Termination of Mobile Processes

R. Demangeon[1]
(joint work with D. Hirschkoff[1] & D. Sangiorgi[2])

Institute of Cybernetics, Tallinn, August 19th 2010

[1] ENS Lyon
[2] University of Bologna

Termination

Sequential Case

- A program terminates is every execution is finite.
- Useful property: in itself, prerequisite for soundness, lock-freedom.
- Termination is well-studied:
  - for functional programs (type systems, realisability),
  - for imperative programs (abstract interpretation, loop analyses),
  - for rewriting systems (rewriting orderings, polynomial interpretations).

Concurrent case

- More challenging: topology evolving at run-time (creation of new services, sharing of informations).
- Existing results for the shared memory setting: `Terminator` by Cook and al.

Starting point

We focus on the message passing setting.
Two different approaches to ensure termination.

Using weights

- [DengSangiorgi06]
- Assigning levels to service calls.
- Defining a notion of weight for a system.
- Ensuring that the weight of the system decreases at each transition.

Using logical relations

- [Sangiorgi06] and [YoshidaBergerHonda04].
- Considering only functional (immutable, replicated) services.
- Assigning types to these functional services.
- Using logical relations to ensure termination.

Contributions of my thesis

Details

- Complexity of type inference of weight-based type systems for termination in the π-calculus (TGC'07).
- Increasing the expressiveness of the weight-based type systems, developing an hybrid analysis (IFIP/TCS'08).
- Termination of the higher-order concurrent systems (FSEN'09).
- Termination in impure languages (CONCUR'10).
- Implicit complexity (work in progress with Ugo Dal Lago – Bologna)

Syntax

$P ::= \mathbf{0} \mid (P \mid P) \mid (\nu a)\ P \mid \overline{a}\langle v\rangle.P \mid a(x).P.$

- Names (or channels) a, b, c, x, y
- $\mathbf{0}$ is the inactive process.
- $a(x).P$: waits for channel x on channel a, then performs P (x bound in P).
- $\overline{a}\langle v\rangle.P$: outputs channel v on channel a, then perform P.
- $P_1 \mid P_2$: parallel composition.
- $(\nu a)\ P$: restriction of a in P.
- Structural congruence $\equiv$: $\mid$ is symmetric, associative and has $\mathbf{0}$ for neutral element + scope extrusion: $(\nu a)\ (P \mid Q) \equiv ((\nu a)\ P) \mid Q$ if a is not free in Q.

Semantics

$$\frac{}{a(x).P_1 \mid \overline{a}\langle v\rangle.P_2 \rightarrow P_1\{v/x\} \mid P_2} \qquad \frac{P_1 \rightarrow P_1'}{(P_1 \mid P_2) \rightarrow (P_1' \mid P_2)}$$

$$\frac{P \equiv Q \qquad Q \rightarrow Q' \qquad Q' \equiv P'}{P \rightarrow P'}$$

Example

- $P_1 = \overline{a}\langle c\rangle.\mathbf{0} \mid a(x).\overline{b}\langle x\rangle.\mathbf{0} \mid a(y).y(z).\mathbf{0} \mid \overline{c}\langle v\rangle.\mathbf{0}$
- either $\rightarrow a(x).\overline{b}\langle x\rangle.\mathbf{0} \mid c(z).\mathbf{0} \mid \overline{c}\langle v\rangle.\mathbf{0} \rightarrow a(x).\overline{b}\langle x\rangle.\mathbf{0}$
- or $\rightarrow \overline{b}\langle c\rangle.\mathbf{0} \mid a(y).y(z).\mathbf{0} \mid \overline{c}\langle v\rangle.\mathbf{0}$

Two prefixes consumed at each reduction: terminating calculus.

$P ::= \mathbf{0} \mid (P \mid P) \mid (\nu a)\ P \mid \overline{a}\langle v\rangle.P \mid a(x).P \mid !a(x).P.$

Replicated inputs

- $!a(x).P$: replicated input on a, persistent.
- Associated semantics:

$$\frac{}{!a(x).P_1 \mid \overline{a}\langle v\rangle.P_2 \rightarrow !a(x).P_1 \mid P_1\{v/x\} \mid P_2}$$

- Source of divergence.

$P ::= \mathbf{0} \mid (P \mid P) \mid (\nu a)\ P \mid \overline{a}\langle v\rangle.P \mid a(x).P \mid !a(x).P.$

Replicated inputs

- $!a(x).P$: replicated input on a, persistent.
- Associated semantics:

$$\frac{}{!a(x).P_1 \mid \overline{a}\langle v\rangle.P_2 \rightarrow !a(x).P_1 \mid P_1\{v/x\} \mid P_2}$$

- Source of divergence.

Removing trailing occurrences of $\mathbf{0}$

Notation for CCS-like channels: $!a.\overline{b}$.

Divergence

Definition

P *diverges* if there exists an infinite reduction sequence starting from P.

Diverging processes

- $D_1 = !a.\overline{a} \mid \overline{a}$
- $D_2 = !a.\overline{b} \mid !b.\overline{a} \mid \overline{a}$
- $D_3 = c(x).!a.\overline{x} \mid \overline{a} \mid \overline{c}\langle a \rangle$

Divergence

Definition

P *diverges* if there exists an infinite reduction sequence starting from P.

Diverging processes

- $D_1 = !a.\overline{a} \mid \overline{a} \rightarrow D_1$
- $D_2 = !a.\overline{b} \mid !b.\overline{a} \mid \overline{a}$
- $D_3 = c(x).!a.\overline{x} \mid \overline{a} \mid \overline{c}\langle a \rangle$

Divergence

Definition

P *diverges* if there exists an infinite reduction sequence starting from P.

Diverging processes

- $D_1 = !a.\overline{a} \mid \overline{a} \rightarrow D_1$
- $D_2 = !a.\overline{b} \mid !b.\overline{a} \mid \overline{a} \rightarrow\rightarrow D_2$
- $D_3 = c(x).!a.\overline{x} \mid \overline{a} \mid \overline{c}\langle a \rangle$

Divergence

Definition

P *diverges* if there exists an infinite reduction sequence starting from P.

