On Hierarchical Communication Topologies of Concurrent Message-passing Systems

Emanuele D'Osualdo[1] **Luke Ong**[2]

[1]Imperial College London
[2]University of Oxford

IFIP WG 1.6 Meeting, Oxford, 9 September 2017

Abstract. We introduce a new, expressive class of inductive invariants for concurrent systems (expressed in the π-calculus), called hierarchical; and a type system for proving a system hierarchical, feasibly. Hierarchical systems are of interest to algorithmic verification because they have decidable semantic properties. A key innovation are special rewrite rules that are shape-invariant.

Outline

1. Automatic analysis of concurrency: depth-bounded pi-calculus
2. Hierarchical systems and a decidable type system
3. Results: algorithmics and expressivity
4. Application 1: verification of cryptographic protocols
5. Conclusions and ongoing/future work

Goal: Automatic analysis of concurrent systems.

Challenging, because:

- Unbounded process creation.
- Message passing leads to dynamic reconfiguration of communication topology.
- Turing completeness: interesting verification problems are undecidable.

Motivation: Soter (2013), safety verification tool for Erlang

Soter applies abstract interpretation and counter abstraction to transform an input Erlang program to a CCS-like model, which is model-checked using a Petri-net coverability checker. http://mjolnir.cs.ox.ac.uk/soter

Limitation (imprecise abstraction): unboundedly many Erlang pids (**p**rocess **id**s) are abstracted into a bounded number of equivalence classes.

1. Soter cannot support analysis requiring precision of process identity.
2. Because mailboxes are merged under the abstraction, certain patterns of communication cannot be analysed accurately.

Solution. Use π-calculus to model pids by names – a more accurate model.

Question. Is there a pi-calculus fragment in which reasoning about process identity (and hence commmunication topology) is precise and decidable?

Review: Pi-calculus (Milner, Parrow & Walker 1992)

- models communications between processes that exchange messages along channels.

- Messages and channels are represented uniformly by names.
- Processes communicate by synchronising on a matching pair of send and receive terms:

  $\overline{a}\langle b\rangle.S$ - sends message b on channel a, then becomes S

  $a(x).R$ - can receive message m on channel a, then becomes $R[m/x]$.
- Restriction (or new name) operator:

  $\nu a.P$ - A fresh name is allocated, and its scope is P.

Syntax of π-terms:

$P := \nu x.P \mid P_1 \parallel P_2 \mid M \mid !M$	process / π-term
$M := \mathbf{0} \mid \pi.P \mid M + M$	choice
$\pi := \overline{a}\langle b\rangle \mid a(x) \mid \tau$	prefix

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

$$\begin{aligned} \nu a.\mathbf{0} &\equiv \mathbf{0} \\ \nu a.\nu b.P &\equiv \nu b.\nu a.P \\ !P &\equiv P \parallel !P && \text{Replication} \\ P \parallel \nu a.Q &\equiv \nu a.(P \parallel Q) \quad (\text{if } a \notin \mathrm{fn}(P)) && \text{Scope Extrusion} \end{aligned}$$

With mobiliy, guarded replication equivalent to recursion.

Reaction relation, $\rightarrow$, is the least compatible relation satisfying:

$$\begin{aligned} \left(\overline{a}\langle b\rangle.S + S'\right) \parallel \left(a(x).R + R'\right) &\rightarrow S \parallel R[b/x] && \text{(React)} \\ \tau.P + M &\rightarrow P && \text{(Tau)} \end{aligned}$$

