Bigraphs and Their Algebra

Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 209 (2008) 5–19
www.elsevier.com/locate/entcs

Bigraphs and Their Algebra

Robin Milner[1]

University of Cambridge
United Kingdom

Abstract

Bigraphs are a framework in which both existing process calculi and new models of behaviour can be formulated, yielding theory that is shared among these models. A short survey of the main features of bigraphs is presented, showing how they can be developed from standard graph theory using elementary category theory. The algebraic manipulation of bigraphs is outlined with the help of illustrations. The treatment of dynamics is then summarised. Finally, origins and some related work are discussed. The paper provides a motivating introduction to bigraphs.

Keywords: Bigraphs, Process Calculi

1 Introduction

Space Even before the digital computer was invented some sixty years ago, computation depended on ways to organise space; not the space of Euclidean geometry, but a discrete space involving notions like order and containment. Arabic numerals use linear space to represent the power of digits; then two-dimensional space can be used to represent the basic numerical algorithms—addition, multiplication, and so on. Spatial structures of data—sequences, matrices and graphs—played an important part before the stored-program computer; indeed, the algorithms for solving differential equations with a manual calculator combined the use of space for data and calculation in sophisticated ways.

Computer programming ramifies the use of space and spatial metaphor, both for writing programs and for explaining them. This shows up in our vocabulary: flow chart, location, send and fetch, pointer, nesting, tree, etc. Concurrency expands the vocabulary further: distributed system, remote procedure call, network, routing, etc.

We are living with a striking phenomenon: the metaphorical space of algorithms—graph, array, and so on—is mixed with the space of physical reality. Consider the

[1] Email: rm135@cam.ac.uk

doi:10.1016/j.entcs.2008.04.002

embedded software distributed among the artefacts it controls on a flying plane, or the physical deployment of sensors and effectors through which the software agents in a ubiquitous computing system will interact with their environment, whether along a highway controlling motor traffic or in a human body monitoring the bloodstream and administering treatment. Informatic objects flow in physical space; physical objects such as mobile telephones manipulate their informatic space.

Model In defining a model for spatially-rich systems we have to consider who will use it, and for what purpose. We must cater for end-users and programmers, as well as for system-designers and theoretical analysts. We have argued that discrete space is intrinsic to model informatics, so it is natural to seek a graphical model—and this will naturally appeal to end-users. On the other hand mathematical presentations suit analysts, programming languages suit programmers, and specification formalisms suit designers. These are all formal, to some degree. One *formalism* will not suit everyone; but there is nothing contradictory about a *model* that can be presented graphically for less technical clients and mathematically for analysts, underlying a design methodology for engineers and providing an executable subset that is a programming language. This is what the bigraph model tries to provide.

2 Bigraphs in pictures

In this section we develop the notion bigraph pictorially, from well-known concepts.

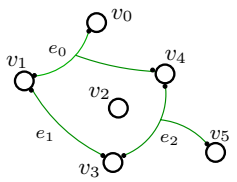
Notation and convention We write $S \uplus S'$ for the union of sets S and S' known or assumed to be disjoint. We consider a non-negative integer k as the finite ordinal $k = \{0, 1, \ldots, k-1\}$. We denote by ORD the category whose objects and arrows are the finite ordinals and the maps between them.

In a *graph* with nodes V and edges E, an edge joins a pair of nodes. A *hypergraph* is a generalisation in which an edge may join any number of nodes. We begin with a form of hypergraph in which each node $v \in V$ has an *arity* $ar(v)$, a finite ordinal, and has *ports* $P_v \stackrel{\text{def}}{=} \{(v, i) \mid i \in ar(v)\}$. Thus the hypergraph has ports $P_V \stackrel{\text{def}}{=} \biguplus_{v \in V} P_v$. Then we define a hypergraph to be a quadruple

$$(V, ar, E, link)$$

where $ar : V \to \text{ORD}$ defines arities, and $link : P_V \to E$ assigns each port to an edge.

a hypergraph



The diagram shows a hypergraph with nodes $\{v_0, \ldots, v_5\}$ and edges $E = \{e_0, e_1, e_2\}$; nodes are circles, ports are blobs, and an edge is shown as a linkage among its ports.

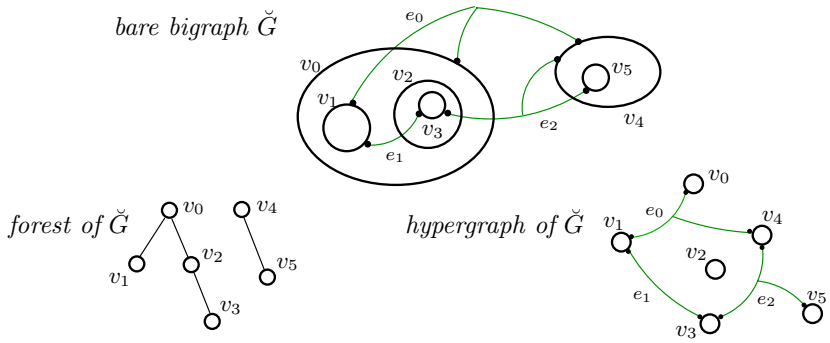
We shall now enrich these hypergraphs into bigraphs, in four steps. First, con-

sidering a hypergraph as *linking* nodes, we endow nodes with an extra structure which we call *placing*—justifying the prefix 'bi'. Second, we introduce *interfaces*, which make parts of a bigraph externally accessible. Third, we introduce *signatures* to classify nodes. Fourth, we show how to *construct* larger bigraphs from smaller ones.

