POLYADIC STOCHASTIC COWS

Stefano Schivo

May 2008

Technical Report # DISI-08-035

Polyadic Stochastic COWS[*]

Stefano Schivo

1 Introduction

Our objective is to extend the process algebra cows presented in [2, 3] by introducing stochastic delays in the language. We will start by presenting a modified version of the semantics of cows which will allow us to ease the subsequent stochastic extension. In order to better explain the changes made to the basic version of the language, we start by presenting the semantics for a monadic (as opposed to polyadic) version of cows, and then move on to our version of the polyadic case. Finally we will present the stochastic extension of polyadic cows.

2 Monadic cows

The syntax of monadic cows is as follows:

$$\begin{array}{lcl} s & ::= & u!w \;\mid\; g \;\mid\; s \mid s \;\mid\; \{\!|s|\!\} \;\mid\; \mathbf{kill}(k) \;\mid\; [\,d\,]s \;\mid\; \mathsf{S}(n_1,\ldots,n_j) \\ g & ::= & \mathbf{0} \;\mid\; p?w.\,s \;\mid\; g+g \end{array}$$

We present in Tables 1 and 2 the operational semantics of monadic cows. We will briefly comment on the semantics and the new auxiliary functions.

As the final objective of this work is the introduction of a stochastic process algebra, we need to constrain the models to produce finitely branching transition systems. This is done so that we will be able to use standard techniques to deal with the Markov processes resulting from the stochastic models. In order to obtain a finitely branching transition system, we use service identifiers instead of service replication and assume that each of these identifiers is guarded in a valid starting process. Another assumption concerns naming and states that no homonymy exists between both bound (i.e. under the scope of a delimiter) and free entities. This last condition is supposed to hold in the starting process and kept valid through the use of functions s_dec and l_dec in rule $\mathsf{ser_{id}}$. These functions are directly drawn from [4] and decorate bound names respectively in the transition label and in the residual service by adding to each entity a finite number of zeroes as superscript.

In addition to the metavariable conventions defined in the original version, we will use a, b to range over $x, y, z, (x), (y), (z), m, n$ (i.e. possibly bound input entities) and h, i

[*]This work has been partially sponsored by the project SENSORIA, IST-2005-016004.

to range over $m, n, (m), (n)$ (i.e. possibly bound output names). Furthermore, we define the (possibly empty) delimiter as

$$t ::= n \mid \varepsilon$$

The writing of the form $[\,t\,]s$ in the residual process of rule com will be intended as s alone if $t = \varepsilon$, and as $[\,n\,]s$ if $t = n$.

The label for communication is of the form $p\ \lfloor\sigma'\rfloor\ a\ h$ and has the following meaning:

- p is the endpoint on which the communication is happening (to improve readability, we merge partner and operation identifiers on a single entity name);
- σ' is the possible substitution to be applied when the appropriate delimiter is encountered (see definition of $\mathcal{M}$ below);
- a is the (possibly bound) variable or name in the request action involved in the communication;
- h is the (possibly bound) name coming from the invoke action involved in the communication.

Notice that the fact that names or variables a and h are bound influences only the application of rules $\mathsf{del_{cl}}$ and $\mathsf{del_{sub}}$, being transparent to all other semantics rules.

As the new semantics is not based on a congruence relation, we need to deal with scope modifications via opening and closing of delimitators. In order to address this issue, we introduce a modified version of the matching function $\mathcal{M}$, which now handles all possible combinations of bound input/output values. Moreover, as in monadic cows all communicated objects only consist of one-element tuples, the matching function used in communications becomes as follows:

$$\mathcal{M}(n,n) = (\emptyset,\emptyset,\varepsilon) \qquad \mathcal{M}(x,n) = (\{^{n}/_{x}\},\emptyset,\varepsilon)$$

$$\mathcal{M}((x),n) = (\emptyset,\{^{n}/_{x}\},\varepsilon) \qquad \mathcal{M}(x,(n)) = (\{^{(n)}/_{x}\},\emptyset,\varepsilon)$$

$$\mathcal{M}((x),(n)) = (\emptyset,\{^{n}/_{x}\},n)$$

Notice that the return value of $\mathcal{M}$ is a triple (σ',σ,t) composed by:

- σ': the substitution to be carried out as soon as the corresponding delimiter is encountered
- σ: the substitution to be carried out directly on the residual process of the communication
- t: the name to be restricted in the residual process

A brief explanation of each case tackled by the function $\mathcal{M}$ is given below:

- $\mathcal{M}(n,n)$: no substitution is needed, as input and output names coincide;

$$\text{(kill)}\ \frac{-}{\mathbf{kill}(k) \xrightarrow{\dagger k} \mathbf{0}} \qquad \text{(req)}\ \frac{-}{p?w.\,s \xrightarrow{p?w} s} \qquad \text{(inv)}\ \frac{-}{p!n \xrightarrow{p!n} \mathbf{0}}$$

$$\text{(prot)}\ \frac{s \xrightarrow{\alpha} s'}{\{\!|s|\!\} \xrightarrow{\alpha} \{\!|s'|\!\}} \qquad \text{(choice)}\ \frac{g_1 \xrightarrow{\alpha} s}{g_1 + g_2 \xrightarrow{\alpha} s}$$

$$\text{(open)}\ \frac{s \xrightarrow{\alpha} s' \quad u \in d(\alpha)}{[\,u\,]s \xrightarrow{\mathsf{op}(\alpha,u)} s'}$$

$$\text{(com)}\ \frac{s_1 \xrightarrow{p?a} s_1{}' \quad s_2 \xrightarrow{p!h} s_2{}' \quad \mathcal{M}(a,h) = (\sigma', \sigma, t) \quad \neg(s_1 \mid s_2) \downarrow_{p,h}^{|\sigma'|+|\sigma|}}{s_1 \mid s_2 \xrightarrow{p\ \lfloor\sigma'\rfloor\ a\ h} [\,t\,](s_1{}'\sigma \mid s_2{}')}$$

Table 1: Semantics for monadic cows (Part 1).

