

The **IT** University
of Copenhagen

Type Systems for Bigraphs

Ebbe Elsborg
Thomas T. Hildebrandt
Davide Sangiorgi

IT University Technical Report Series

TR-2008-110

ISSN 1600-6100

October 2008

**Copyright © 2008, Ebbe Elsborg
Thomas T. Hildebrandt
Davide Sangiorgi**

**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 9788779491847

Copies may be obtained by contacting:

**IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
Denmark**

Telephone: +45 72 18 50 00

Telefax: +45 72 18 50 01

Web: www.itu.dk

Type Systems for Bigraphs^{*}

Ebbe Elsberg¹, Thomas T. Hildebrandt¹, and Davide Sangiorgi²

¹ IT University of Copenhagen (ITU)

² 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 π -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 π -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) π -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 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], π -calculus [7,9,8], CCS [16], Mobile Ambients [7], Homer [3], and λ -calculus [17]. Importantly for our work, a sound and complete term language exists for bigraphs [15,5].

^{*}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).

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 π -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 π -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 π -calculus with capability types and subtyping in bigraphs by: defining and proving safe a *link sorting* – called 'sub-sorting' – 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 Γ 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 π -calculus enjoying subject reduction and type soundness. They express typings Γ as (abstract) CCS-like processes and then check the properties on Γ . 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 π -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. This technical report has the full proofs and omitted background definitions of the TGC'08 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 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 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, and a reaction rule

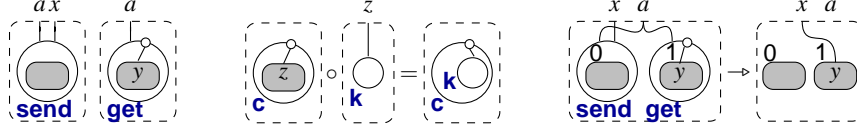


Figure 1. The ions send and get, bigraph composition, and the $\text{sfr}\pi$ reaction rule.

involving the ions. The two ions, depicted with solid circles, model output prefix and input prefix of π -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 π -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 (in order), 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 $b_1 = c_{(z)} : \langle 1, (\{z\}), \{z\} \rangle \rightarrow \langle 1, (\emptyset), \emptyset \rangle$ and $b_0 = (\{z\})K_z : \varepsilon \rightarrow \langle 1, (\{z\}), \{z\} \rangle$ then $b_1 \circ b_0 : \varepsilon \rightarrow \langle 1, (\emptyset), \emptyset \rangle$. 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 prod-*

$uct \mid$, 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 ε .

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 π -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

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

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

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

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 α -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 .

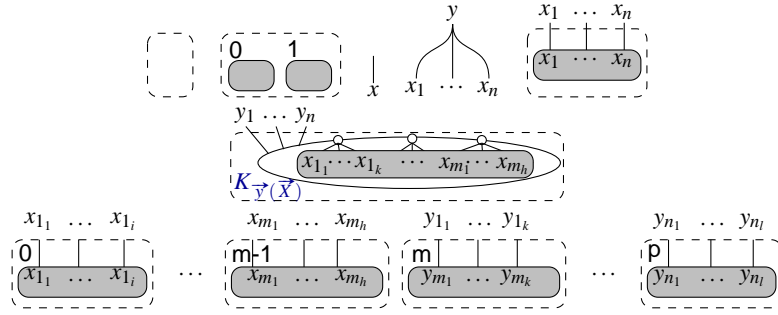


Figure 2. The seven elementary binding bigraphs, graphically.

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 $\{\vec{X}\} \stackrel{\text{def}}{=} \uplus_{i=1}^n X_i$.

$\mathbf{1} : 0 \rightarrow 1$	barren root
$\text{join} : 2 \rightarrow 1$	join two sites
$/x : x \rightarrow \mathbf{0}$	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.

Consider Fig. 2. First row: a barren root $\mathbf{1}$ 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 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 α -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 $\mathbb{B}\text{BG}_{\text{sf}\pi} \stackrel{\text{def}}{=} \mathbb{B}\text{BG}(\Sigma_{\text{sf}\pi}, \mathcal{R}_{\text{sf}\pi})$. 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 $b \circ b'$ well-defined, typically by “wiring through” (and then localising) outer names of b' that are not to be lexically bound by b .

Definition 3 ($\mathbb{B}\text{BG}_{\text{sf}\pi}$, [7])

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

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

$$\llbracket \bar{a}x.P \rrbracket_{(X)} = \text{send}_{ax} \triangleleft \text{id}_X \circ \llbracket P \rrbracket_{(X)} \quad \llbracket P \mid Q \rrbracket_{(X)} = \llbracket P \rrbracket_{(X)} \mid \llbracket Q \rrbracket_{(X)} \quad \llbracket \mathbf{0} \rrbracket_{(X)} = (X)$$

$$\llbracket a(y).P \rrbracket_{(X)} = \text{get}_{a(y)} \triangleleft \text{id}_X \circ \llbracket P \rrbracket_{(Xy)} \quad \llbracket \nu x.P \rrbracket_{(X)} = / (x) \triangleleft \text{id}_X \circ \llbracket P \rrbracket_{(Xx)}$$

The translation of P is indexed by a (local) name set $X \supseteq \text{fn}(P)$ that is needed to secure dynamic correspondence between $\text{sf}\pi$ and the model. This is because reduction in $\text{sf}\pi$ 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 $(\varepsilon, (X))$ of $\mathbb{B}BG_{\text{sf}\pi}$;*
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 $\mathbb{B}BG_{\text{sf}\pi}$.

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 Γ and Δ . 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 Γ . The extension of Γ with the assignment $x : T$ is denoted $\Gamma, x : T$ when $x \notin \text{supp}(\Gamma)$. The disjoint union Γ, Δ is defined when $\text{supp}(\Gamma) \cap \text{supp}(\Delta) = \emptyset$, and is (also) associative and commutative. Γ_\emptyset denotes the empty typing.*

Definition 6 (Syntax of types and typings)

$$T ::= V \mid L \quad V ::= L \mid \bullet \quad L ::= \#V \mid oV \mid iV \quad \Gamma ::= \Gamma, x : V \mid \Gamma, x : L \mid \Gamma_\emptyset .$$

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 iV , output oV , 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 *i/o*-typing rules for $\mathbb{B}BG_{sf\pi}$. The idea is to syntactically define types for elementary bigraphs and the operators on them following their structure inductively. The *i/o*-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 Δ of the inner names and one Γ 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) $\gamma_{I,\varepsilon} = \text{id}_I$ 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) $/y \circ y = \text{id}_\varepsilon$), and substitutions must allow subtyping, see below. We write γ_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 *i/o*-subtyping discipline we must be able to assign to y a subtype of the x_j so as to

<i>Placings</i> :	$\frac{}{\Gamma_0; \Gamma_0 \vdash \mathbf{1}}$	$\frac{}{\Gamma_0; \Gamma_0 \vdash \text{join}}$	$\frac{\Gamma_1 \leq \Gamma_0 \quad \text{supp}(\Gamma_j)^{j=0,1} = Z}{\Gamma_0; \Gamma_1 \vdash \gamma_Z}$	
<i>Linkings</i> :	$\frac{}{x : L; \Gamma_0 \vdash /x}$	$\frac{\Gamma \vdash y : T}{X : T; \Gamma \vdash y/x}$		
<i>Id & 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> :	$\frac{\Gamma \vdash a : \circ T \quad \Gamma' \vdash x : T}{\Gamma_0; \Gamma, \Gamma' \vdash \text{send}_{ax}}$	$\frac{\Gamma \vdash a : \text{i}S}{y : S; \Gamma \vdash \text{get}_{a(y)}}$		
<i>Subtyping</i> :	$\frac{}{T \leq T}$	$\frac{S \leq U \quad U \leq T}{S \leq T}$	$\frac{}{\#T \leq \text{i}T}$	$\frac{}{\#T \leq \circ T}$
	$\frac{S \leq T}{\text{i}S \leq \text{i}T}$	$\frac{T \leq S}{\circ S \leq \circ T}$	$\frac{T \leq S \quad S \leq T}{\#S \leq \#T}$	
<i>Names</i> :	$\frac{S \leq T}{x : S \vdash x : T}$			

Table 2. i/o-typing rules for $\mathbb{B}BG_{\text{sf}\pi}$.

allow substitution of names with a possibly smaller (i.e. more general) capability. In a sense this corresponds to the usual substitution lemma for π -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 π -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 substitu-

tion x/y 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 $x \in \text{glob}(\text{dom}(b))$ and *another* $x \in \text{glob}(\text{cod}(b))$ 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 π -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 Θ 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') :

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

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 $\Gamma_0; \Delta \vdash b_0$ and $b_0 \rightarrow b_1$ then $\Gamma_0; \Delta \vdash b_1$.*

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 $\Gamma_0; \Delta \vdash b_0$, then also $(*) \Gamma_0; \Delta \vdash D \circ r$ by Lemma 3. By Inversion we must have (among others) the following six subderivations from $(*)$: $(1')$ $\Gamma_0; \Gamma, y : S \vdash d$, $(3')$ $a : U \vdash a : \circ T$, $(3'')$ $x : U' \vdash x : T$, $(4')$ $a : R \vdash a : \circ S$, $(5')$ $U \leq R$, 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 $\Gamma_0; \Delta \vdash D \circ r'$. Reuse the derivation of D . $\rho = \text{id}_{(2, \emptyset, \{y\}, \emptyset)}$ so $\rho(d) = d$. 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 $\Gamma_0; \Delta \vdash D \circ r'$. \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 $\Gamma_0; \Gamma$ can only perform input or output actions for which Γ offers the appropriate capabilities. Let \rightarrow^* be the reflexive, transitive closure of \rightarrow .

Proposition 1 (Type Soundness) *Suppose that process bigraph $b = \llbracket P \rrbracket_{(x)}$, $\Gamma_0; \Gamma \vdash b$, 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 π -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 π -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 π -calculus 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 $\Gamma_0; \Gamma \vdash b$ 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 π -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 $\Gamma_0; \Gamma \vdash \llbracket P \rrbracket_{(X)}$ 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 π -calculus, and we then transferred the type system and its properties to the π -calculus. The development of the *i/o*-type system for bigraphs differs significantly from *i/o*-typed π -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 π -calculi corresponds to adding and removing idle names of bigraph terms, respectively.

We have tackled the case of *i/o*-types for the π -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 bigraphs. 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.

A Bigraphs

This appendix contains the relevant definitions of [9,16,7].

Definition 8 (pure signature) A (pure) signature \mathcal{K} is a set whose elements are called controls. For each control K it provides a finite ordinal $ar(K)$, an arity; it also determines which controls are atomic, and which of the non-atomic controls are active. Controls which are not active (including the atomic controls) are called passive.

Presuppose a countably infinite set χ of global names.