2.1 Placing and linking

A bigraph with nodes V and edges E has a hypergraph with nodes V and edges E, and a forest with nodes V.

We wish to allow nodes to be nested. This nesting will represent spatial structure, whether in physical space or in the kind of virtual space that is familiar in programming languages (e.g. the nesting of the scopes of variables) or in process calculi (e.g. the nesting of mobile ambients). The two structures, which we call *placing* and *linking*, are completely independent. We already have linking represented by a hypergraph; placing consists of a forest (of the nodes), i.e. a set of trees. In drawing a bigraph we represent this forest by the nesting of nodes. We shall use the notation $\breve{F}, \breve{G}, \ldots$ to stand for these so-called *bare bigraphs*. Here is a bare bigraph $\breve{G}$ having nodes $V = \{v_0, \ldots, v_5\}$ and edges $E = \{e_0, e_1, e_2\}$, with its forest and hypergraph:

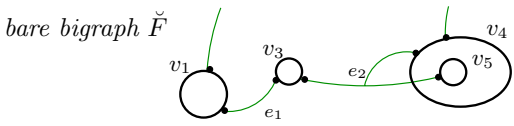


The upper diagram presents both the forest and the hypergraph; it depicts the forest by nesting. The lower two diagrams represent the two structures separately, in a conventional manner. The hypergraph of $\breve{G}$ is the one illustrated earlier.

2.2 Interfaces

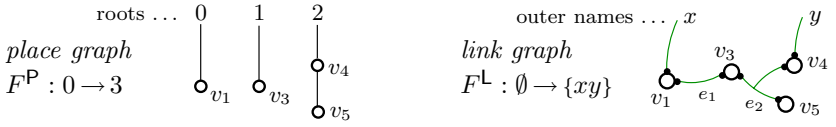
A bigraph has interfaces, which define its use as a construction block.

We want to make larger bigraphs from smaller ones, and to consider one bigraph as a sub-structure of another. For example here is $\breve{F}$, informally a 'part' of $\breve{G}$, having only some of its nodes and with one hyperlink broken. Can we call it a sub-structure of $\breve{G}$?



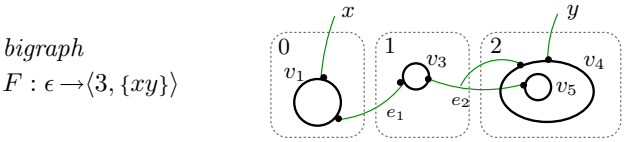
To make it so, we add *interfaces* to bare bigraphs, thus extending $\breve{F}$ and $\breve{G}$ to bigraphs F and G. This will allow us to represent the occurrence of F as a substructure of G by an equation $G = H \circ F$, where H is some 'host' or contextual bigraph. We do this extension independently for forests and hypergraphs; a forest with interfaces will be called a *place graph*, and a hypergraph with interfaces will be called a *link graph*.

A place graph interface is a finite ordinal $n = \{0, 1, \ldots, n-1\}$. The members of a place graph's *outer* and *inner* interfaces—or *faces* as we shall call them—are respectively its *roots* and *sites*, and are disjoint from its nodes. The outer and inner faces of a link graph are *name-sets*: respectively, its *outer* and *inner names*. Names are drawn from an infinite repertoire $\mathcal{X}$.



Let us illustrate with the bare bigraph $\breve{F}$. For the forest of $\breve{F}$ we choose the outer face $3 = \{0, 1, 2\}$, providing distinct roots as parents for the nodes v_1, v_3 and v_4. For its inner face we choose 0, i.e. it has no sites. We write the resulting place graph as $F^{\mathsf{P}} : 0 \to 3$; it is shown at the left of the above diagram. For the hypergraph of $\breve{F}$ we choose outer face $\{xy\}$, thus naming the parts of the broken hyperlink, and inner face $\emptyset$.[2] We write the resulting link graph as $F^{\mathsf{L}} : \emptyset \to \{xy\}$.

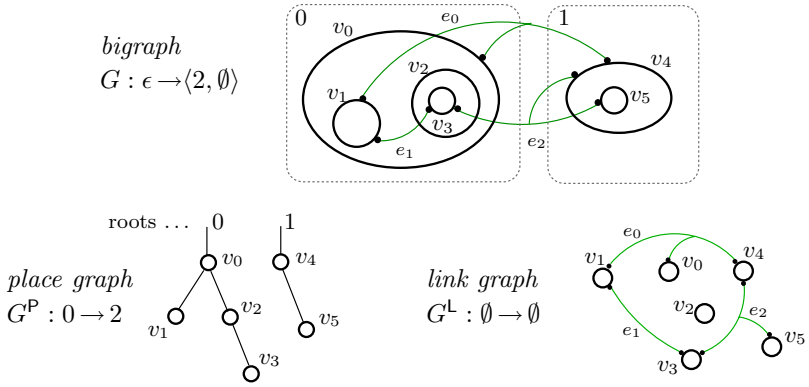
Finally, a *bigraph* is a pair $B = \langle B^{\mathsf{P}}, B^{\mathsf{L}} \rangle$ of a place graph and a link graph with the same node-set; these two graphs are its *constituents*. Its outer face is a pair $\langle n, Y \rangle$, where n and Y are the outer faces of B^{P} and B^{L} respectively. Similarly for its inner face $\langle m, X \rangle$. For our example $F = \langle F^{\mathsf{P}}, F^{\mathsf{L}} \rangle$ these pairs are $\langle 3, \{xy\} \rangle$ and $\langle 0, \emptyset \rangle$ respectively. We call the trivial interface $\epsilon \stackrel{\text{def}}{=} \langle 0, \emptyset \rangle$ the *origin*. We write $F : \epsilon \to \langle 3, \{xy\} \rangle$, and we draw it as follows:



The rectangles in F—sometimes called *regions*—represent its roots. The link graph F^{L} has four *links*. Two of these are the edges e_1 and e_2, also called *closed* links; the other two are named x and y, and are called *open* links.

