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 directed toward both the modelling of ubiquitous systems by focusing on mobile connectivity and mobile locality [4, 5] and the definition of a unifying theory capable 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 introduce 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 version[4] 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 behaviors. 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* $\langle\!\langle \alpha \mid \mu \rangle\!\rangle$ *where* α *is a set of actor names, and* μ *is the set of pending messages.*

Actor configurations are syntactically defined according to the grammar 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 `nil`. A message is a pair $\langle a \Leftarrow v \rangle$ 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 interactions between actors (specified by relation $\rightarrow$) and the internal computations

[4] Dynamic topologies and external actors are not allowed.

$$
\begin{array}{rl}
\mathit{config}: & \langle\!\langle \mathit{actor}^{*} \mid \mathit{msg}^{*} \rangle\!\rangle \\
\mathit{actor}: & [E \vdash b]_a \quad \| \quad (E \vdash b)_a \\
b: & \mathtt{send}(a, e) \quad \| \quad \mathtt{ready}(x) \quad \| \quad \mathtt{new}(b) \quad \| \quad b; b \quad \| \quad f \quad \| \quad \mathtt{nil} \\
\mathit{msg}: & \langle a \Leftarrow v \rangle
\end{array}
$$

Fig. 1. Grammar for actors syntax.

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 $\langle \mathbf{nil} \rangle$ specifies the behaviour of an actor a with empty behaviour `nil`. The rule simply removes a from the set of actors.

Rule $\langle \mathbf{fun} : a \rangle$ 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 $\langle \mathbf{new} : a, a' \rangle$ defines the creation of a new actor a' spawned by a. The rule takes an expression $\mathtt{new}(b'); b$, 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 $\langle \mathbf{snd} : a, \langle a' \Leftarrow v \rangle \rangle$ models actor a sending a message with content v to actor a'. The rule takes an expression $\mathtt{send}(a', v); b$, adds a message $\langle a' \Leftarrow v \rangle$ to the set of pending messages in the configuration, and changes the behaviour of a to b.

Finally, rule $\langle \mathbf{rcv} : a, \langle a \Leftarrow v \rangle \rangle$ defines the evolution of an actor a receiving a message with content v. The rule takes an expression $\mathtt{ready}(x); b$, and a message $\langle a \Leftarrow v \rangle$ 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.

$$\langle \mathbf{nil} : a\rangle \quad \left\langle\!\!\left\langle \alpha, [E \vdash \texttt{nil}]_a \mid \mu \right\rangle\!\!\right\rangle \to \left\langle\!\!\left\langle \alpha \mid \mu \right\rangle\!\!\right\rangle$$

$$\langle \mathbf{fun} : a\rangle \quad \frac{E \vdash f; b \to_\lambda E' \vdash b}{\left\langle\!\!\left\langle \alpha, [E \vdash f; b]_a \mid \mu \right\rangle\!\!\right\rangle \to \left\langle\!\!\left\langle \alpha, [E' \vdash b]_a \mid \mu \right\rangle\!\!\right\rangle}$$

$$\langle \mathbf{new} : a, a'\rangle \quad \left\langle\!\!\left\langle \alpha, [E \vdash \texttt{new}(b'); b]_a \mid \mu \right\rangle\!\!\right\rangle \to \left\langle\!\!\left\langle \alpha, [E \vdash b]_a, [E \vdash b']_{a'} \mid \mu \right\rangle\!\!\right\rangle \quad a' \text{ fresh}$$

$$\langle \mathbf{snd} : a, \langle a' \Leftarrow v\rangle\rangle \quad \left\langle\!\!\left\langle \alpha, [E \vdash \texttt{send}(a', v); b]_a \mid \mu \right\rangle\!\!\right\rangle \to \left\langle\!\!\left\langle \alpha, [E \vdash b]_a \mid \mu, \langle a' \Leftarrow v\rangle \right\rangle\!\!\right\rangle$$

$$\langle \mathbf{rcv} : a, \langle a \Leftarrow v\rangle\rangle \quad \left\langle\!\!\left\langle \alpha, (E \vdash \texttt{ready(x)}; b)_a \mid \mu, \langle a \Leftarrow v\rangle \right\rangle\!\!\right\rangle \to \left\langle\!\!\left\langle \alpha, [E \vdash b[x := v]]_a \mid \mu \right\rangle\!\!\right\rangle$$

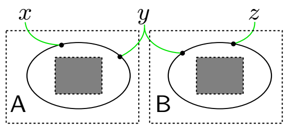
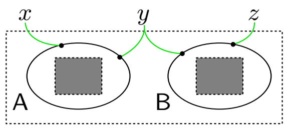
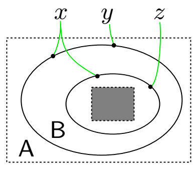
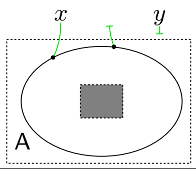
Fig. 2. Actor operational semantics.

