On the Expressiveness of Forwarding in Higher-Order Communication

Cinzia Di Giusto, Jorge A. Pérez, and Gianluigi Zavattaro

Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy

Abstract. In higher-order process calculi the values exchanged in communications may contain processes. There are only two capabilities for received processes: execution and forwarding. Here we propose a limited form of forwarding: output actions can only communicate the parallel composition of statically known closed processes and processes received through previously executed input actions. We study the expressiveness of a higher-order process calculus featuring this style of communication. Our main result shows that in this calculus termination is decidable while convergence is undecidable.

1 Introduction

Higher-order process calculi are calculi in which processes can be communicated. They have been put forward in the early 1990s, with CHOCS [1], Plain CHOCS [2], the Higher-Order π-calculus [3], and others. 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. These calculi are inspired by, and formally closer to the λ-calculus, whose basic computational step — β-reduction — involves term instantiation. As in the λ-calculus, a computational step in higher-order calculi results in the instantiation of a variable with a term, which is then copied as many times as there are occurrences of the variable.

HOCORE is a core calculus for higher-order concurrency, recently introduced in [4]. It is *minimal*, in that only the operators strictly necessary to obtain higher-order communications are retained. This way, continuations following output messages have been left out, so communication in HOCORE is asynchronous. More importantly, HOCORE 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. The grammar of HOCORE processes is:

$$P ::= a(x).\,P \;\mid\; \overline{a}\langle P\rangle \;\mid\; P \parallel P \;\mid\; x \;\mid\; \mathbf{0} \qquad (*)$$

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 $\overline{a}\langle P\rangle$ can send P (the output object) on a; parallel composition allows processes to interact. Despite this minimality, via a termination preserving encoding of Minsky machines [5], 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)[1] are both undecidable. In contrast, somewhat surprisingly, strong bisimilarity is decidable, and several sensible bisimilarities in the higher-order setting coincide with it.

In this paper, we shall aim at identifying the intrinsic source of expressive power in HOCORE. A substantial part of the expressive power of a concurrent language comes from the ability of accounting for infinite behavior. In higher-order process calculi there is no explicit operator for such a behavior, as both recursion and replication can be encoded. We then find that infinite behavior resides in the interplay of higher-order communication, in particular, in the ability of *forwarding* a received process within an *arbitrary context*. For instance, consider the process $R = a(x).\,\overline{b}\langle P_x\rangle$ (here P_x stands for a process P with free occurrences of a variable x). Intuitively, R receives a process on name a and forwards it on name b. It is easy to see that since objects in output actions are built following the syntax given by $(*)$, the actual structure of P_x can be fairly complex. One could even "wrap" the process to be received in x using an arbitrary number of k "output layers", i.e., by letting $P_x \equiv \overline{b_1}\langle\overline{b_2}\langle\ldots\overline{b_k}\langle x\rangle\rangle\ldots\rangle$. This *nesting capability* embodies a great deal of the expressiveness of HOCORE: as a matter of fact, the encoding of Minsky machines in [4] depends critically on nesting-based counters. Therefore, investigating suitable limitations to the kind of processes that can be communicated in an output action appears as a legitimate approach for assessing the expressive power of higher-order concurrency.

With the above consideration in mind, in this paper we propose $\text{HO}^{-\mathsf{f}}$, a sublanguage of HOCORE in which output actions are limited so as to rule out the nesting capability. In $\text{HO}^{-\mathsf{f}}$, output actions can communicate the parallel composition of two kinds of objects: (i) statically known closed processes (i.e. that do not contain free variables), and (ii) processes received through previously executed input actions. Hence, the context in which the output action resides can only contribute to communication by "appending" pieces of code that admit no inspection, available in the form of a black-box. More formally, the grammar of $\text{HO}^{-\mathsf{f}}$ processes is that in $(*)$, excepting the production for output actions, which is replaced by the following one:

$$\overline{a}\langle x_1 \parallel \cdots \parallel x_k \parallel P\rangle$$

where $k \geq 0$ and P is a closed process. This modification directly restricts forwarding capabilities for output processes, which in turn, leads to a more limited structure of processes along reductions.

The limited style of higher-order communication enforced in $\text{HO}^{-\mathsf{f}}$ is relevant from a pragmatic perspective. In fact, communication in $\text{HO}^{-\mathsf{f}}$ is inspired by those cases in which a process P is communicated in a translated format $[\![P]\!]$, and the translation is not compositional. That is, the cases in which, for any process context C, the translation of $C[P]$ cannot be seen as a function of the translation of P, i.e. there exists no context D such that $[\![C[P]]\!] = D[P]$. This setting can be related to several existing programming scenarios. The simplest example is perhaps mobility of already compiled code, on which it is not possible to apply inverse translations (such as reverse engineering). Other examples include *proof-carrying code* [6] and communication of *obfuscated code*

[1] Termination and convergence are sometimes also referred to as *universal* and *existential* termination, respectively.

[7]. The former features communication of executable code that comes with a certificate: a recipient can only check the certificate and decide whether to execute the code or not. The latter consists in the communication of source code that is made difficult to understand for, e.g., security/copyright reasons, while preserving its functionality.

The main contribution of the paper is the study of the expressiveness of $\text{HO}^{-\mathsf{f}}$ in terms of decidability of termination and convergence. Our main results are:

1. Similarly as HOCORE, $\text{HO}^{-\mathsf{f}}$ is Turing complete. Therefore, despite the limitation of forward capabilities motivated above, the calculus has a significant expressive power. The result is obtained by exhibiting an encoding of Minsky machines.
2. In sharp contrast with HOCORE, termination in $\text{HO}^{-\mathsf{f}}$ is *decidable*. This result is obtained by appealing to the theory of well-structured transition systems [8], following the approach used in [9].

As for (1), it is worth commenting that the encoding is not *faithful* in the sense that, unlike the encoding of Minsky machines in HOCORE, it may introduce computations which do not correspond to the expected behavior of the modeled machine. Such computations are forced to be infinite and thus regarded as non-halting computations which are therefore ignored. Only the finite computations correspond to those of the encoded Minsky machine. This way, we prove that a Minsky machine terminates if and only if its encoding in $\text{HO}^{-\mathsf{f}}$ converges. Consequently, convergence in $\text{HO}^{-\mathsf{f}}$ is *undecidable*.