Let us also add interfaces to the bare bigraph $\breve{G}$, extending it to a bigraph G. It has no open links, i.e. all its links are edges, so the name-set in its outer face will be empty. We give it two roots, as parents of v_0 and v_4; then, if G is placed in some larger context, these nodes may be in distinct places—i.e. may have distinct parents. Here are G and its constituents:

[2] We use single letters for names, so we shall often write a set $\{x, y, \ldots\}$ of names as $\{xy \cdots\}$.



Note especially that, in the upper diagram, there is no significance in where a link 'crosses' the boundary of a node or region; this is because the forest and hypergraph structures are independent.

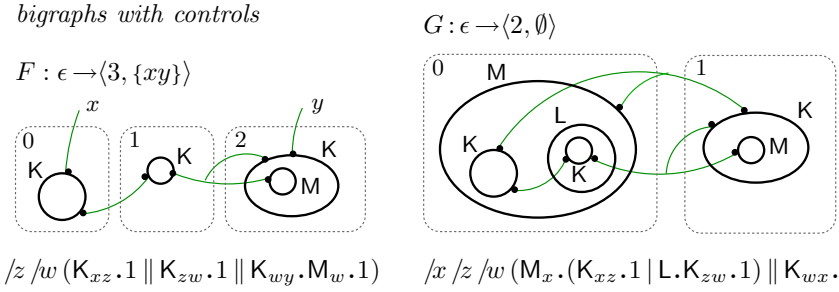
2.3 Classification

The nodes of a bigraph may be of different kinds; this reflects that they may contribute differently to dynamics.

To each node in a bigraph is assigned as kind, called a *control*. For each application we are likely to have different controls, specified—together with their arities—in a *signature* such as

$$\mathcal{K} = \{\mathsf{K} : 2,\ \mathsf{L} : 0,\ \mathsf{M} : 1\}.$$

Thus, in any bigraph over $\mathcal{K}$, the arity of a node is the arity of its control. If we are not interested in the identifier v of each node, but only in its control, then we omit the identifier in diagrams and show the control instead. Thus, for $G : \epsilon \to \langle 2, \emptyset \rangle$ as above, we would draw the following:



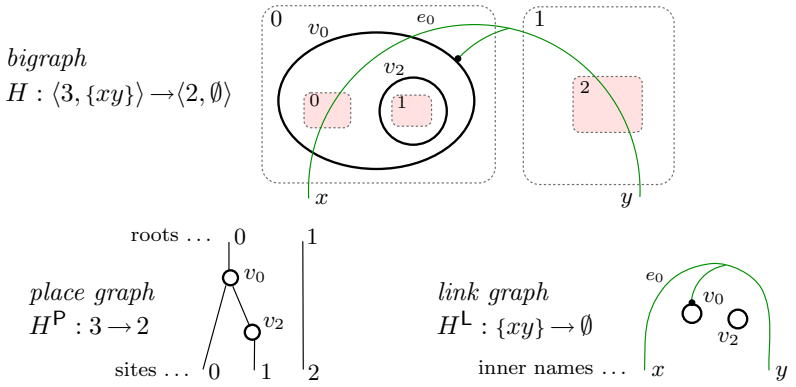
Note that we have also omitted edge identifiers. We call a bigraph *concrete* or *abstract*, according as the node and edge identifiers are present or absent. From now on we shall work mainly with abstract bigraphs.

It is essential to have an algebraic notation for abstract bigraphs, so that we can manipulate them rigorously. The diagram shows algebraic expressions for F and G. From now on we shall give such expressions for every bigraphical diagram, but they can be ignored until the algebra is explained in the ensuing section. There we shall see how the algebra arises directly from standard categorical operations.

2.4 Construction

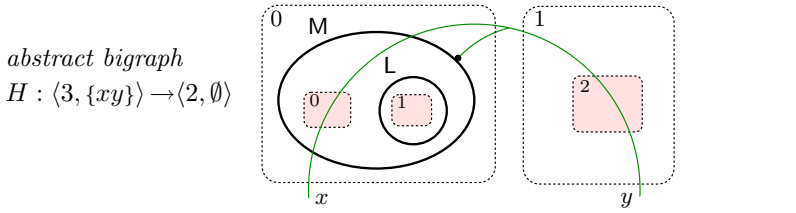
We make larger bigraphs from smaller ones via their interfaces; this construction is defined in terms of the constituent place and link graphs.

We are now ready to construct a bigraph H such that $G = H \circ F$, illustrating categorical composition. The inner face of H must be $\langle 3, \{xy\}\rangle$, the outer face of F; to achieve this, H must have three *sites* 0, 1 and 2, and inner names x and y. Here are H and its constituents, with sites shown as shaded rectangles:



In the place graph, each site and node has a parent, which is a node or a root; in the link graph, each inner name and port belongs to a link, closed or open. We draw inner names below the bigraph and outer names above it; this is merely a convention to indicate their status as inner or outer. A name may be both inner and outer, whether or not in the same link. In a bigraph diagram it is insignificant where a link 'crosses' a site, just as it is insignificant where it 'crosses' a node or root boundary.

Here is H as an abstract bigraph, with its controls and its algebraic representation.



