



The IT University  
of Copenhagen

# **Bigraphs and (Reactive) XML**

**an XML-centric model of computation**

**Thomas Hildebrandt**  
**Jacob W. Winther**

**Copyright © 2005, Thomas Hildebrandt  
Jacob W. Winther**

**IT University of Copenhagen  
All rights reserved.**

**Reproduction of all or part of this work  
is permitted for educational or research use  
on condition that this copyright notice is  
included in any copy.**

**ISSN 1600-6100**

**ISBN 87-7949-084-0**

**Copies may be obtained by contacting:**

**IT University of Copenhagen  
Glentevej 67  
DK-2400 Copenhagen NV  
Denmark**

**Telephone: +45 38 16 88 88  
Telefax: +45 38 16 88 99  
Web [www.itu.dk](http://www.itu.dk)**

# Bigraphs and (Reactive) XML an XML-centric model of computation

Thomas Hildebrandt and Jacob W. Winther

Dept. of Theoretical Computer Science  
IT University of Copenhagen  
Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark  
{hilde, jww}@itu.dk

**Abstract.** XML-centric models of computation have been proposed as an answer to the demand for interoperability, heterogeneity and openness in coordination models. Recently, *Bigraphical reactive systems* has been introduced as a theoretical meta-model for *reactive* systems with semi-structured state. We present an XML representation of bigraphical reactive systems called *Reactive XML*. It may be seen as an open, XML-centric model of computation with a solid theoretical foundation given by the theory of bigraphical reactive systems. In particular, the theory provides a formal denotation of composition and general transformations of XML documents with links. On the other hand, Reactive XML provides a concrete understanding of the theoretical model of bigraphical reactive systems. We report on a prototype for Reactive XML, describing how Reactive XML reactions can be carried out in praxis. In particular, we describe how the abstract notion of evaluation contexts may be represented concretely (and generalised) by XPath expressions. The prototype is currently being extended to a distributed setting, ultimately allowing distributed XML-centric applications to coordinate through computations on shared XML data.

## 1 Introduction

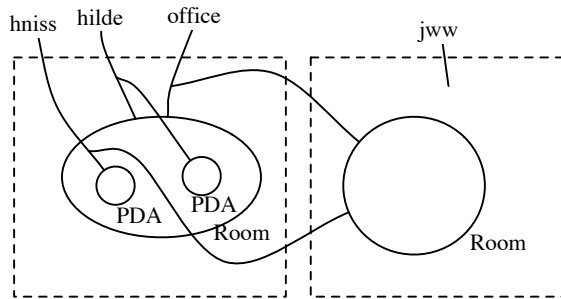
It has early on been observed that models for mobile computation, such as the Ambient calculus [1], describe reconfiguration of spatial structures similar to the data model for semi structured data [2]. This observation has been taken further to suggest XML-centric models of computation, where computation is expressed in terms of transformations of XML data [3]. The use of XML meets the demand for interoperability, heterogeneity and openness in globally distributed applications [4], and coordination languages in particular. One of the reasons why XML has become the de facto standard for exchange of semi structured data is that it is a general meta language, in which one can define domain specific languages and benefit from general XML tools and techniques for processing them. However, as usual the development and use of the technology is ahead of the formal understanding.

*Bigraphical reactive systems* [5] has recently been introduced as a theoretical meta model for reactive, mobile systems with semi structured state. By semi structured state we mean a hierarchically structured state possibly with links across the hierarchical structure. To be precise, a bigraph consists of two finite graphs, the topo graph and the link graph. The topo graph is a tuple of unordered trees. The roots of the trees are referred to as *regions* and the nodes are often referred to as *places*. Used as a model for mobile systems, the topo graph will typically describe the structural composition of the system state. Each place is typed with a *control* and has a number of *ports*. Ports are linked together by the link graph. Some of the links may be *open*, meaning that they are connected to a name in the *outerface* of the bigraph. This allows links to be joined with other links when bigraphs are composed. A bigraph is called open if all links are open. For simplicity, we will mainly consider open bigraphs throughout the paper.<sup>1</sup> In Fig. 1 is shown an open bigraph with two regions (marked with dotted lines).

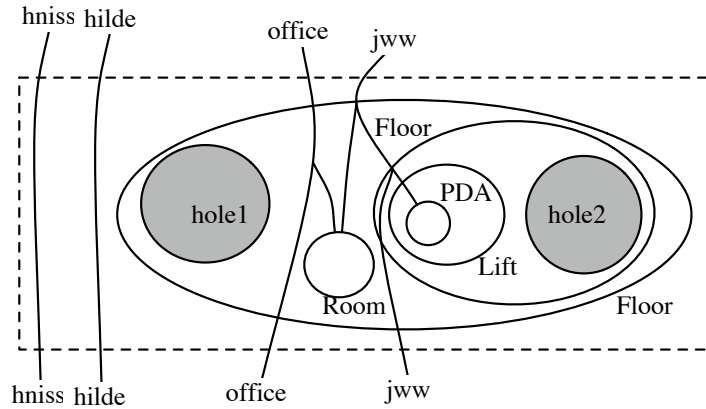
<sup>1</sup> Open and closed links correspond to respectively free and local names in process calculi.

In general, a bigraph may contain an ordered set of *holes* among the leaves of the place graph. A bigraph with  $n$  holes can then be composed (vertically) with a bigraph with  $n$  regions, by placing the contents of the regions of the latter in the holes of the former. Thus, bigraphs with holes are *contexts*. A bigraph without any holes is referred to as a *ground* bigraph. An example of a bigraph context is shown in Fig. 2. The set of names below the bigraph are referred to as the (names of the) *innerface* of the bigraph.

The composition of the two bigraphs is shown in Fig. 3. Note how contents of different regions may be placed in different places, and how links to the innerface of the context are joined with links to the outerface of the bigraph placed in the holes.