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 bigraphs 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 $B : 1 \to \langle 2, \{x, y\}\rangle$ 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 ϵ to denote 0 and the pair $\langle 0, \varnothing\rangle$.

Controls and links in a bigraph can be classified by means of *sorts*. A sorting discipline is a triple $\Sigma = (\Theta, \mathcal{K}, \Phi)$ where Θ is a non empty set of sorts, $\mathcal{K}$ is a signature and Φ is a *formation rule*. Sorts are ranged over by a, b, ... 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 by linked to b-nodes. Disjunctive sorts are written as $\widehat{\mathsf{ab}}$, meaning that a node can either be of sort a or sort b. The interface of a sorted

Table 1. Operations on bigraphs.

Operation	Algebraic	Graphical
Parallel product	$\mathsf{A}_{xy} \parallel \mathsf{B}_{yz}$	
Merge product	$\mathsf{A}_{xy} \mid \mathsf{B}_{yz}$	
Nesting	$\mathsf{A}_{xy}.\mathsf{B}_{xz}$	
Name closure and new name	$/z\, \mathsf{A}_{xz} \parallel y$	

bigraph is expressed as follow: $B : \mathsf{a} \to \langle \mathsf{b}, \{z\}\rangle$. 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 $F \mid G$ 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 $/z\, \mathsf{A}_{xz} \parallel y$ given in Table 1, new name introduction is simply indicated by y. The new name y is not linked to any node. Notation λF is a shorthand for $(/x_0 \otimes \cdots \otimes /x_{n+1})F$.

The elementary bigraphs most commonly used in our application are *identities* and 1. An example identity is $\mathsf{id}_{2,ab}$. It indicates a bigraph with two regions each one containing a site, and two separate links a and b. Bigraph $1 : 0 \to 1$ consists of one single region. Bigraphs in the form $\mathsf{K}_{a_0 \cdots a_n} : 1 \to \langle 1, \{a_0, \ldots, a_n\}\rangle$ 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 $(R : m \to J, R' : m' \to J, \eta)$, where R and R' are bigraphs and $\eta : m' \to m$ an *instantiation map* which determines, for each $j < m'$, which factor of the parameter of R should occupy the jth site of R'. Sometimes, we indicate a rule as $\mathsf{R} = R \longrightarrow R'$ when η 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 $S_t \longrightarrow\!\!\!\triangleright_{\mathsf{R}} S_{t+1}$. Instance function $\overline{\eta}$ on bigraphs is defined as $\overline{\eta}(d) \overset{\text{def}}{=} \lambda(d'_0 \parallel \cdots \parallel d'_{m'+1})$, where $d = \lambda(d_0 \otimes \cdots \otimes d_{m+1})$ 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 $\longrightarrow\!\!\!\triangleright^*$ 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 $\mathsf{R} < \mathsf{R}'$ to indicate that reaction rule R' has higher priority than reaction rule R. A priority class $\mathfrak{P}$ is a set of reaction rules with the same priority. By an abuse of notation, we write $\mathfrak{P} < \mathfrak{P}'$ when, for any two rules $\mathsf{R} \in \mathfrak{P}$ and $\mathsf{R}' \in \mathfrak{P}'$, we have $\mathsf{R} < \mathsf{R}'$. We also say that class $\mathfrak{P}$ has lower priority than class $\mathfrak{P}'$.

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 $\mathsf{b} = \{\mathsf{Snd}, \mathsf{Rdy}, \mathsf{New}, \mathsf{Fun}, \mathsf{Nil}\}$ corresponds to the terminals for symbol b, control Mail indicates a set of pending messages μ, 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 $\Theta_{\mathsf{Act}} = \{\mathsf{a}, \mathsf{x}, \mathsf{b}, \mathsf{e}, \mathsf{n}, \mathsf{m}, \mathsf{a}', \mathsf{v}, \mathsf{i}, \mathsf{s}\}$ and the signature is given by $\mathcal{K}_{\mathsf{Act}} = \bigcup_{S \in \Theta_{\mathsf{Act}}} S$.

Table 2. Controls for encoding $\mathcal{A}[\![\_]\!]$.

