
Copyright © 2014 The Authors

A copy can be downloaded for personal non-commercial research or study, without prior permission or charge

Content must not be changed in any way or reproduced in any format or medium without the formal permission of the copyright holder(s)

When referring to this work, full bibliographic details must be given

http://eprints.gla.ac.uk/94772/

Deposited on: 27 June 2014
Towards a Bigraphical Encoding of Actors

Michele Sevegnani$^{1,*}$ and Eloi Pereira$^{2,3,**}$

$^1$ School of Computing Science, University of Glasgow, UK
Michele.Sevegnani@glasgow.ac.uk
$^2$ Systems Engineering, UC Berkeley, USA
eloi@berkeley.edu
$^3$ Research Center, Portuguese Air Force Academy, Portugal

Abstract. Actors are self-contained, concurrently interacting entities of a computing system. They can perform local computations, communicate via asynchronous message passing with other actors and can be dynamically created. Bigraphs are a fully graphical process algebraic formalism, capable of representing both the position in space of agents and their inter-connections. Their behaviour is specified by a set of reaction rules. In this paper, we present a bigraphical encoding of a simplified actor language with static topology. We express actor configurations in terms of sorted bigraphs while the rules of the actor operational semantics are encoded by bigraphical reactive rules.

1 Introduction

Actors [2] is a model for distributed concurrent computing systems. An actor system, called actor configuration, is a collection of autonomous objects, called actors, and messages that have been sent but not yet received. Each actor is characterised by a unique address used for asynchronous communication with other actors and by an internal state. The local computation carried out by an actor is typically specified in a sequential language. However, the authors in [1] developed an actor semantics in a framework where local computation is specified using an extension of λ-calculus. The operational semantics of actor configurations is defined by a transition relation on configurations. We will describe it in greater detail in the next section.

Bigraphical reactive systems (BRS) is a recent formalism for modelling the temporal and spatial evolution of computation. It was initially introduced by Milner [9] to provide a fully graphical model capable of representing both connectivity and locality. A BRS consists of a set of bigraphs and a set of reaction rules, which defines the dynamic evolution of the system by specifying how the

* The author was supported by The Royal Society (through International Exchanges Scheme IE130886) and by The Scottish Informatics and Computer Science Alliance (SICSA).
** The author was supported by the National Science Foundation (CNS1136141), by the Fundação para a Ciência e Tecnologia (SFRH/BD/43596/2008), by the Portuguese MoD - PITVANT
set of bigraphs can be reconfigured. The development of bigraphs has been di-
rected toward both the modelling of ubiquitous systems by focusing on mobile
connectivity and mobile locality [4, 5] and the definition of a unifying theory ca-
pable of representing many existing calculi for concurrency and mobility within
the same theoretical framework. Some examples are the bigraphical encodings
of CCS, mobile ambients, condition-event Petri nets and λ-calculus [7, 8].

In this paper, we present a bigraphical encoding of a simplified actor language
with static topology, i.e. actor addresses are not communicable variables. We
express actor configurations in terms of sorted bigraphs while the rules of the
actor operational semantics are encoded by bigraphical reactive rules.

The paper is organised as follows. In the next section, we recall the definition
of operational semantics for the actor model. In Section 3, we informally intro-
duce bigraphs and BRS. Section 4, describes our bigraphical encoding of actors
into bigraphs. Conclusions and directions for future work are in Section 5.

2 Actors Operational Semantics

In this section we present an operational semantics for the actor model. We
follow a simplified version4 of the approach presented in [10].

Let us set some notational conventions. We let $a,a',\ldots$ range over the set
of actor addresses, $v,v_0,v_1,\ldots$ range over values, and $x$ range over identifiers.
A value expression, i.e. either a value or an identifier, is indicated by $e$, while
$f,f',\ldots$ denote expressions of a sequential language used to specify actor be-
haviors. Since we do not rely on a specific language, its definition is omitted.
Local environments (written as $E,E',\ldots$) are mappings from identifiers to their
values in the form $[x \mapsto v]$.

The semantics is formalized as a transition relation over the set of actor
configurations.

Definition 1. An actor configuration is a pair $\langle \alpha | \mu \rangle$ where $\alpha$ is a set of actor
names, and $\mu$ is the set of pending messages.

Actor configurations are syntactically defined according to the grammar de-
defined in Figure 1.

A busy actor with unique address $a$, local environment $E$, and local behaviour
$b$ is written $[E \vdash b]_a$. Similarly, an inactive actor is denoted by $(E \vdash b)_a$. The
local behaviour $b$ is specified as a sequential composition of actor commands
send($\_,\_\$), ready($\_\$), and new($\_\$), expressions $f$ that manipulate the state of
the actor, and terminal symbol $\text{nil}$. A message is a pair $(a \leftarrow v)$ consisting of a
destination address $a$ and a value to be communicated $v$.

