On the Expressiveness of Forwarding in Higher-Order Communication

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

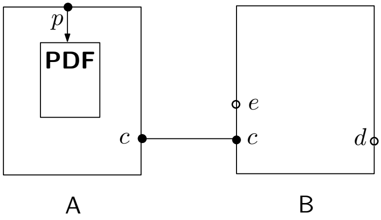
University of Bologna, Italy.

ICTAC'09
Kuala Lumpur, August 2009

Roadmap

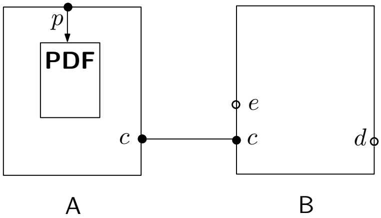
Motivation: Sharing a Resource

Two agents, *A* and *B*, and a resource that A wants to share with B:



Motivation: Sharing a Resource

Two agents, A and B, and a resource that A wants to share with B:

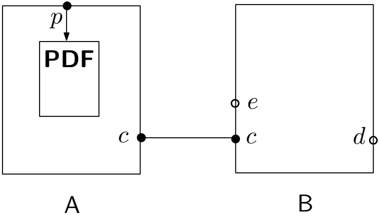


Two approaches:

- First-order (or name-passing) concurrency
- Higher-order (or process-passing) concurrency

Motivation: Sharing a Resource

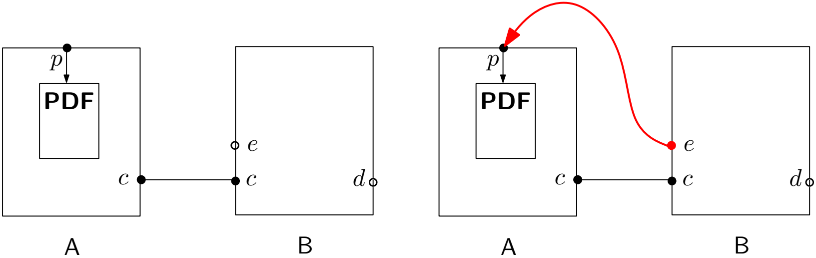
The first-order concurrency approach: send a link to the resource.



(a) Before the interaction(s)

Motivation: Sharing a Resource

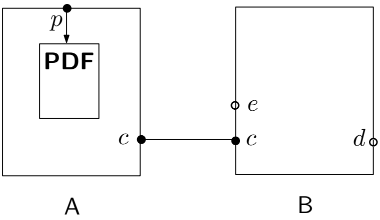
The first-order concurrency approach: send a link to the resource.



(c) Before the interaction(s) (d) After the interaction(s)

Motivation: Sharing a Resource

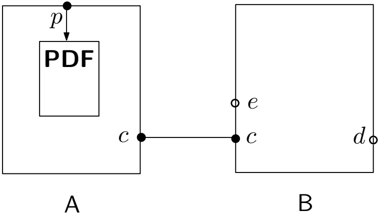
The higher-order concurrency approach: send the resource.



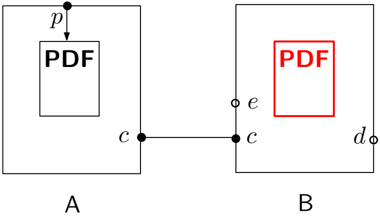
(e) Before the interaction(s)

Motivation: Sharing a Resource

The higher-order concurrency approach: send the resource.



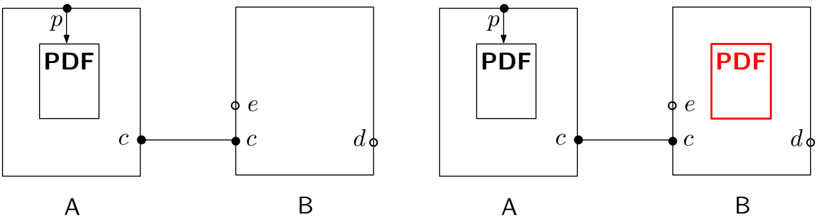
(g) Before the interaction(s)



(h) After the interaction(s)

Motivation: Sharing a Resource

The higher-order concurrency approach: send the resource.