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

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 Φ_{Act} with conditions Φ_i, $1 \leq i \leq 17$, given in Table 3. In particular, conditions Φ_1-Φ_{12} specify a hierarchic structure on the placing of the nodes, condition Φ_{13} ensures that an $\widehat{\mathsf{aa}'}$-node may only share a name with m-nodes, condition Φ_{14} states that v-nodes and Nil-nodes are atomic (*i.e.* they contain nothing) and conditions Φ_{15}, Φ_{16} specify the structure for the encoding of substitutions of free variables. Finally, condition Φ_{17} forbids multiple nodes of controls Mail and Sub. We refer the sorting defined in this section as $\Sigma_{\mathsf{Act}} = (\mathcal{K}_{\mathsf{Act}}, \Theta_{\mathsf{Act}}, \Phi_{\mathsf{Act}})$.

Table 3. Conditions of formation rule Φ_{Act}.

Φ_1	all children of a θ-regions have sort θ, where $\theta \in \Theta_{\mathsf{Act}}$
Φ_2	all children of an a-node have sort $\widehat{\mathsf{be}}$
Φ_3	an a-node has one b-child and one e-child
Φ_4	all children of an x-node have sort m
Φ_5	a Snd-node has one b-child and one m-child
Φ_6	a Rdy-node has one b-child and one i-child
Φ_7	a New-node has one b-child and one a'-child
Φ_8	a Fun-node has one b-child and zero or more n-children
Φ_9	all children of an e-node have control n
Φ_{10}	an n-node has one v-child
Φ_{11}	an m-node has one $\widehat{\mathsf{vi}}$-child
Φ_{12}	an a'-node has one b-child
Φ_{13}	an aa'-node may only be linked to m-nodes
Φ_{14}	v-nodes and Nil-nodes are atomic
Φ_{15}	an i-node may only be linked to i-nodes
Φ_{16}	all children of an s-node have sort i
Φ_{17}	Σ_{Act}-sorted bigraphs have at most one Mail-node and at most one Sub-node

4.2 Syntax Encoding

In Figure 3 we define a formal encoding of actors into bigraphical expressions. Formally, we specify a map $\mathcal{A}[\![\_]\!] : Act \to \mathrm{B}\textsc{g}(\Sigma_{\mathsf{Act}})$ where Act is the language produced by the grammar in Figure 1 and $\mathrm{B}\textsc{g}(\Sigma_{\mathsf{Act}})$ is the set of bigraphs satisfying sort Σ_{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 $\mathsf{A}_a.(\mathsf{Rdy} \mid \mathsf{id})$. An additional encoding $\mathcal{E}[\![\_]\!]_{(\_)} : \mathcal{E} \times \mathcal{N} \to \mathrm{B}\textsc{g}(\Sigma_{\mathsf{Act}})$ 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 $C = \left\langle\!\!\left\langle A, A' \mid \epsilon \right\rangle\!\!\right\rangle$ with

$$\begin{aligned} A &= ([x \mapsto 2] \vdash \mathtt{ready}(x); \mathtt{send}(a', x); \mathtt{nil})_a \\ A' &= [\epsilon \vdash \mathtt{send}(a, 3); \mathtt{nil}]_{a'} \end{aligned}$$