As for (2), the use of the theory of well-structured transition systems is certainly not a new approach for obtaining expressiveness results. However, to the best of our knowledge, this is the first time it is applied in the higher-order setting. This is significant because the adaptation to the $\text{HO}^{-\mathsf{f}}$ case is far from trivial. Indeed, as we shall discuss, this approach relies on approximating an upper bound on the depth of the (set of) derivatives of a process. By *depth* of a process we mean its maximal nesting of input/output actions. Notice that, even with the limitation on forwarding enforced by $\text{HO}^{-\mathsf{f}}$, because of the "term copying" feature of higher-order calculi, variable instantiation might lead to a potentially larger process. Hence, finding suitable ways of bounding the set of derivatives of a process is rather challenging and needs care.

Structure of the paper. The syntax and semantics of $\text{HO}^{-\mathsf{f}}$ are introduced in Section 2. The encoding of Minsky machines into $\text{HO}^{-\mathsf{f}}$, and the undecidability of convergence are discussed in Section 3. The decidability of termination is addressed in Section 4. Some final remarks, as well as a review of related work, are included in Section 5. We omit most proofs; these can be found in the extended version [10].

2 The Calculus

We now introduce the syntax and semantics of $\text{HO}^{-\mathsf{f}}$. We use a, b, c to range over names, and x, y, z to range over variables; the sets of names and variables are disjoint.

$P, Q ::= \overline{a}\langle x_1 \parallel \cdots \parallel x_k \parallel P\rangle$	(with $k \geq 0$, $\mathsf{fv}(P) = \emptyset$)	output
$\mid\ a(x).\, P$		input prefix
$\mid\ P \parallel Q$		parallel composition
$\mid\ x$		process variable
$\mid\ \mathbf{0}$		nil

An input $a(x).\,P$ binds the free occurrences of x in P. We write $\mathsf{fv}(P)$ and $\mathsf{bv}(P)$ for the set of free and bound variables in P, respectively. A process is *closed* if it does not have free variables. We abbreviate $a(x).\,P$, with $x \notin \mathsf{fv}(P)$, as $a.\,P$, $\overline{a}\langle \mathbf{0}\rangle$ as $\overline{a}$, and $P_1 \parallel \ldots \parallel P_k$ as $\prod_{i=1}^{k} P_i$. Hence, an output action can be written as $\overline{a}\langle \prod_{k\in K} x_k \parallel P\rangle$. We write $\prod_1^n P$ as an abbreviation for the parallel composition of n copies of P. Further, $P\{{}^{Q}/_{x}\}$ denotes the substitution of the free occurrences of x with process Q in P.

Now we describe the Labeled Transition System (LTS), which is defined on closed processes. There are three forms of transitions: τ transitions $P \xrightarrow{\tau} P'$; input transitions $P \xrightarrow{a(x)} P'$, meaning that P can receive at a a process that will replace x in the continuation P'; and output transitions $P \xrightarrow{\overline{a}\langle P'\rangle} P''$ meaning that P emits P' at a, and in doing so it evolves to P''. We use α to indicate a generic label of a transition.

$$\textsc{Inp}\quad a(x).\,P \xrightarrow{a(x)} P \qquad\qquad \textsc{Out}\quad \overline{a}\langle P\rangle \xrightarrow{\overline{a}\langle P\rangle} \mathbf{0}$$

$$\textsc{Act1}\ \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 \parallel P_2 \xrightarrow{\alpha} P_1' \parallel P_2} \qquad \textsc{Tau1}\ \frac{P_1 \xrightarrow{\overline{a}\langle P\rangle} P_1' \qquad P_2 \xrightarrow{a(x)} P_2'}{P_1 \parallel P_2 \xrightarrow{\tau} P_1' \parallel P_2'\{{}^{P}/_{x}\}}$$

(We have omitted ACT2 and TAU2, the symmetric counterparts of the last two rules.)

Remark 1. Since we consider closed processes, in rule ACT1, P_2 has no free variables and no side conditions are necessary. As a consequence, alpha-conversion is not needed.

Definition 1. *The* structural congruence *relation is the smallest congruence generated by the following laws:*

$$P \parallel \mathbf{0} \equiv P,\ P_1 \parallel P_2 \equiv P_2 \parallel P_1,\ P_1 \parallel (P_2 \parallel P_3) \equiv (P_1 \parallel P_2) \parallel P_3.$$

Reductions $P \longrightarrow P'$ are defined as $P \xrightarrow{\tau} P'$.

The *alphabet* of an $\text{HO}^{-\mathsf{f}}$ process is defined as follows:

Definition 2 (Alphabet of a process). *Let P be a* $\text{HO}^{-\mathsf{f}}$ *process. The* alphabet of P*, denoted $\mathcal{A}(P)$, is inductively defined as:*

$$\mathcal{A}(\mathbf{0}) = \emptyset \qquad \mathcal{A}(P \parallel Q) = \mathcal{A}(P) \cup \mathcal{A}(Q) \qquad \mathcal{A}(x) = \{x\}$$

$$\mathcal{A}(a(x).\,P) = \{a, x\} \cup \mathcal{A}(P) \qquad \mathcal{A}(\overline{a}\langle P\rangle) = \{a\} \cup \mathcal{A}(P)$$

Proposition 1. *Let P be a* $\text{HO}^{-\mathsf{f}}$ *process. The set $\mathcal{A}(P)$ is finite. Also, if $P \xrightarrow{\alpha} P'$ then $\mathcal{A}(P') \subseteq \mathcal{A}(P)$.*

Given a process P, its internal runs $P \longrightarrow P_1 \longrightarrow P_2 \longrightarrow \ldots$ are given by sequences of reductions. We denote with $\longrightarrow^*$ the reflexive and transitive closure of $\longrightarrow$; notation $\longrightarrow^j$ is to stand for a sequence of j reductions. We use $P \not\rightarrow$ to denote that there is no P' such that $P \longrightarrow P'$. Following [9] we now define process convergence and process termination. Observe that process termination implies process convergence while the opposite does not hold.

