Bigraphical Logics for XML

Giovanni Conforti
Università di Pisa

Damiano Macedonio
Università Ca' Foscari di Venezia

Vladimiro Sassone
University of Sussex

Abstract

Bigraphs are emerging as an interesting model that can represent both the pi-calculus and the ambient calculus. Bigraphs are built orthogonally on two structures: a hierarchical 'place' graph for locations and a 'link' (hyper-)graph for connections. In a previous work (submitted elsewhere and yet unpublished), we introduced a logic for bigraphical structures as a natural composition of a place graph logic and a link graph logic. Here we show that fragments of BiLogic can be used to describe XML data (with `ID` and `IDREF`s) and to reason about programs that manipulate tree-structures with query-oriented update operators.

1 Introduction

The term 'spatial,' as opposed to 'temporal,' has been recently used to refer to logics providing modal operators to express properties of the structure of the model. Such logics are usually equipped with a separation/composition operator that *splits* the current model into two parts, in order to 'talk' about them separately. Looking closely, we observe that notion of *separation* is interpreted differently in different logics.

- In 'separation' logics [23, 21], the separation is used to reason about dynamic update of tree-like structures, and it is *strong* in that it forces names of resources and pointers in separated components to be disjoint. In addition, this constraint usually implies that model composition must partially defined.
- In static spatial logics (e.g. for trees [4], graphs [7] or trees with hidden names [8]), the separation/composition operator is *structural*, and it is used to describe properties of the underlying structure. In this case no constraint on the model is usually required, and *names* may be shared between separated parts.
- In dynamic spatial logics (e.g. for ambients [10] or π-calculus [3]), the separation is intended only for location in space, and names can be shared between separated resources.

Research partially supported by '**MyThS**: Models and Types for Security in Mobile Distributed Systems', EU FET-GC IST-2001-32617 and by '**DisCo**: Semantic Foundations of Distributed Computation', EU IHP 'Marie Curie' HPMT-CT-2001-00290.

Context tree logic, recently introduced in [5], integrates the first approach above with spatial logics for trees. The resulting logic is 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.

Bigraphs are an emerging model for structures in global computing, which can be instantiated to model several well-known examples, including the π-calculus, the ambient calculus and Petri nets [19]. Bigraphs consist essentially of a tree-structured place graph – representing the ambient-like nesting of locations – coupled with a link (hyper-)graph on the same nodes – representing the π-calculus-like communication channels. Such structure has recently been axiomatized in categorical terms and by means of a term algebra in [20]. In [16], we build on such bi-structural nature to introduce a '*contextual spatial logic*' for bigraphs built on two orthogonal sublogics:

- a *place graph logic*, to express properties of resource locations;
- a *link graph logic*, to express connections between resources (or, more precisely, resource names).

For this reason, we name the formalism *Bilogic*. Bilogic is interesting for at least two reasons. Firstly, the place graph logic is a generalization of both *spatial logics* and *context-tree* logic. Secondly, it captures at the same time both the notion of structural and of strong separation. In particular, we are able to express a notion of *separation-upto* that specifies which names must (or should) be shared. In addition, we are currently undertaking to extend Bilogic with temporal modalities, so as to model bigraphical reactive systems (BRS) and therefore generalize dynamic spatial logics (as, e.g., the Ambient Logic). Thus, Bilogic and its subcalculi are very general and promising as logics to talk about resources with names.

XML data are essentially tree-shaped resources, and have been modelled with unordered labelled tree in [6] where an important connection between semistructured data and mobile ambients was uncovered. Starting from *loc. cit.*, several works on spatial logic for semistructured data and XML have been proposed (e.g. [7, 17, 8]). Among these, a query language on semistructured data based on Ambient Logic was studied in [9] and implemented in [12, 15]. The present paper enriches over such model of tree-shaped data by adding links on resource names, so as to obtain a more general model for semistructured data and XML (which, as a matter of fact, it is closer to the standard OEM model). A similar step was taken in [11], which we improve upon by making use of the well-studied categorical structure of bigraph, which internalize the notion of link and makes the difference between strong and structural separation explicit. In addition, bigraphs naturally model XML contexts: we thus obtain with no additional effort a logic to describe XML contexts which can be interpreted as web services or XML transformations.

