Bigraphical Semantics of Higher-Order Mobile Embedded Resources with Local Names

Mikkel Bundgaard
Thomas Hildebrandt

IT University Technical Report Series **TR-2005-70**

ISSN 1600–6100 **September 2005**

ISSN 1600–6100

ISBN 87-7949-104-9

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`

Bigraphical Semantics of Higher-Order Mobile Embedded Resources with Local Names[*]

Mikkel Bundgaard and Thomas Hildebrandt
Department of Theoretical Computer Science
IT University of Copenhagen
Denmark
{mikkelbu,hilde}@itu.dk

Abstract

Bigraphs have been introduced with the aim to provide a topographical meta-model for mobile, distributed agents that can manipulate their own linkages and nested locations, generalising both characteristics of the π-calculus and the Mobile Ambients calculus. We give the first bigraphical presentation of a non-linear, higher-order process calculus with nested locations, non-linear active process mobility, and local names, the calculus of *Higher-Order Mobile Embedded Resources (Homer)*. The presentation is based on Milner's recent presentation of the λ-calculus in local bigraphs. The combination of non-linear active process mobility and local names requires a new definition of parametric reaction rules and a representation of the location of names. We suggest *localised bigraphs* as a generalisation of local bigraphs in which links can be further localised.

Keywords: bigraphs, local names, non-linear process mobility

Introduction

The theory of *Bigraphical Reactive Systems* (BRS) [JM04] has been proposed as a topographical meta-model for mobile, distributed agents that can manipulate their own linkages and nested locations. Bigraphs generalise both the link structure characteristic to the π-calculus and the nested location structure characteristic to the Mobile Ambients calculus. A bigraph consists of two structures: the *place graph* and the *link graph*. The *place graph* is a tuple of unordered trees that represents the topology of the system (why it is also referred to as the topo graph). The roots of the trees are referred to as *regions* and the nodes are often referred to as *places* and may represent locations or other process constructors such as e.g. action prefixing. Some of the leaves may be *sites* (also referred to as holes) making the bigraph a (multi-hole) context. Each non-site place is typed with a *control* and has a number of *ports* linked together by the link graph. The *link graph* represents the connectivity in the system, corresponding to shared names in the π-calculus. Free names are represented by links connected to a set of names in the (outer) *interface* of the bigraph. Figure 1 depicts a pure bigraph with 2 regions (r_0 and r_1) which together with the

[*]Funded by the Danish Research Agency grant: 2059-03-0031 (LaCoMoCo) and IT-Vest networking universities: National Teaching Network: Model-Based Design for Concurrency.

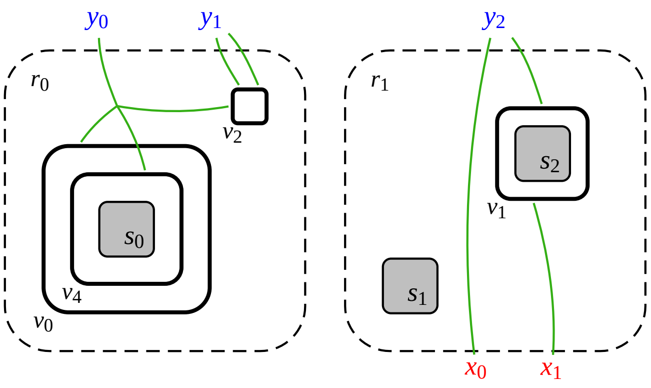

Figure 1: A pure bigraph

names y_0, y_1 and y_3 constitute the *outer interface* $\langle 2, \{y_0, y_1, y_2\}\rangle$, and 3 sites ($s_0$, s_1, and s_2) which together with the names x_0 and x_1 constitute the *inner interface* $\langle 3, \{x_0, x_1\}\rangle$ of the bigraph. A bigraph G can be composed with a bigraph H if the outer interface of H match with the inner interface of G and the result is that the contents of the regions of H is placed in the respective sites of G and the links in the outer interface of H is connected to the links in the inner interface of G.

In so-called *pure* bigraphs, the place and link graph can be considered to be orthogonal structures, since the nesting of the places and the connections of the links have no interrelationship. Pure bigraphs are sufficient to represent calculi such as the pure Mobile Ambient calculus. The orthogonality breaks when we move to so-called *binding* and *local* bigraphs. Binding bigraphs were introduced in [JM03] to capture the notions of binding and scope of names as found in the π-calculus. In binding bigraphs we allow for a node to have *binding ports*, and require that any other port linked to the same link as a binding port to be within the node of the binding port. In [Mil04b], Milner refines the definition of binding bigraphs into *local bigraphs*. In local bigraphs, the free names (i.e. names in the interface) are all explicitly located at the regions of the bigraph, the same name possibly located at several regions. Correspondingly, holes (i.e. sites) are explicitly annotated by a set of names connected to links. Local bigraphs are used to facilitate the presentation of the λ-calculus as the bigraphical reactive system ´ΛBIG in [Mil04c], which demonstrates how higher-order processes (process passing) can be presented in the bigraphical framework using explicit substitutions.

In the present paper we give the first bigraphical presentation of the combination of active processes in nested locations as present in the Mobile Ambients, non-linear higher-order process passing (by explicit substitutions) as present in the λ-calculus and local names as present in the π-calculus. It turns out that the combination of non-linear, active process mobility, and local names needs special care, i.e. we can not simply combine the previous presentations of the Mobile Ambients, the λ-calculus, and the π-calculus.

We take as our starting point the calculus of Higher-Order Mobile Embedded Resources (Homer) [HGB04]. Homer is a pure higher-order calculus inspired by prior higher-order calculi such as Plain CHOCS [Tho93] and HOπ [San92], and can be regarded as an extension of the λ-calculus to contain nested, active locations and concurrent synchronisation over (nested) named channels. Basically, Homer has two prefixes for located resources $\overline{\delta}\langle r\rangle$ (passive) and $\delta[r]$ (active) where δ is a sequence of names representing the address of the resource. Intuitively, these two prefixes correspond respectively to a passive and an active bigraph control with ports connected to the links δ. The interactions are controlled by two corresponding constructors for moving located resources $\delta(x) \,.\, p$ (receive) and $\overline{\delta}(x) \,.\, p$ (take), denoting respectively the usual input-prefixed process waiting to receive a (passive) process on the channel δ, and an input action for taking an *active* process from location δ, in both cases substituting the moved resource in for x in p. The two interactions are captured by the two "dual" reaction rules.

$$\overline{a}\langle r\rangle \,.\, p \mid a(x) \,.\, q \searrow p \mid q[r/x] \tag{1}$$

and

$$a[r] \,.\, p \mid \overline{a}(x) \,.\, q \searrow p \mid q[r/x] \tag{2}$$

The first rule (1) is the basic interaction as known from Plain CHOCS [Tho93]. Here the process to the left wants to send out the *passive* resource r along the name a and the process to the right wants to receive a resource along a and substitute this resource in for x in q. As the process variable x in the receiving prefix can bind any number of occurrences of x in q we have the possibility to discard the input or copy the input, resembling β-reduction know from the λ-calculus [Bar84], since process passing in Homer is non-linear.

In Plain CHOCS the process r is *passive*, meaning that r cannot compute internally, nor can it interact with the surrounding environment. Furthermore, if the receiving process q decides to activate an instance of the received resource, then this instance cannot be moved again. We have chosen to augment the passive process passing of Plain CHOCS with *active* process passing, as in (2), where we have a resource r *computing* at location a and a prefix for taking up the running process at location a and bind it to x, respectively. Compared to the passive prefix it is now possible for r to compute internally while residing at the location a, i.e.

$$r \searrow r' \text{ implies } a[r] \searrow a[r'] \ . \tag{3}$$

When q takes the running resource r, a copy may again be placed at a location where it can continue to run until it is taken again. So we have active *process mobility* in Homer in addition to the passive *code mobility* as e.g. found in Plain CHOCS and HOπ.

We also allow for location prefixes to be nested, hence we have an explicit representation of nested, named locations, as also introduced in the Mobile Ambient calculus [CG00] or the Seal Calculus [CVN04]. However, contrarily to these calculi, we allow interactions with arbitrarily deeply nested, active processes by simply composing addresses. In the example below we send the resource r down to the nested address ab (composed of a and b), and it is received at the address b residing in the location a

$$\overline{ab}\langle r\rangle \,.\, p \mid a[b(x)\,.\,q \mid q']\,.\,q'' \searrow p \mid a[\, q[r/x] \mid q']\,.\,q'' \;\; . \tag{4}$$

Dually, we can also take up resources from nested locations as in

$$a[b[r]\,.\,p \mid p']\,.\,p'' \mid \overline{ab}(x)\,.\,q \searrow a[p \mid p']\,.\,p'' \mid q[r/x] \;\; . \tag{5}$$

In general, we allow interaction with arbitrarily deeply nested sub resources. However, two processes that are neither locally parallel nor in the sub/parent process relation need a common ancestor process to act as a router that mediates communication. We also allow composite names for active locations and receive prefixes. Besides making the calculus symmetric, it also allows us to express nested location names without allowing all sub locations to be moved separately.

As usual, we let $(n)p$ denote a process p in which the name n is local. With local names we also need to handle scope extension. For most of the process constructors scope extension is as expected, but for locations we need to take particular care. For when a resource is moved it may be necessary to extend the scope of a name through the boundary of a location, e.g. if the resource r contains the name n free, we will expect the reaction

$$a[(n)(b[r] \mid p)] \mid \overline{ab}(x)\,.\,q \searrow (n)(a[p] \mid q[r/x]) \;\; , \tag{6}$$

where we have extended the scope of n to cover all possible occurrences of the name n. In [HGB04] we called this kind of scope extension for *vertical* scope extension, to differentiate it from the usual kind of scope extension, where we extend the scope over parallel processes at the same location. In the Mobile Ambients calculus vertical scope extension is performed in the structural congruence (along with the usual scope extension)

$$m[(n)p] \equiv (n)m[p] \;\; , \text{ if } n \neq m \;\; . \tag{7}$$

However, as discovered in several calculi, this rule is not sound when mobile processes may be copied. There exists several solutions to this problem, all of them exclude the vertical scope extension in the structural congruence (7), and instead extend the scope in the reaction relation. This extension is either done *eagerly*, meaning that we always extend the scope, or *if and only if* the name n is free in r. In Homer we have chosen the latter solution, which corresponds to the usual semantics of e.g. HOπ. Combined with nested locations it has the consequence that a context can test if a name is free in a process (assuming that n is free in p)

$$m[(n)(m'[r] \mid p)] \mid \overline{mm'}(y)\,.\,\overline{m}(x)\,.\,(x \mid x) \;\; . \tag{8}$$

Since the scope of n is extended if and only if n is free in r, the process reduces (in two steps) to one of the following processes

$$(n)p \mid (n)p \tag{9}$$

or

$$(n)(p \mid p) \ , \tag{10}$$

depending on whether n is free in r. Consequently, for for any non-trivial congruence related processes must have the same set of free names (see, e.g., [HGB04] for a detailed discussion).

It is sometimes useful, however, to be able to abstract from free, but non-accessible names, as e.g. in the *perfect firewall equation* [CG00]

$$(n)(n[p]) \approx \mathbf{0} \ , \tag{11}$$

stating that the behaviour of a computing resource at a local location is unobservable. However, when the context can test for free names of a processes this equivalence will only hold if $fn(p) = \emptyset$. In [HGB04] the process constructor *free name extension* was introduced to mend this problem. The constructor $\{n\}p$ extends the free names of p with the idle name n. Using an example similar to (8) it was shown that name extension cannot be extended vertically either. So not only does the free names of a process matter, but the *locality* of the names matters as well.

In this paper we chose an equivalent solution by typing processes explicitly with a set of names $\tilde{n}$ containing the free names. The *typed* perfect firewall equation then becomes

$$(n)(n[p]) : \tilde{n} \approx \mathbf{0} : \tilde{n} \quad \text{ for } fn(p) \backslash \{n\} \subseteq \tilde{n}. \tag{12}$$

As for the free name extension we need to be able to able to express the same requirement at all location and send prefixes, as we otherwise loose the information about the free names of a process when inserting it into a context, as the following example illustrates. Assume that the processes

$$p \overset{def}{=} \mathbf{0} \qquad \text{and} \qquad q \overset{def}{=} (m)(m[n[\mathbf{0}]]) \quad (\text{where } m \neq n)$$

are related by our equivalence under the typing $\{n\}$, e.g. $p : \{n\} \approx q : \{n\}$. Then consider the following context which will first lift the scope of n if and only if the inserted process contains the names n free, and the copy the remaining content of the location m'.

$$\mathcal{C} \overset{def}{=} m'[(n)(m''[(-)] \mid p')] \mid \overline{m'm''}(y) \,.\, \overline{m'}(x) \,.\, (x \mid x) \ \text{, where } n \in fn(p'). \tag{13}$$

Then we have the following reactions

$$\mathcal{C}(p) \searrow \searrow (n)p' \mid (n)p' \tag{14}$$

and

$$\mathcal{C}(q) \searrow \searrow (n)(p' \mid p') \ . \tag{15}$$

So when we insert p and q into a context we loose the information that they were related with the same type, e.g. that we considered them to have the same free name n. So we need type annotations on all locations and send prefixes to keep track of the free names, which is done by extending the syntax of prefixes to $\overline{\delta}\langle r\rangle_{\tilde{n}}$ and $\delta[r]_{\tilde{n}}$.

Related Work In [HGB04] Hildebrandt et al. introduce the Homer calculus. The calculus has a simple syntax and semantics that in many ways extends the traditional process passing calculi for concurrency and mobility. Using an extension of Howe's method the authors show that late labelled transition bisimulations are congruences, and hence a sound characterisation of barbed bisimulation congruence in terms of a late contextual bisimulation. The authors also propose a *free name extension* as a process constructor in calculi with explicit locations, local names, and non-linear process mobility. Bundgaard et at. presents in [BHG05a] an encoding of the synchronous π-calculus without summation, replication, and matching in the calculus of Homer. The encoding is proven fully abstract with respect to barbed bisimulation and sound with respect to barbed congruence. These results are proven by utilising an intermediate π-calculus with explicit substitutions. In [BHG05b] this result is extended to a π-calculus containing replication and matching, and an encoding that does not utilise any auxiliary names is presented.

Bigraphs is a refinement of the framework of *action calculi* [Mil96], which is based on process constructors inherited from the π-calculus (see e.g. [Gar00] for a gentle introduction to this connection). The graphical notion of *action graphs* introduced in [Mil96] inspired the bigraphical model. Leifer and Milner discovered that the categorical notion of *relative pushout* (RPO) could be used to characterise minimal labels in order to automatically derive bisimulation relations which are congruences [LM00], and these RPO transitions underly the behavioural theory of *bigraphical reactive systems* (BRS).

There already exists several presentations of well-known calculi for concurrency and mobility as bigraphical reactive systems. In [JM04, JM03] Jensen and Milner set up the basic theory of bigraphical reactive systems and exhibit a presentation of the asynchronous π-calculus Aπ and prove that the derived LTS and its bisimilarity match closely the traditional LTS and bisimilarity of asynchronous π-calculus. A sketch of a presentation of the Mobile Ambient calculus was also given in [JM04], but the formal treatment were postponed to Jensen's Ph.D. Thesis [Jen05]. Milner presented in [Mil04a] a presentation of condition-event Petri nets using only the link graph of bigraphs to represent the Petri nets. Again, a standard equivalence of condition-event nets is recovered using the framework of bigraphical reactive systems.