In general, let $F : I \to J$ and $H : J \to K$ be two bigraphs with disjoint nodes and edges, where $I = \langle \ell, X\rangle$, $J = \langle m, Y\rangle$ and $K = \langle n, Z\rangle$. Then the composite $H \circ F : I \to K$ is just the pair of composites $\langle H^{\mathsf{P}} \circ F^{\mathsf{P}}, H^{\mathsf{L}} \circ F^{\mathsf{L}}\rangle$, whose constituents are constructed as follows (informally):

- To form the place graph $H^{\mathsf{P}} \circ F^{\mathsf{P}} : \ell \to n$, for each $i \in m$ join the i^{th} root of F^{P} with the i^{th} site of H^{P};
- To form the link graph $H^{\mathsf{L}} \circ F^{\mathsf{L}} : X \to Z$, for each $y \in Y$ join the link of F^{L} having

the outer name y with the link of H^{L} having the inner name y.

Thus H and F are joined at every place or link in their common face J, which ceases to exist. The reader may check these constructions for H and F in our example.

This concludes our informal introduction to the structure of bigraphs.

3 Applications, algebra and dynamics

In this section we use bigraphs to model both physical systems and process calculi. We also explain informally how the algebra of bigraphs arises from standard categorical notions; thus we make sense of the algebraic expressions that we have hitherto provided for several diagrammed bigraphs. Finally we represent dynamics as the reconfiguration of bigraphs.

3.1 Application

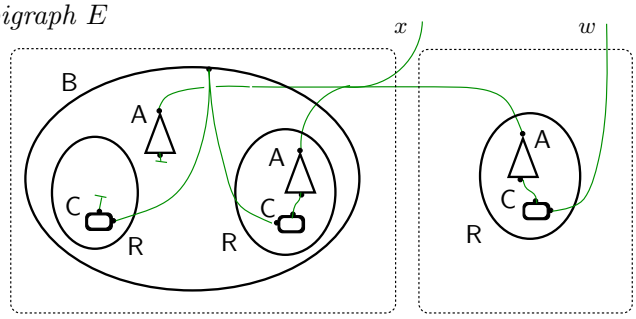
Each application of bigraphs requires a signature.

Consider a model of a built environment in which there are agents, buildings, computers and rooms. These four controls are declared in the signature

$$\{\mathsf{A}:2,\ \mathsf{B}:1,\ \mathsf{C}:2,\ \mathsf{R}:0\}.$$

The next diagram shows a bare bigraph E over this signature. The node-shapes are not significant, except to indicate informally the purpose of each port. The figure represents a state which may change because of the movement of agents, and perhaps other movements. Think of the three agents as conducting a conference call (the open link x). An agent in a room may also be logged in (the short links) to a computer in the room, and the computers in a building are linked to form a local area network.

bigraph E



$$/z\,(\mathsf{B}_z.(\mathsf{R}./y\,\mathsf{C}_{yz} \mid /y\,\mathsf{A}_{xy} \mid \mathsf{R}./y\,(\mathsf{A}_{xy} \mid \mathsf{C}_{yz})) \parallel \mathsf{R}./y\,(\mathsf{A}_{xy} \mid \mathsf{C}_{yw})$$

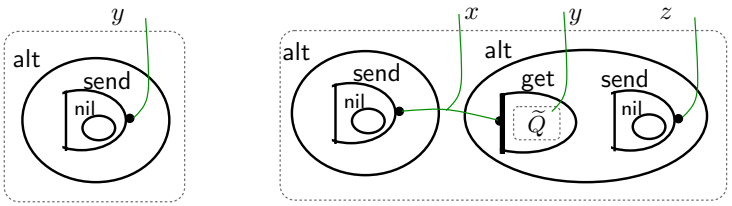
Process calculi can also be modelled. Consider finite CCS, with the following syntax for processes and alternations (sums):

$$P ::= A \mid \nu x P \mid P \,|\, P$$
$$A ::= \mathbf{0} \mid \mu.P \mid A{+}A$$
$$\mu ::= \overline{x} \mid x \quad .$$

To translate these terms into bigraphs we declare the signature

$$\{\mathsf{send}:1,\ \mathsf{get}:1,\ \mathsf{alt}:0\}.$$

Both parallel composition of processes and sums of alternates are represented by the juxtaposition of bigraphs; a sum or alternation is distinguished from a process by being nested in an 'alt' node. The empty process 0 is represented by the empty summation $\mathsf{nil} \stackrel{\text{def}}{=} \mathsf{alt}.1$, and restriction νx is represented by name closure $/x$. Here, for example, is the translation $\widetilde{P}$ of the CCS process $P = \overline{x}.0 \,|\, (x.Q + \overline{z}.0)$ where $Q = \overline{y}.0$:



$\widetilde{Q} \;=\; \mathsf{alt}.\mathsf{send}_y.\mathsf{nil}$ $\qquad$ $\widetilde{P} \;=\; \mathsf{alt}.\mathsf{send}_x.\mathsf{nil} \,|\, \mathsf{alt}.(\mathsf{get}_x.\widetilde{Q} \,|\, \mathsf{send}_z.\mathsf{nil})$

An important development of bigraphs involves the *binding* of links. This allows bigraphs to model both the λ-calculus and the π-calculus, and indeed any calculus whose syntax admits the binding of names. Binding in bigraphs consists simply of confining certain links to certain places; that is, the ports of such a link must lie within its designated place. This involves some refinement of the theory of bigraphs, but much of it remains unaffected. Indeed, the theory of binding bigraphs is best explained against the background of the pure theory in which placing is independent of linking.

