

# Type Systems for Bigraphs<sup>\*</sup>

Ebbe Elsborg<sup>1</sup>, Thomas T. Hildebrandt<sup>1</sup>, and Davide Sangiorgi<sup>2</sup>

<sup>1</sup> IT University of Copenhagen (ITU)

<sup>2</sup> Università di Bologna

**Abstract** We propose a novel and uniform approach to type systems for (process) calculi, which roughly pushes the challenge of designing type systems and proving properties about them to the meta-model of *bigraphs*. Concretely, we propose to define type systems for the term language for bigraphs, which is based on a fixed set of *elementary bigraphs* and *operators* on these. An essential elementary bigraph is an *ion*, to which a *control* can be attached modelling its kind (its ordered number of channels and whether it is a guard), e.g. an input prefix of  $\pi$ -calculus. A model of a calculus is then a set of *controls* and a set of *reaction rules*, collectively a *bigraphical reactive system* (BRS). Possible advantages of developing bigraphical type systems include: a deeper understanding of a type system itself and its properties; transfer of the type systems to the concrete family of calculi that the BRS models; and the possibility of modularly adapting the type systems to extensions of the BRS (with new controls). As proof of concept we present a model of a  $\pi$ -calculus, develop an i/o-type system with subtyping on this model, prove crucial properties (including subject reduction) for this type system, and transfer these properties to the (typed)  $\pi$ -calculus.

## 1 Introduction

Type systems for calculi are important as they can: detect programming errors statically; and classify terms enabling extraction of information that is useful for reasoning rigorously about the behaviour and properties of programs, among other things. Type systems are usually engineered to enjoy subject reduction. The problem is that changing even small details of such a type system might ruin properties. Therefore, to feel confident that a tweak of the type system does not ruin any properties one really has to redo the proofs. This is often tedious. Many such type systems can be considered to be rather *ad hoc* so one would like a uniform way of proving properties of a whole family of calculi, simultaneously.

In this paper we experiment with a novel approach to type systems for (process) calculi, which roughly consists in pushing the problem of designing type systems and proving properties about them (such as subject reduction) to the more abstract level of *bigraphs* [9,8] by Milner and co-workers, a meta-model for (process) calculi. The main advantages are: a meta-model can describe several

---

<sup>\*</sup>This work was funded in part by the Danish Research Agency (grants no.: 2059-03-0031 and 274-06-0415) and the ITU (the LaCoMoCo/BPL and CosmoBiz projects).

concrete calculi, therefore one can hope that a result for a meta-model can be transferred to all of these calculi; and understanding type systems at the level of meta-models can help to achieve a deeper understanding of the type systems themselves. The theory of bigraphs is rich as its expressiveness has been demonstrated in several works in the literature; Petri nets [14,13],  $\pi$ -calculus [7,9,8], CCS [16], Mobile Ambients [7], Homer [3], and  $\lambda$ -calculus [17]. Importantly for our work, a sound and complete term language exists for bigraphs [15,5].

One models a calculus in bigraphs by encoding its terms as bigraphs and representing its *reduction* semantics by bigraphical *reaction rules*. All bigraphs are obtained by combining *elementary bigraphs* via the *operators* of categorical tensor product and composition. An essential elementary bigraph is an *ion*, to which a *control* can be attached modelling its kind (its ordered number of channels and whether it is a guard), e.g. an input prefix of  $\pi$ -calculus. The semantics of a concrete calculus is represented as reaction rules over a *signature* of controls.

A major effort so far has consisted in using bigraphs to automatically derive labelled transition semantics and congruential bisimilarities for concrete calculi with semantics defined by a reduction relation. In this paper we propose a novel use of bigraphs – to derive type systems for the concrete calculi. Our approach can be described in three phases: 1) Define a core BRS that can model the family of concrete calculi one is interested in. 2) Develop *bigraphical type systems* (BTSs) for this core BRS and prove their properties (such as subject reduction). 3) Transfer the type systems and their properties onto the concrete calculi of interest. Transferring the type system rules onto a concrete calculus  $C$  follows almost directly from the encoding of  $C$ 's terms into the BRS and from the typing rules of the BTS. Our approach requires a result of operational correspondence between a concrete calculus and its bigraphical model, which is the most basic and fundamental property to have when mapping a calculus into bigraphs. Hence, we provide a point of origin for studying type systems *for* (not *in*) bigraphs.

As proof of concept we study a *strict* (no summation), *finite* (no replication) and synchronous  $\pi$ -calculus, dubbed  $\text{sf}\pi$ , along with an *i/o*-type system with subtyping for its bigraphical model.  $\text{sf}\pi$  with *i/o*-types is well-suited for three reasons: the relationship between  $\text{sf}\pi$  and bigraphs has been well studied in the literature [7] allowing us to focus on type systems for bigraphs;  $\text{sf}\pi$  is simple but important because it maintains the essence of message-passing process calculi, and the *i/o*-type system with subtyping is technically interesting without being very complex. This constitutes a first study of non-trivial types for bigraphs.

*Related work* In [2] Debois et al. define a *sorting* as a functor from a *sorted s-category*, where *sorts* (think types) are assigned to interfaces (objects) as an extra component, into an unsorted *s-category*. A sorting refines which bigraphs (morphisms) may be composed and thus guarantees a certain structure of the well-sorted bigraphs. Hence, a sorting reduces the set of terms that are considered for reaction. Sortings are *not* defined inductively over bigraphs and give rise to different guarantees than traditional type systems in that they do not attempt to approximate dynamic behaviour of the terms. Thus, it is unclear whether one can recover existing type systems by sortings (in the general case).

In [4] Bundgaard and Sassone develop polyadic  $\pi$ -calculus with capability types and subtyping in bigraphs by: defining and proving safe a *link sorting* – called ‘subsorting’ – which is crucial in securing the desired i/o- and subtyping discipline; extending the theory of bigraphs by introducing controls on edges to retain the type information of restrictions. They inductively map *type derivations* of form  $\Gamma \vdash P : \diamond$  to *sorted* bigraphs by sending processes  $P$  to morphisms and typings  $\Gamma$  to sorted objects  $J$ . They also derive an LTS yielding a coinductive characterisation of a behavioural congruence for the calculus. A large effort in that work went into the sorting and the derivation of the LTS.