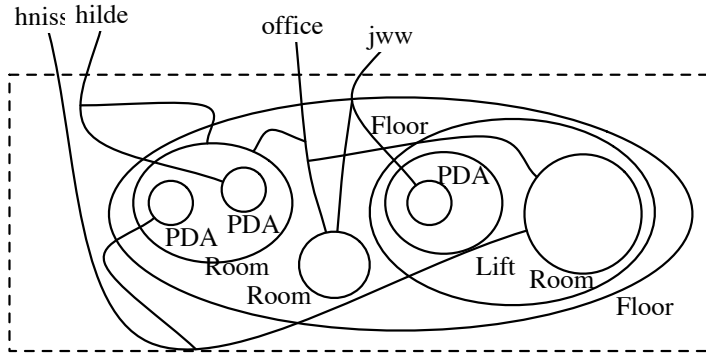


**Fig. 1. A ground bigraph Rooms with two regions**

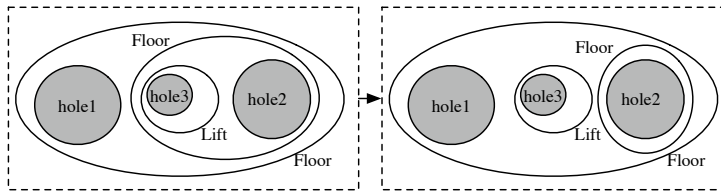


**Fig. 2. A bigraph context Building with two holes.**

Analogously to XML, one may define domain specific bigraph languages by specifying the allowed controls, their nesting and linking. However, in addition to the specification of the valid system states, the theory of bigraphical reactive systems also provides a general format for specification of reaction rules, that is, the *behaviour* of systems. A bigraphical reaction rule consists essentially of a pair  $(L,R)$  of bigraphs, allowing to replace any complete subtree  $L$  with the subtree  $R$  in a bigraph. In general, one specifies a finite set of *parametric* reaction rules, where the two bigraphs  $L$  and  $R$  are contexts, corresponding to



**Fig. 3. The (vertical) composition  $\text{Building} \circ \text{Rooms}$ .**



**Fig. 4. Reaction rule, moving the lift one floor down.**

the usual rule schemas used in operational semantics for process calculi. An example of a parametric reaction rule is given in Fig. 4. The reaction rule moves the lift (and its contents) one floor down.

The examples given above are purely artificial. However, one could imagine a bigraphical language for context aware applications in which context information was represented like in the examples above. Reaction rules for coordinating activities could then depend on e.g. the location of a PDA running a particular application, or the presence of other PDAs in the same location. One could even imagine each PDA having its own set of reaction rules, or that the rules changed over time. To make this real, one would need a concrete representation of the bigraph state, which could be shared across a network, and an implementation of the reaction rules working on this representation.

In the present paper we suggest such a representation of bigraphical reactive systems, which is called *Reactive XML* and is based on the obvious similarities between bigraphs and XML. We describe how the theory of bigraphical reactive systems provides a formal definition of XML contexts, insertion of XML documents in contexts and of XML reactions. One of the strong points of the theory of bigraphical reactive systems is that it automatically provides a notion of labelled bisimulation congruence for any particular bigraphical reactive system. This means that for any language compiled into Reactive XML one would get a labelled bisimulation congruence. Although the bisimulation may be intractable in practice, this provides a good starting point for developing reasoning techniques.

Finally, it is worth noting that Reactive XML provides a concrete understanding of the theoretical model of bigraphical reactive systems. This will hopefully help bringing the theory in work, and may also lead to extensions or revisions of the theory. As an instance of this, the work on Reactive XML suggests a new notion of evaluation contexts than the ones usually applied in process calculi, using XPath expressions to specify the contexts in which reactions may take place.

*Structure of the paper* In Sec. 2 we first give the representation of ground bigraphs as XML documents, using a process calculus notation for bigraphs. We focus on open (pure) bigraphs, but also briefly touch on how to represent closed binding bigraphs and how to represent XML constant data in bigraphs. In Sec. 3 we extend the calculus to (bigraph) contexts and define context composition. Using contexts we define parametric reaction rules. In Sec. 4 we briefly describe how reaction rule matches are found in the prototype, using XPath expressions to describe evaluation contexts. Finally in Sec. 5 we conclude and give pointers for future work.

*Related work:* The recent paper [6] reports on independent work relating bigraphs and XML. As for Reactive XML the aim in [6] is to use the theory of bigraphs as a theoretical foundation for XML processing. However, the focus of [6] is purely on XML as *data* and the use of bigraph-logics (introduced previously by the authors [7]) to describe properties of data. Reactive XML focuses on *reactive systems* and thus on the theory of bigraph *reactions* which is not considered in [6]. The difference also shows up in the subset of XML documents considered. [6] provides a bigraph representation of XML documents close to the OEM data model, where each node has a unique ID reference which can be referred to from any other node. Conversely, Reactive XML provides an XML representation of bigraphs, and thus consider XML documents where the uses of IDREF and ID attributes correspond to respectively names and name binders in process calculi. An obvious joining point of the two lines of work would be to use bigraph-logics for describing properties of reactive XML documents.

The paper [8] introduces the process calculus  $Xd\pi$  based on the  $\pi$ -calculus aimed for modelling XML-centric peer-to-peer systems, and investigates its bisimulation semantics. As such it can be seen as a specific reactive XML language. A good test of Reactive XML would be to describe  $Xd\pi$  and its semantics in Reactive XML and compare the derived bisimulation semantics to the one in [8].

Active XML [9] provides a language and foundation for *active XML* documents. Active XML documents primarily support dynamic inclusion of XML data produced by web-services. It does not in itself provide a foundation for XML as reactive processes. The ideas of Active XML could be used jointly with Reactive XML in a distributed implementation.

The paper [3] surveys several middleware technologies for XML-centric applications which could serve as inspiration to how Reactive XML could be used in a distributed setting. They make a distinction between reactive and proactive XML, which is not to be confused with the use of reactive XML in this paper.

## 2 Bigraphs and XML

In this section we give the representation of ground (open) bigraphs as XML.

Recall that an XML document can be typed by a Document Type Definition (DTD). A DTD defines restrictions on the nesting of elements, the allowed attributes of elements and their types. The most important attribute types are character data (CDATA), unique id (ID) and references to an id (IDREF). For a document to be well typed according to a DTD all attributes typed ID must be uniquely valued. Moreover, for any attribute typed IDREF, there must exist an attribute typed ID with the same value.

We will define a simple abstract notion of signatures that specialises to both simple DTDs and bigraph signatures.

