On the Expressiveness of Polyadicity in Higher-Order Process Calculi (Extended Abstract)*

Ivan Lanese[1], Jorge A. Pérez[1], Davide Sangiorgi[1], and Alan Schmitt[2]

[1] Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy
[2] INRIA Grenoble - Rhône Alpes, France

Abstract. In higher-order process calculi the values exchanged in communications may contain processes. We describe a study of the expressive power of *strictly* higher-order process calculi, i.e. calculi in which only process passing is allowed and no name-passing is present. In this setting, the polyadicity (i.e. the number of parameters) allowed in communications is shown to induce a hierarchy of calculi of strictly increasing expressiveness: a higher-order calculus with n-adic communication cannot be encoded into a calculus with $n-1$-adic communication. In this note we outline this result, and discuss the conditions under which it holds.

Introduction. *Higher-order* process calculi are formal languages for concurrency in which processes can be communicated. They have been put forward in the early 1990s, with CHOCS [1] and Plain CHOCS [2], the Higher-Order π-calculus [3], and others. Recent proposals of higher-order calculi include the Kell calculus [4] and Homer [5]. Higher-order, or process-passing, concurrency is often presented as an alternative paradigm to the first order, or name-passing, concurrency of the π-calculus for the description of mobile systems. Higher-order calculi are inspired by, and are formally closer to, the λ-calculus, whose basic computational step — β-reduction — involves term instantiation.

An important criterion for assessing the significance of a paradigm is its *expressiveness*. The expressiveness of higher-order communication has received little attention in the literature; previous works are mostly concerned about issues of *relative* expressiveness between higher- and first-order calculi. A good example is [6] in which a tight correspondence between name-passing calculi based on internal mobility and process-passing calculi is shown. In a previous work, we have studied expressiveness and decidability issues for HOCORE, a core calculus for higher order concurrency [7]. HOCORE is a *strictly* higher-order process calculus, in that only the operators necessary to obtain higher-order communications are retained. Notably, no name-passing features are present. The grammar of HOCORE is:

$$P ::= a(x).\, P \;\big|\; \overline{a}P \;\big|\; P \parallel P \;\big|\; x \;\big|\; \mathbf{0}$$

An input prefixed process $a(x).\, P$ can receive on name (or channel) a a process that will be substituted in the place of x in the body P; an output message $\overline{a}P$ can send P on

* Research partially supported by the INRIA Équipe Associeé BACON.

a; parallel composition allows processes to interact. HOCORE is *minimal* in that continuations following output messages have been left out (i.e. communication is asynchronous) and, more importantly, it has no restriction operator. Thus all channels are global, and dynamic creation of new channels is impossible. This makes the absence of recursion also relevant, as known encodings of fixed-point combinators in higher-order process calculi require the restriction operator. Despite this minimality, HOCORE was shown to be Turing complete. Therefore, in HOCORE, properties such as *termination* (i.e. non existence of divergent computations) and *convergence* (i.e. existence of a terminating computation) are both undecidable. In contrast, somewhat surprisingly, strong bisimilarity is decidable, and several sensible bisimilarities in the higher-order setting coincide with it. A recent work [8] has studied a fragment of HOCORE in which output actions have limited capabilities over previously received processes. In such a fragment, similarly as in HOCORE, convergence is undecidable but, unlike HOCORE, termination is decidable.

This Work. In this note we continue our study of the fundamental properties of higher-order process calculi. We shall analyze the consequences that *polyadicity* (i.e. the number of parameters in higher-order communications) has on the expressiveness of this kind of calculi. We consider variants of HOCORE with different degrees of polyadicity, and study their relative expressive power. Our main result is a hierarchy of calculi of strictly increasing expressiveness: a higher-order process calculus with n-adic communication cannot be encoded into a calculus with $n-1$-adic communication. In the remainder, as a way of introducing the peculiarities of the higher-order setting, we discuss the classic encoding of polyadic first-order communication into monadic one; then, we comment on a notion of encoding with a refined account of internal communications. Our result depends critically on this notion. We conclude by giving intuitions on the proof of the main result; more details can be found in [9].