Diverging processes

- $D_1 = !a.\overline{a} \mid \overline{a} \rightarrow D_1$
- $D_2 = !a.\overline{b} \mid !b.\overline{a} \mid \overline{a} \rightarrow\rightarrow D_2$
- $D_3 = c(x).!a.\overline{x} \mid \overline{a} \mid \overline{c}\langle a \rangle \rightarrow D_1$

A rewriting-based type system

Principles

- Associate a *level* to each name through types ($T ::= \mathtt{b} \mid \sharp^k\, T$).
- Weight of a process: maximum level of an available output.
- Judgements $\vdash_\Gamma P : n$

A rewriting-based type system

Principles

- Associate a *level* to each name through types ($T ::= \mathtt{b} \mid \sharp^k\, T$).
- Weight of a process: maximum level of an available output.
- Judgements $\vdash_\Gamma P : n$
- Control replications: $k > n$
- Soundness: decreasing of the multiset of the available outputs.

Typing replication

$$\frac{\vdash_\Gamma P : n \qquad \Gamma(a) = \sharp^k\, T \qquad \Gamma(x) = T \qquad k > n}{\vdash_\Gamma !a(x).P : 0}$$

Ruling out diverging examples

- $D_1 = !a^n.\overline{a}^n \mid \overline{a}^n$
- $D_2 = !a^n.\overline{b}^m \mid !b^m.\overline{a}^n \mid \overline{a}^n$.
- $D_3 = c^k(x).!a^n.\overline{x}^m \mid \overline{a}^n \mid \overline{c}^k\langle a\rangle.$

.

Ruling out diverging examples

- $D_1 = !a^n.\overline{a}^n \mid \overline{a}^n$ not typable: $n > n$
- $D_2 = !a^n.\overline{b}^m \mid !b^m.\overline{a}^n \mid \overline{a}^n$
- $D_3 = c^k(x).!a^n.\overline{x}^m \mid \overline{a}^n \mid \overline{c}^k\langle a \rangle$.

Ruling out diverging examples

- $D_1 = !a^n.\overline{a}^n \mid \overline{a}^n$ not typable: $n > n$
- $D_2 = !a^n.\overline{b}^m \mid !b^m.\overline{a}^n \mid \overline{a}^n$ not typable: $n > m > n$.
- $D_3 = c^k(x).!a^n.\overline{x}^m \mid \overline{a}^n \mid \overline{c}^k\langle a\rangle$.

Ruling out diverging examples

- $D_1 = !a^n.\overline{a}^n \mid \overline{a}^n$ not typable: $n > n$
- $D_2 = !a^n.\overline{b}^m \mid !b^m.\overline{a}^n \mid \overline{a}^n$ not typable: $n > m > n$.
- $D_3 = c^k(x).!a^n.\overline{x}^m \mid \overline{a}^n \mid \overline{c}^k\langle a \rangle$.
  $c : \sharp^k\ (\sharp^n\ T)$ implying $n = m$ and $n > m$.

Ruling out diverging examples

- $D_1 = !a^n.\overline{a}^n \mid \overline{a}^n$ not typable: $n > n$
- $D_2 = !a^n.\overline{b}^m \mid !b^m.\overline{a}^n \mid \overline{a}^n$ not typable: $n > m > n$.
- $D_3 = c^k(x).!a^n.\overline{x}^m \mid \overline{a}^n \mid \overline{c}^k\langle a \rangle$.
  $c : \sharp^k (\sharp^n\ T)$ implying $n = m$ and $n > m$.

Decreasing of the weight

- $T_1 = !a\ .(\overline{b}\ \mid \overline{b}\ \mid \overline{c}\ ) \mid !b\ .(\overline{c}\ \mid \overline{c}\ )$
- $(T_1 \mid \overline{a} \mid \overline{b}) \rightarrow (T_1 \mid \overline{b} \mid \overline{b} \mid \overline{b} \mid \overline{c}) \rightarrow\rightarrow\rightarrow\not\rightarrow$.

Ruling out diverging examples

- $D_1 = !a^n.\overline{a}^n \mid \overline{a}^n$ not typable: $n > n$
- $D_2 = !a^n.\overline{b}^m \mid !b^m.\overline{a}^n \mid \overline{a}^n$ not typable: $n > m > n$.
- $D_3 = c^k(x).!a^n.\overline{x}^m \mid \overline{a}^n \mid \overline{c}^k\langle a \rangle$.
  $c : \sharp^k\ (\sharp^n\ T)$ implying $n = m$ and $n > m$.

Decreasing of the weight

- $T_1 = !a^3.(\overline{b}^2 \mid \overline{b}^2 \mid \overline{c}^1) \mid !b^2.(\overline{c}^1 \mid \overline{c}^1)$
- $(T_1 \mid \overline{a} \mid \overline{b}) \to (T_1 \mid \overline{b} \mid \overline{b} \mid \overline{b} \mid \overline{c}) \to\to\to\not\to$.
- $\{3,2\} \to \{2,2,2,1\} \to \{2,2,1,1,1\} \to \{2,1,1,1,1,1\} \to \{1,1,1,1,1,1,1\}$

Ruling out diverging examples

- $D_1 = !a^n.\overline{a}^n \mid \overline{a}^n$ not typable: $n > n$
- $D_2 = !a^n.\overline{b}^m \mid !b^m.\overline{a}^n \mid \overline{a}^n$ not typable: $n > m > n$.
- $D_3 = c^k(x).!a^n.\overline{x}^m \mid \overline{a}^n \mid \overline{c}^k\langle a\rangle$.
  $c : \sharp^k (\sharp^n T)$ implying $n = m$ and $n > m$.

Decreasing of the weight

- $T_1 = !a^3.(\overline{b}^2 \mid \overline{b}^2 \mid \overline{c}^1) \mid !b^2.(\overline{c}^1 \mid \overline{c}^1)$
- $(T_1 \mid \overline{a} \mid \overline{b}) \rightarrow (T_1 \mid \overline{b} \mid \overline{b} \mid \overline{b} \mid \overline{c}) \rightarrow\rightarrow\rightarrow\not\rightarrow$.
- $\{3, 2\} \rightarrow \{2, 2, 2, 1\} \rightarrow \{2, 2, 1, 1, 1\} \rightarrow \{2, 1, 1, 1, 1, 1\} \rightarrow \{1, 1, 1, 1, 1, 1, 1\}$

Limitations

Not able to prove the standard encoding of λ_{ST} into π.

Refining the analysis

Input sequences

- Principles: considering replicated input sequence as a whole.
- In $!a(x).b(y).c(z).P$ comparing $\{a, b, c\}$ with the multiset of outputs in P.

