A Type System for Depth-Bounded Pi-Calculus:

Luke Ong

(Joint work with Emanuele D'Osualdo)

University of Oxford

IFIP WG2.2, 21-23 September 2015

Resource Analysis of Pi-Calculus: Some Questions

Soter: a coverability / safety verification tool for Erlang that uses abstract interpretation, via a CCS intermediate representation, to extract a Petri net model from an input Erlang program.

http://mjolnir.cs.ox.ac.uk/soter Google "soter Erlang"

A source of imprecision of Soter abstraction: unboundedly many Erlang pids (**p**rocess **id**s) are abstracted as finite pid equivalence classes.

- Unable to support analysis that requires precision of process identity.
- Because mailboxes are merged under the abstraction, certain patterns of communication cannot be analysed accurately.

Pi-calculus would be a more accurate model: pids can be modelled by names.

Question

Is there a pi-calculus fragment in which reasoning about process identity can be made precise, while retaining decidability of the analysis?

Resource Analysis of Pi-Calculus: Some Questions

Pi-calculus variants (Spi-Calculus and Applied Pi-Calculus) have been used successfully in reasoning about cryptographic protocols.

Secrecy Problem for Protocol P

Given a secret M, can protocol P leak M?

DEF. Protocol P can leak M if there are (intruder term) I, evaluation context C, channel $c \notin \mathrm{bn}(C)$ and term R such that

$$(P \parallel I) \to^* C[\overline{c}\langle M\rangle.R]$$

without renaming $\mathrm{fn}(M)$.

Protocol secrecy remains undecidable even under drastic restrictions, e.g., bounding message size and encryption depth, but with unbounded sessions and nonces. (Durgin et al. FMSP'99)

Question

Is there an expressive class of protocols P for which Secrecy is decidable?

Outline

1. Depth-bounded and other Resource-bounded Fragments of the Pi-Calculus
2. A New Fragment: Tree-Compatible Pi-Terms
3. A Type System for Tree-Compatible Pi-Terms
4. Algorithmic Properties of Typably Tree-Compatible Pi-Terms
5. Summary and Further Directions

The Pi-Calculus (Milner, Parrow & Walker 1992)

The **pi-calculus** models communications between processes that exchange messages along channels.

- Messages and channels are represented uniformly by names from a countable set $\mathcal{N}$.
- Processes communicate by synchronising on a pair of send and receive prefixes:
  - $\overline{a}\langle b\rangle$ sends message b on channel a
  - $a(x)$ receives message x on channel a.

Syntax of π-terms:

$$
\begin{aligned}
P &:= \nu x.P \mid P_1 \parallel P_2 \mid M \mid M^* && \text{process / } \pi\text{-term} \\
M &:= \mathbf{0} \mid \pi.P \mid M + M && \text{choice} \\
\pi &:= \overline{a}\langle b\rangle \mid a(x) \mid \tau && \text{prefix}
\end{aligned}
$$

M (choice) and M^* (replication) are called sequential.

Operational Semantics

Structural congruence, $\equiv$, is the least relation that respects α-conversion of bound names, where $+$ and $\parallel$ are associative and commutative with neutral element $\mathbf{0}$, satisfying: $\nu a.\mathbf{0} \equiv \mathbf{0}$, $\quad \nu a.\nu b.P \equiv \nu b.\nu a.P$,

$$P^* \equiv P \parallel P^* \qquad \text{Replication}$$

$$P \parallel \nu a.Q \equiv \nu a.(P \parallel Q) \quad (\text{if } a \notin \mathrm{fn}(P)) \qquad \text{Scope Extrusion}$$

Reaction relation, $\rightarrow$, is the least relation closed under $\parallel$, $\nu a.$ and $\equiv$, and satisfying:

$$(\overline{a}\langle b\rangle.S + M_S) \parallel (a(x).R + M_R) \ \rightarrow\ S \parallel R[b/x] \qquad \text{(React)}$$

$$\tau.P + M \ \rightarrow\ P \qquad \text{(Tau)}$$

(We do not consider labelled transition semantics.)

