Bounding Reactions in the π-calculus using Interpretations

Romain Péchoux*

INRIA project Carte, LORIA, and Université de Lorraine
romain.pechoux@loria.fr

Abstract. We present a new resource static analysis for the π-calculus that provides upper bounds on the number of reactions that might occur at runtime for a given process. This work is complementary to previous results on termination of processes by capturing strictly more processes, since it captures all the strongly normalizing processes, and by providing precise upper bounds on the number of communications on each channel. For that purpose, it combines interpretation methods, inspired by polynomial interpretations introduced in order to study the complexity of term rewrite systems, with a notion of resource process that mimics reaction keeping information about resource consumption in terms of communication. We also show that presented analysis is general and can be easily adapted to study space properties of processes (for example, upper bounds on the size of the maximal value sent on a given channel during reaction).

1 Introduction

Motivations. This work belongs to the Implicit Computational Complexity (ICC) research line whose aim is to characterize complexity classes without referring to machines. In the last two decades, contributions have tried to extend the characterizations to several computational paradigms such as function algebra [3, 21], lambda-calculi [22, 10, 2], light and soft linear logics [11, 9], term rewrite systems [4] and imperative programs [15, 25], to mention a few. Surprisingly, only a few works have been performed on resource control and analysis of concurrent models. Such a lack mainly relies on the fact that most of these models do not deal with functions as computational objects. However most of these concurrent models have their own notions of time and space in analogy with sequential models. More specifically for the π-calculus, it has been commonly accepted that the time can be understood as the maximal number of possible communications between processes.

Contribution. In this paper, we try to tackle the problem of (first-order) π-calculus time analysis by finding, if possible, an upper bound on the number of possible reactions on each channel. For that purpose, we extend (polynomial)

* The financial support of ANR Complice and Inria Associate team Cristal is gratefully acknowledged.

interpretations, a well-known termination tool for Term Rewrite Systems to a process calculus. The considered process calculus mainly consists in the classical π-calculus where replication is replaced by the ability to call processes recursively through the use of a case construct. Processes in $\mathcal{P}(\mathrm{N})$ are considered with respect to a fixed set of names N. In order to achieve our goal, we use a notion of interpretation whose inspiration is drawn from polynomial interpretations [23, 20] used to show termination of first order rewrite systems. The analysis is performed in two steps as illustrated by the following diagram:

$$\mathcal{P}(\mathrm{N}) \xrightarrow{[-]} \mathcal{R}(\mathrm{N}) \xrightarrow{[\![-]\!]} \mathbb{N}^{\mathrm{N}\cup\{*\}}$$

The first step corresponds to the interpretation $[-]$, an assignment mapping each process in $\mathcal{P}(\mathrm{N})$ to a *resource process* in $\mathcal{R}(\mathrm{N})$. Resource processes are processes with extra prefixes, recording reactions on each channel, and without process call. Consequently, they are strongly normalizing since they have neither the ability to replicate, nor the ability to recurse. The interpretation enjoys some simulation properties over resource processes guaranteeing that a resource process will have at least as many resources (in terms of reaction) as the initial process it represents.

In the second step, we interpret each resource process by a function of the resource algebra $\mathbb{N}^{\mathrm{N}\cup\{*\}}$ using the semantics $[\![-]\!]$. Such a function maps each channel in N to a natural number in $\mathbb{N}$, the upper bound on its possible reactions. $*$ is a special symbol for process calls that takes account of the recursive calls performed by the studied process.

Results. Our main result (Theorem 1) is that, for a given process P, the function $[\![[P]]\!]$ provides upper bounds on both the number of reactions, for each channel, and the number of recursive calls occurring in a reduction of P. In other words, for a given channel x, $[\![[P]]\!](x)$ will be an upper bound on the number of reactions based on a communication on channel x.

Moreover, we show that the presented analysis is complete for strongly normalizing processes, if no restriction is made on resource processes (Theorem 2) and can be extended to study space properties of processes.

Another interest of our results lies in the transfer and use of well-known termination techniques for term rewrite systems to π-calculus.

Related works. Time properties of processes have been deeply studied in the last decade adapting well-known ICC complexity classes characterizations based on type systems [7, 27, 6], linear logics [14, 29, 19] and access control [17]. Contrarily to the present work, most of these studies are global and do not allow the analyzer to infer precise upper bounds on the use of each channel. An important point to stress is that the results presented in this paper have a greater expressivity than all the aforementioned studies, that are mostly tractable in polynomial time. In opposition, our work captures all the strongly normalizing processes. The pros are a greater expressivity, the cons are that our framework needs to be restricted in order to be effective. However we think that it opens a new line of research since decidable fragments are already delineated by considering restricted function spaces as discussed in the last section of the paper.

Another branch of research [26, 12], has tried to count the number of occurrences of processes during computation sequences using abstract interpretations and relational domains [8]. In the same direction, some studies [16, 28] have tried to infer buffer upper bounds of concurrent imperative programs communicating via buffered channels. In the same spirit, [18] proposes an analysis that infers lower and upper bounds on the number of active channels using a type system combined to a lattice ordered monoïd. The problems tackled in these papers are more related to space consumption analysis during process reaction and, consequently, orthogonal to the present study, though clearly related.
In another direction, Hennessy [13] and Pym [5] that have developed variants of the π-calculus where channels come equipped with a notion of resource. Reactions may only occur when there is enough resource in the owning channel. These studies provide a dynamic analysis of system behaviors depending on whether it has been fed with enough resources, whereas the work presented in this paper is inherently static by trying to give an approximation of resources needed by a process before reduction.
Outline. The syntax and semantics of processes is introduced in Section 2. Section 3 describes the first step of our static analysis by introducing resource processes, particular processes without recursive calls, and defining interpretations and their properties. Section 4 describes the second step, a semantics that gives a meaning to resource processes on a resource algebra. Some intermediate lemmata show that resources decrease during reactions. We show (Theorem 1) that the resource algebra function corresponding to a process provides upper bounds on its reaction number, for each channel, and that the analysis is complete for strongly normalizing processes (Theorem 2). Moreover, some more toy examples (and a counter-example) are developed in order to help the reader to grasp the presented methodology. Section 5 discusses some extension of the presented methodology to space upper bounds. Sections 6 deals with the interpretation inference problem and shows that the presented methodology opens a lot of perspectives in the study of process time and space complexity properties.

