

Termination in Impure Languages

R. Demangeon¹

(joint work with D. Hirschhoff¹ & D. Sangiorgi²)

CONCUR 2010 - Paris

¹ENS Lyon

²University of Bologna

Motivations, the π calculus

Weight-based type systems for termination

An impure π -calculus

Termination proof

Termination

- ▶ A program terminates if every execution is finite.
- ▶ Useful property: in itself, prerequisite for soundness, fairness.

Sequential Case

- ▶ Termination is well-studied:
 - ▶ for functional programs (type systems, logical relations),
 - ▶ for imperative programs (abstract interpretation, loop analyses),
 - ▶ for rewriting systems (rewriting orderings, polynomial interpretations).

Concurrent case

- ▶ Shared memory setting:
extension of Terminator by Cook et al.
- ▶ Message-passing setting: more challenging, dynamic topology.

Starting points

Two different approaches to ensure termination for mobile processes (π -calculus).

Using weights [DengSangiorgi06]

- ▶ Methods from rewriting theory.
- ▶ Ensuring that some measure (weight) of the system decreases along computation.

Using logical relations [Sangiorgi06] and [YoshidaBergerHonda04]

- ▶ Semantical methods.
- ▶ Logical relations ensure termination of a functional subcalculus of π .

π -calculus

Syntax and semantics

- ▶ Names (or channels) a, b, c, x, y
- ▶ Processes $P ::= \mathbf{0} \mid P \mid P \mid (\nu a) P \mid \bar{a}\langle v \rangle.P \mid a(x).P \mid !a(x).P$

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

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

- ▶ $!a(x).P$: modelling servers.
- ▶ Persistence of replication yields divergences.

Divergence

Definition

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

Diverging processes

CCS notation $a.\bar{b}$

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

Divergence

Definition

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

Diverging processes

CCS notation $a.\bar{b}$

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

Divergence

Definition

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

Diverging processes

CCS notation $a.\bar{b}$

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

Divergence

Definition

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

Diverging processes

CCS notation $a.\bar{b}$

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

Motivations, the π calculus

Weight-based type systems for termination

An impure π -calculus

Termination proof

Weight-based type systems [DengSangiorgi06]

Principles

- ▶ Associate a *level* $k \in \mathbb{N}$ to each name through types:

$$T ::= \text{base} \mid \#^k T$$

if $a : \#^k T$, write $\text{lvl}(a) = k$, sometimes $!a^k(x).P$

- ▶ Assign weights to processes
(corresponding to levels of outputs they contain).
- ▶ Typing judgements

$$\vdash P : n \quad \text{implies} \quad \max\{\text{lvl}(c) ; \bar{c}\langle v \rangle \in P\} = n$$

Weight-based type systems [DengSangiorgi06]

Principles

- ▶ Associate a *level* $k \in \mathbb{N}$ to each name through types:

$$T ::= \text{base} \mid \sharp^k T$$

if $a : \sharp^k T$, write $\text{lvl}(a) = k$, sometimes $!a^k(x).P$

- ▶ Assign weights to processes
(corresponding to levels of outputs they contain).
- ▶ Typing judgements

$$\vdash_{\Gamma} P : n \quad \text{implies} \quad \max\{\text{lvl}(c) ; \bar{c}\langle v \rangle \in P\} = n$$

- ▶ Controlling replicated inputs $!a^k(x).P$: $k >$ the weight of P

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

Soundness [DengSangiorgi06]

- ▶ Every typable process terminates.
- ▶ Consider a reduction

$$!a^k(x).P_1 \mid \bar{a}^k\langle v \rangle \rightarrow !a^k(x).P_1 \mid P_1\{v/x\}$$

$\Rightarrow \{a^k\}$ is traded with $\{a_1^{k_1}, \dots, a_n^{k_n}\}$ with $k_i < k$.

- ▶ Counting the multiset of levels of all available outputs.
Available outputs: not under a replication.
- ▶ Well-foundedness of a multiset ordering yields termination.