Definition 9 (concrete pure bigraph) A (concrete) pure bigraph over the signature \mathcal{K} takes the form $G = (V, E, ctrl, G^P, G^L) : I \rightarrow J$ where $I = \langle m, X \rangle$ and $J = \langle n, Y \rangle$ are its inner and outer faces, each combining a width (a finite ordinal) with a finite set of global names drawn from χ . Its first two components V and E are finite sets of nodes and edges respectively. The third component $ctrl : V \rightarrow \mathcal{K}$, a control map, assigns a control to each node. The remaining two are: $G^P = (V, ctrl, prnt) : m \rightarrow n$, $G^L = (V, E, ctrl, link) : X \rightarrow Y$.

Definition 10 (prime interface) An interface $I = \langle m, X \rangle$ consists of a finite ordinal m called a width, a finite set X called a name set. An interface is prime if it has width 1.

Definition 11 (prime bigraph) A prime bigraph $P : m \rightarrow \langle X \rangle$ has no inner names and a prime outer face.

Definition 12 (place graph) A place graph $A = (V, ctrl, prnt) : m \rightarrow n$ has an inner width m and an outer width n , both finite ordinals; a finite set V of nodes with a control map $ctrl : V \rightarrow \mathcal{K}$; and a parent map $prnt : m \uplus V \rightarrow V \uplus n$. The parent map is acyclic, i.e. $prnt^k(v) \neq v$ for all $k > 0$ and $v \in V$. An atomic node – i.e. one whose control is atomic – may not be a parent. We write $w >_A w'$, or just $w > w'$, to mean $w = prnt^k(w')$ for some $k > 0$.

The widths m and n index the sites and roots of A respectively. The sites and nodes – i.e. the domain of $prnt$ – are called places.

Definition 13 (precategory of place graphs) The precategory of place graphs \mathcal{PLG} has finite ordinals as objects and place graphs as arrows. The composition $A_1 \circ A_0 : m_0 \rightarrow m_2$ of two place graphs $A_i = (V_i, ctrl_i, prnt_i) : m_i \rightarrow m_{i+1}$ ($i = 0, 1$) is defined when the two node sets are disjoint; then $A_1 \circ A_0 \stackrel{\text{def}}{=} (V, ctrl, prnt)$ where $V = V_0 \uplus V_1$, $ctrl = ctrl_0 \uplus ctrl_1$, and $prnt = (\text{ld}_{V_0} \uplus prnt_1) \circ (prnt_0 \uplus \text{ld}_{V_1})$. The identity place graph at m is $\text{id}_m \stackrel{\text{def}}{=} (\emptyset, \emptyset_{\mathcal{K}}, \text{ld}_m) : m \rightarrow m$.

Definition 14 (barren, sibling, active, passive) A node or root is barren if it has no children. Two places are siblings if they have the same parent. A site s of A is active if $ctrl(v)$ is active whenever $v > s$; otherwise s is passive. If s is active (resp. passive) in A , we also say that A is active (resp. passive) at s .

Definition 15 (tensor product, PLG) The tensor product \otimes in PLG is defined as follows: On objects, we take $m \otimes n = m + n$. For two place graphs $A_i : m_i \rightarrow n_i$ ($i = 0, 1$) we take $A_0 \otimes A_1 : m_0 + m_1 \rightarrow n_0 + n_1$ to be defined when A_0 and A_1 have disjoint node sets; for the parent map, we first adjust the sites and roots of A_1 by adding them to m_0 and n_0 respectively, then take the union of the two parent maps.

Definition 16 (hard place graphs) A hard place graph is one in which no root or non-atomic node is barren. They form a sub-precateory denoted by PLG_h .

Definition 17 (link graph) A link graph $A = (V, E, \text{ctrl}, \text{link}) : X \rightarrow Y$ has finite sets X of inner names, Y of (outer) names, V of nodes and E of edges. It also has a function $\text{ctrl} : V \rightarrow \mathcal{K}$ called the control map, and a function $\text{link} : X \uplus P \rightarrow E \uplus Y$ called the link map, where $P \stackrel{\text{def}}{=} \sum_{v \in V} \text{ar}(\text{ctrl}(v))$ is the set of ports of A .

We shall call the inner names X and ports P the points of A , and the edges E and outer names Y its links.

Definition 18 (idle, open, closed, peer, lean) a link is idle if it has no preimage under the link map. An (outer) name is an open link, an edge is a closed link. A point (i.e. an inner name or port) is open if its link is open, otherwise closed. Two distinct points are peers if they are in the same link. A link graph is lean if it has no idle edges.

Definition 19 (precategory of link graphs) The precategory LIG has name sets as objects and link graphs as arrows. The composition $A_1 \circ A_0 : X_0 \rightarrow X_2$ of two link graphs $A_i = (V_i, E_i, \text{ctrl}_i, \text{link}_i) : X_i \rightarrow X_{i+1}$ ($i = 0, 1$) is defined when their node sets and edge sets are disjoint; then $A_1 \circ A_0 \stackrel{\text{def}}{=} (V, E, \text{ctrl}, \text{link})$ where $V = V_0 \uplus V_1$, $\text{ctrl} = \text{ctrl}_0 \uplus \text{ctrl}_1$, $E = E_0 \uplus E_1$ and $\text{link} = (\text{ld}_{E_0} \uplus \text{link}_1) \circ (\text{link}_0 \uplus \text{ld}_{P_1})$. The identity link graph at X is $\text{id}_X = (\emptyset, \emptyset, \emptyset_{\mathcal{K}}, \text{ld}_X) : X \rightarrow X$.

Definition 20 (tensor product, LIG) The tensor product \otimes in LIG is defined as follows: On objects, $X \otimes Y$ is simply the union of sets required to be disjoint. For two link graphs $A_i : X_i \rightarrow Y_i$ ($i = 0, 1$) we take $A_0 \otimes A_1 : X_0 \otimes X_1 \rightarrow Y_0 \otimes Y_1$ to be defined when the interface products are defined and when A_0 and A_1 have disjoint node sets and edge sets; then we take the union of their link maps.

Definition 21 (parallel product) The parallel product \parallel in LIG is defined as follows: On objects, $X \parallel Y \stackrel{\text{def}}{=} X \cup Y$. On link graphs $A_i : X_i \rightarrow Y_i$ ($i = 0, 1$) we define $A_0 \parallel A_1 : X_0 \otimes X_1 \rightarrow Y_0 \parallel Y_1$ whenever X_0 and X_1 are disjoint, by taking the union of link maps.

A place graph can be combined with a link graph iff they have the same node set and control map.

Definition 22 (precategory of pure concrete bigraphs) The precategory $\text{BIG}(\mathcal{K})$ of pure concrete bigraphs over a signature \mathcal{K} has pairs $I = \langle m, X \rangle$ as objects (interfaces) and bigraphs $G = (V, E, \text{ctrl}_G, G^P, G^L) : I \rightarrow J$ as arrows (contexts). We call I the inner face of G , and J the outer face. If $H : J \rightarrow K$ is another bigraph with node set

disjoint from V , then their composition is defined directly in terms of the compositions of the constituents as follows:

$$H \circ G \stackrel{\text{def}}{=} \langle H^P \circ G^P, H^L \circ G^L \rangle : I \rightarrow K.$$

The identities are $\langle \text{id}_m, \text{id}_X \rangle : I \rightarrow I$, where $I = \langle m, X \rangle$.

The subprecategory \mathbf{BIG}_h consists of hard bigraphs, those with place graphs in \mathbf{PLG}_h .

Definition 23 (tensor product, \mathbf{BIG}) The tensor product of two bigraph interfaces is defined by $\langle m, X \rangle \otimes \langle n, Y \rangle \stackrel{\text{def}}{=} \langle m+n, X \cup Y \rangle$ when X and Y are disjoint. The tensor product of two bigraphs $G_i : I_i \rightarrow J_i$ ($i = 0, 1$) is defined by

$$G_0 \otimes G_1 \stackrel{\text{def}}{=} \langle G_0^P \otimes G_1^P, G_0^L \otimes G_1^L \rangle : I_0 \otimes I_1 \rightarrow J_0 | J_1$$

when the interfaces exist and the node sets are disjoint. This combination is well-formed, since its constituents share the same node set.

Definition 24 (parallel product, \mathbf{BIG}) The parallel product of two bigraphs is defined on interfaces by $\langle m, X \rangle | \langle n, Y \rangle \stackrel{\text{def}}{=} \langle m+n, X \cup Y \rangle$, and on bigraphs by

$$G_0 | G_1 \stackrel{\text{def}}{=} \langle G_0^P \otimes G_1^P, G_0^L || G_1^L \rangle : I_0 \otimes I_1 \rightarrow J_0 | J_1$$

when the interfaces exist and the node sets are disjoint.

It is easy to verify that $||$ is associative, with unit ε .

Proposition 3 (alternative parallel product, \mathbf{BIG}) Let $G_0 || G_1$ be defined. Then

$$G_0 || G_1 = \sigma(G_0 \otimes \tau G_1),$$

where the substitutions σ and τ are defined as follows: If $z_i (i \in n)$ are the names shared between G_0 and G_1 , and w_i are fresh names in bijection with the z_i , then $\tau(z_i) = w_i$ and $\sigma(w_i) = \sigma(z_i) = z_i (i \in n)$.

Definition 25 (prime product, \mathbf{BIG}) The prime product of two interfaces is given by

$$\langle m, X \rangle | \langle n, Y \rangle \stackrel{\text{def}}{=} \langle 1, X \cup Y \rangle.$$

For two prime bigraphs $\vec{P} : \vec{I} \rightarrow \vec{J}$, if $I_0 \otimes I_1$ defined and n is the sum of the widths of J_0 and J_1 , we define their prime product by

$$P_0 | P_1 \stackrel{\text{def}}{=} \text{merge}_n \circ (P_0 || P_1) : I_0 \otimes I_1 \rightarrow J_0 | J_1.$$

Again $|$ is associative, with unit 1 when applied to primes. Refer to [9] for the definition of \simeq .

Definition 26 (instantiation) An instantiation ρ from (width) m to (width) n , which we write $\rho :: m \rightarrow n$, is determined by function $\bar{\rho} : n \rightarrow m$. For any X this function defines the map

$$\rho : \text{Gr}\langle m, X \rangle \rightarrow \text{Gr}\langle n, X \rangle$$

as follows. Decompose $g : \langle m, X \rangle$ into $g = w(d_0 \otimes \cdots \otimes d_{m-1})$, with $w : Y \rightarrow X$ and each d_i prime and discrete. Then define

$$\rho(g) \stackrel{\text{def}}{=} w(e_0 \parallel \cdots \parallel e_{n-1}),$$

where $e_j \simeq d_{\bar{\rho}(j)}$ for $j \in n$. This map is well-defined (up to support translation), by Propositions 9.16 and 9.17.

Note that the names of $e_0 \parallel \cdots \parallel e_{n-1}$ may be fewer than Y , because ρ may not be surjective. But by our convention the outer names of $\rho(g)$ are determined by the outer names of w , i.e. X .

Definition 27 (binding signature) A binding signature \mathcal{K} is like a pure signature, except that the arity of a control $K : h \rightarrow k$ now consists of a pair of finite ordinals: the binding arity h and the free arity k , determining the number of binding and non-binding ports of any K -node. If K is atomic then $h = 0$.