In [6] Igarashi and Kobayashi propose a generic type system (GTS) for  $\pi$ -calculus enjoying subject reduction and type soundness. They express typings  $\Gamma$  as (abstract) CCS-like processes and then check the properties on  $\Gamma$ . The GTS is parametrised over a subtyping preorder stating when two types have the same behaviour. By adding rules to the basic subtyping relation a type system instance for deadlock-freedom, among others, is obtained. This approach differs from ours in that they consider type systems for  $\pi$ -calculus and not for a meta-model, but we too wish to transfer general results to a family of calculi. In [12] König aims at generalising the concept of type systems to graph rewriting and in particular the concepts of type safety, subject reduction and compositionality. By working at the more abstract level of graphs rather than terms the author claims to be able to simplify the design of type systems, however we believe, at the cost of making it more difficult to transfer back and understand the type systems in terms of the concrete calculi.

In our approach we define type systems inductively on bigraph terms and can thus hope to: *directly* recover existing type systems; and have a computer verify whether a typed bigraph term is well-typed or not.

*Contributions* Our main contribution is conceptual: this work is a first attempt in the novel direction of *using bigraphs as a meta-model for type systems* through the first study of non-trivial inductive types for bigraphs. There are two main technical contributions: an i/o-type system (Tab. 2) for a core BRS capturing the essence of message-passing calculi; and a proof of Subject Reduction (Thm. 2) for this type system.

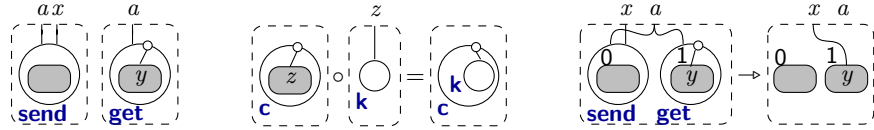
*Outline* In Sect. 2 we explain the necessary parts of bigraphs theory by example and then we present a model of  $\text{sf}\pi$ . On this foundation we develop an i/o-type system for the model, prove important properties of it, and transfer these to i/o-typed  $\text{sf}\pi$ , all in the main Sect. 3. Finally, in Sect. 4, conclusions are drawn and directions for future work outlined. Due to space constraints most proofs and background definitions appear only in the full version<sup>1</sup> of this paper.

## 2 Bigraphs

Bigraphs is a model of computation that emphasis both *locality* and *connectivity* aiming at trustworthy (safe and reliable) computation in global ubiquitous

<sup>1</sup><http://www.itu.dk/people/elsborg/ITU-TR-2008-110.pdf>

computers [20,1], in which highly dynamic topologies and heterogeneous devices are prominent. Mobile locality is captured by a *place graph* and mobile connectivity by a link *link graph*, two largely orthogonal structures that combine into a bigraph. The place graph is an ordered forest of trees representing nested locations of computational nodes, and the link graph is a hypergraph representing interconnection of these nodes. Dynamics are added to bigraphs by defining (parametric) reaction rules. Consider Fig. 1. It depicts two *ions*, bigraph composition



**Figure 1.** The ions *send* and *get*, bigraph composition, and the  $\text{sf}\pi$  reaction rule.

sition, and a reaction rule involving the ions. The two ions, depicted with solid circles, model output prefix and input prefix of  $\pi$ -calculus, respectively. Each ion consists of a *node* assigned a *control* determining its kind. In this case, both controls have two ordered *ports* to which *links* (channels) can be attached. *send* has its (*free*) ports linked to *local outer names*  $a$  (the ‘channel’ port) and  $x$  (the ‘datum’ port), respectively. *Global names* are like unrestricted names in  $\pi$ -calculus, whereas local (think abstracted) names reside at *regions/roots* (dotted rectangle) or *sites* (greyed rectangles). *get* has a *binding port*, which binds a *local inner name*  $y$  with lexical scope below this node in the place graph, and thus resembles a variable of programming languages. Both ions are contexts with a site (hole) into which another suitable bigraph can be “plugged”, yielding another bigraph. This is known as vertical *composition*,  $b_1 \circ b_0$ , and proceeds by plugging the roots of  $b_0$  into the sites of  $b_1$  (in order), and fusing together the outer names of  $b_0$  with the inner names of  $b_1$ , removing the names in the process. The sites and inner names of a bigraph  $b$  are collectively called the *inner face* or *domain* ( $\text{dom}(b)$ ); similarly, the regions and outer names are called the *outer face* or *codomain* ( $\text{cod}(b)$ ). Then,  $b_1 \circ b_0$  requires  $\text{cod}(b_0) = \text{dom}(b_1)$ . The second column of Fig. 1 shows an example; given  $b_1 = \mathbf{c}_{(z)} : \langle 1, (\{z\}), \{z\} \rangle \rightarrow \langle 1, (\emptyset), \emptyset \rangle$  and  $b_0 = (\{z\})\mathbf{K}_z : \epsilon \rightarrow \langle 1, (\{z\}), \{z\} \rangle$  then  $b_1 \circ b_0 : \epsilon \rightarrow \langle 1, (\emptyset), \emptyset \rangle$ . The interface (or *face*) components are: a *width*; a vector of local name sets drawn from the global name set; and a global name set. They are projected by functor **width**, function *loc*, and function *glob*, respectively. Function *glob* projects all names of a face.

A notion that is not shown in Fig. 1 is an *edge* (think restricted name); inner names  $X$  and ports  $P$  can point to edges  $E$  instead of outer names  $Y$ , via the so-called *link* map,  $\text{link} : X \uplus P \rightarrow E \uplus Y$ . If the name  $x$  is *closed* (restricted) then it becomes invisible to the context and any ports which were pointing to this outer name will now instead point to an edge. Edges have no name associated

with them, just in the term language to denote which points map to which edges. An edge is a “floating” binder in that it has no lexical scope.

When representing a calculus in bigraphs one is usually interested in bigraph *terms* that are *ground* and *prime* (also known as *agents*), i.e. bigraph terms that have no sites, no inner names, and outer width 1. Regions (or sites) can be juxtaposed (composed horizontally) by the binary operator *tensor product*  $\otimes$ , if the operands have disjoint name sets (both outer and inner). A derived operator is the *prime product*  $|$ , which takes two regions as operands, but allows them to share outer names, and also collapses the two regions into one, while acting as tensor on sites. The third (basic) operation on binding bigraphs is *abstraction*  $(X)P$  on a *prime*  $P$ , which localises a subset of the global names of  $P$ . A face of width 0 without names is denoted by the unique object  $\epsilon$ .

The reaction rule models communication in  $\text{sf}\pi$ . The *redex* has one region signifying that a *send* and a *get* must be collocated and connected to be able to communicate. The *reactum* shows that the bigraph has performed an action, which has depleted the input/output capability. The outer name  $a$  is *idle* in the reactum, i.e. not pointed to by anything, and the inner name  $y$  points to the outer name  $x$ , explicitly representing meta-level name substitution in  $\pi$ -calculi.