$$(\mathsf{par}_{\mathsf{conf}})\ \frac{s_1 \xrightarrow{p\,\lfloor\sigma\rfloor\,a\,h} {s_1}' \quad \neg s_2 \downarrow^{\#(a,h)}_{p,h}}{s_1 \mid s_2 \xrightarrow{p\,\lfloor\sigma\rfloor\,a\,h} {s_1}' \mid s_2} \qquad (\mathsf{par}_{\mathsf{pass}})\ \frac{s_1 \xrightarrow{\alpha} {s_1}' \quad \alpha \neq p\,\lfloor\sigma\rfloor\,a\,h \quad \alpha \neq \dagger k}{s_1 \mid s_2 \xrightarrow{\alpha} {s_1}' \mid s_2}$$

$$(\mathsf{par}_{\mathsf{kill}})\ \frac{s_1 \xrightarrow{\dagger k} {s_1}'}{s_1 \mid s_2 \xrightarrow{\dagger k} {s_1}' \mid halt(s_2)} \qquad (\mathsf{del}_{\mathsf{pass}})\ \frac{s \xrightarrow{\alpha} s' \quad d \notin d(\alpha) \quad s\downarrow_{kill} \Rightarrow (\alpha = \dagger \text{ or } \alpha = \dagger k)}{[\,d\,]s \xrightarrow{\alpha} [\,d\,]s'}$$

$$(\mathsf{del}_{\mathsf{cl}})\ \frac{s \xrightarrow{p\,\lfloor\{(n)/x\}\rfloor\,x\,n} s'}{[\,x\,]s \xrightarrow{p\,\lfloor\emptyset\rfloor\,x\,n} [\,n\,]s'\{^{n}\!/\!_{x}\}} \qquad (\mathsf{del}_{\mathsf{sub}})\ \frac{s \xrightarrow{p\,\lfloor\{n/x\}\rfloor\,x\,n} s'}{[\,x\,]s \xrightarrow{p\,\lfloor\emptyset\rfloor\,x\,n} s'\{^{n}\!/\!_{x}\}} \qquad (\mathsf{del}_{\mathsf{kill}})\ \frac{s \xrightarrow{\dagger k} s'}{[\,k\,]s \xrightarrow{\dagger} [\,k\,]s'}$$

$$(\mathsf{ser}_{\mathsf{id}})\ \frac{s\{^{m_1,\ldots,m_j}\!/\!_{n_1,\ldots,n_j}\} \xrightarrow{\alpha} s' \quad \mathsf{S}(n_1,\ldots,n_j) = s}{\mathsf{S}(m_1,\ldots,m_j) \xrightarrow{\mathrm{l\_dec}(\alpha)} \mathrm{s\_dec}(\alpha, s')}$$

Table 2: Semantics for monadic cows (Part 2).

- $\mathcal{M}(x, n)$: a substitution will be performed when the delimiter for variable x will be encountered (rule $\mathsf{del_{sub}}$), so that the value of x will be substituted on its entire scope;

- $\mathcal{M}((x), n)$: input variable x is bound (i.e. its delimiter has already been encountered, and thus the scope of x is contained into the scope of n), so the substitution $\{{}^{n}\!/\!_{x}\}$ is performed directly on the residual process and no delimiter needs to be added;

- $\mathcal{M}(x, (n))$: output name n is bound (its scope is contained in the scope of x), so the substitution needs to be performed when the delimiter for x is found, and that delimiter will be substituted by a delimiter for n (rule $\mathsf{del_{cl}}$), effectively closing the scope of n;

- $\mathcal{M}((x), (n))$: both delimiters have already been encountered, so the substitution has to be performed directly on the residual process and the scope of n needs to be closed at the same time (so a delimiter for n is added to the residual process).

Cases in which the input *name* is bound ($\mathcal{M}((n), n)$ and $\mathcal{M}((n), (n))$) are not considered as they are not possible, while cases of the form $\mathcal{M}(n, m)$ with $n \neq m$ are not defined as no match subsists.

We rely on the application of rule open (which in turn rests on function op defined in Table 3) for the opening of scopes. Notice that this rule allows us to take into account also the case in which the scope of the variable contains the scope of the name, which in turn contains *both* communicating processes. For example, consider the following process:

$$s = [\,x\,]\,([\,n\,]\,(p?x.\,\mathbf{0} \mid p!n) \mid q!x)$$

Notice that the basic monadic semantics proposed in [4] does not allow to correctly infer the transition $s \xrightarrow{p\,\lfloor\emptyset\rfloor\,x\,n} [\,n\,]\,(\mathbf{0} \mid \mathbf{0} \mid q!n)$, while the one from Tables 1, 2 does. This is because the opening of the scope of a name involved in a communication when the communication rule has already been applied was not envisaged in [4].

$$\mathsf{op}(p!n, m) = \begin{cases} p!(n) & \text{if } n = m \\ p!n & \text{if } n \neq m \end{cases}$$

$$\mathsf{op}(p?x, y) = \begin{cases} p?(x) & \text{if } x = y \\ p?x & \text{if } x \neq y \end{cases}$$

$$\mathsf{op}(p\,\lfloor\{{}^{n}\!/\!_{x}\}\rfloor\,x\,n, m) = \begin{cases} p\,\lfloor\{{}^{(n)}\!/\!_{x}\}\rfloor\,x\,(n) & \text{if } n = m \\ p\,\lfloor\{{}^{n}\!/\!_{x}\}\rfloor\,x\,n & \text{if } n \neq m \end{cases}$$

Table 3: Definition of function op.