Soundness [DengSangiorgi06]

- ▶ Every typable process terminates.
- ▶ Consider a reduction

$$!a^k(x).P_1 \mid \bar{a}^k\langle v \rangle \rightarrow !a^k(x).P_1 \mid P_1\{v/x\}$$

$\Rightarrow \{a^k\}$ is traded with $\{a_1^{k_1}, \dots, a_n^{k_n}\}$ with $k_i < k$.

- ▶ Counting the multiset of levels of all available outputs.
Available outputs: not under a replication.
- ▶ Well-foundedness of a multiset ordering yields termination.

- ▶ Nota: no control on non-replicated inputs.

$$\frac{\vdash_{\Gamma} P : n \quad \Gamma(a) = \#^k T \quad \Gamma(x) = T}{\vdash_{\Gamma} a(x).P : n}$$

Examples

Example of reduction

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

Examples

Example of reduction

- ▶ $T_1 = !a^3.(\bar{b}^2 \mid \bar{b}^2 \mid \bar{c}^1) \mid !b^2.(\bar{c}^1 \mid \bar{c}^1)$
- ▶ $T_1 \mid \bar{a} \mid \bar{b} \xrightarrow{2} T_1 \mid \bar{a} \mid \bar{c} \mid \bar{c} \xrightarrow{3} \xrightarrow{2} \xrightarrow{2} \nrightarrow$.
- ▶ $\{3, 2\} \xrightarrow{2} \{3, 1, 1\} \xrightarrow{3} \{2, 2, 1, 1, 1\} \xrightarrow{2} \{2, 1, 1, 1, 1, 1\} \xrightarrow{2}$
 $\{1, 1, 1, 1, 1, 1, 1\}$

Examples

Example of reduction

- ▶ $T_1 = !a^3.(\bar{b}^2 \mid \bar{b}^2 \mid \bar{c}^1) \mid !b^2.(\bar{c}^1 \mid \bar{c}^1)$
- ▶ $T_1 \mid \bar{a} \mid \bar{b} \xrightarrow{2} T_1 \mid \bar{a} \mid \bar{c} \mid \bar{c} \xrightarrow{3} \xrightarrow{2} \xrightarrow{2} \nrightarrow$.
- ▶ $\{3, 2\} \xrightarrow{2} \{3, 1, 1\} \xrightarrow{3} \{2, 2, 1, 1, 1\} \xrightarrow{2} \{2, 1, 1, 1, 1, 1\} \xrightarrow{2} \{1, 1, 1, 1, 1, 1, 1\}$

The diverging examples are rejected

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

Examples

Example of reduction

- ▶ $T_1 = !a^3.(\bar{b}^2 \mid \bar{b}^2 \mid \bar{c}^1) \mid !b^2.(\bar{c}^1 \mid \bar{c}^1)$
- ▶ $T_1 \mid \bar{a} \mid \bar{b} \xrightarrow{2} T_1 \mid \bar{a} \mid \bar{c} \mid \bar{c} \xrightarrow{3} \xrightarrow{2} \xrightarrow{2} \nrightarrow$.
- ▶ $\{3, 2\} \xrightarrow{2} \{3, 1, 1\} \xrightarrow{3} \{2, 2, 1, 1, 1\} \xrightarrow{2} \{2, 1, 1, 1, 1, 1\} \xrightarrow{2} \{1, 1, 1, 1, 1, 1, 1\}$

The diverging examples are rejected

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

Examples

Example of reduction

- ▶ $T_1 = !a^3.(\bar{b}^2 \mid \bar{b}^2 \mid \bar{c}^1) \mid !b^2.(\bar{c}^1 \mid \bar{c}^1)$
- ▶ $T_1 \mid \bar{a} \mid \bar{b} \xrightarrow{2} T_1 \mid \bar{a} \mid \bar{c} \mid \bar{c} \xrightarrow{3} \xrightarrow{2} \xrightarrow{2} \nrightarrow$.
- ▶ $\{3, 2\} \xrightarrow{2} \{3, 1, 1\} \xrightarrow{3} \{2, 2, 1, 1, 1\} \xrightarrow{2} \{2, 1, 1, 1, 1, 1\} \xrightarrow{2} \{1, 1, 1, 1, 1, 1, 1\}$