Definition 3. *Let P be a* $\text{HO}^{-\mathsf{f}}$ *process. We say that P* converges *iff there exists P' such that $P \longrightarrow^* P'$ and $P' \not\rightarrow$. We say that P* terminates *iff there exist no $\{P_i\}_{i\in\mathbb{N}}$ such that $P_0 = P$ and $P_j \longrightarrow P_{j+1}$ for any j.*

$$\text{M-INC}\ \frac{i : \texttt{INC}(r_j) \quad m'_j = m_j + 1 \quad m'_{1-j} = m_{1-j}}{(i, m_0, m_1) \longrightarrow_{\text{M}} (i+1, m'_0, m'_1)} \qquad \text{M-JMP}\ \frac{i : \texttt{DECJ}(r_j, s) \quad m_j = 0}{(i, m_0, m_1) \longrightarrow_{\text{M}} (s, m_0, m_1)}$$

$$\text{M-DEC}\ \frac{i : \texttt{DECJ}(r_j, s) \qquad m_j \neq 0 \qquad m'_j = m_j - 1 \qquad m'_{1-j} = m_{1-j}}{(i, m_0, m_1) \longrightarrow_{\text{M}} (i+1, m'_0, m'_1)}$$

Table 1. Reduction of Minsky machines

3 Convergence is Undecidable

In this section we show that HO$^{-\mathsf{f}}$ is powerful enough to model Minsky machines [5], a Turing complete formalism. We present an encoding that is not *faithful*: unlike the encoding of Minsky machines in HOCORE, it may introduce computations which do not correspond to the expected behavior of the modeled machine. Such computations are forced to be infinite and thus regarded as non-halting computations which are therefore ignored. Only finite computations correspond to those of the encoded Minsky machine. More precisely, given a Minsky machine N, its encoding $[\![N]\!]$ has a terminating computation if and only if N terminates. This allows to prove that convergence is undecidable.

We begin by briefly recalling the definition of Minsky machines; we then present the encoding into HO$^{-\mathsf{f}}$ and discuss its correctness.

Minsky machines. A Minsky machine is a Turing complete model composed of a set of sequential, labeled instructions, and two registers. Registers r_j ($j \in \{0, 1\}$) can hold arbitrarily large natural numbers. Instructions $(1 : I_1), \ldots, (n : I_n)$ can be of two kinds: $\texttt{INC}(r_j)$ adds 1 to register r_j and proceeds to the next instruction; $\texttt{DECJ}(r_j, s)$ jumps to instruction s if r_j is zero, otherwise it decreases register r_j by 1 and proceeds to the next instruction. A Minsky machine includes a program counter p indicating the label of the instruction being executed. In its initial state, the machine has both registers set to 0 and the program counter p set to the first instruction. The Minsky machine stops whenever the program counter is set to a non-existent instruction, i.e. $p > n$. A *configuration* of a Minsky machine is a tuple (i, m_0, m_1); it consists of the current program counter and the values of the registers. Formally, the reduction relation over configurations of a Minsky machine, denoted $\longrightarrow_{\text{M}}$, is defined in Table 1.

In the encoding of a Minsky machine into HO$^{-\mathsf{f}}$ we will find it convenient to have a simple form of guarded replication. This construct can be encoded in HO$^{-\mathsf{f}}$ as follows.

Input-guarded replication. We follow the standard encoding of replication in higher-order process calculi, adapting it to input-guarded replication so as to make sure that diverging behaviors are not introduced. As there is no restriction in HO$^{-\mathsf{f}}$, the encoding is not compositional and replications cannot be nested. In [4] the following encoding is shown to preserve termination.

Definition 4. *Assume a fresh name c. The encoding of* input-guarded replication *is as follows:*

$$[\![!a(z).\, P]\!]_{\text{i!}} = a(z).\, (Q_c \parallel P) \parallel \overline{c}\langle a(z).\, (Q_c \parallel P)\rangle$$

REGISTER r_j $\quad [\![r_j = m]\!]_{\mathsf{M}} = \prod_1^m \overline{u_j}$

INSTRUCTIONS $(i : I_i)$

$$
\begin{aligned}
[\![(i : \mathtt{INC}(r_j))]\!]_{\mathsf{M}} &= !p_i.\,(\overline{u_j} \parallel set_j(x).\,\overline{set_j}\langle x \parallel \textsc{Inc}_j\rangle \parallel \overline{p_{i+1}}) \\
[\![(i : \mathtt{DECJ}(r_j, s))]\!]_{\mathsf{M}} &= \quad !p_i.\,\overline{m_i} \\
&\quad \parallel !m_i.\,(\overline{loop} \parallel u_j.\,loop.\,set_j(x).\,\overline{set_j}\langle x \parallel \textsc{Dec}_j\rangle \parallel \overline{p_{i+1}}) \\
&\quad \parallel !m_i.\,set_j(x).\,(x \parallel \overline{set_j}\langle \mathbf{0}\rangle \parallel \overline{p_s}))
\end{aligned}
$$

where

$$\textsc{Inc}_j = \overline{loop} \parallel check_j.\,loop \qquad\qquad \textsc{Dec}_j = \overline{check_j}$$

Table 2. Encoding of Minsky machines

where $Q_c = c(x).\,(x \parallel \overline{c}\langle x\rangle)$, P contains no replications (nested replications are forbidden), and $[\![\cdot]\!]_{\mathsf{i!}}$ is an homomorphism on the other process constructs in $\mathrm{HO}^{-\mathsf{f}}$.

Encoding Minsky machines into $\mathbf{HO}^{-\mathsf{f}}$. The encoding of Minsky machines into $\mathrm{HO}^{-\mathsf{f}}$ is denoted by $[\![\cdot]\!]_{\mathsf{M}}$ and presented in Table 2. We begin by defining the encoding of the configurations of a Minsky machine; we then discuss the encodings of registers and instructions.

Definition 5 (Encoding of Configurations). *Let N be a Minsky machine with registers r_0, r_1 and instructions $(1 : I_1), \ldots, (n : I_n)$. For $j \in \{0, 1\}$, suppose fresh, pairwise different names r_j, $p_1, \ldots, p_n$, set_j, $loop$, $check_j$. Also, let* DIV *be a divergent process (e.g. $\overline{w} \parallel !w.\,\overline{w}$). Given the encodings in Table 2, we have:*

1. *The initial configuration $(1, 0, 0)$ of N is encoded as:*

$$[\![(1, 0, 0)]\!]_{\mathsf{M}} ::= \overline{p_1} \parallel \prod_{i=1}^{n} [\![(i : I_i)]\!]_{\mathsf{M}} \parallel loop.\,\textsc{Div} \parallel \overline{set_0}\langle \mathbf{0}\rangle \parallel \overline{set_1}\langle \mathbf{0}\rangle\,.$$