In Def. 1 reaction rules are defined formally. It uses the notion of *support equivalence*, which for our purposes can be thought of as bigraph equality. Intuitively, a context  $D$  is *active* w.r.t a (ground) bigraph  $r$  if the sites of  $D$  into which  $r$  is plugged are active, and sites are active if the path to the root in the place graph only has (nodes with) active controls. An *instantiation* essentially maps sites of the reactum to sites of the redex, including the possibly renamed local names of the reactum sites. A *discrete parameter*  $d$  is a ground bigraph with no edges and a bijective link map.

**Definition 1 (reaction rules for bigraphs, [9])** A ground (reaction) rule is a pair  $(r, r')$ , where  $r$  and  $r'$  are ground rules with the same outer face. Given a set of ground rules, the reaction relation  $\rightarrow$  over agents is the least, closed under support equivalence ( $\simeq$ ), such that  $D \circ r \rightarrow D \circ r'$  for each active  $D$  and each ground rule  $(r, r')$ .

A parametric (reaction) rule has a redex  $R$  and a reactum  $R'$ , and takes the form

$$(R : I \rightarrow J, R' : I' \rightarrow J, \varrho)$$

where the inner faces  $I$  and  $I'$  are local with widths  $m$  and  $m'$ . The third component  $\varrho :: I \rightarrow I'$  is an instantiation. For every  $X$  and discrete  $d : X \otimes I$  the parametric rule generates the ground reaction rule

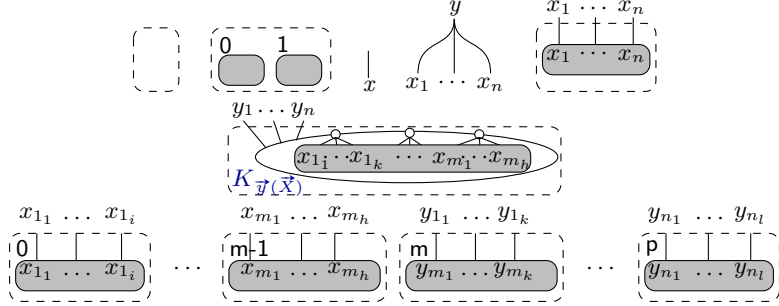
$$( (\text{id}_X \otimes R) \circ d, (\text{id}_X \otimes R') \circ \varrho(d) ) .$$

Reaction is defined over *concrete* bigraphs, i.e. bigraphs where the nodes and edges have identity. However, we are interested in *abstract* bigraphs. Whenever  $b_0 \simeq b_1$  concretely we have  $b_0 = b_1$  abstractly. Notice that inner faces are local.

A signature is a set of controls each with: an *arity map* from its number  $f$  of free ports to its number  $b$  of binding ports; and an *activity map* determining whether it is *active* (an evaluation context), *passive* (guard), or *atomic* (a term).

Bigraphs have an algebraic representation. All bigraphs can be generated from seven *elementary* bigraphs combined by (categorical) tensor product and composition. One can think of these elementary bigraphs and the operations on them as basic building blocks (language concepts) for processes and operators on processes. The faces of the bigraphs determine when tensor product, composition, and abstraction are well-defined. Bigraphs are  $\alpha$ -equivalence classes of *bigraph terms*. The elementary bigraphs are depicted graphically in Fig. 2 and as syntactic terms with algebraic faces in Tab. 1.

**Notation 1 (Placing, linking, wiring, sets)** *For interfaces we often omit: names from placings (node-free place graphs); widths from linkings (node-free linkgraphs); the enclosing  $\langle$  and  $\rangle$  when the width is zero. A wiring is a bigraph with zero width generated by composition and tensor of linkings. Curly brackets are often omitted for singleton sets and names on ions. Sets (usually of names or types) are denoted by capital letters such as  $X, Y, Z$  and  $S, T, U$ , ranged over by minuscule letters. We write  $XY$  for the disjoint union  $\uplus$  of sets  $X$  and  $Y$ .*



**Figure 2.** The seven elementary binding bigraphs, graphically.

**Definition 2 (Flattening)** *Given a vector  $\vec{x}$  of distinct names we write  $\{\vec{x}\}$  for the corresponding (one-to-one) set. Given a vector  $\vec{X} = (X_1, \dots, X_n)$  of disjoint name sets we define their disjoint union as  $\{\vec{X}\} \stackrel{\text{def}}{=} \uplus_{i=1}^n X_i$ .*

Consider Fig. 2. First row: a barren root **1** is an empty region. When plugging a bigraph with two regions and no outer names into *join*, the two regions are merged into one. Name closure  $/x$  acts as a non-lexical binder; it is put on top of a bigraph with a global inner name  $x$  and closes (restricts) this name rendering it invisible to the context. *Substitution*  $y/x$  links a set of global inner names  $X$  to a single global outer name  $y$  by a hyperlink;  $y$  is “substituted for” any  $x \in X$ ; the widths are zero;  $X$  is bound inwards; and  $y$  binds outwards. A special case is  $y/\emptyset$  which introduces an idle name. The *concretion*  $\lceil X \rceil$  bigraph globalises a set of local names  $X$ , dually to the abstraction operator. Second row: an ion  $K_{\vec{y}/\vec{X}}$  is a prime bigraph with a single node of control  $K$  with free ports linked

$\mathbf{1}$	$: 0 \rightarrow 1$	barren root
$join$	$: 2 \rightarrow 1$	join two sites
$/x$	$: x \rightarrow \emptyset$	close global outer name $x$
$y/x$	$: X \rightarrow y$	link all names in global name set $X$ to global name $y$
$\ulcorner X \urcorner$	$: (X) \rightarrow \langle X \rangle$	globalise local outer name set $X$
$K_{\vec{y}(\vec{X})}$	$: (\{\vec{X}\}) \rightarrow \langle \{\vec{y}\} \rangle$	an ion with local name sets $\vec{X}$ and global names $\vec{y}$
$\gamma_{m,n}(\vec{X}, \vec{Y})$	$: \langle m+n, \vec{X}\vec{Y}, \{\vec{X}\} \uplus \{\vec{Y}\} \rangle \rightarrow \langle m+n, \vec{Y}\vec{X}, \{\vec{X}\} \uplus \{\vec{Y}\} \rangle$	transpose $m$ with $n$ regions or sites, keep local names

**Table 1.** The seven elementary binding bigraphs as terms.

severally to a vector  $\vec{y}$  of distinct outer names, and each binding port linked to all local inner names in name set  $X_i$  a vector  $\vec{X}$  of sets of distinct names. Ions are the essence of BRSs as they usually model the interesting entities of systems or calculi. Third row: a *transposition*  $\gamma_{m,n}(\vec{X}, \vec{Y})$  transposes regions keeping their sites and local names. From here on we think of bigraphs represented as terms.