The diverging examples are rejected

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

Examples

Example of reduction

- ▶ $T_1 = !a^3.(\bar{b}^2 \mid \bar{b}^2 \mid \bar{c}^1) \mid !b^2.(\bar{c}^1 \mid \bar{c}^1)$
- ▶ $T_1 \mid \bar{a} \mid \bar{b} \xrightarrow{2} T_1 \mid \bar{a} \mid \bar{c} \mid \bar{c} \xrightarrow{3} \xrightarrow{2} \xrightarrow{2} \nrightarrow$.
- ▶ $\{3, 2\} \xrightarrow{2} \{3, 1, 1\} \xrightarrow{3} \{2, 2, 1, 1, 1\} \xrightarrow{2} \{2, 1, 1, 1, 1, 1\} \xrightarrow{2} \{1, 1, 1, 1, 1, 1, 1\}$

The diverging examples are rejected

- ▶ $D_1 = !a^n.\bar{a}^n \mid \bar{a}^n$ not typable: $n > n$
- ▶ $D_2 = !a^n.\bar{b}^k \mid !b^k.\bar{a}^n \mid \bar{a}^n$ not typable: $n > k > n$.
- ▶ $D_3 = c^l(x).!a^n.\bar{x}^k \mid \bar{a}^n \mid \bar{c}^l\langle a \rangle \mid \bar{c}^l\langle b \rangle$.
 $c : \#^l (\#^n T)$ implies $n = k$ and $n > k$.

Examples

Example of reduction

- ▶ $T_1 = !a^3.(\bar{b}^2 \mid \bar{b}^2 \mid \bar{c}^1) \mid !b^2.(\bar{c}^1 \mid \bar{c}^1)$
- ▶ $T_1 \mid \bar{a} \mid \bar{b} \xrightarrow{2} T_1 \mid \bar{a} \mid \bar{c} \mid \bar{c} \xrightarrow{3} \xrightarrow{2} \xrightarrow{2} \nrightarrow$.
- ▶ $\{3, 2\} \xrightarrow{2} \{3, 1, 1\} \xrightarrow{3} \{2, 2, 1, 1, 1\} \xrightarrow{2} \{2, 1, 1, 1, 1, 1\} \xrightarrow{2} \{1, 1, 1, 1, 1, 1, 1\}$

The diverging examples are rejected

- ▶ $D_1 = !a^n.\bar{a}^n \mid \bar{a}^n$ not typable: $n > n$
- ▶ $D_2 = !a^n.\bar{b}^k \mid !b^k.\bar{a}^n \mid \bar{a}^n$ not typable: $n > k > n$.
- ▶ $D_3 = c^l(x).!a^n.\bar{x}^k \mid \bar{a}^n \mid \bar{c}^l\langle a \rangle \mid \bar{c}^l\langle b \rangle$.
 $c : \#^l (\#^n T)$ implies $n = k$ and $n > k$.

Limitations of this approach

Not able to prove termination for standard encodings of simply-typed λ ($\lambda_{\mathbf{ST}}$) into π .

Motivations, the π calculus

Weight-based type systems for termination

An impure π -calculus

Termination proof

Our main contribution

- ▶ Starting with a *terminating functional* subset of the π -calculus.
- ▶ Adding **imperative** names controlled by levels.
 - ▶ **imperative** replications.
 - ▶ **functional** replications.
- ▶ Extending the type system for levels to the whole calculus:
 - ▶ $k > n$ for imperative replications.
 - ▶ $k \geq n$ for functional replications.
- ▶ Termination proof via an original method.