Partial order

- Encoding of list structures $P_l = !p(a, b).a(x).(\overline{b}\langle x \rangle \mid \overline{p}\langle a, b \rangle)$ with a and b having the same type.
- Introducing partial order in the typing judgements preventing the typability of $P_l \mid \overline{p}\langle u, v \rangle \mid \overline{p}\langle v, u \rangle$.

Refining the analysis II

Hybrid analysis

- Static analysis annotating type-checked processes (which can diverge).
- Controlled execution: maintaining an order between names and aborting the reduction if the system enters a loop.
- Example: $P = c(x).!a.\overline{x} \mid (\overline{c}\langle b\rangle \mid \overline{b}) \mid (\overline{c}\langle a\rangle \mid \overline{a})$
  - Reduction with $\overline{c}\langle b\rangle$: everything is fine.
  - Reduction with $\overline{c}\langle a\rangle$: aborted execution when $[a > a]$ is reached.

Refining the analysis II

Hybrid analysis

- Static analysis annotating type-checked processes (which can diverge).
- Controlled execution: maintaining an order between names and aborting the reduction if the system enters a loop.
- Example: $P = c(x).!a.[a > x]\overline{x} \mid (\overline{c}\langle b \rangle \mid \overline{b}) \mid (\overline{c}\langle a \rangle \mid \overline{a})$
  - Reduction with $\overline{c}\langle b \rangle$: everything is fine.
  - Reduction with $\overline{c}\langle a \rangle$: aborted execution when $[a > a]$ is reached.

Complexity of type inference

Type inference

- For a given process P, does there exists Γ and n such that $\vdash_\Gamma P : n$?
- Syntactically-directed type systems: type inference = level inference.

First system

- $a > b$ comparisons.
- Construction of a domination graph: (a, b) is an edge if $a > b$.
- Topological sort: polynomial inference.

With input sequences

- $\{a_1, \ldots, a_n\} > \{b_1, \ldots, b_n\}$ comparisons.
- Reduction from 3SAT: NP-completeness of the inference.
- Version with algebraic sum: polynomial (linear programming).

A proof-theory-based method

Principles

- Using the encoding of λ_{ST} in π.
- Termination of a functional subset of π by logical relations:
  - Assigning types to processes. ($\vdash_\Gamma P : T$)
  - Defining inductively interpretation of types as set of terminating processes. ($P \in [T]$)
  - Proving that every typable process belongs to the interpretation of its type. ($\vdash_\Gamma P : T \Rightarrow P \in [T]$)

Limitations

- Limited "concurrent" expressiveness.
- Allowed inputs on f: $(\nu f)\ (!f(x).P_1 \mid P_2)$ with f not free in P_1 and no reception of f in P_2.
- Yields a confluent calculus.

Termination of the standard π-calculus

- Difficult:
  - Symmetry of $\mid$.
  - No canonical types for processes.
- Simple typability of channels is not useful: ($!a.\bar{a} \mid \bar{a}$ can be simply typed).

An hybrid solution

Principles

- Combining the two methods to reach greater expressiveness.
- Distinguish some names as functional in *hybrid* processes.
- Extending the weight system to the functional part:
  Using the weight to obtain a diverging functional process from a well-typed hybrid process.
- Derive a contradiction with the termination of the functional calculus.

Ideas

- Papers from G.Boudol and R.Amadio on the termination of λ_{ref} (λ with references).
- Similarities between λ and π.

Functional names

- Some names are functional
  - Only one (replicated) input called *definition*: `def` $f = (x).P_1$ `in` P_2
  - syntactical sugar for $(\nu f)\ (!f(x).P_1 \mid P_2)$
  - No "recursion" $f \notin \mathrm{fn}(P_1)$.
- Semantics with evaluation contexts $\mathbf{E} = [\,] \mid (\mathbf{E} \mid P) \mid \texttt{def}\ f = P_1\ \texttt{in}\ \mathbf{E}$:

$$\frac{}{\mathbf{E}_1[\texttt{def}\ f = (x).P_1\ \texttt{in}\ \mathbf{E}_2[\overline{f}\langle v\rangle.P_2]] \to \mathbf{E}_1[\texttt{def}\ f = (x).P_1\ \texttt{in}\ \mathbf{E}_2[P_1\{v/x\} \mid P_2]]}$$

- Imperative names: $a(x).P \mid !a(x).P \mid \overline{a}\langle v\rangle$.

Typing

- Assigning a level to each name (either functional or imperative).
- Imperative replication: weight of the continuation strictly smaller.

Typing

- Assigning a level to each name (either functional or imperative).
- Imperative replication: weight of the continuation strictly smaller.
- Definition: weight of the continuation smaller or equal. $\texttt{def}\ f = \overline{a}\ \texttt{in}\ !a.\overline{f} \mid \overline{a}$.

Typing

- Assigning a level to each name (either functional or imperative).
- Imperative replication: weight of the continuation strictly smaller.
- Definition: weight of the continuation smaller or equal. $\texttt{def}\ f = \overline{a}\ \texttt{in}\ !a.\overline{f} \mid \overline{a}$.
- Name typing: $T ::= \texttt{b} \mid \sharp^k_{\texttt{F}}\ T \mid \sharp^k_{\texttt{I}}\ T \qquad (\text{and } \sharp^k_{\bullet}\ T)$

Type system I

$$\frac{}{\vdash_\Gamma \mathbf{0} : 0} \qquad \frac{\vdash_\Gamma P : l}{\vdash_\Gamma (\nu a)\, P : l} \qquad \frac{\vdash_\Gamma P_1 : l_1 \qquad \vdash_\Gamma P_2 : l_2}{\vdash_\Gamma P_1 \mid P_2 : \max(l_1, l_2)}$$

$$\frac{\vdash_\Gamma P : l \qquad \vdash_\Gamma v : \sharp^k_\bullet\, T \qquad \vdash_\Gamma w : T}{\vdash_\Gamma \overline{v}\langle w \rangle.P : \max(k, l)}$$

- Γ: typing context.
- Two kind of outputs treated simultaneously ($v = a$ or $v = f$).
- Taking into account the outputs (either imperative or functional) in the weight.