The set of possible computations of an actor configuration is defined in terms
of transition relation $\rightarrow$ defined in Figure 2. The actor model is by principle a
model of concurrency. Thus, the semantics neatly separates concurrent inter-
actions between actors (specified by relation $\rightarrow$) and the internal computations

---

4 Dynamic topologies and external actors are not allowed.
within an actor (modelled by the transition relation $\rightarrow_\lambda$). This provides means for using the actor model with an arbitrary language specified by $\rightarrow_\lambda$. For the sake of readability, we assume that this language is equipped with a sequential composition operator (i.e. $\_;\_\$). For a more abstract actor semantics definition refer to [1].

Our semantics consists of five rules. We now briefly comment on them.

Rule (nil) specifies the behaviour of an actor $a$ with empty behaviour nil. The rule simply removes $a$ from the set of actors.

Rule (fun : $a$) defines the effect on the local state when an actor $a$ performs an internal computation step. It takes an expression $f;b$, evaluates $f$, produces the corresponding side-effects in the local environment (i.e. $E$ becomes $E'$), and changes the behaviour of actor $a$ to $b$.

Rule (new : $a,a'$) defines the creation of a new actor $a'$ spawned by $a$. The rule takes an expression $\text{new}(b') ; b$, creates $a'$ with behaviour $b'$, and changes the behaviour of $a$ to $b$. Note that address $a'$ must be different from all the other actor addresses in the current configuration.

Rule (snd : $a,\{a' \leftarrow v\}$) models actor $a$ sending a message with content $v$ to actor $a'$. The rule takes an expression $\text{send}(a',v) ; b$, adds a message $\{a' \leftarrow v\}$ to the set of pending messages in the configuration, and changes the behaviour of $a$ to $b$.

Finally, rule (rcv : $a,\{a \leftarrow v\}$) defines the evolution of an actor $a$ receiving a message with content $v$. The rule takes an expression $\text{ready}(x) ; b$, and a message $\{a \leftarrow v\}$ from the set of pending messages, substitutes $v$ for all the free occurrences of $x$ in $b$, and changes the behaviour of $a$ to $b[x:=v]$. Note that actor $a$ is inactive in the left-hand side of the rule while it becomes busy in the right-hand side.

3 Bigraphical Reactive Systems

In this section we define informally BRS with rule priorities as in [6], with enough detail to support the encodings we will present in the following sections. Refer to [9] for a complete account.
Constituents of Bigraphs and Graphical Notation Some example bigraphs are depicted in Table 1 (right column). Dashed rectangles denote regions. Their rôle is to describe parts of the system that are not necessarily adjacent. The ovals and circles are nodes, which can represent physical or logical components within the system. Each node has a type, called control, denoted here by the labels $A$ to $D$. The set of controls of a bigraph is called signature. Each node can have zero, one or many ports, indicated by bullets, which represent possible connections. Actual connections are represented as links, depicted by solid (green) lines, which may connect ports and names. In the examples, they are ranged over by $x,y,z$. They can be thought of as links (or potential links) to other bigraphs. Gray squares are called sites. They encode parts of the model that have been abstracted away, also called the parameter of the bigraph. Summarising, nodes represent the spatial placement of agents while links represent their communication capabilities.

Interfaces and Sorting The capabilities of bigraph $B$ to interact with the external environment are recorded in its interface. For example, we write $B : 1 \rightarrow \{2, \{x,y\}\}$ to indicate that $B$ has one site, two regions and its names are $x$ and $y$. Interfaces are ranged over by $I,J,K$. We sometimes use $\epsilon$ to denote 0 and the pair $(0,\emptyset)$.

Controls and links in a bigraph can be classified by means of sorts. A sorting discipline is a triple $\Sigma = (\Theta,K,\Phi)$ where $\Theta$ is a non empty set of sorts, $K$ is a signature and $\Phi$ is a formation rule. Sorts are ranged over by $a,b,\ldots$. A formation rule can be thought of as a set of properties a bigraph has to satisfy. For examples, it can specify that nodes of sort $a$ may only contain $b$-nodes or that $a$-nodes may only be linked to $b$-nodes. Disjunctive sorts are written as $\hat{ab}$, meaning that a node can either be of sort $a$ or sort $b$. The interface of a sorted
Table 1. Operations on bigraphs.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Algebraic</th>
<th>Graphical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel product</td>
<td>$A_{xy} \parallel B_{yz}$</td>
<td>![Parallel product diagram]</td>
</tr>
<tr>
<td>Merge product</td>
<td>$A_{xy} \divides B_{yz}$</td>
<td>![Merge product diagram]</td>
</tr>
<tr>
<td>Nesting</td>
<td>$A_{xy}.B_{xz}$</td>
<td>![Nesting diagram]</td>
</tr>
<tr>
<td>Name closure and new name</td>
<td>$/z A_{xz} \parallel y$</td>
<td>![Name closure diagram]</td>
</tr>
</tbody>
</table>