In a different direction, Milner has refined the theory of binding bigraphs [Mil04b], so that the location of a local name can be more than one region. The multiple locality of a name is utilised by Milner in [Mil04c] to present a variant of the λ-calculus with explicit substitutions, called Λ_{sub}. Several aspects of the presentation given in the current paper are inspired from Milner's presentation. Besides bigraphs there exist several graphical formalisms suitable for presenting calculi for concurrency and mobility: solo diagrams, synchronized hyperedge replacement, tile systems etc., see e.g. [BL05] for references.

Another topic touched upon in [Mil04c] is confluence. Milner uses the notion of *support* of a bigraph to determine if, and how, two ground redexes occurring in a ground bigraph overlap, and he uses the notion of relative pushouts to carry out "syntactic analysis" of bigraphs. O'Conchuir examines Λ_{sub} in [O'C04] and compares it to another λ-calculus with explicit substitutions λxgc [Ros96a, Ros96b] by Rose.

Currently, several other aspects of bigraphs are being investigated, such as the connection to XML in [HW05] by Hildebrandt and Winther (the prototype implementation of Reactive XML is extended to a distributed setting in [HNOW05]) and Conforti et al.'s logics for bigraphs [CMS05b] and its application to XML [CMS05a].

Explicit substitutions have been widely applied in the setting of functional programming languages, but primarily to bridge the gap between the abstract mathematical definition of a programming language and the concrete implementation of this language. In the seminal work of Abadi et al. [ACCL91] on $\lambda\sigma$, a λ-calculus with explicit substitutions, the explicit substitutions are propagated throughout the term and applied locally. The approach chosen in this paper differs from this solution, in the same way as Milner's lambda calculus did, since we also perform the substitution 'at a distance'. The λxgc [Ros96a, Ros96b] calculus resembles $\lambda\sigma$, but retains the variables names instead of introducing De Bruijn indices. Furthermore, in λxgc there is no composition of substitutions and the calculus has an explicit garbage collection of substitutions.

Explicit substitutions have also appeared in process calculi for concurrency and mobility. Ferrari et al. presented in [FMQ96] a π-calculus with explicit substitutions, $\pi\xi$, where the substitutions are recorded in a global environment ξ. The motivation for splitting up the name instantiation from the transitional semantics is to utilise the structural operational semantics meta-theory developed for traditional process calculi such as CCS in the setting of the π-calculus. Another investigation of a π-calculus with explicit substitutions was performed by Hirschkoff in [Hir99] using De Bruijn indices and handling the name instantiation using a term rewrite system. The calculus of explicit fusions, pi-F, [GW00] by Gardner and Wischik contains fusions, processes of the form $x = y$, explicitly in the syntax enabling the interchange of the names x and y in the surrounding context. Zimmer utilised in [Zim04] a π-calculus with explicit substitutions and channels as an intermediate language for proving that the synchronous π-calculus can be encoded in a restricted Mobile Ambient calculus containing only the mobility primitives and the hierarchical structure of the ambients.

Structure of the Paper In Section 1 we present briefly the main concepts of local bigraphs of Milner. In Section 2 we introduce the two variants of the calculus of Higher-Order Mobile Embedded Resources. In the first variant we have substituted the existing free name constructor $\{n\}p$ with simple type annotations at locations and send prefixes, and only consider typed relations between processes. In the second variant, Homerσ, we augment the calculus with explicit substitutions. We prove that there is an operational correspondence between the two variants. Section 3 contains the presentation of Homerσ as a bigraphical reactive system, the translation of Homerσ terms into bigraphs, and the translation of path contexts and the reaction rules. We prove that structural congruence of Homerσ corresponds to graph isomorphism in bigraphs. In Section 3.5 we present the operational correspondence between Homerσ and its presentation ´Homerσ. In Section 4 we present a generalisation of local bigraphs in which links can be further localised, called localised links. In Section 4 we also examine the problems arising from representing the restriction constructor using a closed free link in bigraphs. We conclude and propose further work in Section 5.

1 Local Bigraphs

In this section we introduce the main concepts of local bigraphs [Mil04b] of Milner. We refer the reader to [JM04] for additional information regarding the basic theory of (pure and binding) bigraphs and [Mil04b] and [Mil04c] for the remaining details about local bigraphs.

In this paper we will primarily use a simple term language, introduced in the above mentioned papers, instead of the graphical representation of bigraphs. The term language consists of the following constructors: $h \mid\mid g$ and $h \mid g$ are the *parallel product* and *prime parallel product* of two bigraphs h and g, respectively. Whereas the prime parallel product merges the regions of two single-region (prime) bigraphs, the parallel product juxtaposes the regions. The *closure* constructor $/n \circ g$ is the bigraph g, where we have removed the outer name n by replacing the name with an edge in g.

A bigraphical reactive systems is defined with respect to a signature that specifies the controls and their properties (e.g. the number of ports on the control, and whether the control is passive, active or atomic). Each node in a bigraph is associated with a control. The following definitions are defined as in [Mil04b].

Definition 1.1 (local signature)**.** A *local signature* $\mathcal{K}$ is a set whose elements are called *controls*. For each control K it provides a pair of finite ordinals, the binding arity h and the free arity k, indexing respectively the *binding* and the *free* ports of any K-node, written $K : h \to k$. The signature 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*. If K is atomic then $h = 0$.

Definition 1.2 (local interface)**.** A *local interface* takes the form $I = \langle m, \vec{X} \rangle$, where m is the width of the interface and $\vec{X}$ is a vector of length m, such that X_i is the set of names local to the $i'th$ site. We call $I^u = \langle m, \cup_{0 \leq i < m} X_i \rangle$ the pure interface *underlying* I.

We will often write $\vec{X}$ or $\langle m, loc, X \rangle$, where $X = \cup_{0 \leq i < m} X_i$ and $loc \subseteq m \,\{\rangle\, X$ is a *locality* relation, for the local interface $\langle m, \vec{X} \rangle$. We will also often write X for the local interface $\langle 1, X \rangle$.

Definition 1.3 (local bigraph)**.** If I and J are local interfaces, a *(concrete) local bigraph* $G : I \to J$ consists of an *underlying* pure bigraph $G^u : I^u \to J^u$ satisfying the same scoping conditions as in [Mil04b]. For a local bigraph $G : I \to J$ we call I and J the inner and outer *face* of G, respectively.

We can compose two local bigraphs H and G, if the outer face of G and inner face of H matches, resulting in the bigraph $H \circ G$, where the content of the regions of G have been inserted into the respective sites of H, and the links of corresponding local names have been fused together. We call a local bigraph G with unit inner face, $G : \varepsilon \to J$, a *ground* bigraph and write it as $G : J$. We define the *dynamics* of a bigraphical reactive system in terms of reaction rules and a reaction relation, which are defined precisely as in [Mil04b].

Definition 1.4 (reaction rule, reaction relation)**.** A *ground reaction rule* is a ground pair (r, r'), *redex* and *reactum*, with the same outer face. Given a set of ground rules, the *reaction relation* $\longrightarrow\!\!\!\!\triangleright$ over agents is the least, closed under support equivalence ($\simeq$), such that $D \circ r \longrightarrow\!\!\!\!\triangleright D \circ r'$ for each active D and each ground rule (r, r').

In order to easen the specification of the reaction relation we use parametric reaction rules, which allows rules that arbitrarily transform their parameters. Differently from the original definition in [Mil04b], we require that all outer names of a parameter are specified explicitly by the parametric reaction rule, to ensure that we handle scope extension properly. In the definition of parametric reaction rule we have also left out the specification of $\vec{\iota}$, as we do not need any renaming in any of the rules. But first we need that we can factorise a bigraph if none of its links cross regions.

Proposition 1.5 (factorisation)**.** *We can uniquely factorise any ground bigraph $c : \vec{X}$ with outer width m into primes, if none of its links cross regions, as*

$$c = c_0 \parallel \cdots \parallel c_{m-1} \ \text{, with } c_i : X_i \ .$$

The instantiation maps a parameter for the redex to a parameter for the reactum in a parametric reaction rule and allows for the rule to replicate some of the parameters and discard others.

Definition 1.6 (instantiation)**.** An *instantiation* $\overline{f}$ from I to J, written $f :: I \to J$, where $I = \langle m, \vec{X} \rangle$ and $J = \langle n, \vec{Y} \rangle$ are local, is induced by an underlying function $f : n \to m$. We define the instantiation

$$\overline{f} : \vec{X} \to \vec{Y}$$

in the following way. For interfaces we have $\overline{f}(I) = \vec{Y}$, where $Y_j \overset{def}{=} X_{f(j)}$ for all $j \in n$. For a ground bigraph $a : \vec{X}$ with no links crossing regions, we know that we can factorise it uniquely as

$$a = c_0 \parallel \cdots \parallel c_{m-1} \ \text{, with } c_i : X_i \ .$$

Let $d_j \simeq c_{f(j)}$ for $j \in n$ have disjoint supports, we then define the instantiation of a local bigraph as

$$\overline{f}(a) : \vec{Y} \overset{def}{=} d_0 \,||\, \cdots \,||\, d_{n-1} \ .$$

Parametric reaction rules allow for the rules to contain parameters, that can be replicated, discarded, or just moved.

Definition 1.7 (parametric reaction rule)**.** A *parametric reaction rule* has a *redex R* and *reactum R′*, and takes the form

$$(R : I \to K, R' : I' \to K, f)$$

with the inner faces $I = \langle m, \vec{X} \rangle$ and $I' = \langle m', \vec{X}' \rangle$ and $f : m' \to m$ is a map of ordinals, inducing the instantiation $\overline{f}$. For every parameter $d : I$ the parametric reaction rule generates a ground reaction rule on the form

$$(R \circ d, R' \circ \overline{f}(d)) \ \text{, where } \overline{f}(d) : I'.$$

2 Higher-Order Mobile Embedded Resources

In this section we present two variants of the calculus of Higher-Order Mobile Embedded Resources (Homer), a non-linear, pure higher-order process calculi with local names and named, nested locations. The first variant of Homer contains explicitly typed locations and send prefixes instead of the process constructor, *interface extension*, $\{n\}p$ that was introduced in [HGB04]. The interface extension operator, $\{n\}p$, extended the free names of p with the (possible idle) name n. The solution taken in this paper is that we explicitly annotate every location (and send prefix) with a set of names that must include the free names of the process contained in the prefix. Furthermore, we will only consider relations that relate processes with the same top-level type. The reason why we need to be able to type sub-locations, as well, is that interface extension cannot be extended vertically through a location barrier, as explained in the introduction.

In the second variant of Homer we also add explicit substitutions to the calculus, resembling the approach taken by Milner in [Mil04c]. This variant of Homer serves as an intermediate step between the traditional Homer calculus and our presentation as a bigraphical reactive system, since we need the explicit substitutions to represent the higher-order processes-passing of Homer.

2.1 Syntax and notation

We assume an infinite set of *names* $\mathcal{N}$ ranged over by m and n, and let $\tilde{n}$ range over finite sets of names. We let γ range over (possibly empty) sequences of names, and let δ range over non-empty sequences of names, referred to as *addresses* and let $|\delta|$ denote the length of the path δ, also we let $\varphi ::= \delta \mid \overline{\delta}$. We assume an infinite set of *process variables* $\mathcal{V}$ ranged over by x and y, and let $\tilde{x}$ range over finite sets

Processes:			
$p,q,r ::=$		$\mathbf{0}$	inactive process
	$\mid$	$\pi\,.\,p$	action prefixing
	$\mid$	$p \mid q$	parallel composition
	$\mid$	$(n)p$	let n be local in p
	$\mid$	x	process variable
Prefixes:			
$\pi ::=$		$\delta(x)$	receive a resource at δ and bind it to x
	$\mid$	$\overline{\delta}(x)$	take computing resource from δ and bind it to x
	$\mid$	$\overline{\delta}\langle r\rangle_{\tilde{n}}$	send a passive resource r having type $\tilde{n}$ to δ
	$\mid$	$\delta[r]_{\tilde{n}}$	computing resource r at location δ having type $\tilde{n}$

Table 1: Higher-Order Mobile Embedded Resources

of variables. The set $\mathcal{P}$ of *process expressions* is then defined by the grammar in Table 1.

The processes constructors are the usual process constructors from higher-order concurrent process calculi. As usual, we let the restriction operator (n) bind the name n and the prefixes $\varphi(x)$ bind the variable x. Note that the restriction operator also can bind the names that occur in a type annotation.

The prefix $\delta(x)$ represents the possibility to receive a passive resource sent from a local processes or a processes in a parent-location, whereas the prefix $\overline{\delta}(x)$ represents the possibility to take an active resource from a local location or a sub-location. The prefixes $\overline{\delta}\langle r\rangle_{\tilde{n}}$ and $\delta[r]_{\tilde{n}}$ are responsible for sending a *passive* resource r locally (or down) to the address δ and providing an *active* resource r locally (or up) on the location δ, respectively. In both cases we explicitly annotate the prefix with a set containing the free names of the resource. The prefixes $\overline{\delta}\langle r\rangle_{\tilde{n}}$ and $\delta(x)$ are the usual prefixes of Plain CHOCS [Tho93], except that we allow sequences of names as addresses instead of only a name. The prefixes $\delta[r]_{\tilde{n}}$ and $\overline{\delta}(x)$ are responsible for adding active process mobility to the calculus, as explained in the introduction.

We define the free names and free variables in Definition 2.1. Note that we define the free names of the prefixes $\varphi[\langle r\rangle]_{\tilde{n}}$ as $fn(\varphi[\langle r\rangle]_{\tilde{n}}) = fn(\varphi) \cup \tilde{n}$, so the type annotation of a send or a location prefix determines the free names of the resource in the prefix, under the assumption that the type annotation contains the free names of the resource. We will throughout the paper tacitly assume that this requirement is satisfied.

Definition 2.1 (free names and variables)**.** We define the sets $fn(p)$ and $fv(p)$ of *free*

$$\frac{}{\tilde{x} \vdash \mathbf{0} : \tilde{n}} \qquad \frac{\tilde{x} \vdash p : \tilde{n}_1 \qquad \tilde{x} \vdash q : \tilde{n}_2}{\tilde{x} \vdash p \mid q : \tilde{n}_1 \cup \tilde{n}_2}$$

$$\frac{}{\tilde{x}x \vdash x : \tilde{n}} \qquad \frac{\tilde{x} \vdash p : \tilde{n}n}{\tilde{x} \vdash (n)p : \tilde{n}}$$

$$\frac{\tilde{x}x \vdash p : \tilde{n}}{\tilde{x} \vdash \varphi(x) \,.\, p : \tilde{n} \cup fn(\varphi)} \qquad \frac{\tilde{x} \vdash r : \tilde{m} \qquad \tilde{x} \vdash p : \tilde{n}}{\tilde{x} \vdash \varphi[\langle r \rangle]_{\tilde{m}} \,.\, p : \tilde{m} \cup \tilde{n} \cup fn(\varphi)}$$

Table 2: Typing rules for Homer

names and *free variables* of p inductively in the structure of p.

Free names			Free Variables		
$fn(\mathbf{0})$	$=$	$\emptyset$	$fv(\mathbf{0})$	$=$	$\emptyset$
$fn(\varphi[\langle r \rangle]_{\tilde{n}} \,.\, p)$	$=$	$fn(\varphi) \cup \tilde{n} \cup fn(p)$	$fv(\varphi[\langle r \rangle]_{\tilde{n}} \,.\, p)$	$=$	$fv(r) \cup fv(p)$
$fn(\varphi(x) \,.\, p)$	$=$	$fn(\varphi) \cup fn(p)$	$fv(\varphi(x) \,.\, p)$	$=$	$fv(p) \setminus \{x\}$
$fn(p \mid q)$	$=$	$fn(p) \cup fn(q)$	$fv(p \mid q)$	$=$	$fv(p) \cup fv(q)$
$fn((n)p)$	$=$	$fn(p) \setminus \{n\}$	$fv((n)p)$	$=$	$fv(p)$
$fn(x)$	$=$	$\emptyset$	$fv(x)$	$=$	x

The sets $bn(p)$ and $bv(p)$ of *bound names* and *bound variables* are defined according as usual.