Definition 28 (binding interface) A binding interface $I = \langle m, \text{loc}, X \rangle$, where the width m is as before, X is a finite set of names, and $\text{loc} : X \rightarrow m \uplus \{\perp\}$ is a locality map associating some of the names X with a site in m . If $\text{loc}(x) = s \in m$ then x is located at s , or local (to s); If $\text{loc}(x) = \perp$ then x is global.

We call $I^U = \langle m, X \rangle$ the pure interface underlying I .

Definition 29 (binding bigraphs) A (concrete) binding bigraph $G : I \rightarrow J$ consists of an underlying pure bigraph $G^u : I^u \rightarrow J^u$ with extra structure as follows. Declare its binders to be the binding ports of its nodes together with the local names of its outer face J . Then G must satisfy the following:

SCOPE RULE: If p is a binder located at a node or root w , then every peer p' of p must be located at a place w' (a site or node) such that $w' <_{G^u} w$.

In the precategory $\mathbb{B}BG(\mathcal{K})$ of (concrete) binding bigraphs over \mathcal{K} , composition and identities are defined as for the underlying pure bigraphs; they are easily found to respect the scope rule. the forgetful functor

$$\mathcal{U} : \mathbb{B}BG(\mathcal{K}) \rightarrow \mathbb{B}IG(\mathcal{K})$$

sends each I to I^u and each G to G^u . The analogous definition holds also for hard binding bigraphs $\mathbb{B}BG_h(\mathcal{K})$.

Definition 30 (tensor product, \mathbf{BBG}) The tensor product of interfaces $I = \langle m, \vec{X}, X \rangle$ and $J = \langle n, \vec{Y}, Y \rangle$, where X and Y are disjoint, is

$$I \otimes J = \langle m + n, \vec{X}\vec{Y}, X \uplus Y \rangle.$$

The tensor product $G : I \rightarrow J$ of two binding $G_i : I_i \rightarrow J_i (i = 0, 1)$ with disjoint supports is defined when $I = I_0 \otimes I_1$ and $J = J_0 \otimes J_1$ are defined, and then $G^u = G_0^u \otimes G_1^u$. Thus \mathcal{U} preserves tensor product.

Definition 31 (parallel product, \mathbf{BBG}) Extending the previous definition, the parallel product of two interfaces $J_i = \langle n_i, \vec{X}_i, Y_i \rangle (i = 0, 1)$ keeps their local names disjoint but may share their global names:

$$J_0 \parallel J_1 \stackrel{\text{def}}{=} \langle n_0 + n_1, \vec{X}_0\vec{X}_1, Y_0 \cup Y_1 \rangle.$$

We define a parallel product on binding bigraphs by the equation $G_0 \parallel G_1 = \sigma(G_0 \otimes \tau G_1)$.

Definition 32 (prime product, \mathbf{BBG}) Extending the previous definition, the prime product of two prime interfaces is

$$\langle \langle X' \rangle, X \rangle \mid \langle \langle Y' \rangle, Y \rangle \stackrel{\text{def}}{=} \langle \langle X' \uplus Y' \rangle, X \cup Y \rangle.$$

The expression of the prime product of two prime binding bigraphs in terms of their parallel product is just as before.

Definition 33 (instantiation, \mathbf{BBG}) We replace instantiations $\rho :: m \rightarrow n$ for pure bigraphs by instantiations $\rho :: I \rightarrow J$ for binding bigraphs, where $I = \langle m, \vec{X} \rangle$ and $J = \langle n, \vec{Y} \rangle$ are local. The instantiation consists again of an underlying function $\bar{\rho} : n \rightarrow m$, and also provides bijective local substitutions $\rho_j : (X_{\bar{\rho}(j)}) \rightarrow (Y_j)$ for all $j \in n$. These ensure disjoint local names for each copy of a parameter factor. For any Z , this allows the map

$$\rho : \text{Gr}(I \otimes Z) \rightarrow \text{Gr}(J \otimes Z)$$

to be defined as follows (in terms of DNF as before): Decompose $g : I \otimes Z$ into $g = w(d_0 \otimes \cdots \otimes d_{m_1})$ with $w : W \rightarrow Z$ and each d_i prime and discrete. Then let $e_j \simeq \rho_j \circ d_{\bar{\rho}(j)}$ for each $j \in n$, and define

$$\rho(g) \stackrel{\text{def}}{=} w(e_0 \parallel \cdots \parallel e_{n-1}).$$

Definition 34 (bigraphical reactive system) A bigraphical reactive system (BRS) over signature \mathcal{K} consists of $\mathbf{BBG}(\mathcal{K})$ equipped with a set \mathbf{REACTS} of reaction rules closed under support equivalence (\simeq). We denote it by $\mathbf{BBG}(\mathcal{K}, \mathbf{REACTS})$.

Definition 35 (Insertion, [7]) Given a wiring $\omega : X \rightarrow Y$ and a local prime $A : X' \rightarrow Y'$ the insertion of ω into A is defined iff X and X' are disjoint. The result, written $A \triangleleft \omega : XX' \rightarrow Y \cup Y'$, has the nodes and parent map of A and its link map is the union of those of A and ω . Insertion binds tighter than prime product and composition.

Definition 36 (s-category) An s-category C is a strict symmetric monoidal precategory which has:

– for each arrow f , a finite set $|f|$ called its support, such that $|\text{id}_I| = \emptyset$. For $f : I \rightarrow J$ and $g : J \rightarrow K$ the composition $gf : I \rightarrow K$ is defined iff $|g| \cap |f| = \emptyset$ and $\text{dom}(g) = \text{cod}(f)$; then $|gf| = |g| \uplus |f|$. Similarly, for $f : H \rightarrow I$ and $g : J \rightarrow K$ with $H \otimes J$ and $I \otimes K$ defined, the tensor product $f \otimes g : H \otimes J \rightarrow I \otimes K$ is defined iff $|f| \cap |g| = \emptyset$; then $|f \otimes g| = |f| |g|$.

– for any arrow $f : I \rightarrow J$ and any injective map ρ whose domain includes $|f|$, an arrow $f : I \rightarrow J$ called a support translation of f such that

1. $\rho \cdot \text{id}_I = \text{id}_I$
2. $\rho \cdot (gf) = (\rho \cdot g)(\rho \cdot f)$
3. $\rho \cdot (f \otimes g) = \rho \cdot f \otimes \rho \cdot g$
4. $\text{Id}_{|f|} \cdot f = f$
5. $(\rho_1 \circ \rho_0) \cdot f = \rho_1 \cdot (\rho_0 \cdot f)$
6. $\rho \cdot f = (\rho \upharpoonright |f|) \cdot f$
7. $|\rho \upharpoonright |f|| = \rho(|f|)$.

Each equation is required to hold only when both sides are defined.

We continue on the next page with the axioms for binding bigraphs.

The axioms for binding bigraphs in Def. 37 are from [5], but with explicit composition. A, B, C, G range over bigraphs, H, I, J, K range over interfaces, ε is the empty interface $\langle 0, (), \emptyset \rangle$, x, y range over names, X, Y, Z range over name sets, $\ulcorner X \urcorner^Z \stackrel{\text{def}}{=} (Z) \ulcorner Z \uplus X \urcorner : \langle 1, (Z \uplus X), Z \uplus X \rangle \rightarrow \langle 1, (Z), Z \uplus X \rangle$, P ranges over primes, $K_{\vec{y}}(\vec{X})$ over ions, α ranges over renamings (multiple bijective substitutions), σ ranges over substitutions, and σ^{loc} ranges over local substitutions.

Definition 37 (Axioms for binding bigraphs)

Categorical axioms

- (C1) $A \circ \text{id}_I = A = \text{id}_J \circ A \quad (A : I \rightarrow J)$
- (C2) $A \circ (B \circ C) = (A \circ B) \circ C$
- (C3) $A \otimes \text{id}_\varepsilon = A = \text{id}_\varepsilon \otimes A$
- (C4) $A \otimes (B \otimes C) = (A \otimes B) \otimes C$
- (C5) $\text{id}_I \otimes \text{id}_J = \text{id}_{I \otimes J}$
- (C6) $(A_1 \otimes B_1) \circ (A_0 \otimes B_0) = (A_1 \circ A_0) \otimes (B_1 \circ B_0)$
- (C7) $\gamma_{I, \varepsilon} = \text{id}_I$
- (C8) $\gamma_{J, I} \circ \gamma_{I, J} = \text{id}_{I \otimes J}$
- (C9) $\gamma_{I, K} \circ (A \otimes B) = (B \otimes A) \circ \gamma_{H, J} \quad (A : H \rightarrow I, B : J \rightarrow K)$
- (C10) $\gamma_{I \otimes J, K} \circ (A \otimes B) = (\gamma_{I, K} \otimes \text{id}_J) \circ (\text{id}_I \otimes \gamma_{J, K})$

Link axioms

- (L1) $x/x = \text{id}_x$
- (L2) $/y \circ y/x = /x$
- (L3) $/y \circ y = \text{id}_\varepsilon$
- (L4) $(z/(Y \uplus y)) \circ (\text{id}_Y \otimes y/x) = z/(Y \uplus X)$

Place axioms

- (P1) $\text{join} \circ (\mathbf{1} \otimes \text{id}_1) = \text{id}_1$
- (P2) $\text{join} \circ (\text{join} \otimes \text{id}_1) = \text{join} \circ (\text{id}_1 \otimes \text{join})$
- (P3) $\text{join} \circ \gamma_{1, 1, (\emptyset, \emptyset)} = \text{join}$

Binding axioms

- (B1) $(\emptyset)P = P$
- (B2) $(Y) \ulcorner Y \urcorner = \text{id}_{(Y)}$
- (B3) $(\ulcorner X \urcorner^Z \otimes \text{id}_Y)(X)P = P \quad (P : I \rightarrow \langle 1, (Z), Z \uplus X \uplus Y \rangle)$
- (B4) $((Y)P) \otimes \text{id}_X G = (Y)(P \otimes \text{id}_X)G$
- (B5) $(X \uplus Y)P = (X)((Y)P)$

Ion axioms

- (N1) $(\text{id}_1 \otimes \alpha) \circ K_{\vec{y}}(\vec{X}) = K_{\alpha(\vec{y})}(\vec{X})$
- (N2) $K_{\vec{y}}(\vec{X}) \circ \sigma^{\text{loc}} = K_{\vec{y}((\sigma^{\text{loc}})^{-1}(\vec{X}))}$

B π -calculus

This appendix contains standard π -calculus definitions and an i/o-type system, for easy reference.

Definition 38 (Binding) In each of $a(x).P$ and νxP , the displayed occurrence of x is binding with scope P . An occurrence of a name in a process is bound if it is, or it lies within the scope of, a binding occurrence of the name. An occurrence of a name in a process is free if it is not bound.

Definition 39 (Substitution) A substitution is a function on names that is the identity except on a finite set.