E.g. Let $S = \tau.\nu b.\overline{a}\langle b\rangle$, and $R = a(x).\overline{x}\langle c\rangle$. For prefix π, write $\pi.\mathbf{0}$ as π.

$$
\begin{array}{ll}
& S^* \parallel R^* \\
\equiv & \quad \text{(Replication)} \\
& (\tau.\nu b_1.\overline{a}\langle b_1\rangle \parallel S^*) \parallel (a(x_1).\overline{x_1}\langle c\rangle \parallel R^*) \\
\to & \quad \text{(Tau)} \\
& (\nu b_1.\overline{a}\langle b_1\rangle \parallel S^*) \parallel (a(x_1).\overline{x_1}\langle c\rangle \parallel R^*) \\
\equiv & \quad \text{Comm. \& assoc. of } \parallel \\
& (\nu b_1.\overline{a}\langle b_1\rangle) \parallel a(x_1).\overline{x_1}\langle c\rangle \parallel S^* \parallel R^* \\
\equiv & \quad \text{(Scope extrusion)} \\
& \nu b_1.\big(\overline{a}\langle b_1\rangle \parallel a(x_1).\overline{x_1}\langle c\rangle\big) \parallel S^* \parallel R^* \\
\to & \quad \text{(React)} \\
& \nu b_1.(\mathbf{0} \parallel \overline{b_1}\langle c\rangle) \parallel S^* \parallel R^* \\
\equiv & \quad (\equiv.1) \\
& (\nu b_1.\overline{b_1}\langle c\rangle) \parallel S^* \parallel R^*
\end{array}
$$

Some Resource-Bounded Fragments of Pi-Calculus

Natural fragments arise by restricting the use of channels / restrictions. (Meyer PhD Thesis 2008)

1. Bounding the depth of communication or depth of nested restrictions: depth-bounded processes.
2. Bounding the degree of sharing i.e. number of processes sharing a channel: breadth-bounded processes.
3. Bounding the number of channels used concurrently: name-bounded processes.

Features of these fragments

- Name bounded implies depth bounded; breadth bounded and depth bounded are incomparable.
- They are semantic: membership is undecidable.
- Expressive, yet still support decidable analyses (e.g. coverability).
- Useful for verification. E.g. name boundedness & memory leak.

Example Revisited: Depth-bounded but Name-unbounded

Let $S = \tau.\nu b.\overline{a}\langle b\rangle$, and $R = a(x).\overline{x}\langle c\rangle$ as before.

$$\begin{aligned} S^* \parallel R^* \quad &\to^* \quad \nu b_1.\overline{b_1}\langle c\rangle \parallel S^* \parallel R^* \\ &\to^* \quad \nu b_1.\overline{b_1}\langle c\rangle \parallel \nu b_2.\overline{b_2}\langle c\rangle \parallel S^* \parallel R^* \\ &\to^* \quad \nu b_1.\overline{b_1}\langle c\rangle \parallel \nu b_2.\overline{b_2}\langle c\rangle \parallel \cdots \parallel \nu b_n.\overline{b_n}\langle c\rangle \parallel S^* \parallel R^* \end{aligned}$$

Thus $S^* \parallel R^*$ is:

- depth bounded: every reachable term has maximum nested-restriction depth of 1 (equivalently, every subterm is in the scope of at most 1 restriction).
- name unbounded: for each $n \geq 1$, a term is reachable that uses n channels $(b_1, \cdots, b_n)$ concurrently.

Example: Depth-unbounded

Let $\theta = a(x).\nu c.(\overline{c}\langle x\rangle \parallel \overline{a}\langle c\rangle)$.

$$
\begin{array}{ll}
 & \overline{a}\langle c_0\rangle \parallel \theta^* \\
\equiv & \overline{a}\langle c_0\rangle \parallel a(x).\nu c_1.(\overline{c_1}\langle x\rangle \parallel \overline{a}\langle c_1\rangle) \parallel \theta^* \\
\to & \nu c_1.(\overline{c_1}\langle c_0\rangle \parallel \overline{a}\langle c_1\rangle \parallel \theta^*) \\
\equiv & \nu c_1.\big(\overline{c_1}\langle c_0\rangle \parallel \overline{a}\langle c_1\rangle \parallel a(x).\nu c_2.(\overline{c_2}\langle x\rangle \parallel \overline{a}\langle c_2\rangle) \parallel \theta^*\big) \\
\to & \nu c_1.\big(\overline{c_1}\langle c_0\rangle \parallel \nu c_2.(\overline{c_2}\langle c_1\rangle \parallel \overline{a}\langle c_2\rangle \parallel \theta^*)\big) \\
\to^* & \nu c_1.\Big(\overline{c_1}\langle c_0\rangle \parallel \nu c_2.\big(\overline{c_2}\langle c_1\rangle \parallel \cdots \parallel \nu c_n.(\overline{c_n}\langle c_{n-2}\rangle \parallel \overline{a}\langle c_n\rangle \parallel \theta^*)\big)\Big)
\end{array}
$$