**Definition 1.** *A signature is a tuple  $(\Sigma, Att, ar)$ , where the set  $\Sigma$  is a set of element names,  $Att$  is a set of finite attribute sets, and  $ar : \Sigma \rightarrow Att$  is a function assigning a finite attribute set (of attribute names) to each element.*

A signature corresponds to a simple DTD that imposes no restrictions on the nesting of elements. The map  $ar$  is specifying the set of (names of) attributes for each element. Types

of attributes may be represented in the signature by taking attribute sets to be finite subsets of a disjoint union of (infinite) sets of character data names, identifiers and references, that is  $Att = \mathcal{P}_{fin}(CDATA \uplus ID \uplus IDREF)$ . In that case we will refer to the signature as an *XML signature*.

We consider XML documents with a distinguished *root* element having a set of ID attributes, and children elements being a number of *region* elements. We assume that  $root, region \notin \Sigma$  and the remaining elements belongs to  $\Sigma$  for some XML signature  $(\Sigma, Att, ar)$ . Depending on the allowed types of attributes of elements in  $\Sigma$  we identify different classes of XML documents which denote corresponding classes of ground (open) bigraphs.

**Definition 2.** An XML signature  $(\Sigma, \mathcal{P}_{fin}(CDATA \uplus ID \uplus IDREF), ar)$  is

- pure if  $\forall \kappa \in \Sigma. ar(\kappa) \subset IDREF$ ,
- binding if  $\forall \kappa \in \Sigma. ar(\kappa) \subset ID \uplus IDREF$
- pure with constants if  $\forall \kappa \in \Sigma. ar(\kappa) \subset CDATA \uplus IDREF$  and
- binding with constants otherwise

Pure and binding XML signatures correspond to respectively pure and binding signatures of bigraphs as defined in [5]. XML signatures are slightly more general, since bigraphs abstract from names of attributes, and just take attribute sets to be finite ordinals. Thus, for  $Att = \omega$  (the set  $\{\{0, \dots, n-1\} \mid n \geq 0\}$  of finite ordinals), the signatures are exactly pure bigraph signatures as defined in [5], except element names are called *controls* for bigraphs and attributes are called *ports*. Similarly, signatures of binding bigraphs are the signatures with  $Att = \omega \times \omega$ . The set  $\omega \times \omega$  is isomorphic to the set  $\{\{(id, 0), \dots, (id, n-1)\} \cup \{(idref, 0), \dots, (idref, m-1)\} \mid n, m \geq 0\}$ . In this way, any binding bigraph signature can be seen as an instance of a binding XML signature, taking  $ID = \{(id, i) \mid i \geq 0\}$  and  $IDREF = \{(idref, i) \mid i \geq 0\}$ .

Bigraphs generalise trivially to named ports. On the other hand, there is no direct correspondence to constants in bigraphs. In Sec. 2.3 below we consider different ways of encoding constants in bigraphs and also briefly touch on closed links.

## 2.1 Pure Bigraphs and XML

We will use a process calculus notation for bigraphs inspired by the CNF expressions of [10]. To keep the presentation simple we consider only open bigraphs. A calculus for closed bigraphs would include the usual local name constructor.

First we define process expressions that correspond to ground bigraphs and XML documents. In the next section we consider general context expressions, that correspond to general bigraphs with holes and XML contexts.

**Definition 3.** For a signature  $\Sigma = (\Sigma, Att, ar)$  define for any finite set of names  $X \subset N$  respectively the open tree, prime and wide  $\Sigma$ -process expressions with names  $X$  by the grammar

$$\begin{aligned}
 t &::= \kappa\{i : x_i\}_{i \in ar(\kappa)}.p && \text{Tree processes} \\
 p &::= t \mid p|p \mid 1 && \text{Prime processes} \\
 r &::= r \parallel r \mid p \mid 0 && \text{Wide processes}
 \end{aligned}$$

for  $\kappa \in \Sigma$  and  $x_i \in X$ .

Following [5] we refer to  $|$  and  $\parallel$  as respectively the prime and wide parallel composition. For  $Att = \omega$ , the prefix  $\kappa\{i : n_i\}_{i \in ar(\kappa)}$  can be written shortly as  $\kappa(n_0, n_1, \dots, n_{ar(\kappa)})$  which corresponds to the basic ion bigraph  $\kappa_{\vec{n}}$  in [10]. We generalize this to ions with named ports, writing  $\kappa_{\{i:n_i\}_{i \in ar(\kappa)}}$  for the ion with ports named  $ar(\kappa)$  and where port  $i$  is connected to the outer name  $n_i$ . We postpone the denotation of process expressions as bigraphs to the next section when process expressions are generalised to contexts.

We assume a structural congruence  $\equiv$  on process expressions, making prime parallel composition associative and commutative, wide parallel composition associative and the nil process 1 the identity for prime parallel composition and 0 the identity for wide parallel composition.

**Definition 4.** *Structural congruence  $\equiv$  is the least congruence on process expressions such that*

$$p_1|(p_2|p_3) \equiv (p_1|p_2)|p_3 \quad p|1 \equiv p \quad 1|p \equiv p \quad p|q \equiv q|p$$

and

$$r_1 \parallel (r_2 \parallel r_3) \equiv (r_1 \parallel r_2) \parallel r_3 \quad r \parallel 0 \equiv r \quad 0 \parallel r \equiv r$$

The structural congruence allows us to leave out parenthesis for prime and wide parallel composition, writing respectively  $\prod_{i \in n} p_i$  and  $\prod_{i \in n} r_i$  for the  $n$  times prime and wide parallel composition and letting  $\prod_{i \in 0} p_i = 1$  and  $\prod_{i \in 0} r_i = 0$ . As usual we will often leave out trailing nil processes, writing  $\kappa\{i : x_i\}_{i \in ar(\kappa)}$  for  $\kappa\{i : x_i\}_{i \in ar(\kappa)}.1$ .

The equation  $p|q \equiv q|p$  makes prime parallel composition commutative, meaning that process expressions up to structural congruence correspond to *abstract* bigraphs. Abstract bigraphs correspond to XML documents where the content of regions is considered unordered (semi-structured data).

We say that the *width* of a wide process expression  $r$  is  $n$  if  $r \equiv \prod_{i \in n} p_i$  for  $n \geq 0$ , i.e. the process consists of  $n$  primes. We explicitly type wide process expressions, writing  $r : \langle n, X \rangle$  for a wide process expression with name set  $X$  and width  $n$ . Note that the name set  $X$  may contain names not occurring in  $r$ . We will often write  $r : X$  for  $r : \langle 1, X \rangle$ .