Notation 2 (Substitution on names) Use σ to range over substitutions, and write σx for σ applied to x . The support of σ , $\text{supp}(\sigma)$, is $\{x \mid \sigma x \neq x\}$, and the co-support of σ , $\text{cosupp}(\sigma)$, is $\{\sigma x \mid x \in \text{supp}(\sigma)\}$. Write $\text{n}(\sigma)$ for the set of names of σ , which is $\text{supp}(\sigma) \cup \text{cosupp}(\sigma)$. Write $\{y_1, \dots, y_n / x_1, \dots, x_n\}$ for the substitution σ such that $\sigma x_i = y_i$ for each $i \in \{1, \dots, n\}$ and $\sigma x = x$ for $x \notin \{x_1, \dots, x_n\}$. If X is a set of names, write σX for $\{\sigma x \mid x \in X\}$.

Definition 40 (α -convertibility)

1. If the name x does not occur in the process P , then $\{x/y\}P$ is the process obtained by replacing each free occurrence of y in P by x .
2. A change of bound names in a process P is the replacement of a subterm $a(x).Q$ of P by $a(y).\{y/x\}Q$, or the replacement of a subterm νxQ of P by $\nu y\{y/x\}Q$, where in each case y does not occur in Q .
3. Processes P and Q are α -convertible, $P =_\alpha Q$, if Q can be obtained from P by a finite number of changes of bound names.

Convention 1 When considering a collection of processes and substitutions, it is assumed that the bound names of the processes are chosen to be different from their free names and from the names of the substitutions.

Definition 41 (Substitution on prefixes) The effect of applying a substitution σ to a prefix π is to replace each occurrence of each name x in π by σx .

Definition 42 (Substitution on processes) The process σP , obtained by applying σ to P is defined as follows, avoiding capture of names by binders:

$$\begin{aligned} \sigma(\pi.P) &\stackrel{\text{def}}{=} \sigma\pi.\sigma P \\ \sigma(P \mid Q) &\stackrel{\text{def}}{=} \sigma P \mid \sigma Q \\ \sigma(\nu xP) &\stackrel{\text{def}}{=} \nu x(\sigma P) \\ \sigma \mathbf{0} &\stackrel{\text{def}}{=} \mathbf{0} . \end{aligned}$$

Notation 3 (Operator precedence) When writing processes as linear expressions parentheses are used to resolve ambiguity, and observe the conventions that prefixing and restriction bind more tightly than parallel composition. Further, substitutions bind more tightly than process operators. Sometimes parentheses are inserted merely to aid reading.

Definition 43 (Process context) A process context is a process term in which exactly one process subterm has been left out leaving a “hole” represented with notation $[\cdot]$. For a context C write $C[P]$ for the process resulting from “plugging” the process P into the hole of C , where the hole in C must occur in a position such that $C[P]$ is well-formed for an arbitrary process term P .

Definition 44 (Process congruence) An equivalence relation \mathcal{R} on processes is a process congruence if $(P, Q) \in \mathcal{R}$ implies $(C[P], C[Q]) \in \mathcal{R}$ for every process context C .

$$\begin{array}{l}
\text{Processes : } \frac{}{\Gamma \vdash \mathbf{0} : \diamond} \quad \frac{\Gamma, x : L \vdash P : \diamond}{\Gamma \vdash (vx : L)P : \diamond} \quad \frac{\Gamma \vdash P : \diamond \quad \Gamma \vdash Q : \diamond}{\Gamma \vdash P | Q : \diamond} \\
\frac{\Gamma \vdash a : iS \quad \Gamma, y : S \vdash P : \diamond}{\Gamma \vdash a(y).P : \diamond} \quad \frac{\Gamma \vdash a : oT \quad \Gamma \vdash x : T \quad \Gamma \vdash P : \diamond}{\Gamma \vdash \bar{a}x.P : \diamond} \\
\\
\text{Subtyping : } \frac{}{T \leq T} \quad \frac{S \leq S' \quad S' \leq T}{S \leq T} \quad \frac{}{\#T \leq iT} \quad \frac{}{\#T \leq oT} \\
\frac{S \leq T}{iS \leq iT} \quad \frac{T \leq S}{oS \leq oT} \quad \frac{T \leq S \quad S \leq T}{\#S \leq \#T} \\
\\
\text{Names : } \frac{}{\Gamma, x : T \vdash x : T} \quad \frac{\Gamma \vdash x : S \quad S \leq T}{\Gamma \vdash x : T}
\end{array}$$

Table 3. i/o-type rules for sfr .

C Full proofs

This appendix contains the full proofs.

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

Proof. The proof is by induction on the height of the derivation of $\Delta; \Gamma, x : T \vdash b$.

- The cases for **1**, *join*, and $/x$ are vacuously true.
- The cases for transpositions, identities, and concretions hold by Inversion and transitivity of the subtyping relation \leq .
- The case for substitutions: Assume a derivation of $X : T'; \Gamma, x : T \vdash y/x$ with premise $\Gamma, x : T \vdash y : T'$ by Inversion. Because typings are strong we must have $\Gamma = \Gamma_0$ and $y = x$. Thus, we have a derivation of $y : T \vdash y : T'$. Clearly, $T \leq T'$. By transitivity of subtyping and the assumption $S \leq T$ we obtain $S \leq T'$. Hence, by the rule for names (subsumption) we can derive $y : S \vdash y : T'$, which is required to derive $X : T'; \Gamma, x : S \vdash y/x$.
- The case for abstraction follows immediately from Inversion and the induction hypothesis.
- The case for output, send_{az} : Either $x = a$ or $x = z$ but not both. Case $x = a$: Assume a derivation of $\Gamma_0; a : T, \Gamma \vdash \text{send}_{az}$ with premises (1) $a : T \vdash a : \circ T'$ and (2) $\Gamma \vdash x : T'$ by Inversion. From (1) we know that $T \leq \circ T'$. Then, the assumption $S \leq T$ and transitivity of subtyping yield $S \leq \circ T'$. Then, by subsumption we derive (1') $a : S \vdash a : \circ T'$. Using (1') and (2) we derive $\Gamma_0; a : S, \Gamma' \vdash \text{send}_{az}$ as required. The case for $x = z$ is analogous to the case where $x = a$.
- The case for input: The proof is analogous to the proof for output where $x = a$.
- The case for tensor product: Assume a derivation of $\Delta_0, \Delta_1; \Gamma_0, \Gamma_1 \vdash b_0 \otimes b_1$ with premises $\Delta_0; \Gamma_0 \vdash b_0$ and $\Delta_1; \Gamma_1 \vdash b_1$ by Inversion. Either $x \in \text{supp}(\Gamma_0)$ or $x \in \text{supp}(\Gamma_1)$ but not both. In either case the *deciderata* follows from the induction hypothesis on $\Delta_i; \Gamma_i \vdash b_i$.
- The case for composition: Assume a derivation of $\Gamma_0; \Gamma_1 \vdash b_1 \circ b_0$ with premises (1) $\Gamma_0; \Delta \vdash b_0$ and (2) $\Delta; \Gamma_1 \vdash b_1$ by Inversion. We have that $x \in \text{supp}(\Gamma_1)$ so the *deciderata* follows by induction hypothesis on (2). \square

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

Proof. The proof is by induction on the height of the derivation of $\Delta, x : S; \Gamma \vdash b$.

- The cases for **1**, *join*, and output hold vacuously.
- The cases for transpositions, identities, and concretions hold by transitivity of subtyping.
- The case for closure is by axiom.
- The case for substitutions: Assume a derivation of $X : S; \Gamma \vdash y/x$ with premise $\Gamma \vdash y : S$ by Inversion. Clearly, $\Gamma(y) \leq S$, and because $S \leq T$ we obtain $\Gamma(y) \leq T$ by transitivity of subtyping, which derives $X : T; \Gamma \vdash y/x$ by subsumption. Hence, we can derive $X : T; \Gamma \vdash y/x$ as required.

- The cases for abstraction, tensor, and composition are by Inversion and then one application of the induction hypothesis.
- The case for input: Assume a derivation of $y : S; \Gamma \vdash \text{get}_{a(y)}$ with premise $\Gamma \vdash a : iS$ by Inversion. $S \leq T$ derives $iS \leq iT$ by covariance of subtyping on input types. Clearly, $\Gamma(a) \leq iS$, and because $iS \leq iT$ we obtain $\Gamma(a) \leq iT$ by transitivity of subtyping. We can thus derive $\Gamma \vdash a : iT$ and then conclude $y : T; \Gamma \vdash \text{get}_{a(y)}$ as required. \square

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

Proof. The proof is by induction on the height of the derivation of $b_0 = b_1$ and has a case for each axiom. The proof consists of 60 cases; there is one case for each direction of each axiom, casing on whether the ion is send or get, and checks for reflexivity, symmetry, transitivity, and congruence. Inversion is used frequently in a straightforward manner so we omit explicit mention of it. We proceed by case analysis.

- case $A \circ \text{id}_I = A$ for $A : I \rightarrow J$.
 - “ \Rightarrow ” : Assume a derivation of $\Delta; \Gamma \vdash A \circ \text{id}_I$ with premises (1) $\Delta; \Theta \vdash \text{id}_I$ and (2) $\Theta; \Gamma \vdash A$. From (1) we know that $\Theta \leq \Delta$ so by Widening on (2) we obtain $\Delta; \Gamma \vdash A$ as required.
 - “ \Leftarrow ” : Assume a derivation of (1) $\Delta; \Gamma \vdash A$. Clearly, we may directly derive (2) $\Delta; \Delta \vdash \text{id}_I$ by the rule for identities. Now we can build the required derivation of $\Delta; \Gamma \vdash A \circ \text{id}_I$ by the rule for composition using (2) and (1).
- case $A = \text{id}_J \circ A$ for $A : I \rightarrow J$.
 - “ \Rightarrow ” : Assume a derivation of (1) $\Delta; \Gamma \vdash A$. Clearly, we may directly derive (2) $\Gamma; \Gamma \vdash \text{id}_J$ by the rule for identities. Now we can build the required derivation of $\Delta; \Gamma \vdash \text{id}_J \circ A$ by the rule for composition using (1) and (2).
 - “ \Leftarrow ” : Assume a derivation of $\Delta; \Gamma \vdash \text{id}_J \circ A$ with premises (1) $\Delta; \Theta \vdash A$ and (2) $\Theta; \Gamma \vdash \text{id}_J$. From (2) we know that $\Gamma \leq \Theta$ so by Narrowing on (1) we obtain $\Delta; \Gamma \vdash A$.
- case $A \circ (B \circ C) = (A \circ B) \circ C$.
 - “ \Leftrightarrow ” : Clearly, because exactly the same subderivations are needed in both derivations, and the disjoint union on typings is associative.
- case $A \otimes \text{id}_E = A$.
 - “ \Rightarrow ” : Reuse the subderivation $\Gamma; \Delta \vdash A$.
 - “ \Leftarrow ” : Reuse the subderivation $\Gamma; \Delta \vdash A$, and $\Gamma_0; \Gamma_0 \vdash \text{id}_E$ is by axiom.
- case $A = \text{id}_E \otimes A$.
 - Analogous to the previous case.
- case $A \otimes (B \otimes C) = (A \otimes B) \otimes C$.
 - Clearly, because exactly the same subderivations are needed in both derivations, and the disjoint union on typings is associative.
- case $\text{id}_I \otimes \text{id}_J = \text{id}_{I \otimes J}$.
 - “ \Rightarrow ” : Assume a derivation of $\Delta_0, \Delta_1; \Gamma_0, \Gamma_1 \vdash \text{id}_I \otimes \text{id}_J$ with premises (1) $\Delta_0; \Gamma_0 \vdash \text{id}_I$ and (2) $\Delta_1; \Gamma_1 \vdash \text{id}_J$, where $\text{supp}(\Delta_0) = \text{supp}(\Gamma_0) = \text{glob}(I)$ and