Type system II

$$\frac{\vdash_\Gamma P : l \quad \vdash_\Gamma a : \sharp^k_{\mathrm{I}}\ T \quad \vdash_\Gamma x : T \quad k > l}{\vdash_\Gamma !a(x).P : 0}$$

$$\frac{\vdash_\Gamma P : l \quad \vdash_\Gamma a : \sharp^k_{\mathrm{I}}\ T \quad \vdash_\Gamma x : T \quad k > l}{\vdash_\Gamma a(x).P : 0}$$

Imperative inputs

- Replication controlled by $k > l$.
- Non-replicated imperative inputs treated as the replicated one: $\texttt{def}\ f = a(x).(\overline{x} \mid \overline{a}\langle x\rangle)\ \texttt{in}\ \overline{f} \mid \overline{a}\langle f\rangle$

Type system III

$$\frac{\vdash_\Gamma P : l \quad \vdash_\Gamma a : \sharp^k_{\mathtt{I}}\, T \quad \vdash_\Gamma x : T \quad k > l}{\vdash_\Gamma\, !a(x).P : 0}$$

$$\frac{\vdash_\Gamma P_1 : l \qquad \vdash_\Gamma P_2 : l' \qquad \vdash_\Gamma f : \sharp^k_{\mathtt{F}}\, T \qquad \vdash_\Gamma x : T \qquad k \geq l \qquad f \notin \mathrm{fn}(P_1)}{\vdash_\Gamma \mathtt{def}\ f = (x).P_1\ \mathtt{in}\ P_2 : l'}$$

- Control $k \geq l$ on definition.

Type system III

$$\frac{\vdash_\Gamma P : l \quad \vdash_\Gamma a : \sharp^k_{\mathtt{I}}\ T \quad \vdash_\Gamma x : T \quad k > l}{\vdash_\Gamma !a(x).P : 0}$$

$$\frac{\vdash_\Gamma P_1 : l \quad \vdash_\Gamma P_2 : l' \quad \vdash_\Gamma f : \sharp^k_{\mathtt{F}}\ T \quad \vdash_\Gamma x : T \quad k \geq l \quad f \notin \mathrm{fn}(P_1)}{\vdash_\Gamma \mathtt{def}\ f = (x).P_1\ \mathtt{in}\ P_2 : l'}$$

- Control $k \geq l$ on definition.
- Possibility of typing the encoding of λ_{ST} with level 0 everywhere (no imperative names).

Terminating example

$$
\begin{array}{l}
(\nu\, lock_1, .., lock_k)\big(lock_1 \mid .. \mid lock_k \mid \\
\quad \texttt{def}\ s = (c, x).\overline{c}\langle \mathbf{enc}[c, x]\rangle\ \texttt{in} \\
\quad \texttt{def}\ c_1 = C_1\ \texttt{in} \quad \ldots \quad \texttt{def}\ c_k = C_k\ \texttt{in} \\
\quad (\overline{s}\langle c_1, \mathrm{msg}_1\rangle \mid \ldots \mid \overline{s}\langle c_1, \mathrm{msg}_n\rangle)\ \big)
\end{array}
$$

with $$C_i = (y_i).\overline{lock_i}.\overline{s}\langle c_{i+1}, \mathbf{dec}_i[y_i]\rangle.lock_i$$

- ▶ Encryption server s
  - Clients c_i into a chain data structure
  - No direct access to successor (informations travel through s)
  - Imperative locks.

Terminating example

$$
\begin{array}{l}
(\nu\, lock_1, .., lock_k)\big(lock_1 \mid .. \mid lock_k \mid \\
\quad \texttt{def}\ s = (c, x).\overline{c}\langle \mathbf{enc}[c, x]\rangle\ \texttt{in} \\
\quad \texttt{def}\ c_1 = C_1\ \texttt{in} \quad \ldots \quad \texttt{def}\ c_k = C_k\ \texttt{in} \\
\quad (\overline{s}\langle c_1, \mathrm{msg}_1\rangle \mid \ldots \mid \overline{s}\langle c_1, \mathrm{msg}_n\rangle)\ \big)
\end{array}
$$

with $$C_i = (y_i).\overline{lock_i}.\overline{s}\langle c_{i+1}, \mathbf{dec}_i[y_i]\rangle.lock_i$$

-
  - Encryption server s
  - Clients c_i into a chain data structure
  - No direct access to successor (informations travel through s)
  - Imperative locks.
- Typing: $lock_i^{0}$, c_i^{1}, s^{1}.
- Not typable with weights only (circularity s,c).
- Logical relations not usable ($lock_i$ are imperative)

Proof by pruning

Principles

- Define a pruning $\mathrm{pr}_\Gamma^p()$ of hybrid processes into functional processes.
- Obtain a simulation lemma: If $P \to P'$ then:
  1. either $\mathrm{pr}_\Gamma^p(P) \simeq \mathrm{pr}_\Gamma^p(P')$
  2. or $\mathrm{pr}_\Gamma^p(P) \to \mathrm{pr}_\Gamma^p(P')$.
- Transform a well-typed diverging hybrid process into a functional diverging process (2. happens infinitely often)
  Raising a contradiction.

Maximum reduction level

- $\rightarrow_{\mathrm{F}}^{n}$: reduction of a definition of level n.
- $\rightarrow_{\mathrm{I}}^{n}$: reduction of an imperative communication on level n.

Decreasing lemma

We define $\mathbf{Os}^p(P)$ the number of available outputs of level p in P.
We prove:

- If $P \rightarrow_{\mathrm{F}}^{n} P'$ for $n < p$, $\mathbf{Os}^p(P) = \mathbf{Os}^p(P')$.
- If $P \rightarrow_{\mathrm{I}}^{n} P'$ for $n < p$, $\mathbf{Os}^p(P) = \mathbf{Os}^p(P')$.
- If $P \rightarrow_{\mathrm{I}}^{p} P'$, $\mathbf{Os}^p(P) > \mathbf{Os}^p(P')$.

Maximum reduction level

- $\rightarrow^n_{\mathrm{F}}$: reduction of a definition of level n.
- $\rightarrow^n_{\mathrm{I}}$: reduction of an imperative communication on level n.

Decreasing lemma

We define $\mathbf{Os}^p(P)$ the number of available outputs of level p in P.
We prove:

- If $P \rightarrow^n_{\mathrm{F}} P'$ for $n < p$, $\mathbf{Os}^p(P) = \mathbf{Os}^p(P')$.
- If $P \rightarrow^n_{\mathrm{I}} P'$ for $n < p$, $\mathbf{Os}^p(P) = \mathbf{Os}^p(P')$.
- If $P \rightarrow^p_{\mathrm{I}} P'$, $\mathbf{Os}^p(P) > \mathbf{Os}^p(P')$.