bigraph is expressed as follow: $B : a \rightarrow \{b, \{z\}\}$. The notation indicates that $B$’s site is of sort $a$, its region is of sort $b$ and its name is $z$. From now on, all bigraphs are assumed sorted.

**Algebraic Definition** The structure of a bigraph can also be formulated in an algebraic form closing resembling traditional process calculi. This is done by combining *elementary bigraphs* via the operations listed in Table 1. Note that, the algebraic form is equivalent to the graphical notation.

Parallel product $F \parallel G$ expresses a bigraphical term obtained by juxtaposing bigraphs $F$ and $G$ and merging their common names. Bigraphs $F$ and $G$ are called the factors of bigraph $F \parallel G$. When bigraphs $F$ and $G$ do not have common names, the same operation is denoted by $F \otimes G$. Note that the two operations described above are not commutative. Similarly to parallel product, merge product $F \divides G$ denotes the juxtaposition of bigraphs $F$ and $G$ which is then placed inside a single region. Common names are merged. Nesting operation $F.G$ allows us to place bigraph $G$ inside $F$. Also in this case, common names are merged. Name closure $/x F$ is used to disallow connections on name $x$ in bigraph $F$. This means that all $F$’s links to $x$ are broken and $x$ removed. In example $/z A_{xz} \parallel y$ given in Table 1, new name introduction is simply indicated.
by \( y \). The new name \( y \) is not linked to any node. Notation \( \lambda F \) is a shorthand for \((/\text{slash.left } x_0 \otimes \cdots \otimes /\text{slash.left } x_{n+1}) F\).

The elementary bigraphs most commonly used in our application are identities and \( 1 \). An example identity is \( \text{id}_{a,b} \). It indicates a bigraph with two regions each one containing a site, and two separate links \( a \) and \( b \). Bigraph \( 1 : 0 \rightarrow 1 \) consists of one single region. Bigraphs in the form \( K_{a_0,\ldots,a_n} : 1 \rightarrow (1,\{a_0,\ldots,a_n\}) \) are called ions.

**Bigraphical Reactive Systems** A Bigraphical Reactive System (BRS) consists of a set of bigraphs representing the state of the system, and a set of reaction rules, defining how the system can reconfigure itself. A reaction rule \( R \) is a triple \((R : m \rightarrow J, R' : m' \rightarrow J, \eta)\), where \( R \) and \( R' \) are bigraphs and \( \eta : m' \rightarrow m \) an instantiation map which determines, for each \( j < m' \), which factor of the parameter of \( R \) should occupy the \( j \)th site of \( R' \). Sometimes, we indicate a rule as \( R = R \rightarrow R' \) when \( \eta \) is the identity map. We also define the interface of a reaction as the interface of its left-hand side \( R \). The evolution of a bigraph \( S_t \) is derived by checking if \( R \) is an occurrence in \( S_t \) (this is also called bigraph matching) and by substituting \( R.d \) with \( R'.\overline{\eta}(d) \) to obtain a new bigraph \( S_{t+1} \). Such a reaction is indicated with \( S_t \rightarrow\text{R} S_{t+1} \). Instance function \( \eta \) on bigraphs is defined as \( \overline{\eta}(d) \overset{\text{def}}{=} \lambda \left(d_0 \otimes \cdots \otimes d_{m+1}\right) \), where \( d = \lambda \left(d_0 \otimes \cdots \otimes d_{m+1}\right) \) and with \( d'_j = d_{\eta(j)} \) for each \( j < m' \). The instance function allows to easily duplicate or discard parts of a bigraph after the application of a reaction rule. We use \( \rightarrow\text{R}^* \) to indicate zero or more applications of a reaction.

**Priority BRS** A Priority BRS (PBRS) is a BRS with rule priorities in the style of [3], i.e. by introducing a partial ordering on the rules of the reactive system. A reaction rule of lower priority can be applied only if no rule of higher priority is applicable. We write \( R < R' \) to indicate that reaction rule \( R' \) has higher priority than reaction rule \( R \). A priority class \( \mathcal{P} \) is a set of reaction rules with the same priority. By an abuse of notation, we write \( \mathcal{P} < \mathcal{P}' \) when, for any two rules \( R \in \mathcal{P} \) and \( R' \in \mathcal{P}' \), we have \( R < R' \). We also say that class \( \mathcal{P} \) has lower priority than class \( \mathcal{P}' \).