## 2.1 A Bigraphical Model of $\text{sf}\pi$

We consider  $\text{sf}\pi$ . Following [7] we add an axiom to the usual structural congruence:  $\nu x (\pi.P) \equiv \pi.\nu x P$ , if  $x \notin (\text{fn}(\pi) \cup \text{bn}(\pi))$ . This axiom naturally complements the similar axiom for parallel composition and secures that structural congruence coincides with graph isomorphism yielding a nice graphical representation of bigraphs. Equivalences on processes remain unchanged even though more processes are related by  $\equiv$  with this axiom. This axiom is not important for our development. However, we remark that to represent, e.g., a replicated input prefix in bigraphs one needs an *outward-binding control* [7]. Processes that are  $\alpha$ -convertible are identified.

We model  $\text{sf}\pi$  with the BRS of [7] (a signature  $\Sigma_{\text{sf}\pi}$  and a set of reaction rules  $\mathcal{R}_{\text{sf}\pi}$ ) but name it  $\text{Bbg}_{\text{sf}\pi} \stackrel{\text{def}}{=} \text{Bbg}(\Sigma_{\text{sf}\pi}, \mathcal{R}_{\text{sf}\pi})$ . Prefixes (input and output) are modelled by the *passive* controls *send* and *get* of Fig. 1, because prefixes are guards. *get* has a binding port. The semantics is modelled by a single reaction rule, where prime product models parallel composition, name closure models restriction, and an *insertion operator*  $\triangleleft$  inserts a wiring into a bigraph  $b$  making composition  $bob'$  well-defined, typically by “wiring through” (and then localising) outer names of  $b'$  that are not to be lexically bound by  $b$ .

**Definition 3** ( $\text{Bbg}_{\text{sf}\pi}$ , [7])

$$\begin{aligned} \Sigma_{\text{sf}\pi} &\stackrel{\text{def}}{=} \{ \text{send} : 0 \rightarrow 2 \text{ (passive)}, \text{get} : 1 \rightarrow 1 \text{ (passive)} \} \\ \mathcal{R}_{\text{sf}\pi} &\stackrel{\text{def}}{=} (R, R', \varrho) = (\text{send}_{ax} \mid \text{get}_{a(y)} : \langle 2, (\emptyset, \{y\}), \emptyset \rangle \rightarrow \langle 1, (\{a, x\}), \emptyset \rangle, \\ &\quad (\text{id}_1 \mid \text{id}_1 \triangleleft x/y) \triangleleft a : \langle 2, (\emptyset, \{y\}), \emptyset \rangle \rightarrow \langle 1, (\{a, x\}), \emptyset \rangle, \\ &\quad \text{id}_{\langle 2, (\emptyset, \{y\}), \emptyset \rangle}) . \end{aligned}$$

This (parametric) reaction rule is *linear*, because its instantiation is bijective, so no parameters are replicated or discarded. It is parametric to model arbitrary subterms under prefixes. In bigraph *terms* the names  $a$ ,  $x$ , and  $y$  are not metavariables so we stipulate that  $a \neq x \neq y$ , because the bigraph corresponding to the term is different in the cases where some of these are equal. If  $a$  and  $x$  need to be identified for a reaction, then the context does it. We name the morphisms (bigraphs) of  $\mathbb{B}BG_{sf\pi}$  *process bigraphs*.

Processes are mapped to bigraph terms by the compositional, semantic function  $\llbracket \cdot \rrbracket$  of Def. 4. For technical reasons the inactive process is modelled by  $(X)$ , an empty ground local prime.

**Definition 4 (Encoding  $sf\pi$  in  $\mathbb{B}bg_{sf\pi}$ , [7])** *The function  $\llbracket \cdot \rrbracket_{(X)}$  maps every process  $P$  of  $sf\pi$  with  $\text{fn}(P) \subseteq X$  into the homset  $(\epsilon, (X))$  of  $\mathbb{B}BG_{sf\pi}$  as follows:*

$$\begin{aligned} \llbracket \bar{a}x.P \rrbracket_{(X)} &= \text{send}_{ax} \triangleleft \text{id}_X \circ \llbracket P \rrbracket_{(X)} & \llbracket P \mid Q \rrbracket_{(X)} &= \llbracket P \rrbracket_{(X)} \mid \llbracket Q \rrbracket_{(X)} & \llbracket \mathbf{0} \rrbracket_{(X)} &= (X) \\ \llbracket a(y).P \rrbracket_{(X)} &= \text{get}_{a(y)} \triangleleft \text{id}_X \circ \llbracket P \rrbracket_{(Xy)} & \llbracket \nu xP \rrbracket_{(X)} &= / (x) \triangleleft \text{id}_X \circ \llbracket P \rrbracket_{(Xx)} \end{aligned}$$

The translation of  $P$  is indexed by a (local) name set  $X \supseteq \text{fn}(P)$  that is needed to secure dynamic correspondence between  $sf\pi$  and the model. This is because reduction in  $sf\pi$  can discard a channel (name) after use, i.e. reduce the set of free names, but outer faces (of agents) are preserved by bigraphical reaction rules so here the name persists, although idle. (For an example see [9].)  $P$  will have an image for each choice of  $X$ , i.e. countably many bigraphs. Not unusually, the translation requires that  $\text{bn}(P) \cap \text{fn}(P) = \emptyset$  and unique binding names. The model enjoys structural and dynamic correspondence theorems, here combined.

**Theorem 1 (Correspondence, [7]).**

1. *The function  $\llbracket \cdot \rrbracket_{(X)}$  is surjective onto the homset  $(\epsilon, (X))$  of  $\mathbb{B}BG_{sf\pi}$  ;*
2.  *$P \equiv Q$  iff  $\llbracket P \rrbracket_{(X)} = \llbracket Q \rrbracket_{(X)}$ .*
3. *Given  $X \supseteq \text{fn}(P)$ , then  $P \longrightarrow P'$  iff  $\llbracket P \rrbracket_{(X)} \rightarrow \llbracket P' \rrbracket_{(X)}$ .*

We are now ready to define a type system on  $\mathbb{B}BG_{sf\pi}$ .

### 3 A Bigraphical i/o-Type System

In this main section we develop a bigraphical i/o-type system and prove important properties of it.