2. *A configuration (i, m_0, m_1) of N, after k_j increments and l_j decrements of register r_j, is encoded as:*

$$[\![(i, m_0, m_1)]\!]_{\mathsf{M}} = \overline{p_i} \parallel [\![r_0 = m_0]\!]_{\mathsf{M}} \parallel [\![r_1 = m_1]\!]_{\mathsf{M}} \parallel \prod_{i=1}^{n} [\![(i : I_i)]\!]_{\mathsf{M}} \parallel$$
$$loop.\,\textsc{Div} \parallel \overline{set_0}\langle \textsc{Log}_0[k_0, l_0]\rangle \parallel \overline{set_1}\langle \textsc{Log}_1[k_1, l_1]\rangle\,.$$

A register r_j that stores the number m is encoded as the parallel composition of m copies of the unit process $\overline{u_j}$. To implement the test for zero it is necessary to record how many increments and decrements have been performed on the register r_j. This is done by using a special process $\textsc{Log}_j$, which is communicated back and forth on channel set_j. More precisely, every time an increment instruction occurs, a new copy of the process $\overline{u_j}$ is created, and the process $\textsc{Log}_j$ is updated by adding the process $\textsc{Inc}_j$ in parallel. Similarly for decrements: a copy of $\overline{u_j}$ is consumed and the process $\textsc{Dec}_j$ is added to $\textsc{Log}_j$. As a result, after k increments and l decrements on register r_j, we have that $\textsc{Log}_j = \prod_k \textsc{Inc}_j \parallel \prod_l \textsc{Dec}_j$, which we abbreviate as $\textsc{Log}_j[k, l]$.

Each instruction $(i : I_i)$ is a replicated process guarded by p_i, which represents the program counter when $p = i$. Once p_i is consumed, the instruction is active and an interaction with a register occurs. We already described the behavior of increments. Let us

now focus on decrements, the instructions that can introduce divergent —unfaithful— computations. In this case, the process can internally choose either to actually perform a decrement and proceed with the next instruction, or to jump. This can be seen as a guess the process makes on the actual number stored by the register r_j. Therefore, two situations can occur:

1. *The process chooses to decrement* r_j. In this case instruction p_{i+1} is immediately enabled, and the process launches process $\overline{loop}$ and then tries to consume a copy of $\overline{u_j}$. If this operation succeeds (i.e. the content of r_j is greater than 0) then a synchronization with the input on $loop$ that guards the updating of LOG_j (represented as an output on name set_j) takes place. Otherwise, the unit process $\overline{u_j}$ could not be consumed (i.e. the content of r_j is zero and the process made a wrong guess). Process $\overline{loop}$ then synchronizes with the external process $loop$. DIV, thus spawning a divergent computation.
2. *The process chooses to jump to instruction* p_s. In this case instruction p_s is immediately enabled, and it is necessary to check if the actual value stored by r_j is zero. To do so, the process receives the process LOG_j and launches it. If the number of increments is equal to the number of decrements then complementary signals on the name $check_j$ will match each other. In turn, this allows each signal $\overline{loop}$ executed by an INC_j process to be matched by a complementary one. Otherwise, then it is the case that at least one of those $\overline{loop}$ signals remains active (i.e. the content of the register is not zero); a synchronization with the process $loop$. DIV then takes place, and a divergent computation is spawned.

Before executing the instructions, we require both registers in the Minsky machine to be set to zero. This is to guarantee correctness: starting with values different from zero in the registers (without proper initialization of the logs) can lead to inconsistencies. For instance, the test for zero would succeed (i.e. without spawning a divergent computation) even for a register whose value is different from zero.

With a little abuse of notation, we use notation $Q \nrightarrow$ also for configurations of Minsky machines. We now state that the encoding is correct.

Theorem 1. *Let N be a Minsky machine with registers $r_0 = m_0$, $r_1 = m_1$, instructions $(1 : I_1), \ldots, (n : I_n)$ and configuration (i, m_0, m_1). Let P be the process $[\![(i, m_0, m_1)]\!]_{\mathsf{M}}$. Then (i, m_0, m_1) terminates if and only if P converges.*

As a consequence of the results above we have that convergence is undecidable.

Corollary 1. *Convergence is undecidable in* $\text{HO}^{-\mathsf{f}}$.

4 Termination is Decidable

In this section we prove that termination is decidable for $\text{HO}^{-\mathsf{f}}$ processes. As hinted at in the introduction, this is in sharp contrast with the analogous result for HOCORE. The proof appeals to the theory of well-structured transition systems, whose main definitions and results we summarize next.

Well-Structured Transition Systems. The following results and definitions are from [8], unless differently specified. Recall that a *quasi-order* (or, equivalently, preorder) is a reflexive and transitive relation.

Definition 6 (Well-quasi-order). *A* well-quasi-order *(wqo) is a quasi-order $\leq$ over a set X such that, for any infinite sequence $x_0, x_1, x_2 \ldots \in X$, there exist indexes $i < j$ such that $x_i \leq x_j$.*

Note that if $\leq$ is a wqo then any infinite sequence $x_0, x_1, x_2, \ldots$ contains an infinite increasing subsequence $x_{i_0}, x_{i_1}, x_{i_2}, \ldots$ (with $i_0 < i_1 < i_2 < \ldots$). Thus well-quasi-orders exclude the possibility of having infinite strictly decreasing sequences.

We also need a definition for (finitely branching) transition systems. This can be given as follows. Here and in the following $\rightarrow^*$ denotes the reflexive and transitive closure of the relation $\rightarrow$.

Definition 7 (Transition system). *A* transition system *is a structure $TS = (S, \rightarrow)$, where S is a set of states and $\rightarrow \subseteq S \times S$ is a set of transitions. We define $Succ(s)$ as the set $\{s' \in S \mid s \rightarrow s'\}$ of immediate successors of S. We say that TS is finitely branching if, for each $s \in S$, $Succ(s)$ is finite.*

Fact 1 *The LTS for* $\mathrm{HO}^{-\mathsf{f}}$ *given in Section 2 is finitely branching.*

The function $Succ$ will be used also on sets by assuming that in this case they are defined by the point-wise extension of the above definitions. The key tool to decide several properties of computations is the notion of *well-structured transition system*. This is a transition system equipped with a well-quasi-order on states which is (upward) compatible with the transition relation. Here we will use a strong version of compatibility, hence the following definition.

Definition 8 (Well-structured transition system). *A* well-structured transition system with strong compatibility *is a transition system $TS = (S, \rightarrow)$, equipped with a quasi-order $\leq$ on S, such that the two following conditions hold:*

1. *$\leq$ is a well-quasi-order;*
2. *$\leq$ is strongly (upward) compatible with $\rightarrow$, that is, for all $s_1 \leq t_1$ and all transitions $s_1 \rightarrow s_2$, there exists a state t_2 such that $t_1 \rightarrow t_2$ and $s_2 \leq t_2$ holds.*

The following theorem is a special case of Theorem 4.6 in [8] and will be used to obtain our decidability result.