2 Preliminaries

Syntax of processes. The extended concurrent model considered in this paper merely consists in the classical simply typed π-calculus constructs of [24], inaction, input, output, parallel composition, sum and restriction where replication is replaced by the ability to call process definitions and where values are extended to inductive data types. For simplicity, we will restrict our study to integer data type. For that purpose, let $V = \{l, m, n, \ldots\}$ be a set of integer variables. Moreover, let Op be a set of basic operators op of fixed arity over integers. We consider a (possibly infinite) set $\mathrm{N} = \{a, b, c, \ldots, x, y, w, \ldots\}$ of names and a distinguished element τ for reaction (i.e. $\mathrm{N} \cap \{\tau\} = \emptyset$). We also assume a fixed set $\mathcal{F} = \{F, G, \ldots\}$ of process symbols of fixed arity. In what follows, let $\overrightarrow{t}$ be a notation for the sequence $t_1, \cdots, t_n$ when n is clear from the context. Expressions in $\mathcal{E}$, values in $\mathcal{V}$ and processes in $\mathcal{P}(\mathrm{N})$ are defined by the

grammar of Figure 2. For a given process P, define $fn(P)$, $bn(P)$ and $n(P)$ to

$$\begin{array}{rl} \mathcal{E} \ni e ::= & n \mid 0, 1, 2, \ldots \mid op(\overrightarrow{e}) \\ \mathcal{V} \ni v, w ::= & x \mid e \\ \mathcal{P}(\mathrm{N}) \ni P, Q ::= & \mathtt{0} \mid x(y).P \mid \overline{x}v.P \mid F(\overrightarrow{v}) \mid P|P \mid P + P \mid (\nu x)P \end{array}$$

Fig. 1. Syntax of processes

be respectively the free names, bound names and names of the process P. Define also $fv(P)$ to be the free integer variables in P. A closed process is a process such that $fv(P) = \emptyset$. Let $\overline{\mathcal{V}}$ be the set of ground values, i.e. $v \in \overline{\mathcal{V}}$ if $fv(v) = \emptyset$. A process call $F(\overrightarrow{v})$ corresponds to the application of the process symbol F. Each process symbol F comes with exactly one process definition of the shape:

$$F(\overrightarrow{X}) = \mathtt{Case}\ \overrightarrow{X}\ \mathtt{of}\ \overrightarrow{v_1} \to P_1,\ \ldots,\ \overrightarrow{v_k} \to P_k$$

We will assume that the distinct rules in a process definition are non-overlapping. Moreover, we will assume that $\forall i,\ fv(P_i) \subseteq fv(\overrightarrow{v_i})$. For convenience, the notation $F(\overrightarrow{v_i}) = P_i$ will be used to denote the i-th pattern matching rule. Definitions may be recursive. Consequently, the presented framework is at least as expressive as the π-calculus since the definition $F() = F()|P$ encodes the replication $!P$.

We will sometimes use the notation μ to denote either the input prefix $x(y)$ or the output prefix $\overline{x}v$ when there is no need to distinguish prefixes. Moreover, let $\tilde{\nu}$ be a notation for either the restriction νx or the empty restriction, depending on the context. Finally, for readability, we will omit inaction, that is we write μ instead of $\mu.\mathtt{0}$. Given a process P, $D(P)$ is the set of process symbols called either in P or (inductively) in the definition of process symbols called in P.

The structural congruence $\equiv$ is defined as the least congruence relation on processes that satisfies:

- $\mathtt{0} + P \equiv P$, $P + Q \equiv Q + P$, $(P + Q) + R \equiv P + (Q + R)$,
- $\mathtt{0}|P \equiv P$, $P|Q \equiv Q|P$, $(P|Q)|R \equiv P|(Q|R)$,
- $(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$, $(\nu x)\mathtt{0} \equiv \mathtt{0}$, $(\nu x)(P|Q) \equiv ((\nu x)P)|Q$, if $x \notin fn(Q)$.

For simplicity, we will suppose that all considered processes have already been carefully alpha-converted.

Operational semantics. We do not provide an explicit operational semantics for expressions and we suppose that each operator corresponds to some fixed total function $[\![op]\!]$ over ground values. We provide an annotated operational semantics for processes, a slight change of usual operational semantics. It consists in the transition system described in Figure 2. A transition has the shape $P \xrightarrow{\alpha} Q$, where α is a transition label that is equal to:

- xw if the transition corresponds to an input of the name w with respect to input prefix $x(y)$ (rule (In)),

– $\tilde{\nu}\overline{x}w$ if the transition corresponds to an output of the name w with respect to output prefix $\overline{x}w$ (rules (Out) and (Open)),
– $\tau(\chi),\ \chi \in \mathrm{N} \cup \{*\}$ if the transition corresponds to a reaction (rules (Com) and (Close)), in this case $\chi \in \mathrm{N}$, or to a process call (rule (Rec)), in this case $\chi = *$. Notice that a $n(\tau(x)) = fn(\tau(x)) = \emptyset$. χ plays the role of label recording either the reacting channel or the application of a process definition, whenever $\chi = *$

$$\frac{}{x(y).P \overset{xw}{\to} P\{w/y\}}\ (\text{In}) \qquad \frac{}{\overline{x}w.P \overset{\overline{x}w}{\to} P}\ (\text{Out})$$

$$\frac{P \overset{\alpha}{\to} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P \mid Q \overset{\alpha}{\to} P' \mid Q}\ (\text{Par}) \qquad \frac{P \overset{\alpha}{\to} P' \quad Q \equiv P \quad P' \equiv Q'}{Q \overset{\alpha}{\to} Q'}\ (\text{Var})$$

$$\frac{P \overset{\alpha}{\to} P' \quad x \notin n(\alpha)}{(\nu x)P \overset{\alpha}{\to} (\nu x)P'}\ (\text{Res}) \qquad \frac{P \overset{\overline{x}w}{\to} P' \quad x \neq w}{(\nu w)P \overset{(\nu w)\overline{x}w}{\to} P'}\ (\text{Open})$$

$$\frac{P \overset{\overline{x}w}{\to} P' \quad Q \overset{xw}{\to} Q'}{P \mid Q \overset{\tau(x)}{\to} P' \mid Q'}\ (\text{Com}) \qquad \frac{P \overset{(\nu w)\overline{x}w}{\to} P' \quad Q \overset{xw}{\to} Q' \quad w \notin fn(Q)}{P \mid Q \overset{\tau(x)}{\to} (\nu w)(P' \mid Q')}\ (\text{Close})$$

$$\frac{\overrightarrow{v_i}\sigma = \overrightarrow{v} \quad F(\overrightarrow{v_i}) = P_i}{F(\overrightarrow{v}) \overset{\tau(*)}{\to} P_i\sigma}\ (\text{App}) \qquad \frac{P \overset{\alpha}{\to} P'}{P + Q \overset{\alpha}{\to} P'}\ (\text{Sum})$$