- The subterm $\overline{a}\langle c_n\rangle$ is in the scope of n restrictions.
- For each $n \geq 1$, a term with nested restriction of depth n is reachable.

Definition: Depth-bounded (DB) Pi-Terms

Nested depth of restriction of P, $nest_\nu(P)$.

$$\begin{aligned} nest_\nu(S) &:= 0 \text{ for sequential } S \\ nest_\nu(P_1 \parallel P_2) &:= \max(nest_\nu(P_1), nest_\nu(P_2)) \\ nest_\nu(\nu a.P) &:= 1 + nest_\nu(P). \end{aligned}$$

Define $depth(Q) := \min\{nest_\nu(Q') \mid Q \equiv Q'\}$.

Example. $\underbrace{\nu a.\nu b.\nu c.\big(a(x) \parallel \bar{b}\langle c\rangle \parallel c(y)\big)}_{P} \equiv \underbrace{\nu a.a(x) \parallel \nu c.\big((\nu b.\bar{b}\langle c\rangle) \parallel c(y)\big)}_{Q}$

$nest_\nu(P) = 3$; $nest_\nu(Q) = 2$; but $depth(P) = depth(Q) = 2$.

A π-term P is depth bounded if $\exists k \geq 0 \,.\, \forall P' \in \mathrm{Reach}(P) \,.\, depth(P') \leq k$.

Depth-bounded (DB) terms are a semantic fragment of π-calculus – many known fragments[1] with decidable analyses are subfragments of DB.

[1] Finite control, bounded, finite handler, structurally stationary, restriction-free, etc.

Theorem (Meyer 2008) DB π-terms form a well-structured transition system (WSTS). Thus coverability and termination are decidable.

The set DB $:= \bigcup_{i \in \omega}$ DB(i) is highly expressive:
- Terms of DB(0) can represent Petri nets.
- Reachability is undecidable for DB(1).

It is decidable, given $k \geq 0$ and a term P, whether $P \in$ DB(k); but has non-PR lower bound (Hüchting et al. CONCUR'14).

Goal. DB is an important resource-bounded fragment for analysing concurrent and distributed computation, yet we scarcely understand it.

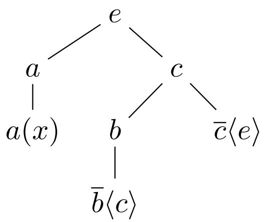
1. Instead of depth bound (a number), we define an expressive subfragment of DB using a richer structure (finite forests): tree-compatible pi-terms
2. Develop static analysis of DB π-terms: a feasibly decidable type system for DB / tree-compatible terms.

Abstract Syntax Tree AST(P)

- Internal nodes are labelled by active restricted names of P.
- Leaves are labelled by sequential subterms of P.

Example. Let $P = \nu e.\Big(\nu a.a(x) \parallel \Big(\nu c.\big((\nu b.\bar{b}\langle c\rangle) \parallel \bar{c}\langle e\rangle\big)\Big)\Big)$

AST(P) =



$\mathcal{T}$ =

Fix a forest $(\mathcal{T}, \prec)$; write $<$ for $\prec^+$. Let $\phi : \mathcal{N} \to \mathcal{T}$. Say P ϕ-matches $\mathcal{T}$ if for each trace $\langle x_1, \cdots, x_n, S\rangle$ in AST(P), $\phi(x_1) < \cdots < \phi(x_n)$ in $\mathcal{T}$.

Example. P id-matches $\mathcal{T}$.