To make notation shorter, we add an "utility" function for counting the number of substitutions needed for the match of two tuples, defined as follows:

$$\#(a,h) = \begin{cases} |\sigma'| + |\sigma| & \text{if } \mathcal{M}(a,h) = (\sigma', \sigma, t) \\ \infty & \text{otherwise} \end{cases}$$

In order to correctly check that the matching components in a communication are the best possible, we make use of the same predicate as the one introduced in [3], with the appropriate changes to adapt it to our context:

$$\frac{-}{p?n.\, s \downarrow^{1}_{p,n}} \qquad \frac{s \downarrow^{c}_{p,n} \quad d \neq p}{[\, d\,] s \downarrow^{c}_{p,n}} \qquad \frac{s \downarrow^{c}_{p,n}}{\{\!|s|\!\} \downarrow^{c}_{p,n}}$$

$$\frac{g_1 \downarrow^{c}_{p,n} \vee g_2 \downarrow^{c}_{p,n}}{(g_1 + g_2) \downarrow^{c}_{p,n}} \qquad \frac{s_1 \downarrow^{c}_{p,n} \vee s_2 \downarrow^{c}_{p,n}}{(s_1 \mid s_2) \downarrow^{c}_{p,n}}$$

$$\frac{s\{^{m_1,\ldots,m_j}/_{n_1,\ldots,n_j}\} \downarrow^{c}_{p,n} \quad \mathsf{S}(n_1,\ldots,n_j) = s}{\mathsf{S}(m_1,\ldots,m_j) \downarrow^{c}_{p,n}}$$

The predicate $s \downarrow^{c}_{p,n}$ can thus be read as "service s can perform a request action on endpoint p, matching with h through (strictly) less than c substitutions". As a shortcut, and in order to simplify the definition, we establish that $s \downarrow^{c}_{p,n} \Leftrightarrow s \downarrow^{c}_{p,(n)}$.

Predicate $\downarrow_{kill}$ is defined the same way as the one defined in [3].

Function d is defined the same way as in [3] and consistently extended so to include our additions in naming conventions.

3 Polyadic cows

The next step is to introduce polyadic communications into the base language.

The grammar of the language has to be modified in order to deal with tuples of values. The syntax of polyadic cows becomes as follows:

$$\begin{array}{lcl} s & ::= & u!\tilde{w} \mid g \mid s \mid s \mid \{\!|s|\!\} \mid \mathbf{kill}(k) \mid [\,d\,]s \mid \mathsf{S}(n_1, \ldots, n_j) \\ g & ::= & \mathbf{0} \mid p?\tilde{w}.\,s \mid g + g \end{array}$$

We indicate tuples by the same metavariables used in the monadic context, to which we add a characterising sign: e.g., a tuple of input entities will be called $\tilde{w}$. A tuple made up by a single element can be written as the element itself: i.e., if $\tilde{w} = \langle x \rangle$, then x can be used instead of $\tilde{w}$.

In order to correctly define the matching function $\mathcal{M}$ in the polyadic context, we need to introduce proper operators for the joining of substitution functions and tuples.

Definition 1 *Given the two substitution functions σ_1 and σ_2, the* disjoint union *of the functions, written $\sigma_1 \uplus \sigma_2$, is the union of σ_1 and σ_2 when they have disjoint domains.*

Definition 2 *Given the two tuples $\tilde{t_1} = \langle t_{1,1}, t_{1,2}, \ldots, t_{1,i} \rangle$ and $\tilde{t_2} = \langle t_{2,1}, t_{2,2}, \ldots, t_{2,j} \rangle$, the* concatenation *of the two tuples, written $\tilde{t_1} :: \tilde{t_2}$, is the tuple obtained by juxtaposing the two original tuples: $\langle t_{1,1}, t_{1,2}, \ldots, t_{1,i}, t_{2,1}, t_{2,2}, \ldots, t_{2,j} \rangle$.*

The substitution of entities in tuples is defined the trivial way and written as $\tilde{w}\{^n/_x\}$.

The matching function is defined as shown in Table 4 by adding a rule which will allow to deal with tuples of entities. In order to deal with the multiple substitutions generated by function $\mathcal{M}$ we extend also function op through an auxiliary function named $\mathsf{op_s}$ as shown in Table 5.

$$\begin{array}{lclclcl} \mathcal{M}(n,n) & = & (\emptyset, \emptyset, \varepsilon) & \quad & \mathcal{M}(x,n) & = & (\{^n/_x\}, \emptyset, \varepsilon) \\ \mathcal{M}((x),n) & = & (\emptyset, \{^n/_x\}, \varepsilon) & \quad & \mathcal{M}(x,(n)) & = & (\{^{(n)}/_x\}, \emptyset, \varepsilon) \end{array}$$

$$\mathcal{M}((x),(n)) = (\emptyset, \{^n/_x\}, n)$$

$$\frac{\mathcal{M}(a_1,h_1) = (\sigma_1{}', \sigma_1, \tilde{t_1}) \quad \mathcal{M}\left(\tilde{a_2}, \tilde{h_2}\right) = (\sigma_2{}', \sigma_2, \tilde{t_2})}{\mathcal{M}\left(\langle a_1 \rangle :: \tilde{a_2}, \langle h_1 \rangle :: \tilde{h_2}\right) = (\sigma_1{}' \uplus \sigma_2{}', \sigma_1 \uplus \sigma_2, \tilde{t_1} :: \tilde{t_2})}$$

Table 4: Definition of the matching function $\mathcal{M}$.

As can be seen in Table 4, the complete definition of function $\mathcal{M}$ produces a tuple $\tilde{t}$ of names to be bound in the residual process in com rule. This implies that definition of metavariable $\tilde{t}$ is now as follows:

$$\tilde{t} ::= \tilde{n} \mid \varepsilon$$

$$
\begin{array}{rcl}
\mathsf{op}(p!\tilde{h}, n) & = & p!\tilde{h}' \text{ where } \tilde{h}' = \tilde{h}\{^{(n)}/_{n}\} \\
\mathsf{op}(p?\tilde{a}, x) & = & p?\tilde{a}' \text{ where } \tilde{a}' = \tilde{a}\{^{(x)}/_{x}\} \\
\mathsf{op}(p\,\lfloor\sigma\rfloor\,\tilde{a}\,\tilde{h}, n) & = & p\,\lfloor\mathsf{op_s}(\sigma, n)\rfloor\,\tilde{a}\,\tilde{h}' \text{ where } \tilde{h}' = \tilde{h}\{^{(n)}/_{n}\} \\
\\
\mathsf{op_s}(\emptyset, n) & = & \emptyset \\
\mathsf{op_s}(\{^{n}/_{x}\} \uplus \sigma, m) & = & \begin{cases} \{^{(n)}/_{x}\} \uplus \mathsf{op_s}(\sigma, m) & \text{if } n = m \\ \{^{n}/_{x}\} \uplus \mathsf{op_s}(\sigma, m) & \text{if } n \neq m \end{cases}
\end{array}
$$

Table 5: Complete definition of functions op and $\mathsf{op_s}$.

with the same "reading rules" as before.