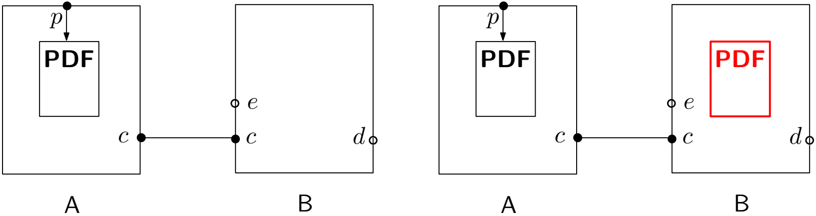


(i) Before the interaction(s)   (j) After the interaction(s)

Upon reception, B can do only two things with the resource:

Motivation: Sharing a Resource

The higher-order concurrency approach: send the resource.



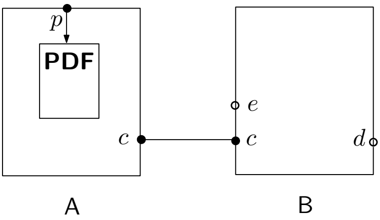
(k) Before the interaction(s) (l) After the interaction(s)

Upon reception, B can do only two things with the resource:

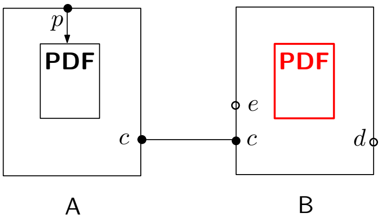
1. Execute it

Motivation: Sharing a Resource

The higher-order concurrency approach: send the resource.



(m) Before the interaction(s)



(n) After the interaction(s)

Upon reception, B can do only two things with the resource:

1. Execute it
2. Forward it

Roadmap

1. Motivation
2. This Talk
3. The HO^{-f} calculus
4. Convergence is Undecidable in HO^{-f}
5. Termination is Decidable in HO^{-f}

This talk, informally

A study of the forwarding capabilities in higher-order communication.

This talk, informally

A study of the forwarding capabilities in higher-order communication.

- A core calculus for higher-order concurrency.
  - Only processes can be communicated.
  - No links can be passed around.
- Our interest: expressive power and decidability properties.

Higher-Order Process Calculi

- Calculi in which processes can be communicated.
- Usual operators: parallel composition, input and output prefixes, restriction. Infinite behavior can be encoded.
- As in the λ-calculus, computation involves term instantiation.

HOCORE: a calculus for higher-order concurrency

P, Q	$::=$	$\overline{a}\langle P \rangle$	output
	$\mid$	$a(x).\, P$	input prefix
	$\mid$	x	process variable
	$\mid$	$P \parallel Q$	parallel composition
	$\mid$	$\mathbf{0}$	nil

HOCORE: a calculus for higher-order concurrency

$$
\begin{array}{rcll}
P,\ Q & ::= & \overline{a}\langle P\rangle & \text{output} \\
 & \mid & a(x).\,P & \text{input prefix} \\
 & \mid & x & \text{process variable} \\
 & \mid & P \parallel Q & \text{parallel composition} \\
 & \mid & \mathbf{0} & \text{nil}
\end{array}
$$

- No name passing is allowed.
- No output prefix: asynchronous calculus.
- No restriction operator

HOCORE: a calculus for higher-order concurrency

$$
\begin{array}{rclll}
P, Q & ::= & \overline{a}\langle P \rangle & & \text{output} \\
& \mid & a(x).\, P & & \text{input prefix} \\
& \mid & x & & \text{process variable} \\
& \mid & P \parallel Q & & \text{parallel composition} \\
& \mid & \mathbf{0} & & \text{nil}
\end{array}
$$

- No name passing is allowed.
- No output prefix: asynchronous calculus.
- No restriction operator
  - Every communication is public. Behavior is exposed.
  - Dynamic creation of channels is impossible.