We define capture-free substitution in usual manner, though with the proper update of type annotation.

Definition 2.2 (substitutions)**.** We define the process $p[q : \tilde{n}/x]$ to be p with all free occurrences of x replaced by q of type $\tilde{n}$, where we have changed the annotations of all sub-terms $\varphi[\langle r \rangle]_{\tilde{m}}$ in p to $\varphi[\langle r \rangle]_{\tilde{m} \cup \tilde{n}}$, if and only if r contains a free occurrence of x, and if necessary α-converting p such that no free names and variables in q are bound.

As mentioned in the beginning of this section, compared to [HGB04] we have removed the process constructor $\{n\}p$ that extends the set of free names of a process p with the (possibly idle) name n and instead added type annotations at every location and send prefix.

Definition 2.3 (well-typed process)**.** We define the valid typing judgements of the form $\tilde{x} \vdash p : \tilde{n}$ inductive by the rules in Table 2.

From now on we will only consider well-typed processes. Note that a process p is well-typed with respect to a finite set of variables $\tilde{x}$ and names $\tilde{n}$, written $\tilde{x} \vdash p : \tilde{n}$, if and only if the free names (variables) of p are included in the set $\tilde{n}$ ($\tilde{x}$), and for every sub-term $\varphi[\langle r\rangle]_{\tilde{m}}$ and $q[x := r : \tilde{m}]$ in p we have that r can be typed with the type $\tilde{m}$. We will say that the annotations in a process are *valid* if for all sub-terms $\varphi[\langle r\rangle]_{\tilde{m}}$ it is the case that $\tilde{m} \supseteq fn(r)$. We say that a process with no free variables is *closed* and let $\mathcal{P}_c$ denote the set of closed processes. We write $p \equiv_\alpha q$, if p and q are α-convertible (both with respect to names and variables), we let $\mathcal{P}_{/\alpha}$ (and $\mathcal{P}_{c/\alpha}$) denote the set of α-equivalence classes of (closed) process expressions, and we consider processes up to α-equivalence.

Notation. We omit trailing **0**s, and hence write π instead of $\pi\,.\,\mathbf{0}$. We write $\vdash p : \tilde{n}$ for $\emptyset \vdash p : \tilde{n}$, and we let prefixing and restriction be right associative and bind stronger than parallel composition, hence writing e.g. $\pi\,.\,p \mid (n)q \mid r$ instead of $(\pi\,.\,p) \mid ((n)q) \mid r$. For a set of names $\tilde{n} = \{n_1, \dots, n_k\}$ we let $(\tilde{n})p$ denote $(n_1)\cdots(n_k)p$. We write $\tilde{m}\tilde{n}$ for $\tilde{m} \cup \tilde{n}$, always implicitly assuming $\tilde{m} \cap \tilde{n} = \emptyset$.

2.2 Reaction Semantics

We provide Homer with a reaction semantics defined in the Chemical Abstract Machine [BB90] style using structural congruence, evaluation contexts, and reaction rules.

Definition 2.4 (contexts and congruence)**.** A *context* $\mathcal{C}$ is defined by taking the grammar defined in Table 1 and augmenting the production of process expressions to also contain a special symbol called a *hole*

$$\mathcal{C} ::= \ \dots \ \mid \ (-)_{\tilde{n}}$$

and by requiring that the hole only occur once in the term. We annotate the hole with a type, meaning we can only place a process with type $\tilde{n}$ into the hole $(-)_{\tilde{n}}$. We write $\mathcal{C}(p)$ for the insertion of p into the hole of the context $\mathcal{C}$, assuming that the hole in $\mathcal{C}$ is annotated with the type $\tilde{n}$ and we have $\vdash p : \tilde{n}$. We extend $fn()$ to contexts by $fn(\mathcal{C}) = fn(\mathcal{C}(\mathbf{0}))$, and we extend $fv()$ accordingly. For typing contexts, we add the following rule to the typing rules of Table 2.

$$\frac{}{\tilde{x} \vdash (-)_{\tilde{n}} : \tilde{n}}$$

A binary relation $\mathcal{R}$ on well-typed processes is called *well-typed* if and only if it relates processes p and q with the same type $\tilde{n}$ ($\tilde{x}$), written $\tilde{x} \vdash p \mathrel{\mathcal{R}} q : \tilde{n}$. We will only consider well-typed relations in this paper. A relation $\mathcal{R}$ is called a *congruence* if and only if it is a well-typed equivalence relation and it satisfies that $\tilde{x} \vdash p \mathrel{\mathcal{R}} q : \tilde{n}$ implies $\tilde{x}' \vdash \mathcal{C}(p) \mathrel{\mathcal{R}} \mathcal{C}(q) : \tilde{n}'$ for all contexts $\mathcal{C}$, where the hole is annotated with the type $\tilde{n}$ and the type of the context is $\tilde{n}'$.

Structural congruence $\equiv$ is defined as the least congruence on well-typed processes relating $\tilde{x} \vdash p \equiv q : \tilde{n}$, if $\tilde{x} \vdash p : \tilde{n}$, $\tilde{x} \vdash q : \tilde{n}$, and $p \equiv q$ can be derived using the rules in Table 3, as structural congruence does not affect the typing of a process. The first row of the equations express that $(P, |, \mathbf{0})$ is a commutative monoid, the next two rows enforce the rules of scope of name restriction.

$$p \mid \mathbf{0} \equiv p \qquad (p \mid p') \mid p'' \equiv p \mid (p' \mid p'') \qquad p \mid q \equiv q \mid p$$
$$(n)p \mid q \equiv (n)(p \mid q),\ \text{if } n \notin fn(q) \qquad \pi\,.\,(n)p \equiv (n)\pi\,.\,p,\ \text{if } n \notin fn(\pi)$$
$$(n)(m)p \equiv (m)(n)p \qquad (n)p \equiv p,\ \text{if } n \notin fn(p)$$

Table 3: Structural congruence

As Homer permits reactions arbitrarily deep in the location hierarchy and also permits reactions between a process and an arbitrarily deeply nested sub-resource, we define in Definition 2.5 the concepts of evaluation and path contexts.

Definition 2.5 (evaluation contexts and path contexts)**.** An *evaluation context* $\mathcal{E}$ is a context with no free variables and whose hole is not guarded by a prefix, nor does it occur as the object of a send prefix. We define evaluation contexts by the following grammar

$$\mathcal{E} ::= (-)_{\tilde{n}} \ \mid\ \mathcal{E} \mid p \ \mid\ (n)\mathcal{E} \ \mid\ \delta[\mathcal{E}]_{\tilde{n}}\,.\,p,\ \text{for } p \in \mathcal{P}_c\ \ .$$

We define a family of multi-hole *path contexts* $\mathcal{C}^{\tilde{n}}_{\gamma}$, indexed by a path address $\gamma \in \mathcal{N}^*$ and a set of names $\tilde{n}$, inductively in $\tilde{n}$ and γ

$$\begin{array}{lcl} \mathcal{C}^{\emptyset}_{\varepsilon} & ::= & (-)_{\tilde{n}} \\ \mathcal{C}^{\tilde{n}\tilde{m}}_{\delta\gamma} & ::= & \delta[(\tilde{n})(\mathcal{C}^{\tilde{m}}_{\gamma} \mid (-)_{\tilde{n}'})]_{\tilde{m}'}\,.\,(-)_{\tilde{m}''}, \end{array}$$

whenever $\tilde{n} \cap \gamma = \emptyset$.

Remark. Note that the evaluation context $\delta[\mathcal{E}]_{\tilde{n}}\,.\,p$ enables internal reactions of active resources, and that for a path context $\mathcal{C}^{\tilde{n}}_{\gamma}$, the path address γ indicates the path under which the first hole of the context is found, and the set of names $\tilde{n}$ indicates the bound names of the hole. The side condition in the definition of path contexts ensures that none of the names in the path address of the hole are bound. The bound names $(\tilde{n})$ in the definition of path contexts are needed since the structural congruence does not permit vertical scope extension, as described in the introduction.

We handle the vertical scope extension and the update of type annotations of a location using an *open* operator, defined on path contexts.

Definition 2.6 (open operator on path contexts)**.** We define an *open* operator on path contexts $\tilde{m} \odot \mathcal{C}^{\tilde{n}}_{\gamma}$ inductively by:

$$\begin{array}{lcl} \tilde{m} \odot \mathcal{C}^{\emptyset}_{\varepsilon} & = & \mathcal{C}^{\emptyset}_{\varepsilon} \\ \tilde{m} \odot \mathcal{C}^{\tilde{n}_1\tilde{n}_2}_{\delta\gamma} & = & \delta[(\tilde{n}_1 \setminus \tilde{m})(\tilde{m} \odot \mathcal{C}^{\tilde{n}_2}_{\gamma} \mid (-)_{\tilde{n}'})]_{\tilde{m}'\cup\tilde{m}}\,.\,(-)_{\tilde{m}''}\ \ , \end{array}$$

if $\mathcal{C}^{\tilde{n}_1\tilde{n}_2}_{\delta\gamma} = \delta[(\tilde{n}_1)(\mathcal{C}^{\tilde{n}_2}_{\gamma} \mid (-)_{\tilde{n}'})]_{\tilde{m}'}\,.\,(-)_{\tilde{m}''}$ and $\tilde{m} \cap \tilde{n}_1\tilde{n}_2 \cap fn(\mathcal{C}^{\tilde{n}_1\tilde{n}_2}_{\delta\gamma}) = \emptyset$.

$$
\begin{array}{ll}
(\mathit{send}) & \vdash \overline{\gamma\delta}\langle r\rangle_{\tilde{n}} \,.\, q \mid \mathcal{C}^{\tilde{m}}_{\gamma}(\delta(x) \,.\, p, \vec{p}) \;\searrow\; q \mid \tilde{n} \odot \mathcal{C}^{\tilde{m}}_{\gamma}(p[r : \tilde{n}/x], \vec{p}) : \tilde{n}' \\
& \text{if } \tilde{m} \cap (\delta \cup \tilde{n}) = \emptyset \\
(\mathit{take}) & \vdash \mathcal{C}^{\tilde{m}}_{\gamma}(\delta[r]_{\tilde{n}} \,.\, q, \vec{p}) \mid \overline{\gamma\delta}(x) \,.\, p \;\searrow\; (\tilde{n} \cap \tilde{m})\big(\tilde{n} \odot \mathcal{C}^{\tilde{m}}_{\gamma}(q, \vec{p}) \mid p[r : \tilde{n}/x]\big) : \tilde{n}' \\
& \text{if } \tilde{m} \cap (\delta \cup \mathit{fn}(p)) = \emptyset
\end{array}
$$

Table 4: Reaction rules for Homer

Intuitively, the open operator in $\tilde{m} \odot \mathcal{C}^{\tilde{n}}_{\gamma}$ removes the names $\tilde{m}$ from the bound names of the hole and adds them to the type annotations of the locations that are part of the address path. When applied in the reaction rule, the latter condition of the open operator can always be met by α-conversion, the condition ensures us that we can extend the scope by using the open operator and place the restriction at top level, without any name captures.

As for the structural congruence, we define the reaction relation for Homer, written $\searrow$, as the least well-typed binary relation between well-typed processes satisfying the rules in Table 4 and closed under all evaluation contexts $\mathcal{E}$ and structural congruence. The rules are essentially the reaction rules of [HGB04] altered to use type annotations instead of the free name constructor.

Remark. The (*send*) rule expresses how a passive resource r is sent (down) to the (sub) location γ, where it is received at the address δ and is substituted in for x in p, possibly in several copies, updating the type annotations as necessary. The side conditions ensure that the location path is not bound in the context and that no free names of r get bound during movement. Note that the open operator only extends the type annotations of the locations constituting the location path and does not lift any restrictions, since $\tilde{m} \cap \tilde{n} = \emptyset$.

The (*take*) rule captures that a computing resource r is taken from the (sub) location γ, where it is running at the address δ, and is substituted in for x in p, possibly in several copies. Again, the side conditions ensure that the location path is not bound in the context, and that no free name is bound, when we lift the restriction. In this rule it is possible that the open operator both lifts restrictions and extends the type annotation of the locations.

In both rules we use multi-hole path contexts. The last k holes in a $k+1$-hole path context can be filled with an arbitrary process, written as the vector $\vec{p}$, since the reaction rules only affect the first hole in the path context. The types ensure that no names can disappear from the free names of a location, a send prefix, or from top-level during reaction. However, note that locations or send prefixes in the process that receives the moved resource r can get their type annotation extended by the type of r that do not already appear in their annotation.

2.3 Homer with explicit substitutions

In this subsection we present a variant of the Homer calculus, called Homerσ, where we have introduced explicit substitutions in the syntax instead of the meta-notion $p[q:\tilde{n}/x]$. We augment the grammar of Homer presented in Table 1 with an explicit syntactic substitution $p[x := q:\tilde{n}]$, representing the processes p in a context that can substitute q (of type $\tilde{n}$) in for x. The typing rule for explicit substitution, defined below, ensures that q is closed and that the free names of q are contained in $\tilde{n}$.

Definition 2.7 (Homer with explicit substitutions)**.** We augment the grammar in Table 1 with an explicit substitution

$$p,q,r ::= \ \dots \ \mid \ p[x := q:\tilde{n}] \ .$$

We augment Definition 2.1 of free names and variables as follows

Free names			Free Variables		
$fn(p[x:=q:\tilde{n}])$	$=$	$fn(p)\cup\tilde{n}$	$fv(p[x:=q:\tilde{n}])$	$=$	$(fv(p)\setminus\{x\})\cup fv(q)$

For the typing judgement $\tilde{x}\vdash p:\tilde{n}$ we add the following rule

$$\frac{\tilde{x}x\vdash p:\tilde{n} \qquad \vdash q:\tilde{m}}{\tilde{x}\vdash p[x:=q:\tilde{m}]:\tilde{n}\cup\tilde{m}} \ .$$

We let $\mathcal{P}\sigma_c$ denote the set of closed Homerσ processes and let $\mathcal{P}\sigma_{/\alpha}$ (and $\mathcal{P}\sigma_{c/\alpha}$) denote the set of α-equivalence classes of (closed) Homerσ process expressions, and again we consider processes up to α-equivalence.

Remark. Note that the explicit substitution $p[x:=r:\tilde{n}]$ binds x in p. We let prefixing and restriction be right associative and bind stronger than explicit substitution and let explicit substitution bind stronger than parallel composition hence writing e.g. $\pi\,.\,p[x:=r:\tilde{n}]$ instead of $(\pi\,.\,p)[x:=r:\tilde{n}]$.