Fig. 2. Annotated operational semantics

Let $\mathbb{D}$ be a notation for $\mathrm{N} \cup \{*\}$. Note that the (App) rule of Figure 2 uses a standard notion of substitution σ, a mapping from free names and free variables to values of the same type. Among substitutions, we distinguish the ground substitutions, substitution mapping variables to ground values in $\overline{\mathcal{V}}$. Let $SN_{\tau(\chi)}$ be the set of processes P such that each sequence of reduction starting from the closed process $P\sigma$ has a finite number of transitions labeled by $\tau(\chi)$, for any substitution $\sigma : fv(P) \to \mathbb{N}$. For a given total function $f : \mathbb{N} \to \mathbb{N}$, let $SN_{\tau(\chi)}(f)$ be the set of processes P such that for each substitution $\sigma : fv(P) \to \mathbb{N}$, if $\forall n \in fv(P),\ n\sigma \leq k$ then $P\sigma$ has at most $f(k)$ transitions labeled by $\tau(\chi)$. For each $f : \mathbb{N} \to \mathbb{N}$ and each $\chi \in \mathbb{D}$, the following inclusion holds $SN_{\tau(\chi)}(f) \subset SN_{\tau(\chi)}$. Moreover, we define $SN = \cap_{\chi \in \mathbb{D}} SN_{\tau(\chi)}$ and we write $P \in SN_{\tau(\chi)}(f(k))$ as an abuse of notation for $P \in SN_{\tau(\chi)}(f)$. Notice that the definition of SN is standard since process syntax prevents us from building a diverging process with a finite number of reactions on an infinite number of channel names.

Example 1. Consider the process $fac(m, a)$ and the process definition:

$$fac(n, r) = \texttt{Case}\ n\ \texttt{of}\ 0 \to \overline{r}\langle 1\rangle$$
$$m + 1 \to (\nu r')(fac(m, r') | r'(x).\overline{r}\langle x \times (m + 1)\rangle)$$

For each substitution σ such that $n\sigma = k$, $k \in \mathbb{N}$, and $r\sigma = a$, $a \in \mathrm{N}$, $fac(n, r)\sigma$ computes $k!$ through k communications on internal channels r', $k + 1$ recursive

calls of the process symbol *fac* and outputs the result on channel a. Indeed, we have the following sequence of reductions:

$$fac(n,r)\sigma = fac(k,a)\underbrace{\xrightarrow{\tau(*)} \dots \xrightarrow{\tau(*)}}_{k+1 \text{ times}} \dots \underbrace{\xrightarrow{\tau(r')} \dots \xrightarrow{\tau(r')}}_{k \text{ times}} \overline{a}\langle (k-1)! \times k \rangle$$

Consequently, we have $fac(n,r) \in SN_{\tau(*)}(k{+}1)$, $fac(n,r) \in SN_{\tau(r')}(k)$,$fac(n,r) \in SN_{\tau(r)}(0)$ and $fac(n,r) \in SN$.

3 Resource analysis by interpretation

A resource analysis based on interpretation methods in order to control the time complexity of process computations is presented in this section.

3.1 Resource processes

Resource processes will be the codomain of the first interpretation. They mainly consist in processes with no process call and with extra input prefixes of the shape τ_χ^n, with $\chi \in \mathbb{D}$ and $n \in \mathbb{N}$, and are defined by the following grammar:

$$\mathcal{R}(\mathrm{N}) \ni R, S ::= 0 \mid \tau_\chi^n.R \mid \mu.R \mid R|R \mid R + R \mid (\nu x)R$$

Intuitively, a resource process $\tau_\chi^n.R$ will be able to perform n transitions labeled by $\tau(\chi)$. The notions of names, bounded names, free names, and free variables are defined in a standard way. Resource processes structural congruence is similar to process structural congruence but extended by $\tau_\chi^0.R \equiv R$. In what follows, a context $C[\overrightarrow{\diamond}]$ is a resource process with exactly one occurrence of the hole $\diamond_i$. Let $C[\overrightarrow{R}]$ be the resource process obtained by filling the holes $\overrightarrow{\diamond}$ with $\overrightarrow{R}$.

Resource process operational semantics is defined by the rules of Figure 2 and is extended by the following rule:[1]

$$\frac{}{\tau_\chi^{n+1}.R \xrightarrow{\tau(\chi)} \tau_\chi^n.R}\ (T_{n+1})$$

Definition 1. *The partial preorder $\gtrsim$ on resource processes is a simulation defined by $R \gtrsim R'$ if $\forall S' \in \mathcal{R}(N)$ s.t. $R' \xrightarrow{\alpha} S'$, $\exists S \in \mathcal{R}(N)$ s.t. $R \xrightarrow{\alpha} S$ and $S \gtrsim S'$.*

Lemma 1. *The relation $\gtrsim$ enjoys the following properties:*

(1) $\forall \sigma$, *if* $R \gtrsim S$ *then* $R\sigma \gtrsim S\sigma$ *(Stability by substitution)*

(2) $\forall C[\overrightarrow{\diamond}]$, *if* $\forall i$, $\overrightarrow{R} \gtrsim \overrightarrow{S}$ *then* , $C[\overrightarrow{R}] \gtrsim C[\overrightarrow{S}]$ *(Stability by context)*

[1] Notice that rule (App) of Figure 2 is meaningless for resource processes since they do not involve process calls.

3.2 Process assignment

An assignment consists in annotations provided by the analyzer on the resources consumed by process symbols and is defined as follows:

Definition 2 (Assignment). *Given a process P s.t. $D(P) = \{F_1, \cdots, F_m\}$, an assignment $[-]$ is a total map from $D(P)$ to $\mathcal{V}^* \to \mathcal{R}(N)$ assigning to each process definition F_i of arity n a total function $[F_i]$ such that for each $v_1, \cdots, v_n \in \overline{\mathcal{V}}$, $[F_i](v_1, \cdots, v_n) \in \mathcal{R}(N)$.*

Example 2. Consider the process $fac(k, a)$ of Example 1: We have $D(fac(k, a)) = \{fac\}$. Consequently, the function $[-]$ such that $[fac](n, r) = \tau_*^{n+1}|(\nu r')\tau_{r'}^n|\overline{r}\langle n!\rangle$ is an assignment. The informal meaning of this assignment is that the process $fac(n, r)$ will react at most n times on internal names r', performs at most $n+1$ transitions labeled by $\tau(*)$ and outputs at most once the value $n!$ on channel r. Notice that all the internal channels r' under a restriction in the initial process are abstracted as a whole. However we still infer upper bounds since n is, by definition, an upper bound on the number of reactions of each internal channel. Notice also that internal names are treated as constants and, consequently, do not need to appear in the parameters of the assignment.