Some Results for HOCORE[1]

HOCORE was shown to be Turing complete.
Moreover, properties such as

- Termination, i.e. non-existence of divergent computations
- Convergence, i.e. existence of a terminating computation

[1] I. Lanese, J. A. Pérez, D. Sangiorgi, and A. Schmitt. *On the Expressiveness and Decidability of Higher-Order Process Calculi*. LICS'08.

Some Results for HOCORE[1]

HOCORE was shown to be Turing complete.
Moreover, properties such as

- Termination, i.e. non-existence of divergent computations
- Convergence, i.e. existence of a terminating computation

are undecidable in HOCORE.

[1] I. Lanese, J. A. Pérez, D. Sangiorgi, and A. Schmitt. *On the Expressiveness and Decidability of Higher-Order Process Calculi*. LICS'08.

What's the origin of the expressivity of HOCORE?

Arbitrary Forwarding

Emitting a received process in an arbitrary context

What's the origin of the expressivity of HOCORE?

Arbitrary Forwarding

Emitting a received process in an arbitrary context

- Take a forwarder process $F = a(x).\, \overline{b}\langle P_x \rangle$

What's the origin of the expressivity of HOCORE?

Arbitrary Forwarding

Emitting a received process in an arbitrary context

- Take a forwarder process $F = a(x).\, \overline{b}\langle P_x \rangle$
- The structure of P_x can be very complex.

What's the origin of the expressivity of HOCORE?

Arbitrary Forwarding

Emitting a received process in an arbitrary context

- Take a forwarder process $F = a(x).\, \overline{b}\langle P_x \rangle$
- The structure of P_x can be very complex.
- Arbitrary nested outputs, e.g. $P_x = \overline{b_1}\langle \overline{b_2}\langle \ldots \overline{b_n}\langle x \rangle\rangle\rangle$.

What's the origin of the expressivity of HOCORE?

Arbitrary Forwarding

Emitting a received process in an arbitrary context

- Take a forwarder process $F = a(x).\,\overline{b}\langle P_x \rangle$
- The structure of P_x can be very complex.
- Arbitrary nested outputs, e.g. $P_x = \overline{b_1}\langle \overline{b_2}\langle \ldots \overline{b_n}\langle x \rangle\rangle\rangle$.

Nested outputs are essential to show Turing completeness for HOCORE (they allow to define counters and test for zero).

Towards Limited Forwarding

Forwarding can be limited by restricting the shape of output objects.

Consider output objects which can only be the composition of:

1. Statically known closed processes
2. Processes received in previous input actions

Towards Limited Forwarding

Forwarding can be limited by restricting the shape of output objects.

Consider output objects which can only be the composition of:

1. Statically known closed processes
2. Processes received in previous input actions

For instance, given a closed process R:

- $P = \overline{a}\langle S \rangle \parallel a(x).\,\overline{b}\langle x \parallel R \rangle$ is a valid process

Towards Limited Forwarding

Forwarding can be limited by restricting the shape of output objects.

Consider output objects which can only be the composition of:

1. Statically known closed processes
2. Processes received in previous input actions

For instance, given a closed process R:

- $P = \overline{a}\langle S \rangle \parallel a(x).\, \overline{b}\langle x \parallel R \rangle$ is a valid process
- whereas $Q = \overline{a}\langle S \rangle \parallel a(x).\, \overline{b}\langle \overline{c}\langle x \parallel R \rangle \rangle$ is not.

Limited Forwarding is Still Interesting

It reminds us of scenarios in which outputs can only "append" pieces of code, available as "black-boxes" that admit no inspection.

Limited Forwarding is Still Interesting

It reminds us of scenarios in which outputs can only “append” pieces of code, available as “black-boxes” that admit no inspection.

Examples

- Communication of compiled code
- Distribution of obfuscated (protected) code
- Proof-carrying code.

This talk, less informally

What is the impact of limiting forwarding in HOCORE?

This talk, less informally

What is the impact of limiting forwarding in HOCORE?

- Do limited output actions affect absolute expressiveness?
  If so, to what extent?
- Do they have influence on the decidability of termination and convergence?