$\Gamma_0 \leq \Delta_0$, and $\text{supp}(\Delta_1) = \text{supp}(\Gamma_1) = \text{glob}(J)$ and $\Gamma_1 \leq \Delta_1$. We need to establish (A) $\text{supp}(\Delta_0, \Delta_1) = \text{supp}(\Gamma_0, \Gamma_1) = \text{glob}(I \otimes J)$ and (B) $\Gamma_0, \Gamma_1 \leq \Delta_0, \Delta_1$. We know that $\text{supp}(\Delta_0) = \text{glob}(I)$ and $\text{supp}(\Delta_1) = \text{glob}(J)$ so we obtain that $\text{supp}(\Delta_0, \Delta_1) = \text{glob}(I \otimes J)$. Likewise for Γ_0 and Γ_1 w.r.t J so (A) is established. We know that $\Gamma_k \leq \Delta_k$ (for $k = 0, 1$) so clearly $\Gamma_0, \Gamma_1 \leq \Delta_0, \Delta_1$, which establishes (B). We can now derive $\Delta_0, \Delta_1; \Gamma_0, \Gamma_1 \vdash \text{id}_{I \otimes J}$ as required.

“ \Leftarrow ”: Assume a derivation of $\Delta; \Gamma \vdash \text{id}_{I \otimes J}$ with premises (1) $\text{supp}(\Delta) = \text{glob}(I \otimes J)$, (2) $\text{supp}(\Gamma) = \text{glob}(I \otimes J)$, and (3) $\Gamma \leq \Delta$ by Inversion. Clearly, Δ and Γ can be split into parts Δ_0, Δ_1 and Γ_0, Γ_1 such that $\text{supp}(\Theta_0) = \text{glob}(I)$ and $\text{supp}(\Theta_1) = \text{glob}(J)$ (for $\Theta \in \{\Delta, \Gamma\}$). It is also obvious that $\Gamma_0 \leq \Delta_0$ and $\Gamma_1 \leq \Delta_1$. Hence, we can derive $\Delta_0; \Gamma_0 \vdash \text{id}_I$ and $\Delta_1; \Gamma_1 \vdash \text{id}_J$, and thus also $\Delta; \Gamma \vdash \text{id}_I \otimes \text{id}_J$ as required.

– case $(A_1 \otimes B_1) \circ (A_0 \circ B_0) = (A_1 \circ A_0) \otimes (B_1 \circ B_0)$.

“ \Leftrightarrow ”: Clearly, because exactly the same subderivations are needed in both derivations, albeit slightly rearranged.

– case $\gamma_{I, \varepsilon} = \text{id}_I$.

Recall that $\gamma_{I, \varepsilon} \stackrel{\text{def}}{=} \gamma_{m, 0, (\vec{X}_B, ())} \otimes \text{id}_{X_F} \otimes \text{id}_0 \stackrel{\text{def}}{=} \gamma_{m, 0, (\vec{X}_B, ())} \otimes \text{id}_{X_F}$ for

$I = \langle m, \vec{X}_B, \{\vec{X}_B\} \uplus X_F \rangle$ w.l.o.g.

“ \Rightarrow ”: Assume a derivation of $\Delta_0, \Delta_1; \Gamma_0, \Gamma_1 \vdash \gamma_{m, 0, (\vec{X}_B, ())} \otimes \text{id}_{X_F}$ with premises $\Delta_0; \Gamma_0 \vdash \gamma_{m, 0, (\vec{X}_B, ())}$ and $\Delta_1; \Gamma_1 \vdash \text{id}_{X_F}$. From these premises we clearly have $\text{supp}(\Delta_0, \Delta_1) = \text{supp}(\Gamma_0, \Gamma_1) = \{\vec{X}_B\} \uplus X_F = \text{glob}(I)$, and also $(\Gamma_0, \Gamma_1)(i) \leq (\Delta_0, \Delta_1)(i)$ for any $i \in \text{glob}(I)$. Thus, a derivation of $\Delta_0, \Delta_1; \Gamma_0, \Gamma_1 \vdash \text{id}_I$ can be built.

“ \Leftarrow ”: Assume a derivation of $\Gamma_0; \Gamma_1 \vdash \text{id}_I$ with premises $\text{supp}(\Gamma_j) = \text{glob}(I)$ (for $j = 0, 1$) and $\Gamma_1 \leq \Gamma_0$. $\text{glob}(I) = \{\vec{X}_B\} \uplus X_F$ by assumption. Clearly, Γ_j can be split into typings Γ_j for $\{\vec{X}_B\}$ and Γ'_j for X_F . Therefore, the rule for tensor can be used to build the required derivation of $\Gamma_0, \Gamma'_0; \Gamma_1, \Gamma'_1 \vdash \gamma_{m, 0, (\vec{X}_B, ())} \otimes \text{id}_{X_F}$.

– $\gamma_{J, I} \circ \gamma_{I, J} = \text{id}_{I \otimes J}$.

Recall that $\gamma_{J, I} \stackrel{\text{def}}{=} \gamma_{n, m, (\vec{Z}_B, \vec{X}_B)} \otimes \text{id}_{Z_F} \otimes \text{id}_{X_F}$ and $\gamma_{I, J} \stackrel{\text{def}}{=} \gamma_{m, n, (\vec{X}_B, \vec{Z}_B)} \otimes \text{id}_{X_F} \otimes \text{id}_{Z_F}$ for $I = \langle m, \vec{X}_B, \{\vec{X}_B\} \uplus X_F \rangle$ and $J = \langle n, \vec{Z}_B, \{\vec{Z}_B\} \uplus Z_F \rangle$ w.l.o.g.

“ \Rightarrow ”: Clearly, $\gamma_{J, I} \circ \gamma_{I, J}$ is typable in some Γ_j (for $j = 0, 1$) assigning types to exactly the names of I and J , disjointly, with $\Gamma_1 \leq \Gamma_0$. Thus, a derivation of $\Gamma_0; \Gamma_1 \vdash \text{id}_{I \otimes J}$ can be built.

“ \Leftarrow ”: Reverse the argument.

– case $\gamma_{I, J} \circ (A \otimes B) = (B \otimes A) \circ \gamma_{H, J}$ for $A : H \rightarrow I, B : J \rightarrow K$.

“ \Rightarrow ”: Assume a derivation of $\Gamma_0; \Gamma_1 \vdash \gamma_{I, J} \circ (A \otimes B)$, where $\text{supp}(\Gamma_0) = \text{glob}(H) \uplus \text{glob}(J)$ and $\text{supp}(\Gamma_1) = \text{glob}(I) \uplus \text{glob}(K)$. Exactly the same subderivations are needed to build a derivation of $(B \otimes A) \circ \gamma_{H, J}$, albeit slightly rearranged.

“ \Leftarrow ”: The argument is analogous.

– case $\gamma_{I \otimes J, K} = (\gamma_{I, K} \otimes \text{id}_J) \circ (\text{id}_I \otimes \gamma_{J, K})$.

Essentially, this case holds because exactly the same names are typed on both sides, albeit somewhat rearranged.

Recall that $\gamma_{I \otimes J, K} \stackrel{\text{def}}{=} \gamma_{(I \otimes J)_B, K_B} \otimes \text{id}_{(I \otimes J)_F} \otimes \text{id}_{K_F}$, where the subscripts B and F signify the set of bound/local names and the set of free names of an interface, respectively.

For interfaces I , let I_F denote the free (i.e. $\text{glob}(I) \setminus \text{loc}(I)$) and I_B the bound (i.e. $\text{loc}(I)$) names.

“ \Rightarrow ”: Assume a derivation of $\Gamma; \Gamma' \vdash \gamma_{(I \otimes J)_B, K_B} \otimes \text{id}_{(I \otimes J)_F} \otimes \text{id}_{K_F}$ with subderivations (1) $\Gamma_1; \Gamma'_1 \vdash \gamma_{(I \otimes J)_B, K_B}$, (2) $\Gamma_2; \Gamma'_2 \vdash \text{id}_{(I \otimes J)_B}$, and (3) $\Gamma_3; \Gamma'_3 \vdash \text{id}_{K_F}$, where $\Gamma = \Gamma_1, \Gamma_2, \Gamma_3$ and $\Gamma' = \Gamma'_1, \Gamma'_2, \Gamma'_3$, $\text{supp}(\Gamma_1) = (I \otimes J)_B \uplus K_B$, $\text{supp}(\Gamma_2) = (I \otimes J)_F$, and $\text{supp}(\Gamma_3) = K_F$.

Recall that $\gamma_{J, K} \stackrel{\text{def}}{=} \gamma_{J_B, K_B} \otimes \text{id}_{J_F} \otimes \text{id}_{K_F}$ and $\gamma_{I, K} \stackrel{\text{def}}{=} \gamma_{I_B, K_B} \otimes \text{id}_{I_F} \otimes \text{id}_{K_F}$.

We need to build a derivation of $\Gamma; \Gamma' \vdash (\gamma_{I, K} \otimes \text{id}_J) \circ (\text{id}_I \otimes \gamma_{J, K})$. This requires two subderivations: (A) $\Gamma; \Gamma'' \vdash \text{id}_I \otimes \gamma_{J, K}$ and (B) $\Gamma; \Gamma'' \vdash \gamma_{I, K} \otimes \text{id}_J$. Both (A) and (B) require two subderivations. (A1) $\Gamma_I; \Gamma''_I \vdash \text{id}_I$ and (A2) $\Gamma_{J \otimes K}; \Gamma''_{J \otimes K} \vdash \gamma_{J, K}$, where Γ_I denotes $\Gamma \upharpoonright \text{glob}(I)$ for any typing Γ and interface I . The case for (B) is analogous. So, we must find a suiting Γ'' . Pick $\Gamma'' = \Gamma''_I, \Gamma''_{J \otimes K} = \Gamma_I, \Gamma_{J \otimes K} = ((\Gamma_1 \upharpoonright I_B), \Gamma_2 \upharpoonright I_F), ((\Gamma_1 \upharpoonright (J_B \uplus J_K)), (\Gamma_2 \upharpoonright J_F), \Gamma_3)$. All that remains is to check (i) $\Gamma''_I \leq \Gamma_I$ and $\text{supp}(\Gamma_I) = \text{supp}(\Gamma''_I) = \text{glob}(I)$, (ii) $\Gamma''_{J \otimes K} \leq \Gamma_{J \otimes K}$ and $\text{supp}(\Gamma_{J \otimes K}) = \text{supp}(\Gamma''_{J \otimes K}) = \text{glob}(J) \uplus \text{glob}(K)$, and (iii) $\Gamma_I, \Gamma_{J \otimes K} = \Gamma$ and $\Gamma''_I, \Gamma''_{J \otimes K} = \Gamma''$. (i) follows from the fact that $(\Gamma_1 \upharpoonright I_B), \Gamma_2 \upharpoonright I_F = \Gamma_I = \Gamma''_I$. (ii) follows from the fact that $(\Gamma_1 \upharpoonright (J_B \uplus K_B)), (\Gamma_2 \upharpoonright J_F), \Gamma_3 = \Gamma_{J \otimes K} = \Gamma''_{J \otimes K}$. (iii) follows from our choice of $\Gamma'' = \Gamma$.