$$
\begin{aligned}
\mathcal{A}\left[\!\left[\left\langle\!\left\langle \alpha \mid \mu \right\rangle\!\right\rangle\right]\!\right] &\overset{\text{def}}{=} \mathcal{A}[\![\alpha]\!] \mid \mathsf{Mail}.\mathcal{A}[\![\mu]\!] \mid \mathsf{Sub}.1 \\
\mathcal{A}[\![\epsilon]\!] &\overset{\text{def}}{=} 1 \\
\mathcal{A}[\![\alpha]\!] &\overset{\text{def}}{=} \mathcal{A}[\![A]\!] \mid \mathcal{A}[\![\alpha']\!] && \text{with } \alpha = A, \alpha' \\
\mathcal{A}[\![\mu]\!] &\overset{\text{def}}{=} \mathcal{A}[\![m]\!] \mid \mathcal{A}[\![\mu']\!] && \text{with } \mu = m, \mu' \\
\mathcal{A}[\![m]\!] &\overset{\text{def}}{=} \mathsf{M}_a.\mathsf{Int}^v.1 && \text{with } m = \langle a \Leftarrow v \rangle \\
\mathcal{A}[\![A]\!] &\overset{\text{def}}{=} \mathsf{A}_a.(\mathsf{E}.\mathcal{E}[\![E]\!]_\varnothing \mid \mathcal{A}[\![b]\!]) && \text{with } A \in \mathit{actor} \\
\mathcal{A}[\![\texttt{send}(a,e);b]\!] &\overset{\text{def}}{=} \mathsf{Snd}.(\mathsf{M}_a.\mathcal{A}[\![e]\!] \mid \mathcal{A}[\![b]\!]) \\
\mathcal{A}[\![\texttt{ready}(x);b]\!] &\overset{\text{def}}{=} \mathsf{Rdy}./x\,(\mathsf{X}_x.1 \mid \mathcal{A}[\![b]\!]) \\
\mathcal{A}[\![\texttt{new}(b);b']\!] &\overset{\text{def}}{=} \mathsf{New}.(\mathsf{A'}_a.\mathcal{A}[\![b]\!] \mid \mathcal{A}[\![b']\!]) && \text{with } a \text{ fresh} \\
\mathcal{A}[\![f;b]\!] &\overset{\text{def}}{=} \mathsf{Fun}.(\mathcal{A}[\![f]\!] \mid \mathcal{A}[\![b]\!]) \\
\mathcal{A}[\![\texttt{nil}]\!] &\overset{\text{def}}{=} \mathsf{Nil}.1 \\
\mathcal{A}[\![v]\!] &\overset{\text{def}}{=} \mathsf{Int}^v.1 \\
\mathcal{A}[\![x]\!] &\overset{\text{def}}{=} \mathsf{X}_x.1
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{E}[\![E]\!]_N &\overset{\text{def}}{=} \mathcal{E}[\![E']\!]_{N\cup\{x\}} \mid \mathcal{E}[\![x \mapsto v]\!]_N && \text{with } E = E'[x \mapsto v] \\
\mathcal{E}[\![x \mapsto v]\!]_N &\overset{\text{def}}{=} \begin{cases} \mathsf{N}^x.\mathsf{Int}^v.1 & \text{if } x \notin N \\ 1 & \text{otherwise} \end{cases} \\
\mathcal{E}[\![\epsilon]\!]_N &\overset{\text{def}}{=} 1
\end{aligned}
$$

Fig. 3. Encodings $\mathcal{A}[\![\_]\!]$ and $\mathcal{E}[\![\_]\!]_{(\_)}$.

The corresponding bigraphical encoding is as follows:

$$
\begin{aligned}
\mathcal{A}[\![A]\!] &= \mathsf{A}_a.(\mathsf{E}.\mathsf{N}^x.\mathsf{Int}^2.1 \mid \mathsf{Rdy}./x\,(\mathsf{X}_x.1 \mid \mathsf{Snd}.(\mathsf{M}_{a'}.\mathsf{X}_x.1 \mid \mathsf{Nil}.1))) \\
\mathcal{A}[\![A']\!] &= \mathsf{A}_{a'}.(\mathsf{E}.1 \mid \mathsf{Snd}.(\mathsf{M}_a.\mathsf{Int}^3.1 \mid \mathsf{Nil}.1)) \\
\mathcal{A}[\![C]\!] &= \mathcal{A}[\![A]\!] \mid \mathcal{A}[\![A']\!] \mid \mathsf{Mail}.1 \mid \mathsf{Sub}.1
\end{aligned}
$$

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 Σ_{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 $\langle \mathbf{nil} : a\rangle$ removes an empty actor from the current configuration. It is encoded by the following bigraphical reaction rule:

$$\mathsf{R}_{\mathsf{nil}} \stackrel{\text{def}}{=} \mathsf{A}_a.(\mathsf{Nil}.1 \mid \mathsf{id}) \longrightarrow a \mid 1$$

with instantiation map $\eta = []$ and interface $\mathsf{e} \to \mathsf{a}$. This means that the contents of site 0 (*i.e.* the site contained by ion A_a indicated by 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 $\mathsf{R}_{\mathsf{nil}}$ is given in Figure 4. The node of control Nil is indicated by a solid red box.

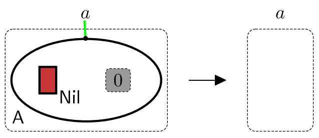


Fig. 4. Bigraphical encoding of rule $\langle \mathbf{nil} : a\rangle$.

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

$$\mathsf{R}_{\mathsf{new}} \stackrel{\text{def}}{=} \mathsf{A}_a.(\mathsf{New}.(\mathsf{A}'_{a'} \mid \mathsf{id}) \mid \mathsf{id}) \longrightarrow \mathsf{A}_a.(\mathsf{id} \mid \mathsf{id}) \mid \mathsf{A}_{a'}.(\mathsf{id} \mid \mathsf{id})$$

with $\eta = [\{1\}, \{2\}, \{0\}, \{2\}]$ and interface $\mathsf{bbe} \to \langle \mathsf{a}, \{a, a'\}\rangle$. The duplication of site 2 encodes the duplication of the local environment.