This talk, less informally

What is the impact of limiting forwarding in HOCORE?

- Do limited output actions affect absolute expressiveness? If so, to what extent?
- Do they have influence on the decidability of termination and convergence?

Approach

We study $\mathrm{HO}^{-\mathrm{f}}$: the subcalculus of HOCORE with limited forwarding.

Main Results

1. In contrast to HOCORE, termination in $\mathrm{HO}^{-\mathrm{f}}$ is decidable

Main Results

1. In contrast to HOCORE, termination in $\text{HO}^{-\text{f}}$ is decidable
2. Similarly as HOCORE, convergence in $\text{HO}^{-\text{f}}$ is undecidable

Roadmap

The $\textsc{Ho}^{-f}$ calculus

Syntax

$$
\begin{array}{rcll}
P, Q & ::= & \overline{a}\langle x_1 \parallel \cdots \parallel x_k \parallel P \rangle & \text{(with } k \geq 0,\ \text{fv}(P) = \emptyset) \\
& \mid & a(x).\,P & \\
& \mid & P \parallel Q & \\
& \mid & x & \\
& \mid & \mathbf{0} &
\end{array}
$$

The $\textsc{Ho}^{-f}$ calculus

Semantics

A (finitely-branching) labeled transition system on *closed* processes:

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

$$\textsc{Act1}\quad \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 \parallel P_2 \xrightarrow{\alpha} P_1' \parallel P_2}$$

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

Notice: In rule $\textsc{Act1}$, P_2 has no free variables and no side conditions are necessary. Hence, alpha-conversion is not needed.

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

Convergence and Termination

We denote with $\longrightarrow^*$ the reflexive and transitive closure of $\longrightarrow$.
We use $P \not\longrightarrow$ to denote that there is no P' such that $P \longrightarrow P'$

Definition

Let P be a Ho^{-f} process.

- P converges iff there exists a P' such that $P \longrightarrow^* P'$ and $P' \not\longrightarrow$.
- 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.

Note: Termination implies convergence, but the opposite doesn't hold.

Roadmap

Convergence is Undecidable in $\textsc{Ho}^{-f}$

We prove undecidability by encoding Minsky machines into $\textsc{Ho}^{-f}$.

Convergence is Undecidable in Ho^{-f}

We prove undecidability by encoding Minsky machines into Ho^{-f}.

Two-counter Minsky machines

Turing complete model with n labeled instructions and two registers.

- Registers r_j ($j \in \{0, 1\}$) can hold arbitrarily large natural numbers.
- Instructions can be of two kinds:

Instruction	$r_j == 0$	$r_j > 0$
$\texttt{INC}(r_j)$	$r_j = r_j + 1$	$r_j = r_j + 1$
$\texttt{DECJ}(r_j, k)$	jump to k	$r_j = r_j - 1$

- A program counter indicates the instruction being executed.

Encoding Minsky machines into Ho^{-f}

Limited output actions make it difficult to test for zero precisely. The encoding is *not faithful*:

- It may introduce divergent computations which do not correspond to the behavior of the modeled machine.
- However, such computations are infinite and regarded as non-halting computations which are ignored.
- Only finite computations correspond to those of the encoded Minsky machine.

Encoding Minsky machines into Ho^{-f}

Limited output actions make it difficult to test for zero precisely. The encoding is *not faithful*:

- It may introduce divergent computations which do not correspond to the behavior of the modeled machine.
- However, such computations are infinite and regarded as non-halting computations which are ignored.
- Only finite computations correspond to those of the encoded Minsky machine.

Given a Minsky machine N, its encoding $[\![N]\!]$ converges iff N terminates. This allows to prove that convergence is undecidable.

Encoding Minsky machines into Ho^{-f}

Registers, Instructions, Increments.

- A register r_j storing number m: the parallel composition of m copies of the "unit process" $\overline{u_j}$.
  Each register keeps a log of the operations performed on it.
- Each instruction is a replicated process guarded by p_i, representing the program counter when it contains instruction i.
- An increment of r_j creates a new copy of $\overline{u_j}$, and updates the log of r_j.