Our Setting. We recall the encoding of the polyadic π-calculus into the monadic one in [10]:

$$\llbracket x(z_1, \ldots, z_n).\, P \rrbracket = x(w).\, w(z_1). \cdots .\, w(z_n).\, \llbracket P \rrbracket$$
$$\llbracket \overline{x}\langle a_1, \ldots, a_n \rangle.\, P \rrbracket = \nu w\, \overline{x}w.\, \overline{w}a_1. \cdots .\, \overline{w}a_n.\, \llbracket P \rrbracket$$

(where $\llbracket \cdot \rrbracket$ is an homomorphism for the other operators). A single n-adic synchronization is encoded as $n+1$ monadic synchronizations. The first such synchronizations establishes a *private link* w: the encoding of output creates a private name w and sends it to the encoding of input. In other words, by virtue of the synchronization on x, private name w is now shared, and its scope is *extruded*. As a result, name w can be used to communicate each of $a_1, \ldots, a_n$ through (monadic) synchronizations.

This encoding is very intuitive, and satisfies a tight operational correspondence property: a public synchronization of the source term (i.e. a synchronization on an unrestricted name such as x) is matched by the encoding with a public synchronization on the same name that is followed by a number of internal synchronizations (i.e. synchronizations on a private name such as w). The observable behavior is thus preserved: the encoding does not perform visible actions different from those performed by the source term. Also, thanks to the ability of setting a private link on w, the encoding is *robust with respect to interferences*: once the private link has been established between two parties, any surrounding —possibly malicious— context cannot get access to the monadic communications on w. We observe that both name-passing and a disciplined use of private links are crucial for an *atomic* implementation of polyadic communication.

Higher-order process calculi are much more constrained: in the absence of name-passing scope extrusions have only a partial effect. Let us explain what we mean by this. In a process-passing setting, received processes can only be executed, forwarded, or discarded. A receiving context then cannot gain access to the (private) names of those processes it receives; to the context these are much like a "black box". Although higher-order communications might lead to scope extrusion of the private names contained in the transmitted processes, such extrusions are vacuous: without name-passing, a receiving context can not *make use* of such names. (It is worth remarking that here we understand process-passing that does *not* consider abstraction-passing, i.e. the communication of functions from processes to processes. With abstraction-passing the situation is rather different.) This way, e.g., in the communication of a process P with an (input, ouput) capability on a restricted name x, any receiver R will not be able to exploit the (input, output) capability of P on x. In a loose analogy with security protocols, the communication of private names with process-passing only would correspond to the communication of an encrypted message to some recipient that does not have the key to decrypt the message (nor a way of obtaining it). The sharing of (private) names one obtains from using process-passing only is then incomplete: names can be sent (as part of processes) but they cannot be actually used. This suggests that an encoding of polyadic process-passing into monadic process-passing that enjoys basic properties (notably, compositionality and robustness with respect to interferences) might not exist. However, formalizing this intuition into a non encodability result is far from trivial.

Our Notion of Encoding. A crucial aspect in any expressiveness study is the notion of encoding considered. There is no general agreement on the definition of "good" encoding, even if this has been a matter of discussion for several years now, and proposals have been made (see, e.g., [11]). A notion of encoding usually consists of syntactic and semantic criteria; the former state basic requirements on the translation of terms (such as, e.g., compositionality), whereas the latter define the relationship between the behavior of a term and that of its encoding. In the case of negative results, as in our case, one is interested in the most liberal notion possible, for the results to hold for the largest possible class of encodings.

To prove our main result, we shall require a notion of encoding that departs from "traditional" notions by taking a more refined standpoint with respect to internal communications. Indeed, in our notion, semantic criteria enforce the distinction between public and internal synchronizations discussed for the encoding in the first-order case. This is achieved by decreeing synchronizations on public names to be *visible* actions of the associated labeled transition system. This way, we use $\xrightarrow{a\tau}$ to denote the synchronization on a public name a. As a result, internal behavior $\xrightarrow{\tau}$ is only obtained by synchronizations on restricted names. This kind of distinctions between internal and public communications is studied in, e.g., [12]. As customary, we use $\Rightarrow$ to stand for the reflexive, transitive closure of $\xrightarrow{\tau}$. Given an action α, notation $\overset{\alpha}{\Longrightarrow}$ stands for $\Rightarrow\xrightarrow{\alpha}\Rightarrow$. The observability predicates (barbs) are defined in the expected way. Given a process P and a name a, we write $P\downarrow_a$ (resp. $P\downarrow_{\overline{a}}$) if P can perform an input (resp. output) action with subject a. Given $\mu \in \{a, \overline{a}\}$, we define a *weak* barb $P \Downarrow_\mu$ if, for some P', $P \Rightarrow P' \downarrow_\mu$.