4 Encoding the Actor Model in Bigraphs

In this section we define a sorting for the class of bigraphs used to represent actors, an encoding of actors into bigraphical expressions and a translation of the actor model operational semantics given in Figure 2 into bigraphical reaction rules.

4.1 Sorting

The controls listed in Table 2 are used to represent the syntactical structure of actor model configurations specified by the grammar given in Figure 1. In
more detail, sort $b = \{ \text{Snd}, \text{Rdy}, \text{New}, \text{Fun}, \text{Nil} \}$ corresponds to the terminals for symbol $b$, control Mail indicates a set of pending messages $\mu$, control M represents symbol $msg$ and control A encodes symbol $actor$. The other controls are required by the encoding to express an actor’s local environment ($E$), variable names (sort $n$), identifiers (sort $i$), values (sort $v$) and dynamically instantiated actors ($A'$). Finally, control Sub is used by the encoding to represent substitutions of free variables in the body of actor behaviours. The set of sorts is written $\Theta_{\text{Act}} = \{ a, x, b, e, n, m, a', v, i, s \}$ and the signature is given by $K_{\text{Act}} = \bigcup_{S \in \Theta_{\text{Act}}} S$.

Table 2. Controls for encoding $A[\_]$.  

<table>
<thead>
<tr>
<th>Description</th>
<th>Control</th>
<th>Arity</th>
<th>Atomic</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>A</td>
<td>1</td>
<td>no</td>
<td>a</td>
</tr>
<tr>
<td>Mail Box</td>
<td>Mail</td>
<td>0</td>
<td>no</td>
<td>x</td>
</tr>
<tr>
<td>Send a message</td>
<td>Snd</td>
<td>0</td>
<td>no</td>
<td>b</td>
</tr>
<tr>
<td>Ready to receive</td>
<td>Rdy</td>
<td>0</td>
<td>no</td>
<td>b</td>
</tr>
<tr>
<td>Spawn a new actor</td>
<td>New</td>
<td>0</td>
<td>no</td>
<td>b</td>
</tr>
<tr>
<td>Internal computation</td>
<td>Fun</td>
<td>0</td>
<td>no</td>
<td>b</td>
</tr>
<tr>
<td>Termination</td>
<td>Nil</td>
<td>0</td>
<td>yes</td>
<td>b</td>
</tr>
<tr>
<td>Environment</td>
<td>E</td>
<td>0</td>
<td>no</td>
<td>e</td>
</tr>
<tr>
<td>Variable name $x$</td>
<td>$N^x$</td>
<td>0</td>
<td>no</td>
<td>n</td>
</tr>
<tr>
<td>Message</td>
<td>M</td>
<td>1</td>
<td>no</td>
<td>m</td>
</tr>
<tr>
<td>New actor</td>
<td>$A'$</td>
<td>1</td>
<td>no</td>
<td>$a'$</td>
</tr>
<tr>
<td>Integer $v$</td>
<td>$\text{Int}^v$</td>
<td>0</td>
<td>yes</td>
<td>$v$</td>
</tr>
<tr>
<td>Other value type</td>
<td>$\cdots$</td>
<td>0</td>
<td>yes</td>
<td>$v$</td>
</tr>
<tr>
<td>Identifier $x$</td>
<td>$X^x$</td>
<td>1</td>
<td>no</td>
<td>i</td>
</tr>
<tr>
<td>Substitution</td>
<td>Sub</td>
<td>0</td>
<td>no</td>
<td>$s$</td>
</tr>
</tbody>
</table>

The sorting discipline ensures that only bigraphs with a meaningful structure are constructed. For example, it forces actors to have an environment and a unique address and messages to contain a value and have a destination address. This is formalised in formation rule $\Phi_{\text{Act}}$ with conditions $\Phi_i$, $1 \leq i \leq 17$, given in Table 3. In particular, conditions $\Phi_1$-$\Phi_{12}$ specify a hierarchic structure on the placing of the nodes, condition $\Phi_{13}$ ensures that an $\overline{a}a'$-node may only share a name with $m$-nodes, condition $\Phi_{14}$ states that $v$-nodes and Nil-nodes are atomic ($i.e.$ they contain nothing) and conditions $\Phi_{15}$, $\Phi_{16}$ specify the structure for the encoding of substitutions of free variables. Finally, condition $\Phi_{17}$ forbids multiple nodes of controls Mail and Sub. We refer the sorting defined in this section as $\Sigma_{\text{Act}} = (K_{\text{Act}}, \Theta_{\text{Act}}, \Phi_{\text{Act}})$. 
Table 3. Conditions of formation rule $\Phi_{\text{Act}}$.