The function for counting the number of substitutions in a match is extended to the polyadic setting as follows:

$$
\#(\tilde{a}, \tilde{h}) = \begin{cases} |\sigma'| + |\sigma| & \text{if } \mathcal{M}\big(\tilde{a}, \tilde{h}\big) = (\sigma', \sigma, \tilde{t}) \\ \infty & \text{otherwise} \end{cases}
$$

The introduction of tuples implies that the predicate $\cdot \downarrow^{c}_{p,\tilde{h}}$ has to be updated as follows:

$$
\frac{\#(\tilde{w}, \tilde{h}) < c}{p?\tilde{w}.\, s \downarrow^{c}_{p,\tilde{h}}} \qquad \frac{s \downarrow^{c}_{p,\tilde{h}} \quad d \neq p}{[\,d\,]s \downarrow^{c}_{p,\tilde{h}}} \qquad \frac{s \downarrow^{c}_{p,\tilde{h}}}{\{\!|s|\!\} \downarrow^{c}_{p,\tilde{h}}}
$$

$$
\frac{g_1 \downarrow^{c}_{p,\tilde{h}} \lor g_2 \downarrow^{c}_{p,\tilde{h}}}{(g_1 + g_2) \downarrow^{c}_{p,\tilde{h}}} \qquad \frac{s_1 \downarrow^{c}_{p,\tilde{h}} \lor s_2 \downarrow^{c}_{p,\tilde{h}}}{(s_1 \mid s_2) \downarrow^{c}_{p,\tilde{h}}}
$$

$$
\frac{s\{^{m_1,\ldots,m_j}/_{n_1,\ldots,n_j}\} \downarrow^{c}_{p,\tilde{h}} \quad \mathsf{S}(n_1, \ldots, n_j) = s}{\mathsf{S}(m_1, \ldots, m_j) \downarrow^{c}_{p,\tilde{h}}}
$$

We show the semantics for polyadic cows in Tables 6 and 7.

$$\text{(kill)}\ \frac{-}{\mathbf{kill}(k) \xrightarrow{\dagger k} \mathbf{0}} \qquad \text{(req)}\ \frac{-}{p?\tilde{w}.\, s \xrightarrow{p?\tilde{w}} s} \qquad \text{(inv)}\ \frac{-}{p!\tilde{n} \xrightarrow{p!\tilde{n}} \mathbf{0}}$$

$$\text{(prot)}\ \frac{s \xrightarrow{\alpha} s'}{\{\!|s|\!\} \xrightarrow{\alpha} \{\!|s'|\!\}} \qquad \text{(choice)}\ \frac{g_1 \xrightarrow{\alpha} s}{g_1 + g_2 \xrightarrow{\alpha} s}$$

$$\text{(open)}\ \frac{s \xrightarrow{\alpha} s' \quad u \in d(\alpha)}{[\,u\,]s \xrightarrow{\mathsf{op}(\alpha,u)} s'}$$

$$\text{(com)}\ \frac{s_1 \xrightarrow{p?\tilde{a}} s_1{}' \quad s_2 \xrightarrow{p!\tilde{h}} s_2{}' \quad \mathcal{M}\big(\tilde{a},\tilde{h}\big) = (\sigma',\sigma,\tilde{t}) \quad \neg(s_1 \mid s_2) \downarrow^{|\sigma'|+|\sigma|}_{p,\tilde{h}}}{s_1 \mid s_2 \xrightarrow{p\,\lfloor\sigma'\rfloor\,\tilde{a}\,\tilde{h}} [\,\tilde{t}\,](s_1{}'\sigma \mid s_2{}')}$$

Table 6: Semantics for polyadic cows (Part 1).

$$(\mathsf{par}_{\mathsf{conf}})\ \frac{s_1 \xrightarrow{p\,\lfloor\sigma\rfloor\,\tilde{a}\,\tilde{h}} s_1{}' \quad \neg s_2 \downarrow^{\#(\tilde{a},\tilde{h})}_{p,\tilde{h}}}{s_1 \mid s_2 \xrightarrow{p\,\lfloor\sigma\rfloor\,\tilde{a}\,\tilde{h}} s_1{}' \mid s_2} \qquad (\mathsf{par}_{\mathsf{pass}})\ \frac{s_1 \xrightarrow{\alpha} s_1{}' \quad \alpha \neq p\,\lfloor\sigma\rfloor\,\tilde{a}\,\tilde{h} \quad \alpha \neq \dagger k}{s_1 \mid s_2 \xrightarrow{\alpha} s_1{}' \mid s_2}$$

$$(\mathsf{par}_{\mathsf{kill}})\ \frac{s_1 \xrightarrow{\dagger k} s_1{}'}{s_1 \mid s_2 \xrightarrow{\dagger k} s_1{}' \mid halt(s_2)} \qquad (\mathsf{del}_{\mathsf{pass}})\ \frac{s \xrightarrow{\alpha} s' \quad d \notin d(\alpha) \quad s{\downarrow_{kill}} \Rightarrow (\alpha = \dagger \text{ or } \alpha = \dagger k)}{[\,d\,]s \xrightarrow{\alpha} [\,d\,]s'}$$

$$(\mathsf{del}_{\mathsf{cl}})\ \frac{s \xrightarrow{p\,\lfloor\sigma \uplus \{(n)/x\}\rfloor\,\tilde{a}\,\tilde{h}} s'}{[\,x\,]s \xrightarrow{p\,\lfloor\sigma\rfloor\,\tilde{a}\,\tilde{h}} [\,n\,]s'\{^n/_x\}} \qquad (\mathsf{del}_{\mathsf{sub}})\ \frac{s \xrightarrow{p\,\lfloor\sigma \uplus \{n/x\}\rfloor\,\tilde{a}\,\tilde{h}} s'}{[\,x\,]s \xrightarrow{p\,\lfloor\sigma\rfloor\,\tilde{a}\,\tilde{h}} s'\{^n/_x\}} \qquad (\mathsf{del}_{\mathsf{kill}})\ \frac{s \xrightarrow{\dagger k} s'}{[\,k\,]s \xrightarrow{\dagger} [\,k\,]s'}$$

$$(\mathsf{ser}_{\mathsf{id}})\ \frac{s\{^{m_1,\ldots,m_j}/_{n_1,\ldots,n_j}\} \xrightarrow{\alpha} s' \quad \mathsf{S}(n_1,\ldots,n_j) = s}{\mathsf{S}(m_1,\ldots,m_j) \xrightarrow{\mathrm{l\_dec}(\alpha)} \mathrm{s\_dec}(\alpha, s')}$$

Table 7: Semantics for polyadic cows (Part 2).