$$S[s] := !s(x).(\nu d.\overline{x}\langle d \rangle)$$

$$C[s, m] := \overline{s}\langle m \rangle \parallel m(x).C[s, m]$$

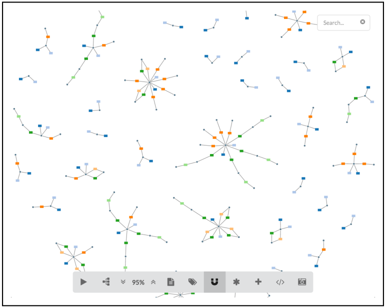
$$E[s] := !\tau.(\nu m.C[s, m])$$

Initial term: $\nu s.(S[s] \parallel E[s])$

The client/server example: evolution of communication topology

Stargazer π-calculus simulator

https://www.tcs.cs.tu-bs.de/group/dosualdo/stargazer/

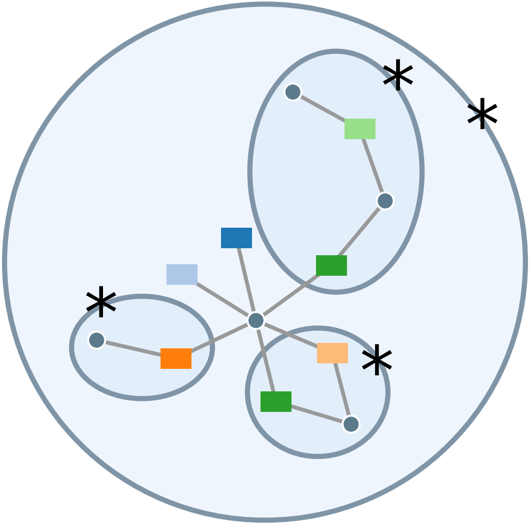


Correctness property: mailboxes have at most 1 message.

- Typical abstractions ignore topology: too imprecise to prove property.

- Alternatively prove the property using suitable inductive invariants.

The picture represents a set of configurations: each bubble can be cloned any number of times

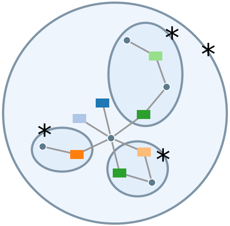


Client/server example: inductive invariant

An property (of terms) *Inv* is an inductive invariant of P just if

1. P satisfies *Inv*
2. *Inv* is closed under the transition relation.

Thus, an inductive invariant of P is a property of $Reach(P)$.



Want to prove: "each mailbox has at most 1 message" is inductive invariant of c/s system.

Problem: such (safety) properties are not inductive invariants of arbitrary π-terms.

Solution: there is a fragment of π-calculus for which such properties *are* invariants – depth-bounded fragment.

Depth boundedness (Roland Meyer 2008)

Def. A term is depth-bounded if there is some $d \geq 1$ such that all reachable terms from it have nested restriction depths $\leq d$.

E.g. $\nu a.\Big(\cdots(\nu b.\cdots(\nu c.\cdots)\cdots)\cdots\Big)$ has nested restriction depth ≥ 3.

Remarkably some semantic properties of depth-bounded terms are decidable:

- termination – an important liveness property
- coverability – weak form of reachability, hence safety.

Proof. Depth-bounded terms are a well-structured transition system (Finkel & Schnoebelen; Abdulla et al. Winner of 2017 CAV Award).

Depth boundedness is one of the most expressive fragments of π-calculus with decidable semantic properties.

Examples

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

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

Thus $!S \parallel !R$ is:

- depth bounded: every reachable term has nested-restriction depth of 1 (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 (i.e., $b_1, \cdots, b_n$) concurrently.

2. The client/server example is also depth-bounded.

Example: depth-unbounded

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

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

- 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.

Membership of depth boundedness is undecidable!

- Checking if a term is bounded in depth by a given number k is non-primitive-recursive. (Hütchting & Meyer 2014)

- We want a more structured measure for resources. Our approach: *trees* rather than *numbers* (for depth), leading to hierarchical systems.

Key contributions: (1) hierarchical systems have decidable semantic properties, (2) a (feasibly) decidable type system for proving a system hierarchical.

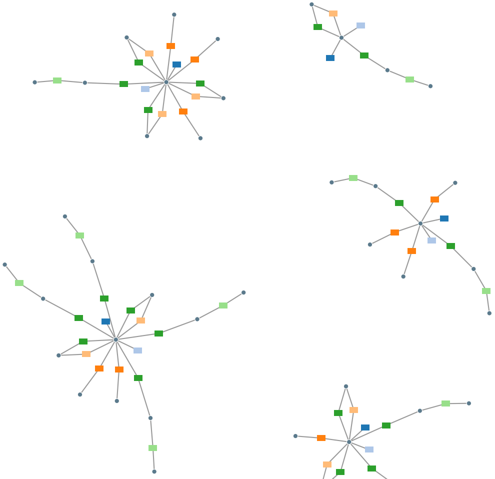
Outline

1. Automatic analysis of concurrency: depth-bounded pi-calculus
2. Hierarchical systems and a decidable type system
3. Results: algorithmics and expressivity
4. Application 1: verification of cryptographic protocols
5. Conclusions and ongoing/future work