| $\phi_1$ | all children of a $\theta$-regions have sort $\theta$, where $\theta \in \Theta_{\text{Act}}$ |
| $\phi_2$ | all children of an $a$-node have sort $\hat{b}$e |
| $\phi_3$ | an $a$-node has one $b$-child and one $e$-child |
| $\phi_4$ | all children of an $x$-node have sort $m$ |
| $\phi_5$ | a $\text{Snd}$-node has one $b$-child and one $m$-child |
| $\phi_6$ | a $\text{Rdy}$-node has one $b$-child and one $i$-child |
| $\phi_7$ | a $\text{New}$-node has one $b$-child and one $a'$-child |
| $\phi_8$ | a $\text{Fun}$-node has one $b$-child and zero or more $n$-children |
| $\phi_9$ | all children of an $e$-node have control $n$ |
| $\phi_{10}$ | an $n$-node has one $v$-child |
| $\phi_{11}$ | an $m$-node has one $\tilde{v}$i-child |
| $\phi_{12}$ | an $a'$-node has one $b$-child |
| $\phi_{13}$ | an $aa'$-node may only be linked to $m$-nodes |
| $\phi_{14}$ | $v$-nodes and $\text{Nil}$-nodes are atomic |
| $\phi_{15}$ | an $i$-node may only be linked to $i$-nodes |
| $\phi_{16}$ | all children of an $s$-node have sort $i$ |
| $\phi_{17}$ | $\Sigma_{\text{Act}}$-sorted bigraphs have at most one $\text{Mail}$-node and at most one $\text{Sub}$-node |

4.2 Syntax Encoding

In Figure 3 we define a formal encoding of actors into bigraphical expressions. Formally, we specify a map $A[\_] : \text{Act} \rightarrow \text{BG}(\Sigma_{\text{Act}})$ where $\text{Act}$ is the language produced by the grammar in Figure 1 and $\text{BG}(\Sigma_{\text{Act}})$ is the set of bigraphs satisfying sort $\Sigma_{\text{Act}}$. The active/inactive state of an actor is not modelled explicitly in the encoding. However, observe that inactive actors always correspond to bigraphs in the form $A_a(\text{Rdy} | \text{id})$. An additional encoding $E[\_]_0 : \mathcal{E} \times \mathcal{N} \rightarrow \text{BG}(\Sigma_{\text{Act}})$ specifies an encoding of environments into bigraphs, with $\mathcal{E}$ and $\mathcal{N}$ the sets of local environments and identifiers, respectively. The main feature of this encoding is that only the rightmost assignment of a variable is translated, effectively rendering the stack structure of an actor’s environment. For simplicity, we assume all values are integers.

Finally, note that the actor model assumes a sequential language that specifies the internal computation performed by an actor. Since the terms of this language (i.e. $f$ expressions) are left unspecified in the grammar for actors, we also do not encode them.

Example 1. Take actor configuration $C = \langle A, A' | \epsilon \rangle$ with

$$A = ([x \mapsto 2]_a \triangleright \text{ready}(x); \text{send}(a',x); \text{nil})_a$$

$$A' = [\epsilon \leftarrow \text{send}(a,3); \text{nil}]_{a'}$$
The corresponding bigraphical encoding is as follows:

\[
A[A] = A_\alpha.(E.N_x.\text{Int}_v^x.1 \mid \text{Rdy}./x (X_2.1 \mid \text{Snd}.(M_\alpha.\text{X}_2.1 \mid \text{Nil}.1)))
\]

\[
A[A'] = A_{\alpha'}.(E.1 \mid \text{Snd}.(M_\alpha.\text{Int}_v^x.1 \mid \text{Nil}.1))
\]

\[
A[C] = A[A] \mid A[A'] \mid \text{Mail}.1 \mid \text{Sub}.1
\]

### 4.3 Semantics Encoding

We now encode the rules defining the actor operational semantics given in Figure 2 into bigraphical reaction rules. All the rules respect the sorting \(\Sigma_{\text{Act}}\). Note that an actor transition may correspond to a sequence of one or more bigraphical reactions. This will be discussed at the end of the section.
Rule \( \text{nil} : a \) removes an empty actor from the current configuration. It is encoded by the following bigraphical reaction rule:

\[
R_{\text{nil}} \overset{\text{def}}{=} A_a . (\text{Nil}.1 \mid \text{id}) \rightarrow a \mid 1
\]

with instantiation map \( \eta = [] \) and interface \( e \rightarrow a \). This means that the contents of site 0 (i.e. the site contained by ion \( A_a \) indicated by \( \text{id} \)) in the left-hand side is discarded. In this reaction rule, site 0 acts as a placeholder for the actor’s local environment. Note that actor address \( a \) is still defined in the right-hand side of the reaction rule. A graphical representation of \( R_{\text{nil}} \) is given in Figure 4. The node of control \( \text{Nil} \) is indicated by a solid red box.