Encoding Minsky machines into Ho^{-f}

A decrement and jump makes a "guess" on the value of the register.

Encoding Minsky machines into Ho^{-f}

A decrement and jump makes a “guess” on the value of the register. Wrong guesses lead to divergent behavior.

Encoding Minsky machines into Ho^{-f}

A decrement and jump makes a "guess" on the value of the register.
Wrong guesses lead to divergent behavior.
The encoding either

- Performs the decrement and proceeds with the next instruction
  The decrement tries to consume a copy of $\overline{u_j}$.
  If this succeeds, then the log is updated.
  Otherwise, a divergent computation is spawned.

Encoding Minsky machines into HO^{-f}

A decrement and jump makes a "guess" on the value of the register.
Wrong guesses lead to divergent behavior.
The encoding either

- Performs the decrement and proceeds with the next instruction
  The decrement tries to consume a copy of $\overline{u_j}$.
  If this succeeds, then the log is updated.
  Otherwise, a divergent computation is spawned.

OR

Encoding Minsky machines into Ho^{-f}

A decrement and jump makes a "guess" on the value of the register.
Wrong guesses lead to divergent behavior.
The encoding either

- Performs the decrement and proceeds with the next instruction
  The decrement tries to consume a copy of $\overline{u_j}$.
  If this succeeds, then the log is updated.
  Otherwise, a divergent computation is spawned.

OR

- Jumps
  Exploiting the log of the register, a test for zero is performed.
  If the test fails then a divergent computation is spawned.

Correctness of the Encoding

$[\![\cdot]\!]_M$ denotes the encoding of Minsky machines into Ho^{-f}.

Theorem

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). Then (i, m_0, m_1) terminates iff process $[\![(i, m_0, m_1)]\!]_M$ converges.

Correctness of the Encoding

$[\![\cdot]\!]_M$ denotes the encoding of Minsky machines into Ho^{-f}.

Theorem

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). Then (i, m_0, m_1) terminates iff process $[\![(i, m_0, m_1)]\!]_M$ converges.

Corollary

Convergence is undecidable in Ho^{-f}

Roadmap

Well-structured transition systems

We prove decidability of termination by exploiting the theory of well-structured transition systems [Finkel and Schnoebelen, 2001].

Well-structured transition systems

We prove decidability of termination by exploiting the theory of well-structured transition systems [Finkel and Schnoebelen, 2001].

Intuition: A transition system enriched with an ordering relation over the set of states.

Well-structured transition systems

We prove decidability of termination by exploiting the theory of well-structured transition systems [Finkel and Schnoebelen, 2001].

Intuition: A transition system enriched with an ordering relation over the set of states.

Definition (Well-structured transition system)

A well-structured transition system with strong compatibility is a transition system $TS = (S, \rightarrow, \leq)$ such that:

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

Well-structured transition systems

Theorem (Finkel and Schnoebelen, 2001)

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

Termination is Decidable in HO^{-f}

The proof scheme can be summarized in the following steps:

1. Define a normal form for HO^{-f} processes

Termination is Decidable in Ho^{-f}

The proof scheme can be summarized in the following steps:

1. Define a normal form for Ho^{-f} processes
2. Characterize an upper bound for the derivatives of an Ho^{-f} process in normal form, and define an ordering $\preceq$ over them

Termination is Decidable in $\text{Ho}^{-\text{f}}$

The proof scheme can be summarized in the following steps:

1. Define a normal form for $\text{Ho}^{-\text{f}}$ processes
2. Characterize an upper bound for the derivatives of an $\text{Ho}^{-\text{f}}$ process in normal form, and define an ordering $\preceq$ over them
3. Show that $\preceq$ is a wqo strongly compatible wrt the LTS of $\text{Ho}^{-\text{f}}$

Step 1: A normal form for $\textsc{Ho}^{-f}$ processes

Definition (Normal Form)

Let $P \in \textsc{Ho}^{-f}$. P is in *normal form* iff

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

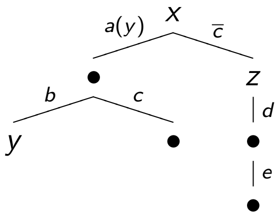
where each P_i and P'_j are in normal form.