Outline

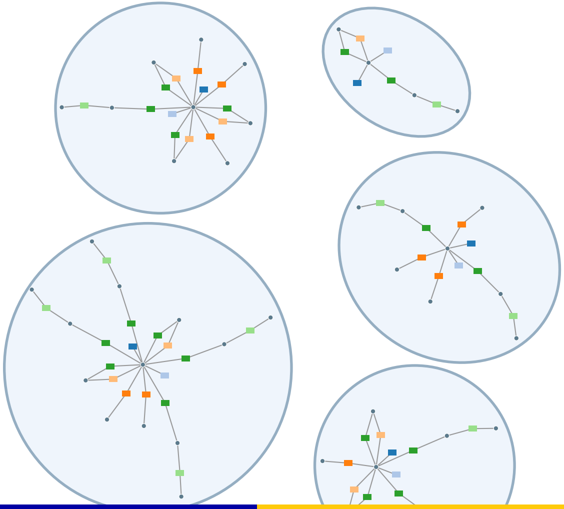
1. Automatic analysis of concurrency: depth-bounded pi-calculus
2. Hierarchical systems and a decidable type system
3. Results: algorithmics and expressivity
4. Application 1: verification of cryptographic protocols
5. Conclusions and ongoing/future work

Most naturally-occurring systems are (by design) organisable into a tree-shaped hierarchy, whose ordering intuitively means “nominal knowledge”.



Hierarchical systems: some intuition

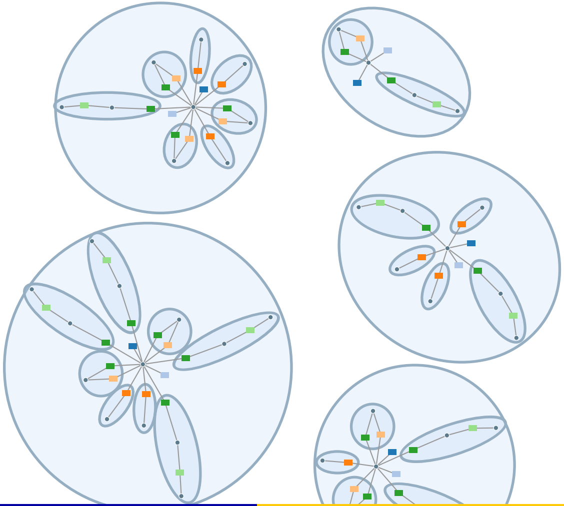
Most naturally-occurring systems are (by design) organisable into a tree-shaped hierarchy, whose ordering intuitively means "nominal knowledge".



server

Hierarchical systems: some intuition

Most naturally-occurring systems are (by design) organisable into a tree-shaped hierarchy, whose ordering intuitively means “nominal knowledge”.

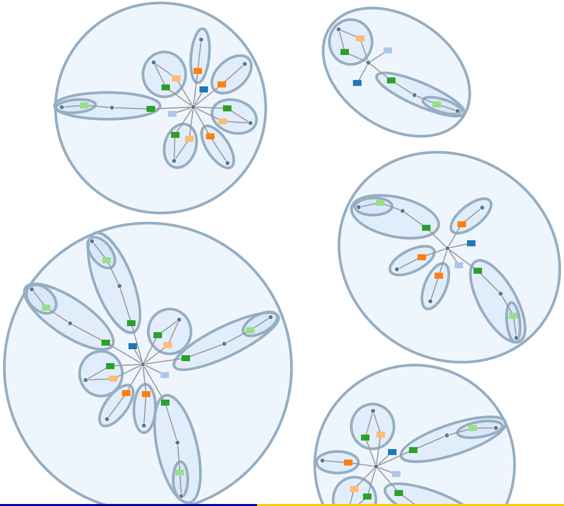


server

mailb

Hierarchical systems: some intuition

Most naturally-occurring systems are (by design) organisable into a tree-shaped hierarchy, whose ordering intuitively means “nominal knowledge”.

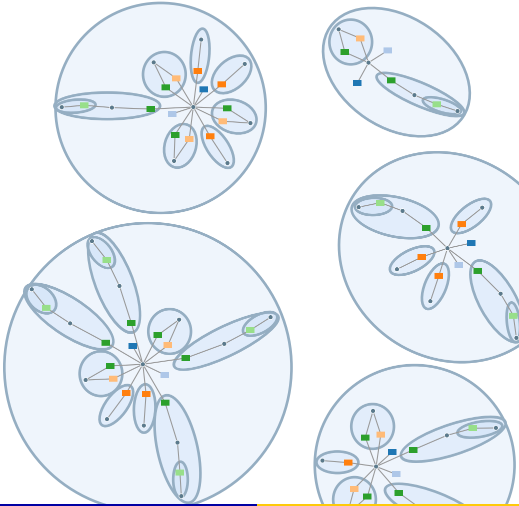


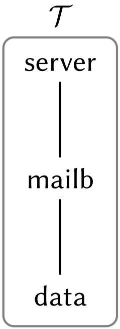
server

mailb

data

Most naturally-occurring systems are (by design) organisable into a tree-shaped hierarchy, whose ordering intuitively means “nominal knowledge”.