Theorem 2. *Let $TS = (S, \rightarrow, \leq)$ be a finitely branching, well-structured transition system with strong compatibility, decidable $\leq$ and computable $Succ$. Then the existence of an infinite computation starting from a state $s \in S$ is decidable.*

We will also need a result due to Higman [11] which allows to extend a well-quasi-order from a set S to the set of the finite sequences on S. To be more precise, given a set S let use denote by S^* the set of finite sequences built by using elements in S. We can define a quasi-order on S^* as follows.

Definition 9. *Let S be a set and $\leq$ a quasi-order over S. The relation $\leq_*$ over S^* is defined as follows. Let $t, u \in S^*$, with $t = t_1 t_2 \dots t_m$ and $u = u_1 u_2 \dots u_n$. We have that $t \leq_* u$ iff there exists an injection f from $\{1, 2, \dots m\}$ to $\{1, 2, \dots n\}$ such that $t_i \leq u_{f(i)}$ and $i \leq f(i)$ for $i = 1, \dots, m$.*

The relation $\leq_*$ is clearly a quasi-order over S^*. It is also a wqo, since we have the following result.

Lemma 1 ([11]). *Let S be a set and $\leq$ a wqo over S. Then $\leq_*$ is a wqo over S^*.*

Finally we will use also the following proposition, whose proof is immediate.

Proposition 2. *Let S be a finite set. Then the equality is a wqo over S.*

Termination is Decidable in $\mathrm{HO}^{-\mathsf{f}}$. Here we prove that termination is decidable in $\mathrm{HO}^{-\mathsf{f}}$. The crux of the proof consists in finding an upper bound for a process and its derivatives. This is possible in $\mathrm{HO}^{-\mathsf{f}}$ because of the limited structure allowed in output actions. We proceed as follows. We begin by defining a notion of normal form for $\mathrm{HO}^{-\mathsf{f}}$ processes. We then characterize an upper bound for the derivatives of a given process, and we define an ordering over them. This ordering is then shown to be a wqo that is strongly compatible with respect to the LTS of $\mathrm{HO}^{-\mathsf{f}}$ given in Section 2. The decidability result is then obtained by resorting to the results from [8] reported before.

Definition 10 (Normal Form). *Let $P \in \mathrm{HO}^{-\mathsf{f}}$. P is in* normal form *iff*

$$P = \prod_{k=1}^{l} x_k \parallel \prod_{i=1}^{m} a_i(y_i).\, P_i \parallel \prod_{j=1}^{n} \overline{b_j}\langle P'_j \rangle$$

where each P_i and P'_j are in normal form.

Lemma 2. *Every process $P \in \mathrm{HO}^{-\mathsf{f}}$ is structurally congruent to a normal form.*

We now define an ordering over normal forms. Intuitively, a process is larger than another if it has more parallel components.

Definition 11 (Relation $\preceq$). *Let $P, Q \in \mathrm{HO}^{-\mathsf{f}}$. We write $P \preceq Q$ iff there exist $x_1 \dots x_l$, $P_1 \dots P_m$, $P'_1 \dots P'_n$, $Q_1 \dots Q_m$, $Q'_1 \dots Q'_n$, and R such that*

$$P \equiv \textstyle\prod_{k=1}^{l} x_k \parallel \prod_{i=1}^{m} a_i(y_i).\, P_i \parallel \prod_{j=1}^{n} \overline{b_j}\langle P'_j \rangle$$
$$Q \equiv \textstyle\prod_{k=1}^{l} x_k \parallel \prod_{i=1}^{m} a_i(y_i).\, Q_i \parallel \prod_{j=1}^{n} \overline{b_j}\langle Q'_j \rangle \parallel R$$

with $P_i \preceq Q_i$ and $P'_j \preceq Q'_j$, for $i \in [1..m]$ and $j \in [1..n]$.

The normal form of a process can be intuitively represented in a tree-like manner. More precisely, given the process in normal form

$$P = \prod_{k=1}^{l} x_k \parallel \prod_{i=1}^{m} a_i(y_i).\, P_i \parallel \prod_{j=1}^{n} \overline{b_j}\langle P'_j \rangle$$

we shall decree its associated tree to have a root node labeled $x_1, \ldots, x_k$. This root node has $m + n$ children, corresponding to the the trees associated to processes $P_1, \ldots, P_m$ and $P'_1, \ldots, P'_m$; the outgoing edges connecting the root node and the children are labeled $a_1(y_1), \ldots, a_m(y_m)$ and $\overline{b_1}, \ldots, \overline{b_n}$.

This intuition on the tree representation of a process in normal form will be useful now to reason about the structure of $\text{HO}^{-\mathsf{f}}$ terms. We begin by defining the *depth* of a process. Notice that the depth of a process corresponds to the maximum depth of its tree representation.

Definition 12 (Depth). *Let* $P = \prod_{k=1}^{l} x_k \parallel \prod_{i=1}^{m} a_i(y_i).\, P_i \parallel \prod_{j=1}^{n} \overline{b_j}\langle P'_j \rangle$ *be a* $\text{HO}^{-\mathsf{f}}$ *process in normal form. The* depth *of* P *is given by*

$$\mathsf{depth}(P) = \max\{1 + \mathsf{depth}(P_i), 1 + \mathsf{depth}(P'_j) \mid i \in [1..m] \wedge j \in [1..n]\}.$$

Given a natural number n and a process P, the set $\mathcal{P}_{P,n}$ contains all those processes in normal form that (i) can be built using the alphabet of P and (ii) whose depth is at most n.

Definition 13. *Let* n *be a natural number and* $P \in \text{HO}^{-\mathsf{f}}$. *We define the set* $\mathcal{P}_{P,n}$ *as follows:*

$$\begin{aligned}\mathcal{P}_{P,n} = \{Q \mid\ & Q \equiv \textstyle\prod_{k\in K} x_k \parallel \prod_{i\in I} a_i(y_i).\, Q_i \parallel \prod_{j\in J} \overline{b_j}\langle Q'_j \rangle \\ & \wedge\ \mathcal{A}(Q) \subseteq \mathcal{A}(P) \\ & \wedge\ Q_i, Q'_j \in \mathcal{P}_{P,n-1}\ \forall i \in I, j \in J\}\end{aligned}$$

where $\mathcal{P}_{P,0}$ *contains processes that are built out only of variables in* $\mathcal{A}(P)$.