> Lemma. P is DB iff there exists a finite forest $\mathcal{T}$ and $\phi : \mathcal{N} \to \mathcal{T}$ such that for all $Q \in \mathrm{Reach}(P)$, some congruent of Q ϕ-matches $\mathcal{T}$.

Simple types generated by finite forest $\mathcal{T}$

Fix a finite forest $(\mathcal{T}, \prec)$, writing $<$ for $\prec^+$.
Recall: P ϕ-matches $\mathcal{T}$ just if for each trace $\langle x_1, \cdots, x_n, S\rangle$ in AST(P), $\phi(x_1) < \cdots < \phi(x_n)$ in $\mathcal{T}$.

Approach. Define ϕ statically, using simple types.

Simple types generated by $\mathcal{T}$ (or just types), are of the form

$$\mathbb{T}_{\mathcal{T}} \ni \tau \ := \ t \ | \ t[\tau]$$

where $t \in \mathcal{T}$ is called a base type. Cf. sorts (Milner 1993).

1. A name of type $t \in \mathcal{T}$ can only be used as a message.
2. A name of type $t[\tau]$ is used as a channel to transmit a name of type τ.

Define $\mathrm{base}(t[\tau]) := t$ and $\mathrm{base}(t) := t$.

A $\mathcal{T}$-annotated term is just a π-term except restrictions take the form $\nu x : \tau.P$. The semantics is the same, except type annotations are preserved when a name is duplicated or renamed by structural congruence.

DEF. A $\mathcal{T}$-annotated P is $\mathcal{T}$-compatible if P has a congruent that matches $\mathcal{T}$.

Lemma (Canonical Witness for $\mathcal{T}$-Compatibility)

There is a transformation Φ on $\mathcal{T}$-annotated π-terms such that for all P, (i) $\Phi(P)$ matches $\mathcal{T}$, and (ii) $\Phi(P) \equiv P$ iff P is $\mathcal{T}$-compatible.

Observation. If every $Q \in \mathrm{Reach}(P)$ is $\mathcal{T}$-compatible, then P is DB.

Goal. Develop a type system such that typability implies DB.

S.T.P. (i) Subject Reduction: Let $P \to P'$. If P is typable, so is P'.
(ii) Reduction of typable terms preserves $\mathcal{T}$-compatiblity.

Overview of the Type System

Typing judgement: $\Gamma \vdash_{\mathcal{T}} P$, where Γ is a set of $\mathcal{T}$-annotated names, $x : \tau$.

Salient features

1. Typing is defined on normal forms $P \in \mathcal{P}_{\mathsf{nf}}$, where $X = \{x_1 : \tau_1, \cdots, x_n : \tau_n\}$

$$\mathcal{P}_{\mathsf{nf}} \ni N ::= \nu X. \prod_{i \in I} A_i$$
$$A ::= \sum_{i \in I} \pi_i.N_i \mid \Big(\sum_{i \in I} \pi_i.N_i\Big)^*$$

2. The typing rules guarantee that reduction preserves $\mathcal{T}$-compatibility, a property of congruence classes.
3. While typing rules are compositional, the parameter $\mathcal{T}$ is "global".
4. Type inference: If $\mathcal{T}$ is unknown, type checking generates a constraint system which is polytime solvable.

How to Cheat Scope Extrusion

Want to prove: Assume $P \to P'$ (and P typable). If P is $\mathcal{T}$-compatible, so is P'.

Problem: Scope extrusion can break $\mathcal{T}$-compatibility. E.g. When the scope of νb is extruded in the following communication:

$$(\nu b : t_b.\overline{a}\langle b\rangle.S) \parallel (\nu c : t_c.a(x).R) \;\to\; \nu b : t_b.(S \parallel \nu c : t_c.R[b/x])$$

the reductum may not match $\mathcal{T}$!

Key idea

- In some situations, the effect of scope extrusion can be achieved by migrating the (substituted) receiver, $R[b/x]$, of the synchronising pair, so as to enter the scope of the restriction operator νb. E.g.

$$(\nu b.\overline{a}\langle b\rangle.S) \parallel a(x).R \;\to\; (\nu b.S \parallel R[b/x]) \parallel \mathbf{0}$$

- The type system is designed to capture these situations.