4 Polyadic Stochastic COWS

4.1 Syntax

Following the common approach in stochastic extensions [1, 5, 4], we add a stochastic delay to each basic action of the language. These delays are identified with exponentially distributed random variables, the rates of which are included in the syntax of basic actions:

$$\begin{array}{lll} s & ::= & (u!\tilde{w}, \gamma) \mid g \mid s \mid s \mid \{\!|s|\!\} \mid (\mathbf{kill}(k), \lambda) \mid [\,d\,]s \mid \mathsf{S}(n_1, \ldots, n_j) \\ g & ::= & \mathbf{0} \mid (p?\tilde{w}, \delta).\, s \mid g + g \end{array}$$

4.2 Semantics for Polyadic Stochastic COWS

The semantics for Polyadic Stochastic COWS is shown in Tables 13 and 14. We will now comment on the most relevant novelties with respect to the non-stochastic version of the language.

The stochastic execution step of a closed service is

$$s \xrightarrow[\rho]{\vartheta(\alpha)} s'$$

with either $\alpha = \dagger$ or $\alpha = p\ \lfloor \emptyset \rfloor\ \tilde{a}\ \tilde{h}$ and where

- ρ is the "basic" rate of the action (i.e. not the apparent one. In the case of a communication, it is a couple of two values, one for each of the interacting components' rate),
- ϑ is inspired by the same approach in [4] and used in rules choice_0 and choice_1 in order to distinguish between cases which would otherwise result identical[1].

The rate of such transition is calculated as follows:

$$\mathtt{rate}\left(s \xrightarrow[\rho]{\vartheta(\alpha)} s'\right) = \begin{cases} \text{(see (1))} & \text{if } \alpha = p\ \lfloor \emptyset \rfloor\ \tilde{a}\ \tilde{h} \\ \rho & \text{if } \alpha = \dagger \end{cases}$$

4.3 Calculating the rate of a communication

In [2] the communication between two COWS processes is presented as an asynchronous (as invoke actions are independent processes), asymmetric (since a request is adapted to its corresponding invoke through the matching function $\mathcal{M}$) event. Following this same approach, we assume that request actions are dependent from invoke actions: i.e., the choice of the "transmitting" invoke action determines the set of possible "receiving" request actions. This happens because the set of request actions matching with the

[1]As an example, consider the process $(p!n, \delta) \mid (p?n, \gamma).\,\mathbf{0} + (p?n, \gamma).\,\mathbf{0}$. We would have two identical transitions departing from this process, and both leading to the same residue $\mathbf{0} \mid \mathbf{0}$. Would there not be a way to distinguish between these two transitions, the generation of the CTMC would become troublesome.

smallest number of substitutions is different from one invoke action to another. This is best shown in a simple example. Consider the service

$$S = [X, Y]((\underbrace{p!\langle 1, 2\rangle, \delta_1)}_{A} \mid \underbrace{(p!\langle 1, 3\rangle, \delta_2)}_{B} \mid \underbrace{(p?\langle 1, X\rangle, \gamma_1).\,\mathbf{0}}_{C} \mid \underbrace{(p?\langle Y, 3\rangle, \gamma_2).\,\mathbf{0}}_{D})$$

The matching paradigm states that the only possible communications are the following:

- A - C (matching 1 with 1 and substituting X by 2)
- B - C (matching 1 with 1 and substituting X by 3)
- B - D (substituting Y by 1 and matching 3 with 3)

Note that, as each of the two communications B - C and B - D makes one substitution, they are equally viable from the non-stochastic point of view. We can see from the example that the choice of the invoke action also determines the set of possible communications: when choosing to make a communication involving process A, there is only one possible communication (A - C), while when choosing process B the possible communications are two (B - C and B - D).

The rate of a communication event is obtained multiplying the apparent rate[2] of the communication by the probability to choose the two participants among all possible competitors for the same communication. Adopting a classical way of approximating exponential rates [1], we take the apparent rate of a communication to be the minimum between the apparent rates of the participating processes (i.e., the communication proceeds at the speed of the "slowest" of the participants). So, the ideal formula for the rate of a communication between processes A and B has the following form:

$$\texttt{rate}\,(\text{communication}) = P(A \cap B) \cdot \texttt{min}\,(\texttt{rate}\,(A), \texttt{rate}\,(B))$$

where $P(A \cap B)$ is the probability to have both processes A and B involved in the communication.

As we consider request actions to be dependent from invoke actions, the probability to choose a pair invoke-request has to be computed via the conditional probability formula:

$$P(A \cap B) = P(B) \cdot P(A \mid B)$$

In our case, this means that the probability to have a match between request $A = (p?\tilde{w}, \gamma)$ and invoke $B = (p!\tilde{u}, \delta)$ is given by the product of the probability to have chosen B among all possible invoke actions on endpoint p and the probability to choose A among the request actions made available by the choice of B. In the example above, the probability to choose communication between B and C is calculated as follows:

$$\begin{array}{rcl} P(B \cap C) & = & P(B) \cdot P(C \mid B) \\ & = & \frac{\delta_2}{\delta_1 + \delta_2} \cdot \frac{\gamma_1}{\gamma_1 + \gamma_2} \end{array}$$

[2]We use the term *apparent rates* because they are the rates at which respective actions would be seen happening by an external observer (i.e. an observer who cannot distinguish between individual actions).

while the probability to choose communication between A and C is basically the probability to choose A (as C becomes the only request action available for the communication):

$$\begin{array}{rcl} P(A \cap C) & = & P(A) \cdot P(C \mid A) \\ & = & \frac{\delta_1}{\delta_1+\delta_2} \cdot 1 \end{array}$$

Finally, we set a way to compute the apparent rates of invoke and request actions involved in a communication.