Fix a set of base types; build up channel types.

$$\mathsf{S}[s] := !s(x).(\nu(d : \mathsf{data}).\overline{x}\langle d\rangle)$$

$$\mathsf{C}[s, m] := \overline{s}\langle m\rangle \parallel m(x).\mathsf{C}[s, m]$$

$$\mathsf{E}[s] := !\tau.(\nu(m : \mathsf{mailb} \qquad).\mathsf{C}[s, m])$$

Initial term: $\nu(s : \mathsf{server} \qquad).(\mathsf{S}[s] \parallel \mathsf{E}[s])$

---

d : data = d has *base type* data

m : mailb[data] = m has *channel type* that can pass messages of type data

Fix a set of base types; build up channel types.

$$\mathsf{S}[s] := \,!s(x).(\nu(d:\mathsf{data}).\overline{x}\langle d\rangle)$$

$$\mathsf{C}[s,m] := \overline{s}\langle m\rangle \parallel m(x).\mathsf{C}[s,m]$$

$$\mathsf{E}[s] := \,!\tau.(\nu(m:\mathsf{mailb}[\mathsf{data}]).\mathsf{C}[s,m])$$

Initial term: $\nu(s:\mathsf{server}[\mathsf{mailb}[\mathsf{data}]]).(\mathsf{S}[s] \parallel \mathsf{E}[s])$

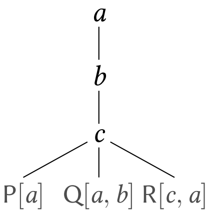
$d : \mathsf{data}$ = d has *base type* data

$m : \mathsf{mailb}[\mathsf{data}]$ = m has *channel type* that can pass messages of type data

$\mathcal{T}$-shapedness: a property of congruence class

1. View a π-term as a labelled forest.

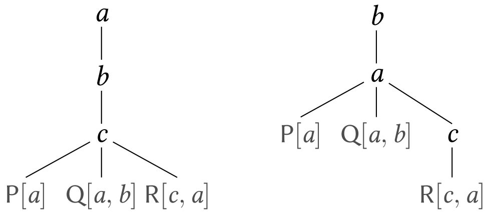
$$\nu a.\nu b.\nu c.(\mathrm{P}[a] \parallel \mathrm{Q}[a, b] \parallel \mathrm{R}[c, a])$$



$\mathcal{T}$-shapedness: a property of congruence class

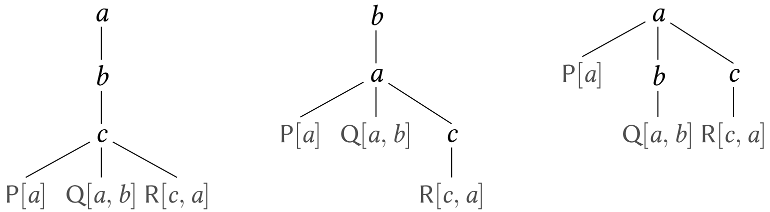
1. View a π-term as a labelled forest.

$$\nu a.\nu b.\nu c.(\mathrm{P}[a] \parallel \mathrm{Q}[a, b] \parallel \mathrm{R}[c, a]) \equiv \nu b.\nu a.(\mathrm{P}[a] \parallel \mathrm{Q}[a, b] \parallel \nu c.\mathrm{R}[c, a])$$



1. View a π-term as a labelled forest.

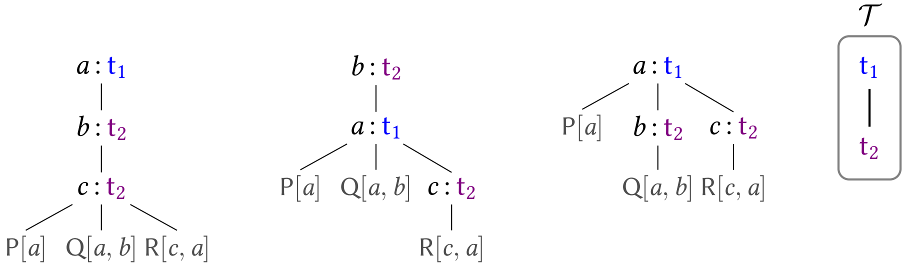
$$\nu a.\nu b.\nu c.(\mathrm{P}[a] \parallel \mathrm{Q}[a, b] \parallel \mathrm{R}[c, a]) \equiv \nu a.(\mathrm{P}[a] \parallel \nu b.\mathrm{Q}[a, b] \parallel \nu c.\mathrm{R}[c, a])$$