**Definition 5 (Type environment)** *A type environment (or typing) is an unordered finite assignment of types to names, ranged over by  $\Gamma$  and  $\Delta$ . The support  $\text{supp}(\Gamma)$  is the set of names. When regarded as a finite function from names to types we write  $\Gamma(x)$  for the type assigned to  $x$  by  $\Gamma$ . The extension of  $\Gamma$  with the assignment  $x : T$  is denoted  $\Gamma, x : T$  when  $x \notin \text{supp}(\Gamma)$ . The disjoint union  $\Gamma, \Delta$  is defined when  $\text{supp}(\Gamma) \cap \text{supp}(\Delta) = \emptyset$ , and is (also) associative and commutative.  $\Gamma_\emptyset$  denotes the empty typing.*

**Definition 6 (Syntax of types and typings)**

$$T ::= V \mid L \quad V ::= L \mid \bullet \quad L ::= \#V \mid \circ V \mid \text{i}V \quad \Gamma ::= \Gamma, x : V \mid \Gamma, x : L \mid \Gamma_\emptyset .$$

A *link* is a name that may be used for communication. The *values* are the objects (names) that can be communicated along links. The *link types* ( $L$ ) are the types that can be ascribed to links. The *value types* ( $V$ ) are the types that can be ascribed to values (names). Link types are value types so that processes can exchange links, allowing mobility. Links can either be used in input  $\text{i}V$ , output  $\circ V$ , or both  $\#V$  (the *connection* type). The inhabitants of unit type  $\bullet$  are names. Names assigned type  $\bullet$  are “base values” that can only be passed around. There is no special unit value as this would clutter the presentation.

In message-passing process calculi the channels (links) are the essential part because communication is the primitive notion studied. Therefore, in the present paper, we only type links, not nodes. Tab. 2 presents the  $\text{i}/\circ$ -typing rules for  $\mathbb{B}\mathbb{G}_{\text{sf}\pi}$ . The idea is to syntactically define types for elementary bigraphs and the operators on them following their structure inductively. The  $\text{i}/\circ$ -type system guarantees that ions (images of processes) use their links in accordance with their capabilities on them. The subtyping preorder  $\leq$  can be thought of as inclusion between the sets of the values of the types. So we would have, e.g.,  $\text{Int} \leq \text{Real}$ . We write  $\Gamma_1 \leq \Gamma_0$  whenever  $\text{supp}(\Gamma_1) = \text{supp}(\Gamma_0) = X$  and  $\forall x \in X. \Gamma_1(x) \leq \Gamma_0(x)$ .

**Definition 7 (Judgments)** A bigraph type judgment is of the form  $\Delta; \Gamma \vdash b$ , where  $b$  is a bigraph term. A name type judgment is of the form  $\Gamma \vdash x : T$ .

Consider Tab. 2. Bigraphs are contexts and thus typed by two typings; a typing  $\Delta$  of the inner names and one  $\Gamma$  for the outer names. When assigning types it does not matter whether a name is local or global, we just type the third component of interfaces, projected by *glob*. The type system is *strong* in the sense that the typings carry no information about names that do not appear in the interfaces of the bigraph. The reason for this will become clear later when we treat properties of the type system. In the following we refer to the axioms that govern bigraphical term equality, which are defined in [5,15].

Nameless elementary bigraphs, i.e. the barren root and *join*, are typed using two empty typings. Transpositions allow subtyping because they partially coincide with identities (by axioms (C7)  $\gamma_{I,\epsilon} = \text{id}_I$  and (C8)  $\gamma_{J,I} \circ \gamma_{I,J} = \text{id}_{I \otimes J}$ ), which in turn partially coincide with substitutions (by axioms (L1)  $x/x = \text{id}_x$  and (L3)  $/y \circ y = \text{id}_\epsilon$ ), and substitutions must allow subtyping, see below. We write  $\gamma_Z$  for  $\gamma_{m,n,(\vec{X},\vec{Y})}$  when we are merely interested in the names collectively.

A closure – name creation – can be given any link type. Actually, it is only useful when it is a connection type ( $\#V$ ) because for two processes to communicate over a link one needs to use the link for output and the other for input, simultaneously. Bigraphical substitutions  $y/x$  demand that all  $x \in X$  have the same type  $T$  (denoted by  $X : T$ ), which is natural because substituting in a  $y$  for any  $x$  really identifies these  $x_j$ , namely they are  $y$  from the viewpoint of the context. In harmony with the  $\text{i}/\circ$ -subtyping discipline we must be able to assign to  $y$  a subtype of the  $x_j$  so as to allow substitution of names with a possibly

<i>Placings</i> :	$\frac{}{\Gamma_\emptyset; \Gamma_\emptyset \vdash \mathbf{1}}$	$\frac{}{\Gamma_\emptyset; \Gamma_\emptyset \vdash \text{join}}$	$\frac{\Gamma_1 \leq \Gamma_0 \quad \text{supp}(\Gamma_j)^{j=0,1} = Z}{\Gamma_0; \Gamma_1 \vdash \gamma_Z}$
<i>Linkings</i> :	$\frac{}{x : L; \Gamma_\emptyset \vdash /x}$		$\frac{\Gamma \vdash y : T}{X : T; \Gamma \vdash y/x}$
<i>Id &amp; Conc.</i> :	$\frac{\Gamma_1 \leq \Gamma_0 \quad \text{supp}(\Gamma_j)^{j=0,1} = \text{glob}(I)}{\Gamma_0; \Gamma_1 \vdash \text{id}_I}$	$\frac{\Gamma_1 \leq \Gamma_0 \quad \text{supp}(\Gamma_j)^{j=0,1} = X}{\Gamma_0; \Gamma_1 \vdash \ulcorner X \urcorner}$	
<i>Operators</i> :	$\frac{\Delta; \Gamma \vdash b}{\Delta; \Gamma \vdash (X)b}$	$\frac{\Delta_0; \Gamma_0 \vdash b_0 \quad \Delta_1; \Gamma_1 \vdash b_1}{\Delta_0, \Delta_1; \Gamma_0, \Gamma_1 \vdash b_0 \otimes b_1}$	$\frac{\Gamma_0; \Delta \vdash b_0 \quad \Delta; \Gamma_1 \vdash b_1}{\Gamma_0; \Gamma_1 \vdash b_1 \circ b_0}$
<i>Ions</i> :	$\frac{\Gamma \vdash a : \text{o}T \quad \Gamma' \vdash x : T}{\Gamma_\emptyset; \Gamma, \Gamma' \vdash \text{send}_{ax}}$		$\frac{\Gamma \vdash a : \text{i}S}{y : S; \Gamma \vdash \text{get}_{a(y)}}$
<i>Subtyping</i> :	$\frac{}{T \leq T}$	$\frac{S \leq U \quad U \leq T}{S \leq T}$	$\frac{}{\#T \leq \text{i}T} \quad \frac{}{\#T \leq \text{o}T}$
	$\frac{S \leq T}{\text{i}S \leq \text{i}T}$	$\frac{T \leq S}{\text{o}S \leq \text{o}T}$	$\frac{T \leq S \quad S \leq T}{\#S \leq \#T}$
<i>Names</i> :	$\frac{S \leq T}{x : S \vdash x : T}$		

**Table 2.** i/o-typing rules for  $\text{BBG}_{\text{sf}\pi}$ .

smaller (i.e. more general) capability. In a sense this corresponds to the usual substitution lemma for  $\pi$ -calculi.