“ \Leftarrow ”: Obviously, we can make the same appropriate splits of typings.

– case $x/x = \text{id}_x$.

“ \Rightarrow ”: Assume a derivation of $x : T; \Gamma \vdash x/x$ with premise (*) $\Gamma \vdash x : T$. (*) implies (1) $\text{supp}(\Gamma) = \{x\}$ and (2) $\Gamma(x) \leq T$. Also, clearly (3) $\text{supp}(x : T) = \{x\}$. Thus, we can from (1), (2), and (3) build the desired derivation of $x : T; \Gamma \vdash \text{id}_x$.

“ \Leftarrow ”: Assume a derivation of $\Gamma_0; \Gamma_1 \vdash \text{id}_x$ with premises $\text{supp}(\Gamma_i) = \{x\}$ (for $i = 0, 1$) and (*) $\Gamma_1(x) \leq \Gamma_0(x)$. (*) implies that $\Gamma_1 \vdash x : \Gamma_0(x)$, which allows us to conclude $\Gamma_0; \Gamma_1 \vdash x/x$ as required.

– case $/y \circ y/x = /x$.

“ \Rightarrow ”: Assume a derivation of $x : L; \Gamma_0 \vdash /y \circ y/x$ with premises $x : L; y : L' \vdash y/x$ and $y : L'; \Gamma_0 \vdash /y$, for some link types L and L' such that $L' \leq L$. By axiom, $x : L; \Gamma_0 \vdash /x$.

“ \Leftarrow ”: Assume a derivation of $x : L; \Gamma_0 \vdash /x$. Build derivations of $x : L; y : L \vdash y/x$ and $y : L; \Gamma_0 \vdash /y$ by axioms. From these two subderivations we construct a derivation of $x : L; \Gamma_0 \vdash /y \circ y/x$.

– case $/y \circ y = \text{id}_e$.

Recall that y is shorthand for y/\emptyset .

“ \Rightarrow ”: Assume a derivation of $\Gamma_0; \Gamma_0 \vdash /y \circ y$ with premises $\Gamma_0; y : L \vdash y/\emptyset$ and $y : L; \Gamma_0 \vdash /y$. As required, $\Gamma_0; \Gamma_0 \vdash \text{id}_e$ is trivially derivable.

“ \Leftarrow ”: Assume a derivation of $\Gamma_0; \Gamma_0 \vdash \text{id}_e$. To build a derivation of $\Gamma_0; \Gamma_0 \vdash /y \circ y$ two subderivations are needed: $\Gamma_0; y/\emptyset \vdash y : L$ and $y : L; \Gamma_0 \vdash /y$, for some link type L . They are both trivially derivable.

– case $z/(Y \uplus \{y\}) \circ (\text{id}_Y \otimes y/x) = z/(Y \uplus X)$.

“ \Rightarrow ”: Assume a derivation of $\Gamma_0, X : T; \Gamma^z \vdash z/(Y \uplus \{y\}) \circ (\text{id}_Y \otimes y/x)$ with three subderivations: (1) $\Gamma_0; \Gamma_1 \vdash \text{id}_Y$, (2) $X : T; \Gamma^y \vdash y/x$, and (3) $\Gamma_1, \Gamma^y; \Gamma^z \vdash z/(Y \uplus \{y\})$, with the following premises: (1A) $\text{supp}(\Gamma_j) = Y$ (for $j = 0, 1$), (1B) $\Gamma_1 \leq \Gamma_0$, (2A) $\Gamma^y \vdash y : T$, and (3A) $\Gamma^z \vdash z : (\Gamma_1, \Gamma^y)(Y \uplus \{y\})$.

We need to establish $(*) \Gamma^z \vdash z : (\Gamma_0, X : T)(Y \uplus X)$ to derive the required conclusion $\Gamma_0, X : T; \Gamma^z \vdash z/(Y \uplus X)$. Establishing $(*)$ is obviously equivalent to showing $(*1) \Gamma^z \vdash z : \Gamma_0(Y)$ and $(*2) \Gamma^z \vdash z : T$, because $(X : T)(X) = T$.

From (1A) we know that $\text{supp}(\Gamma_0) = Y$ so $\text{supp}(\Gamma_0, X : T) = Y \uplus X$. From (3A) we have $\Gamma^z \vdash z : \Gamma_1(Y)$, in particular, so $\Gamma^z(z) \leq \Gamma_1(Y) \leq \Gamma_0(Y)$, by (1B). Thus, by transitivity of subtyping $\Gamma^z(z) \leq \Gamma_0(Y)$ and we can derive $(*1)$ by subsumption.

For $(*2)$ we must establish that $\Gamma^z(z) \leq T$. From (3A) we know that $\Gamma^z \vdash z : \Gamma^y(y)$ so $\Gamma^z(z) \leq \Gamma^y(y)$. From (2A) we know that $\Gamma^y(y) \leq T$. Combining these two facts and transitivity of subtyping we obtain $\Gamma^z(z) \leq T$. Hence, we can derive $(*2)$ by subsumption.

Finally, we derive the *deciderata* by $(*1)$ and $(*2)$ and the rule for substitutions.

“ \Leftarrow ”: Assume a derivation of $(Y \uplus X) : T; \Gamma \vdash z/(Y \uplus X)$ with premise $(*) \Gamma \vdash z : T$.

We need to show $(Y \uplus X) : T; \Gamma \vdash z/(Y \uplus \{y\}) \circ (\text{id}_Y \otimes y/x)$. Thus, to use the rule for composition we must establish two premises: (1) $(Y \uplus X) : T; \Delta \vdash \text{id}_Y \otimes y/x$ and (2) $\Delta; \Gamma \vdash z/(Y \uplus \{y\})$, for some suitable Δ . We pick $\Delta = Y : T, y : T$. Then, (2) is derivable if we can establish $\Gamma \vdash z : T$, but this follows from $(*)$. Hence, only (1) remains. We can use the rule for tensor with premises (1A) $Y : T; Y : T \vdash \text{id}_Y$ and (1B) $X : T; y : T \vdash y/x$, because $Y : T, X : T \stackrel{\text{def}}{=} (Y \uplus X) : T$. (1A) is trivial by the rule for identities and (1B) trivial by the rule for substitutions.

– case $\text{join} \circ (\mathbf{1} \otimes \text{id}_1) = \text{id}_1$.

“ \Leftarrow ”: Clearly, $\Gamma_0; \Gamma_0$ types both sides by axioms.

– case $\text{join} \circ (\text{join} \otimes \text{id}_1) = \text{join} \circ (\text{id}_1 \otimes \text{join})$.

“ \Leftarrow ”: Clearly, $\Gamma_0; \Gamma_0$ types both sides by axioms.

– case $\text{join} \circ \gamma_{1,1}(\vec{\emptyset}, \vec{\emptyset}) = \text{join}$.

“ \Leftarrow ”: Clearly, $\Gamma_0; \Gamma_0$ types both sides by axioms.

– case $(\emptyset)P = P$.

“ \Rightarrow ”: Assume a derivation of $\Delta; \Gamma \vdash (\emptyset)P$ with premise $\Delta; \Gamma \vdash P$. Reuse the premise on the right-hand side.

“ \Leftarrow ”: Reuse the assumed derivation of $\Delta; \Gamma \vdash P$ as premise in the rule for abstraction.

– case $(Y)^\top Y^\top = \text{id}_{(Y)}$.

Recall that $\text{id}_{(Y)} \stackrel{\text{def}}{=} (Y)/(Y) \stackrel{\text{def}}{=} (Y)((\otimes_{i=1}^n y_i/y_i) \otimes \text{id}_1) \circ \ulcorner Y^\top$ for $Y = \{y_1, \dots, y_n\}$.

“ \Rightarrow ”: Assume a derivation of $\Gamma_0; \Gamma_1 \vdash (Y)^\top Y^\top$ with premise $(*) \Gamma_0; \Gamma_1 \vdash \ulcorner Y^\top$.

To build a derivation we essentially need two subderivations: (1) $\Gamma_0; \Gamma_1 \vdash \ulcorner Y^\top$ and (2) $\Gamma_1; \Gamma_1 \vdash \otimes_{i=1}^n y_i/y_i$, where (1) is simply by $(*)$. (2) follows from n subderivations of form $\Gamma_1 \upharpoonright y_i; \Gamma_1 \upharpoonright y_i \vdash y_i/y_i$ (where $\Gamma_1 \upharpoonright y_i$ restricts Γ_1 to y_i), because $\Gamma_1 \upharpoonright y_i \vdash y : (\Gamma_1 \upharpoonright y_i)(y_i)$ is by axiom and reflexivity of subtyping.