We are now ready to state our notion of encoding. It comprises four of the five criteria suggested in [11]. The definition of encoding assumes a *language* $\mathcal{L}$ is defined as a set of processes $\mathcal{P}$, a behavioral equivalence $\approx$, and a reduction relation $\longrightarrow$ (which is characterized by the τ actions of a labeled transition system). A *translation* is an injective function from a *source* language $\mathcal{L}_\mathsf{s} = (\mathcal{P}_\mathsf{s}, \approx_\mathsf{s}, \longrightarrow_\mathsf{s})$ into a *target* language $\mathcal{L}_\mathsf{t} = (\mathcal{P}_\mathsf{t}, \approx_\mathsf{t}, \longrightarrow_\mathsf{t})$. An *encoding* is then a translation that satisfies the following syntactic and semantic conditions. (Below we use notation $\mathsf{fn}(P)$ to stand for the set of free names of a process P.)

Definition 1 *Suppose a translation* $[\![\cdot]\!] : \mathcal{P}_\mathsf{s} \to \mathcal{P}_\mathsf{t}$*. We say that* $[\![\cdot]\!]$ *is*

1. Compositional*: if for every k-ary operator* op *of* $\mathcal{L}_\mathsf{s}$ *and for all* $S_1, \ldots, S_k$ *with* $\mathsf{fn}(S_1, \ldots, S_k) = N$*, then there exists a k-ary context* $C^N_\mathsf{op} \in \mathcal{P}_\mathsf{t}$ *such that*

$$[\![\mathsf{op}(S_1, \ldots, S_k)]\!] = C^N_\mathsf{op}[[\![S_1]\!], \ldots, [\![S_k]\!]].$$

2. Name invariant*: if* $[\![\sigma(P)]\!] = \sigma([\![P]\!])$*, for any injective permutation of names* σ*.*

Definition 2 *Suppose a translation* $[\![\cdot]\!] : \mathcal{P}_\mathsf{s} \to \mathcal{P}_\mathsf{t}$*.*

1. *We say that* $[\![\cdot]\!]$ *is* operational corresponding *if the following properties hold:*
   (a) Completeness/Preservation*: For every* $S, S' \in \mathcal{P}_\mathsf{s}$ *such that* $S \overset{\alpha}{\Longrightarrow}_\mathsf{s} S'$*, it holds that* $[\![S]\!] \overset{\alpha}{\Longrightarrow}_\mathsf{t} \approx_\mathsf{t} [\![S']\!]$
   (b) Soundness/Reflection*: For every* $S \in \mathcal{P}_\mathsf{s}$*,* $T \in \mathcal{P}_\mathsf{t}$ *such that* $[\![S]\!] \overset{\alpha}{\Longrightarrow}_\mathsf{t} T$ *there exists an* $S' \in \mathcal{P}_\mathsf{s}$ *such that both* $S \overset{\alpha}{\Longrightarrow}_\mathsf{s} S'$ *and* $T \overset{\alpha}{\Longrightarrow}_\mathsf{t} \approx_\mathsf{t} [\![S']\!]$*.*

Furthermore, we shall require adequacy*: if* $P \approx_\mathsf{s} Q$ *then* $[\![P]\!] \approx_\mathsf{t} [\![Q]\!]$*.*

Notice that adequacy is necessary because we make no assumptions on the nature of $\approx_\mathsf{s}$ and $\approx_\mathsf{t}$.