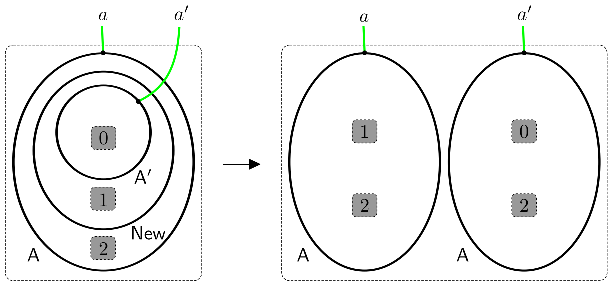


Fig. 5. Bigraphical encoding of rule $\langle \mathbf{new} : a, a'\rangle$.

Rule $\langle \mathbf{snd} : a, \langle a' \Leftarrow v\rangle\rangle$ defines an asynchronous send action. This behaviour is encoded by moving the message contained by a 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

$$\mathsf{R_{snd}} \overset{\text{def}}{=} \mathsf{A}_a.(\mathsf{Snd}.(\mathsf{M}_{a'}.\mathsf{Int}^v.1 \mid \mathsf{id}) \mid \mathsf{id}) \mid \mathsf{Mail} \longrightarrow \mathsf{A}_a.(\mathsf{id} \mid \mathsf{id}) \mid \mathsf{Mail}.(\mathsf{M}_{a'}.\mathsf{Int}^v.1 \mid \mathsf{id})$$

with $\eta = [\{0\}, \{1\}, \{2\}]$ and interface $\mathsf{bem} \to \langle \mathsf{ax}, \{a, a'\}\rangle$. 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.

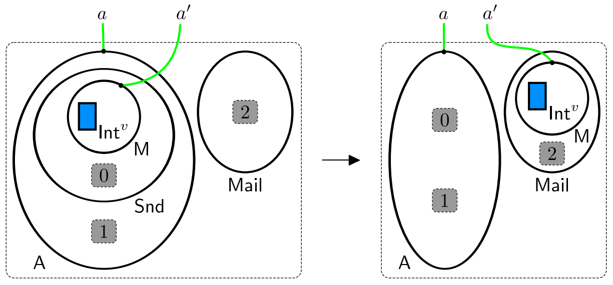


Fig. 6. Bigraphical encoding of rule $\langle \mathbf{snd} : a, \langle a' \Leftarrow v\rangle\rangle$.

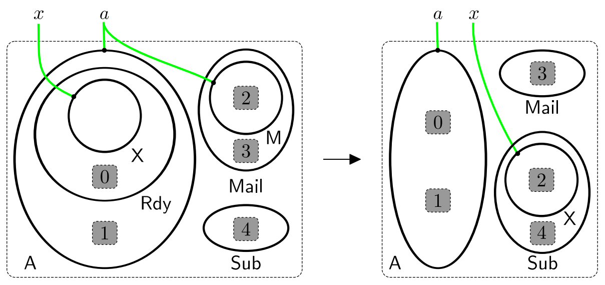
Rule $\langle \mathbf{rcv} : a, \langle a \Leftarrow v\rangle\rangle$ defines the behaviour of an actor when a message is received. The following three reaction rules are required for the encoding:

$$\begin{aligned}\mathsf{R_{rcv}} &\overset{\text{def}}{=} \mathsf{A}_a.(\mathsf{Rdy}.(\mathsf{X}_x.1 \mid \mathsf{id}) \mid \mathsf{id}) \mid \mathsf{Mail}.(\mathsf{M}_a \mid \mathsf{id}) \mid \mathsf{Sub}\\ &\longrightarrow \mathsf{A}_a.(\mathsf{id} \mid \mathsf{id}) \mid \mathsf{Mail} \mid \mathsf{Sub}.(\mathsf{X}_x \mid \mathsf{id})\end{aligned}$$

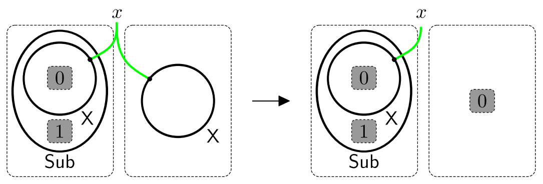
$$\mathsf{R_{sub}} \overset{\text{def}}{=} \mathsf{Sub}.(\mathsf{X}_x \mid \mathsf{id}) \parallel \mathsf{X}_x.1 \longrightarrow \mathsf{Sub}.(\mathsf{X}_x \mid \mathsf{id}) \parallel \mathsf{id}$$

$$\mathsf{R_{rem}} \overset{\text{def}}{=} \mathsf{Sub}.(/x\, \mathsf{X}_x \mid \mathsf{id}) \longrightarrow \mathsf{Sub}$$