Definition 2.8 (contexts, structural congruence, and evaluation contexts)**.** We define contexts for Homerσ by augmenting the grammar of Definition 2.7 with a hole

$$\mathcal{C} ::= \ \dots \ \mid \ (-)_{\tilde{n}}$$

and by requiring that it only appear once in the entire term. As for Homer we require that only processes of type $\tilde{n}$ are placed into the hole $(-)_{\tilde{n}}$. We define structural congruence $\equiv_\sigma$ for Homerσ by extending the rules of Table 3 with the following rule

$$(n)(p[x:=r:\tilde{n}]) \equiv_\sigma (n)p[x:=r:\tilde{n}], \text{ if } n\notin\tilde{n} \ .$$

One might expect a rule stating that we can move restriction under the object of an explicit substitution as

$$(n)(p[x:=r:\tilde{n}]) \equiv_\sigma p[x:=(n)r:\tilde{n}\setminus\{n\}], \ \text{ if } n\notin fn(p) \ .$$

$(send\sigma)$	$\vdash \overline{\gamma\delta}\langle r\rangle_{\tilde{n}} \,.\, q \mid \mathcal{C}^{\tilde{m}}_{\gamma}(\delta(x)\,.\,p,\vec{p}) \searrow_{\sigma} q \mid \tilde{n} \odot \mathcal{C}^{\tilde{m}}_{\gamma}(p[x := r : \tilde{n}],\vec{p}) : \tilde{n}'$, if $\tilde{m} \cap (\delta \cup \tilde{n}) = \emptyset$
$(take\sigma)$	$\vdash \mathcal{C}^{\tilde{m}}_{\gamma}(\delta[r]_{\tilde{n}}\,.\,q,\vec{p}) \mid \overline{\gamma\delta}(x)\,.\,p \searrow_{\sigma} (\tilde{n} \cap \tilde{m})\big(\tilde{n} \odot \mathcal{C}^{\tilde{m}}_{\gamma}(q,\vec{p}) \mid p[x := r : \tilde{n}]\big) : \tilde{n}'$, if $\tilde{m} \cap (\delta \cup fn(p)) = \emptyset$
$(apply\sigma)$	$\vdash \mathcal{C}(x)[x := r : \tilde{n}] \searrow_{\sigma} \tilde{n} \odot \mathcal{C}(r)[x := r : \tilde{n}] : \tilde{n}'$, if $\mathcal{C}$ does not bind x or the names in $\tilde{n}$
$(garbage\sigma)$	$\vdash p[x := q : \tilde{n}] \searrow_{\sigma} p : \tilde{n}'$, where $x \notin fv(p)$

Table 5: Reaction rules for Homerσ

But for the same reasons as mentioned in the introduction this is not sound. The definition of evaluation contexts and path contexts remains the same for Homerσ as for Homer, and hence the open operator remains unchanged. We then define the reaction relation $\searrow_{\sigma}$ for Homerσ as the least binary well-typed relation on $\mathcal{P}\sigma_{c/\alpha}$ satisfying the rules in Table 5 and closed under evaluation contexts and structural congruence $\equiv_{\sigma}$.

Remark. Compared to Table 4 we have made the following changes to the reaction relation. We have added a syntactic explicit substitution $p[x := r : \tilde{n}]$ instead of the substitution $p[r : \tilde{n}/x]$, and we have added rules for applying and garbage collecting an explicit substitution. The rules $(send\sigma)$ and $(take\sigma)$ mimic the rules $(send)$ and $(take)$ of Homer, respectively. The only difference is that we have replaced the substitution $p[r : \tilde{n}/x]$ with $p[x := r : \tilde{n}]$. The rule $(apply\sigma)$ replaces one occurrence of the variable (arbitrarily deep in the context) with the content of the explicit substitution. Note that we have overloaded the use of $\odot$, since we in $(apply\sigma)$ apply the operator to a general context and not only a path context. However, the result of the operator is the same, it extends the type annotations of all the locations (and send prefix) containing this occurrence of the variable. The latter condition of the rule can always be satisfied using α-conversion of the context. The $(garbage\sigma)$ rule is responsible for garbage collecting superfluous substitutions.

2.4 Connection between the two variants

In this subsection we relate the two presented variants of Homer calculus: Homer and Homerσ. We will in the following subsection sometimes use a Homer process to denote the Homerσ process with the same syntax, as the set of expressions of Homer, $\mathcal{P}_{c/\alpha}$, is a subset of the set of process expressions of Homerσ, $\mathcal{P}\sigma_{c/\alpha}$ (i.e. the subset of Homerσ processes that has no sub-terms of the form $p[x := r : \tilde{n}]$).

For one direction in the correspondence we use that an explicit substitution $p[x := r : \tilde{n}]$ can react (in Homerσ) to become the process that arises from the meta-notation

$p[r : \tilde{n}/x]$ by utilising the reaction rules (*apply*σ) and (*garbage*σ).

Proposition 2.9. *If p and r are Homer processes and r is closed, then $\vdash p[x := r : \tilde{n}] \searrow_\sigma^* p[r : \tilde{n}/x] : \tilde{n}$.*

Lemma 2.10. *If $\vdash p \searrow q : \tilde{n}$ then $\vdash p \searrow_\sigma^* q : \tilde{n}$.*

Proof. There are only two rules for inferring a reaction in Homer: (*send*) and (*take*). In both cases the result follows by choosing the matching rule in Homerσ ((*send*σ) or (*take*σ)) and applying Proposition 2.9. □

For the other direction in the correspondence we need the following function

$$app : \mathcal{P}\sigma_{c/\alpha} \to \mathcal{P}_{c/\alpha} \ , \tag{16}$$

which takes a Homerσ process p and applies and garbage collects all the explicit substitutions in p. Note that the crucial property of *app* is the following

$$app(p[x := r : \tilde{n}]) = app(p[r : \tilde{n}/x]) \ .$$

We also need the following proposition stating that the function *app* preserves redexes for the rules (*send*σ) and (*take*σ).

Proposition 2.11. *Assume that p is a Homerσ process and it contains a (sendσ) or a (takeσ) redex, then the Homer process $app(p)$ also have this redex.*

Proposition 2.12. *If $\vdash p \searrow_\sigma q : \tilde{n}$ then $\vdash app(p) \equiv app(q) : \tilde{n}$ or $\vdash app(p) \searrow app(q) : \tilde{n}$.*

Proof. We consider each of the four rules defining $\searrow_\sigma$ in turn.

- (*send*σ) Assume that $\vdash p \searrow_\sigma q : \tilde{n}$ because of the (*send*σ) rule, then $\vdash app(p) \searrow app(q) : \tilde{n}$ using the (*send*) rule, since we know from Proposition 2.11 that *app* preserves the redex and since the rules (*send*σ) and (*send*) only differ in the substitution and *app* gives the same result on this.

- (*take*σ) Similar to the case for (*send*σ).

- (*apply*σ) Assume that $\vdash p \searrow_\sigma q : \tilde{n}$ because of the (*apply*σ) rule, then $\vdash app(p) \equiv app(q) : \tilde{n}$, as the only difference between p and q is that we have applied one of the explicit substitutions to one occurrence of the bound variable, and since *app* will apply and garbage collect all the explicit substitutions.

- (*garbage*σ) Assume that $\vdash p \searrow_\sigma q : \tilde{n}$ because of the (*garbage*σ) rule, then $\vdash app(p) \equiv app(q) : \tilde{n}$, as the explicit substitution garbage collected by (*garbage*σ) is also garbage collected by *app*. □

3 Bigraphical Semantics of Homerσ

In this section we give the bigraphical presentation of Homerσ as the bigraphical reactive system´ Homerσ. First, we present the signature for´ Homerσ, and give a fully compositional translation of Homerσ-terms into bigraphs. Second, we translate the path contexts and the reaction relation $\searrow_{\sigma}$. An important criteria for the presentation is to show that there is a static and operational correspondence between Homerσ and its presentation as a bigraphical reactive system, meaning that structural congruence of Homerσ corresponds to graph isomorphism in the bigraphical presentation, and that reactions match.

Intuitively, we define a signature which has one control for each syntactic constructor in the Homerσ-terms, and use local links and nesting of controls to represent the structure of the abstract syntax tree of a given Homerσ-term. We have chosen to represent the path addresses of the respective prefixes with one port for each element in the sequence. Hence, for each kind of prefix we have an infinite family of controls indexed by the length of the address. To avoid this infinite number of controls, we could represent the sequences of names as nested nodes of a certain kind of control, each with one port connected to an outer name and let nesting of the nodes express the sequencing, but to keep the presentation succinct we have chosen to elide this option.

The signature has controls **rece** and **take** representing the two input prefixes, and **send** and **loca** representing the two kinds (passive and active) of located resources. We use **resi** (shorthand for **residual**) to hinder reactions behind prefixes. Controls **var**, **sub**, and **def** represent a variable and the constructs for explicit substitutions, respectively. Finally, the signature also has controls **tname** (abbreviation for **typename**) and **ann** (abbreviation for **annotation**) to represent the explicit type annotation of resource and send prefixes. We will discuss this in more detail after having presented the reaction rules in the bigraphical framework. In total, the signature for´ Homerσ is defined as follows.

Definition 3.1 (´ Homerσ signature)**.** The signature for´ Homerσ has an infinite number of controls.

- The controls **var**: $0 \to 1$ and **tname**: $0 \to 1$ are atomic
- The families of controls: **rece**$_{|\delta|}$: $1 \to |\delta|$, **take**$_{|\delta|}$: $1 \to |\delta|$, and **send**$_{|\delta|}$: $0 \to |\delta|$ are all inactive
- The family of controls **loca**$_{|\delta|}$: $0 \to |\delta|$ is active
- The controls **resi**: $0 \to 0$, **def**: $0 \to 1$, **sub**: $1 \to 0$, and **ann**: $0 \to 0$ are inactive

Remark. Note that we cannot represent restriction using an enclosing control, since this would break the static correspondence, as stated in Theorem 3.7. For instance the rule $(n)(m)p \equiv_{\sigma} (m)(n)p$ would be problematic, since the place graph is defined as a forest. We will examine this and an alternative definition of restriction in Section 4.2.

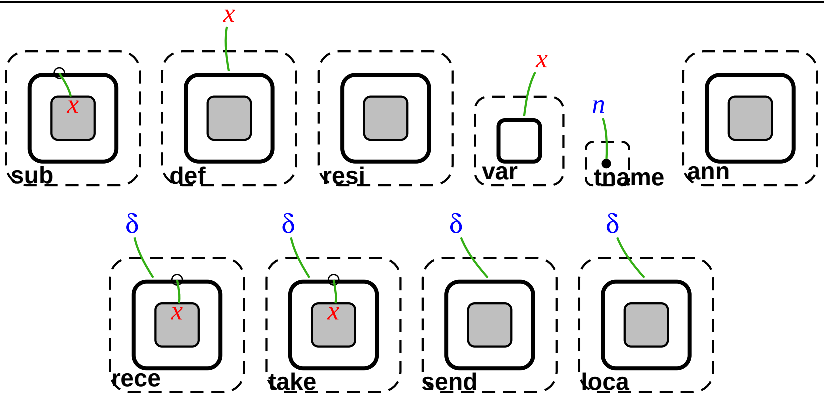

Figure 2: Ions and atoms for´Homerσ

For the same reasons, we do not introduce a control representing the inactive process **0** as the rule $p \mid \mathbf{0} \equiv_\sigma p$ would also break the static correspondence.

Notation. In Figure 2 we depict the ions and the atoms used in the translation in Definition 3.4. As usual we do not depict the arbitrary names that can be exported from the ions, we have also chosen to depict the control **tname** as just a dot, •, in order to be able to distinguish graphically between **tname** and **var** controls.

Following the convention of Milner [Mil04c], we write $\mathbf{var}_x$ and $\mathbf{tname}_n$ for the atoms, and we denote the ions as follows

$$\mathbf{sub}_{(x)} \,\overline{\oplus}\, \mathbf{id}_Z \qquad \mathbf{def}_x \,\overline{\oplus}\, \mathbf{id}_Z \qquad \mathbf{resi} \,\overline{\oplus}\, \mathbf{id}_Z \qquad \mathbf{ann} \,\overline{\oplus}\, \mathbf{id}_Z$$
$$\mathbf{rece}_{\delta(x)} \,\overline{\oplus}\, \mathbf{id}_Z \qquad \mathbf{take}_{\delta(x)} \,\overline{\oplus}\, \mathbf{id}_Z \qquad \mathbf{send}_\delta \,\overline{\oplus}\, \mathbf{id}_Z \qquad \mathbf{loca}_\delta \,\overline{\oplus}\, \mathbf{id}_Z$$

using the $\overline{\oplus}$ operator defined in Definition 3.2 below. Following the convention of [Mil04c], we write the binding port names in parenthesis and last.

We need to introduce a variant of the extension operator of local bigraphs in order to translate Homerσ-terms into bigraphs directly. The idea behind the extension operator, denoted $\oplus$, of [Mil04b] is that we can allow for a parameter to have more names than expected by a context, thus extending the inner face of the context. The insertion of the parameter into the context should then in the same sense extend the outer face of the context.

The problem with the extension operator, for our usage, is that it is only defined for $G \oplus \omega$, if the names in the outer and inner face of the bigraph G and the wiring ω are disjoint (and G and ω have equal width). However, in our presentation we need to be able to express an operation like $\oplus$, but where the outer faces can share names. To this end we define a derived operator, written $\overline{\oplus}$, based on the extension operator. We first define shared extension of outer interfaces, and then shared extension of bigraphs.

Definition 3.2 (Shared extension)**.** For two interfaces $J = \langle m, loc, X\rangle$ and $J' = \langle m, loc', X'\rangle$ of equal width, we define their shared extension, $J \overline{\oplus} J'$ as

$$J \overline{\oplus} J' \quad \overset{def}{=} \quad \langle m, loc \cup loc', X \cup X'\rangle \ .$$

Given a bigraph $G : I \to J$ and a wiring $\omega : I' \to J'$ of equal width and disjoint supports, where $J = \langle m, loc, X \uplus X''\rangle$ and $J' = \langle m, loc', X' \uplus X''\rangle$, and a *renaming*[1] $\alpha : X'' \to X'''$, where $X''' \cap (X \cup X') = \emptyset$, then if $I \oplus I'$ is defined, the *shared extension* of G by ω is defined as

$$G \overline{\oplus}\, \omega : I \oplus I' \to J \overline{\oplus} J' \quad \overset{def}{=} \quad \alpha^{-1} \circ (G \oplus ((\alpha \oplus \mathrm{id}_{X'}) \circ \omega)) \ ,$$

where α^{-1} is defined as the inverse of α extended with the identities on X, X', and X''.

So we define shared extension by first mapping one part of the shared names to a disjoint set of names, perform the extension, and then on top-level join the shared names with a substitution. The following proposition states that we can always find a bijective linking, so that shared extension is defined.

Proposition 3.3. *Given a bigraph $G : I \to J$ a wiring $\omega : I' \to J'$ satisfying the requirements of Definition 3.2, we can always find a renaming α also satisfying the requirements of Definition 3.2.*

We will only use a simple instance of the $\overline{\oplus}$ operator in the presentation, since we only need to extend a bigraph (of inner and outer width 1) with an identity wiring, hereby extending the inner and outer face of the bigraph. E.g. the ion $\mathbf{send}_\delta \overline{\oplus}\, \mathbf{id}_Z$ has Z as inner names and $Z \cup \delta$ as outer names.

3.1 The Translation

We have a fully compositional translation from Homerσ to bigraphs. The translation is defined inductively in the typing derivation of a process. The reason for this is that we need the typings for propagating the relevant links, representing names and variables, throughout the entire bigraph. As in the presentation in Section 2 we let m and n range over Homer names, let δ range over non-empty sequences of Homer names, and let x and y range over Homer variables. We will follow the convention and write $\tilde{n} \overline{\oplus} \tilde{x}$ for the bigraph with unit inner face and $\tilde{n} \overline{\oplus} \tilde{x}$ outer face.

Definition 3.4 (Translation of Homerσ-terms into bigraphs)**.** We define the translation of a well-typed Homerσ-term p into a bigraph inductively in the inference of

[1]a bijective substitution.