A terminating functional π -calculus

“Functional subsets” of π

- ▶ [Sangiorgi06] and [YoshidaBergerHonda04].
- ▶ Termination of a functional subset of π using logical relations.
- ▶ Drawing inspiration from the encoding of λ_{ST} in π .

Limitations

- ▶ Limited “concurrent expressiveness”.
- ▶ Yields a confluent calculus.
- ▶ Inputs in [Sangiorgi06]:

$$(\nu f) (!f(x).P_1 \mid P_2)$$

with f not free in P_1 and no reception on f in P_2 .

An impure π -calculus

$P ::= \mathbf{0} \mid P|P \mid \bar{v}\langle w \rangle.P \mid (\nu a) P \mid a(x).P \mid !a(x).P \mid \mathbf{def} f = (x).P \mathbf{in} P$

Functional names

- ▶ Some names are **functional**:
 - ▶ Inputs replaced by *definitions*: $\mathbf{def} f = (x).P_1 \mathbf{in} P_2$
 - ▶ syntactical sugar for $(\nu f) (!f(x).P_1 \mid P_2)$
 - ▶ No “recursion”: $f \notin \text{fn}(P_1)$.
 - ▶ No input on f in P_1 or P_2 .
 - ▶ Standard outputs $\bar{f}\langle v \rangle$.
- ▶ Reduction

$$\mathbf{def} f = (x).P_1 \mathbf{in} (\bar{f}\langle v \rangle.P_2 \mid P_3)$$
$$\rightarrow \mathbf{def} f = (x).P_1 \mathbf{in} (P_1\{v/x\} \mid P_2 \mid P_3)$$

Imperative names

- ▶ Standard inputs and outputs $a(x).P \mid !a(x).P \mid \bar{a}\langle v \rangle.P$

Type system: Outputs and other rules

Types for names: $T ::= \text{base} \mid \#_F^k T \mid \#_I^k T$

$$\frac{}{\vdash_{\Gamma} \mathbf{0} : 0} \quad \frac{\vdash_{\Gamma} P : n}{\vdash_{\Gamma} (\nu a) P : n} \quad \frac{\vdash_{\Gamma} P_1 : n_1 \quad \vdash_{\Gamma} P_2 : n_2}{\vdash_{\Gamma} P_1 \mid P_2 : \max(n_1, n_2)}$$

$$\frac{\vdash_{\Gamma} P : n \quad \Gamma(v) = \#_{\bullet}^k T \quad \Gamma(w) = T}{\vdash_{\Gamma} \bar{v}\langle w \rangle.P : \max(k, n)}$$

- ▶ Rule for output

$v : \#_{\bullet}^k T$: v is either functional or imperative ($v = f$ or $v = a$).

Type system: Imperative inputs

$$\frac{\vdash_{\Gamma} P : n \quad \Gamma(a) = \#_I^k T \quad \Gamma(x) = T \quad k > n}{\vdash_{\Gamma} !a(x).P : 0}$$

$$\frac{\vdash_{\Gamma} P : n \quad \Gamma(a) = \#_I^k T \quad \Gamma(x) = T \quad k > n}{\vdash_{\Gamma} a(x).P : 0}$$

- ▶ Strict condition $k > n$.
- ▶ Non-replicated imperative inputs treated as the replicated ones: $\text{def } f^1 = a^1(x).(\overline{x^1} \mid \overline{a^1}\langle x \rangle) \text{ in } (\overline{f^1} \mid \overline{a^1}\langle f \rangle)$

Type system: Functional definitions

$$\frac{\begin{array}{l} \vdash_{\Gamma} P_1 : n \quad \vdash_{\Gamma} P_2 : n' \quad \Gamma(f) = \#_{\mathbb{F}}^k T \quad \Gamma(x) = T \\ k \geq n \quad f \notin \text{fn}(P_1) \end{array}}{\vdash_{\Gamma} \text{def } f = (x).P_1 \text{ in } P_2 : n'}$$

- ▶ Non-strict condition $k \geq n$.
- ▶ Consequence on expressiveness:
the encoding of λ_{ST} in π can be typed by setting all levels to 0
- ▶ Allowing $k < n$ is dangerous: $\text{def } f^1 = \bar{a}^2 \text{ in } (!a^2.\bar{f}^1 \mid \bar{a}^2)$.