The apparent rate of an invoke action taking part in a communication internal to service s is given by the sum of the rates of all invoke actions in s able to perform a communication on the same endpoint on which the communication happens. Function `inv` (defined in Table 8) is used for this aim.

Definition 3 *Request action $(p'?\tilde{w}, \gamma)$ is* activated *by invoke action $(p!\tilde{u}, \delta)$ if the communication between the two actions is possible.*

So, in a specific service s, an unguarded request action $(p'?\tilde{w}, \gamma)$ is activated by invoke action $(p!\tilde{u}, \delta)$ if $p = p'$ and $\#(\tilde{w}, \tilde{u})$ is minimal w.r.t. any other unguarded request action $(p'?\tilde{w}', \gamma')$ in s for which $\mathcal{M}(\tilde{w}', \tilde{u})$ is defined.

Definition 4 *The* best-matching set *for invoke action $(p!\tilde{u}, \delta)$ in service s is a set of unguarded request actions on endpoint p, all matching with tuple $\tilde{u}$ through the minimal number of substitutions among all request actions in s activated by invoke $(p!\tilde{u}, \delta)$.*

The apparent rate of a request action $(p?\tilde{w}, \gamma)$ involved in a communication inside service s is calculated as follows:

- consider a set $\mathcal{B}$ of best-matching sets, each relative to a different unguarded invoke action in s on channel p, and such that $(p?\tilde{w}, \gamma)$ is contained in all sets in $\mathcal{B}$;
- for each set $N \in \mathcal{B}$, sum up the rates of all actions contained in N;
- multiply each of such sums by the probability to select the corresponding activating invoke action (i.e. the rate of such invoke action divided by $\texttt{inv}(s, p)$);
- sum all results obtained this way to obtain the apparent rate of $(p?\tilde{w}, \gamma)$ in s.

Now consider communication B - C from the example above. The apparent rate of request action C depends from two best-matching groups: $\{C\}$ and $\{C, D\}$, activated respectively by A and B. The apparent rate of C results therefore as follows:

$$\texttt{rate}(C) = \underbrace{\frac{\delta_1}{\delta_1 + \delta_2}}_{P(A)} \cdot \gamma_1 + \underbrace{\frac{\delta_2}{\delta_1 + \delta_2}}_{P(B)} \cdot (\gamma_1 + \gamma_2)$$

where we have highlighted the probabilities to select the activating invoke actions for each best-matching group to which action C belongs.

The rate of communication B - C results therefore as follows:

$$\texttt{rate}\,(B - C) = \frac{\delta_2}{\delta_1 + \delta_2} \cdot \frac{\gamma_1}{\gamma_1 + \gamma_2} \cdot \texttt{min}\left(\delta_1 + \delta_2, \frac{\delta_1}{\delta_1 + \delta_2} \cdot \gamma_1 + \frac{\delta_2}{\delta_1 + \delta_2} \cdot (\gamma_1 + \gamma_2)\right)$$

We can finally state the formula for the calculation of the rate of a communication action on endpoint p between request process expecting to receive tuple $\tilde{a}$ and invoke process sending tuple $\tilde{h}$ in service s:

$$\texttt{rate}\left(s \xrightarrow[{[\gamma,\delta]}]{\vartheta(p\ \lfloor\emptyset\rfloor\ \tilde{a}\ \tilde{h})} s'\right) = \frac{\delta}{\texttt{inv}(s,p)} \cdot \frac{\gamma}{\texttt{req}(s,p,\tilde{h},\#(\tilde{a},\tilde{h}))} \cdot \texttt{min}\,(\texttt{inv}(s,p), \texttt{aR}(\texttt{bms}(s), p, \tilde{a}, s)) \quad (1)$$

where functions `inv`, `req`, `bms`, `sumRates` and `aR` are defined in Tables 8, 9, 10, 11 and 12.

$$\begin{array}{rcl}
\texttt{inv}(s,p) & = & \texttt{inv}'(s,s,p) \\[1em]
\multicolumn{3}{c}{\texttt{inv}'(s,(\mathbf{kill}(k),\lambda),p) = \texttt{inv}'(s,(p'?\tilde{w},\gamma).\,s',p) = \texttt{inv}'(s,\mathbf{0},p) = 0} \\[1em]
\texttt{inv}'(s,(p'!\tilde{v},\delta),p) & = & \begin{cases} \delta & \text{if } p = p' \text{ and } \texttt{sumRates}(s,\tilde{v},p) \neq (0,\infty) \\ 0 & \text{otherwise} \end{cases} \\[1em]
\texttt{inv}'(s,s_1 \mid s_2,p) & = & \texttt{inv}'(s,s_1,p) + \texttt{inv}'(s,s_2,p) \\[1em]
\texttt{inv}'(s,g_1 + g_2,p) & = & \texttt{inv}'(s,g_1,p) + \texttt{inv}'(s,g_2,p) \\[1em]
\texttt{inv}'(s,[\,d\,]s',p) & = & \texttt{inv}'(s,s',p) \\[1em]
\texttt{inv}'(s,\{\!|s'|\!\},p) & = & \texttt{inv}'(s,s',p) \\[1em]
\texttt{inv}'(s,\mathsf{S}(m_1,\ldots,m_j),p) & = & \texttt{inv}'(s,s'\{^{m_1,\ldots,m_j}/_{n_1,\ldots,n_j}\},p) \\
 & & \text{if } \mathsf{S}(n_1,\ldots,n_j) = s'
\end{array}$$

Table 8: Definition of function `inv` for the computation of apparent rate of an invoke action. The apparent rate sums up to 0 if there are no request actions available for communication with the given invoke action.