As it will be shown later, the set of all derivatives of P is a subset of $\mathcal{P}_{P,2\cdot\mathsf{depth}(P)}$.

When compared to processes in languages such as Milner's CCS, higher order processes have a more complex structure. This is because, by virtue of reductions, an arbitrary process can take the place of possibly several occurrences of a single variable. As a consequence, the depth of (the syntax tree of) a process cannot be determined (or even approximated) before its execution: it can vary arbitrarily along reductions. Crucially, in $\text{HO}^{-\mathsf{f}}$ it is possible to bound such a depth. Our approach is the following: rather than solely depending on the depth of a process, we define measures on the relative position of variables within a process. Informally speaking, such a position will be determined by the number of prefixes that guard a variable. Notice that since variables are allowed only at the top level of the output objects, the distance of such variables remains invariant during reductions. This allows to obtain a bound on the structure for $\text{HO}^{-\mathsf{f}}$ processes. Finally, it is worth stressing that even if the same notions of normal form, depth, and distance can be defined for HOCORE, a finite upper bound for such a language does not exist. We first define the maximum distance between a variable and its binder.

Definition 14. *Let* $P = \prod_{k\in K} x_k \parallel \prod_{i\in I} a_i(y_i).\, P_i \parallel \prod_{j\in J} \overline{b_j}\langle P'_j \rangle$ *be a* $\text{HO}^{-\mathsf{f}}$ *process in normal form. We define the* maximum distance of P *as:*

$$\begin{aligned}\mathsf{maxDistance}(P) = \max\{&\mathsf{maxDist}_{y_i}(P_i), \\ &\mathsf{maxDistance}(P_i), \mathsf{maxDistance}(P'_j) \mid i \in I, j \in J\}\end{aligned}$$

where

$$\mathsf{maxDist}_x(P) = \begin{cases} 1 & \text{if } P = x, \\ 1 + \mathsf{maxDist}_x(P_z) & \text{if } P = a(z).\, P_z \wedge\ x \neq z, \\ 1 + \mathsf{maxDist}_x(P') & \text{if } P = \overline{a}\langle P' \rangle, \\ \max\{\mathsf{maxDist}_x(R), \mathsf{maxDist}_x(Q)\} & \text{if } P = R \parallel Q, \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 3 (Properties of maxDistance). *Let P be a* $\mathrm{HO}^{-\mathsf{f}}$ *process. It holds that:*

1. $\mathsf{maxDistance}(P) \leq \mathsf{depth}(P)$
2. *For every Q such that $P \xrightarrow{\alpha} Q$,* $\mathsf{maxDistance}(Q) \leq \mathsf{maxDistance}(P)$.

We now define the maximum depth of processes that can be communicated. Notice that the continuations of inputs are considered as they could become communication objects themselves along reductions:

Definition 15. *Let $P = \prod_{k \in K} x_k \parallel \prod_{i \in I} a_i(y_i).\, P_i \parallel \prod_{j \in J} \overline{b_j}\langle P'_j \rangle$ be a* $\mathrm{HO}^{-\mathsf{f}}$ *process in normal form. We define the* maximum depth *of a process that can be communicated (*$\mathsf{maxDepCom}(P)$*) in P as:*

$$\mathsf{maxDepCom}(P) = \max\{\mathsf{maxDepCom}(P_i), \mathsf{depth}(P'_j) \mid i \in I, j \in J\}\ .$$

Lemma 4 (Properties of maxDepCom). *Let P be a* $\mathrm{HO}^{-\mathsf{f}}$ *process. It holds that:*

1. $\mathsf{maxDepCom}(P) \leq \mathsf{depth}(P)$
2. *For every Q such that $P \xrightarrow{\alpha} Q$,* $\mathsf{maxDepCom}(Q) \leq \mathsf{maxDepCom}(P)$.

Notation 1 *We use $P \xrightarrow{\widetilde{\alpha}} P'$ if, for some $n \geq 0$, there exist $\alpha_1, \ldots, \alpha_n$ such that $P \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} P'$.*

Generalizing Lemmata 3 and 4 we obtain:

Corollary 2. *Let P be a* $\mathrm{HO}^{-\mathsf{f}}$ *process. For every Q such that $P \xrightarrow{\widetilde{\alpha}} Q$, it holds that:*

1. $\mathsf{maxDistance}(Q) \leq \mathsf{depth}(P)$
2. $\mathsf{maxDepCom}(Q) \leq \mathsf{depth}(P)$.

We are interested in characterizing the derivatives of a given process P. We shall show that they are over-approximated by means of the set $\mathcal{P}_{P,2\cdot\mathsf{depth}(P)}$. Over such an approximation we will investigate the properties of the relation $\preceq$; such properties will then hold also for the set of derivatives.

Definition 16. *Let $P \in \mathrm{HO}^{-\mathsf{f}}$. Then we define* $\mathsf{Deriv}(P) = \{Q \mid P \longrightarrow^* Q\}$

The following results hold because of the limitations we have imposed on the output actions for $\mathrm{HO}^{-\mathsf{f}}$ processes. Any process that can be communicated in P is in $\mathcal{P}_{P,n-1}$ and its maximum depth is also bounded by $\mathsf{depth}(P)$. The deepest position for a variable is when it is a leaf in the tree associated to the normal form of P. That is, when its depth is exactly $\mathsf{depth}(P)$. Hence the following holds:

Proposition 3. *Let P be a $\mathrm{HO}^{-\mathsf{f}}$ process. Suppose, for some n, that $P \in \mathcal{P}_{P,n}$. For every Q such that $P \xrightarrow{\alpha} Q$, it holds that $Q \in \mathcal{P}_{P,2\cdot n}$.*

The lemma below generalizes Proposition 3 to a sequence of reductions.

Lemma 5. *Let P be a $\mathrm{HO}^{-\mathsf{f}}$ process. Suppose, for some n, that $P \in \mathcal{P}_{P,n}$. For every Q such that $P \xrightarrow{\widetilde{\alpha}} Q$, it holds that $Q \in \mathcal{P}_{P,2\cdot n}$.*

Corollary 3. *Let $P \in \mathrm{HO}^{-\mathsf{f}}$. Then $\mathsf{Deriv}(P) \subseteq \mathcal{P}_{P,2\cdot \mathsf{depth}(P)}$.*

To prove that $\preceq$ is a wqo, we first show that it is a quasi order.