Migration of the Receiver Is Not Always Possible

Migration of the Receiver:

$$(\nu b.\overline{a}\langle b\rangle.S) \parallel a(x).R \;\rightarrow\; (\nu b.S \parallel R[b/x]) \parallel \mathbf{0}$$

However it is not always possible / sound. For example:

$$(\nu b.\overline{a}\langle b\rangle.S) \parallel (\nu c.a(x).R(x,c)) \;\not\rightarrow\; (\nu b.S \parallel R(x,c)[b/x]) \parallel (\nu c.\mathbf{0})$$

The Typing System

$$\frac{\begin{array}{c}\forall i \in I.\, \Gamma, X \vdash_{\mathcal{T}} A_i \\ \forall i \in I.\, \forall x : \tau_x \in X.\, x \lhd_P i \implies \mathrm{base}(\Gamma(\mathrm{fn}(A_i))) < \mathrm{base}(\tau_x)\end{array}}{\Gamma \vdash_{\mathcal{T}} \nu X.\, \prod_{i \in I} A_i} \textsc{Par}$$

$$\frac{\forall i \in I.\, \Gamma \vdash_{\mathcal{T}} \pi_i.P_i}{\Gamma \vdash_{\mathcal{T}} \sum_{i \in I} \pi_i.P_i} \textsc{Choice} \qquad \frac{\Gamma \vdash_{\mathcal{T}} A}{\Gamma \vdash_{\mathcal{T}} A^*} \textsc{Repl} \qquad \frac{\Gamma \vdash_{\mathcal{T}} A}{\Gamma \vdash_{\mathcal{T}} \boldsymbol{\tau}.A} \textsc{Tau}$$

$$\frac{a : t_a[\tau_b] \in \Gamma \qquad b : \tau_b \in \Gamma \qquad \Gamma \vdash_{\mathcal{T}} Q}{\Gamma \vdash_{\mathcal{T}} \overline{a}\langle b \rangle.Q} \textsc{Out}$$

$$\frac{\begin{array}{c} a : t_a[\tau_x] \in \Gamma \qquad \Gamma, x : \tau_x \vdash_{\mathcal{T}} P \\ \mathrm{base}(\tau_x) \leq t_a \vee \big(\forall i \in I.\ \mathrm{Mig}_{a(x).P}(i) \implies \mathrm{base}(\Gamma(\mathrm{fn}(A_i) \setminus \{a\})) < t_a\big)\end{array}}{\Gamma \vdash_{\mathcal{T}} a(x).\nu X.\, \prod_{i \in I} A_i} \textsc{In}$$

Rule Par

$$\frac{\forall i \in I.\, \Gamma, X \vdash_{\mathcal{T}} A_i \qquad \forall i \in I.\, \forall (x : \tau_x) \in X.\, \big(x \lhd_P i \implies \mathrm{base}(\Gamma(\mathrm{fn}(A_i))) < \mathrm{base}(\tau_x)\big)}{\Gamma \vdash_{\mathcal{T}} \underbrace{\nu X.\, \prod_{i \in I} A_i}_{P}} \;\textsc{Par}$$

Recall: Normal form P has shape $\nu X.\, \prod_{i \in I} A_i$.

Say A_i is linked to A_j in P, written $i \leftrightarrow_P j$, if $\exists (x : \tau) \in X \,.\, x \in \mathrm{fn}(A_i) \cap \mathrm{fn}(A_j)$.

Define the tied-to relation, $\frown_P$, as the transitive closure of $\leftrightarrow_P$.

Say that a name $y \in X$ is tied to A_i in P, written $y \lhd_P i$, if $\exists j \in I.\, y \in \mathrm{fn}(A_j) \wedge j \frown_P i$.

Other Versions of Typing Systems: $\mathcal{T}$ with multiplicities, to model replication more accurately.

Rule In

$$\frac{a : t_a[\tau_x] \in \Gamma \qquad \Gamma, x : \tau_x \vdash_{\mathcal{T}} P \\ \mathrm{base}(\tau_x) \leq t_a \vee \big(\forall i \in I.\ \mathrm{Mig}_{a(x).P}(i) \implies \mathrm{base}(\Gamma(\mathrm{fn}(A_i) \setminus \{a\})) < t_a\big)}{\Gamma \vdash_{\mathcal{T}} a(x).\nu X. \prod_{i \in I} A_i} \textsc{In}$$

Given an input-prefixed normal form $a(y).P$ where $P = \nu X. \prod_{i \in I} A_i$, we say that A_i is migratable in $a(y).P$, written $\mathrm{Mig}_{a(y).P}(i)$, if x is tied to A_i in $\nu X. \prod_{i \in I} A_i$.