Expressiveness

- ▶ The resulting system combines existing approaches for termination in the π -calculus
 - ▶ Encoding λ_{ST} is typable.
 - ▶ Terminating non-confluent behaviours.
- ▶ It goes beyond [DengSangiorgi06] \cup [Sangiorgi06].

Example (in the paper):

clients and servers able to call each other,
critical resources protected by locks.

- ▶ Not typable with weights only (circularity of calls between clients and the server).
- ▶ Logical relations cannot be used (locks are imperative).

Motivations, the π calculus

Weight-based type systems for termination

An impure π -calculus

Termination proof

Termination of impure processes

- ▶ By absurd:
consider an infinite derivation from a well-typed process P_0 :

$$P_0 \xrightarrow{2} \xrightarrow{5} \xrightarrow{4} \xrightarrow{5} \xrightarrow{4} \xrightarrow{3} \xrightarrow{5} \dots$$

- ▶ There is a maximal p s.t. infinitely often: \xrightarrow{p} or \xrightarrow{p} .
- ▶ The type system implies that \xrightarrow{p} happens infinitely often.
(\xrightarrow{p} is a decreasing)
- ▶ Infinitely many **functional** reductions at level p .
Contradiction ?

Proof by pruning

Pruning and reduction sequences

$$\begin{array}{ccccccc} \rightarrow^2 & \rightarrow^5 & \rightarrow^4 & \rightarrow^5 & \rightarrow^4 & \rightarrow^3 & \rightarrow^5 & \dots \\ & & & \downarrow \text{pr}^5 & & & & \\ \simeq & \simeq & \simeq & \rightarrow & \simeq & \simeq & \rightarrow & \dots \end{array}$$

Principles

- ▶ **Pruning** $\text{pr}^P()$ of impure into **pure functional processes**.
 - ▶ Removing all imperative parts and some functional parts.
- ▶ **Simulation lemma**: If $P \rightarrow P'$ then:
 1. either $\text{pr}^P(P) \simeq \text{pr}^P(P')$
 2. or $\text{pr}^P(P) \rightarrow \text{pr}^P(P')$.
 3. \rightarrow^P transformed by pr^P into \rightarrow .
- ▶ **Contradiction with termination of the pure functional calculus.**

Pruning

$$\text{pr}^p(a(x).P) = \text{pr}^p(!a(x).P) = \mathbf{0} \qquad \text{pr}^p(\bar{a}\langle v \rangle.P) = \text{pr}^p(P)$$

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

$$\text{pr}^p(\bar{f}^n\langle v \rangle.P) = \begin{cases} \bar{f}^n\langle v \rangle.\text{pr}^p(P) & \text{if } n = p \\ P & \text{otherwise} \end{cases}$$

- ▶ Pruning has to be done *at level p*.
- ▶ Removing all **imperative parts**.
- ▶ Removing **functional parts** at levels $\neq p$.

Simulation

Simulation Lemma

- ▶ If $P \rightarrow^n P'$ for $n \leq p$ then $\text{pr}^p(P) \simeq \text{pr}^p(P')$.
- ▶ If $P \rightarrow^n P'$ for $n < p$ then $\text{pr}^p(P) \simeq \text{pr}^p(P')$.
- ▶ If $P \rightarrow^p P'$ then $\text{pr}^p(P) \rightarrow \text{pr}^p(P')$.

Holds thanks to conditions $k > l$ and $k \geq l$ in the typing rules.

Soundness

- ▶ Taking a well-typed diverging process P_0 .
- ▶ By the Simulation Lemma, $\text{pr}^p(P_0)$ diverges.
- ▶ Contradicts the termination of the pure functional calculus.