Below we define the XML documents corresponding to ground open bigraphs.

**Definition 5.** *Assume a pure XML signature  $\Sigma$ . A pure  $\Sigma$ -XML document  $D$  with interface  $\langle n, \{x_1, \dots, x_k\} \rangle$  is a valid document of the form*

$$\begin{aligned} D &:= \langle \text{root } x_1 = \text{"id}_1\text{"} \dots x_k = \text{"id}_k\text{"} \rangle R_1 \dots R_n \langle / \text{root} \rangle \\ R &:= \langle \text{region} \rangle P \langle / \text{region} \rangle \\ P &:= T_1 \dots T_h \\ T &:= \langle \kappa a_1 = \text{"v}_1\text{"} \dots a_j = \text{"v}_j\text{"} \rangle P \langle / \kappa \rangle \end{aligned}$$

where  $\text{root}, \text{region} \notin \Sigma$ ,  $x_i$  are ID attributes,  $h \geq 0$ ,  $\kappa \in \Sigma$  and  $ar(\kappa) = \{a_1, \dots, a_j\}$ .

That the document is valid means only that it satisfies the requirement that ID attributes are uniquely valued and all IDREF values occur as an ID value.<sup>2</sup>

We are now ready to define formally the translation back and forth between pure XML documents and wide pure process expressions. For readability we will use a  $\$$ -prefix for ID and IDREF values.

**Definition 6.** *For a pure wide  $\Sigma$  process expression  $r : \langle n, X \rangle$  define the corresponding XML document  $\mathcal{X}[[r : \langle n, X \rangle]]$  inductively by*

$$\begin{aligned} \mathcal{X}[[r : \langle n, \{x_1, \dots, x_k\} \rangle]] &= \langle \text{root } x_1 = \text{"\$x}_1\text{"} \dots x_k = \text{"\$x}_k\text{"} \rangle \mathcal{X}[[r]] \langle / \text{root} \rangle \\ \mathcal{X}[[p \parallel r]] &= \langle \text{region} \rangle \mathcal{X}[[p]] \langle / \text{region} \rangle \mathcal{X}[[r]] \\ \mathcal{X}[[p|q]] &= \mathcal{X}[[p]] \mathcal{X}[[q]] \\ \mathcal{X}[[\kappa\{a_i : x_i\}_{a_i \in ar(\kappa)}.p]] &= \langle \kappa a_1 = \text{"\$x}_1\text{"} \dots a_j = \text{"\$x}_j\text{"} \rangle \mathcal{X}[[p]] \langle / \kappa \rangle \\ \mathcal{X}[[0]] &= \mathcal{X}[[1]] = \varepsilon \end{aligned}$$

where  $\varepsilon$  is the empty document.

<sup>2</sup> Strictly speaking, a DTD does not allow elements to have more than one ID attribute. This can be remedied by using the more general notion of keys and keyrefs found in XML Schema.

**Definition 7.** For a pure wide XML document  $D$  define the corresponding  $\Sigma$  process expression  $\mathcal{P}[[D]]$  inductively by

$$\begin{aligned}
\mathcal{P}[[\langle \text{root } x_1 = \text{"id}_1" \dots x_k = \text{"id}_k" \rangle R_1 \dots R_n \langle / \text{root} \rangle]] &= \prod_{i \in n} \mathcal{P}[[R_i]]_{\sigma} : \langle n, \{x_1, \dots, x_k\} \rangle \\
\mathcal{P}[[\langle \text{region} \rangle P \langle / \text{region} \rangle]]_{\sigma} &= \mathcal{P}[[P]]_{\sigma} \\
\mathcal{P}[[T_1 \dots T_m]] &= \prod_{i \in m} \mathcal{P}[[T_i]]_{\sigma} \\
\mathcal{P}[[\langle \kappa \ a_1 = \text{"id}'_1" \dots a_j = \text{"id}'_j" \rangle P \langle / \kappa \rangle]]_{\sigma} &= \kappa \{a_i : \sigma(\text{id}'_i)\}_{a_i \in \text{ar}(\kappa)}. \mathcal{P}[[P]]_{\sigma}
\end{aligned}$$

for  $\sigma$  the substitution mapping ID values to names defined by  $\sigma(\text{id}_i) = x_i$ .

The proposition below states that the translation from process-expressions to XML and back given in the previous section is an identity up to structural congruence.

**Proposition 1.** For a pure wide  $\Sigma$ -process expression  $r : \langle n, X \rangle$  it holds that  $\mathcal{P}[[X[[r : \langle n, X \rangle]]]] \equiv r : \langle n, X \rangle$

*Proof.* (Sketch) Straightforward, unfolding the definitions.

*Examples* Below we give a few examples of the correspondence between XML and process expressions. The pure document

```

<root office="$roomtype1" hilde="$n1" jww="$n2" hniss="$n3" >
  <region>
    <room type="$roomtype1" name="$n1">
      <person name="$n1"/>
      <person name="$n3"/>
    </room>
  </region>
  <region>
    <room type="$roomtype1" name="$n3"/>
  </region>
</root>

```

corresponds to the process expression of width 2

$$\begin{aligned}
&\text{room}\{\text{type} : \text{office}, \text{name} : \text{hilde}\}. \\
&\quad (\text{person}\{\text{name} : \text{hilde}\} | \text{person}\{\text{name} : \text{hniss}\}) || \\
&\text{room}\{\text{type} : \text{office}, \text{name} : \text{hniss}\} : \langle 2, \{\text{office}, \text{hilde}, \text{jww}, \text{hniss}\} \rangle
\end{aligned}$$

Conversely, one can reconstruct the XML document from the process expression, up to reordering of attributes and choices of ID values. As an example of representing process expressions as XML, the CCS-like process expression

$$\text{act}\{\text{ch} : a\}.\text{act}\{\text{ch} : b\} | \text{coact}\{\text{ch} : a\} : \{a, b\}$$

is represented by the pure XML document

```

<root a="$a" b="$b">
  <region>
    <act ch="$a">
      <act ch="$b"/>
    </act>
    <coact ch="$a"/>
  </region>
</root>

```

## 2.2 Binding Bigraphs and XML

In this section we briefly describe XML documents that corresponds to concrete ground binding bigraphs. Binding bigraphs introduce *binding* ports and bound names at regions. In XML this correspond to ID attributes. For the definition of binding bigraphs see [5].