3.2 Algebra

> *Diagrams are valuable for rapid appreciation of a system's structure. On the other hand algebra is essential, to express and manipulate the ways in which a system may be resolved into components.*

We explain the algebraic forms which have been associated with many bigraphs in preceding sections. This algebra is concerned only to express the structure of bigraphs, not their dynamics.

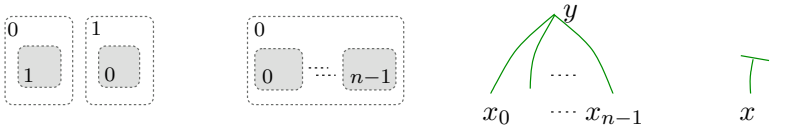
Interfaces Recall that an interface takes the form $I = \langle n, X\rangle$. If $X = \emptyset$ we abbreviate this to $I = n$; if $n = 0$ we abbreviate it to $I = X$, or $I = x$ if $X = \{x\}$. If $n = 1$ the interface I is said to be *prime*. The empty interface $\epsilon = \langle 0, \emptyset\rangle$ is called the *origin*.

The category of bigraphs The abstract bigraphs over a given signature form a category with interfaces $I, J, \ldots$ as objects and bigraphs $F : I \to J$ as arrows. If $I = \epsilon$ then F is said to be *ground*; if J is prime, then F is said to be *prime*. Given $F : I \to J$ and $G : J \to K$, we have already illustrated how to form the composite $G \circ F$, by placing the roots of F in the sites of G and eliding each open link y of F with every link of G that contains the inner name y.

This category is *strict symmetric monoidal (ssm)*; this means that it has a well-behaved operation for juxtaposing two disjoint bigraphs $F_0 : I_0 \to J_0$ and $F_1 : I_1 \to J_1$. This is called the *tensor product*, written $F_0 \otimes F_1 : I_0 \otimes I_1 \to J_0 \otimes J_1$. If $I_i = \langle m_i, X_i\rangle$ $(i = 0, 1)$ and X_0, X_1 are disjoint, then $I_0 \otimes I_1 \stackrel{\text{def}}{=} \langle m_0{+}m_1, X_0 \uplus X_1\rangle$; similarly for $J_0 \otimes J_1$.[3] Then the product $F_0 \otimes F_1$ of F_0, F_1 is formed just by laying them side-by-side. Product and composition enjoy pleasant properties, and all our algebraic expressions are definable in terms of them. Thus bigraphs have a secure mathematical foundation.

Elementary bigraphs There are three kinds of elementary bigraph. The first two kinds are node-free. If a node-free bigraph also has no links it is called a *placing*; if it has no places, it is called a *linking*. Here are the elementary placings and linkings, from which all others can be formed using composition and product.

elementary placings and linkings



$swap : 2 \to 2$ $\quad merge_n : n \to 1$ $\quad {}^{y}/_{X} : X \to y$ $\quad /x : x \to \epsilon$

We often need the product of a placing and a linking. for example, if F has the outer face $\langle 2, X\rangle$ then $(swap \otimes \mathsf{id}_X) \circ F$ swaps the two regions of F. On the other hand $(\mathsf{id}_2 \otimes {}^{y}/_{X}) \circ F$ replaces all its outer names X by y. Without ambiguity we shall often abbreviate such compositions, writing them respectively as $swap\, F$ and ${}^{y}/_{X}\, F$.

A placing $merge_n$ is called a *merge*. The special case when $n = 0$ consists of a single 'idle' region (one that is empty); we write it as 1.

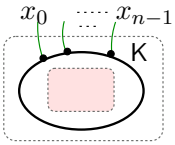
A linking ${}^{y}/_{X}$ is called a *substitution*. The special case when $X = \emptyset$ consists of a single 'idle' link (one that links nothing); we write it as y. We write a singleton $X = \{x\}$ as x.

The linking $/x$ is called a *closure*; it closes an x-link.

[3] To be precise: in an ssm category the tensor product is a total operation, while here $I_0 \otimes I_1$ is defined only when the names of I_0 and I_1 are disjoint. This relaxation makes little difference for our purposes.

The third kind of elementary bigraph is a *discrete ion*, i.e. a single K-node of arity n with its ports linked to n distinct outer names $\boldsymbol{x}$. It has a single site:

discrete ion



$$\mathsf{K}_{\boldsymbol{x}} : 1 \to \langle 1, \{\boldsymbol{x}\}\rangle$$

Operations It is remarkable that all bigraphs can be constructed from the elementary ones using composition ($\circ$) and tensor product ($\otimes$). But these operations hardly appear in the expressions for various bigraphs in our illustrations. This is partly because we have used abbreviations for composition with the elementary bigraphs, but mainly because we have used three derived operations which we now describe.

The first two operations, the *parallel product* $F_0 \parallel F_1$ and the *prime product* $F_0 \mid F_1$, resemble tensor product except that the outer names of F_0 and F_1 need not be disjoint—in other words, they share outer names. We first define these products on arbitrary interfaces $J_i = \langle n_i, Y_i\rangle$ $(i = 0, 1)$, as follows:

$$J_0 \parallel J_1 \stackrel{\text{def}}{=} \langle n_0{+}n_1, Y_0 \cup Y_1\rangle$$
$$J_0 \mid J_1 \stackrel{\text{def}}{=} \langle 1, Y_0 \cup Y_1\rangle \ .$$

Then the products of bigraphs $F_i : I_i \to J_i$ $(i = 0, 1)$ are

$$\textit{parallel product: } F_0 \parallel F_1 : I_0 \otimes I_1 \to J_0 \parallel J_1$$
$$\textit{prime product: } F_0 \mid F_1 : I_0 \otimes I_1 \to J_0 \mid J_1 \ .$$

They are defined exactly like tensor product except that the links of shared outer names in $Y_0 \cap Y_1$ are coalesced, and the prime product has a prime outer face. The operations can be defined from tensor product with the help of substitution and merging; in fact we find that $F_0 \mid F_1 = \textit{merge}\,(F_0 \parallel F_1)$.