Proposition 4. *The relation $\preceq$ is a quasi-order.*

We are now in place to state that $\preceq$ is a wqo.

Theorem 3 (Well-quasi-order). *Let $P \in \mathrm{HO}^{-\mathsf{f}}$ and $n \geq 0$. The relation $\preceq$ is a well-quasi-order over $\mathcal{P}_{P,n}$.*

Proof. The proof is by induction on n.

(–) Let $n = 0$. Then $\mathcal{P}_{P,0}$ contains processes containing only variables taken from $\mathcal{A}(P)$. The equality on finite sets is a well-quasi-ordering; by Lemma 1 (Higman's Lemma) also $=_*$ is a well quasi-ordering: it corresponds to the ordering $\preceq$ on processes containing only variables.

(–) Let $n > 0$. Take an infinite sequence of processes $s = P_1, P_2, \ldots, P_l, \ldots$ with $P_l \in \mathcal{P}_{P,n}$. We shall show that the thesis holds by means of successive filterings of the normal forms of the processes in s. By Lemma 2 there exist K_l, I_l and J_l such that

$$P_l \equiv \prod_{k \in K_l} x_k \parallel \prod_{i \in I_l} a_i(y_i).\, P_i^l \parallel \prod_{j \in J_l} \overline{b_j}\langle P_j'^l \rangle$$

with P_i^l and $P_j'^l \in \mathcal{P}_{P,n-1}$. Hence each P_l can be seen as composed of 3 finite sequences: (i) $x_1 \ldots x_k$, (ii) $a_1(y_1).\, P_1^l \ldots a_i(y_i).\, P_i^l$, and (iii) $\overline{b_1}\langle P_1'^l \rangle \ldots \overline{b_j}\langle P_j'^l \rangle$. We note that the first sequence is composed of variables from the finite set $\mathcal{A}(P)$ whereas the other two sequences are composed by elements in $\mathcal{A}(P)$ and $\mathcal{P}_{P,n-1}$. Since we have an infinite sequence of $\mathcal{A}(P)^*$, as $\mathcal{A}(P)$ is finite, by Proposition 2 and Lemma 1 we have that $=_*$ is a wqo over $\mathcal{A}(P)^*$. By inductive hypothesis, we have that $\preceq$ is a wqo on $\mathcal{P}_{P,n-1}$, hence by Lemma 1 relation $\preceq_*$ is a wqo on $\mathcal{P}^*_{P,n-1}$. We start filtering out s by making the finite sequences $x_1 \ldots x_k$ increasing with respect to $=_*$; let us call this subsequence t. Then we filter out t, by making the finite sequence $a_1(y_1).\, P_1^l \ldots a_i(y_i).\, P_i^l$ increasing with respect to both $\preceq_*$ and $=_*$. This is done in two steps: first, by considering the relation $=_*$ on the subject of the actions (recalling that $a_i, y_i \in \mathcal{A}(P)$), and then by applying another filtering to the continuation using the inductive hypothesis. For the first step, it is worth remarking that we do not consider symbols of the alphabet but pairs of symbols. Since the set of pairs on a finite set is still finite, we know by Higman's Lemma that $=_*$ is a wqo on the set of sequences of pairs (a_i, y_i). For the sequence of outputs $\overline{b_1}\langle P_1'^l \rangle \ldots \overline{b_j}\langle P_j'^l \rangle$ this is also done in two steps: the subject of the outputs are ordered with respect to $=_*$ and the objects of the output action are ordered with respect to $\preceq_*$ using the inductive hypothesis. At the end of the process we obtain an infinite subsequence of s that is ordered with respect to $\preceq$. □

The last thing to show is that the well-quasi-ordering $\preceq$ is strongly compatible with respect to the (finitely branching) LTS associated to $\text{HO}^{-\mathsf{f}}$. We need the following auxiliary lemma:

Lemma 6. *Let P, P', Q, and Q' be $\text{HO}^{-\mathsf{f}}$ processes in normal form such that $P \preceq P'$ and $Q \preceq Q'$. It holds that $P\{^{Q}/_{x}\} \preceq P'\{^{Q'}/_{x}\}$.*

Theorem 4 (Strong Compatibility). *Let $P, Q, P' \in \text{HO}^{-\mathsf{f}}$. If $P \preceq Q$ and $P \xrightarrow{\alpha} P'$ then there exists Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \preceq Q'$.*

Theorem 5. *Let $P \in \text{HO}^{-\mathsf{f}}$. The transition system $(\mathsf{Deriv}(P), \longrightarrow, \preceq)$ is a finitely branching well-structured transition system with strong compatibility, decidable $\preceq$, and computable $Succ$.*

Proof. The transition system of $\text{HO}^{-\mathsf{f}}$ is finitely branching (Fact 1). The fact that $\preceq$ is a well-quasi-order on $\mathsf{Deriv}(P)$ follows from Corollary 3 and Theorem 3. Strong compatibility follows from Theorem 4. □

We can now state the main result of the section. It follows from Theorems 2 and 5.

Corollary 4. *Let $P \in \text{HO}^{-\mathsf{f}}$. Termination of P is decidable.*

5 Concluding Remarks

We have studied $\text{HO}^{-\mathsf{f}}$, a higher-order process calculi featuring a limited form of higher-order communication. In $\text{HO}^{-\mathsf{f}}$, output actions can only include previously received processes in composition with closed ones. This is reminiscent of programming scenarions with forms of code mobility in which the recipient is not authorized or capable to access/modify the structure of the received code. We have shown that such a weakening of the forward capabilities of higher-order processes has consequences both on the expressiveness of the language and on the decidability of termination.