Maximum reduction level

- In every reduction sequence $(P_i)_{i\in\mathbb{N}}$, there exists a maximum level p such that for an infinite number of indices i, $P_i \rightarrow^p_{\bullet} P_{i+1}$.
- With the previous lemma, an infinite number of times, $P_i \rightarrow^p_{\mathrm{F}} P_{i+1}$

Pruning

$$\mathrm{pr}^p_\Gamma(a(x).P) = \mathrm{pr}^p_\Gamma(!a(x).P) = \mathrm{pr}^p_\Gamma(\mathbf{0}) = \mathbf{0} \qquad \mathrm{pr}^p_\Gamma(P_1 \mid P_2) = \mathrm{pr}^p_\Gamma(P_1) \mid \mathrm{pr}^p_\Gamma(P_2)$$

$$\mathrm{pr}^p_\Gamma((\boldsymbol{\nu} a)\, P) = \mathrm{pr}^p_\Gamma(P) \qquad \mathrm{pr}^p_\Gamma(\overline{a}\langle v\rangle.P) = \mathrm{pr}^p_\Gamma(P)$$

$$\mathrm{pr}^p_\Gamma(\mathtt{def}\ f^n = (x).P_1\ \mathtt{in}\ P_2) = \begin{cases} \mathtt{def}\ f = (x).\mathrm{pr}^p_\Gamma(P_1)\ \mathtt{in}\ \mathrm{pr}^p_\Gamma(P_2) & \text{if } n = p \\ \mathrm{pr}^p_\Gamma(P_2) & \text{otherwise} \end{cases}$$

$$\mathrm{pr}^p_\Gamma(\overline{f}^n\langle v\rangle.P) = \begin{cases} \overline{f}^n\langle v\rangle.P & \text{if } n = p \\ \mathbf{0} & \text{otherwise} \end{cases}$$

- Pruning *at level* p.
- Removing all imperative parts.
- Removing functional parts of level $\neq p$.

Simulation

Simulation Lemma

- If $P \to_{\mathrm{I}}^{n} P'$ for $n \leq p$ then $\mathsf{pr}_{\Gamma}^{p}(P) \simeq \mathsf{pr}_{\Gamma}^{p}(P')$.
- If $P \to_{\mathrm{F}}^{n} P'$ for $n < p$ then $\mathsf{pr}_{\Gamma}^{p}(P) \simeq \mathsf{pr}_{\Gamma}^{p}(P')$.
- If $P \to_{\mathrm{F}}^{p} P'$ then $\mathsf{pr}_{\Gamma}^{p}(P) \to \mathsf{pr}_{\Gamma}^{p}(P')$.

Simulation

Simulation Lemma

- If $P \to_{\mathrm{I}}^{n} P'$ for $n \leq p$ then $\mathrm{pr}_{\Gamma}^{p}(P) \simeq \mathrm{pr}_{\Gamma}^{p}(P')$.
- If $P \to_{\mathrm{F}}^{n} P'$ for $n < p$ then $\mathrm{pr}_{\Gamma}^{p}(P) \simeq \mathrm{pr}_{\Gamma}^{p}(P')$.
- If $P \to_{\mathrm{F}}^{p} P'$ then $\mathrm{pr}_{\Gamma}^{p}(P) \to \mathrm{pr}_{\Gamma}^{p}(P')$.

Sketch

- $P = (!a^{n}(x).P_1 \mid \bar{a}^{n}\langle v\rangle.P_2)$ with $n \leq p$.
  Thus $\mathrm{pr}_{\Gamma}^{p}(P) = \mathrm{pr}_{\Gamma}^{p}(P_2)$.
- $P \to P' = (!a^{n}(x).P_1 \mid P_2 \mid P_1\{v/x\})$
  Thus $\mathrm{pr}_{\Gamma}^{p}(P') = \mathrm{pr}_{\Gamma}^{p}(P_1\{v/x\}) \mid \mathrm{pr}_{\Gamma}^{p}(P_2)$
- Typing $!a^{n}(x).P_1$ implies $\mathrm{pr}_{\Gamma}^{p}(P_1\{v/x\}) \simeq \mathbf{0}$.

Simulation

Simulation Lemma

- If $P \to_{\mathrm{I}}^{n} P'$ for $n \leq p$ then $\mathsf{pr}_{\Gamma}^{p}(P) \simeq \mathsf{pr}_{\Gamma}^{p}(P')$.
- If $P \to_{\mathrm{F}}^{n} P'$ for $n < p$ then $\mathsf{pr}_{\Gamma}^{p}(P) \simeq \mathsf{pr}_{\Gamma}^{p}(P')$.
- If $P \to_{\mathrm{F}}^{p} P'$ then $\mathsf{pr}_{\Gamma}^{p}(P) \to \mathsf{pr}_{\Gamma}^{p}(P')$.

Sketch

- $P = (!a^{n}(x).P_1 \mid \bar{a}^{n}\langle v\rangle.P_2)$ with $n \leq p$.
  Thus $\mathsf{pr}_{\Gamma}^{p}(P) = \mathsf{pr}_{\Gamma}^{p}(P_2)$.
- $P \to P' = (!a^{n}(x).P_1 \mid P_2 \mid P_1\{v/x\})$
  Thus $\mathsf{pr}_{\Gamma}^{p}(P') = \mathsf{pr}_{\Gamma}^{p}(P_1\{v/x\}) \mid \mathsf{pr}_{\Gamma}^{p}(P_2)$
- Typing $!a^{n}(x).P_1$ implies $\mathsf{pr}_{\Gamma}^{p}(P_1\{v/x\}) \simeq \mathbf{0}$.

Remember: an infinite number of times, $P_i \to_{\mathrm{F}}^{p} P_{i+1}$.
We derive a contradiction. Every typable process terminates

Parametricity

Principles

- Using the proof of functional calculus as an argument of the method.
- Replacing the functional π-calculus by any terminating functional calculus.

Examples

- Functional calculus with integer recursions:
  $\texttt{def}\ f = (n, x).P_1\ \texttt{in}\ P_2$ typed if f appears in P_1 only in: $\overline{f}\langle n-1, v\rangle$.
- Existential polymorphism. (No level polymorphism)
- Iterating the method. (Several functional calculi).