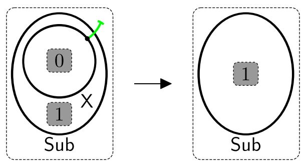
The corresponding graphical notation is in Figure 7. Reaction rule $\mathsf{R_{rcv}}$ consumes the Rdy-node in the left-hand side and initiates a substitution of the free occurrences of x by placing X_x inside Sub. Moreover, the value carried by message M_a is placed inside ion X_x in the right-hand side. This is specified precisely by instantiation map $\eta_{\mathsf{rcv}} = [\{0\}, \{1\}, \{3\}, \{2\}, \{4\}]$. In particular, sites 3 and 4 are swapped to represents the value contained in the message moving into identifier X_x. Observe that since messages are placed in the Mail-node by rule $\mathsf{R_{snd}}$ they are guaranteed to always contain a v-node. The interface is given by $\mathsf{R_{rcv}} : \mathsf{bevmi} \to \langle \mathsf{axs}, \{x, a\}\rangle$. The previous rule only initiates the substitution of the free occurrences of x in the behaviour of the actor. Reaction rule $\mathsf{R_{sub}} : \mathsf{vi} \to \langle \mathsf{si}, \{x\}\rangle$ needs to be applied to actually replace each occurrence of x. By definition of encoding $\mathcal{A}[\![\_]\!]$, all the free occurrences of x are linked to an ion X_x inside the 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 $\eta_{\mathsf{sub}} = [\{0\},\{1\},\{0\}]$. 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 $\mathsf{R}_{\mathsf{rem}} : \mathsf{vi} \to \mathsf{s}$. In this case, the instantiation map is $\eta_{\mathsf{rem}} = [\{1\}]$ because site 0, *i.e.* the content of ion X_x is discarded. In order to force rules $\mathsf{R}_{\mathsf{sub}}$ and $\mathsf{R}_{\mathsf{rem}}$ 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.



(a) Reaction rule $\mathsf{R}_{\mathsf{rcv}}$



(b) Reaction rule $\mathsf{R}_{\mathsf{sub}}$



(c) Reaction rule $\mathsf{R}_{\mathsf{rem}}$

Fig. 7. Bigraphical encoding of rule $\langle \mathbf{rcv} : a, \langle a \Leftarrow v\rangle\rangle$.

Finally, rule $\langle \mathbf{fun} : a\rangle$ models the execution of a local computation step. Since expressions f and transition relation $\to_\lambda$ 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 $\mathcal{E}[\![x \mapsto v]\!]_\varnothing = \mathsf{N}^x.\mathsf{Int}^v.1$. 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 $\mathcal{A}[\![f]\!] = \mathsf{N}^x.\mathsf{Int}^4.1 \mid \mathsf{N}^y.\mathsf{Int}^3.1$. At this point, the following three bigraphical reaction rules can be specified:

$$\mathsf{R}_{\mathsf{fun}} \overset{\text{def}}{=} \mathsf{A}_a.(\mathsf{Fun} \mid \mathsf{id}) \longrightarrow \mathsf{A}_a.(\mathsf{id} \mid \mathsf{id})$$

$$\mathsf{R}_{\mathsf{upd}} \overset{\text{def}}{=} \mathsf{A}_a.(\mathsf{Fun}.(\mathsf{N}^x.\mathsf{Int}^v.1 \mid \mathsf{id}) \mid \mathsf{E}.(\mathsf{N}^x \mid \mathsf{id})) \longrightarrow \mathsf{A}_a.(\mathsf{Fun} \mid \mathsf{E}.(\mathsf{N}^x.\mathsf{Int}^v.1 \mid \mathsf{id}))$$

$$\mathsf{R}_{\mathsf{add}} \overset{\text{def}}{=} \mathsf{A}_a.(\mathsf{Fun}.(\mathsf{N}^x.\mathsf{Int}^v.1 \mid \mathsf{id}) \mid \mathsf{E}) \longrightarrow \mathsf{A}_a.(\mathsf{Fun} \mid \mathsf{E}.(\mathsf{N}^x.\mathsf{Int}^v.1 \mid \mathsf{id}))$$