$$\mathtt{req}((\mathbf{kill}(k),\lambda),p,\tilde{h},c) = \mathtt{req}((p'!\tilde{v},\delta),p,\tilde{h},c) = \mathtt{req}(\mathbf{0},p,\tilde{h},c) = 0$$

$$\begin{aligned}
\mathtt{req}((p'?\tilde{w},\gamma).\,s,p,\tilde{h},c) &= \begin{cases}\gamma & \text{if } p = p' \text{ and } c = \#(\tilde{w},\tilde{h})\\ 0 & \text{otherwise}\end{cases}\\
\mathtt{req}(s_1 \mid s_2,p,\tilde{h},c) &= \mathtt{req}(s_1,p,\tilde{h},c) + \mathtt{req}(s_2,p,\tilde{h},c)\\
\mathtt{req}(g_1 + g_2,p,\tilde{h},c) &= \mathtt{req}(g_1,p,\tilde{h},c) + \mathtt{req}(g_2,p,\tilde{h},c)\\
\mathtt{req}([\,d\,]s,p,\tilde{h},c) &= \mathtt{req}(s,p,\tilde{h},c)\\
\mathtt{req}(\{\!|s|\!\},p,\tilde{h},c) &= \mathtt{req}(s,p,\tilde{h},c)\\
\mathtt{req}(\mathsf{S}(m_1,\ldots,m_j),p,\tilde{h},c) &= \mathtt{req}(s\{{}^{m_1,\ldots,m_j}/_{n_1,\ldots,n_j}\},p,\tilde{h},c) \quad \text{if } \mathsf{S}(n_1,\ldots,n_j) = s
\end{aligned}$$

Table 9: Definition of function req used in equation (1).

$$\begin{aligned}
\mathtt{bms}(s) &= \mathtt{bms}'(s,s)\\
\mathtt{bms}'((p!\tilde{u},\delta),s) &= \{(\mathtt{sumRates}(s,\tilde{u},p),(p!\tilde{u},\delta))\}
\end{aligned}$$

$$\mathtt{bms}'((p?\tilde{w},\gamma).\,s',s) = \mathtt{bms}'((\mathbf{kill}(k),\lambda),s) = \emptyset$$

$$\mathtt{bms}'(g_1 + g_2,s) = \mathtt{bms}'(\mathbf{0},s) = \emptyset$$

$$\begin{aligned}
\mathtt{bms}'(s_1 \mid s_2,s) &= \mathtt{bms}'(s_1,s) \cup \mathtt{bms}'(s_2,s)\\
\mathtt{bms}'([\,d\,]s',s) &= \mathtt{bms}'(s',s)\\
\mathtt{bms}'(\{\!|s'|\!\},s) &= \mathtt{bms}'(s',s)
\end{aligned}$$

Table 10: Definition of function bms for the generation of all best-matching sets for each invoke action in given service s. The sum of all rates of each best-matching set is calculated "on the fly" and is coupled with its corresponding activating invoke action.

$$\mathtt{sumRates}((p'?\tilde{w},\gamma).\,s,\tilde{u},p) = \begin{cases} (\gamma,\#(\tilde{w},\tilde{u})) & \text{if } p = p' \\ (0,\infty) & \text{otherwise} \end{cases}$$

$$\mathtt{sumRates}((p'!\tilde{u}',\delta),\tilde{u},p) = \mathtt{sumRates}((\mathbf{kill}(k),\lambda),\tilde{u},p) = \mathtt{sumRates}(\mathbf{0},\tilde{u},p) = (0,\infty)$$

$$\mathtt{sumRates}([\,d\,]s,\tilde{u},p) = \mathtt{sumRates}(s,\tilde{u},p)$$

$$\mathtt{sumRates}(\{\!|s|\!\},\tilde{u},p) = \mathtt{sumRates}(s,\tilde{u},p)$$

$$\frac{\mathtt{sumRates}(s_1,\tilde{u},p) = (\gamma_1,c_1) \quad \mathtt{sumRates}(s_2,\tilde{u},p) = (\gamma_2,c_2)}{\mathtt{sumRates}(s_1 \mid s_2,\tilde{u},p) = (\gamma_3,c_3)} \quad (*)$$

$$\frac{\mathtt{sumRates}(g_1,\tilde{u},p) = (\gamma_1,c_1) \quad \mathtt{sumRates}(g_2,\tilde{u},p) = (\gamma_2,c_2)}{\mathtt{sumRates}(g_1 + g_2,\tilde{u},p) = (\gamma_3,c_3)} \quad (*)$$

$$(*) \text{ where } c_3 = \mathtt{min}\,(c_1,c_2) \text{ and } \gamma_3 = \begin{cases} \gamma_1 & \text{if } c_1 < c_2 \\ \gamma_2 & \text{if } c_2 < c_1 \\ \gamma_1 + \gamma_2 & \text{if } c_1 = c_2 \end{cases}$$

Table 11: Definition of function sumRates for the calculation of the sum of rates in best-matching sets.

$$\mathtt{aR}(\emptyset,p',\tilde{a},s) = 0$$

$$\mathtt{aR}(\{((\gamma,c),(p!\tilde{u},\delta))\} \cup M,p',\tilde{a},s) = \begin{cases} \mathtt{aR}(M,p',\tilde{a},s) + \gamma \cdot \frac{\delta}{\mathtt{inv}(s,p)} & \text{if } C \\ \mathtt{aR}(M,p',\tilde{a},s) & \text{otherwise} \end{cases}$$

$$\text{where } C = (p = p' \text{ and } \#(\tilde{a},\tilde{u}) = c)$$

Table 12: Definition of function aR for the calculation of the apparent rate of a request action involved in a communication in given service s. Condition C is used in order to check wether request action under consideration belongs to a best-matching set.