Definition 3 (Process assignment). *Given an assignment $[-]$, a process assignment is a map from $\mathcal{P}(N)$ to $\mathcal{R}(N)$ defined inductively by:*

$$\begin{array}{cc} [0] = 0 & [(\nu x)P] = (\nu x)[P] \\ [P + Q] = [P] + [Q] & [F(\overrightarrow{v})] = [F](\overrightarrow{v}) \\ [P|Q] = [P]|[Q] & [\mu.P] = \mu.[P] \end{array}$$

Example 3. Consider the subprocess $(\nu r')(fac(m, r')|r'(x).\overline{r}\langle x \times (m+1)\rangle)$ of Example 1 together with the assignment $[fac](n, r) = (\nu r')\tau_{r'}^n|\tau_*^{n+1}|\overline{r}\langle n!\rangle$:

$$\begin{aligned} &[(\nu r')(fac(m, r')|r'(x).\overline{r}\langle x \times (m+1)\rangle)] \\ &= (\nu r')[fac(m, r')|r'(x).\overline{r}\langle x \times (m+1)\rangle] \\ &= (\nu r')([fac](m, r')|[r'(x).\overline{r}\langle x \times (m+1)\rangle]) \\ &= (\nu r')(((\nu r')\tau_{r'}^m|\tau_*^{m+1}|\overline{r'}\langle m!\rangle)|(r'(x).\overline{r}\langle x \times (m+1)\rangle) \end{aligned}$$

Structural congruence and reaction are preserved on resource processes:

Lemma 2. *If $P \equiv Q$ then $[P] \equiv [Q]$.*

Lemma 3. *Given a process P and an assignment $[-]$, $\forall \alpha \in \{\tilde{\nu}\overline{x}w, xw, \tau(x)\}$, if $P \xrightarrow{\alpha} P'$ then $[P] \xrightarrow{\alpha} [P']$.*

Proof. By induction on the annotated semantics. □

3.3 Interpretation

Now we introduce the notion of interpretation, the cornerstone of the presented analysis. Interpretations enforce the semantics of assignments to satisfy inequalities that allow the analyzer to infer upper bounds on reactions.

Definition 4 (Interpretation). *Given a process P, an assignment $[-]$ is an interpretation of P if for each process definition $F \in D(P)$ of the shape:*

$$F(\overrightarrow{X}) = \texttt{Case } \overrightarrow{X} \texttt{ of } \overrightarrow{v_1} \to P_1, \ldots, \overrightarrow{v_k} \to P_k$$

for each ground substitution σ the following holds:

$$\forall i \in \{1, \ldots, k\},\ [F(\overrightarrow{v_i})\sigma] \gtrsim \tau_*^1.[P_i\sigma]$$

In such a case, we say that P has interpretation [-].

The informal meaning of an interpretation is that the process call has more resources than its evaluation since $\gtrsim$ is the simulation preorder. The prefix τ_*^1 is needed since one $\tau(*)$ transition has already been performed when process call the computation is under consideration.

Example 4. Consider the process $fac(k, a)$ and the assignment $[-]$ of Example 3:

$$\begin{aligned}
&[fac](0, r) = \tau_{r'}^0|\overline{r}\langle 0!\rangle|\tau_*^1 \equiv \overline{r}\langle 1\rangle|\tau_*^1 \gtrsim \tau_*^1.\overline{r}\langle 1\rangle = \tau_*^1.[\overline{r}\langle 1\rangle] && \text{Since } \tau_{r'}^0.0 \equiv 0\\
&[fac](m+1, r) = (\nu r')\tau_{r'}^{m+1}|\tau_*^{m+2}|\overline{r}\langle (m+1)!\rangle\\
&\gtrsim \tau_*^1.[(\nu r')(fac(m, r')|r'(x).\overline{r}\langle x \times (m+1)\rangle)]\\
&= \tau_*^1.(\nu r')(((\nu r')\tau_{r'}^m|\tau_*^{m+1}|\overline{r'}\langle m!\rangle)|(r'(x).\overline{r}\langle x \times (m+1)\rangle) && \text{By Example 3}
\end{aligned}$$

Note that the above inequality $\gtrsim$ for simulation holds since the traces of

$$\tau_*^1.(\nu r')(((\nu r')\tau_{r'}^m|\tau_*^{m+1}|\overline{r'}\langle m!\rangle)|(r'(x).\overline{r}\langle x \times (m+1)\rangle)$$

are strictly included in the traces of $[fac](m+1, r)$. Consequently, $[-]$ is an interpretation of the process $fac(k, a)$.

Lemma 4. *Given a closed process P of interpretation $[-]$, if $P \overset{\tau(*)}{\to} P'$ then $[P] \gtrsim \tau_*^1.[P']$.*

Proof. By induction on the $\overset{\tau(*)}{\to}$ transitions. □

4 Resource algebra for bounded reaction

4.1 Resource algebra.

Resource algebra will be used to compute the resource upper bounds of the presented analysis. The resource algebra is the set of total functions $\mathbb{N}^{\mathbb{D}}$ mapping each annotation $\chi \in \mathbb{D}$ to a natural number $n \in \mathbb{N}$.

Let $n_A, n \in \mathbb{N}, A \subseteq \mathbb{D}$ be a notation for the characteristic function of A defined by:

$$n_A(\chi) = \begin{cases} n, \forall \chi \in A \\ 0, \forall \chi \in \mathbb{D} - A \end{cases}$$

In what follows, let δ_0 be a notation for the function defined by $\forall \chi \in \mathbb{D},\ \delta_0(\chi) = 0$. The commutative operator $\otimes$ is defined by:

$$\forall \chi \in \mathbb{D}, (\delta \otimes \delta')(\chi) = \delta(\chi) + \delta'(\chi)$$

where $(\mathbb{N}, +, 0)$ is the usual commutative monoïd over natural numbers. Note that δ_0 is the neutral element for $\otimes$ since $\delta \otimes \delta_0 = \delta$. Moreover, we have $n_A \otimes m_A = (n+m)_A$. The commutative operator $\oplus$ is defined in the same way by:

$$\forall \chi \in \mathbb{D}, (\delta \oplus \delta')(\chi) = \max(\delta(\chi), \delta'(\chi))$$

where $(\mathbb{N}, \max, 0)$ is the usual commutative monoid over natural numbers.

Lemma 5. $(\mathbb{N}^{\mathbb{D}}, \otimes, \oplus)$ *is a max-plus algebra.*

The resource algebra underlies a poset structure $(\mathbb{N}^{\mathbb{D}}, \succeq)$, where $\succeq$ is defined by $\delta \succeq \delta'$ iff $\delta \oplus \delta' = \delta$.