Definition 3 (Encoding) *We shall call* encoding *any translation that satisfies both the syntactic conditions in Def. 1 and the semantic conditions in Def. 2.*

A Hierarchy of Higher-Order Process Calculi. We now give an intuition on our main result: a hierarchy of higher-order process calculi that is induced by the degree of polyadicity associated to each such languages. As for the calculi, we shall consider HOπ^m, the family of languages that is a synchronous variant of HOCORE, with polyadicity $m \geq 1$, and extended with the construct $\nu r\, P$ which allows to (statically) define a private name r in the scope of P. This way, e.g., HOπ^1 stands for a strictly higher-order calculus allowing monadic process passing. We first discuss the non existence of an encoding (cf. Def. 3) of HOπ^2 into HOπ^1. Then, we generalize such a result so as to define a hierarchy of expressiveness in which the calculus HOπ^{n+1} is strictly more expressive than the calculus HOπ^{n}, for any $n > 0$.

The result relies on the intuition mentioned above: higher-order processes cannot share private names by means of process passing only. The set of private names of a process then remains invariant along its evolution. This is captured by the notion of *disjoint form* (below, we use $\widetilde{P}$ to represent the sequence $P_1, \ldots, P_k$, for some $k > 1$):

Definition 4 (Disjoint Form) *Let $T \equiv \nu\widetilde{n}(P \parallel C[\widetilde{R}])$ be a HOπ^1 process where*

1. *$\widetilde{n}$ is a set of names such that $\widetilde{n} \subseteq \mathsf{fn}(P, \widetilde{R})$ and $\widetilde{n} \cap \mathsf{fn}(C) = \emptyset$;*
2. *C is a k-ary (multihole) context;*
3. *$\widetilde{R}$ contains k guarded, closed processes.*

*We then say that T is in k-*adic disjoint form with respect to $\widetilde{n}$, $\widetilde{R}$, and P.

Intuitively, a disjoint form $T \equiv \nu\widetilde{n}(P \parallel C[\widetilde{R}])$ represents a result of a public synchronization in which a series of objects $\widetilde{R}$ has been transmitted. Since $\widetilde{R}$ and its (receiving) context C only share public names —and sharing of private names is not possible—, their private names cannot really "mix": C is not able to gain access to the private names in $\widetilde{R}$ and viceversa. The arity of $\widetilde{R}$ corresponds to the degree of polyadicity considered; this way, in the case of HOπ^1, one has *monadic* disjoint forms, with a single R. Similarly, when the arity of R is 2, one obtains *biadic* disjoint forms. Disjoint forms (of any arity) enjoy the following properties:

Lemma 5 (Stability of Disjoint Forms) *Disjoint forms are preserved by internal synchronizations and output actions that do not extrude names. Also, the notion of encoding in Def. 3 respects disjoint forms: if T is in disjoint form with respect some $\widetilde{n}$, $\widetilde{R}$, and P then $[\![T]\!]$ is in disjoint form with respect some $\widetilde{n}$, $[\![\widetilde{R}]\!]$, and $[\![P]\!]$ too.*

Lemma 6 (Swapping Lemma) *Let $T \equiv \nu\widetilde{n}(P \parallel C[\widetilde{R}])$ be a disjoint form as in Def. 4. If $T \xrightarrow{\alpha}\xrightarrow{\beta} T'$, where α originates in P and β originates in $C[\widetilde{R}]$, then $T \xrightarrow{\beta}\xrightarrow{\alpha} T'$ also holds.*

The following theorem gives the base case of the hierarchy of languages.

Theorem 7 *There is no encoding of HOπ^2 into HOπ^1.*

Proof (Sketch). The proof proceeds by contradiction, assuming such an encoding does exist. Take the following HOπ^2 process

$$\begin{aligned} P_0 = {} & \nu m_1, m_2\, (\overline{a}\langle S_1,\, S_2\rangle.\, \mathbf{0}) \parallel \\ & \quad \nu b\, (a(x_1, x_2).\, (\overline{b}\langle \overline{b_1}.\, x_1\rangle.\, \mathbf{0} \parallel \overline{b}\langle \overline{b_2}.\, x_2\rangle.\, \mathbf{0} \parallel b(y_1).\, b(y_2).\, y_1) \end{aligned}$$