Identities and concretions allow subtyping. Concretions merely globalise outer names but are allowed to subtype. This enables a Narrowing lemma.

Localising names does not affect types so the rule for abstraction is straightforward. The rule for tensor product splits the typing in its two branches according to the names of each tensor component. The rule for composition demands the types of the common interface to be identical, which is natural when considering that bigraphs are really categorical morphisms between objects (interfaces).

The rules for ions are essential as they type the prefixes. Channels are forced to be of output and input type, respectively. Notice the asymmetry between how  $x$  and  $y$  are typed; the type of  $y$  is fixed in the inner typing because it is a binder. Just like the cases for the inner typings of closure and substitution.

The rule for names encompasses subsumption because the typings are strong rendering obsolete the need to have two separate rules.

This type system differs from traditional type systems, e.g. the i/o-type system of  $\pi$ -calculus (see e.g. [19]) in the following respects: 1) We type contexts, not terms, and therefore we have to account for (categorical) composition. 2) Explicit substitution  $y/x$  is a syntactic term and hence needs to be typed. This

fundamental difference is important because it pervades the properties of the type system in that subtyping of substitution in a sense represents a Substitution Lemma. 3) The tensor product is more fundamental than parallel product. 4) There is a distinction between local and global names. An important insight is that a name  $x \in glob(dom(b))$  and *another*  $x \in glob(cod(b))$  are really two different names if they are not linked in  $b$ .

The type system enjoys two crucial properties; subject reduction and type soundness. These results rest upon the Main Lemma establishing that the bigraphical typing relation is closed under bigraph *term* equality, which in turn requires *Narrowing* and *Widening*. The typing and subtyping relations enjoy *Inversion*, i.e. can be read “bottom-up”, because they are syntax-directed.

**Lemma 1 (Narrowing).** *If  $\Delta; \Gamma, x : T \vdash b$  and  $S \leq T$  then  $\Delta; \Gamma, x : S \vdash b$ .*

**Lemma 2 (Widening).** *If  $\Delta, x : S; \Gamma \vdash b$  and  $S \leq T$  then  $\Delta, x : T; \Gamma \vdash b$ .*

Widening is unusual (for process calculi) in that it is defined on contexts. It is in a sense the dual lemma to Narrowing as it allows widening of inner typings.

The congruence relation  $=$  of the Main Lemma is the involved, axiomatised bigraph equality on terms (see [5,15]). The Main Lemma states that if two bigraph terms are equal then they can be typed in the same environments so the type system is robust w.r.t. bigraph equality: term equality on bigraphs coincides with graph isomorphism so this lemma allows us to think of types on the underlying graphs. This lemma is (technically) crucial and not one usually found for type systems for  $\pi$ -calculus.

**Lemma 3 (Main Lemma).** *Suppose  $b_0 = b_1$ . Then  $\Delta; \Gamma \vdash b_0$  if and only if  $\Delta; \Gamma \vdash b_1$ .*

Cor. 1 of the Main Lemma tells us that the type system is robust w.r.t. decomposition of the term as a graph, which is important for Subject Reduction.

**Corollary 1 (Decompositionality)** *If  $\Delta; \Gamma \vdash b$  and  $b = b_1 \circ b_0$  then there exists a typing  $\Theta$  such that  $\Delta; \Theta \vdash b_0$  and  $\Theta; \Gamma \vdash b_1$ .*

Before stating and proving a subject reduction theorem we consider the grounded rules generated by the parametric rule of Def. 3, because the type derivations of this rule’s redex and reactum are a key to understanding the proof of the subject reduction theorem. The generated ground rules are of form  $(r, r')$ :

$$\begin{aligned} & ((\text{id}_X \otimes R) \circ d, (\text{id}_X \otimes R') \circ \varrho(d)) \\ \stackrel{\text{def}}{=} & ((\text{id}_X \otimes (\text{send}_{ax} \mid \text{get}_{a(y)})) \circ d, (\text{id}_X \otimes ((\text{id}_1 \mid \text{id}_1 \triangleleft x/y \triangleleft a)) \circ \varrho(d)) \\ \stackrel{\text{def}}{=} & ((\text{id}_X \otimes ((\text{join} \otimes \text{id}_{(ax)}) \circ (\sigma \circ (\text{send}_{ax} \otimes (\tau \circ \text{get}_{a(y)})))) \circ d, \\ & (\text{id}_X \otimes ((\text{join} \otimes \text{id}_{(ax)}) \circ (((\text{join} \otimes \text{id}_{(x)}) \circ (\text{id}_1 \otimes (x)/(y))) \otimes (a)))) \circ \varrho(d)) \end{aligned}$$

where  $\{a, x\} \cap X = \emptyset$ ,  $\tau = (a')/(a)$ , and  $\sigma = (a)/(\{a, a'\}) \otimes (x)/(x)$  w.l.o.g. We remark that  $(x)/(y) \stackrel{\text{def}}{=} (x)(x/y \otimes \text{id}_1) \circ \ulcorner y \urcorner$ . Subject Reduction (Thm. 2) is the main

theorem and guarantees that typings are preserved over reaction. The core in the proof of the theorem is an analysis of redex and reactum as the type derivations of the context, and in this case also the parameters, are preserved by reaction. Hence, the theorem is really a property of reaction rules. For a BRS with multiple (possibly overlapping) reaction rules one would analyse the redex-reactum pair of each one and then simply combine the results to obtain the theorem.

**Theorem 2 (Subject Reduction).** *For process bigraphs  $b_0$  and  $b_1$ , if  $\Gamma_\emptyset; \Delta \vdash b_0$  and  $b_0 \rightarrow b_1$  then  $\Gamma_\emptyset; \Delta \vdash b_1$ .*

*Proof.* The proof is by analysis of the derivation of  $b_0 \rightarrow b_1$  by the sole reaction rule. Because  $b_0 \rightarrow b_1$ , then by Def. 1 there exists an active context  $D$  such that  $b_0 = D \circ r$  and  $b_1 \simeq D \circ r'$ . Assume a derivation of  $\Gamma_\emptyset; \Delta \vdash b_0$ , then also (\*)  $\Gamma_\emptyset; \Delta \vdash D \circ r$  by Lemma 3. By Inversion we must have (among others) the following six subderivations from (\*): (1')  $\Gamma_\emptyset; \Gamma, y : S \vdash d$ , (3')  $a : U \vdash a : \circ T$ , (3'')  $x : U' \vdash x : T$ , (4')  $a : R \vdash a : \circ S$ , (5')  $U \leq R$ , and (8')  $\Gamma', a : W, x : W'; \Delta \vdash D$ . We also know that  $W \leq U$  and  $W' \leq U'$ . By (3'), (5') and (4') we conclude  $T \leq S$  (cf. [19]).  $W' \leq U'$ , and by (3'') we have  $U' \leq T \leq S$ , so  $W' \leq S$ .