4.2 Resource process semantics

We introduce a semantics of resource processes $[\![-]\!]$, a partial map from $\mathcal{R}(\mathrm{N})$ to $\mathbb{N}^{\mathbb{D}}$. For that purpose, we define the reachability relation between resource processes. Given a resource process R and a name $\chi \in \mathbb{D}$, define the set of reachable processes $Reach^{\chi}(R)$ from R through χ by $Reach^{\chi}(R) = \{S \mid R \overset{\tau(\chi)}{\to} S\}$. Finally, define the set of reachable processes from R by $Reach(R) = \cup_{\chi \in \mathbb{D}} Reach^{\chi}(R)$.

Definition 5 (Resource Process semantics). *The resource process semantics* $[\![-]\!]$ *is defined by:*

$$[\![R]\!] = \oplus\{1_{\{\chi\}} \otimes [\![S]\!] \mid \forall \chi \in \mathbb{D},\ \forall S \in Reach^{\chi}(R)\} \qquad \text{if } Reach(R) \neq \emptyset$$
$$[\![R]\!] = \delta_0 \qquad \text{otherwise}$$

Lemma 6. *If* $R \gtrsim S$ *then* $[\![R]\!] \succeq [\![S]\!]$.

Proof. There are two cases to consider:

– If $Reach(S) \neq \emptyset$ then by definition of the resource process semantics $[\![-]\!]$:

$$[\![S]\!] = \oplus\{1_{\{\chi\}} \otimes [\![S']\!] \mid \forall \chi \in \mathbb{D},\ \forall S' \in Reach^{\chi}(S)\}$$

However for each $S' \in Reach^{\chi}(S)$, there exists $R' \in Reach^{\chi}(R)$ such that $R' \gtrsim S'$, by definition of $\gtrsim$. Suppose by induction hypothesis (I.H.) that $[\![R']\!] \succeq [\![S']\!]$. We obtain that:

$$\begin{aligned} [\![S]\!] &= \oplus\{1_{\{\chi\}} \otimes [\![S']\!] \mid \forall S' \text{ s.t. } S \overset{\tau(\chi)}{\to} S'\} && \text{By definition of } [\![-]\!] \\ &\preceq \oplus\{1_{\{\chi\}} \otimes [\![R']\!] \mid \forall R' \text{ s.t. } R \overset{\tau(\chi)}{\to} R'\}) && \text{(I.H.) and monotonicity of } \oplus, \otimes \\ &= [\![R]\!] && \text{Since } R' \in Reach^{\chi}(R) \end{aligned}$$

– If $Reach(S) = \emptyset$ then $[\![S]\!] = \delta_0 \preceq [\![R]\!]$. □

Lemma 7 (Resource consumption). *Given a closed process P of interpretation $[-]$, if $P \stackrel{\tau(\chi)}{\to} P'$, $\chi \in \mathbb{D}$, then $[\![[P]]\!] \succeq 1_\chi \otimes [\![[P']]\!]$.*

Proof. There are two cases to consider:

– If $\chi \in \mathrm{N}$ then, by Lemma 3, we have $[P] \stackrel{\tau(\chi)}{\to} [P']$. Consequently, $[P'] \in Reach^\chi([P]) \neq \emptyset$. By definition of $[\![-]\!]$, we obtain that $[\![[P]]\!] \succeq 1_{\{\chi\}} \otimes [\![[P']]\!]$.
– If $\chi = *$ then, by Lemma 4, $[P] \gtrsim \tau_*^1.[P']$. Consequently, by Lemma 6, we obtain that $[\![[P]]\!] \succeq [\![\tau_*^1.[P']]\!] \succeq 1_{\{*\}} \otimes [\![[P']]\!]$, by definition of $[\![-]\!]$. □

4.3 Bounded reactions and completeness of the analysis

Theorem 1. *Given a process P of interpretation $[-]$, $\forall \chi\ P \in SN_{\tau(\chi)}([\![[P]]\!](\chi))$.*

Proof. By Lemma 7, if $P \stackrel{\tau(\chi)}{\to} P'$, $\chi \in \mathbb{D}$, then $[\![[P]]\!] \succeq 1_\chi \otimes [\![[P']]\!]$. Consequently, $[\![[P]]\!](\chi) \geq (1_\chi \otimes [\![[P']]\!])(\chi) = 1 + [\![[P']]\!](\chi)$. Moreover, if $P \stackrel{\tau(\chi')}{\to} P'$, $\chi' \neq \chi$ then $[\![[P]]\!](\chi) \geq (1_{\chi'} \otimes [\![[P']]\!])(\chi) = [\![[P']]\!](\chi)$. Consequently, there are at most $[\![[P]]\!](\chi)$ reactions labeled by χ. □

Theorem 2. *A process P has interpretation $[-]$ if and only if $P \in SN$.*

Proof. Soundness: By Theorem 1, $P \in SN_{\tau(\chi)}([\![[P]]\!](\chi)), \forall \chi \in \mathbb{D}$. Consequently, $P \in \cap_{\chi \in \mathbb{D}} SN_{\tau(\chi)}([\![[P]]\!](\chi)) \subset SN$. Completeness: If $P \in SN$ then the traces of P are finite. From these traces, we can rebuilt a resource process simulating P. We do exactly the same for all the P_i appearing in the right hand side of a definition. The simulation inequalities are clearly satisfied since for each trace t_i of P_i, $\tau(*).t_i$ is a trace of P. □

4.4 Examples

In this section we illustrate some small examples and one counter-example to help the reader to understand the interpretation based method.

Example 5. For the process $fac(a, k)$ of Example 1, we have:

$$[fac(k, a)] = (\nu r')\tau_{r'}^k | \tau_*^{k+1} | \overline{a}\langle k! \rangle$$

Consequently, $[\![[fac(k, a)]]\!] = k_{\{r'\}} \otimes (k + 1)_{\{*\}}$ and:

$$\begin{array}{ll} fac(a, k) \in SN_{\tau(r')}([\![[fac(k, a)]]\!](r')) = SN_{\tau(r')}(k) & \text{By Theorem 1} \\ fac(a, k) \in SN_{\tau(*)}([\![[fac(k, a)]]\!](*)) = SN_{\tau(*)}(k + 1) & \\ fac(a, k) \in SN_{\tau(r)}([\![[fac(k, a)]]\!](r)) = SN_{\tau(r)}(0) & \end{array}$$

which means that the process $fac(k, a)$ may react at most k times on the internal channels r', may call processes at most $k + 1$ times and will never react on a (note that r' occurs freely here since internal channels are abstracted as a whole). Clearly $fac(a, k) \in SN$, by Theorem 2.