![Fig. 4. Bigraphical encoding of rule \( \text{nil} : a \).](image)

Rule \( \text{new} : a, a' \) spawns a new actor. A visual description of its encoding is in Figure 5. Intuitively, the \( A' \)-ion inside \( \text{New} \) becomes an actor \( A \), with new address \( a' \) and a copy of \( a \)'s environment. The algebraic definition is as follows:

\[
R_{\text{new}} \overset{\text{def}}{=} A_a . (\text{New}.(A'_a \mid \text{id}) \mid \text{id}) \rightarrow A_a . (\text{id} \mid \text{id}) \mid A_{a'} . (\text{id} \mid \text{id})
\]

with \( \eta = \{1, \{2\}, \{0\}, \{2\} \} \) and interface \( \text{bbe} \rightarrow \{a, \{a, a'\}\} \). The duplication of site 2 encodes the duplication of the local environment.

![Fig. 5. Bigraphical encoding of rule \( \text{new} : a, a' \).](image)

Rule \( \text{snd} : a, (a' \leftarrow v) \) defines an asynchronous send action. This behaviour is encoded by moving the message contained by a \( \text{Snd} \)-node to the node of control.
Mail which represents the set of pending messages of an actor configuration. The algebraic form of the reaction rule for the encoding is

\[ R_{\text{snd}} \overset{\text{def}}{=} A_a.(\text{Snd}.(M_{a'}.\text{Int}.1 \mid \text{id}) \mid \text{Mail} \longrightarrow A_a.(\text{id} \mid \text{id}) \mid \text{Mail}.(M_{a'}.\text{Int}.1 \mid \text{id}) \]

with \( \eta = \left\{ \{0\}, \{1\}, \{2\} \right\} \) and interface \( \text{bevm} \rightarrow \langle \text{axs}, \{a,a'\} \rangle \). Note that the reaction rule can only be applied when message \( M \) contains a value, i.e. a node of sort \( v \). The equivalent graphical representation is shown in Figure 6. Nodes encoding values are depicted as solid blue boxes.

![Fig. 6. Bigraphical encoding of rule \( \text{snd} : a, (a' \leftrightarrow v) \).](image)

Rule \( \{\text{rcv} : a, (a \leftrightarrow v)\} \) defines the behaviour of an actor when a message is received. The following three reaction rules are required for the encoding:

\[ R_{\text{rcv}} \overset{\text{def}}{=} A_a.\text{Rdy}.(X_x.1 \mid \text{id}) \mid \text{Mail}.(M_a \mid \text{id}) \mid \text{Sub} \longrightarrow A_a.(\text{id} \mid \text{id}) \mid \text{Mail} \mid \text{Sub}.(X_x \mid \text{id}) \]

\[ R_{\text{sub}} \overset{\text{def}}{=} \text{Sub}.(X_x \mid \text{id}) \parallel X_x.1 \longrightarrow \text{Sub}.(X_x \mid \text{id}) \parallel \text{id} \]

\[ R_{\text{rem}} \overset{\text{def}}{=} \text{Sub}.(\langle x X_x \mid \text{id} \rangle \longrightarrow \text{Sub} \]