where S_1, S_2 have different private names (m_1 and m_2, resp.) and observable behavior. While after the communication on a, P_0 evolves into a P that is in biadic disjoint form, after the corresponding synchronization on a the HOπ^1 process $[\![P]\!]$ can be shown to be in monadic disjoint form. Once in P, either S_1 or S_2 is executed; this depends on a mutually exclusive choice that relies on the private name b. Notice that P only involves output actions and *internal* synchronizations. By completeness, P and $[\![P]\!]$ should be behaviorally equivalent; during the bisimulation game, since output actions and *internal* synchronizations preserve (monadic) disjoint forms (Lemma 5), $[\![P]\!]$ (and its derivatives) will be in (monadic) disjoint form as well. Using Lemma 6 it is possible to prove that certain enabling capabilities associated to the mutually exclusive choice of the source term are lost in its encoding, basically because they rely on a private name. As a result, the encoded term (which is in monadic disjoint form) can exhibit observable behavior that is different from the observable behavior in the source term (which is in biadic disjoint form), thus leading to a fail in the bisimulation game and therefore to a contradiction. $\square$

The above lemma can be generalized by a straightforward extension of the notion and properties of disjoint forms to processes with arbitrary polyadicity. As a result, we have the following theorem, which defines the desired hierarchy of strictly higher-order process calculi:

Theorem 8 *For any $n \geq 1$, there is no encoding of HOπ^{n+1} into HOπ^{n}.*

Concluding Remarks. We have given a very succinct account of an expressiveness study of polyadicity in process calculi with a process-passing communication discipline. By identifying the limits of process-passing, we have better understood the significance of name-passing in this setting, and have deepen our understanding of higher-order calculi as a whole. Current investigations study the expressiveness that communication of abstractions (i.e. functions from processes to processes) can have on the kind of languages discussed here.

References

1. Thomsen, B.: A calculus of higher order communicating systems. In: Proc. of POPL'89, ACM Press (1989) 143–154
2. Thomsen, B.: Plain CHOCS: A second generation calculus for higher order processes. Acta Inf. **30**(1) (1993) 1–59
3. Sangiorgi, D.: Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms. PhD thesis CST–99–93, University of Edinburgh, Dept. of Comp. Sci. (1992)
4. Schmitt, A., Stefani, J.B.: The kell calculus: A family of higher-order distributed process calculi. In: Proc. of Global Computing. Volume 3267 of LNCS., Springer (2004) 146–178
5. Hildebrandt, T., Godskesen, J.C., Bundgaard, M.: Bisimulation congruences for homer — a calculus of higher order mobile embedded resources. Technical Report TR-2004-52, IT University of Copenhagen (2004)
6. Sangiorgi, D.: π-calculus, internal mobility and agent-passing calculi. Theor. Comput. Sci. **167**(2) (1996) 235–274
7. Lanese, I., Pérez, J.A., Sangiorgi, D., Schmitt, A.: On the expressiveness and decidability of higher-order process calculi. In: Proc. of LICS'08, IEEE Computer Society (2008) 145–155
8. Di Giusto, C., Pérez, J.A., Zavattaro, G.: On the Expressiveness of Forwarding in Higher-Order Communication. In: Proc. of ICTAC'09. Volume 5684 of LNCS., Springer (2009) 155–169
9. Lanese, I., Pérez, J.A., Sangiorgi, D., Schmitt, A.: Separation Results for Higher-Order Process Calculi (Preliminary Title) (2009) In preparation. Available at `http://www.cs.unibo.it/~perez/hocore`.

10. Milner, R.: The Polyadic pi-Calculus: A Tutorial. Technical Report ECS-LFCS-91-180, University of Edinburgh (1991)
11. Gorla, D.: Towards a unified approach to encodability and separation results for process calculi. In: Proc. of CONCUR. Volume 5201 of Lecture Notes in Computer Science., Springer (2008) 492–507
12. Lanese, I.: Concurrent and located synchronizations in *pi*-calculus. In: Proc. of SOFSEM. Volume 4362 of Lecture Notes in Computer Science., Springer (2007) 388–399