$\mathcal{T}$-shapedness: a property of congruence class

1. View a π-term as a labelled forest.
2. Assign types to (active) names.

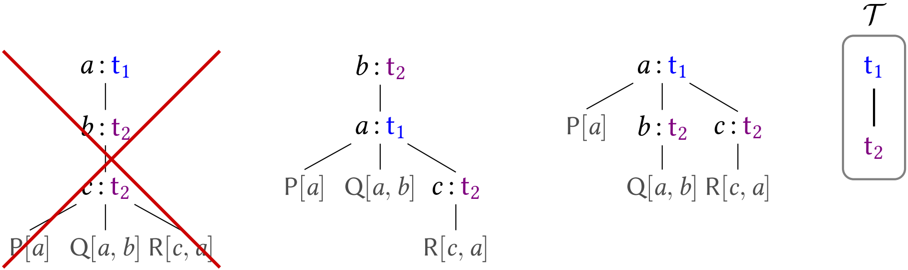
$$\nu(a : t_1).\nu(b : t_2).\nu(c : t_2).(\mathsf{P}[a] \parallel \mathsf{Q}[a, b] \parallel \mathsf{R}[c, a])$$



$\mathcal{T}$-shapedness: a property of congruence class

1. View a π-term as a labelled forest.
2. Assign types to (active) names.
3. A term respects (tree) $\mathcal{T}$ if each trace (of types) is a chain in poset $\mathcal{T}$.

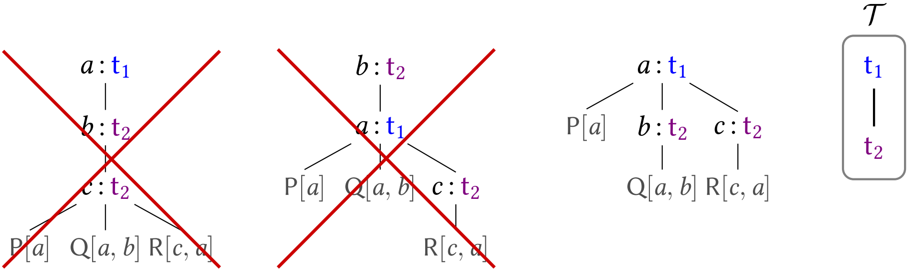
$$\nu(a : \mathsf{t}_1).\nu(b : \mathsf{t}_2).\nu(c : \mathsf{t}_2).(\mathrm{P}[a] \parallel \mathrm{Q}[a, b] \parallel \mathrm{R}[c, a])$$



$\mathcal{T}$-shapedness: a property of congruence class

1. View a π-term as a labelled forest.
2. Assign types to (active) names.
3. A term respects (tree) $\mathcal{T}$ if each trace (of types) is a chain in poset $\mathcal{T}$.

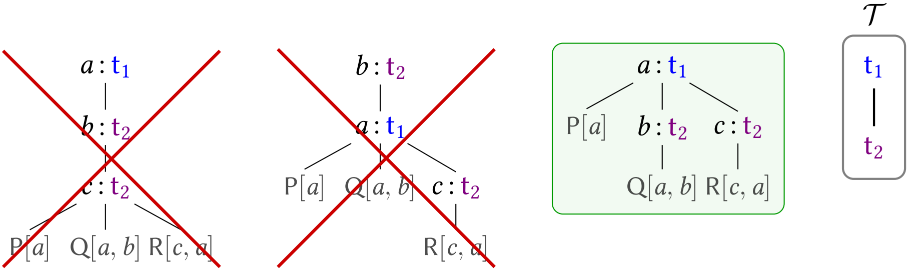
$$\nu(a:\mathrm{t}_1).\nu(b:\mathrm{t}_2).\nu(c:\mathrm{t}_2).(\mathrm{P}[a] \parallel \mathrm{Q}[a,b] \parallel \mathrm{R}[c,a])$$



$\mathcal{T}$-shapedness: a property of congruence class

1. View a π-term as a labelled forest.
2. Assign types to (active) names.
3. A term respects (tree) $\mathcal{T}$ if each trace (of types) is a chain in poset $\mathcal{T}$.