The third operation, *nesting*, is a derived form of composition. If $F : I \to \langle m, X\rangle$ and $G : m \to \langle n, Y\rangle$ then the nesting of F within G is defined by

$$G.F \stackrel{\text{def}}{=} (\mathsf{id}_X \parallel G) \circ F : I \to \langle n, X \cup Y\rangle \ .$$

A good example is when $G = \mathsf{K}_{\boldsymbol{x}}$, an ion; in this special case we have $m = n = 1$. We may think of F placed inside G, but the outer names of F are rendered accessible as outer names of $G.F$, and indeed may share with the outer names of G.

Having understood the elementary bigraphs, and these derived operations, you are invited to examine the algebraic expressions associated with several bigraphs in the preceding diagrams, to convince yourself that they do indeed denote those bigraphs.

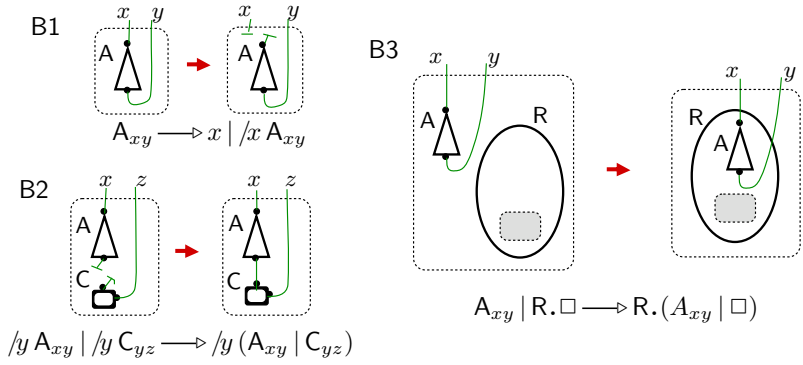
Every such expression represents a way to resolve a bigraph into components. Each bigraph can be expressed in many ways. But there exists a sound and complete axiomatisation of bigraphs, i.e. a set of equational axioms such that two expressions can be proved equal by the equations if and only if they denote the same bigraph. These equations represent the so-called *structural congruence* of bigraphs.

3.3 Dynamics

Bigraphs can reconfigure themselves according to reaction rules, which can be defined arbitrarily.

We now explain how bigraphs can reconfigure themselves, using both our built environment and CCS as illustrations. For each application we are free to define reconfiguration by means of *reaction rules*, each consisting of a *redex* (the pattern to be changed) and a *reactum* (the changed pattern). These patterns are both bigraphs, so they may involve both placing and linking.

A rule may induce a reaction in a bigraph G if its redex matches a part of G; we omit the precise details of matching. Here are three possible rules for built environments, such as the system E shown above:



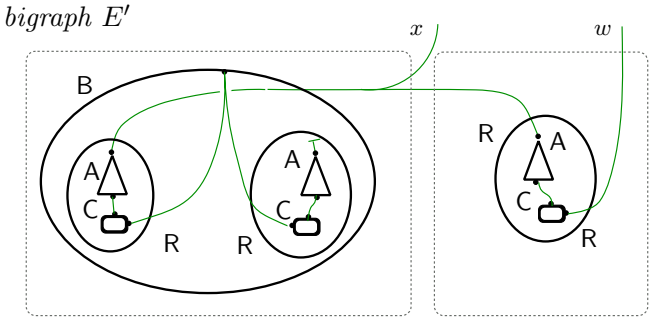
Rule B1 is the simplest: an agent can leave a conference call. The redex—the left-hand pattern—can match any agent; the out-pointing links mean that her ports may at first be linked to *zero or more* other ports, in the same place or elsewhere. If she is linked via x in a conference call to other agents, perhaps in other buildings, the reaction by B1 will unlink her; any link to a computer is retained.

Rule B2 shows a computer connecting to an agent in the same place (presumably a room). The redex insists that at first the agent is linked to no computer and the computer is linked to no agent. Rules B1 and B2 change only the linking—not the placing—in a bigraph, though the redex of B2 does insist on juxtaposition.

Rule B3, by contrast, changes the placing; an agent enters a room. Again, the rule requires the agent and the room to be in the same place (presumably a building). The site (shaded) represents a *parameter* of the rule; it allows the room to contain other occupants, e.g. a computer. The matching discipline allows these occupants to be linked anywhere, either to each other or to nodes lying outside the room.

Another feature of B3 is that its redex allows the ports of the agent to be already linked to nodes elsewhere; the reactum retains any such link. Equally, there may be no such link—the context in which the rule is applied may close it off. Thus B3 can be applied to the system represented by E, allowing an agent in the left-hand building to enter a room.