Boudol07/Amadio09/Madet10

Types & Effects

- Store divided into regions.
- Regions denoted by integers.
- Effect n: can act on the memory up to region n.
- $T\ \texttt{ref}_n$ well-formed, $\forall N \geq n, N \notin T$.
- $T_1 \to^n T_2$ function whose body has effect n.

$$\frac{\vdash_\Gamma M : (T_1 \to^{n} T_2, m) \qquad \vdash_\Gamma N : (T_1, k)}{\vdash_\Gamma M\ N : (T_2, \max(m, n, k))}$$

$$\frac{\vdash_\Gamma M : (T_2, n) \qquad \Gamma(x) = T_1}{\vdash_\Gamma \lambda x.\, M : (T_1 \to^{n} T_2, 0)} \qquad \frac{\vdash_\Gamma M : (T\ \texttt{ref}_n, m)}{\vdash_\Gamma \texttt{deref}_n(M) : (T, \max(m, n))}$$

Annotated reductions

Store δ:

- $(\mathbf{E}[\lambda x.M_1\ V], \delta) \mapsto^{n}{}_{\mathrm{F}} (\mathbf{E}[M_1\{V/x\}], \delta)$ if $(\lambda x.M_1\ V)$ effect n.
- $(\mathbf{E}[\mathtt{deref}_n(u)], \delta\langle u \rightsquigarrow V\rangle) \mapsto^{n}{}_{\mathrm{I}} (V, \delta\langle u \rightsquigarrow V\rangle)$.
- ...

Proof by pruning

Similar principles:

- Removing imperative parts.
- Removing functional parts of level $\neq p$.
- Proving a simulation.
- Deriving a contradiction.

Pruning

- Subterm M related to the level p if:
  - Type of M contains $N \geq p$,
  - or M has effect $\geq p$.

Pruning

- Subterm M related to the level p if:
  - Type of M contains $N \geq p$,
  - or M has effect $\geq p$.
- We cannot prune subterms to $\mathbf{0}$ here:
  - We introduce $\mathtt{V}_T$: generic value of type T.

Pruning

- Subterm M related to the level p if:
  - Type of M contains $N \geq p$,
  - or M has effect $\geq p$.
- We cannot prune subterms to $\mathbf{0}$ here:
  - We introduce $\mathtt{V}_T$: generic value of type T.
- How to prune $\mathtt{deref}_n(M)$?
  - M can diverges.

Pruning

- Subterm M related to the level p if:
  - Type of M contains $N \geq p$,
  - or M has effect $\geq p$.
- We cannot prune subterms to $\mathbf{0}$ here:
  - We introduce $\mathtt{V}_T$: generic value of type T.
- How to prune $\mathtt{deref}_n(M)$?
  - M can diverges.
  - We use $\Pi^{(1,2)} = \lambda x.\lambda y.x$.
  - If M is of type $(T\ \mathtt{ref}_n)$, then $\mathrm{pr}^p_\Gamma(\mathtt{deref}_n(M)) = \Pi^{(1,2)}\ \mathtt{V}_T\ \mathrm{pr}^p_\Gamma(M)$.

Pruning II

Definition

If M is not related with level p:	$\mathrm{pr}^p_\Gamma(M) = \mathtt{V}_T$
Otherwise:	$\mathrm{pr}^p_\Gamma(M_1\ M_2) = \mathrm{pr}^p_\Gamma(M_1)\ \mathrm{pr}^p_\Gamma(M_2)$
	$\mathrm{pr}^p_\Gamma(\mathtt{ref}_n\ M_1) = (\Pi^{(1,2)}\ ()\ \mathrm{pr}^p_\Gamma(M_1))$
	$\mathrm{pr}^p_\Gamma(\mathtt{deref}_n(M_1)) = (\Pi^{(1,2)}\ \mathtt{V}_T\ \mathrm{pr}^p_\Gamma(M_1))$
	$\mathrm{pr}^p_\Gamma(u) = ()$
	$\ldots$

Pruning II

Definition

$$
\begin{array}{rl}
\text{If } M \text{ is not related with level } p: & \mathsf{pr}^p_\Gamma(M) = \mathtt{V}_T \\
\hline
\text{Otherwise:} & \mathsf{pr}^p_\Gamma(M_1\ M_2) = \mathsf{pr}^p_\Gamma(M_1)\ \mathsf{pr}^p_\Gamma(M_2) \\
& \mathsf{pr}^p_\Gamma(\mathtt{ref}_n\ M_1) = (\Pi^{(1,2)}\ ()\ \mathsf{pr}^p_\Gamma(M_1)) \\
& \mathsf{pr}^p_\Gamma(\mathtt{deref}_n(M_1)) = (\Pi^{(1,2)}\ \mathtt{V}_T\ \mathsf{pr}^p_\Gamma(M_1)) \\
& \mathsf{pr}^p_\Gamma(u) = () \\
& \ldots
\end{array}
$$

- Pruning of types: $\mathsf{pr}^p_\Gamma(T\ \mathtt{ref}_n) = \mathtt{unit}$
- If $\vdash_\Gamma M : (T, n)$ then $\vdash_{\mathrm{ST}} \mathsf{pr}^p_\Gamma(M) : \mathsf{pr}^p_\Gamma(T)$

Proof by pruning

Simulation

- If $(M, \delta) \mapsto_{\mathrm{I}}^{n} (M', \delta')$ for $n \leq p$ then $\mathrm{pr}_{\Gamma}^{p}(M) \rightharpoonup^{*} \mathrm{pr}_{\Gamma}^{p}(M')$.
- If $(M, \delta) \mapsto_{\mathrm{F}}^{n} (M', \delta')$ for $n < p$ then $\mathrm{pr}_{\Gamma}^{p}(M) \rightharpoonup^{*} \mathrm{pr}_{\Gamma}^{p}(M')$.
- If $(M, \delta) \mapsto_{\mathrm{F}}^{p} (M', \delta')$ then $\mathrm{pr}_{\Gamma}^{p}(M) \rightharpoonup^{+} \mathrm{pr}_{\Gamma}^{p}(M')$