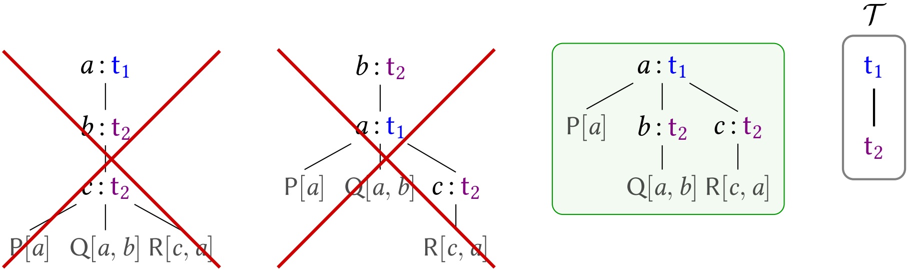
$$\nu(a : t_1).\nu(b : t_2).\nu(c : t_2).(P[a] \parallel Q[a, b] \parallel R[c, a])$$



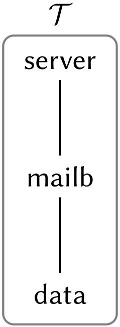
1. View a π-term as a labelled forest.
2. Assign types to (active) names.
3. A term respects (tree) $\mathcal{T}$ if each trace (of types) is a chain in poset $\mathcal{T}$.

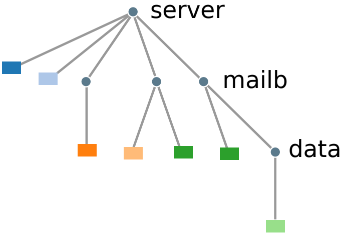
$\nu(a : \mathsf{t}_1).\nu(b : \mathsf{t}_2).\nu(c : \mathsf{t}_2).(\mathrm{P}[a] \parallel \mathrm{Q}[a, b] \parallel \mathrm{R}[c, a])$ is $\mathcal{T}$-shaped

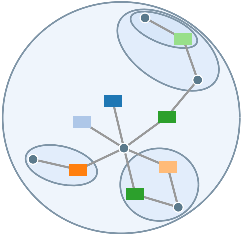
because at least one of its presentations respects $\mathcal{T}$



Example: client/server example is $\mathcal{T}$-shaped







Every reachable term is $\mathcal{T}$-shaped

(but note that the communication topology is not a tree)

Definition

A π-term P is hierarchical if

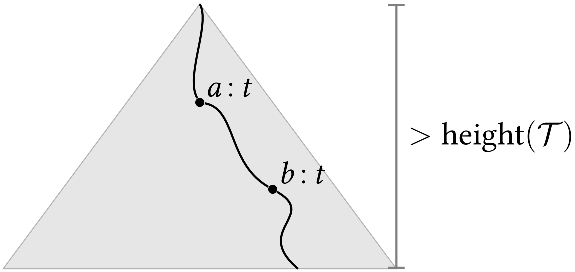
$$\exists \text{ finite-tree } \mathcal{T} \,.\, \forall Q \,.\, \big(P \to^* Q \implies Q \text{ is } \mathcal{T}\text{-shaped}\big)$$

Hierarchical := $\mathcal{T}$-shapedness is an invariant.

Non-hierarchical terms

There are terms for which $\mathcal{T}$-shapedness is not an invariant, for any finite $\mathcal{T}$.

If the term is not depth-bounded, one can reach forests of unbounded height



Proposition

Hierarchical $\subset$ depth-bounded

Problem: Membership of hierarchical is still undecidable.

Solution: But now we have a more structured measure (tree *vs* number), which we exploit to design a type system satisfying:

Theorem (Pre-inductive Invariant)

If P is typable (i.e. $\Gamma \vdash_{\mathcal{T}} P$ for some Γ) then

$$\big(P \text{ is } \mathcal{T}\text{-shaped} \quad \wedge \quad P \to Q\big) \implies Q \text{ is } \mathcal{T}\text{-shaped}$$

Hence, if P is typable and $\mathcal{T}$-shaped then P is hierarchical (i.e. all reachable terms of P are $\mathcal{T}$-shaped).

The type system

(All rules are shown here.)

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

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

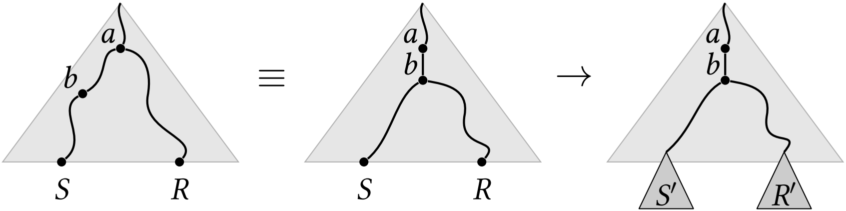
$$\frac{\forall i \in I.\ \Gamma \vdash_{\mathcal{T}} \pi_i.P_i}{\Gamma \vdash_{\mathcal{T}} \sum_{i \in I} \pi_i.P_i} \ \textsc{Choice} \qquad \frac{\Gamma \vdash_{\mathcal{T}} A}{\Gamma \vdash_{\mathcal{T}} !A} \ \textsc{Repl} \qquad \frac{\Gamma \vdash_{\mathcal{T}} P}{\Gamma \vdash_{\mathcal{T}}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

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

Given tree $\mathcal{T}$, the type system identifies terms (i) that are $\mathcal{T}$-shaped, and (ii) whose reduction preserves $\mathcal{T}$-shapedness.