Step 1: A normal form for $\mathrm{Ho}^{-\mathrm{f}}$ processes

Normal forms have a tree-like representation.
Depth of a process: the maximum depth of its tree representation.

Example (A process and its tree representation)

$$P = x \parallel a(y).\,(b.\,y \parallel c) \parallel \overline{c}\langle z \parallel d.\,e\rangle$$



Step 2: An upper bound for Ho^{-f} processes

In ordinary higher-order process calculi (including HOCORE):

- After a reduction, an arbitrary process can take the place of possibly several occurrences of a single variable.
- The depth of a process cannot be determined before its execution: It can vary arbitrarily along reductions.

Step 2: An upper bound for Ho^{-f} processes

In ordinary higher-order process calculi (including HOCORE):

- After a reduction, an arbitrary process can take the place of possibly several occurrences of a single variable.
- The depth of a process cannot be determined before its execution: It can vary arbitrarily along reductions.

In Ho^{-f} *it is possible* to bound the depth of a process!

Step 2: An upper bound for $\textsc{Ho}^{-f}$ processes

In ordinary higher-order process calculi (including HOCORE):

- After a reduction, an arbitrary process can take the place of possibly several occurrences of a single variable.
- The depth of a process cannot be determined before its execution: It can vary arbitrarily along reductions.

In $\textsc{Ho}^{-f}$ *it is possible* to bound the depth of a process!

- Idea: to consider the relative position of variables within a process.
- Variables only occur at the top level of the output objects. Hence, their relative position remains invariant along reductions.

Step 2: An ordering over $\textsc{Ho}^{-f}$ processes

Intuition: A process is larger than another if it has more parallel components.

Step 2: An ordering over Ho^{-f} processes

Intuition: A process is larger than another if it has more parallel components.

Definition (Relation $\preceq$)

Let $P, Q \in \text{Ho}^{-f}$. We write $P \preceq Q$ iff there exist $x_1 \ldots x_l$, $P_1 \ldots P_m$, $P'_1 \ldots P'_n$, $Q_1 \ldots Q_m$, $Q'_1 \ldots Q'_n$, and R such that

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

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

Step 3: The ordering $\preceq$ is a wqo

Let n and Q be a natural number and an Ho^{-f} process, resp.
The "bounded set" $\mathcal{P}_{Q,n}$ contains all those processes that

- can be built using the alphabet of Q
- have trees whose depth is at most n

Step 3: The ordering $\preceq$ is a wqo

Let n and Q be a natural number and an $\textsc{Ho}^{-f}$ process, resp.
The "bounded set" $\mathcal{P}_{Q,n}$ contains all those processes that

- can be built using the alphabet of Q
- have trees whose depth is at most n

Theorem (Relation $\preceq$ is a wqo)

Let $P \in \textsc{Ho}^{-f}$ and $n \geq 0$. The relation $\preceq$ is a wqo over $\mathcal{P}_{P,n}$.

Step 3: The ordering $\preceq$ is a wqo

Let n and Q be a natural number and an $\textsc{Ho}^{-f}$ process, resp.
The "bounded set" $\mathcal{P}_{Q,n}$ contains all those processes that

- can be built using the alphabet of Q
- have trees whose depth is at most n

Theorem (Relation $\preceq$ is a wqo)

Let $P \in \textsc{Ho}^{-f}$ and $n \geq 0$. The relation $\preceq$ is a wqo over $\mathcal{P}_{P,n}$.

Theorem (Strong Compatibility)

Let $P, Q, P' \in \textsc{Ho}^{-f}$.
If $P \preceq Q$ and $P \xrightarrow{\alpha} P'$ then $\exists Q'$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \preceq Q'$.

Concluding the proof

Below we use Deriv(P) to denote the set of derivatives of process P.