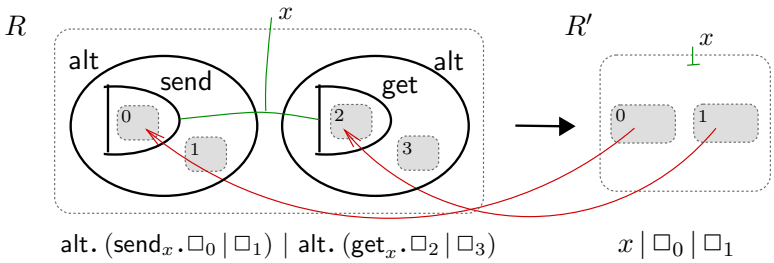
You may check that, after each of these three rules has been applied once, the system reaches the state E' as shown:



In Section 3.1 we discussed the translation of finite CCS into bigraphs. We now show the translation of the single reaction rule of CCS:

$$(\overline{x}.P + A) \,|\, (x.Q + B) \;\longrightarrow\; P \,|\, Q \;.$$

Since the redex and reactum have respectively four and two parameters, the same applies to the bigraphical rule, shown here:



Note that the rule, with redex R and reactum R', is also equipped with a map—the long arrows—showing where the parameters of R are instantiated in R'. In general this parameter map is neither injective nor surjective; thus bigraphical reconfiguration can involve both discard and replication of components.

This concludes our summary of bigraph dynamics. Much more has been done; in particular, it has been shown how to derive labelled transition systems for several process calculi, in such a way that the resulting behavioural pre-orders and equivalences agree well with the original theories of those calculi. Thus bigraphs provide some unity among disparate models of concurrent behaviour, as well as providing a framework in which new phenomena, such as ubiquitous systems, may be modelled.

4 Origins and related work

Bigraphs [25] were developed from action calculi [24], by adoption of the main idea in Section 2: that linking and placing should be independent, at least in the basic model. This independence yields a double gain: not only does it reflect real-life systems (we need only think of wireless networks) but it also dramatically simplifies the theory.

One criterion for the success of bigraphs has been that they should recover theory for existing process calculi, in particular their behavioural equivalences and preorders, which are often based upon labelled transition systems. It was shown [21] how to derive these transition systems in any categorical model possessing relative pushouts. The demonstration that action calculi possess that property [20] was difficult, but under the independency assumption in bigraphs it became quite tractable [18], and has been applied to several calculi [17,18,22,27,16,3].

There is a long tradition in graph-rewriting based upon the *double pushout* (DPO) construction originated by Hartmut Ehrig [10]. That work typically uses a category in which the objects are graphs and the arrows are embeddings. In contrast, our approach has interfaces as objects and graphs as arrows. There are links between these formulations, both via cospans [12] and via a categorical isomorphism between graph embeddings and a coslice over s-categories [4]. Ehrig [11] investigated these links further, after discussion with the author, and we believe that useful cross-fertilisation is possible. Gadducci, Heckel and Llabrés Segura [12] represent graph-rewriting by 2-categories, whose 2-cells correspond to our reactions. Several other formulations of graph-rewriting employ hypergraphs, for example Hirsch and Montanari [15]; their hypergraphs are not nested, but rewriting rules may replace a hyperedge by an arbitrary graph. Another use of 2-categories is by Sassone and Sobocinski [29], where the notion of relative pushout is generalised.

There is a variety of frameworks for modelling concurrent interactive behaviour. [4] We have already discussed one: *graph rewriting*. Others are: *term rewriting* by a group of authors led by J.W. Klop [30], which can accommodate arbitrary equational axioms; *rewriting logic* led by J. Meseguer [23,5] which includes Maude, an automated logic for rewriting; *the tile model* led by U. Montanari [13], whose tiles represent rewriting rules and can be composed in two dimensions, one to yield longer rewritings and one to yield compound rules. The bigraph model is also a framework; to obtain a specific calculus one defines both a signature and a set of reaction rules. As we have seen, this admits calculi that differ widely. But the model makes a commitment to a particular kind of graph; this was suggested by the observation that both placing and linking are fundamental to informatic systems, so that a theory of these two notions deserves specific treatment. This theory has three significant aspects. First, as shown in this paper and many others already cited, it is committed to a special family (indexed by signatures) of graphical categories; second, it enjoys a specific algebraic theory that has been soundly and completely axiomatized [26,8].

[4] As in the abstract of this paper, we use the term 'framework' to mean not just a single process calculus (e.g. CCS) but a method or style for defining a family of such calculi.

Third, extending the well-established link between process calculi and modal logics, researchers are exploring the link between bigraphs and spatial logics [6,7].

The modelling of large-scale informatic systems is still at an experimental stage. Moreover, as with programming languages, the useful experiments are those carried out with real applications, involving real users and an assessment of their experience. With this in mind, a group [2] led by Lars Birkedal at the IT University of Copenhagen has embarked on the design and implementation [1] of a bigraphical language for specification and programming, and its implementation as a simulator. The first experiments with the language are now being carried out in their laboratory. In the same group bigraphs are also being applied experimentally to the modelling of business processes [14].

Finally, work is proceeding with a stochastic treatment of the behaviour of bigraphs [19], in the spirit of the stochastic κ-calculus [9]; it associates a stochastic rate to each reaction rule. This work shows how rates for labelled transitions can be derived uniformly, and applies the model to cell behaviour (membrane budding) in biology. Many applications of bigraphs, including biology, are non-deterministic; thus the stochastic treatment has special relevance to implementation, in order to yield useful simulation.

Acknowledgement

I gratefully acknowledge the Préfecture of the Île-de-France Region for the award of a Blaise Pascal International Research Chair, which has enabled me to clarify and advance this work. I held this post for one year in the Laboratoire d'Informatique at l'École Polytechnique in Paris. I warmly thank my hosts there, Professors Jean-Pierre Jouannaud and Catuscia Palamidessi, not only for the chance to devote my whole effort to research in collaboration with their teams, but also for organising—with Frank Valencia—the successful conference on *New Trends in Concurrency* (reported in this volume) to coincide with the beginning of my stay in France.