We will say that a document is *scoped* if the scope of an ID attribute is restricted to IDREFS attributes of the descendents, i.e. (grand) children, of its element. In other words, for any IDREF value, the corresponding ID value must belong to an attribute of an ancestor node.<sup>3</sup> The scope condition may seem odd if one think of the OEM data model for semi structured data and representations of relational data in XML as e.g. described in [11]. However, it is completely natural in the world of process calculi, using ID values to represent local names. In particular, it will correspond to the scope condition enforced for binding bigraphs.

**Definition 8.** Assume a binding XML signature  $\Sigma$ . A binding  $\Sigma$ -XML document  $D$  with interface  $\langle n, \tilde{y}_1 \dots \tilde{y}_n, \{x_1, \dots, x_k\} \rangle$  is a valid document of the form

$$\begin{aligned} D &:= \langle \text{root } x_1 = \text{"id}_1\text{"} \dots x_k = \text{"id}_k\text{"} \rangle R_1 \dots R_n \langle / \text{root} \rangle \\ R_i &:= \langle \text{region } y_{i,1} = \text{id}_{i,1} \dots y_{(i,\tilde{y}_i)} = \text{id}_{(i,\tilde{y}_i)} \rangle P \langle / \text{region} \rangle \\ P &:= T_1 \dots T_h \\ T &:= \langle \kappa a_1 = \text{"v}_1\text{"} \dots a_j = \text{"v}_j\text{"} \rangle P \langle / \kappa \rangle \end{aligned}$$

where  $\text{root}, \text{region} \notin \Sigma$ ,  $x_i, y_{i,j}$  are ID attributes,  $h \geq 0$ ,  $\kappa \in \Sigma$  and  $\text{ar}(\kappa) = \{a_1, \dots, a_j\}$ .

The  $n$  sequences of names  $\tilde{y}_i = y_{i,1} \dots y_{(i,k_i)}$  in the interface gives the names bound to regions.

As an example of binding documents, consider the  $\pi$ -calculus like process expression

$$\text{send}\{\text{ref.subj} : a, \text{ref.obj} : b\} \mid \text{rec}\{\text{ref.subj} : a, \text{id.obj} : c\}.\text{send}\{\text{ref.subj} : a, \text{ref.obj} : c\} .$$

It is represented by the binding XML document

```
<root a="$a" b="$b">
  <region>
    <send ref.subj="$a" ref.obj="$b"/>
    <rec ref.subj="$a" id.obj="$c">
      <send ref.subj="$a" ref.obj="$c"/>
    </rec>
  </region>
</root>
```

where  $\text{id.obj}$  is an ID attribute of the *rec* element.

For the binding bigraph signature the prefix  $\kappa\{i : n_i\}_{i \in \text{ar}(\kappa)}$  may be written as

$$\kappa(n_0, n_1, \dots, n_{\text{ar}(\kappa) \cap \text{ID}})(m_0, m_1, \dots, m_{\text{ar}(\kappa) \cap \text{IDREF}}),$$

where  $\text{ID}$  and  $\text{IDREF}$  are respectively the (binding) indexes in the left and the (non-binding) indexes in the right injection of the attribute set. This corresponds to a binding ion bigraph  $\kappa_{\vec{n}, \vec{m}}$ .

<sup>3</sup> The scope condition can not be expressed using DTDs, but can be expressed using the more general keys and keyrefs in XMLSchema.

### 2.3 Dealing with PCDATA and CDATA and closed links

Text content (PCDATA) may be treated as atomic elements with no attributes. This is essentially the approach in [6]. That is, we may assume that XML signatures  $\Sigma$  includes elements  $text(s)$  for any text string  $s$  such that  $att(text(s)) = \emptyset$ . The XML document

```
<root office="$roomtype1" hilde="$p1" jww="$p2" >
  <region>
    <room type="$roomtype1" name="$p1">
      <person name="$p1"/>
      <person name="$p2">
        <message>
          Hello World!
        </message>
      </person>
    </room>
  </region>
</root>
```

would then correspond to the process expression

$$\text{room}\{type : office, name : hilde\} . (\text{person}\{name : hilde\} | \text{person}\{name : jww\} . \text{message}\{\}. \text{text}(\text{HelloWorld!})\{\})$$

and vice versa.

Constant (CDATA) attributes may be handled in several ways. One could internalize the constants in the element names. However, this seems is not very elegant. Another alternative is given in [6], encoding constant attributes as PCDATA elements with a local link to the attribute. A more direct way would be to apply the notion of *distinctions* as used for the  $\pi$ -calculus.

Closed links may be represented simply by introducing  $\langle edge\ id = "\$id" \rangle$  elements as children of the root (and siblings to the region elements). However, note that the scope condition should then not be enforced for references to edge ID attributes (this could be captured in an XMLSchema definition). Alternatively, one could use reserved names among the root ID attributes for closed links.

We leave it for future work to formalize CDATA and closed links.

## 3 Contexts and Reactions

In this section we define context expressions denoting general, non-ground bigraphs and define how contexts compose. Using the translation between XML and bigraph expressions given in the previous section this gives a formal definition of insertion of pure XML documents in a context.

**Definition 9.** For finite sets  $X, Y \subseteq N$  of names and finite ordinals  $n$  and  $m$ , the prime  $\Sigma$ -process contexts  $P$  and the set  $W_\Sigma$  of wide  $\Sigma$ -process contexts  $W : \langle m, Y \rangle \rightarrow \langle n, X \rangle$  with outer names  $X$ , inner names  $Y$  and  $m$ -indexed holes are then defined by the grammar

$$\begin{aligned} P &::= \kappa\{i : x_i\}_{i \in ar(\kappa)}.P \mid P|P \mid []_j \mid 1 \\ R &::= R \parallel R \mid P \mid 0 \\ W &::= \sigma \parallel R \end{aligned}$$

where  $\kappa \in \Sigma$ ,  $x_i \in X$  and  $j \in m$  and  $\sigma : Y \rightarrow X$  is a map from  $Y$  to  $X$ . A context  $W : \langle m, Y \rangle \rightarrow \langle n, X \rangle$  is linear if every index  $j \in m$  appears exactly once at a hole  $[]_j$ .