$\tilde{x} \vdash p : \tilde{n}$

$$\begin{aligned}
&\llbracket \tilde{x} \vdash \mathbf{0} : \tilde{n} \rrbracket && = \tilde{n} \mathbin{\overline{\oplus}} \tilde{x} \\
&\llbracket \tilde{x} \vdash p \mid q : \tilde{n}_1 \cup \tilde{n}_2 \rrbracket && = \llbracket \tilde{x} \vdash p : \tilde{n}_1 \rrbracket \mid \llbracket \tilde{x} \vdash q : \tilde{n}_2 \rrbracket \\
&\llbracket \tilde{x} \vdash (n)p : \tilde{n} \rrbracket && = /n \circ (\llbracket \tilde{x} \vdash p : \tilde{n}n \rrbracket) \\
&\llbracket \tilde{x}x \vdash x : \tilde{n} \rrbracket && = \textsf{\textbf{var}}_x \mathbin{\overline{\oplus}} \tilde{n} \mathbin{\overline{\oplus}} \tilde{x} \\
&\llbracket \tilde{x} \vdash p[x := r : \tilde{n}'] : \tilde{n} \cup \tilde{n}' \rrbracket && = (\textsf{\textbf{sub}}_{(x)} \mathbin{\overline{\oplus}} \textsf{\textbf{id}}_{\tilde{n} \cup \tilde{n}', \tilde{x}})(\llbracket \tilde{x}x \vdash p : \tilde{n} \rrbracket \mid \\
& && \qquad (\textsf{\textbf{def}}_x \mathbin{\overline{\oplus}} \textsf{\textbf{id}}_{\tilde{n}'})(\llbracket \vdash r : \tilde{n}' \rrbracket \mid (\textsf{\textbf{ann}} \mathbin{\overline{\oplus}} \textsf{\textbf{id}}_{\tilde{n}'}) \llbracket \tilde{n}' \rrbracket)) \\
&\llbracket \tilde{x} \vdash \delta[r]_{\tilde{n}'} \,.\, p : \tilde{n}' \cup \tilde{n} \cup fn(\delta) \rrbracket && = (\textsf{\textbf{loca}}_{\delta} \mathbin{\overline{\oplus}} \textsf{\textbf{id}}_{\tilde{n} \cup \tilde{n}', \tilde{x}})(\llbracket \tilde{x} \vdash r : \tilde{n}' \rrbracket \mid \\
& && \qquad ((\textsf{\textbf{ann}} \mathbin{\overline{\oplus}} \textsf{\textbf{id}}_{\tilde{n}'}) \llbracket \tilde{n}' \rrbracket) \mid (\textsf{\textbf{resi}} \mathbin{\overline{\oplus}} \textsf{\textbf{id}}_{\tilde{n}, \tilde{x}}) \llbracket \tilde{x} \vdash p : \tilde{n} \rrbracket) \\
&\llbracket \tilde{x} \vdash \overline{\delta}\langle r \rangle_{\tilde{n}'} \,.\, p : \tilde{n}' \cup \tilde{n} \cup fn(\delta) \rrbracket && = (\textsf{\textbf{send}}_{\delta} \mathbin{\overline{\oplus}} \textsf{\textbf{id}}_{\tilde{n} \cup \tilde{n}', \tilde{x}})(\llbracket \tilde{x} \vdash r : \tilde{n}' \rrbracket \mid \\
& && \qquad ((\textsf{\textbf{ann}} \mathbin{\overline{\oplus}} \textsf{\textbf{id}}_{\tilde{n}'}) \llbracket \tilde{n}' \rrbracket) \mid (\textsf{\textbf{resi}} \mathbin{\overline{\oplus}} \textsf{\textbf{id}}_{\tilde{n}, \tilde{x}}) \llbracket \tilde{x} \vdash p : \tilde{n} \rrbracket) \\
&\llbracket \tilde{x} \vdash \delta(x) \,.\, p : \tilde{n} \cup fn(\delta) \rrbracket && = (\textsf{\textbf{rece}}_{\delta(x)} \mathbin{\overline{\oplus}} \textsf{\textbf{id}}_{\tilde{n}, \tilde{x}}) \llbracket \tilde{x}x \vdash p : \tilde{n} \rrbracket \\
&\llbracket \tilde{x} \vdash \overline{\delta}(x) \,.\, p : \tilde{n} \cup fn(\delta) \rrbracket && = (\textsf{\textbf{take}}_{\delta(x)} \mathbin{\overline{\oplus}} \textsf{\textbf{id}}_{\tilde{n}, \tilde{x}}) \llbracket \tilde{x}x \vdash p : \tilde{n} \rrbracket
\end{aligned}$$

and we translate the type annotations as follows

$$\llbracket \tilde{n} \rrbracket \quad = \quad \mathop{\Big|}_{n \in \tilde{n}} \textsf{\textbf{tname}}_n \ .$$

Remark. We represent the inactive process **0** as an empty bigraph with the correct outer face. The parallel composition of Homerσ is represented using the corresponding operator in bigraphs, prime product. We use a closure $/n$, which closes the open link n, to represent the restriction of the name n. A variable is represented as a node of control **var** which is connect to the name x, which is disjoint from the set $\tilde{x}$.

We represent the explicit substitutions in Homerσ in the same way as [Mil04c]. However, note that the control **def** is inactive, since otherwise we could have reactions under prefix, and that we have augmented the explicit substitution with a type annotation. The translation also ensures that the name x is disjoint from the set $\tilde{x}$ containing the set of free variables.

The first two prefixes $\delta[r]_{\tilde{n}} \,.\, p$ and $\overline{\delta}\langle r \rangle_{\tilde{n}} \,.\, p$ are represented by a node of the matching control (indexed by the length of δ) containing the representation of the object of the prefix and the residual process enclosed in a node of control **resi**. We use the control **resi** to ensure that the residual processes after a prefix cannot make reactions. Actually, the control is superfluous in the representation of the send prefix, since the control **send** itself is inactive, but we have kept it to emphasise the connection between the two prefixes $\delta[r]_{\tilde{n}}$ and $\overline{\delta}\langle r \rangle_{\tilde{n}}$.

We have chosen to represent the type annotations as a set of **tname** nodes enclosed by a node of control **ann**. The last two prefixes $\delta(x) \,.\, p$ and $\overline{\delta}(x) \,.\, p$ are represented straightforwardly by a node of the respective control, where the variable x is bound in the enclosed representation of p. As for the representation of the explicit substitution we require that x and $\tilde{x}$ are disjoint. Finally note that we have decided

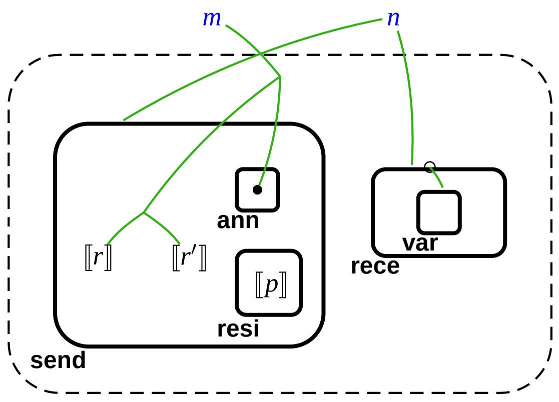

Figure 3: Example on translation of the term $\overline{n}\langle r \mid r' \rangle_{\{m\}} \,.\, p \mid n(x)\,.\,x$ into a bigraph

to use $\overline{\oplus}$ throughout the translation, even though we often compose bigraphs where the names are disjoint.

Example (translation of Homerσ-terms)**.** As an example on the translation from Homerσ-terms to bigraphs, we depict in Figure 3 the result of the translation of $\overline{n}\langle r \mid r' \rangle_{\{m\}} \,.\, p \mid n(x)\,.\,x$, where we for clarity have chosen not to draw the free names of p.

When presenting a calculi as a bigraphical reactive system we would like for the property that structural congruence of the calculi corresponds to graph isomorphism in the bigraphical representation.

Lemma 3.5. *$\tilde{x} \vdash p \equiv_\sigma q : \tilde{n}$ implies $[\![\tilde{x} \vdash p : \tilde{n}]\!] = [\![\tilde{x} \vdash q : \tilde{n}]\!]$.*

Proof. Since the translation is compositional we can consider each of the axioms defining $\equiv_\sigma$ separately. We only present some of the cases

- Each of the axioms

  $$\tilde{x} \vdash p \mid \mathbf{0} \equiv_\sigma p : \tilde{n} \quad \tilde{x} \vdash (p \mid p') \mid p'' \equiv_\sigma p \mid (p' \mid p'') : \tilde{n} \quad \tilde{x} \vdash p \mid q \equiv_\sigma q \mid p : \tilde{n}$$

  follows directly from the translation, since we translate parallel composition in Homerσ as the prime product in bigraphs '$\mid$', which can be shown to be associative and commutative, and as we translate $\mathbf{0}$ into the unit for $\mid$.

- To prove the case for the axiom for reordering of restrictions

  $$\tilde{x} \vdash (n)(m)p \equiv (m)(n)p : \tilde{n}$$

  we show that the two bigraphs $[\![\tilde{x} \vdash (n)(m)p : \tilde{n}]\!]$ and $[\![\tilde{x} \vdash (m)(n)p : \tilde{n}]\!]$ can be constructed in the same manner (we assume that m and n are distinct names of

p). We construct $[\![\tilde{x} \vdash p : \tilde{n}nm]\!]$ and add two edges to its link graph e_m and e_n and make all points of m (n) point to e_m (e_n). Finally we remove the names m and n.

- The axiom for scope extension

$$\tilde{x} \vdash (n)p \mid q \equiv_\sigma (n)(p \mid q) : \tilde{n}, \text{ if } n \notin fn(q)$$

can be proven in the same way. We construct the bigraphs $[\![\tilde{x} \vdash (n)p \mid q : \tilde{n}]\!]$ and $[\![\tilde{x} \vdash (n)(p \mid q) : \tilde{n}]\!]$ in the following way. Without loss of generality we assume that $\tilde{n} = \tilde{n}_1 \cup \tilde{n}_2$, where $\tilde{n}_1 n\tilde{x}$ and $\tilde{n}_2\tilde{x}$ are the names in the outer face of $[\![\tilde{x} \vdash p : \tilde{n}_1 n]\!]$ and $[\![\tilde{x} \vdash q : \tilde{n}_2]\!]$, respectively. First we build $[\![\tilde{x} \vdash p : \tilde{n}_1 n]\!]$ and $[\![\tilde{x} \vdash q : \tilde{n}_2]\!]$, and then we combine them using the prime product, add one edge e_n to the link graph of this bigraph, and make all points of the name n point to e_n. Since $n \notin fn(q)$ we only touch points in $[\![\tilde{x} \vdash p : \tilde{n}_1 n]\!]$. Finally we remove the name n.

- For the remaining cases we proceed in the same manner by exhibiting a constructing that forms both bigraphs. □

Proposition 3.6. *If $[\![\tilde{x} \vdash p : \tilde{n}]\!] = g$ and $[\![\tilde{x} \vdash q : \tilde{n}]\!] = g'$ and g and g' are isomorphic, then $\tilde{x} \vdash p \equiv_\sigma q : \tilde{n}$.*

Theorem 3.7 (Static correspondence)**.** *$\tilde{x} \vdash p \equiv_\sigma q : \tilde{n}$ if and only if $[\![\tilde{x} \vdash p : \tilde{n}]\!] = [\![\tilde{x} \vdash p : \tilde{n}]\!]$.*

3.2 Representing Path Contexts and the Open Operation

In order to present the reaction rules of Homerσ we need to be able to represent the path contexts and the open operation. In this subsection we describe how we translate the path contexts defined in Section 2.5. We do not need to represent the evaluation contexts of Homerσ, since these are inherent in the bigraphical setting, due to the specification of controls as being either active or inactive.

Definition 3.8 (path bigraphs)**.** We define the translation of a path context $\mathcal{C}_\gamma^{\tilde{n}}$ into a bigraph of a certain form, called a *path bigraph*, inductively in the structure of $\mathcal{C}_\gamma^{\tilde{n}}$ (using Definition 2.5)

$$\begin{aligned}
[\![\vdash \mathcal{C}_\varepsilon^{\emptyset} : \tilde{m}'']\!] &= \mathbf{id}_{\tilde{m}''} \\
[\![\vdash \mathcal{C}_{\delta\gamma}^{\tilde{n}\tilde{m}} : \tilde{m}'']\!] &= (\mathbf{loca}_\delta \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{m}''})(/\tilde{n} \circ ([\![\vdash \mathcal{C}_\gamma^{\tilde{m}} : \tilde{n}']\!] \mid \mathbf{id}_{\tilde{n}'}) \mid \\
&\qquad ((\mathbf{ann} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{m}'})[\![\tilde{m}']\!]) \mid (\mathbf{resi} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{m}''}))
\end{aligned}$$

if $\mathcal{C}_{\delta\gamma}^{\tilde{n}\tilde{m}} = \delta[(\tilde{n})(\mathcal{C}_\gamma^{\tilde{m}} \mid (-)_{\tilde{n}'})]_{\tilde{m}'} . (-)_{\tilde{m}''}$. We let F, F' range over path bigraphs. And as for Homerσ we will sometimes use subscript to denote the address of the hole and superscript to denote the bound names of the hole.

We also define an operation equivalent to the open operator in Homerσ.

Definition 3.9 (open operator on bigraphs)**.** We define an *open operator* on path bigraphs, $\tilde{m} \odot_b F$, extending the type annotations with $\tilde{m}$

$$\begin{array}{lcl} \tilde{m} \odot_b \mathbf{id}_{\tilde{m}''} & = & \mathbf{id}_{\tilde{m}'' \cup \tilde{m}} \\ \tilde{m} \odot_b F & = & (\mathbf{loca}_\delta \,\overline{\oplus}\, \mathbf{id}_{\tilde{m}'',\tilde{m}})(/(\tilde{n} \setminus \tilde{m}) \circ ((\tilde{m} \odot_b [\![\vdash C_\gamma^{\tilde{m}} : \tilde{n}']\!]) \mid \mathbf{id}_{\tilde{n}'}) \mid \\ & & \quad ((\mathbf{ann} \,\overline{\oplus}\, \mathbf{id}_{\tilde{m}',\tilde{m}})[\![\tilde{m}' \cup \tilde{m}]\!]) \mid (\mathbf{resi} \,\overline{\oplus}\, \mathbf{id}_{\tilde{m}''})) \end{array}$$

if $F = (\mathbf{loca}_\delta \,\overline{\oplus}\, \mathbf{id}_{\tilde{m}''})(/\tilde{n} \circ ([\![\vdash C_\gamma^{\tilde{m}} : \tilde{n}']\!] \mid \mathbf{id}_{\tilde{n}'}) \mid ((\mathbf{ann} \,\overline{\oplus}\, \mathbf{id}_{\tilde{m}'})[\![\tilde{m}']\!]) \mid (\mathbf{resi} \,\overline{\oplus}\, \mathbf{id}_{\tilde{m}''})$.

Remark. We cannot just juxtaposition the type annotations as $[\![\tilde{n}']\!] \mid [\![\tilde{m}]\!]$, since we represent the individual elements of a type annotation explicitly with one node per element in the annotation, since this would result in our annotations being multisets rather than sets. E.g. $[\![\tilde{n}']\!] \mid [\![\tilde{m}]\!] = [\![\tilde{n}' \cup \tilde{m}]\!]$ does not hold in general. So instead we only add the elements in $\tilde{m}$ that are not already present in the type annotation.

3.3 A Simple Sorting on´Homerσ

In this subsection we present a simple sorting to ensure that we only work with a subset of ground bigraphs, that is the bigraphs that are 'correct' with respect to our presentation of Homerσ. The sorting introduces a requirement on the possible nesting of nodes and on how the linkage is performed, particularly that the sets of free names and variables are kept disjoint.