Example 6. Consider the process $F(a,k)$ with respect to the process definition:

$$\begin{aligned} F(c,n) = \texttt{Case}\ c,n\ \texttt{of}\ & c,0 \to \overline{c}\langle 0\rangle \\ & c,m+1 \to \overline{c}\langle m+1\rangle | F(c,m) \end{aligned}$$

It admits the following interpretation $[F](c,n) = \tau_*^{n+1}|\overline{c}\langle 0\rangle|\ldots|\overline{c}\langle n\rangle$ since:

$$\begin{aligned} [F(c,0)] &= \tau_*^1|\overline{c}\langle 0\rangle \geq \tau_*^1.\overline{c}\langle 0\rangle = \tau_*^1.[\overline{c}\langle 0\rangle] \\ [F(c,m+1)] &= \tau_*^{m+2}|\overline{c}\langle 0\rangle|\ldots|\overline{c}\langle m+1\rangle \gtrsim \tau_*^1.(\overline{c}\langle m+1\rangle|\tau_*^{m+1}|\overline{c}\langle 0\rangle|\ldots|\overline{c}\langle m\rangle) \\ &\gtrsim \tau_*^1.[\overline{c}\langle m+1\rangle|F(c,m)] \end{aligned}$$

Moreover, $[\![[F(a,k)]]\!] = [\![\tau_*^{k+1}|\overline{a}\langle 0\rangle|\ldots|\overline{a}\langle k\rangle]\!] = (k+1)_{\{*\}}$. Consequently, $F(a,k) \in SN_{\tau(*)}(k+1)$ and $F(a,k) \in SN_{\tau(a)}(0)$.

Example 7. Consider the process $F(a,k)$ with respect to the process definition:

$$\begin{aligned} F(c,n) = \texttt{Case}\ c,n\ \texttt{of}\ & c,0 \to \overline{c}\langle 0\rangle \\ & c,m+1 \to F(c,m) + F(c,m) \end{aligned}$$

It admits the following interpretation $[F](c,n) = \tau_*^{n+1}|\overline{c}\langle 0\rangle$ since:

$$\begin{aligned} [F(c,0)] &= \tau_*^1|\overline{c}\langle 0\rangle \gtrsim \tau_*^1.\overline{c}\langle 0\rangle = \tau_*^1.[\overline{c}\langle 0\rangle] \\ [F(c,m+1)] &= \tau_*^{m+2}|\overline{c}\langle 0\rangle \\ &\gtrsim \tau_*^1.((\tau_*^{m+1}|\overline{c}\langle 0\rangle) + (\tau_*^{m+1}|\overline{c}\langle 0\rangle)) \\ &= \tau_*^1.[F(c,m) + F(c,m)] \end{aligned}$$

Moreover, $[\![[F(a,k)]]\!] = [\![\tau_*^{k+1}|\overline{a}\langle 0\rangle]\!] = (k+1)_{\{*\}}$ and $F(a,k) \in SN_{\tau(*)}(k+1)$ and $F(a,k) \in SN_{\tau(a)}(0)$.

Example 8. Consider the process $F(a,k)$ with respect to the process definition:

$$\begin{aligned} F(c,n) = \texttt{Case}\ c,n\ \texttt{of}\ & c,0 \to \overline{c}\langle 0\rangle \\ & c,m+1 \to F(c,m)|F(c,m) \end{aligned}$$

It admits the following interpretation $[F](c,n) = \tau_*^{2^{2n}}|\underbrace{\overline{c}\langle 0\rangle|\ldots|\overline{c}\langle 0\rangle}_{2^{n+1}\text{ times}}$ since:

$$\begin{aligned} [F(c,0)] &= \tau_*^1|\overline{c}\langle 0\rangle|\overline{c}\langle 0\rangle \gtrsim \tau_*^1.\overline{c}\langle 0\rangle = \tau_*^1.[\overline{c}\langle 0\rangle] \\ [F(c,m+1)] &= \tau_*^{2^{2m+4}}|\underbrace{\overline{c}\langle 0\rangle|\ldots|\overline{c}\langle 0\rangle}_{2^{m+2}\text{ times}} \\ &\gtrsim \tau_*^1.((\tau_*^{2^{2m+2}}|\underbrace{\overline{c}\langle 0\rangle|\ldots|\overline{c}\langle 0\rangle}_{2^{m+1}\text{ times}})|(\tau_*^{2^{2m+2}}|\underbrace{\overline{c}\langle 0\rangle|\ldots|\overline{c}\langle 0\rangle}_{2^{m+1}\text{ times}})) \\ &= \tau_*^1.[F(c,m)|F(c,m)] \end{aligned}$$

Moreover, $[\![[F(a,k)]]\!] = (2^{2k})_{\{*\}}$ and $F(a,k) \in SN_{\tau(*)}(2^{2k})$ and $F(a,k) \in SN_{\tau(a)}(0)$.

Example 9. As a counter-example consider the process definition:

$$bip() = bip()|P$$

for some process P. Clearly there is no assignment $[-]$ such that $[bip()] \gtrsim \tau_*^1.[bip()|P]$ since $[bip()] \gtrsim \tau_*^1.[bip()]$ does not hold, for any $[-]$, which is reasonable since the process $bip()$ does not terminate (See Theorem 2).

Example 10. Consider the process $F(a,b,i,j)$ wrt the process definition:

$$\begin{aligned} F(a,b,n,m) = \texttt{Case}\ a,b,n,m\ \texttt{of}\ & a,b,0,0 \to \texttt{0} \\ & a,b,0,m+1 \to F(a,b,m+1,m) \\ & a,b,n+1,m \to (a(x).\overline{b}x|\overline{a}v) + F(a,b,n,m) \end{aligned}$$

It admits the following interpretation $[F](a,b,n,m) = \tau_*^{n+(m+1)^2}|(a(x).\overline{b}x|\overline{a}v)$ since:

$$\begin{aligned} [F(a,b,0,0)] &= \tau_*^1 + (a(x).\overline{b}x|\overline{a}v) \geq \tau_*^1.\texttt{0} = \tau_*^1.[\texttt{0}] \\ [F(a,b,0,m+1)] &= \tau_*^{(m+2)^2}|(a(x).\overline{b}x|\overline{a}v) \\ &\gtrsim \tau_*^1.(\tau_*^{m+1+(m+1)^2}|(a(x).\overline{b}x|\overline{a}v)) \\ &= \tau_*^1.[F(a,b,m+1,m)] \\ [F(a,b,n+1,m)] &= \tau_*^{n+1+(m+1)^2}|(a(x).\overline{b}x|\overline{a}v) \\ &\gtrsim \tau_*^1.((\tau_*^{n+(m+1)^2}|(a(x).\overline{b}x|\overline{a}v)) + (a(x).\overline{b}x|\overline{a}v) \\ &= \tau_*^1.[F(a,b,n,m) + a(x).\overline{b}x|\overline{a}v] \end{aligned}$$

Moreover, $[\![[F(a,b,i,j)]]\!] = (i+(j+1)^2)_{\{*\}} + 1_{\{a\}}$. Consequently, $F(a,b,i,j) \in SN_{\tau(*)}(k+(k+1)^2)$ and $F(a,b,i,j) \in SN_{\tau(a)}(1)$.