Linear contexts  $W : \langle m, Y \rangle \rightarrow \langle n, X \rangle$  correspond to open bigraphs. We will often omit the map  $\sigma$  if it is the inclusion  $Y \subseteq X$  and in that case say that the context has a trivial link map. Note that a wide  $\Sigma$ -process expression  $r : \langle n, X \rangle$  is a ground  $\Sigma$ -context expression  $r : \langle 0, \emptyset \rangle \rightarrow \langle n, X \rangle$  with trivial link map.

We can now define the denotation of context expressions as bigraphs in terms of the ion bigraph, vertical composition of bigraphs  $\circ$ , the empty prime bigraph  $1$  (i.e. the single barren region), the empty bigraph  $\emptyset$  (with width 0), prime parallel  $|$  and wide parallel  $\parallel$  composition of bigraphs.

**Definition 10.** For a pure XML signature  $\Sigma$  and pure  $\Sigma$  process expression  $r$  define the corresponding pure bigraph (generalised to named ports)  $\llbracket r \rrbracket$  inductively by

$$\begin{aligned} \llbracket r \parallel r' \rrbracket &= \llbracket r \rrbracket \parallel \llbracket r' \rrbracket \\ \llbracket \kappa\{i : n_i\}_{i \in \text{ar}(\kappa)} \cdot p \rrbracket &= (\kappa\{i : n_i\}_{i \in \text{ar}(\kappa)} \mid \text{id}_{fn(p)}) \circ \llbracket p \rrbracket \\ \llbracket p | p' \rrbracket &= \llbracket p \rrbracket | \llbracket p' \rrbracket \\ \llbracket \emptyset \rrbracket &= \emptyset \\ \llbracket 1 \rrbracket &= 1 \end{aligned}$$

The prime parallel composition essentially joins the regions of two prime bigraphs. The wide parallel composition essentially places the two wide bigraphs next to each other. The denotation of prefix is given by the ion bigraph (extended with the identity link graph on the free names of  $p$ ) composed with the denotation of the residual process. See [10] and [5] for the formal definition of the basic bigraph constructors.

**Definition 11.** For contexts  $W : \langle m, Y \rangle \rightarrow \langle n, X \rangle$  and  $W' : \langle n, X \rangle \rightarrow \langle n', X' \rangle$  define the composite context  $W' \circ W : \langle m, Y \rangle \rightarrow \langle n', X' \rangle$  by  $\sigma' \circ \sigma \parallel R'[j : P_j \sigma']_{j \in n}$  if

- $W = \sigma \parallel \prod_{j \in n} P_j$ ,
- $W' = \sigma' \parallel R'$
- $P_j \sigma'$  denotes the substitution of names  $x \in X$  in  $P_j$  with  $\sigma'(x) \in X'$  and
- $R'[j : P_j \sigma']_{j \in n}$  denotes the insertion of  $P_j \sigma'$  in all holes of  $R'$  having index  $j$ .

The composition of context-expressions is consistent with the definition of composition on bigraphs.

**Proposition 2.** For contexts  $W : \langle m, Y \rangle \rightarrow \langle n, X \rangle$  and  $W' : \langle n, X \rangle \rightarrow \langle n', X' \rangle$  it holds that  $\llbracket W' \circ W \rrbracket = \llbracket W' \rrbracket \circ \llbracket W \rrbracket$ .

In the special case where  $W' = \sigma' \parallel R' : \langle n, X \rangle \rightarrow \langle n', X' \rangle$  and  $W = \prod p_j : \langle 0, \emptyset \rangle \rightarrow \langle n, X \rangle$  is a ground  $\Sigma$ -context with trivial link map, that is, a wide  $\Sigma$ -process with type  $\langle n, X \rangle$ , the composite  $W' \circ W : \langle 0, \emptyset \rangle \rightarrow \langle n', X' \rangle$  is the wide  $\Sigma$ -process  $R'[j : p_j \sigma'] : \langle n', X' \rangle$ .

**Proposition 3.** For pure  $\Sigma$  process expressions  $r$  and  $r'$  it holds that  $r \equiv r'$  implies  $\llbracket r \rrbracket = \llbracket r' \rrbracket$ , where  $\equiv$  is the identity of abstract bigraphs [5].

*Proof.* (Sketch) Show that the defining equations for  $\equiv$  given in Def. 4 are preserved by the map  $\llbracket \cdot \rrbracket$  and apply Prop. 2.

We conjecture that the reverse direction of the above proposition is also true, i.e. that the process expression denoting an abstract bigraph is unique up to structural congruence.

Using the translation between XML and bigraph expressions given in the previous section we also get a formal definition of insertion of pure XML documents in a context.

**Definition 12.** Assume a pure XML signature  $\Sigma$ . For a pure  $\Sigma$ -XML document  $D$  with interface  $\langle n, X \rangle$  and a  $\Sigma$ -context  $W : \langle n, X \rangle \rightarrow \langle n', X' \rangle$  the insertion of  $D$  in  $W$  is defined by  $W(D) = X[\llbracket W \circ \mathcal{P}[D] \rrbracket]$ .

As described in [6] XML contexts may be regarded as positive (monotone) web services.

**Definition 13.** A parametric reaction rule is a pair of wide contexts with trivial link maps  $(R : \langle m, X \rangle \rightarrow \langle n, X \rangle, R' : \langle m, X \rangle \rightarrow \langle n, X \rangle)$  such that  $R$  is linear.

*Example:* The usual CCS reaction rules is written as the pair

$$(\text{act}\{ch : \$a\}.\llbracket \rrbracket_1 \mid \text{coact}\{ch : \$a\}.\llbracket \rrbracket_2, \llbracket \rrbracket_1 \mid \llbracket \rrbracket_2)$$

In semantics for process calculi, not all contexts allow reactions. Contexts that allow reactions are usually referred to as *evaluation contexts*. In the theory of bigraphical reactive systems, evaluation contexts are defined as (linear) contexts over a sub signature  $\Xi \subseteq \Sigma$  of *active prefixes*.

