraphical Reactive System [17]. When the transparency predicate enables the inspection of 'dynamic' controls, BiLog

is '*intensional*' in the sense of [25], as it can observe internal structures. In the case of the bigraphical system describing CCS [22], 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 [15]. Finally the papers [13, 14] suggest applications and extensions for BiLog.

References

[1] Birkedal, L., Debois, S., Elsborg, E., Hildebrandt, T., Niss, H.: Bigraphical Models of Context-aware Systems, *FOSSACS*, 2006.

[2] Birkedal, L., Debois, S., Hildebrandt, T.: Sortings for Reactive Systems, *CONCUR*, 2006.

[3] Caires, L., Cardelli, L.: A Spatial Logic for Concurrency (Part I), *TACS*, 2001.

[4] Calcagno, C., Cardelli, L., Gordon, A.: Deciding Validity in a Spatial Logic for Trees, *TLDI*, 2003.

[5] Calcagno, C., Gardner, P., Zarfaty, U.: A Context Logic for Tree Update, *POPL*, 2005.

[6] Cardelli, L., Gardner, P., Ghelli, G.: A Spatial logic for querying graphs, *ICALP*, 2002.

[7] Cardelli, L., Gardner, P., Ghelli, G.: Manipulating Trees with Hidden Labels, *FOSSACS*, 2003.

[8] Cardelli, L., Gordon, A.: Ambient Logic, *Mathematical Structures in Computer Science*, To appear.

[9] Conforti, G.: *Spatial Logics for Semistructured Resources*, PhD Thesis, Univ. of Pisa, 2005.

[10] Conforti, G., Ghelli, G.: Decidability of Freshness, Undecidability of Revelation, *FOSSACS*, 2004.

[11] Conforti, G., Macedonio, D., Sassone, V.: Bigraphical Logics for XML, *SEBD*, 2005.

[12] Conforti, G., Macedonio, D., Sassone, V.: Spatial Logics for Bigraphs, *ICALP*, 2005.

[13] Damgaard, T., Birkedal, L.: Axiomatizing binding bigraphs, *Nordic Journal of Computing*, **13**(1), 2006.

[14] Grohmann, D., Miculan, M.: Directed Bigraphs, *MFPS*, 2007.

[15] Hirschkoff, D.: An Extensional Spatial Logic for Mobile Processes, *CONCUR*, 2004.

[16] Jensen, O.: Forthcoming PhD Thesis, Aalborg Univ.

[17] Jensen, O., Milner, R.: *Bigraphs and mobile processes (revised)*, Tech. rep., Univ. of Cambridge, 2004.

[18] Macedonio, D.: *Logics for Distributed Resources*, PhD Thesis, Univ. Ca' Foscari of Venice, 2006.

[19] Milner, R.: Bigraphical Reactive Systems, *CONCUR*, 2001.

[20] Milner, R.: Bigraphs for Petri nets, *Lectures on Concurrency and Petri Nets: Advances in Petri Nets*, 2004.

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

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

[23] O'Hearn, P., Reynolds, J., Yang, H.: Local Reasoning about Programs that Alter Data Structures, *CSL*, 2001.

[24] Pitts, A.: Nominal Logic: a First Order Theory of Names and Binding, *TACS*, 2001.

[25] Sangiorgi, D.: Extensionality and Intensionality of the Ambient Logic, *POPL*, 2001.