Recall: standard π-calculus reductions assume scope-extrusion

$$S = \overline{a}\langle b\rangle.S' \qquad\qquad R = a(x).R'$$



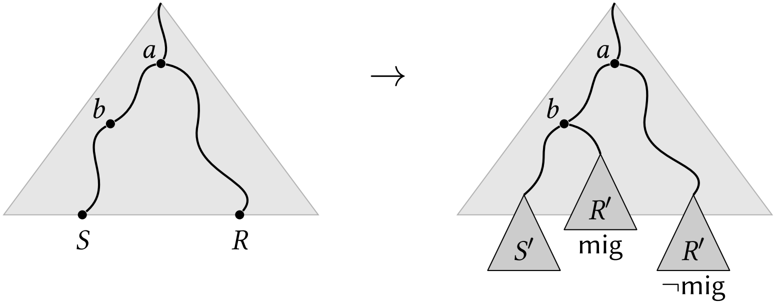
$$\nu a.\big((\nu b.S) \parallel R\big) \quad\equiv\quad \nu a.\nu b.\big(S \parallel R\big) \;\rightarrow\; \nu a.\nu b.(S' \parallel R'[\,b/x\,])$$

(Assume: $b \notin \mathrm{fn}(R)$.)

Key rewriting idea: (special) $\mathcal{T}$-shapedness-preserving reductions

$\mathcal{T}$-shaped reductions eschew scope extrusion; instead receiving term "extrudes" a migratable part to the sending term.

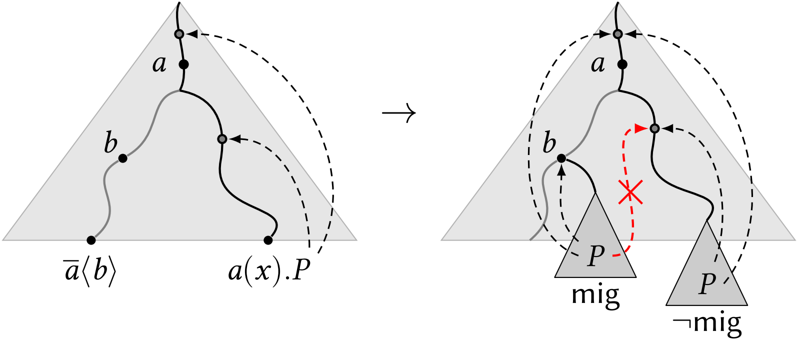
$$S = \overline{a}\langle b\rangle.S' \qquad R = a(x).(\,\underbrace{R'_{\mathsf{mig}}}_{\text{uses } x} \parallel R'_{\neg\mathsf{mig}}\,)$$



$$\nu a.\big((\nu b.S) \parallel R\big) \quad \to \quad \nu a.\big((\nu b.S' \parallel R'_{\mathsf{mig}}[\,b/x\,]) \parallel R'_{\neg\mathsf{mig}}\big)$$

$\mathcal{T}$, qua reaction context, $\nu a.\big((\nu b.[\,]) \parallel [\,]\big)$, is unchanged by the reduction.

$\mathcal{T}$-shaped reductions are valid provided the part of the receiving term R that uses x ("migratable") does not have names whose binder is in R.



Outline

1 Automatic analysis of concurrency: depth-bounded pi-calculus

2 Hierarchical systems and a decidable type system

3 Results: algorithmics and expressivity

4 Application 1: verification of cryptographic protocols

5 Conclusions and ongoing/future work

Soundness

Lemma (Subject reduction)

If $\Gamma \vdash_{\mathcal{T}} P$ and $P \to Q$, then $\Gamma \vdash_{\mathcal{T}} Q$

Theorem

If $\Gamma \vdash_{\mathcal{T}} P$ and P is $\mathcal{T}$-shaped $\implies$ P is hierarchical

Def. We say P is typably hierarchical just if for some $\mathcal{T}$

$$\Gamma \vdash_{\mathcal{T}} P \quad \text{and} \quad P \text{ is } \mathcal{T}\text{-shaped.}$$