For a set  $S$  of parametric reaction rules and sub signature  $\Xi \subseteq \Sigma$  of active prefixes. we define the set of ground reaction rules by

$$\text{React}_{S,\Xi} = \left\{ (L, E \circ (\text{id}_Y \parallel R') \circ r) \mid \begin{array}{l} L \equiv E \circ (\text{id}_Y \parallel R) \circ r, \\ (R, R' : \langle n, X \rangle \rightarrow \langle m, X \rangle) \in S, \\ E : \langle m, Y \rangle \rightarrow \langle m', Z \rangle \in W_\Xi \text{ and} \\ r : \langle n, Y \rangle \text{ for } X \subseteq Y \end{array} \right\}$$

Given a set  $S$  of parametric  $\Sigma$ -reaction rules and a sub signature  $\Xi \subseteq \Sigma$  of active prefixes we can then define that a  $\Sigma$ -XML document  $D$  reacts to a  $\Sigma$ -XML document  $D'$ , written  $D \rightarrow D'$ , if  $(\mathcal{P}[\llbracket D \rrbracket], R') \in \text{React}_{S,\Xi}$  and  $\mathcal{X}[\llbracket R' \rrbracket] = D'$

The set  $\text{React}_{S,\Xi}$  is typically an infinite set of process pairs, thus this definition of reactions for documents is not very operational.

## 4 Reactive XML Prototype

In this section we sketch how the Reactive XML Prototype has been implemented. In particular we describe how evaluation contexts are described conveniently as XPath expressions. For simplicity, we will only consider prime reaction rules, that is, reaction rules  $(R, R' : \langle n, X \rangle \rightarrow \langle 1, X \rangle)$ . This means that we only need to consider evaluation contexts with one hole, and that changes are always performed inside the same region.

XML documents are manipulated and queried as ordered labelled trees (a DOM representation). As we never change the root element, the ordered labelled trees correspond to process expressions (up to structural congruence). We will thus use process expressions for the DOM representation of XML documents below. A sub prime process  $p$  then correspond to a set of sibling sub trees (i.e. sub trees with the same parent) inside a region. A sub tree process  $t$  corresponds to a sub tree inside a region.

Assume a finite set  $S$  of parametric prime  $\Sigma$ -reaction rules and a sub signature  $\Xi \subseteq \Sigma$ , determining the active prefixes. Performing a reaction  $D \rightarrow D'$  of a  $\Sigma$ -XML document  $D$  amounts to finding a reaction rule  $(R, R' : \langle n, X \rangle \rightarrow \langle 1, X \rangle) \in S$ , an evaluation context  $\sigma \parallel R_E : \langle 1, Y \rangle \rightarrow \langle m, Z \rangle \in W_\Xi$  and a wide process expression  $r = \prod_{j \in n} p_j$  such that

$$\mathcal{P}[\llbracket D \rrbracket] = R_E[R[j : p_j]_{j \in n} \sigma] \tag{1}$$

and then compute the document  $D' = \mathcal{X}[\llbracket R_E[R[j : p_j]_{j \in n} \sigma] \rrbracket]$ .

Now note that  $R_E[R[j : p_j]_{j \in n} \sigma] = R_E[R\sigma[j : p_j \sigma]_{j \in n}] = R_E \circ R\sigma \circ r\sigma$ . For any process  $r'$  and  $\sigma$  there exists a process  $r$  such that  $r' = r\sigma$ . Thus, solving equation (1) amounts to finding a complete subtree  $t_R = R\sigma \circ r'$  in  $\mathcal{P}[\llbracket D \rrbracket]$  for some  $r'$  and substitution  $\sigma$  such that the subtree  $t_R$  has a root that is purely nested inside elements in  $\Xi$ . The updated document  $D'$  is computed by replacing the subtree  $t_R$  with the tree  $t_{R'} = R'\sigma \circ r'$ .

The nodes purely nested in elements in  $\Xi$  can be identified by a simple XPath expression.

**Definition 14.** For a sub signature  $\Xi \subseteq \Sigma$  define the XPath expression

$$\begin{aligned} \phi_{\Xi, \Sigma} = & // * \text{not}(\text{ancestor-or-self}:: * [\text{name}() = \kappa_1 \text{ or} \\ & \text{name}() = \kappa_2 \text{ or} \\ & \dots \\ & \text{name}() = \kappa_k ]) \end{aligned}$$

for  $\Sigma \setminus \Xi = \{\kappa_1, \dots, \kappa_k\}$ . For a  $\Sigma$ -process  $\mathcal{P}[[D]]$  and XPath expression  $\phi$  let  $Xpath(\phi, \mathcal{P}[[D]])$  denote the set of roots of subtrees in  $\mathcal{P}[[D]]$  that satisfies  $\phi$ .

Of course, one could consider different XPath expressions, defining evaluation different kinds of evaluation contexts. One could also provide each rule with its own XPath expression, thereby making the definition of evaluation contexts dependent on the rule. This could be used as a kind of context-dependent reaction rules.

To find a sub tree  $t_R = R\sigma \circ r'$  in  $\mathcal{P}[[D]]$  for some wide process  $r'$  and substitution  $\sigma$  we search for the context  $R$  up to a possible substitution  $\sigma$  (computed as constraints during the attempted match) of the names in  $R$ , and allowing the holes in  $R$  to match any prime process, even an empty tree (i.e. a nil process 1). If the context  $R\sigma$  is found for some substitution  $\sigma$ , and prime processes  $p_j$  matched with holes, it is checked if the root of the context  $R\sigma$  belongs to the solution set of the XPath expression  $\phi_{\Xi, \Sigma}$ . If so, the matching algorithm reports back the substitution  $\sigma$ , the root of the context  $R\sigma$  and the (roots of the) sub prime processes  $p_j$  matched with holes. This is a generalisation of the standard (ordered) sub tree problem for trees. As for the standard problem the matching algorithm is extended to unordered trees by using a bipartite matching algorithm each time a set of children in the pattern  $R$  is matched against a set of children in the source tree  $\mathcal{P}[[D]]$ . To perform the reaction all that is needed is to replace sub tree  $t_R$  in  $\mathcal{P}[[D]]$  with  $R'\sigma[j : p_j]_{j \in n}$ .

#### 4.1 Representing Reaction Rules in XML

We can represent contexts in XML simply by introducing an element name *hole* with a single attribute *index*.

*Examples:* The lefthand context  $\text{act}\{\text{name} : \$a\} \cdot [ ]_1 \mid \text{coact}\{\text{name} : \$a\} \cdot [ ]_2$  of the CCS reaction rule can be represented as the XML document

```
<root v1=" $a ">
<region>
  <act name=" $a ">
    <hole index=" $1 ">
  </act>
  <coact name=" $a ">
    <hole index=" $2 ">
  </coact>
</region>
</root>
```

The representation of contexts as XML can be used to represent reaction rules as XML. This is used as input format for the tool. One could also represent rules as part of the bigraph and provide a (universal) rule for performing the reactions. We leave this idea for future work.