As for the expressiveness issues, by exhibiting an encoding of Minsky machines into $\text{HO}^{-\mathsf{f}}$, we have shown that convergence is undecidable. Hence, from an *absolute expressiveness* standpoint, $\text{HO}^{-\mathsf{f}}$ is Turing complete. Now, given the analogous result for HOCORE [4], a *relative expressiveness* issue also arises. Indeed, our encoding of Minsky machines into $\text{HO}^{-\mathsf{f}}$ is not faithful, which reveals a difference on the criteria each encoding satisfies. This reminds us of the situation in [12], where faithful and unfaithful encodings of Turing complete formalisms into calculi with interruption and compensation are compared. Using the terminology in [12], we can say that the presented encoding satisfies a *weakly Turing completeness* criterion, as opposed to the (stronger) *Turing completeness* criterion that is satisfied by the encoding of Minsky machines into HOCORE in [4]. The discrepancy on the criteria satisfied by each encoding might be interpreted as an expressiveness gap between $\text{HO}^{-\mathsf{f}}$ and HOCORE; nevertheless, it seems clear that the loss of expressiveness resulting from limiting the forwarding capabilities in HOCORE is much less dramatic than what one would have expected. The communication style of $\text{HO}^{-\mathsf{f}}$ causes a separation result with respect to HOCORE. In

fact, because of the limitation on output actions, it was possible to prove that termination in HO$^{-\mathsf{f}}$ is decidable. This is in sharp contrast with the situation in HOCORE, for which termination is undecidable. In HO$^{-\mathsf{f}}$, it is possible to provide an upper bound on the depth (i.e. the level of nesting of actions) of the (set of) derivatives of a process. In HOCORE such an upper bound does not exist. This was essential for obtaining the decidability result; for this, we appealed to the approach developed in [9], which relies on the theory of well-structured transition systems [8]. As far as we are aware, this approach for expressiveness has not previously been used in the higher-order setting. The decidability of termination is significant, as it might shed light on the development of verification techniques for higher-order processes.

The HO$^{-\mathsf{f}}$ calculus is a sublanguage of HOCORE. As such, HO$^{-\mathsf{f}}$ inherits the many results and properties of HOCORE [4]; most notably, a notion of (strong) bisimilarity which is decidable and coincides with a number of sensible equivalences in the higher-order context. Our results thus complement those in [4] and deepen our understanding of the expressiveness of core higher-order calculi as a whole. Furthermore, by recalling that CCS without restriction is not Turing complete and has decidable convergence, the present results have shaped an interesting expressiveness hierarchy, namely one in which HOCORE is strictly more expressive than HO$^{-\mathsf{f}}$ (because of the discussion above), and in which HO$^{-\mathsf{f}}$ is strictly more expressive than CCS without restriction.

Remarkably, our undecidability result can be used to prove that (weak) barbed bisimilarity is undecidable in the calculus obtained by extending HO$^{-\mathsf{f}}$ with restriction. Consider the encoding of Minsky machines used in Section 3 to prove the undecidability of convergence in HO$^{-\mathsf{f}}$. Consider now the restriction operator $(\nu\widetilde{x})$ used as a binder for the names in the tuple $\widetilde{x}$. Take a Minsky machine N (it is not restrictive to assume that it executes at least one increment instruction) and its encoding P, as defined in Definition 5. Let $\widetilde{x}$ be the tuple of the names used by P, excluding the name w. We have that N terminates if and only if $(\nu\widetilde{x})P$ is (weakly) barbed equivalent to the process $(\nu d)(\overline{d} \mid d \mid d.\,(\overline{w} \mid !w.\,\overline{w}))$.

Related Work. The most closely related work is [4], which was already discussed along the paper. We do not know of other works that study the expressiveness of higher-order calculi by restricting higher-order outputs. The recent work [13] studies finite-control fragments of Homer (a higher-order process calculus with locations). While we have focused on decidability of termination and convergence, the interest in [13] is on the decidability of barbed bisimilarity. One of the approaches explored in [13] is based on a type system that bounds the size of processes in terms of their syntactic components (e.g. number of parallel components, location nesting). Although the restrictions such a type system imposes might be considered as similar in spirit to the limitation on outputs in HO$^{-\mathsf{f}}$ (in particular, location nesting resembles the output nesting HO$^{-\mathsf{f}}$ forbids), the fact that the synchronization discipline in Homer depends heavily on the structure of locations makes it difficult to establish a more detailed comparison.

Also similar in spirit to our work, but in a slightly different context, are some studies on the expressiveness (of fragments) of Ambient calculus [14]. Ambient calculi are related to higher-order calculi in that both allow the communication of objects with complex structure. Some works on the expressiveness of fragments of Ambient calculi are similar to ours. In particular, in [15] it is determined that, for calculi without restric-

tion (as $\text{HO}^{-\text{f}}$ and HOCORE), when the capability of ambient movement is ruled out from the language process termination is decidable for the fragment with replication while it turns out to be undecidable for the fragment with recursion. Hence, in [15] the separation between fragments comes from the source of infinite behavior considered, and not from the structures allowed in output action, as in our case. We find, however, that the connections between Ambient-like and higher-order calculi are rather loose, so a proper comparison is difficult also in this case.

Future Work. Here we have studied an alternative for limiting the output capabilities of the calculus; it would be interesting to understand if suitable limitations on input actions are possible, and whether they have influence on the expressiveness and decidability of higher-order calculi. Another interesting direction would be to compare higher-order and Ambient calculi from the expressiveness point of view.

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. 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
5. Minsky, M.: Computation: Finite and Infinite Machines. Prentice-Hall (1967)
6. Necula, G.C., Lee, P.: Safe, untrusted agents using proof-carrying code. In: Mobile Agents and Security. Volume 1419 of Lecture Notes in Computer Science., Springer (1998) 61–91
7. Collberg, C.S., Thomborson, C.D., Low, D.: Manufacturing cheap, resilient, and stealthy opaque constructs. In: Proc. of POPL'98, ACM Press (1998) 184–196
8. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! Theor. Comput. Sci. **256**(1-2) (2001) 63–92
9. Busi, N., Gabbrielli, M., Zavattaro, G.: On the expressive power of recursion, replication, and iteration in process calculi. Mathematical Structures in Computer Science (2009) To appear. A preliminary version appeared in the Proc. of ICALP'04.
10. Di Giusto, C., Pérez, J.A., Zavattaro, G.: On the Expressiveness of Forwarding in Higher-Order Communication (Extended Version) (2009) Available at `http://www.cs.unibo.it/~perez/hocore`.
11. Higman, G.: Ordering by divisibility in abstract algebras. Proceedings of the London Mathematical Society (3) **2**(7) (1952) 326–336
12. Bravetti, M., Zavattaro, G.: On the expressive power of process interruption and compensation. Mathematical Structures in Computer Science (2009) To appear.
13. Bundgaard, M., Godskesen, J.C., Haagensen, B., Huttel, H.: Decidable fragments of a higher order calculus with locations. In: Proc. of EXPRESS'08. Electronic Notes in Theoretical Computer Science, Elsevier. To appear.
14. Cardelli, L., Gordon, A.D.: Mobile ambients. Theor. Comput. Sci. **240**(1) (2000) 177–213
15. Busi, N., Zavattaro, G.: On the expressive power of movement and restriction in pure mobile ambients. Theor. Comput. Sci. **322**(3) (2004) 477–515