$$\text{(kill)}\ \frac{-}{(\mathbf{kill}(k),\lambda)\xrightarrow[\lambda]{(\dagger k)}\mathbf{0}} \qquad \text{(req)}\ \frac{-}{(p?\tilde{w},\gamma).\,s\xrightarrow[\gamma]{(p?\tilde{w})} s} \qquad \text{(inv)}\ \frac{-}{(p!\tilde{n},\delta)\xrightarrow[\delta]{(p!\tilde{n})}\mathbf{0}}$$

$$\text{(prot)}\ \frac{s\xrightarrow[\rho]{\vartheta(\alpha)} s'}{\{\!|s|\!\}\xrightarrow[\rho]{\vartheta(\alpha)}\{\!|s'|\!\}} \qquad \text{(choice}_0\text{)}\ \frac{g_1\xrightarrow[\rho]{\vartheta(\alpha)} s}{g_1+g_2\xrightarrow[\rho]{+_0\vartheta(\alpha)} s} \qquad \text{(choice}_1\text{)}\ \frac{g_2\xrightarrow[\rho]{\vartheta(\alpha)} s}{g_1+g_2\xrightarrow[\rho]{+_1\vartheta(\alpha)} s}$$

$$\text{(open)}\ \frac{s\xrightarrow[\rho]{\vartheta(\alpha)} s' \quad u\in d(\alpha)}{[\,u\,]s\xrightarrow[\rho]{\vartheta(\mathsf{op}(\alpha,u))} s'}$$

$$\text{(com)}\ \frac{s_1\xrightarrow[\rho_1]{\vartheta_1(p?\tilde{a})} s_1{}' \quad s_2\xrightarrow[\rho_2]{\vartheta_2(p!\tilde{h})} s_2{}' \quad \mathcal{M}\big(\tilde{a},\tilde{h}\big)=(\sigma',\sigma,\tilde{t}) \quad \neg(s_1\mid s_2)\downarrow^{|\sigma'|+|\sigma|}_{p,\tilde{h}}}{s_1\mid s_2\xrightarrow[[\rho_1,\rho_2]]{(\vartheta_1,\vartheta_2)(p\,\lfloor\sigma'\rfloor\,\tilde{a}\,\tilde{h})}[\,\tilde{t}\,](s_1{}'\sigma\mid s_2{}')}$$

Table 13: Semantics for Polyadic Stochastic cows (Part 1).

$$(\mathsf{par_{conf}})\ \frac{s_1 \xrightarrow[\rho]{\vartheta(p\,\lfloor\sigma\rfloor\,\tilde{a}\,\tilde{h})} {s_1}' \quad \neg s_2 \downarrow^{\#(\tilde{a},\tilde{h})}_{p,\tilde{h}}}{s_1 \mid s_2 \xrightarrow[\rho]{\vartheta(p\,\lfloor\sigma\rfloor\,\tilde{a}\,\tilde{h})} {s_1}' \mid s_2} \qquad (\mathsf{par_{pass}})\ \frac{s_1 \xrightarrow[\rho]{\vartheta(\alpha)} {s_1}' \quad \alpha \neq p\,\lfloor\sigma\rfloor\,\tilde{a}\,\tilde{h} \quad \alpha \neq \dagger k}{s_1 \mid s_2 \xrightarrow[\rho]{\vartheta(\alpha)} {s_1}' \mid s_2}$$

$$(\mathsf{par_{kill}})\ \frac{s_1 \xrightarrow[\rho]{\vartheta(\dagger k)} {s_1}'}{s_1 \mid s_2 \xrightarrow[\rho]{\vartheta(\dagger k)} {s_1}' \mid halt(s_2)} \qquad (\mathsf{del_{pass}})\ \frac{s \xrightarrow[\rho]{\vartheta(\alpha)} s' \quad d \notin d(\alpha) \quad s{\downarrow_{kill}} \Rightarrow (\alpha = \dagger \text{ or } \alpha = \dagger k)}{[\,d\,]s \xrightarrow[\rho]{\vartheta(\alpha)} [\,d\,]s'}$$

$$(\mathsf{del_{cl}})\ \frac{s \xrightarrow[\rho]{\vartheta(p\,\lfloor\sigma \uplus \{(n)/x\}\rfloor\,\tilde{a}\,\tilde{h})} s'}{[\,x\,]s \xrightarrow[\rho]{\vartheta(p\,\lfloor\sigma\rfloor\,\tilde{a}\,\tilde{h})} [\,n\,]s'\{^{n}/_{x}\}} \qquad (\mathsf{del_{sub}})\ \frac{s \xrightarrow[\rho]{\vartheta(p\,\lfloor\sigma \uplus \{n/x\}\rfloor\,\tilde{a}\,\tilde{h})} s'}{[\,x\,]s \xrightarrow[\rho]{\vartheta(p\,\lfloor\sigma\rfloor\,\tilde{a}\,\tilde{h})} s'\{^{n}/_{x}\}} \qquad (\mathsf{del_{kill}})\ \frac{s \xrightarrow[\rho]{\vartheta(\dagger k)} s'}{[\,k\,]s \xrightarrow[\rho]{\vartheta(\dagger)} [\,k\,]s'}$$

$$(\mathsf{ser_{id}})\ \frac{s\{^{m_1,\ldots,m_j}/_{n_1,\ldots,n_j}\} \xrightarrow[\rho]{\vartheta(\alpha)} s' \quad \mathsf{S}(n_1,\ldots,n_j) = s}{\mathsf{S}(m_1,\ldots,m_j) \xrightarrow[\rho]{\vartheta(\mathrm{1\_dec}(\alpha))} \mathrm{s\_dec}(\alpha, s')}$$

Table 14: Semantics for Polyadic Stochastic cows (Part 2).

References

[1] Jane Hillston. *A compositional approach to performance modelling*. Cambridge University Press, New York, NY, USA, 1996. ISBN 0-521-57189-8.

[2] A. Lapadula, R. Pugliese, and F. Tiezzi. A Calculus for Orchestration of Web Services. In R. De Nicola, editor, *Proc. of 16th European Symposium on Programming (ESOP'07)*, Lecture Notes in Computer Science. Springer.

[3] A. Lapadula, R. Pugliese, and F. Tiezzi. A formal account of WS-BPEL. In *Proc. 10th international conference on Coordination Models and Languages (COORDINATION'08)*, Lecture Notes in Computer Science. Springer.

[4] Davide Prandi and Paola Quaglia. Stochastic COWS. In *Proc. 5th International Conference on Service Oriented Computing, ICSOC '07*, volume 4749 of *LNCS*, 2007.

[5] Corrado Priami. Stochastic pi-calculus. *Comput. J.*, 38(7):578–589, 1995.