Proof by pruning

Simulation

- If $(M, \delta) \mapsto_{\mathrm{I}}^{n} (M', \delta')$ for $n \leq p$ then $\mathrm{pr}_{\Gamma}^{p}(M) \rightarrow\!\!\triangleright^{*} \mathrm{pr}_{\Gamma}^{p}(M')$.
- If $(M, \delta) \mapsto_{\mathrm{F}}^{n} (M', \delta')$ for $n < p$ then $\mathrm{pr}_{\Gamma}^{p}(M) \rightarrow\!\!\triangleright^{*} \mathrm{pr}_{\Gamma}^{p}(M')$.
- If $(M, \delta) \mapsto_{\mathrm{F}}^{p} (M', \delta')$ then $\mathrm{pr}_{\Gamma}^{p}(M) \rightarrow\!\!\triangleright^{+} \mathrm{pr}_{\Gamma}^{p}(M')$

Proof sketch

- $M = \mathtt{deref}_{n}(u)$ with $n \leq K$ and $u : T\ \mathtt{ref}_{n}$
  thus $\mathrm{pr}_{\Gamma}^{p}(M) = \sqcap^{(1,2)}\ \mathtt{V}_{T}\ ()$
- $(M, \delta\langle u \rightsquigarrow V\rangle) \mapsto_{\mathrm{I}}^{n} (M', \delta\langle u \rightsquigarrow V\rangle)$ with $M' = V$
- Typability + Well-formedness $\Rightarrow$ V not related with level p
  thus $\mathrm{pr}_{\Gamma}^{p}(V) = \mathtt{V}_{T}$.
- Finally $\mathrm{pr}_{\Gamma}^{p}(M) \rightarrow\!\!\triangleright\!\!\rightarrow\!\!\triangleright\ \mathrm{pr}_{\Gamma}^{p}(M')$.

Proof by pruning II

Decreasing

We define the number of operators of level p of a term $\mathbf{Os}^p(M)$

- If $(M, \delta) \mapsto_{\mathrm{F}}^{n} (M', \delta')$ for $n < p$ then $\mathbf{Os}(M') \leq \mathbf{Os}(M)$.
- If $(M, \delta) \mapsto_{\mathrm{I}}^{n} (M', \delta')$ for $n < p$ then $\mathbf{Os}(M') \leq \mathbf{Os}(M)$.
- If $(M, \delta) \mapsto_{\mathrm{I}}^{p} (M', \delta')$ then $\mathbf{Os}(M') < \mathbf{Os}(M)$.

Proof by pruning II

Decreasing

We define the number of operators of level p of a term $\mathbf{Os}^p(M)$

- If $(M, \delta) \mapsto_{\mathrm{F}}^{n} (M', \delta')$ for $n < p$ then $\mathbf{Os}(M') \leq \mathbf{Os}(M)$.
- If $(M, \delta) \mapsto_{\mathrm{I}}^{n} (M', \delta')$ for $n < p$ then $\mathbf{Os}(M') \leq \mathbf{Os}(M)$.
- If $(M, \delta) \mapsto_{\mathrm{I}}^{p} (M', \delta')$ then $\mathbf{Os}(M') < \mathbf{Os}(M)$.

Proof is done as in the π case.

Conclusion / Future Works

- Common refinements of the weight-based type systems (sequences, finer analyses).
- Parametricity (Iterating of the method).
- Polymorphism (Types, Regions).
- Encoding $\lambda \to \pi$ (System T, λ_{ref}).

Appendix 1: The Example

$$
\begin{array}{l}
Sys_1 \stackrel{\text{def}}{=} (\nu\, lock_1, .., lock_k) \\
\quad \Big(\; lock_1 \mid .. \mid lock_k \mid \\
\quad\quad \texttt{def } s = (c, x).\overline{c}\langle \mathbf{enc}[c, x] \rangle \texttt{ in} \\
\quad\quad\quad \texttt{def } c_1 = C_1 \texttt{ in} \\
\quad\quad\quad\quad \ldots \\
\quad\quad\quad\quad \texttt{def } c_{k-1} = C_{k-1} \texttt{ in} \\
\quad\quad\quad\quad\quad \texttt{def } c_k = C_k \texttt{ in} \\
\quad\quad\quad\quad\quad\quad (\overline{s}\langle c_1, \mathrm{msg}_1 \rangle \mid \ldots \mid \overline{s}\langle c_1, \mathrm{msg}_n \rangle) \; \Big)
\end{array}
$$

with:

$$
\begin{array}{lcl}
C_i & \stackrel{\text{def}}{=} & (y_i).\overline{lock_i}.(lock_i \mid \overline{s}\langle c_{i+1}, \mathbf{dec}_i[y_i] \rangle) \\
C_k & \stackrel{\text{def}}{=} & (y_k).\overline{lock_k}.(lock_k \mid \overline{d}\langle \mathbf{dec}_k[y_k] \rangle)
\end{array}
$$

Levels

$lock_i : \sharp^0_{\mathtt{I}}\, \mathtt{unit}, \quad c_i : \sharp^1_{\mathtt{F}}\, \mathtt{b}, \quad s : \sharp^1_{\mathtt{F}}\, (\sharp^1_{\mathtt{F}}\, \mathtt{b} \times \mathtt{b}), \quad \mathrm{msg}_i : \mathtt{b}, \quad d : \sharp^1_{\mathtt{I}}\, \mathtt{b}$

Appendix 2: Another example

$$
\begin{array}{lll}
Sys_2 & \stackrel{\text{def}}{=} & \texttt{def}\ s = (n, r, f). \\
& & \quad (\ \ \texttt{if}\ f = \textbf{tintin}\ \texttt{then}\ (\nu h)\ (\overline{r}\langle h\rangle.\overline{h}) \\
& & \quad\ \ \mid \texttt{if}\ f = \textbf{asterix}\ \texttt{then}\ \ldots \\
& & \quad\ \ \ldots \\
& & \quad\ \ \mid \texttt{if}\ n > 0\ \texttt{then}\ \overline{s}\langle n-1, r, f\rangle\ ) \\
& & \texttt{in}\ \ (\boldsymbol{\nu} r')\,(\qquad \overline{s}\langle 15, r', \textbf{tintin}\rangle \\
& & \qquad\qquad\quad \mid (\boldsymbol{\nu} c)\,(\overline{c} \mid\ !c.r'(z).(\overline{c} \mid z))\,)
\end{array}
$$

Levels

$f, \textbf{tintin}, \textbf{asterix} : \mathtt{b}, \quad h, z : \mathtt{b}; \quad r, r' : \sharp^1_{\mathtt{F}}\,\mathtt{b}, \quad s : \sharp^2_{\mathtt{F}}\,(\mathtt{nat} \times \sharp^1_{\mathtt{F}}\,\mathtt{b} \times \mathtt{b}), \quad c : \sharp^0_{\mathtt{I}}\,\mathtt{b}$