Now, consider the derivation to be built.  $b_1 \simeq D \circ r'$  implies that  $b_1 = D \circ r'$  abstractly. By Lemma 3 it suffices to derive  $\Gamma_\emptyset; \Delta \vdash D \circ r'$ . Reuse the derivation of  $D$ .  $\varrho = \text{id}_{\langle 2, \{\emptyset, \{y\}\}, \emptyset \rangle}$  so  $\varrho(d) = d$ . This means that we can also reuse (1'). We still need to justify a derivation of  $y : S; x : W' \vdash (x)/(y)$ . This merely requires justification of  $W' \leq S$  because we may choose not to subtype in the other substitutions.  $W' \leq S$  has already been established so we can build the desired derivation of  $\Gamma_\emptyset; \Delta \vdash D \circ r'$ .  $\square$

We remark that the inner typings are preserved because the reaction rule is linear, but that need not be the case in general, where the theorem could instead relate the inner typings by something weaker than equality (since sites, including local inner names, can be discarded or replicated and renamed).

Type Soundness (Prop. 1) states that a process bigraph  $b$  well-typed in  $\Gamma_\emptyset; \Gamma$  can only perform input or output actions for which  $\Gamma$  offers the appropriate capabilities. Let  $\rightarrow^*$  be the reflexive, transitive closure of  $\rightarrow$ .

**Proposition 1 (Type Soundness)** *Suppose that process bigraph  $b = \llbracket P \rrbracket_{(X)}$ ,  $\Gamma_\emptyset; \Gamma \vdash b$ , and  $b \rightarrow^* b'$ . Then, for each non-idle  $a \in \text{glob}(\text{cod}(b'))$  it holds that:*

1. *If  $\Gamma \vdash a : \circ S$  then  $a$  is either linked to the channel port of a get ion or linked to the datum port of a send ion.*
2. *If  $\Gamma \vdash a : \circ T$  then  $a$  is linked to a send ion.*

*Proof (Sketch).* The proof is by ind. on the length of the reduction  $b \rightarrow^* b'$ . The base case is by struct. ind. on  $P$  using Lemma 3, Cor. 1, and Lemma 1. The inductive case uses Thm. 2.

Type Soundness gives guarantees about outer names, but not closed names because edges have no type. To achieve a stronger type soundness property – such as “well-typed processes do not reduce to *wrong*” – one could introduce a tagged

version of the BRS in which each name is permanently tagged with the intended i/o usage, like in [18] for  $\pi$ -calculus. Or, we could follow [4] and type edges to possibly obtain a result of intermediate strength.

Idle names are merely the residue of reaction in bigraphs so adding or removing them corresponds, in a precise way to be shown below, to Weakening and Strengthening of a type system in  $\pi$ -calculus. Adding and removing idle names actually changes the bigraph (a context) because the codomain changes. These different bigraphs should however correspond to the same source calculus term because they only differ up to names that do not occur in the source term.

**Lemma 4 (Weakening).** *If  $\Delta; \Gamma \vdash b$  and  $x \notin \text{supp}(\Gamma)$  then  $\Delta; \Gamma, x : T \vdash b \otimes (x)$ .*

**Lemma 5 (Strengthening).** *If  $\Delta; \Gamma, x : T \vdash b \otimes (x)$  then  $\Delta; \Gamma \vdash b$ .*

Even though these two properties on the surface appear different from those of  $\pi$ -calculi they really do correspond to the usual properties of typed  $\text{sf}\pi$ .

With these important properties in hand it is time to transfer them to the i/o-typed source calculus  $\text{sf}\pi$ . The standard way to map an untyped process calculus into bigraphs is to consider a trivial type system for the process calculus with just a single type and map derivations of form  $\Gamma \vdash P : \diamond$  (see e.g. [19]) to the (untyped) bigraph  $\llbracket P \rrbracket_{(X)}$  exactly when  $\text{fn}(P) \subseteq X = \text{supp}(\Gamma)$ . The choice  $X = \text{supp}(\Gamma)$  coerces a connection between a process bigraph and its (outer) typing. Names in  $\text{supp}(\Gamma) \setminus \text{fn}(P)$  become idle names in  $\llbracket P \rrbracket_{(X)}$  by the translation, recalling that  $\llbracket \mathbf{0} \rrbracket_{(X)} = (X)$ . This is made precise by Lemma 6.

**Lemma 6.** *Suppose  $b = \llbracket P \rrbracket_{(X)}$  and  $\Gamma_\emptyset; \Gamma \vdash b$  with  $\text{fn}(P) \subseteq X = \text{supp}(\Gamma)$ . Then  $\llbracket P \rrbracket_{\text{supp}(\Gamma)} \otimes (x) = \llbracket P \rrbracket_{(\text{supp}(\Gamma, x:T))}$  for any  $T$ .*

Using Lemma 6 we conclude:  $\llbracket P \rrbracket_{(X)} \otimes (x) = \llbracket P \rrbracket_{(\text{supp}(\Gamma))} \otimes (x) = \llbracket P \rrbracket_{(\text{supp}(\Gamma, x:T))}$  for any type  $T$ . Then, by Lemma 3 we have that  $\Delta; \Gamma, x : T \vdash \llbracket P \rrbracket_{(\text{supp}(\Gamma))} \otimes (x)$  if and only if  $\Delta; \Gamma, x : T \vdash \llbracket P \rrbracket_{(\text{supp}(\Gamma, x:T))}$ .

We can “read back” the typing rules over the term translation (to be made precise shortly), and thus also the properties of the type system, including Weakening and Strengthening by courtesy of Lemma 6. We read back typing rules as follows: the rules for the inactive process and input prefix are straightforward; restriction is type annotated in  $\text{sf}\pi$  using its premise; parallel composition is derived from tensor and composition; the case for output prefix has the twist that in  $\text{sf}\pi$  the typings used to type the two channels should be the same; and split the rule for names into a rule for names and one for subsumption. Recall that typings in typed  $\text{sf}\pi$  are not strong. Hence, we recover exactly the fragment of the well-known Pierce-Sangiorgi i/o-type system for the  $\pi$ -calculus [18,19]. Prop. 2 precisely relates the bigraphical type derivations with the ones for  $\text{sf}\pi$ .