Here we focus on the applications of bigraphical logics to XML data. In particular, we first show how XML data (and, more generally, contexts or positive web services) can be interpreted as a bigraph. Equipped with such 'bigraphical' representation of XML data and contexts, we then give a gentle introduction to different fragments of Bilogic and show how they can be applied to describe and reason about XML. The

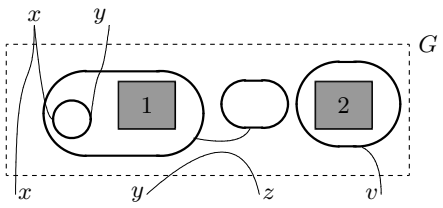

Figure 1: Bigraph G

contribution of the paper is therefore to identify (fragments of) Bilogic as a suitable formalism for semistructured data, and illustrate its expressiveness by means of selected examples.

Structure of the paper. In §2 we recall the basic notions of bigraphs; for a fuller exposition the reader is referred to [19]; §3 shows how to interpret XML contexts as bigraphs, while §4 illustrates several examples.

2 Bigraphs

We give a crush introduction to bigraphs [19]. We restrict our attention exclusively to *abstract pure bigraph*, whose algebraic axiomatization has been provided in [20].

The bigraph G in Fig.1 is a structure built on two orthogonal graphs over the same group of nodes, shown here with bold outlines. Nodes may be nested in a hierarchical tree structure (the *place graph*, shown as the inclusion of a node in another), and have *ports* that may be connected by *links* (the *link graph*, shown as edges connecting nodes). The nesting of nodes imposes no constrain upon the linkage of their ports, hence the orthogonality of the structures.

Every node of the place graph has a *control*, which represent what kind of node it is and its *arity*, i.e., an ordinal telling the number of ports and marking them unambiguously. In the Figures 1 and 2, arity is one for oval shaped nodes, two for the round ones, three for the triangular one.

At the top level of the nesting structure are the *roots*. in the figure there is a unique root, shown as a dotted outline. In general however there may be multiple roots. Inside a node there may be *holes*, shown as shaded boxes in the figure, which formalize *contexts*. Roughly speaking, a place graph is a list of (unordered) trees that can have *holes* as leaves. Place graphs are characterized by a couple of ordinals, written as $m \to n$, denoting respectively the number of holes and the number of roots in the bigraph. The use of ordinals allows us to mark holes and roots uniquely.

The link graph is characterized by a couple of set of names $X \to Y$. The set X represents the *inner names* (drawn in the figure below the bigraph) and Y represent the set of *outer names* (drawn above the bigraph). The link graph connects a port to a name or an *edge*. In Fig. 1 an edge is represented by a line between nodes. Ports can be associated both to names and to edge, in any finite number. A link to a name is

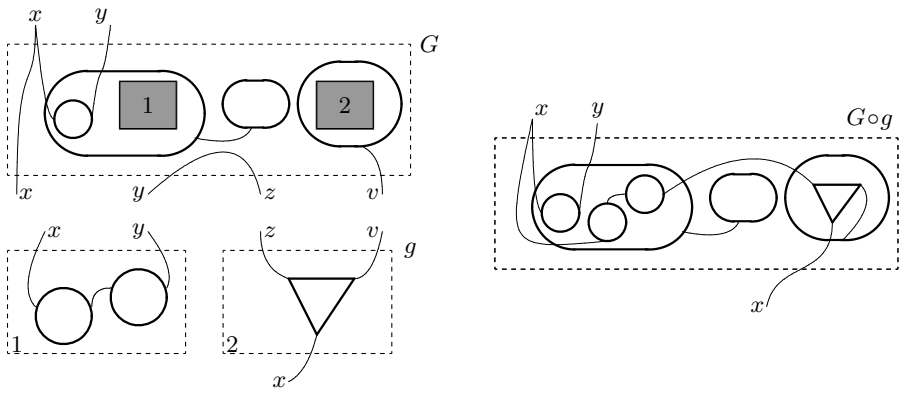

Figure 2: Bigraphical composition

open and may be connected to other nodes in case of composition between bigraphs. A link to an edge is *closed* and cannot be connected to other ports. The role of names becomes clearer after introducing the composition operator of bigraphs.