The corresponding graphical notation is in Figure 7. Reaction rule \( R_{\text{rcv}} \) consumes the \( \text{Rdy} \)-node in the left-hand side and initiates a substitution of the free occurrences of \( x \) by placing \( X_x \) inside \( \text{Sub} \). Moreover, the value carried by message \( M_a \) is placed inside ion \( X_x \) in the right-hand side. This is specified precisely by instantiation map \( \eta_{\text{rcv}} = \left\{ \{0\}, \{1\}, \{3\}, \{2\}, \{4\} \right\} \). In particular, sites 3 and 4 are swapped to represent the value contained in the message moving into identifier \( X_x \). Observe that since messages are placed in the \( \text{Mail} \)-node by rule \( R_{\text{snd}} \) they are guaranteed to always contain a \( v \)-node. The interface is given by \( R_{\text{rcv}} : \text{bevmi} \rightarrow \langle \text{axs}, \{x,a\} \rangle \). The previous rule only initiates the substitution of the free occurrences of \( x \) in the behaviour of the actor. Reaction rule \( R_{\text{sub}} : \{\text{si} \rightarrow \{x\} \} \) needs to be applied to actually replace each occurrence of \( x \). By definition of encoding \( A \{ y \} \), all the free occurrences of \( x \) are linked to an ion \( X_x \) inside the \( \text{Sub} \)-node. This is also specified by the left-hand side of the rule.
where the two X-nodes share name x. In the right-hand side, the content of the X-node inside Sub (i.e. site 0) is duplicated to replace the other X-node as specified by instantiation map \( \eta_{\text{sub}} = \{\{0\}, \{1\}, \{0\}\} \). The third reaction rule removes substitutions that are no longer required. This is encoded by closed link x in the right-hand side. The interface is \( R_{\text{rem}} : vi \rightarrow s \). In this case, the instantiation map is \( \eta_{\text{rem}} = \{\{1\}\} \) because site 0, i.e. the content of ion X\textsubscript{s} is discarded. In order to force rules \( R_{\text{sub}} \) and \( R_{\text{rem}} \) to be applied before any other rule, they are assigned a higher priority. We formalise the complete PBRS resulting from the encoding at the end of the section.

Finally, rule \( \{\text{fun} : a\} \) models the execution of a local computation step. Since expressions f and transition relation \( \rightarrow_{\lambda} \) are left unspecified in the actor operational semantics, it is only possible to define a bigraphical encoding of the effect of the internal computation on the local environment. Our approach is to consider f as an explicit update of the environment. More precisely, we let nodes of control Fun contain a collection of mappings from identifiers to values in the form \( \mathcal{E}[x \mapsto v]_{\emptyset} = N^x.\text{Int}^4.1 \). For instance, if the execution of expression f in environment \( E \) leads to a new environment \( E' = E, [x \mapsto 4, y \mapsto 3] \), then \( A[f] = N^x.\text{Int}^4.1 | N^y.\text{Int}^3.1 \). At this point, the following three bigraphical reaction rules

![Bigraphical encoding](image-url)
can be specified:

\[ R_{\text{fun}} \overset{\text{def}}{=} A_u.(\text{Fun} \mid \text{id}) \rightarrow A_u.(\text{id} \mid \text{id}) \]

\[ R_{\text{upd}} \overset{\text{def}}{=} A_u.(\text{Fun}.(N^x.\text{Int}^x.1 \mid \text{id}) \mid E.(N^x \mid \text{id})) \rightarrow A_u.(\text{Fun} \mid E.(N^x.\text{Int}^x.1 \mid \text{id})) \]

\[ R_{\text{add}} \overset{\text{def}}{=} A_u.(\text{Fun}.(N^x.\text{Int}^x.1 \mid \text{id}) \mid E) \rightarrow A_u.(\text{Fun} \mid E.(N^x.\text{Int}^x.1 \mid \text{id})) \]

In the first one, the Fun-node is removed, the instantiation map is the identity function and the interface is \( \text{be} \rightarrow (a, \{a\}) \). The second reaction rule models a value update of an identifier already present inside the E-node in the left-hand side. Here the instantiation map is \( \eta_{\text{upd}} = \{\{0\}, \{2\}\} \) because the old value for identifier \( N^x \) (i.e. site 1) is discarded. The interface is \( \text{bnn} \rightarrow (a, \{a\}) \). In the right-hand side the new value \( \text{Int}^x \) is in the environment. The last rule is similar to the previous one and handles the case when a new binding is added to the environment. The interface is \( \text{bnn} \rightarrow (a, \{a\}) \). Again the instantiation map is the identity function. Note that in order to have the expected behaviour, \( R_{\text{upd}} \) has to be applied before \( R_{\text{add}} \) and both have to be applied before \( R_{\text{fun}} \). We formalise this by defining priorities over the reaction rules in the encoding as follows:

\[ \Psi_0 = \{R_{\text{nil}}, R_{\text{new}}, R_{\text{snd}}, R_{\text{rcv}}, R_{\text{fun}}\} \]

\[ \Psi_1 = \{R_{\text{sub}}, R_{\text{rem}}\} \]

\[ \Psi_2 = \{R_{\text{add}}\} \]

\[ \Psi_3 = \{R_{\text{upd}}\} \]

where \( \Psi_i < \Psi_j \) if \( i < j \).

The PBRS defining our encoding of the actor operational semantics is given by \( (\text{Bg}(\Sigma_{\text{Act}}), \mathcal{R}) \) with \( \mathcal{R} = \bigcup_{i < 4} \Psi_i \).