“ \Leftarrow ”: Assume a derivation of $\Gamma_0; \Gamma_1 \vdash \text{id}_{(Y)}$ with premises (1) $\Gamma_0; \Gamma_2 \vdash \ulcorner Y^\top$ and (2) $\Gamma_1 \upharpoonright y_i \vdash y_i : (\Gamma_2 \upharpoonright y_i)(y_i)$, for some Γ_2 such that $\Gamma_0; \Gamma_2 \vdash \ulcorner Y^\top$ and $\Gamma_2; \Gamma_1 \vdash$

- $\otimes_{i=1}^n y_i/y_i$. Then, (1) implies that $\Gamma_2 \leq \Gamma_0$ and (2) implies that $\Gamma_1 \leq \Gamma_2$, so $\Gamma_1 \leq \Gamma_0$ by transitivity of subtyping. Hence, we can derive $\Gamma_0; \Gamma_1 \vdash \ulcorner Y \urcorner$ and thus $\Gamma_0; \Gamma_1 \vdash (Y) \ulcorner Y \urcorner$ as required.
- case $(\ulcorner X \urcorner^Z \otimes \text{id}_Y) \circ (X)P = P$ for $P : I \rightarrow \langle 1, Z, Z \uplus X \uplus Y \rangle$.
Recall that $\ulcorner X \urcorner^Z \stackrel{\text{def}}{=} (Z) \ulcorner Z \uplus X \urcorner : \langle 1, Z \uplus X, Z \uplus X \rangle \rightarrow \langle 1, Z, Z \uplus X \rangle$.
“ \Rightarrow ”: Assume a derivation of $\Delta; \Gamma \vdash (((Z) \ulcorner Z \uplus X \urcorner) \otimes \text{id}_Y) \circ (X)P$ with subderivations (1) $\Delta; \Gamma' \vdash P$, (2) $\Gamma_0; \Gamma_1 \vdash \ulcorner Z \uplus X \urcorner$, and (3) $\Gamma_2; \Gamma_3 \vdash \text{id}_Y$, with $\Gamma = \Gamma_1, \Gamma_2$ and $\Gamma' = \Gamma_0, \Gamma_2$. (2) implies $\Gamma_1 \leq \Gamma_0$ and (3) implies $\Gamma_3 \leq \Gamma_2$, so together they imply $\Gamma = \Gamma_1, \Gamma_3 \leq \Gamma_0, \Gamma_2 = \Gamma'$. We have $\Delta; \Gamma' \vdash P$ from (1), and because $\Gamma \leq \Gamma'$ we obtain $\Delta; \Gamma \vdash P$ by Narrowing.
“ \Leftarrow ”: Assume a derivation (*) $\Delta; \Gamma \vdash P$. We can easily build the desired derivation of $\Delta; \Gamma \vdash (((Z) \ulcorner Z \uplus X \urcorner) \otimes \text{id}_Y) \circ (X)P$ as follows: We need the same subderivations as were assumed in the previous case, but by picking $\Gamma_0 = \Gamma_1$ and $\Gamma_2 = \Gamma_3$ we clearly have subderivations (2) and (3) from above by trivial uses of the rule for concretions and identities, respectively. Thus, we can derive $\Gamma; \Gamma \vdash (Z) \ulcorner Z \uplus X \urcorner \otimes \text{id}_Y$. Finally, we can reuse (*) and by the rule for composition we obtain the *deciderata*.
 - case $((Y)P \otimes \text{id}_X) \circ G = ((Y)(P \otimes \text{id}_X)) \circ G$.
“ \Leftarrow ”: Clearly, because the subderivations are the same on both sides albeit slightly rearranged, and only the rules for abstraction, tensor and composition are used to build the required derivations.
 - case $(X \uplus Y)P = (X)((Y)P)$.
“ \Leftarrow ”: Clearly, because the names are the same on both sides, and the rule for abstraction merely propagates information.
 - case $(\text{id}_1 \otimes \alpha) \circ K_{\vec{y}(\vec{x})} = K_{\alpha(\vec{y})(\vec{x})}$. Two cases: Either K is send or get.
 - $K_{\vec{y}(\vec{x})} = \text{send}_{az}$. We have $\alpha \stackrel{\text{def}}{=} a'/a \otimes z'/z$ for some names a', z where $a' \notin \{z', z\}$ and $z' \notin \{a', a\}$ because α is a renaming, i.e. a bijective substitution. ($a \neq z$ by definition.)
“ \Rightarrow ”: Assume a derivation of $\Gamma_0; \Gamma \vdash (\text{id}_1 \otimes \alpha) \circ \text{send}_{az}$ with subderivations (A) $\Gamma_0; \Theta \vdash \text{send}_{az}$ and (B) $\Theta; \Gamma \vdash \text{id}_1 \otimes \alpha$ with $\text{supp}(\Theta) = \{a, z\}$ and $\text{supp}(\Gamma) = \{a', z'\}$. (A) has premises (A1) $\Theta^a \vdash a : \circ T$ and (A2) $\Theta^z \vdash z : T$, where Θ^a denotes $\Theta \upharpoonright \{a\}$ and so forth. (B) has premises (B1) $\Gamma^{a'} \vdash a' : \Theta^a(a)$ and (B2) $\Gamma^{z'} \vdash z' : \Theta^z(z)$.
Build the desired derivation of $\Gamma_0; \Gamma \vdash \text{send}_{a'z'}$ with premises (i) $\Gamma^{a'} \vdash a' : \circ T$ and (ii) $\Gamma^{z'} \vdash z' : T$. (i) is justified as follows: By (B1), $\Gamma^{a'}(a') \leq \Theta^a(a)$. So, by Narrowing on (A1) we obtain $\Gamma^{a'} \vdash a' : \circ T$, bearing in mind that (*) $x : S \vdash x : T \text{ iffy } S \vdash y : T$. (ii) is justified in an analogously.
“ \Leftarrow ”: Find $\Theta = \Theta^a, \Theta^z$ with $\Theta^a(a) = \Gamma^{a'}(a')$ and $\Theta^z(z) = \Gamma^{z'}(z')$, yielding (B1) and (B2). Then, (A1) follows easily from (i) and (A2 from (ii), because of (*).
 - $K_{\vec{y}(\vec{x})} = \text{get}_{a(z)}$. We have $\alpha \stackrel{\text{def}}{=} a'/a$ for some $a' \neq z$ w.l.o.g.
“ \Rightarrow ”: Assume a derivation of $z : S; \Gamma \vdash (\text{id}_1 \otimes a'/a) \circ \text{get}_{a(z)}$ with premises (1) $a : U \vdash a : iS$ and (2) $\Gamma \vdash a' : U$, for some types S and U . (2) implies

that $\Gamma(a') \leq U$ so we may apply Narrowing to (1) and obtain $\Gamma \vdash a' : iS$, because $a : U \vdash a : iS$ iff $a' : U \vdash a' : iS$. Then we can build a derivation of $z : S; \Gamma \vdash \text{get}_{a'(z)}$, as required.

“ \Leftarrow ”: Analogous to the “ \Leftarrow ” direction of the send case.

- case $K_{\vec{y}(\vec{X})} \circ \sigma^{\text{loc}} = K_{\vec{y}((\sigma^{\text{loc}})^{-1}(\vec{X}))}$. Two cases. Either K is send or get.
 - $K_{\vec{y}(\vec{X})} = \text{send}_{ax}$. This case holds both ways trivially because send has no local names.
 - $K_{\vec{y}(\vec{X})} = \text{get}_{a(z)}$. We have $\sigma^{\text{loc}} = (z)/(Z)$ for some name set Z . Recall that $(z)/(Z) \stackrel{\text{def}}{=} (z)((z/z \otimes \text{id}_1) \circ \Gamma Z^\top)$.
 - “ \Rightarrow ”: Assume a derivation of $Z : T; \Gamma \vdash \text{get}_{a(z)} \circ (z)((z/z \otimes \text{id}_1) \circ \Gamma Z^\top)$ with premises (1) $S \leq T$ and (2) $\Gamma \vdash a : iS$. Building a derivation of $Z : T; \Gamma \vdash \text{get}_{a(z)}$ requires $\Gamma \vdash a : iT$, which in turn requires (A) $\Gamma \vdash a : iS$ and (B) $iS \leq iT$. (A) follows from (2) and (B) from (1) by the subtyping rule of input types.
 - “ \Leftarrow ”: The same premises are needed.

- case $b = b$ (reflexivity).
Show that $\Delta; \Gamma \vdash b$ if and only if $\Delta; \Gamma \vdash b$, but this is immediate.
- case $b = b' \implies b' = b$ (symmetry).

Read the statement as a rule: $\frac{b = b'}{b' = b}$.

Assume: $b_0 = b_1$. Show: $\Delta; \Gamma \vdash b_0$ if and only if $\Delta; \Gamma \vdash b_1$. We are proceeding by induction on the derivation of $b_0 = b_1$ and the last rule used was the rule for symmetry, so we must have had $b_1 = b_0$. Now, by induction hypothesis on $b_1 = b_0$ we obtain $\Delta; \Gamma \vdash b_1$ if and only if $\Delta; \Gamma \vdash b_0$. Because “iff” is symmetric we have the *desiderata*.

- case $b = b' \& b' = b'' \implies b = b''$ (transitivity).

Read the statement as a rule: $\frac{b = b' \quad b' = b''}{b = b''}$.

Assume: $b_0 = b_1$. Show: $\Delta; \Gamma \vdash b_0$ if and only if $\Delta; \Gamma \vdash b_1$. We have must have had $b_0 = b_2$ and $b_2 = b_1$ for some b_2 . By induction hypothesis: (1) $\Delta; \Gamma \vdash b_0$ if and only if $\Delta; \Gamma \vdash b_2$, and (2) $\Delta; \Gamma \vdash b_2$ if and only if $\Delta; \Gamma \vdash b_1$. Because “iff” is transitive we obtain $\Delta; \Gamma \vdash b_0$ if and only if $\Delta; \Gamma \vdash b_1$, as required.

- case $b = b' \implies C \circ b = C \circ b'$ (congruence).

Read the statement as a rule: $\frac{b = b'}{C \circ b = C \circ b'}$.

Assume: $C \circ b = C \circ b'$. Show: $\Delta; \Gamma \vdash C \circ b$ if and only if $\Delta; \Gamma \vdash C \circ b'$. We must have had $b = b'$. By induction hypothesis on $b = b'$ we obtain $\Delta; \Theta \vdash b$ if and only if $\Delta; \Theta \vdash b'$, for some Θ . Typings of $C \circ b$ and $C \circ b'$ must have ended with the composition rule. Thus, we just need to argue that we have a derivation of $\Theta; \Gamma \vdash C$ if and only if $\Theta; \Gamma \vdash C$, but this is immediate (because “iff” is reflexive).

□

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

$$(iii) : \frac{\frac{\frac{(iii')}{W' \leq S}}{y : S; x : W' \vdash x/y} \quad \Gamma_0; \Gamma_0 \vdash id_1}{y : S; x : W' \vdash x/y \otimes id_1}}{y : S; y : S \vdash \ulcorner y \urcorner} \quad y : S; x : W' \vdash (x/y \otimes id_1) \circ \ulcorner y \urcorner$$

Remark: In (iii), $(x)/_y \stackrel{\text{def}}{=} (x)((x/y \otimes id_1) \circ \ulcorner y \urcorner)$.

$$(iv) : \frac{(iv')}{x : W'; x : W' \vdash \psi} \quad (v) : \frac{\Gamma_0; a : W \vdash a/\emptyset}{\Gamma_0; a : W \vdash (a)(a/\emptyset)}$$

Remarks: In the subderivation (iv') we choose not to subtype. In derivation (v), $(a) \stackrel{\text{def}}{=} (a)(a/\emptyset)$, and any type may be chosen for a – pick W .

$$(vi) : \frac{(vi')}{a : W, x : W'; a : W, x : W' \vdash \phi}$$

Remark: In the subderivation (vi') we choose not to subtype. \square

Proposition 1 (Type Soundness) *Suppose that process bigraph $b = \llbracket P \rrbracket_{(x)}$, $\Gamma_0; \Gamma \vdash b$, and $b \rightarrow^* b'$. Then, for each non-idle $a \in \text{glob}(\text{cod}(b'))$ it holds that:*

1. *If $\Gamma \vdash a : iS$ 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 $\Gamma \vdash a : oT$ then a is linked to a send ion.*

Proof. The proof is by ind. on the length n of the reduction $b \rightarrow^* b'$. The base case is by structural induction on P using Cor. 1. The inductive case uses Subject Reduction.