But before stating the definition of the class of bigraphs that we are interested in we need some nomenclature to differentiate the different kinds of links and ports. We have two kinds of ports: name- and variable-ports.

- The *name-ports* are either the single port on a **tname** node or all the free ports of a **rece**, **take**, **send**, or a **loca** node.
- The *variable-ports* are the free port of a **def** node or a **var** node or the binding port of a **sub**, **rece**, or a **take** node.

In the same way we define two kinds of links:

- A *name-link* is a link with only name-ports, and if free a name.
- A *variable-link* is a link with only variable-ports connected to it, and if free a variable name.

Definition 3.10 (bigraphs good for Homerσ)**.** We define a sub-class I of ground bigraphs in´Homerσ as the bigraphs that satisfy the following requirements

- We only allow name- and variable-links as links in the bigraph.
- A variable-link can be connected to any number of **var**-ports.
  - If a variable-link is bound by either a **rece**-or a **take**-port, then it contains no **def**-ports.

  - If a variable-link is bound by a port on a **sub**-node v, then it also has one **def**-port, which resides on a child of v, and this is the only location where a **def** node can occur.

- A name-link can be connected to any number of name-ports.
- For every pair of distinct **tname** nodes enclosed in the same **ann** node their name-ports must be connected to distinct links.
- Every **loca**, **send**, and **def** node must contain a unique **ann** child node. Furthermore, **loca** and **send** nodes must contain a unique **resi** child node. And these are the only locations where **ann** and **resi** nodes can occur.
- All **tname** nodes must be in a **ann** node and no other kind of nodes can reside here.
- For every outer name of the parameter of a **loca**, **send**, or a **sub** node a **tname** node must exist that points to this name.

Remark. We have introduced all the abovementioned restrictions to enforce that we only work with bigraphs, that have a structure corresponding to how we interpret Homerσ in bigraphs. In Homerσ the sets of names and variables are by definition disjoint, but since we use the links of bigraphs to represent both sets, we need some additional requirements to enforce this distinction in kinds of links.

The requirements enforce that a **loca** node and a **send** node contains unique **resi** and **ann** nodes. We also require that **def** can only appear as a child of a **sub** node. Finally, we require that the **tname** nodes representing a type annotation only occur in a **ann** node, that they are unique in the sense that they all are linked to different name-links, and that they contain all free names of the parameter.

Proposition 3.11 (invariant)**.** *The class of bigraphs I is preserved by the reaction relation $\longrightarrow\!\!\!\!\triangleright$ defined in Section 3.4 and contains all images of the translation given in Definition 3.4.*

3.4 Reaction rules of ´Homerσ

In this subsection we present the reaction rules of ´Homerσ. As mentioned in Section 1 we have chosen to present the rules using a term language instead of the graphical representation, due to the complexities of the rules.

Definition 3.12 (reaction rules of ´Homerσ)**.** We depict the reaction rules of ´Homerσ in Table 6 (the send and take rules) and Table 7 (applying and garbage collecting the substitution).

Remark. In all the rules we have chosen to enumerate the holes from left to right in the terms representing the bigraphs, but omitting the last k holes in the $k+1$-hole path contexts F_γ and $F_\gamma^{\tilde{m}}$ on which the instantiation acts as the identity. In both the

	R
Send	$(\mathbf{send}_{\gamma\delta} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})\big(\mathbf{id}_{\tilde{n}} \mid (\mathbf{ann} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}}) \mid (\mathbf{resi} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})\big) \mid F_{\gamma} \circ (\mathbf{rece}_{\delta(x)} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})$
	R'
	$\mathbf{id}_{\tilde{n}'} \mid (\tilde{n} \odot_b F_{\gamma}) \circ (\mathbf{sub}_{(x)} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})(\mathbf{id}_{x\tilde{n}'} \mid (\mathbf{def}_x \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}})(\mathbf{id}_{\tilde{n}} \mid (\mathbf{ann} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}})))$
	f
	$\{0 \mapsto 2, 1 \mapsto 3, 2 \mapsto 0, 3 \mapsto 1\}$

	R
Take	$F^{\tilde{m}}_{\gamma} \circ (\mathbf{loca}_{\delta} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})(\mathbf{id}_{\tilde{n}} \mid (\mathbf{ann} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}}) \mid (\mathbf{resi} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})) \mid (\mathbf{take}_{\gamma\delta(x)} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})$
	R'
	$/(\tilde{m} \cap \tilde{n}) \circ ((\tilde{n} \odot_b F^{\tilde{m}}_{\gamma}) \circ \mathbf{id}_{\tilde{n}'}) \mid$ $(\mathbf{sub}_{(x)} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})(\mathbf{id}_{x\tilde{n}'} \mid (\mathbf{def}_x \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}})(\mathbf{id}_{\tilde{n}} \mid (\mathbf{ann} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}})))$
	f
	$\{0 \mapsto 2, 1 \mapsto 3, 2 \mapsto 0, 3 \mapsto 1\}$

Table 6: Reaction rules of ´Homerσ: send and take

	R
Apply	$(\mathbf{sub}_{(x)} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})(\mathcal{C} \circ \mathbf{var}_x \mid (\mathbf{def}_x \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}})(\mathbf{id}_{\tilde{n}} \mid (\mathbf{ann} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}})))$
	R'
	$(\mathbf{sub}_{(x)} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})(\tilde{n} \odot_b \mathcal{C} \circ \mathbf{id}_{\tilde{n}} \mid (\mathbf{def}_x \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}})(\mathbf{id}_{\tilde{n}} \mid (\mathbf{ann} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}})))$
	f
	$\{0, 1 \mapsto 0, 2 \mapsto 1\}$

	R	R'	f
Garbage	$(\mathbf{sub}_{(x)} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})\big(\mathbf{id}_{\tilde{n}'} \mid (\mathbf{def}_x \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}})\big)$	$\mathbf{id}_{\tilde{n}'}$	$\{0 \mapsto 0\}$

Table 7: Reaction rules of ´Homerσ: apply and garbage collecting

rules Send and Take the path bigraph F_γ does not bind the names in δ. In both rules the content of the **ann** node is used in the open operator, that is the set $\tilde{n}$. Both rules mimic their counterparts in Homerσ closely. In the rule Apply we utilise a general homer context $\mathcal{C}$ satisfying the sorting requirement and which does not close the variable-link x. The reaction rule Garbage, which discards the explicit substitution, is defined as for ´ΛBIG in [Mil04c].

As mentioned in Section 1 we have explicitly typed the holes in the rules and furthermore defined instantiation and parametric reaction rules such that we do not allow parameters to contain outer names not mentioned explicitly in the rules. We have done this to ensure that the scope of a restriction is not lifted as part of fitting a parameter to a parametric reaction rule. For instance, if we look at the last part of the Apply rule, concerning the **def** node in the explicit substitution

$$(\mathbf{def}_x \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}})(\mathbf{id}_{\tilde{n}} \mid (\mathbf{ann} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}})) \enspace ,$$

then we explicitly state that the names of the process in the substitution and the names of the annotation must be the same, $\tilde{n}$. So it is not possible to lift a restriction from the process, since this would break this correspondence. We will return to this subject in Section 4.2.

The rules Send and Take in Table 6 differ from the usual specification of bigraphical parametric reaction rules. In addition to the usual parameters the rules are also parametrised over *path contexts*, as in place of F_γ and $F_\gamma^{\tilde{m}}$, respectively. In Figure 4 we have sketched the reaction rule Send in order to illustrate this point (we have elided to draw the free names of the sites). In the figure we have drawn the path context F_γ as a site containing a bigraph representing the receive prefix. So in order to utilise this reaction rule we need to both instantiate the parameters and the path context.

There are two reasons why we cannot use a *wide* reaction rule, as Milner [Mil04c], between two complementary prefixes (e.g. the send and receive prefix) to specify the reaction. First and foremost, we need some special structure on the context surrounding the receive prefix, it cannot just be an arbitrary context, as it constitutes part of the address path of the receive prefix. In Figure 4 the path context F_γ specifies that the nested location structure of the path context constitutes the address path γ. Second, the context surrounding the receive prefix can be changed by the reaction rule, due to the update of the type annotations, hence it cannot be an evaluation context as evaluation contexts remain fixed under reaction.

One can consider the parametrisation on path bigraphs in the Send and Take reaction rule as a kind of higher-order operator, which acts as a function on bigraphs, taking a bigraph as argument and returning a new bigraph. However, only a subset of bigraphs (the path bigraphs) are valid arguments.

Example (Mimicking reactions)**.** We consider the reaction

$$\overline{on}\langle r \mid r' \rangle_{\{m\}} \,.\, p \mid o[n(x)\,.\,x]_{\{n\}} \searrow p \mid o[r \mid r']_{\{n,m\}}$$

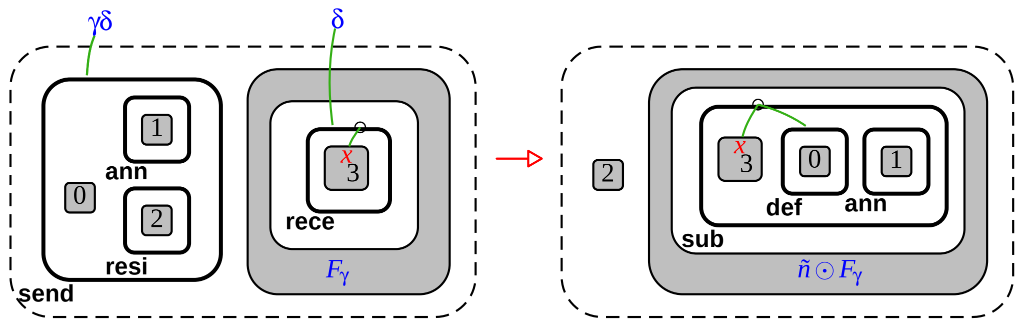

Figure 4: Sketch of the rule Send

in Homer and show how the variant Homerσ and its presentation as a bigraphical reactive system mimics this reaction. Note that we have omitted the top-level type. The prefix $\overline{on}\langle r \mid r'\rangle_{\{m\}}$ sends down the process $r \mid r'$ to the receiving process $n(x)\,.\,x$ residing at location o and as a side-effect updates the type annotation of the location. In Homerσ we have the following sequence of reactions

$$\begin{gathered}
\overline{on}\langle r \mid r'\rangle_{\{m\}}\,.\,p \mid o[n(x)\,.\,x]_{\{n\}} \searrow_\sigma \\
p \mid o[x[x := (r \mid r') : \{m\}]]_{\{n,m\}} \searrow_\sigma \\
p \mid o[(r \mid r')[x := (r \mid r') : \{m\}]]_{\{n,m\}} \searrow_\sigma \\
p \mid o[r \mid r']_{\{n,m\}}
\end{gathered}$$

using the rules, *send*σ, *apply*σ, and *garbage*σ. In the second line we have the location o containing the process variable x enclosed in an explicit substitution, which can substitute $r \mid r'$ of type $\{m\}$ in for x.

In bigraphs we have the matching sequence of reactions depicted in Figure 5. Note that we have chosen not to draw the free names of p and the possible free name m of r and r'.

3.5 Correspondence

In this subsection we present the operational correspondence between Homerσ and its presentation as a bigraphical reactive system ´Homerσ. We prove the operational correspondence by considering the individual rules constituting the reaction relations $\searrow_\sigma$ and $\multimap$. By inspecting the translation we can see that evaluation contexts in Homerσ are translated to active contexts, and conversely if the image under the translation is an active context then the preimage must have been an evaluation context. We follow the same method as Jensen and Milner by first characterising the reactions in both Homerσ and ´Homerσ by the forms of the expressions involved.

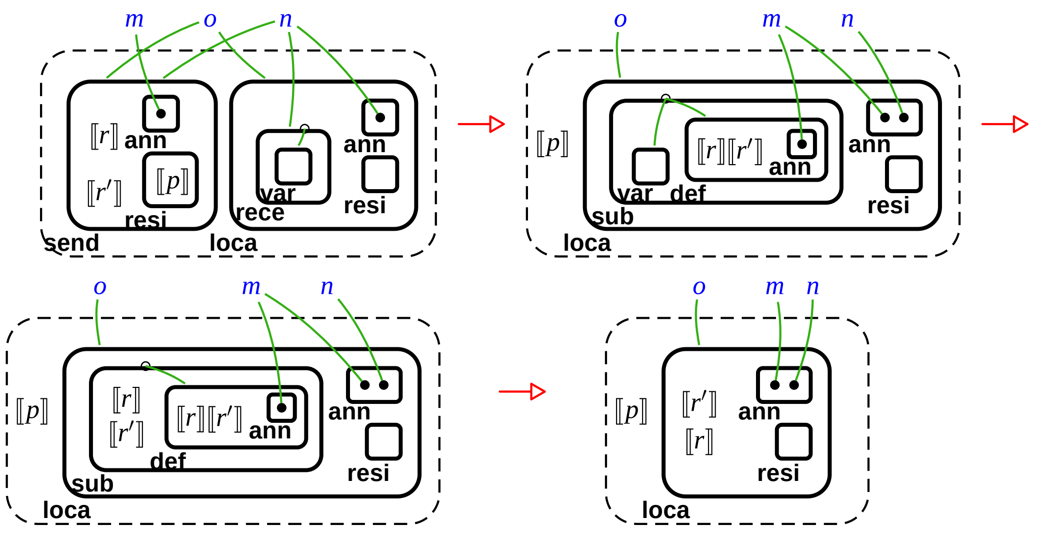

Figure 5: Mimicking $\overline{on}\langle r \mid r'\rangle_{\{m\}} . p \mid o[n(x) . x]_{\{n\}} \searrow p \mid o[r \mid r']_{\{n,m\}}$

Then we use the definition of the translation to connect the characterisations. We only present two of the cases (*garbageσ*) and (*sendσ*) the remaining two are similar.

Proposition 3.13 and Proposition 3.14 characterise the reaction relations $\searrow_\sigma$ and $\multimap$ (for the rules (*garbageσ*) and Garbage, respectively) in terms of the form of the processes and bigraphs.

Proposition 3.13. *$\vdash p \searrow_\sigma p' : \tilde{n}$ by the rule (garbageσ) if and only if p and p' are of the forms*

$$\begin{aligned} \vdash p &\equiv_\sigma \mathcal{E}(q[x := r : \tilde{n}']) : \tilde{n} \\ \vdash p' &\equiv_\sigma \mathcal{E}(q) : \tilde{n} \ , \end{aligned}$$

if $x \notin fv(q)$ for some evaluation context $\mathcal{E}$ and closed processes q and r.

Proposition 3.14. *$g \multimap g'$ by the rule Garbage if and only if g and g' are of the forms*

$$\begin{aligned} g &= E \circ \big((\mathbf{sub}_{(x)} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}})\big(h \mid (\mathbf{def}_x \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})h'\big)\big) \\ g' &= E \circ h \ , \end{aligned}$$

for some bigraphs h and h' with outer face $\tilde{n}$ and $\tilde{n}'$, respectively, and an active context E.

Lemma 3.15 (operational correspondence on (*garbageσ*) and Garbage)**.**
$\vdash p \searrow_\sigma p' : \tilde{n}$ by the rule (garbageσ) and $[\![\vdash p' : \tilde{n}]\!] = s$ if and only if $[\![\vdash p : \tilde{n}]\!] \multimap s$ by the rule Garbage.