In the first one, the Fun-node is removed, the instantiation map is the identity function and the interface is $\mathsf{be} \to \langle \mathsf{a}, \{a\}\rangle$. 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 $\eta_{\mathsf{upd}} = [\{0\}, \{2\}]$ because the old value for identifier N^x (*i.e.* site 1) is discarded. The interface is $\mathsf{bnvn} \to \langle \mathsf{a}, \{a\}\rangle$. In the right-hand side the new value Int^v 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 $\mathsf{bnn} \to \langle \mathsf{a}, \{a\}\rangle$. Again the instantiation map is the identity function. Note that in order to have the expected behaviour, $\mathsf{R}_{\mathsf{upd}}$ has to be applied before $\mathsf{R}_{\mathsf{add}}$ and both have to be applied before $\mathsf{R}_{\mathsf{fun}}$. We formalise this by defining priorities over the reaction rules in the encoding as follows:

$$\begin{aligned}
\mathfrak{P}_0 &= \{\mathsf{R}_{\mathsf{nil}}, \mathsf{R}_{\mathsf{new}}, \mathsf{R}_{\mathsf{snd}}, \mathsf{R}_{\mathsf{rcv}}, \mathsf{R}_{\mathsf{fun}}\} \\
\mathfrak{P}_1 &= \{\mathsf{R}_{\mathsf{sub}}, \mathsf{R}_{\mathsf{rem}}\} \\
\mathfrak{P}_2 &= \{\mathsf{R}_{\mathsf{add}}\} \\
\mathfrak{P}_3 &= \{\mathsf{R}_{\mathsf{upd}}\}
\end{aligned}$$

where $\mathfrak{P}_i < \mathfrak{P}_j$ if $i < j$.

The PBRS defining our encoding of the actor operational semantics is given by $(\mathrm{B}\textsc{g}(\Sigma_{\mathsf{Act}}), \mathcal{R})$ with $\mathcal{R} = \bigcup_{i<4} \mathfrak{P}_i$.

Example 2. Take bigraph $\mathcal{A}[\![C]\!] = S_0$ defined in Example 1. Since the node of control Mail is empty, only reaction rule $\mathsf{R}_{\mathsf{snd}}$ can be applied:

$$S_0 \xrightarrow[\mathsf{snd}]{} \mathcal{A}[\![A]\!] \mid \mathsf{A}_{a'}.(\mathsf{E}.1 \mid \mathsf{Nil}.1) \mid \mathsf{Mail}.(\mathsf{M}_a.\mathsf{Int}^3.1) \mid \mathsf{Sub}.1 = S_1$$

Then, either rule $\mathsf{R}_{\mathsf{rcv}}$ or rule $\mathsf{R}_{\mathsf{nil}}$ can be applied. We show the transitions when a message is received:

$$\begin{aligned}
S_1 &\xrightarrow[\mathsf{rcv}]{} /x\,(\mathsf{A}_a.(E \mid \mathsf{Snd}.(\mathsf{M}_{a'}.\mathsf{X}_x.1 \mid \mathsf{Nil}.1)) \mid A' \mid \mathsf{Mail}.1 \mid \mathsf{Sub}.\mathsf{X}_x.\mathsf{Int}^3.1) \\
&\xrightarrow[\mathsf{sub}]{} \mathsf{A}_a.(E \mid \mathsf{Snd}.(\mathsf{M}_{a'}.\mathsf{Int}^3.1 \mid \mathsf{Nil}.1)) \mid A' \mid \mathsf{Mail}.1 \mid \mathsf{Sub}./x\,\mathsf{X}_x.\mathsf{Int}^3.1 \\
&\xrightarrow[\mathsf{rem}]{} \mathsf{A}_a.(E \mid \mathsf{Snd}.(\mathsf{M}_{a'}.\mathsf{Int}^3.1 \mid \mathsf{Nil}.1)) \mid A' \mid \mathsf{Mail}.1 \mid \mathsf{Sub}.1 = S_2
\end{aligned}$$

with $E = \mathsf{E}.\mathsf{N}^x.\mathsf{Int}^2.1$ and $A' = \mathsf{A}_{a'}.(\mathsf{E}.1 \mid \mathsf{Nil}.1)$. 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:

$$\begin{aligned} S_2 &\xrightarrow[\mathsf{nil}]{} \mathsf{A}_a.(E \mid \mathsf{Snd}.(\mathsf{M}_{a'}.\mathsf{Int}^3.1 \mid \mathsf{Nil}.1)) \mid \mathsf{Mail}.1 \mid \mathsf{Sub}.1 \\ &\xrightarrow[\mathsf{snd}]{} \mathsf{A}_a.(E \mid \mathsf{Nil}.1) \mid \mathsf{Mail}.\mathsf{M}_{a'}.\mathsf{Int}^3.1 \mid \mathsf{Sub}.1 = S_3 \end{aligned}$$

Finally, actor a is removed from the configuration:

$$S_3 \xrightarrow[\mathsf{nil}]{} a \mid \mathsf{Mail}.\mathsf{M}_{a'}.\mathsf{Int}^3.1 \mid \mathsf{Sub}.1 = S_3$$