**Example 2.** Take bigraph \( A[C] = S_0 \) defined in Example 1. Since the node of control \( \text{Mail} \) is empty, only reaction rule \( R_{\text{snd}} \) can be applied:

\[ S_0 \xrightarrow{\text{snd}} A[C] \mid A_u'.(E.1 \mid \text{Nil}.1) \mid \text{Mail}.(M_a.\text{Int}^3.1) \mid \text{Sub}.1 = S_1 \]

Then, either rule \( R_{\text{rcv}} \) or rule \( R_{\text{nil}} \) can be applied. We show the transitions when a message is received:

\[ S_1 \xrightarrow{\text{rcv}} A_u.(E \mid \text{Snd}.(M_{a'}.X_x.1 \mid \text{Nil}.1)) \mid A' \mid \text{Mail}.1 \mid \text{Sub}.X_x.\text{Int}^3.1 \]

\[ \xrightarrow{\text{sub}} A_u.(E \mid \text{Snd}.(M_{a'}.\text{Int}^3.1 \mid \text{Nil}.1)) \mid A' \mid \text{Mail}.1 \mid \text{Sub}.X_x.\text{Int}^3.1 \]

\[ \xrightarrow{\text{rem}} A_u.(E \mid \text{Snd}.(M_{a'}.\text{Int}^3.1 \mid \text{Nil}.1)) \mid A' \mid \text{Mail}.1 \mid \text{Sub}.1 = S_2 \]

with \( E = E.N^x.\text{Int}^2.1 \) and \( A' = A_{u'}.(E.1 \mid \text{Nil}.1) \). In \( S_2 \), either actor \( a' \) can be removed from the configuration or an asynchronous send can be executed. The
trace showing the evolution of the bigraph when the events happen in this order is:

\[
S_2 \rightarrow_{\text{nil}} A_a (E \mid \text{Snd} \cdot (M_a \cdot \text{Int}^3.1 \mid \text{Nil}.1)) \mid \text{Mail}.1 \mid \text{Sub}.1 \\
\rightarrow_{\text{snd}} A_a (E \mid \text{Nil}.1) \mid \text{Mail} \cdot M_a \cdot \text{Int}^3.1 \mid \text{Sub}.1 = S_3
\]

Finally, actor \(a\) is removed from the configuration:

\[
S_3 \rightarrow_{\text{nil}} a \mid \text{Mail} \cdot M_a \cdot \text{Int}^3.1 \mid \text{Sub}.1 = S_3
\]

### 4.4 Operational Correspondence

We now turn to showing that there is a close operational correspondence between actor configurations \([\alpha \mid \mu]\) and their encodings \(A([\alpha \mid \mu])\). We give some of the correspondences (the others are similar):

\[
\begin{align*}
&\text{snd} : \alpha \iff A([\alpha \mid \mu]) \rightarrow_{\text{snd}} A([\alpha' \mid \mu']) \\
&\text{fun} : \alpha \iff A([\alpha \mid \mu]) \rightarrow_{\text{FUN}} A([\alpha' \mid \mu']) \\
&\text{rcv} : \alpha \iff A([\alpha \mid \mu]) \rightarrow_{\text{RCV}} A([\alpha' \mid \mu'])
\end{align*}
\]

with

\[
\begin{align*}
&\text{FUN} \equiv \text{upd} \circ \text{add} \circ \text{fun} \circ \text{rcv} \circ \text{sub} \circ \text{rem}
\end{align*}
\]

The proofs are by induction over the transition derivation (one case for each rule).

### 5 Conclusion and Future Work

We presented a bigraphical encoding of a simplified actor language with static topology. In more detail, we defined map \(A[\_] : \text{Act} \rightarrow \text{Bg}(\Sigma_{\text{Act}})\) to encode the terms of the actor language into sorted bigraphs and a set of prioritised reaction rules \(R\) to match the behaviours specified by the actor operational semantics. Sorting discipline \(\Sigma_{\text{Act}}\) allowed us to define an encoding introducing a minimal amount of entities unrelated to the input language. The only example of “artificial” entity introduced by our encoding is the node of control \(\text{Sub}\) used to express substitutions of free names. Similarly, rule priorities allowed us to define a PBRS with a small number of reaction rules that do not have an immediate equivalent rule in the actor semantics.
The main limitations of our work derive from the fact that our target actor language does not allow for dynamic topologies, external actors and does not specify terms for internal computation. Future work is to define a similar encoding for a richer actor language in the style of [1]. We will also investigate how the algebraic operators (e.g., composition) on actor configurations can be encoded into bigraphs.

Finally, we will extend our encoding to include BigActors [11] a hybrid model that combines actors with bigraphs.

References