References

[1] Birkedal, L. Damgaard, C.D., Glenstrup, A.J. and Milner R, (2006), Matching of bigraphs. In: *Proc. Workshop in Graph Transformation for Verification and Concurrency*, Electronic Notes in Theoretical Computer Science, Elsevier.

[2] Birkedal, L. and Hildebrandt, T. (2004), Bigraphical programming languages. Laboratory for Context-Dependent Mobile Communication, IT University, Denmark. http://www.LaCoMoCo.itu.dk .

[3] Bundgaard, M. and Sassone, V. (2006), Typed polyadic pi-calculus in bigraphs. In: *Proc. 8th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, pp1–12.

[4] Cattani, G.L., Leifer, J.J. and Milner, R. (2000), Contexts and embeddings for closed shallow action graphs. University of Cambridge Computer Laboratory, Technical Report 496.

[5] Clavel, M., Eker, S., Lincoln, P. and Meseguer, J. (1996), Priniciples of Maude. In: J. Meseguer (ed.) *Proc. First International Workshop on Rewriting Logic and ite Applications*, Electronic Notes in Theoretical Computer Science 4, Elsevier.

[6] Conforti, G., Macedonio, D. and Sassone, V. (2005), Spatial logics for bigraphs. In: *International Conference on Automata, Languages and Programming*, pp766–778.

[7] Conforti, G., Macedonio, D. and Sassone, V. (2005), Bigraphical Logics for XML. In: *Proc. 13th Italian Symposium on Advanced Datebase Systems (SEBD)*, pp392-399.

[8] Damgaard, C.D. and Birkedal, L. (2006), Axiomatizing binding bigraphs. Nordic Journal of Computing 13(1–2), pp58–77.

[9] Danos, V., Feret, J., Fontana, W. and Krivine, J. (2007), Scalable modelling of biological pathways. In: Z. Shao (ed.), *Proceedings of APLAS*, 4807, pp139–157.

[10] Ehrig, H. (1979), Introduction to the theory of graph grammars. In: *Graph Grammars and their Application to Computer Science and Biology*, LNCS 73, Springer Verlag, pp1–69.

[11] Ehrig, H. (2002), Bigraphs meet double pushouts. EATCS Bulletin 78, October 2002, pp72–85.

[12] Gadducci, F., Heckel, R. and Llabrés Segura, M. (1999), A bi-categorical axiomatisation of concurrent graph rewriting. In: *Proc. 8th Conference on Category Theory in Computer Science (CTCS)*, Electronic Notes in TCS 29, Elsevier Science.

[13] Gadducci, F. and Montanari, U. (2000), The tile model. In: G. Plotkin, C. Stirling and M. Tofte (eds.) In: *Proof, Language and interaction*, MIT Press, pp133–166.

[14] Hildebrandt, T., Niss, H. and Olsen M. (2006), Formalising business process execution with bigraphs and reactive XML. In: *Proc. 8th International Conference on Coordination Models and Languages*, LNCS 4038, Springer Verlag, pp113–129.

[15] Hirsch, D. and Montanari, U. (2001), Synchronised hyperedge replacement with name mobility. In: *Proc. 12th International Conference on Concurrency Theory (CONCUR)*, LNCS 2154, pp121–136.

[16] Jensen, O.H. (2005), Forthcoming PhD Dissertation.

[17] Jensen, O.H. and Milner, R. (2003), Bigraphs and transitions. In: *30th SIGPLAN-SIGACT Symposium on Principles of Programming Languages*.

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

[19] Krivine, J., Milner, R. and Troina, A. (2007), Stochastic bigraphs. Submitted for publication (17pp).

[20] Leifer, J.J. (2001), Operational congruences for reactive systems. PhD Dissertation, University of Cambridge Computer Laboratory. Distributed in revised form as Technical Report 521. Available from `http://pauillac.inria.fr/~leifer`.

[21] Leifer, J.J. and Milner, R. (2000), Deriving bisimulation congruences for reactive systems. In: *Proc. CONCUR 2000, 11th International Conference on Concurrency Theory*, pp243–258. Available at `http://pauillac.inria.fr/~leifer`.

[22] Leifer, J.J. and Milner, R. (2004), Transition systems, link graphs and Petri nets. Mathematical Structures in Computer Science 16, pp989–1047.

[23] Meseguer, J. (1992), Conditional rewriting logic as a unified model of concurrency. Theoretical Computer Science 96, pp73–155.

[24] Milner, R. (1996), Calculi for interaction. Acta Informatica 33, pp707–737.

[25] Milner, R. (2001), Bigraphical reactive systems. In: *Proc. 12th International Conference on Concurrency Theory*, LNCS 2154, pp16–35.

[26] Milner, R. (2005), Axioms for bigraphical structure. Mathematical Structures in Computer Science 15, pp1005–1032.

[27] Milner, R. (2006), Pure bigraphs: Structure and dynamics. Information and Computation 204, pp60–122.

[28] Regev, A., Silverman, W. and Shapiro, E. (2001), Representation and simulation of biochemical processes using the π-calculus process algebra. In: *Proc. Pacific Symposium of Biocomputing 2001 (PSB2001)*, Vol 6, pp459–470.

[29] Sassone, V. and Sobocinski, P. (2002), Deriving bisimulation congruences: a 2-categorical approach. Electronic Notes in Theoretical Computer Science, Vol 68 (2), 19pp.

[30] Terese (2003) *Term Rewriting Systems*. Cambridge University Press.