## 5 Conclusion and Future Work

We have presented an XML representation of bigraphical reactive systems, providing a general language for describing reactive processes in XML and a formal denotation of composition and transformations of XML processes and documents with links. In particular we have described how bigraph reactions can be carried out in praxis on the XML representations of bigraphs, using simple XPath expressions to capture the notion of evaluation contexts. This has been implemented in a rough prototype called Reactive XML, developed as part of a masters thesis at ITU [12].

So far, Reactive XML is mostly a proof of concept. Much work remains before it can be called a real XML framework, as for instance proper use of namespaces and XML schemas. So far, the prototype is just a simulator allowing to perform reactions on a single document. The prototype is currently being extended to a concurrent, distributed setting using a peer-to-peer XML storage being developed by researches and students at ITU and Copenhagen University [13–16]. This will allow to experiment with distributed XML-centric applications written in, or compiled to Reactive XML, that coordinate through computations on shared XML data. A concrete test case is likely to be context aware applications, which may draw on the work in [17]. The concrete applications are likely to suggest extensions or revisions of the theory. For instance, it seems reasonable to extend bigraphs with a notion of *constant names* to reflect the appearance of constant attribute values in XML. This could for instance be done using *distinctions* as used for the  $\pi$ -calculus. Also, the use of XPath expressions for evaluation contexts may be applied to define context-dependent reaction rules.

Making the system into a distributed system opens up a lot of questions, in particular how to provide access control, but also e.g. on how to combine Reactive XML with the ideas of Active XML and different types of XML spaces [3]. One could also imagine different levels of the system described in bigraphs, and only some of them represented concretely as Reactive XML documents. Perhaps on the more speculative side, one could imagine peers running each their own universal bigraphical reactive systems, performing reactions based on reaction rules described as part of the bigraph. This would provide an open system in which reaction rules could change dynamically.

On the theoretical side plenty of interesting questions also remain open. First of all, the work should be extended from open bigraphs to general, and possibly binding, bigraphs. Our investigations so far suggests that for general bigraphs the work should probably be based on DNF expressions for bigraphs instead of the CNF process calculus-like expressions used in this paper.

A more far reaching question is if one can find general conditions for achieving tractable proof techniques based on the derived labelled bisimulation congruences, for instance as restricted formats for reaction rules or proof techniques such as bisimulation up to context. Another obvious question is how to use ideas from spatial logics for describing properties of Reactive XML processes.

Many of these questions are pursued in the *Bigraphical Programming Language* (BPL) project at ITU [18], which aims to develop a general prototype programming language on top of the theory of bigraphs.

*Acknowledgements* Thanks to the anonymous referees and the other members of the BPL research group for helpful suggestions, Henning Niss, Mikkel Bundgaard and Lars Birkedal in particular.

## References

1. Cardelli, L., Gordon, A.D.: Mobile ambients. In Nivat, M., ed.: Proceedings of the First International Conference of Foundations of Software Science and Computation Structures (FOSACS'98). Volume 1378 of Lecture Notes in Computer Science., Springer Verlag (1998) 140–155

2. Cardelli, L.: Semistructured computation. In Connor, R.C.H., Mendelzon, A.O., eds.: Research Issues in Structured and Semistructured Database Programming. Proceedings of the 7th International Workshop on Database Programming Languages (DBPL'99), Invited Paper. Volume 1949 of Lecture Notes in Computer Science., Springer Verlag (2000) 1–16
3. Ciancarini, P., Tolksdorf, R., Zambonelli, F.: Coordination Middleware for XML-centric Applications. In: Proc. ACM/SIGAPP Symp. on Applied Computing (SAC), ACM Press (2002)
4. Stefani, J.B.: Requirements for a global computing programming model. Mikado Deliverable D1.1.2 (2003)
5. Jensen, O.H., Milner, R.: Bigraphs and mobile processes (revised). Technical Report UCAM-CL-TR-580, University of Cambridge, Computer Laboratory (2004)
6. Conforti, G., Macedonio, D., Sassone, V.: Bigraphical logics for XML. (2004)
7. Conforti, G., Macedonio, D., Sassone, V.: Bilogics: Spatial-nominal logics for bigraphs. (2004)
8. Gardner, P., Maffei, S.: Modelling dynamic web data. In Lausen, G., Suciu, D., eds.: 9th International Workshop on Database Programming Languages (DBPL'03). Volume 2921 of Lecture Notes in Computer Science., Springer Verlag (2003) 130–146
9. Abiteboul, S., Benjelloun, O., Manolescu, I., Milo, T., Weber, R.: Active XML:peer-to-peer data and web services integration (demo). In: Proceedings of the international VLDB Conference. (2002)
10. Milner, R.: Axioms for bigraphical structure. Technical Report UCAM-CL-TR-581, University of Cambridge, Computer Laboratory (2004)
11. Abiteboul, S., Buneman, P., Suciu, D.: Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann (1999)
12. Winther, J.W.: Reactive xml. Master's thesis, IT University of Copenhagen (2004)
13. Thorn, T., Fennestad, M., Baumann, A.: A distributed, value oriented xml store. Master's thesis, IT University of Copenhagen (2002)
14. Pedersen, K.B., Pedersen, J.T.: Value-oriented xml store. Master's thesis, IT University of Copenhagen (2002)
15. Thorn, T., Fennestad, M., Baumann, A., Sestoft, P.: A peer-to-peer, value oriented xml store. Short introduction to the first version of XML Store (2003)
16. Ambus, T.: Multiset discrimination for internal and external data. Master's thesis, Dept. of Computer Science Copenhagen University (DIKU) (2004)
17. Braione, P., Picco, G.P.: On calculi for context-aware coordination. In De Nicola, R., Ferrari, G., Meredith, G., eds.: Proceedings of the 7th International Conference on Coordination Models and Languages (COORDINATION'04). Volume 2949 of Lecture Notes in Computer Science., Springer Verlag (2004) 38–54
18. Lars Birkedal (principal investigator): Bigraphical Programming Languages (BPL) Project (2004-2008)