Proof. From Proposition 3.13 we know that $\vdash p \searrow_\sigma p' : \tilde{n}$ if and only if p and p' have the forms

$$\begin{aligned} \vdash p \quad &\equiv_\sigma \quad \mathcal{E}(q[x := r : \tilde{n}']) : \tilde{n} \\ \vdash p' \quad &\equiv_\sigma \quad \mathcal{E}(q) : \tilde{n} \ , \end{aligned} \tag{17}$$

for some evaluation context $\mathcal{E}$, closed processes q and r, and where $x \notin fv(q)$. From α-conversion we can assume that all bound names are distinct and disjoint from the free names, and without loss of generality that the hole of $\mathcal{E}$ is annotated with the type $\tilde{n}''$. From the correspondence between structural congruence and graph isomorphism (Theorem 3.7) and Definition 3.4 (17) holds if and only if

$$\begin{aligned} \llbracket \vdash p : \tilde{n} \rrbracket \quad &= \quad \llbracket \vdash \mathcal{E} : \tilde{n} \rrbracket \circ (\llbracket \vdash q[x := r : \tilde{n}'] : \tilde{n}'' \rrbracket) \\ &= \quad \llbracket \vdash \mathcal{E} : \tilde{n} \rrbracket \circ ((\mathbf{sub}_{(x)} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}''})(\llbracket \vdash q : \tilde{n}'' \rrbracket \mid (\mathbf{def}_x \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})(h'))) \\ \llbracket \vdash p' : \tilde{n} \rrbracket \quad &= \quad \llbracket \vdash \mathcal{E} : \tilde{n} \rrbracket \circ (\llbracket \vdash q : \tilde{n}'' \rrbracket) \ , \end{aligned}$$

since $x \notin fv(q)$ and letting $h' = \llbracket \vdash r : \tilde{n}' \rrbracket \mid (\mathbf{ann} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})\llbracket \tilde{n}' \rrbracket$. By Proposition 3.14, taking $h = \llbracket \vdash q : \tilde{n}'' \rrbracket$, this holds if and only if $\llbracket \vdash p : \tilde{n} \rrbracket \rightarrowtriangle \llbracket \vdash p' : \tilde{n} \rrbracket$ by the rule Garbage. $\square$

We proceed in the same manner with the case for (*send*σ). Proposition 3.16 and Proposition 3.17 characterise the reaction relations $\searrow_\sigma$ and $\rightarrowtriangle$ (for the rules (*send*σ) and Send, respectively) in terms of the form of the processes and bigraphs.

Proposition 3.16. *$\vdash p \searrow_\sigma p' : \tilde{n}$ by the rule (sendσ) if and only if p and p' are of the forms*

$$\begin{aligned} \vdash p \quad &\equiv_\sigma \quad \mathcal{E}(\overline{\gamma\delta}\langle r \rangle_{\tilde{n}'} \,.\, q \mid \mathcal{C}_\gamma^{\tilde{m}}(\delta(x) \,.\, q', \vec{q})) : \tilde{n} \\ \vdash p' \quad &\equiv_\sigma \quad \mathcal{E}(q \mid \tilde{n}' \odot \mathcal{C}_\gamma^{\tilde{m}}(q'[x := r : \tilde{n}'], \vec{q})) : \tilde{n} \ , \end{aligned}$$

if $\tilde{m} \cap (\delta \cup \tilde{n}) = \emptyset$ for some evaluation context $\mathcal{E}$, path context $\mathcal{C}_\gamma^{\tilde{m}}$, closed processes r, q, and $\vec{q}$, and some process q' where $fv(q') \subseteq \{x\}$.

Proposition 3.17. *$g \rightarrowtriangle g'$ by the rule Send if and only if g and g' are of the forms*

$$\begin{aligned} g \quad &= \quad E \circ ((\boldsymbol{send}_{\gamma\delta} \mathbin{\overline{\oplus}} \boldsymbol{id}_{\tilde{n}''})\big(h \mid (\boldsymbol{ann} \mathbin{\overline{\oplus}} \boldsymbol{id}_{\tilde{n}'})h' \mid (\boldsymbol{resi} \mathbin{\overline{\oplus}} \boldsymbol{id}_{\tilde{n}''})h''\big) \mid \\ &\qquad\quad F_\gamma \circ ((\boldsymbol{rece}_{\delta(x)} \mathbin{\overline{\oplus}} \boldsymbol{id}_{\tilde{n}'''})h''')) \\ g' \quad &= \quad E \circ (h'' \mid (\tilde{n}' \odot_b F_\gamma) \circ (\boldsymbol{sub}_{(x)} \mathbin{\overline{\oplus}} \boldsymbol{id}_{\tilde{n}'''})(\\ &\qquad\qquad h''' \mid (\boldsymbol{def}_x \mathbin{\overline{\oplus}} \boldsymbol{id}_{\tilde{n}'})(h \mid (\boldsymbol{ann} \mathbin{\overline{\oplus}} \boldsymbol{id}_{\tilde{n}'})h'))) \ , \end{aligned}$$

for some bigraphs h and h' with outer face $\tilde{n}'$, h'' with outer face $\tilde{n}''$, and h''' with outer face $\tilde{n}'''$, and an active context E with inner face $\tilde{n}''$, and a path bigraph F_γ with inner face $\tilde{n}'''$.

Note that we leave the last k holes in the $k+1$-hole path context F_γ unspecified, as the content of these holes remains fixed under the reaction rule.

Lemma 3.18 (operational correspondence on (*send*σ) and Send)**.** *$\vdash p \searrow_\sigma p' : \tilde{n}$ by the rule (sendσ) and $\llbracket \vdash p' : \tilde{n} \rrbracket = s$ if and only if $\llbracket \vdash p : \tilde{n} \rrbracket \rightarrowtriangle s$ by the rule Send.*

Proof. From Proposition 3.16 we know that $\vdash p \searrow_\sigma p' : \tilde{n}$ if and only if p and p' have the forms

$$\begin{array}{lll} \vdash p & \equiv_\sigma & \mathcal{E}(\overline{\gamma\delta}\langle r\rangle_{\tilde{n}'} \,.\, q \mid \mathcal{C}^{\tilde{m}}_{\gamma}(\delta(x)\,.\,q', \vec{q})) : \tilde{n} \\ \vdash p' & \equiv_\sigma & \mathcal{E}(q \mid \tilde{n}' \odot \mathcal{C}^{\tilde{m}}_{\gamma}(q'[x := r : \tilde{n}'], \vec{q})) : \tilde{n} \ , \end{array}$$

for some evaluation context $\mathcal{E}$, path context $\mathcal{C}^{\tilde{m}}_{\gamma}$, closed processes r, q, and $\vec{q}$, and some process q' where $fv(q') \subseteq \{x\}$, and where $\tilde{m} \cap (\delta \cup \tilde{n}) = \emptyset$. From α-conversion we can assume that all bound names are distinct and disjoint from the free names, and without loss of generality that the hole of $\mathcal{E}$ is annotated with $\tilde{n}''$. From the correspondence between structural congruence and graph isomorphism we have

$$\begin{array}{lll} [\![\vdash p : \tilde{n}]\!] & = & [\![\vdash \mathcal{E} : \tilde{n}]\!] \circ ([\![\vdash \overline{\gamma\delta}\langle r\rangle_{\tilde{n}'} \,.\, q \mid \mathcal{C}^{\tilde{m}}_{\gamma}(\delta(x)\,.\,q', \vec{q}) : \tilde{n}'']\!]) \\ & = & [\![\vdash \mathcal{E} : \tilde{n}]\!] \circ ((\mathbf{send}_{\gamma\delta} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}''})([\![\vdash r : \tilde{n}']\!] \mid ((\mathbf{ann} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})[\![\tilde{n}']\!]) \mid \\ & & (\mathbf{resi} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}''})[\![\vdash q : \tilde{n}'']\!]) \mid [\![\vdash \mathcal{C}^{\tilde{m}}_{\gamma} : \tilde{n}'']\!] \circ (\mathbf{rece}_{\delta(x)} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'''})[\![\{x\} \vdash q' : \tilde{n}''']\!]) \\ [\![\vdash p' : \tilde{n}]\!] & = & [\![\vdash \mathcal{E} : \tilde{n}]\!] \circ ([\![\vdash q \mid \tilde{n}' \odot \mathcal{C}^{\tilde{m}}_{\gamma}(q'[x := r : \tilde{n}'], \vec{q}) : \tilde{n}'']\!]) \\ & = & [\![\vdash \mathcal{E} : \tilde{n}]\!] \circ ([\![\vdash q : \tilde{n}'']\!] \mid (\tilde{n}' \odot_b [\![\vdash \mathcal{C}^{\tilde{m}}_{\gamma}]\!]) \circ (\mathbf{sub}_{(x)} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'''}) \\ & & ([\![\{x\} \vdash q' : \tilde{n}''']\!] \mid (\mathbf{def}_x \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})([\![\vdash r : \tilde{n}']\!] \mid (\mathbf{ann} \mathbin{\overline{\oplus}} \mathbf{id}_{\tilde{n}'})[\![\tilde{n}']\!]))) \end{array}$$

By Proposition 3.17 this holds if and only if $[\![\vdash p : \tilde{n}]\!] \rightarrow [\![\vdash p' : \tilde{n}]\!]$ by the rule Send.
□

Theorem 3.19 (Operational Correspondence)**.** *For every well-typed process* $\vdash p : \tilde{n}$*, we have*

$$\vdash p \searrow_\sigma p' : \tilde{n} \textit{ and } [\![\vdash p' : \tilde{n}]\!] = s \textit{ if and only if } [\![\vdash p : \tilde{n}]\!] \rightarrow s \ .$$

4 Extensions and Ideas

In this section we present some of the thoughts and ideas that have arisen during the work on this paper. We describe two orthogonal directions: one is a possible change to the definition of local bigraphs, called localised links, that would simplify the presentation presented in this paper by facilitating an alternative representation of type annotations. The other address the location of a closed link arising from the closure operator used to represent the restriction constructor in Homerσ, and how our type annotations affects this location.

4.1 Localised Links

In the presentation of Homerσ in Section 3 we utilised the dots • (the nodes of control **tname**) to represent the type annotations of location and send prefixes. Since the type annotations in Homerσ are sets we needed a way to associate an arbitrary number of links (or names)[2] to a node in an *unordered* way. We chose the same solution

[2]Throughout this section we will use the words 'links' and 'names' interchangeably, as the extension affects both kinds of entities.

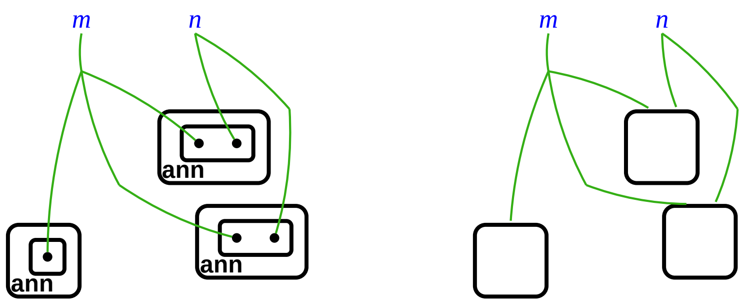

Figure 6: Original representation and using localised links

as presented in the presentation of "The Game of Life" in [DD05]. In the left-hand side of Figure 6 we have sketched a situation where we have 3 nodes representing arbitrary Homerσ-prefixes, and where we would like to associate the name m with all three nodes and the name n only with the two rightmost nodes. The solution used in this paper is to introduce an **ann** node as a child of the node and let it contain one **tname** node per name that we want to associate with the grand-parent node. The single port on the **tname** nodes are then linked to the names. Using this method we can associate an arbitrary number of names to a node in an *unordered* way.

A desired solution would be, to be able to associate a name directly to a node instead of a port on the node, hence to be able to associate names to a node in an *unordered* way, as illustrated on the right-hand side of Figure 6. So the names m and n on the right-hand side of Figure 6 are *not* connected to any ports on the three nodes, but directly to the nodes. In this section we will briefly expand on this possibility, which we will call *localised links*. A direct consequence of this extension will be that we can remove the controls **tname** and **ann** from the presentation and instead represent the type annotations directly using localised links. In Figure 7 we have illustrated the final result in Figure 5, the process $p \mid o[r \mid r']_{\{n,m\}}$, using localised links. Note that the name o is connected to the only port on the **loca** node, and the names m and n are connected to the **loca** node. We will expand on an additional advantage of localised links in Section 4.2 in connection with the location of a closed link.

Recall that for a local bigraph we require that both its interfaces are local, meaning that all names in the interface are given one (or more) places in the interface. However contrary to the way we define the association between names and nodes, we define the association between names and places in the interface in an unordered way. Note that we do not propose localised links as a replacement for traditional links, but rather as an addition to these, as we in most presentations need the ordering of ports. Note that a link can both be a traditional link and a localised link at the same time. In the presentation of $m[p]_{\{m\}}$ using localised links the name m will both be used as a traditional link, to represent the name of the location, and as a localised

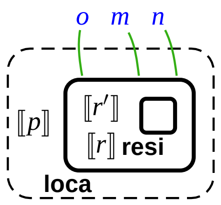

Figure 7: Presenting $p \mid o[r \mid r']_{\{n,m\}}$ with localised links

link, to represent the type annotation $\{m\}$ of the location.

Formally, we suggest to introduce a new function to the definition of a local bigraph. For a local bigraph $G : \langle m, \vec{X} \rangle \to \langle n, \vec{Y} \rangle$ with the set of edges E and the set of nodes V, we let the function *localise* map edges and outer names to a set of locations, $\mathit{localise} : E \uplus Y \to \mathcal{P}(V)$, where we let $\mathcal{P}(V)$ denote the powerset of the set of nodes. We require that this map satisfies a scoping condition as for traditional links, meaning that for an outer name y it is only mapped to nodes that are located in regions, where the name is also located. We define the composition of two bigraphs

$$F : \langle m, \vec{X} \rangle \to \langle n, \vec{Y} \rangle \qquad \text{with nodes } V \text{, edges } E \text{, and function } \mathit{localise}$$

and

$$G : \langle l, \vec{Z} \rangle \to \langle m, \vec{X} \rangle \qquad \text{with nodes } V' \text{, edges } E' \text{, and function } \mathit{localise}'$$

as usual for local bigraphs. The localisation function

$$\mathit{localise}'' : E \uplus E' \uplus Y \to \mathcal{P}(V) \uplus \mathcal{P}(V')$$

for $F \circ G$ is defined as follows (using the link map, *link*, of F)

$$\mathit{localise}''(x) = \begin{cases} \mathit{localise}'(x) & \text{if } x \in E' \ , \\ \mathit{localise}(x) \biguplus_{x' \in X \text{ and } \mathit{link}(x') = x} \mathit{localise}'(x') & \text{if } x \in E \uplus Y \ , \end{cases}$$

So the locations of an edge in E' remain unchanged by the composition, whereas for a name in Y or an edge in E we might need to combine the locations of *localise* and *localise*$'$, if a name in X links to the name or edge, respectively.

4.2 The Location of Restriction

As mentioned in the introduction we have to be careful when combining local names and non-linear process passing. Since the two processes

$$(n)m[P] \quad \text{and} \quad m[(n)P] \qquad \text{(assuming } n \neq m\text{)} \tag{18}$$

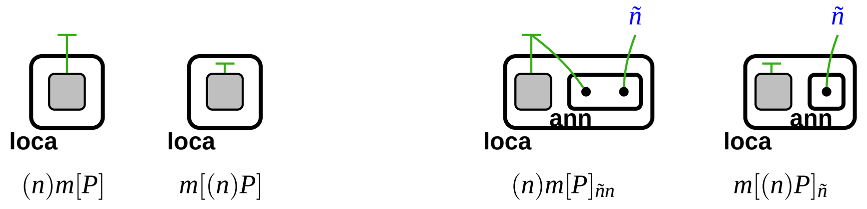

Figure 8: Location of a restriction

are not structural congruent in general, they should not give rise to isomorphic bigraphs under the translation defined in Section 3.1. If we consider our presentation without type annotations then the two processes in (18) will give rise to isomorphic bigraphs, since we have no means to detect whether the closure occur outside or inside the location, as illustrated in the left part in Figure 8 (note that we have ignored the residual process after the location prefix and the name of the location).