Appendix 3: Reductions for λ

$$(\beta)\,\frac{}{(\lambda x.\,M\;V,\delta)\mapsto(M\{V/x\},\delta)}$$

$$(\textbf{ref})\,\frac{u\notin\mathrm{supp}(\delta)\qquad\vdash_\Gamma V:(T,\_)}{(\mathtt{ref}_n\;V,\delta)\mapsto(u,(\delta\langle u\rightsquigarrow V\rangle))}\qquad(\textbf{deref})\,\frac{\delta(u)=V}{(\mathtt{deref}_n(u),\delta)\mapsto(V,\delta)}$$

$$(\textbf{store})\,\frac{\vdash_\Gamma V:(T,\_)}{(u\mathtt{:=}_n V,(\delta))\mapsto((),(\delta\langle u\rightsquigarrow V\rangle))}$$

$$(\textbf{context})\,\frac{(M,\delta)\mapsto(M',\delta')}{(\mathbf{E}[M],\delta)\mapsto(\mathbf{E}[M'],\delta')}$$

Appendix 4: Typability for λ

$$(\textbf{App})\ \frac{\vdash_\Gamma M : (T_1 \to^n T_2, m) \qquad \vdash_\Gamma N : (T_1, k)}{\vdash_\Gamma M\ N : (T_2, \max(m, n, k))}$$

$$(\textbf{Abs})\ \frac{\vdash_\Gamma M : (T_2, n) \qquad \Gamma(x) = T_1}{\vdash_\Gamma \lambda x.\, M : (T_1 \to^n T_2, 0)}$$

$$(\textbf{Ref})\ \frac{\vdash_\Gamma M : (T_1, m)}{\vdash_\Gamma \mathtt{ref}_n\ M : (T_1\ \mathtt{ref}_n, \max(n, m))} \qquad (\textbf{Var})\ \frac{\Gamma(x) = T_1}{\vdash_\Gamma x : (T_1, 0)}$$

$$(\textbf{Uni})\ \frac{}{\vdash_\Gamma () : (\mathtt{unit}, 0)} \qquad (\textbf{Add})\ \frac{}{\vdash_\Gamma u_{(n, T_1)} : (T_1\ \mathtt{ref}_n, 0)}$$

$$(\textbf{Asg})\ \frac{\vdash_\Gamma M : (T_1\ \mathtt{ref}_n, m) \qquad \vdash_\Gamma N : (T_1, k)}{\vdash_\Gamma M \mathtt{:=}_n N : (\mathtt{unit}, \max(m, n, k))}$$

$$(\textbf{Drf})\ \frac{\vdash_\Gamma M : (T\ \mathtt{ref}_n, m)}{\vdash_\Gamma \mathtt{deref}_n(M) : (T, \max(m, n))} \qquad (\textbf{Emp})\ \frac{}{\vdash_\Gamma \emptyset}$$

$$(\textbf{Sto})\ \frac{\vdash_\Gamma \delta \qquad \vdash_\Gamma V : (T, 0)}{\vdash_\Gamma \delta\langle u_{(n,T)} \rightsquigarrow V\rangle}$$

Appendix 5: Pruning for λ

If M is not related to p:

$$\mathrm{pr}^p_\Gamma(M) = \mathtt{V}_T$$

Otherwise:

$$\begin{array}{rcl}
\mathrm{pr}^p_\Gamma(M_1\ M_2) & = & \mathrm{pr}^p_\Gamma(M_1)\ \mathrm{pr}^p_\Gamma(M_2) \\
\mathrm{pr}^p_\Gamma(x) & = & x \\
\mathrm{pr}^p_\Gamma(\lambda x.M_1) & = & \lambda x.\mathrm{pr}^p_\Gamma(M_1) \\
\mathrm{pr}^p_\Gamma(\mathtt{ref}_n\ M_1) & = & (\Pi^{(1,2)}\ ()\ \mathrm{pr}^p_\Gamma(M_1)) \\
\mathrm{pr}^p_\Gamma(\mathtt{deref}_n(M_1)) & = & (\Pi^{(1,2)}\ \mathtt{V}_T\ \mathrm{pr}^p_\Gamma(M_1)) \\
\mathrm{pr}^p_\Gamma(M_1 \mathtt{:=}_n M_2) & = & (\Pi^{(1,3)}\ ()\ \mathrm{pr}^p_\Gamma(M_1)\ \mathrm{pr}^p_\Gamma(M_2)) \\
\mathrm{pr}^p_\Gamma(u_{(n,T_1)}) & = & ()
\end{array}$$

Appendix 6: 2 Lemmas for λ

Dividing evaluation contexts

Let $\mathbf{E}$ be an evaluation context and p an integer.

1. Either for all M, $\mathrm{pr}^p_\Gamma(\mathbf{E}[M]) = \mathrm{pr}^p_\Gamma(\mathbf{E})[\mathrm{pr}^p_\Gamma(M)]$
2. Or there exists $\mathbf{E}_1$ and $\mathbf{E}_2 \neq []$ s.t. $\mathbf{E} = \mathbf{E}_1[\mathbf{E}_2]$ and, for all M:
   - 2.1 If M has effect $\geq p$, then $\mathrm{pr}^p_\Gamma(\mathbf{E}[M]) = \mathrm{pr}^p_\Gamma(\mathbf{E}[M]) = \mathrm{pr}^p_\Gamma(\mathbf{E})[\mathrm{pr}^p_\Gamma(M)]$.
   - 2.2 If M has effect $< p$, then $\mathrm{pr}^p_\Gamma(\mathbf{E}[M]) = \mathrm{pr}^p_\Gamma(\mathbf{E}_1)[\mathtt{V}_{T''}]$ (where T'' is the type of $\mathbf{E}_2$).

Context reduction

If $\vdash_\Gamma \mathbf{E}_2 : (T'', m)$ and $\mathbf{E}_2$ is not related with level p, for all terms M, M',

1. $\mathrm{pr}^p_\Gamma(\mathbf{E}_2)[(\Pi^{(1,2)}\ \mathtt{V}_T\ M)] \rightharpoonup^+ \mathtt{V}_{T''}$;
2. $\mathrm{pr}^p_\Gamma(\mathbf{E}_2)[(\Pi^{(1,3)}\ \mathtt{V}_T\ M\ M')] \rightharpoonup^+ \mathtt{V}_{T''}$.