**Proposition 2 (Transfer of Type Derivations)**  $\Gamma \vdash P : \diamond$  if and only if  $\Gamma_\emptyset; \Gamma \vdash \llbracket P \rrbracket_{(X)}$  when  $\text{fn}(P) \subseteq X = \text{supp}(\Gamma)$ .

*Proof (Sketch).* The proof is by struct. induction on  $P$  using Lemmas 6 and 3.

The proof is naturally by structural induction on  $P$  because we follow the translation of terms when transferring type derivations. We remark that to extend a BRS to accommodate a new source calculus operator one encodes it, and for a new process construct (e.g. a prefix) one adds an ion to the BRS, encodes the extended source calculus in the extended BRS and finally one gives a typing rule for this new ion. In conclusion: all of the bigraphical properties are transferable.

## 4 Conclusion

We have demonstrated a novel and uniform approach for developing type systems for (process) calculi, through bigraphs. Type systems are defined inductively over the structure of elementary bigraphs and their operators, as opposed to using a sorting [4]. Thus, a computer may possibly verify that a typed term is well-typed. Concretely, we have illustrated the approach by developing a sound i/o-type system enjoying a general form of Weakening and Strengthening for a bigraphical model of a core  $\pi$ -calculus, and we then transferred the type system and its properties to the  $\pi$ -calculus. The development of the i/o-type system for bigraphs differs significantly from i/o-typed  $\pi$ -calculus: bigraphs are contexts with richer structure than ordinary process calculus terms, which is reflected in the axioms governing bigraphical term equality, leading to technical intricacies in the Main Lemma used in Subject Reduction; Weakening and Strengthening of typed  $\pi$ -calculi corresponds to adding and removing idle names of bigraph terms, respectively.

We have tackled the case of i/o-types for the  $\pi$ -calculus because, being non-trivial and well-studied, this type system seemed to be an ideal test for our programme. In the future we would like to consider more sophisticated type systems. Here, some of the potential advantages of bigraphs (in particular, their modularity, the possibility of transferring the type results to a family of concrete calculi, and the insights gained on the type systems themselves) could be particularly valuable. A good example of this might be type systems for deadlock-freedom and lock-freedom, such as Kobayashi and co-workers' [11,10]. These type systems yield fundamental behavioural guarantees on processes such as absence of deadlock. However, one may argue that they are not fully understood yet, as a number of variations have appeared, with different expressive power. Also, they seem very sensitive to the grammar of the underlying process language, so transferring them to a different formalism may be troublesome. Formulating these types at the more abstract level of bigraphs could shed light into their design and facilitate their application.

We would like also to: consider different process languages, for instance with primitives for distribution such as Mobile Ambients or Homer; a deeper investigation of the relation between our work and sortings; generalise our approach to capture several interesting type systems simultaneously; to automatically derive an LTS for the BRS and then lift Subject Reduction to that semantics, to help bridge prior efforts in bigraphs concerning in expressiveness and derivation of LTSs with our approach; and support tools for type inference and type checking.

**Acknowledgments** The first author wishes to thank Mikkel N. Bundgaard, Søren Debois and Troels C. Damgaard for useful technical discussions. We thank the anonymous referees for suggestions on improving this paper’s presentation.

## References

1. Lars Birkedal, Søren Debois, Ebbe Elsborg, Thomas T. Hildebrandt, and Henning Niss. Bigraphical Models of Context-aware Systems. In *Proceedings of FoSSaCS’06*, volume 3921 of *LNCS*, pages 187–201. Springer, 2006.
2. Lars Birkedal, Søren Debois, and Thomas T. Hildebrandt. On the Construction of Sorted Reactive Systems. In *Proceedings of CONCUR’08*, LNCS, pages 218–232. Springer, 2008.
3. Mikkel N. Bundgaard and Thomas T. Hildebrandt. Bigraphical Semantics of Higher-Order Mobile Embedded Resources with Local Names. In *Proceedings of GT-VC’05*, volume 154 of *ENTCS*, pages 7–29. Elsevier, 2006.
4. Mikkel N. Bundgaard and Vladimiro Sassone. Typed polyadic pi-calculus in bi-graphs. In *Proceedings of PPDP’06*, pages 1–12. ACM Press, 2006.
5. Troels C. Damgaard and Lars Birkedal. Axiomatizing Binding Bigraphs. *Nordic Journal of Computing*, 13(1-2):58–77, 2006.
6. Atsushi Igarashi and Naoki Kobayashi. A generic type system for the Pi-calculus. *TCS*, 311(1-3):121–163, 2004.
7. Ole Høgh Jensen. *Mobile Processes in Bigraphs (Draft)*. PhD thesis, King’s College, University of Cambridge, 2007. Submitted.
8. Ole Høgh Jensen and Robin Milner. Bigraphs and Transitions. In *Proceedings of POPL’03*, pages 38–49. ACM Press, 2003.
9. Ole Høgh Jensen and Robin Milner. Bigraphs and mobile processes (revised). Technical Report UCAM-CL-TR-580, University of Cambridge, 2004.
10. Naoki Kobayashi. A type system for lock-free processes. *Inf. & Comp.*, 177:122–159, 2002.
11. Naoki Kobayashi. A new type system for deadlock-free processes. In *Proceedings of CONCUR’06*, volume 4137 of *LNCS*, pages 233–247. Springer, 2006.
12. Barbara König. A General Framework for Types in Graph Rewriting. *Acta Inf.*, 42(4):349–388, Dec. 2005. Special issue: Types in concurrency, Part II.
13. James J. Leifer and Robin Milner. Transition systems, link graphs, and Petri nets. *MSCS*, 16(6):989–1047, Dec. 2006.
14. Robin Milner. Bigraphs for Petri Nets. In *Lectures on Concurrency and Petri Nets: Advances in Petri Nets*, volume 3098 of *LNCS*, pages 686–701. Springer, 2004.
15. Robin Milner. Axioms for bigraphical structure. *MSCS*, 15(6):1005–1032, Dec. 2005.
16. Robin Milner. Pure Bigraphs: Structure and dynamics. *Inf. & Comp.*, 204(1), Jan. 2006.
17. Robin Milner. Local Bigraphs and Confluence: Two Conjectures. *ENTCS*, 175(3), June 2007.
18. Benjamin C. Pierce and Davide Sangiorgi. Typing and Subtyping for Mobile processes. *MSCS*, 6(5):409–453, 1996.
19. Davide Sangiorgi and David Walker. *The Pi-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.
20. Mark Weiser. Hot Topics – Ubiquitous Computing. *IEEE Computer*, 26(10):71–72, Oct. 1993.