Applications of the pruning method

Parametricity

- ▶ Termination proof for **pure functional** π = argument of the method.
- ▶ **Pure functional** $\pi \rightsquigarrow$ any terminating **functional** calculus
 - ▶ Functional calculus with integer recursions:
 $\text{def } f = (n, x). P_1 \text{ in } P_2$
well-typed if f appears in P_1 only in $\bar{f}\langle n-1, v \rangle$.
 - ▶ Polymorphism (types can be quantified, not levels).
 - ▶ Iteration of the method. (Several functional calculi $F_1 \subsetneq F_2 \subsetneq \dots$).

λ -calculus with references

- ▶ Type and effect system for λ_{ref} : λ_{ST} with stores ([Boudo107], [Amadio09])
- ▶ New proof: λ_{ST} functional, **memory operators** imperative.
- ▶ Pruning λ_{ref} into λ_{ST} .

Conclusion / Future Works

- ▶ Refining the method for π :
 - ▶ Existing refinements of the weight-based type systems.
 - ▶ Studying the parametricity.
 - ▶ Encoding $\lambda \rightarrow \pi$ (System T , λ_{ref}).
- ▶ Applying the method to other impure languages (λ -calculi, object calculi).

Appendix 1: The Example

$$\begin{aligned} \text{Sys}_1 &\stackrel{\text{def}}{=} (\nu \text{lock}_1, \dots, \text{lock}_k) \\ &\quad \left(\text{lock}_1 \mid \dots \mid \text{lock}_k \mid \right. \\ &\quad \quad \text{def } s = (c, x). \overline{c} \langle \text{enc}[c, x] \rangle \text{ in} \\ &\quad \quad \text{def } c_1 = C_1 \text{ in} \\ &\quad \quad \dots \\ &\quad \quad \text{def } c_{k-1} = C_{k-1} \text{ in} \\ &\quad \quad \text{def } c_k = C_k \text{ in} \\ &\quad \quad \left. (\overline{s} \langle c_1, \text{msg}_1 \rangle \mid \dots \mid \overline{s} \langle c_1, \text{msg}_n \rangle) \right) \end{aligned}$$

with:

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

Levels

$\text{lock}_i : \#_1^0 \text{ unit}$, $c_i : \#_F^1 \text{ base}$, $s : \#_F^1 (\#_F^1 \text{ base} \times \text{base})$, $\text{msg}_i : \text{base}$, $d : \#_1^1 \text{ base}$

Appendix 2: Another example

$$\begin{aligned} \text{Sys}_2 &\stackrel{\text{def}}{=} \text{def } s = (n, r, f). \\ &\quad (\text{if } f = \mathbf{tintin} \text{ then } (\nu h) (\bar{r}\langle h \rangle.\bar{h}) \\ &\quad \quad | \text{if } f = \mathbf{asterix} \text{ then } \dots \\ &\quad \quad \dots \\ &\quad \quad | \text{if } n > 0 \text{ then } \bar{s}\langle n-1, r, f \rangle) \\ &\text{in } (\nu r') (\quad \bar{s}\langle 15, r', \mathbf{tintin} \rangle \\ &\quad \quad | (\nu c) (\bar{c} \mid !c.r'(z).(\bar{c} \mid z))) \end{aligned}$$

Levels

$f, \mathbf{tintin}, \mathbf{asterix} : \text{base}$, $h, z : \text{base}$; $r, r' : \#_F^1 \text{ base}$,
 $s : \#_F^2 (\text{nat} \times \#_F^1 \text{ base} \times \text{base})$, $c : \#_I^0 \text{ base}$

Appendix 3: Pruning has to be done on a level p

- ▶ $\text{pr}(!a.\bar{f} \mid \bar{a}) = \mathbf{0}$
- ▶ $\text{pr}(!a.\bar{f} \mid \bar{f}) = \bar{f}$
- ▶ $!a.\bar{f} \mid \bar{a} \rightarrow !a.\bar{f} \mid \bar{f}$