The right hand side of Fig. 2 represents the *composition* $G \circ g$ between the two bigraphs G and g described on the left hand side. The main idea is to insert g into the context G. The operation is partially defined, since it requires the inner names and the number of holes of G to match to the number of roots and the outer names of g respectively. Shared names create the new links between the two structures. Intuitively, composition *first* places root region of g in the proper hole of G and *then* joins equal inner names of G and outer names of g. In particular note the edge connecting the inner names *y*,*z* in G, its presence produces a link between two internal nodes of g after the composition.

As explained in [19], the tensor product$\otimes$of bigraphs is defined only if they do not have common inner names or outer names. By relating names to resources, the tensor can be seen a separation operator: the product puts the place graphs one next to the other (in order), obtaining a graph with more roots and holes, and operates the disjoint union between inner names and outer names of the place graphs.

3 Modelling XML Contexts as Bigraphs

As proved in [20], the class of bigraphs can be axiomatized using a small set of elements. We recall the constructions below, and then relate it to XML. In our formalization, XML data are bigraphs with no holes (i.e., *ground*), while those with holes represent XML contexts.

The main constituent of a bigraph is the *discrete ion* $K_{\vec{a}}$, which represents a node with one root and containing one hole. The hole can be filled with other ions in order to build a more complex tree-structure. The ion's control is K, with arity $ar(K) = |\vec{a}|$,

and every port of $K_{\vec{a}}$ is linked to a name in the (ordered) list of names $\vec{a}$. Every name in $\vec{a}$ represent an outer name of the bigraph. Thinking in terms of XML data, a ion is seen as a *tag* with some *attributes*. Since arity is an ordinal, it is possible to identify the ports unambiguously and it is easy to associate them to attributes. We assume one designed port to be associated to a (unique) name used to identify the element, as an `ID` attribute. Other ports may be linked to other nodes' `ID` names, so acting effectively as `IDREF`s, or to internal edges connected to internal nodes, representing the general attributes of the element. Embedding a ion into the hole of another ion, represents the inclusion of the corresponding elements.

The basic place graphs are 1, id_n and *merge*. Term 1 is the empty single rooted bigraph, it is the empty XML document; id_n is a context with n holes, n roots and no internal node. It behaves like a neutral XML context if composed with a compatible XML document. Term *merge* achieves the merging of two bigraphs: it consists of two holes, one root and no links. By composing merge with the product of two single-rooted XML documents we obtain a XML document whose single root has the two component documents as children.

The basic components of a link graph are $(a \leftarrow b)$, $(a \leftleftarrows b)$ and $/a$. Operator $(a \leftarrow b)$ represents renaming, and in the context of our interpretation of XML, it acts on `ID` attributes. Operator $(a \leftleftarrows b)$ associates name b to name a: when a represent an `ID` and b a `IDREF`, then $(a \leftleftarrows b)$ makes b a reference to a. Finally, $/a$ makes name a private, and allows nodes to be joined to one another in closed links. This operator will be used in our encoding of XML to express a link between attributes and their values.

In the presence of (unfilled) holes, terms represent *contexts* for XML data, i.e., documents with holes to be filled by XML data: composition acts by inserting documents in such holes. The tensor is defined only if the names appearing in the two components are disjoint. Therefore, any reference 'going across' must be created after the product. Note that since link graphs in general perform substitution and renaming, the outer names of g may not be outer names of $G \circ g$. This may happen either because they are renamed or because an edge has been added to the structure as effect of the composition, which makes the link *private*, i.e., without an externally-visible name.