5 Extension to space upper bounds

In this section, we show that playing on the parameters of the analysis (language, simulation, semantics,...), we may infer several other challenging results about resource consumption of processes. In particular, interpretations also allow us to infer upper bounds about the maximal size of the values sent on each channel during a communication. For that purpose, we put extra annotations, an upper bound on the integer value sent, on the rule (Com) of Figure 2 as follows:

$$\frac{P \overset{\overline{x}w}{\to} P' \quad Q \overset{xw}{\to} Q'}{P \mid Q \overset{\tau(x(\#w))}{\to} P' \mid Q'}\ (\text{Com}) \qquad \text{with } \#w = \begin{cases} 0, \text{ if } w \in \mathbb{D} \\ w, \text{ if } w \in \mathbb{N} \end{cases}$$

The rule (Close) can be annotated in a similar manner. Moreover, these annotation are extended to resource processes by $\tau_{\chi(k)}^{n+1}.R \overset{\tau(\chi(k))}{\to} \tau_{\chi(k)}^{n}.R$ and to the

preorder $\succeq$ in a standard way. The aim of these new annotations is to keep record of the integer value sent on a given channel or 0 by default (if a name has been sent or if a process definition has been applied).

Define $Reach^{\chi(k)}(R) = \{S \mid R \overset{\tau(\chi(k))}{\rightarrow} S\}$. We are now ready to introduce a new semantics $(\!|-|\!)$, a partial map from $\mathcal{R}(\mathrm{N})$ to $\mathbb{N}^{\mathbb{D}}$, for space analysis:

$$(\!|R|\!) = \oplus\{k_{\{\chi\}} \oplus (\!|S|\!) \mid \forall \chi \in \mathbb{D},\ \forall k \in \mathbb{N},\ \forall S \in Reach^{\chi(k)}(R)\} \quad \text{if } Reach(R) \neq \emptyset$$
$$(\!|R|\!) = \delta_0 \quad \text{otherwise}$$

Now we obtain an upper bound on each integer value sent during reaction:

Theorem 3 (Space upper bound). *Given a closed process P_0 of interpretation $[-]$, for each sequence of reduction $P_0 \overset{\tau(\chi_1(n_1))}{\rightarrow} P_1 \ldots \overset{\tau(\chi_k(n_k))}{\rightarrow} P_k$, we have $(\!|[P_0]|\!)(\chi_i) \geq n_i$.*

Proof. There are two cases to consider:

- If $\chi_{i+1} \in \mathrm{N}$ then, by Lemma 3[2], we have $[P_i] \overset{\tau(\chi_{i+1}(n_{i+1}))}{\rightarrow} [P_{i+1}]$ and $[P_{i+1}] \in Reach^{\chi(n_{i+1})}([P_i]) \neq \emptyset$. By definition of $(\!|-|\!)$, we obtain that $(\!|[P_i]|\!) \succeq n_{i+1\{\chi_{i+1}\}} \oplus (\!|[P_{i+1}]|\!)$ and, consequently, $(\!|[P_0]|\!)(\chi_{i+1}) \geq (\!|[P_i]|\!)(\chi_{i+1}) \geq n_{i+1}$, by transitivity and by definition of $\oplus$.
- If $\chi_{i+1} = *$ then trivially $n_{i+1} = 0$ (by definition of the # operator) and the result holds. □

Example 11. For the process *fac* of Example 5, the assignment: $[fac](k, a) = (\nu r')\tau^2_{r'(1)}|\tau^1_{r'(2)}|\ldots|\tau^1_{r'((k-1)!)}|\tau^{k+1}_{*(0)}|\overline{a}\langle k!\rangle$, is an interpretation and, consequently, we have $(\!|[fac(k, a)]|\!) = (k-1)!_{\{r'\}}$ which means that $(k-1)!$ is an upper bound on the values sent on the internal names r' whereas 0 is an upper bound on the values sent on all the other name (indeed, the value on a has not been sent yet).

Remark 1. Note that the preorder $\gtrsim$ might be seen as too restrictive for space control. Indeed it requires the simulating resource process to output exactly the same integers than the simulated process. Since we are only interested in an upper bound, $\gtrsim$ can be replaced by a finer relation $\gtrapprox$ defined as follows: Given two labels α and β, $\beta \gtrapprox \alpha$ holds if either $\beta = \alpha$ or $\beta = \tilde{\nu}\overline{x}n$, $\alpha = \tilde{\nu}\overline{x}m$, $n, m \in \mathbb{N}$, $x \in \mathrm{N}$ and $n \geq m$. The partial preorder $\gtrapprox$ on resource processes is a simulation defined by $R \gtrapprox R'$ if $\forall S' \in \mathcal{R}(\mathrm{N})$ s.t. $R' \overset{\alpha}{\rightarrow} S'$, $\exists S \in \mathcal{R}(\mathrm{N})$ s.t. $R \overset{\beta}{\rightarrow} S$, $S \gtrapprox S'$ and $\beta \gtrapprox \alpha$. We claim that Theorem 3 holds if we substitute $\gtrapprox$ to $\gtrsim$ in the definition of interpretations. Turning back to Example 11, the interpretation of *fac* can be set to $[fac](k, a) = (\nu r')\tau^k_{r'((k-1)!)}|\tau^{k+1}_{*(0)}|\overline{a}\langle k!\rangle$.

6 Some words on inference.

An important issue related to the presented analysis is interpretation inference which consists in finding an assignment that is an interpretation of a given

[2] This Lemma trivially remains valid for the new annotations.

process P. Interpretation inference is clearly undecidable in general since finding an interpretation is as difficult as the halting problem (see Theorem 2). The undecidability of such a problem lies in the fact that one needs to guess functions f from $\mathbb{N}$ to $\mathbb{N}$ as annotations of the resource processes (for example, $\tau_*^{f(n)}$) and to check the simulation preorder for each process definition. Note that the difficulty of this latter check lies in the parametric definition and not in the simulation itself since resource processes are strongly normalizing (they have neither the ability to recurse, nor the ability to replicate). This result is not surprising if we compare our notion of interpretation to the notion of polynomial interpretation for term rewrite systems, also known to be an undecidable tool. As for polynomial interpretations, the inference problem will become decidable if we restrict the class of functions under consideration (max-plus, bounded polynomials, ...).