Theorem

Let $P \in \textsc{Ho}^{-\mathrm{f}}$. The transition system $(\mathrm{Deriv}(P), \longrightarrow, \preceq)$ is a finitely branching, well-structured transition system with strong compatibility.

Concluding the proof

Below we use Deriv(P) to denote the set of derivatives of process P.

Theorem

Let $P \in \text{Ho}^{-\text{f}}$. The transition system $(\text{Deriv}(P), \longrightarrow, \preceq)$ is a finitely branching, well-structured transition system with strong compatibility.

Corollary

Termination is decidable in $\text{Ho}^{-\text{f}}$.

Concluding Remarks

Weakening the forwarding capabilities of higher-order communication has consequences on

- the decidability of termination
- the expressiveness of the language

Concluding Remarks

- In contrast with HOCORE, in HO^{-f} termination is decidable. The limited communication style of HO^{-f} thus causes a separation result with respect to HOCORE.

Concluding Remarks

- In contrast with HOCORE, in HO^{-f} termination is decidable.
  The limited communication style of HO^{-f} thus causes a separation result with respect to HOCORE.
- Similarly as HOCORE, HO^{-f} is Turing complete and its convergence problem is undecidable.
  The calculus retains a significant expressive power despite of the limited forwarding capabilities.
  The encoding into Minsky machines is not faithful, though.

Thanks!

Any Questions?

Alphabet of an $\textsc{Ho}^{-f}$ process

Definition (Alphabet of a process)

Let P be an $\textsc{Ho}^{-f}$ 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

Let P be an $\textsc{Ho}^{-f}$ process. The set $\mathcal{A}(P)$ is finite.

Proposition

Let P and P' be $\textsc{Ho}^{-f}$ processes. If $P \xrightarrow{\alpha} P'$ then $\mathcal{A}(P') \subseteq \mathcal{A}(P)$.

Expressivity of Ho^{-f}

Input-guarded replication

Divergence-free adaptation of the usual encoding of replication:

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

where

- $Q_c = c(x).\, (x \parallel \overline{c}\langle x \rangle)$
- P contains no replications (nested replications are forbidden)
- $[\![\cdot]\!]_{i!}$ is an homomorphism for the other operators.

Encoding Minsky machines into $\textsc{Ho}^{-f}$

$\textsc{Register}\ r_j \qquad [\![r_j = m]\!]_{\mathsf{M}} = \prod_1^m \overline{u_j}$

$\textsc{Instructions}\ (i : I_i)$

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

where

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

Well-structured transition systems

A quasi-order is a reflexive and transitive relation.

Definition (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 $x_i \leq x_j$.

Definition (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.

Step 2: An upper bound for $\mathrm{HO}^{-\mathrm{f}}$ processes

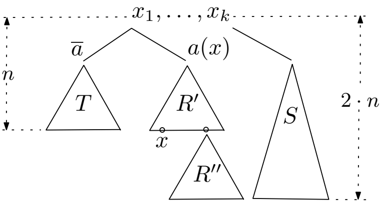
Invariance along reductions of the depth of a process, graphically.

Take a process $P = x_1 \parallel \cdots \parallel x_k \parallel \overline{a}\langle T \rangle \parallel a(x).\, R' \parallel S$.
It reduces to $Q = x_1 \parallel \cdots \parallel x_k \parallel R'\{^T/_x\} \parallel S$.

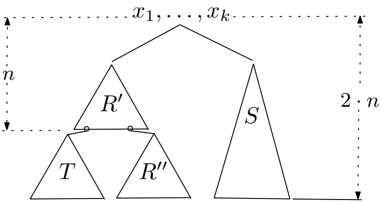
Step 2: An upper bound for $\textsc{Ho}^{-f}$ processes

Invariance along reductions of the depth of a process, graphically.

Take a process $P = x_1 \parallel \cdots \parallel x_k \parallel \overline{a}\langle T \rangle \parallel a(x).\, R' \parallel S$.
It reduces to $Q = x_1 \parallel \cdots \parallel x_k \parallel R'\{{}^{T}\!/x\} \parallel S$.



(q) Tree representation of P



(r) Tree representation of Q