Recall that we have decided to use closure $/n$ in bigraphs to represent the restriction constructor in Homerσ. In bigraphical reactive systems which copy parameters this can lead to the same kind of problems as mentioned in the introduction. In the right-hand part in Figure 8 we have illustrated how the type annotations helps us in distinguishing the two bigraphs. If the restricted name appears in the type annotation then the closure must be outside the location and every copy of the parameter will share this link. On the other hand, if the restricted name does not appear in the type annotation then the closure must be inside the location and every copy of the parameter will have a distinct link.

Even though the type annotations solve the problem in distinguishing the representations of the Homerσ processes in (18), the solution does not, however, exactly match our intuition about where the closure resides. We illustrate this in Figure 9, where the two bigraphs both represent the process $(n)m[P]_{\tilde{n}n}$ under the translation. The problem with our representation of type annotations is that the annotations are located inside the **loca** node and not on the "border" or outside of the node as in Homerσ. So we cannot ensure that the closure remains outside the border of the **loca** node. Note that in our syntactic term language representation of bigraphs is not possible to express the bigraph on the right-hand side of Figure 9, as this would imply that a restriction *inside* a location can bind the names in the type annotation, hence breaking the lexical scoping of restriction.

This ambiguity does not, however, affect the reaction relation of´Homerσ as our reaction rules, presented in Definition 3.12, are explicitly typed and since we define instantiation without forcing the parameters on discrete normal form. In particular the explicitly typing of holes in the reaction rules involving type annotations (Send, Take, and Apply) forces the hole of the resource to have the same set of free names as the hole of the type annotation. Since we at the same time define instantiation where the parameter has no superfluous outer names, we can ensure that if the re-

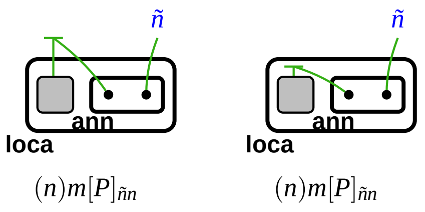

Figure 9: Ambiguousness of the location of a restriction

stricted name appears in the type annotation (meaning that the restriction is outside the node) then every copy of the parameter will share this link. On the other hand, if the restricted name does not appear in the type annotation then every copy of the parameter will have a distinct link.

An immediate suggestion for an alternative to the type annotations is to represent name closures explicitly as a control with a binding port. However, then the usual scope condition would require the place with the binding port in the representation of $(n)p$ to be *around* the process p, which would break the usual structural congruence equalities such as

$$(n)(m)p \equiv_\sigma (m)(n) \quad \text{and} \quad (n)p \mid q \equiv_\sigma (n)(p \mid q), \text{ for } n \notin fn(q). \tag{19}$$

Recently Jensen and Milner have proposed a solution to the same problem of copying parameters with closed links unambiguously. In their solution they make use of an atomic **res** node for the restriction with a new kind of *outward-binding* port. The sole purpose of the **res** node is to facilitate this binding port, but contrary to the binding ports seen so far this port is outward-binding. As the name implies an outward-binding port does not bind inside the node (as the node is atomic), but instead it binds inside the parent node, however the scope of the binding port cannot extend outside the parent node, so its scope covers sibling nodes and their descendants. In Figure 10 we have illustrated the concept of outward-binding. We have an outward-binding **res** node which scope covers the entire content of the surrounding **loca** node. Besides this change in the scope of the binder the outward-binding port behaves as a traditional binding port.

This explicit representation of restriction using one **res** place per restriction behaves well wrt the structural equalities in (19), but instead it breaks the equalities:

$$\pi\,.\,(n)p \equiv_\sigma (n)\pi\,.\,p, \text{ if } n \notin fn(\pi) \qquad \text{and} \qquad (n)p \equiv_\sigma p, \text{ if } n \notin fn(p).$$

In the first case the location of the restriction differs and in the second case the left-hand side contains an outward-binding **res** node whereas the right-hand side does not. So if we consider our presentation with outward-binding then graph isomorphism will correspond to an equivalence strictly finer than structural congruence. Another solution would be to consider bigraphs quotiented by an equivalence coarser than lean-support equivalence.

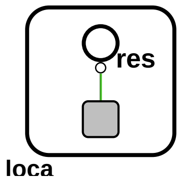

Figure 10: Outward-binding **res** node

More importantly even if we add outward-binding to our presentation we will still need the type annotations, as we loose information about free names when we compose bigraphs. We consider the following counterexample from the introduction with the processes

$$p \overset{def}{=} \mathbf{0}$$

and

$$q \overset{def}{=} (m)(m[n[\mathbf{0}]]) \quad \text{(where } m \neq n\text{)}$$

as bigraphs with the outer name n. If we insert them into the representation of the following context (using outward-binding to represent the restriction)

$$C \overset{def}{=} (n)(m'[(-)]) \ ,$$

then we for p's case have lost the information that the process residing at location m' knows the name n, whereas we still have this information for q. So we can create a counterexample as done in (13) in the introduction. So without the explicit localisation of links within active sub locations we loose local information about the outer names of a process when we place it in a context. So this solution does not provide the desired *bisimulation* congruence.

An alternative solution for making the connection to Homerσ more tight will probably be to use localised links or to locate the **ann** and **resi** nodes of a prefix outside the prefix. In Figure 11 we have sketched how we will represent the Homerσ process $\delta[r]_{\tilde{n}} \,.\, q$ using this alternative representation. This solution will force the closure to be outside the **loca** node if the name appears in the annotation, as the scope of the closure must cover both the **loca** and the **ann** node. The solution with localised links will also force the closure to be outside the **loca** node if the name appears in the annotation, since the localised links representing the type annotation will be connected on the **loca** node.

5 Conclusions and Further Work

We have presented a higher-order calculus with non-linear active process mobility and local names, Homer as a bigraphical reactive system ´Homerσ. To this end

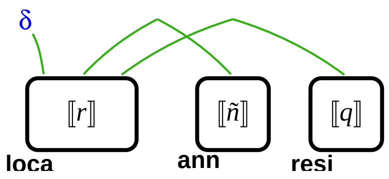

Figure 11: An alternative representation of the process $\delta[r]_{\tilde{n}} . q$

we have introduced a variant of Homer called Homerσ, where we have introduced explicit substitutions. The presentation of Homer in this paper also differs from existing presentations in that we have replaced the free name extension operator with type annotations and in that we only consider well-typed relations between processes with the same type.

We prove that structural congruence of Homerσ corresponds to graph isomorphism in ´Homerσ and that there is a tight operational correspondence between the reaction relation of Homerσ and the reaction relation of ´Homerσ. The presentation highlights the importance of keeping explicit track of the names of parameters in the reaction rules of bigraphs. This ensures us that we can handle the problems with local names and non-linear process passing properly. It also address the issue of localisation of names (links) which suggests an extension to local bigraphs called *bigraphs with localised links*. The presentation in this paper extends the one given in [BH05] to include the full Homer calculus.

Several interesting questions arise from the work done in this paper. First and foremost, we plan to examine the labelled transition bisimulation congruence derivable using the general theory of bigraphs and compare it to the labelled transition bisimulation congruences for Homer in [HGB04]. In this process we plan to examine proof techniques known from calculi for concurrency and mobility in the setting of bigraphs. Especially we plan to investigate the notion of *up-to* proof techniques related to bisimulation equivalences in bigraphs. We would also like to further examine the extension of localised links, both with respect to facilitate presentations as bigraphical reactive systems and with respect to the behavioural theory of bigraphical reactive systems. In particular we would like to examine if the extension retains relative pushouts. Also we would like to examine to which extend we can express one kind of linkage using the other. In this paper we have briefly sketched how we can represent localised links using traditional links and nested controls. For the other direction we would probably need one control for each ordinal, or use nesting, to represent the ordering of ports on a node. However both solutions seems to require a large and complicated sorting scheme.

Another direction could be to pursue the alternative presentation sketched in the last part of Section 4.2 and see how it relates to the presentation presented in this paper. An immediate consequence of the alternative presentation is that the reaction rules will become more complicated as we need to introduce more linking information in the rules. A yet unexamined direction is to investigate a variant of Homer with type annotations, but without the restriction constructor in the syntax. From a typed Homer process with type annotations we can calculate which names that are bound and the scope of the restriction binding them.

Currently several proposals exists for augmenting either the controls or interfaces of bigraphs with additional information for expressing constraints on the possible nesting of nodes, the possible linkage between ports etc. It would be interesting to see whether the sorting presented in Section 3.3 can be expressed in these settings, and whether we can enforce a more strict control with the movement and locations of closed free links. Hence to capture some of the same informations as the outward-binding node, but without introducing an explicit node representing the restriction. A first try could be to augment the controls with information about whether a control permits that closed free links penetrate the boundary of the control.

Acknowledgements The authors would like to thank the members of the *Bigraphical Programming Languages*-group at ITU, in particular Troels C. Damgaard, for helpful discussions about how to represent certain properties and constructs in bigraphs. The authors would also like to thank Søren Debois for creating the initial PSTricks macros for drawing bigraphs.

References

[ACCL91] Martin Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Levy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[Bar84] Hendrik P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., 1984.

[BB90] Gerard Berry and Gérard Boudol. The chemical abstract machine. In *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming laguages (POPL'90)*, pages 81–94. ACM Press, 1990.

[BH05] Mikkel Bundgaard and Thomas Hildebrandt. Bigraphical semantics of higher-order mobile embedded resources with local names. In *Proceedings of the Graph Transformation for Verification and Concurrency workshop (GT-VC'05)*, Electronic Notes in Theoretical Computer Science. Elsevier, 2005. To appear.

[BHG05a] Mikkel Bundgaard, Thomas Hildebrandt, and Jens Chr. Godskesen. A CPS encoding of name-passing in higher-order mobile embedded resources. In Jos Baeten and Flavio Corradini, editors, *Proceedings of the*

11th International Workshop on Expressiveness in Concurrency (EXPRESS'04), volume 128 of *Electronic Notes in Theoretical Computer Science*, pages 131–150. Elsevier, 2005.

[BHG05b] Mikkel Bundgaard, Thomas Hildebrandt, and Jens Chr. Godskesen. A CPS encoding of name-passing in higher-order mobile embedded resources. *Theoretical Computer Science*, 2005. . Accepted for publication in a special issue of TCS.

[BL05] Roberto Bruni and Ivan Lanese. On graph(ic) encodings. In Barbara Koenig, Ugo Montanari, and Philippa Gardner, editors, *Graph Transformations and Process Algebras for Modeling Distributed and Mobile Systems*, number 04241 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2005.

[CG00] Luca Cardelli and Andrew D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, 2000.

[CMS05a] Giovanni Conforti, Damiano Macedonio, and Vladimiro Sassone. Bigraphical logics for XML. Submitted for publication, 2005.

[CMS05b] Giovanni Conforti, Damiano Macedonio, and Vladimiro Sassone. BiLogics: Spatial-nominal logics for bigraphs. Submitted for publication, 2005.

[CVN04] Giuseppe Castagna, Jan Vitek, and Fracesco Zappa Nardelli. The Seal calculus. accepted for publication in *Information and Computation*, 2004.

[DD05] Søren Debois and Troels C. Damgaard. Bigraphs by example. Technical Report TR-2005-61, IT University of Copenhagen, 2005.

[FMQ96] Gianluigi Ferrari, Ugo Montanari, and Paola Quaglia. A π-calculus with explicit substitutions. *Theoretical Computer Science*, 168(1):53–103, 1996.

[Gar00] Philippa Gardner. From process calculi to process frameworks. In Catuscia Palamidessi, editor, *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR'00)*, volume 1877 of *Lecture Notes in Computer Science*, pages 69–88. Springer Verlag, 2000.

[GW00] Philippa Gardner and Lucian Wischik. Explicit fusions. In Mogens Nielsen and Branislav Rovan, editors, *Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science (MFCS'00)*, volume 1893 of *Lecture Notes in Computer Science*, pages 373–382. Springer Verlag, 2000. Full version to appear in TCS.

[HGB04] Thomas Hildebrandt, Jens Chr. Godskesen, and Mikkel Bundgaard. Bisimulation congruences for Homer — a calculus of higher order mobile embedded resources. Technical Report TR-2004-52, IT University of Copenhagen, 2004.

[Hir99] Daniel Hirschkoff. Handling substitutions explicitely in the π-calculus. In *Proceedings of Second International Workshop on Explicit Substitutions: Theory and Applications to Programs and Proofs (WESTAPP'99)*, pages 28–43, 1999.

[HNOW05] Thomas Hildebrandt, Henning Niss, Martin Olsen, and Jacob W. Winter. Distributed reactive XML - an XML-centric coordination middleware. Technical Report TR-2005-62, IT University of Copenhagen, 2005.

[HW05] Thomas Hildebrandt and Jacob W. Winther. Bigraphs and (reactive) XML — an XML-centric model of computation. Technical Report TR-2005-56, IT University of Copenhagen, 2005.

[Jen05] Ole Høgh Jensen. *Mobile Processes in Bigraphs*. PhD thesis, Department of Computer Science, Aalborg University, 2005. Forthcoming.

[JM03] Ole Høgh Jensen and Robin Milner. Bigraphs and transitions. In *Proceedings of the 30rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL'03)*, pages 38–49. ACM Press, 2003.

[JM04] Ole Høgh Jensen and Robin Milner. Bigraphs and mobile processes (revised). Technical Report UCAM-CL-TR-580, University of Cambridge, Computer Laboratory, 2004.

[LM00] James J. Leifer and Robin Milner. Deriving bisimulation congruences for reactive systems. In Catuscia Palamidessi, editor, *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR'00)*, volume 1877 of *Lecture Notes in Computer Science*, pages 243–258. Springer Verlag, 2000.

[Mil96] Robin Milner. Calculi for interaction. *Acta Informatica*, 33(8):707–737, 1996.

[Mil04a] Robin Milner. Bigraphs for petri nets. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Lectures on Concurrency and Petri Nets: Advances in Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 686–701. Springer Verlag, 2004.

[Mil04b] Robin Milner. Bigraphs whose names have multiple locality. Technical Report UCAM-CL-TR-603, University of Cambridge, Computer Laboratory, 2004.

[Mil04c] Robin Milner. Local bigraphs, confluence and λ-calculus, 2004. Draft of October 31, 2004.

[O'C04] Shane O'Conchuir. A note on λ_{sub} (draft). Unpublished draft, December 2004.

[Ros96a] Kristoffer H. Rose. Explicit substitution — tutorial & survey. Lecture Series LS-96-3, BRICS, Department of Computer Science, University of Aarhus, 1996. v+150 pp.

[Ros96b] Kristoffer H. Rose. *Operational Reduction Models for Functional Programming Languages*. PhD thesis, Department of Computer Science, University of Copenhagen, 1996.

[San92] Davide Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis, Department of Computer Science, University of Edinburgh, 1992.

[Tho93] Bent Thomsen. Plain CHOCS: A second generation calculus for higher order processes. *Acta Informatica*, 30(1):1–59, 1993.

[Zim04] Pascal Zimmer. On the expressiveness of pure mobile ambients. *Journal of Mathematical Structures in Computer Science*, 13(5):721–770, 2004.