The parent-child relationship on nodes in bigraphs does not capture order among children of the same node. So bigraphs can be seen as a (ordered) list of unordered (contexts of) trees connected through links. This model can be used for XML data whose document order is not relevant. Such documents arise for instance in XML encodings of relational databases [2], in the integration of semi-structured database sources, or in the case of distributed XML documents in a P2P environment [22] (in the latter case the document order is not defined).

The importance of the underlying hierarchical structure in XML, and the fact that links are used sporadically only for modelling relations between nodes, suggests the bigraphical model as a good model for XML documents. We interpret these documents as ground bigraphs by using the encoding explained below. The encoding is trivial in case the tree contains no attribute, when we can in fact easily map the tree structure of XML elements into the place graph by associating controls to tags and values. In this case, there is no link between nodes, all controls have arity zero, and the XML file is completely modelled by the place graph only (in a kind of ambient like formalism [6]).

In the case of elements with attributes, we need names to represent XML links

between elements (e.g., like ID-IDREF relationships), and edges to represent elements' attributes. We consider the IDs used in XML data as names in bigraphs. The encoding is defined by assuming two functions on values:

- $K_{val}(v)$, mapping the value v to a ground bigraph corresponding to a single rooted node with no outer names, no nodes and no holes inside.
- $K_{val}(v)_a$, mapping the value v to a ground bigraph corresponding to a single rooted node with outer name a, no nodes and no holes inside.

The former function is used actually to encode values with bigraphs, the latter is auxiliary and encodes values linked to attributes.

Moreover we associate tags with ions. We assume a class $\mathcal{K}_{tag}$ of controls. We consider a tag t and firs we observe that the list *Att* of its attributes is finite and ordered, hence we associate the list to an ordinal *#Att*, and the elements of the list are identified by their position. Then we associate t with $K_{tag}(t, \#Att)_{\vec{u}}$, that is a ion with control $K_{tag}(t, \#Att) \in \mathcal{K}_{tag}$ and arity *#Att*. The vector $\vec{u}$ indicates the names connected to the control; we assume the names in $\vec{u}$ to be the IDs associated to the attributes in *Att*.

A value attribute is encoded as a value inside the node and connected to the port whose position marks the corresponding attribute. Identifiers (like ID) and links (like IDREF) attributes have a special interpretation. They become *names* of the tag and can be connected with other names in order to model references. As mentioned before, the connection is performed by using the link graphs constructors: $(a \Leftarrow b)$, to create a reference, and $/a$, to create a closed connection for attributes. The general definition for the encoding is formalized in Tab. 1

Table 3.1. *XML documents as ground bigraphs*

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

In the table above, the encoding of values is simply the function $K_{val}(\,)$. The auxiliary encoding of values linked to attributes is given by $K_{val}(\,)_a$. Term 1 corresponds to the empty tree. The core of the translation is the encoding of (non empty) trees. Here, the role of *merge* is to group together the (encodings of the) set of children of T and the (encodings of the) values linked to attributes. In this case, values linked to attributes are associate with a name. Observe in the encoding the use the renamings α_i to guarantee the product is defined, since it requires the names to be distinct. We choose fresh names, i.e., not appearing in T, and we obtain the renamings α_i by combining different operators such as $(a \leftarrow b)$. The obtained bigraph is single rooted, hence it fits in the ion associated to the tag t. After the composition with the ion, we have to rename the names in order to formalize all the references, finally we need to close the link between the root and the evaluations of the values linked to attributes. The renaming is obtaining by considering the inverse of α_i (definable by using operators such as $(a \leftarrow b)$ and $(a \leftleftarrows b)$), and the closure is obtained by combining the closure of every name associated to an attribute.