Soundness

Say Γ is P-safe if for each $x \in \mathrm{fn}(P)$ and $(y : \tau) \in \mathrm{bn}_\nu(P)$, $\mathrm{base}(\Gamma(x)) < \mathrm{base}(\tau)$.

Lemma (Subject Reduction)

Let $P, Q \in \mathcal{P}_{\mathsf{nf}}$ and Γ be a P-safe environment. If $\Gamma \vdash_{\mathcal{T}} P$ and $P \to Q$ then $\Gamma \vdash_{\mathcal{T}} Q$.

Say P is $\mathcal{T}$-shaped if all its subterms are $\mathcal{T}$-compatible.

Theorem (Invariance of $\mathcal{T}$-shapeness)

Let $P, Q \in \mathcal{P}_{\mathsf{nf}}$ with $P \to Q$ and, Γ be a P-safe environment such that $\Gamma \vdash_{\mathcal{T}} P$. Then, if P is $\mathcal{T}$-shaped, so is Q.

Hence, if $\emptyset \vdash_{\mathcal{T}} P$ and $\mathcal{T}$-compatible, then P is DB.

Type Inference

The type system generates two types of constraints on base types.

1. Equality between types, inducing equality between base types.
2. Ancestor relation, $<$, between base types.

Lemma (Feasible Type Inference)

There is an (polytime) algorithm that decides $\textsc{Typability}$*: Given* $P \in \mathcal{P}_{\mathsf{nf}}$*, are there finite forest* $\mathcal{T}$ *and* P*-safe* Γ *such that* $\Gamma \vdash_{\mathcal{T}} P$*?*

Example

Take normal form $P = \nu a\, b\, c.(A^* \parallel \overline{a}\langle c\rangle)$ where

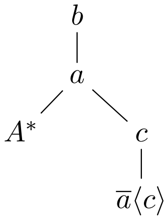
$$A = a(x).\nu d.\big(\overline{a}\langle d\rangle \parallel \overline{b}\langle x\rangle\big)$$

Type inference yields:

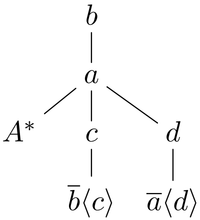
- $\mathcal{T}$ satisfying constraints: $t_b < t_a < t$
- typing of names: $a : t_a[t]$, $b : t_b[t]$, $c : t$, $d : t$

satisfying $\emptyset \vdash_{\mathcal{T}} P$.

$\Phi(P) =$



$\rightarrow$



Nested-Data Class Memory Automata (NDCMA)

NDCMA (Cotton-Barratt, Murawski & O., LATA 2015) are a version of class memory automata whose data set is a finite-depth forest in which every non-leaf node has infinitely many children.

Here we view NDCMA as a transition system.

Theorem

Pi-terms that are typable and tree-compatible are equi-expressive with NDCMA:

1. *Given a typable pi-term P, there is a NDCMA $\mathcal{A}_P$ such that P and $\mathcal{A}_P$ are weakly bisimilar as transition systems.*
2. *The converse is also true: given a NDCMA $\mathcal{A}$ there is a typable pi-term $P_\mathcal{A}$ such that $\mathcal{A}$ and $P_\mathcal{A}$ are weakly bisimilar.*

Conclusion

1. We have developed a static analysis for depth-bounded π-terms, an important resource-bounded fragment of π-calculus.
2. We have constructed a sound type system for tree-compatible terms; type checking and type inference are polytime computable.
3. Pi-terms that are typable are equi-expressive with nested-data class memory automata (NDCMA).

Further Directions

1. Develop a more precise analysis of the DB fragment, using context-sensitivity and subtyping.
2. We plan to develop Soter Version 2.0, which will use a new abstraction based on the π-calculus as the intermediate model of computation.
3. Conjecture. Secrecy is decidable for depth-bounded protocols.