- $n = 0$. We have $b = b'$. Either $\Gamma(a) = iS$ or $\Gamma(a) = oT$.

Assume $\Gamma(a) = iS$. Clearly a can be linked to a get ion, but not to a send ion because a premise of the send rule requires a to have a type of form oT , which contradicts the assumption. The other possibility could be that a is linked to the datum channel of a send ion, say send_{ax} , because links x of input type can be communicated, as long as $\Gamma \vdash a : oiS$ and $\Gamma \vdash x : iS$ for some Γ , S and T .

Assume $\Gamma(a) = oT$. Clearly a must be linked to a send ion because the premise of the get rule requires a to have a type of form iS , which contradicts the assumption. Again, a can either be linked to a channel port of send or the datum port if it is to be communicated.

- $n > 0$. Assume the induction hypothesis for $b \rightarrow^n b^n$ and show it for $b \rightarrow^{n+1} b'$. Thus, b^n exhibits the property. Show that b^{n+1} does too. By Subject Reduction we know that $\Gamma_0; b^n \vdash \Gamma$ and taking another step with \rightarrow^1 establishes $\Gamma_0; b^{n+1} \vdash \Gamma$. The outer typing being preserved we just need arguments as for $n = 0$ to establish the desiderata. \square

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

Proof. The proof is by induction on the height of the derivation of $\Delta; \Gamma \vdash b$. Recall that $(x) \stackrel{\text{def}}{=} (x)(x/\emptyset)$ and can thus be typed by using the rule for abstraction and then the rule for substitutions.

All cases follow the same pattern, where the desired derivation is constructed as follows:

$$\frac{\frac{\text{"by assumption"}}{\Delta; \Gamma \vdash b} \quad \frac{\Gamma_0; x : T \vdash x/\emptyset}{\Gamma_0; x : T \vdash (x)}}{\Delta; \Gamma \vdash b \otimes (x)}$$

The case for closure $/x$ holds by the insight that an outer name x is different from an inner name x if they are not linked. \square

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

Proof. The proof is by induction on the height of the derivation of $\Delta; \Gamma, x : T \vdash b \otimes (x)$. Recall that $(x) \stackrel{\text{def}}{=} (x)(x/\emptyset)$ and can thus be typed by using the rule for abstraction and then the rule for substitutions.

All cases follow the same pattern, where the assumed derivation has form:

$$\frac{\frac{(*)}{\Delta; \Gamma \vdash b} \quad \Gamma_0; x : T \vdash (x)}{\Delta; \Gamma, x : T \vdash b \otimes (x)}$$

The desired derivation is simply constructed by reusing $(*)$. \square

Lemma 6. *Suppose $b = \llbracket P \rrbracket_{(X)}$ and $\Gamma_0; \Gamma \vdash b$ 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 .*

Proof. The proof is by structural induction on P . Let LHS mean “left-hand side” and RHS “right-hand side”. We sometimes write Xy for $X \uplus \{y\}$, for instance.

- case $P = \mathbf{0}$. LHS: $\llbracket \mathbf{0} \rrbracket_{\text{supp}(\Gamma)} \otimes (x) \stackrel{\text{def}}{=} (X) \otimes (x) = (X \uplus \{x\})$. RHS: $\llbracket \mathbf{0} \rrbracket_{\text{supp}(\Gamma, x:T)} = (X \uplus \{x\})$.
- case $P = \nu z.Q$.
 LHS: $\llbracket \nu z.Q \rrbracket_{(X)} \otimes (x) \stackrel{\text{def}}{=} / (z) \triangleleft \text{id}_X \circ \llbracket Q \rrbracket_{(Xz)} \otimes (x)$.
 RHS: $\llbracket \nu z.Q \rrbracket_{(\text{supp}(\Gamma, x:T))} \stackrel{\text{def}}{=} / (z) \triangleleft \text{id}_{Xx} \circ \llbracket Q \rrbracket_{(Xxz)}$. We know that x is idle in $\llbracket Q \rrbracket$ so we may further rewrite to $/ (z) \triangleleft \text{id}_X \circ \llbracket Q \rrbracket_{Xz} \otimes (x)$.
- case $P = \bar{a}z.Q$.
 LHS: $\llbracket \bar{a}z.Q \rrbracket_{(X)} \otimes (x) \stackrel{\text{def}}{=} \text{send}_{az} \triangleleft \text{id}_X \circ \llbracket Q \rrbracket_{(X)} \otimes (x)$.
 RHS: $\llbracket \bar{a}z.Q \rrbracket_{(\text{supp}(\Gamma, x:T))} \stackrel{\text{def}}{=} \text{send}_{az} \triangleleft \text{id}_{Xx} \circ \llbracket Q \rrbracket_{(Xx)} = \text{send}_{az} \triangleleft \text{id}_{Xx} \circ \llbracket Q \rrbracket_{(X)} \otimes (x)$ because x is idle in $\llbracket Q \rrbracket$.

- case $P = a(y).Q$.
 - LHS: $\llbracket a(y).Q \rrbracket_{(X)} \otimes (x) \stackrel{\text{def}}{=} \text{get}_{a(y)} \triangleleft \text{id}_X \circ \llbracket Q \rrbracket_{(X)} \otimes (x)$.
 - RHS: $\llbracket a(y).Q \rrbracket_{(\text{supp}(\Gamma, x; T))} \stackrel{\text{def}}{=} \text{get}_{a(y)} \triangleleft \text{id}_{Xx} \circ \llbracket Q \rrbracket_{(Xx)} \otimes (x) = \text{get}_{a(y)} \triangleleft \text{id}_X \circ \llbracket Q \rrbracket_{(X)} \otimes (x)$ because x is idle in $\llbracket Q \rrbracket$.
- case $P = Q \mid Q'$.
 - LHS: $\llbracket Q \mid Q' \rrbracket_{(X)} \otimes (x) \stackrel{\text{def}}{=} (\llbracket Q \rrbracket_{(X)} \mid \llbracket Q' \rrbracket_{(X)}) \otimes (x)$.
 - RHS: $\llbracket Q \mid Q' \rrbracket_{(\text{supp}(\Gamma, x; T))} \stackrel{\text{def}}{=} \llbracket Q \rrbracket_{(\text{supp}(\Gamma, x; T))} \mid \llbracket Q' \rrbracket_{(\text{supp}(\Gamma, x; T))} \stackrel{\text{def}}{=} \llbracket Q \rrbracket_{(Xx)} \mid \llbracket Q' \rrbracket_{(Xx)} = \llbracket Q \rrbracket_{(X)} \mid \llbracket Q' \rrbracket_{(X)} \otimes (x)$ because x is idle in $\llbracket Q \rrbracket$ and $\llbracket Q' \rrbracket$. \square

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. The proof is by struct. induction on P using Lemmas 6 and 3.

- case $P = \mathbf{0}$.
 - “ \Rightarrow ”:
- case $P = (vx : L)Q$.
 - “ \Rightarrow ”:
- case $P = \bar{a}x.Q$.
 - “ \Rightarrow ”:
- case $P = a(y).Q$.
 - “ \Rightarrow ”:

– case $P = Q \mid Q'$.

“ \Rightarrow ”: Assume $\Gamma \vdash Q \mid Q' : \diamond$ with premises (1) $\Gamma \vdash Q : \diamond$ and (2) $\Gamma \vdash Q' : \diamond$. We

have $\llbracket Q \mid Q' \rrbracket_{(X)} \stackrel{\text{def}}{=} \sigma \circ (\llbracket Q \rrbracket_{(X)} \otimes \tau \circ \llbracket Q' \rrbracket_{(X)})$, for suitable substitutions σ and τ . Now, let $W = \text{fn}(Q) \cap \text{fn}(Q')$. Then, if $Z = \text{fn}(Q) \setminus W$ then $\tau = Z/Z \otimes F/W$ and $Y = Z \uplus F$. To build $\Gamma_\emptyset; \Gamma \vdash \sigma \circ (\llbracket Q \rrbracket_{(X)} \otimes \tau \circ \llbracket Q' \rrbracket_{(X)})$ we must establish three premises: (i) $\Gamma_\emptyset; \Gamma \upharpoonright Q \vdash \llbracket Q \rrbracket_{(\text{fn}(Q))}$, (ii) $\Gamma_\emptyset; \Gamma \upharpoonright Q' \vdash \llbracket Q' \rrbracket_{(\text{fn}(Q'))}$, (iii) $\Gamma \upharpoonright Q'; \Gamma \upharpoonright Y \vdash \tau$, and (iv) $(\Gamma \upharpoonright Q), (\Gamma \upharpoonright Y); \Gamma \vdash \sigma$. By induction hypothesis on (1) and (2) we obtain $\Gamma_\emptyset; \Gamma \vdash \llbracket Q \rrbracket_{(X)}$ and $\Gamma_\emptyset; \Gamma \vdash \llbracket Q' \rrbracket_{(X)}$. Then, by Lemma 6 and Strengthening we obtain $\Gamma_\emptyset; \Gamma \upharpoonright \text{fn}(Q) \vdash \llbracket Q \rrbracket_{(X)}$ and $\Gamma_\emptyset; \Gamma \upharpoonright \text{fn}(Q') \vdash \llbracket Q' \rrbracket_{(X)}$, because $X \setminus \text{fn}(Q)$ is idle in $\llbracket Q \rrbracket$ and likewise for Q' . (iii) and (iv) follow trivially from the rule for substitutions, where we choose not to subtype.

“ \Leftarrow ”: Assume $\Gamma_\emptyset; \Gamma \vdash \sigma \circ (\llbracket Q \rrbracket_{(X)} \otimes \tau \circ \llbracket Q' \rrbracket_{(X)})$ with the following four premises (i) $\Gamma_\emptyset; \Gamma \upharpoonright \text{fn}(Q) \vdash \llbracket Q \rrbracket_{(\text{fn}(Q))}$, (ii) $\Gamma_\emptyset; \Gamma \upharpoonright \text{fn}(Q') \vdash \llbracket Q' \rrbracket_{(\text{fn}(Q'))}$, (iii) $\Gamma \upharpoonright \text{fn}(Q'); \Gamma \upharpoonright Y \vdash \tau$, and (iv) $(\Gamma \upharpoonright \text{fn}(Q)), (\Gamma \upharpoonright Y); \Gamma \vdash \sigma$. (i) and (ii) imply (1) and (2), respectively, by the induction hypothesis, Lemma 6, and Weakening. τ and σ are immaterial because when using Weakening we pick the “right” types. Thus, we have established $\Gamma \vdash Q : \diamond$ and $\Gamma \vdash Q' : \diamond$ so by the rule for parallel composition we obtain $\Gamma \vdash Q \mid Q' : \diamond$, as required. \square