Example. As an example of the encoding of XML trees into bigraphs, we consider a database that stores scientific papers and information about their authors. We focus on the fragment quoted in the document below.

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

Tag *Author* has an identifier, $\texttt{ID}_i$, a link to another author, `coauth`, that is an `IDREF` attribute, and a general attribute, `name`. The encoding is illustrated in Fig. 3. Every tag *Author* is associated to a control of arity three. Exploiting the order of the ports, we identify a port with the corresponding XML attribute unambiguously. In the picture we assume the ports ordered clockwise. The first port corresponds to the general attribute `name`, and is connected by a close link (an edge) to a value. The second one corresponds to the identifier, `ID`, and is connected to an outer name. The final attribute corresponds to the reference , `coauth`, and is connected to a name that correspond to another *Author* tag.

More generally a bigraph can be seen as a context for unordered XML data, just because there can be holes in it. So in the previous example we can imagine to put holes in place of some node. This yields a context that can be interpreted as a contextual

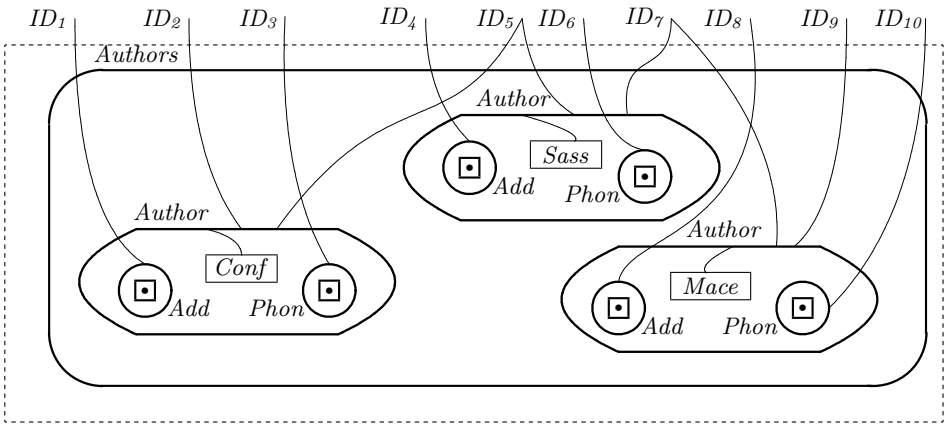

Figure 3: XML encoding

XML document, a function (*Web service*) that takes a list of XML files and returns their composition in context, by fitting every file in the relative position (as marked by its position on the list). In this way we can model Web services, besides plain XML documents. In order to model the order of XML elements, we would need to add a notion of ordered locality, i.e., to consider molecules with an ordered "list" of holes, and extend the theory of bigraphs accordingly to this notion.

4 Bilogics for XML Contexts

In [16] we introduced Bilogic for bigraphical structures. In §3 we have shown that XML (unordered) data and contexts can be modelled as a bigraphical structures. This section briefly introduces Bilogic, and explains how it can be used for describing, querying and reasoning about XML. In particular, we analyze three possible cases:

- Logics for place graphs to model XML data and contexts without `IDs`;
- Logics for *discrete bigraphs* (essentially trees with unique identifiers) for XML with `IDs`, but without links;
- Bigraphical logics for XML with `IDs` and links.

4.1 XML without `IDs`

As mentioned previously, without attributes – or anyway handling them as children elements themselves – XML amounts to unordered labelled tree. In [6] the author shows that such a model has some similarities with ambient calculus terms, building on which [9] introduces a query language for semistructured data based on Ambient

Logic. In [16] we show that the static fragment of ambient logic (STL) can be easily extended to the Place Graph Logic (PGL in the following) to model general contexts of tree-shaped resources. In particular PGL can describe place graphs, that is bigraphs without links, and so it can be used to talk about XML contexts (without attributes) using the encoding we defined in the previous section. We briefly present here some operators of PGL, and we informally show their semantics in the XML case. (Some of connectives are derived; the reader is referred to [16] for the details.)