For a given process P such that $D(P) = \{F_1, \cdots, F_n\}$, the inferred assignments $[F_i](\overrightarrow{n}, \overrightarrow{a})$ have the shape:

$$\nu\,\overrightarrow{r}(\tau_{r_1}^{f^i_{r_1}(\overrightarrow{n})}|\dots|\tau_{r_l}^{f^i_{r_l}(\overrightarrow{n})})|\tau_{a_1}^{f^i_{a_1}(\overrightarrow{n})}|\dots|\tau_{a_k}^{f^i_{a_k}(\overrightarrow{n})}|\tau_*^{f^i_*(\overrightarrow{n})}|f^i_{\overrightarrow{a_1}}(a_1,\overrightarrow{n})|\dots|f^i_{\overrightarrow{a_k}}(a_k,\overrightarrow{n})$$

where $\overrightarrow{r} = r_1, \cdots, r_l$ corresponds to the names of P under a restriction, $fv(P) = \{\overrightarrow{n}\}$ and $\{\overrightarrow{a}\} = \{a_1, \cdots, a_k\} = fn(P)$. Consequently, the interpretation inference consists in finding all the functions $f^i_\chi : \mathbb{N}^k \to \mathbb{N}$ and checking inequalities. As demonstrated for polynomial interpretation, this problem becomes decidable for small classes of functions (it is NP-difficult for some max-plus algebra [1]).

7 Conclusion.

The work presented in this paper is the most intuitive and simplest version in the set of all possible applications. Indeed we claim that playing on the parameters of the analysis (language, simulation, semantics,...), we may infer several other challenging results about resource consumption of processes:

- the analysis may be extended in a simple way to other inductive data structures such as lists, trees...
- the framework can be extended in order to capture processes that do not terminate but still have bounded reactions on some channels.
- Finally, we can capture complexity classes, such as FPTIME, in a classical way by restricting the considered processes to processes computing functions.

References

1. R. Amadio. Synthesis of max-plus quasi-interpretations. *Fundamenta Informaticae*, 65(1–2), 2005.
2. P. Baillot and V. Mogbil. Soft lambda-calculus: a language for polynomial time computation. In *FOSSACS 2004*, vol. 2987 of *LNCS*, pages 27–41. Springer, 2004.
3. S. Bellantoni and S. Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2:97–110, 1992.

4. G. Bonfante, J.-Y. Marion, and J.-Y. Moyen. Quasi-interpretations a way to control resources. *Theoretical Computer Science*, 412(25):2776–2796, 2011.
5. M. Collinson, B. Monahan, and D. J. Pym. A logical and computational theory of located resource. *J. Log. Comput.*, 19(6):1207–1244, 2009.
6. R. Demangeon, D. Hirschkoff, and D. Sangiorgi. Termination in higher-order concurrent calculi. *J. Log. Algebr. Program.*, 79(7):550–577, 2010.
7. Y. Deng and D. Sangiorgi. Ensuring termination by typability. *Information and Computation*, 204(7):1045–1082, 2006.
8. J. Feret. Occurrence counting analysis for the pi-calculus. *Electronic Notes in Theoretical Computer Science*, 39(2):1–18, 2001.
9. M. Gaboardi, S.R. Della Rocca, and J.Y. Marion. A Logical Account of PSPACE. *ACM SIGPLAN-IGACT POPL*, 2008.
10. M. Gaboardi and S. Ronchi Della Rocca. A soft type assignment system for λ-calculus. vol. 4646 of *LNCS*, pages 253–267. Springer, 2007.
11. J.-Y. Girard. Light linear logic. *I.&C.*, 143(2):175–204, 1998.
12. R.R. Hansen, J.G. Jensen, F. Nielson, and H.R. Nielson. Abstract interpretation of mobile ambients. In *SAS*, vol. 1664 of *LNCS*, pages 134–148. Springer, 1999.
13. M. Hennessy. A calculus for costed computations. *Logical Methods in Computer Science*, 7(1), 2011.
14. A. Igarashi and N. Kobayashi. Type reconstruction for linear -calculus with i/o subtyping. *Inf. Comput.*, 161(1):1–44, 2000.
15. N.D. Jones and Lars Kristiansen. A flow calculus of *mwp*-bounds for complexity analysis. *ACM Trans. Comput. Log.*, 10(4), 2009.
16. N. Kobayashi, M. Nakade, and A. Yonezawa. Static analysis of communication for asynchronous concurrent programming languages. In *SAS*, vol. 983 of *LNCS*, pages 225–242. Springer, 1995.
17. N. Kobayashi, K. Suenaga, and L. Wischik. Resource usage analysis for the π-calculus. In *VMCAI*, vol. 3855 of *LNCS*, pages 298–312. Springer, 2006.
18. B. König. Analysing input/output-capabilities of mobile processes with a generic type system. *Automata, Languages and Programming*, pages 403–414, 2000.
19. U. Dal Lago, S. Martini, and D. Sangiorgi. Light logics and higher-order processes. In *EXPRESS'10*, vol. 41 of *EPTCS*, pages 46–60, 2010.
20. D.S. Lankford. On proving term rewriting systems are noetherian. Technical report, 1979.
21. D. Leivant. A foundational delineation of poly-time. *I.&C.*, 110(2):391–420, 1994.
22. D. Leivant and J.-Y. Marion. Lambda calculus characterizations of poly-time. *Fundamenta Informaticae*, 19(1,2):167,184, September 1993.
23. Z. Manna and S. Ness. On the termination of Markov algorithms. In *Third hawaii international conference on system science*, pages 789–792, 1970.
24. R. Milner. *Communicating and mobile systems: the π-calculus.* Cambridge, 1999.
25. J.-Y. Moyen. Resource control graphs. *ACM Trans. Comput. Log.*, 10(4), 2009.
26. H.R. Nielson and F. Nielson. Shape analysis for mobile ambients. In *Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 142–154. ACM, 2000.
27. D. Sangiorgi. Termination of processes. *Mathematical Structures in Computer Science*, 16(1):1–39, 2006.
28. T. Terauchi and A. Megacz. Inferring channel buffer bounds via linear programming. In *ESOP*, vol. 4960 of *LNCS*, pages 284–298, 2008.
29. N. Yoshida, M. Berger, and K. Honda. Strong normalisation in the pi-calculus. *Inf. Comput.*, 191(2):145–202, 2004.