4.4 Operational Correspondence

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

$$\begin{aligned} \langle\!\langle \alpha \mid \mu \rangle\!\rangle \xrightarrow{\langle \mathbf{snd}{:}a, \langle a' \Leftarrow v \rangle\rangle} \langle\!\langle \alpha' \mid \mu' \rangle\!\rangle \quad &\text{implies} \quad \mathcal{A}[\![\langle\!\langle \alpha \mid \mu \rangle\!\rangle]\!] \xrightarrow[\mathsf{snd}]{} \mathcal{A}[\![\langle\!\langle \alpha' \mid \mu' \rangle\!\rangle]\!] \\ \langle\!\langle \alpha \mid \mu \rangle\!\rangle \xrightarrow{\langle \mathbf{fun}{:}a \rangle} \langle\!\langle \alpha' \mid \mu' \rangle\!\rangle \quad &\text{implies} \quad \mathcal{A}[\![\langle\!\langle \alpha \mid \mu \rangle\!\rangle]\!] \xrightarrow[\mathsf{FUN}]{} \mathcal{A}[\![\langle\!\langle \alpha' \mid \mu' \rangle\!\rangle]\!] \\ \langle\!\langle \alpha \mid \mu \rangle\!\rangle \xrightarrow{\langle \mathbf{rcv}{:}a, \langle a \Leftarrow v \rangle\rangle} \langle\!\langle \alpha' \mid \mu' \rangle\!\rangle \quad &\text{implies} \quad \mathcal{A}[\![\langle\!\langle \alpha \mid \mu \rangle\!\rangle]\!] \xrightarrow[\mathsf{RCV}]{} \mathcal{A}[\![\langle\!\langle \alpha' \mid \mu' \rangle\!\rangle]\!] \end{aligned}$$

with

$$\begin{aligned} \xrightarrow[\mathsf{FUN}]{} &\overset{\text{def}}{=} \xrightarrow[\mathsf{upd}]{}^{*} \xrightarrow[\mathsf{add}]{}^{*} \xrightarrow[\mathsf{fun}]{} \\ \xrightarrow[\mathsf{RCV}]{} &\overset{\text{def}}{=} \xrightarrow[\mathsf{rcv}]{} \xrightarrow[\mathsf{sub}]{}^{*} \xrightarrow[\mathsf{rem}]{} \end{aligned}$$

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 $\mathcal{A}[\![\_]\!] : \mathit{Act} \to \mathrm{B}\textsc{g}(\Sigma_{\mathsf{Act}})$ to encode the terms of the actor language into sorted bigraphs and a set of prioritised reaction rules $\mathcal{R}$ to match the behaviours specified by the actor operational semantics. Sorting discipline Σ_{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 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

1. Agha, G., Mason, I., Smith, S., Talcott, C.: A foundation for actor computation. Journal of Functional Programming 7(1), 1–72 (1997)
2. Agha, G.: Actors: a model of concurrent computation in distributed systems. MIT Press, Cambridge, MA, USA (1986)
3. Baeten, J., Bergstra, J., Klop, J., Weijland, W.: Term-rewriting systems with rule priorities. Theoretical Computer Science 67, 283–301 (October 1989)
4. Birkedal, L., Debois, S., Elsborg, E., Hildebrandt, T., Niss, H.: Bigraphical models of context-aware systems. In: Foundations of software science and computation structures. pp. 187–201. Springer (2006)
5. Calder, M., Koliousis, A., Sevegnani, M., Sventek, J.: Real-time verification of wireless home networks using bigraphs with sharing. Science of Computer Programming 80, 288–310 (2014)
6. Calder, M., Sevegnani, M.: Modelling IEEE 802.11 CSMA/CA RTS/CTS with stochastic bigraphs with sharing. Formal Aspects of Computing pp. 1–25 (2013)
7. Milner, R.: Bigraphs for petri nets. In: Lectures on Concurrency and Petri Nets, pp. 686–701. Springer (2004)
8. Milner, R.: Local bigraphs and confluence: Two conjectures: (extended abstract). Electronic Notes in Theoretical Computer Science 175(3), 65 – 73 (2007), proceedings of the 13th International Workshop on Expressiveness in Concurrency (EXPRESS 2006)
9. Milner, R.: The Space and Motion of Communicating Agents. Cambridge University Press (2009)
10. Nielsen, B., Agha, G.: Semantics for an actor-based real-time language. Proceedings of the 4th International Workshop on Parallel and Distributed Real-Time Systems pp. 223–228 (1996)
11. Pereira, E., Kirsch, C., Sengupta, R., Borges de Sousa, J.: Bigactors - a model for structure-aware computation. In: 4th International Conference on Cyber-Physical Systems. ACM/IEEE (April 2013)