Table 4.1. *PGL: Place Graph Logic (some operators)*

$A, B ::=$	formulas
$\mathbf{F}$	false
$A \Rightarrow B$	implication
$\mathbf{1}$	empty single rooted bigraph
$\mathbf{id}_n$	identity on n number of holes (even zero)
$A \otimes B$	decomposing in two place graphs one next to the other
$A \circ_s B$	possible s-holes filling of A with a place graphs satisfying B
$A \circ\!\!-_s B$	if inserted in a context A (with s holes) then B
$K[A]$	the molecule K containig something satisfying A
$A \mid B$	decomposing in two trees whose merge is the current model

$\mathbf{F}$ and $A \Rightarrow B$ are the standard propositional operators (the other propositional connectives $\mathbf{T}$, $\neg$, $\wedge$, $\vee$, $\Leftrightarrow$ are derived as usual). There are spatial constants $\mathbf{1}$, **merge**, and id_n denoting a singleton place graph (interpreted as a single XML context). We interpret $\mathbf{1}$ to be the empty XML context, **merge** the context merging two XML contexts in one, while **id** is the identity context, which transforms XML trees to themselves. The two spatial operators $A \otimes B$ and $A \circ_s B$ express two ways of composing contexts. The first is *horizontal* and produces a (ordered and separated) pair of contexts one next to the other. The second one is *vertical* and corresponds to fill the s holes of a context satisfying A with the context satisfying B. They are both non commutative. Then, there are ambient-like operators for trees: $K[A]$ is the context that inserts a new root labelled K in the top of a single XML context satisfied by A, and $A \mid B$ (parallel composition) denotes contexts obtained by merging the tree contexts satisfying A and B in a single root. Note that, since parallel composition performs a merge of the contexts, it provides a commutative monoid with $\mathbf{1}$ as neutral element. An interesting connective is $A \circ\!\!-_s B$, which essentially expresses that whenever the current model is inserted inside a XML s-ary context satisfying A, then the resulting context satisfies B.

In general, models of PGL are *positive* functions from m to n that given a list of m XML contexts produces a list of n XML contexts. By 'positive' we mean that they can only add structure to the parameters, and not remove or replace parts of them. In this sense, XML contexts are viewed as positive XML Web services that take XML documents (possibly with calls to other Web services, so that they effectively are XML contexts), and return XML documents. This is similar to the model of Positive Active XML proposed in [1], but with a remarkable difference: since our model does not handle ordered trees, we cannot restrict attention to functions between XML (active) documents. We need to use with *list* of parameters and a *list* of resulting contexts. To

understand better the idea, consider the Web service below.

$$wb : K_1[id_1] \mid K_2[merge \circ id_2]$$

It takes three trees and puts the first inside a node labelled K_1, merges the second and third trees and puts the result inside a node labelled K_2, and finally produces the parallel composition of the two resulting trees. We need ordered parameters to put the right root in the right hole. A Web service like this can be solely identified by a characteristic formula (corresponding to the tree), but more generally a formula like $K_1[id_1] \mid \mathbf{T}$ can match all Web services having at least one hole and decomposable as a node of arity one labelled K_1 in parallel to something else. In this sense a notion of *type* for Web services arises. Similarly to [13], where the spatial tree logic is used to describe XML types and constraints, we can use PGL to formalize Web service types and constraints.

Since also XML (active) documents are contexts, we can actually use the PGL to describe Active XML documents and Web service in an unique framework. In addition, we can use an approach like TQL [9] to query Active XML documents and Web service, and eventually use types to avoid Web service useless invocations. To make the idea more precise, with reference to the previous example, take a query $wb \circ (T_1 \otimes T_2 \otimes T_3) \models K_1[X] \mid \mathbf{T}$, which essentially determines all contexts reachable from the result of the Web service invocation through a path $/K_1$. If we know that the type of wb is $K_1[\mathbf{id}_1] \mid K_2[\mathbf{T}]$ we can avoid to evaluate the web service by observing that $(K_1[\mathbf{id}_1] \mid K_2[\mathbf{T}]) \circ\!\!-\!\!\!\!\!\!\supset_3 K_1[X] \mid \mathbf{T}$, and so:

$$wb \circ (T_1 \otimes T_2 \otimes T_3) \models K_1[X] \mid K_2[\mathbf{T}] \quad \Longleftrightarrow \quad T_1 \models \mathbf{id} \circ X.$$

4.2 XML Contexts with `ID`

In the previous section we focused on the place structure only. Since logic and model have no way to directly identify resources, it is only possible to access a resource through navigation. A different approach is possible when the XML document has identifiers for and pointers to elements. In this case, the tree model can be seen as an extension of a heap memory model in which locations are referred to by names. Such names are intrinsically separated by the tensor product, which is defined only on structures which disjoint name sets. We can see such models as discrete bigraphs, i.e., place graphs with named resources but no name sharing between different resources. A logic for these is introduced in [16] as an extension of the PGL with named (identified) controls K_x and renamings $(x \leftarrow y)$. Such a logic is able to express properties of (contexts of) resources that can be accessed in two ways: as usual, by navigation through the tree structure, and by using names controls as pointers.

The logic essentially adds two operators to PGL:

$$\begin{array}{rl} K_{\vec{a}} & \text{for named nodes;} \\ (a \leftarrow b) & \text{for renaming these names.} \end{array}$$

The ion $K_{\vec{a}}$ has a list of names, although in the case of XML with identifiers and no links only one name is needed. Thus, we write K_x to denote the node (with an hole) inside labelled K with name identifier x, and the formula K_x denotes this XML context

only. The rename ($a \leftarrow b$) is needed in order to map names of different sources to different identifiers (e.g., $(x \leftarrow y) \circ K_y = K_x$). The tensor product now constraints the models to be separated both in locality and in names, i.e., when we write $A \otimes B$ we mean that the models satisfying A and B have disjoint sets of identifiers (that is disjoint outer faces). On the other hand the composition $A \circ_s B$ is defined when the inner face of A and outer face of B coincide.

With this logic we have two ways to address separation: by means of the tensor product, or by using the composition. For example we can think to encode a heap-like structure as $(x_1 \mapsto a_1) \# ... \# (x_n \mapsto a_n)$ in two ways:

- *horizontally*, as a one level forest $K(a_1)_{x_1} \circ 1 \otimes \ldots \otimes K(a_n)_{x_n} \circ 1$;
- *vertically*, as a line $K(a_1)_{x_1} \circ \ldots \circ K(a_n)x_n \circ 1$. (This matches with [5], where separation logic can be expressed using composition only.)

where $K(a_i)_{x_i}$ is a ion associated to an a_i, and 1 fills the holes.

While the first approach can be lifted to a commutative monoid if we use the merge operation, the latter cannot be interpreted in a commutative way directly. In this sense discrete bigraphs are more general than heaps of the form above (as used in [23, 21]). We believe we can extend the result of [16], viz. the encoding of Context Tree Logics on unidentified nodes into PGL, to Discrete Bigraphs Logics, so as to model the entire Context Tree Logic of [5], including identifiers. Thus, this logic can also be used to reason about programs in tree-shaped memory models.

4.3 XML Contexts with Links

In order to model sharing of names between resources and treat structures with pointers, we have to extend the logic of discrete bigraph with a notion of sharing. The sharing is obtained in bigraphs through links between names of resources. In our case, we have encoded identifiers as tag names and `IDREFs` as pointers to names in the same document. In [16] we have introduced a logic for general bigraphs as a composition of a link graph logic and a place graph spatial logic. Such a combination is very expressive, and induces a hiding operator for local/private/hidden names. For the present application to XML this is only needed for the encoding of value attributes. On the other hand, we require a notion of separating conjunction with sharing, in order to express properties like: "*The author of paper X has a relationship with the author of paper Y.*" In fact, this property expresses separation on resources (different authors of different papers), but sharing on linked names. Such operator is explicitly introduced in [16] by using the tensor product of Bilogic, the renaming function and the *freshness* operator of nominal logics. The main idea is that a link between names can be seen as a separation between separated names that are then linked by means of substitution. In [14, 8] a notion of link similar to this is hidden in the model because the ambient-like operation shares names by default (which may be the main reason for undecidability of logics with abstract names).

5 Conclusions

In the paper we have sketched the application of Bilogic to describe and reason about XML data. This is however not the main reason why Bilogic and bigraphs are interesting for XML and other global resources in general. Bigraphs were introduced basically to model dynamic concurrent systems (cf. [18]), where they are used as a contextual way to specificy reaction rules. We believe that Bilogic, inheriting the flexibility and universality of such model, will help creating a general logic framework uniformly applicable to several actual calculi. The study of the case of XML was initiated here, and in the future we plan to extending to more sophisticated semistructured data models.

Acknowledgment We would like to thank Philippe Bidinger, Robin Milner and Peter O'Hearn for useful discussions. A special thank to Carlo Sartiani.

References

[1] S. Abiteboul, O. Benjelloun, and T.Milo. Positive active xml. In *Proc. of PODS 2004*, 2004.

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

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

[4] C. Calcagno, L. Cardelli, and A. D. Gordon. Deciding validity in a spatial logic for trees. In *Proc. of ACM SIGPLAN Workshop on Types in Language Design and Implementation (TLDI'03)*, 2003.

[5] 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.

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

[7] 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.

[8] L. Cardelli, P. Gardner, and G. Ghelli. Manipulating trees with hidden labels. In *Proc. of FOSSACS '03*, volume 2620 of *LNCS*, pages 216–232. Springer-Verlag, 2003.

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

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

[11] L. Cardelli, P.Gardner, and G. Ghelli. Querying trees with pointers. Unpublished notes.

[12] G. Conforti, O. Ferrara, and G. Ghelli. TQL Algebra and its Implementation (Extended Abstract). In *Proc. of IFIP TCS*, pages 422–434. Kluwer Academic Publishers, 2002.

[13] G. Conforti and G. Ghelli. Spatial logics to reason about semistructured data. In *Proc. of SEBD 2003: Eleventh Italian Symposium on Advanced Database Systems*. Rubettino Editore, 2003.

[14] G. Conforti and G. Ghelli. Decidability of freshness, undecidability of revelation. In *Foundations of Software Science and Computation Structures, FOSSACS 2004*, pages 105–120. Springer, 2004.

[15] G. Conforti, G. Ghelli, A. Albano, D. Colazzo, P. Manghi, and C. Sartiani. The Query Language TQL. In *Proc. of 5th International Workshop on Web and Databases (WebDB 2002)*, 2002.

[16] G. Conforti, D. Macedonio, and V. Sassone. Bilogics: Spatial-nominal logics for bigraphs (extended abstract). Submitted for publication. Available from `http://www.di.unipi.it/~confor/publications.html`, October 2004.

[17] Silvano Dal Zilio and Denis Lugiez. A logic you can count on. In *POPL 2004 – 31st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2004.

[18] O. H. Jensen and R. Milner. Bigraphs and transitions. In *Proc. of the 30th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 38–49. ACM Press, 2003.

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

[20] R. Milner. Axioms for bigraphical structure. Technical Report UCAM-CL-TR-581. University of Cambridge, February 2004.

[21] Peter O'Hearn, John C. Reynolds, and Hongseok Yang. Local reasoning about programs that alter data structures. In *In Proc. of CSL*, volume 2142 of *LNCS*, pages 1–19. Springer-Verlag, 2001.

[22] Sigmod record volume 3 number 1, 2004. special topic section on peer to peer data management, 2004.

[23] John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Proc. LICS'02*, pages 55–74. IEEE Computer Society, 2002.