Properties of typing

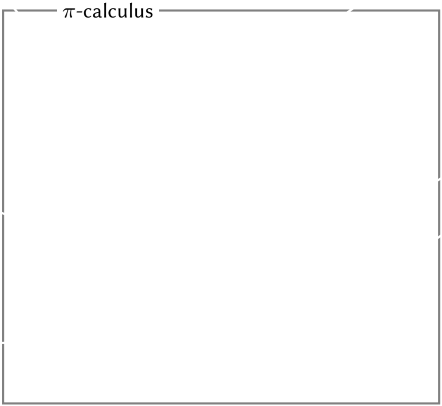
Theorem

1. **Type checking** *is decidable in P*
2. **Type inference** *is computable in NP.*

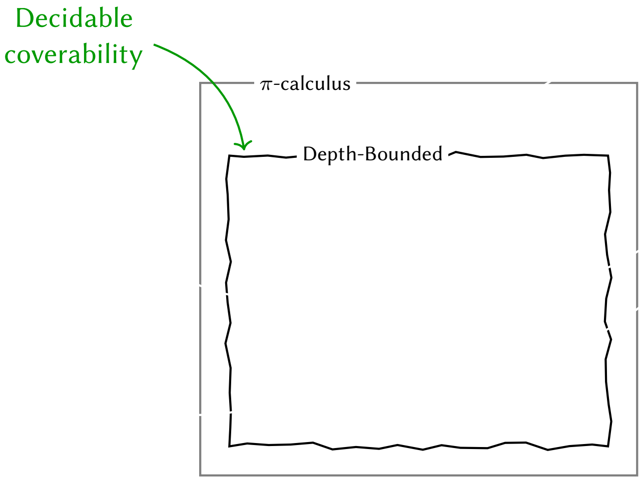
This is the first type system capable of inferring (shaped) properties of communication topologies.

Implementation available at `github.com/bordaigorl/jamesbound`

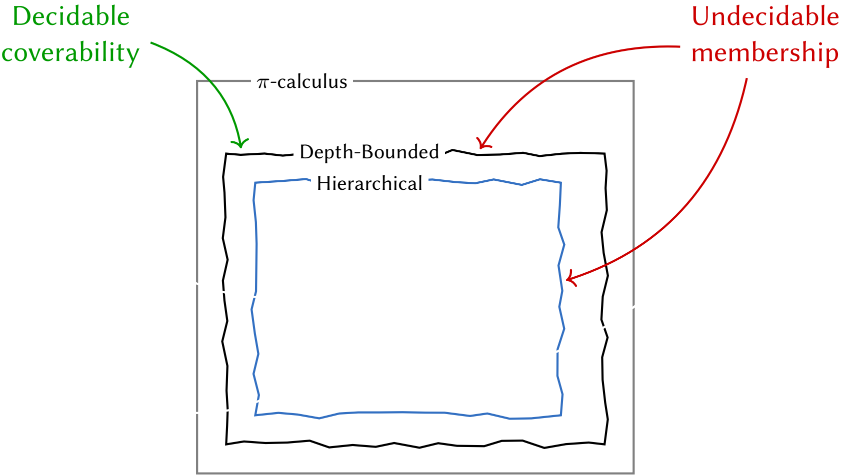
Expressivity



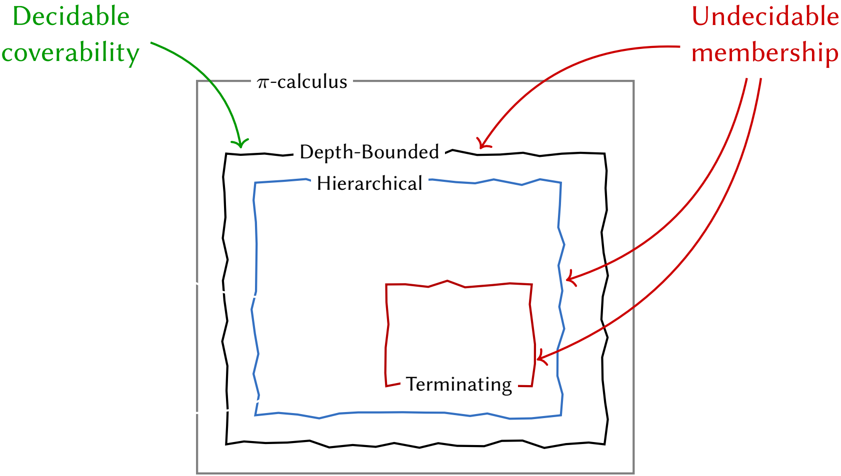
Expressivity



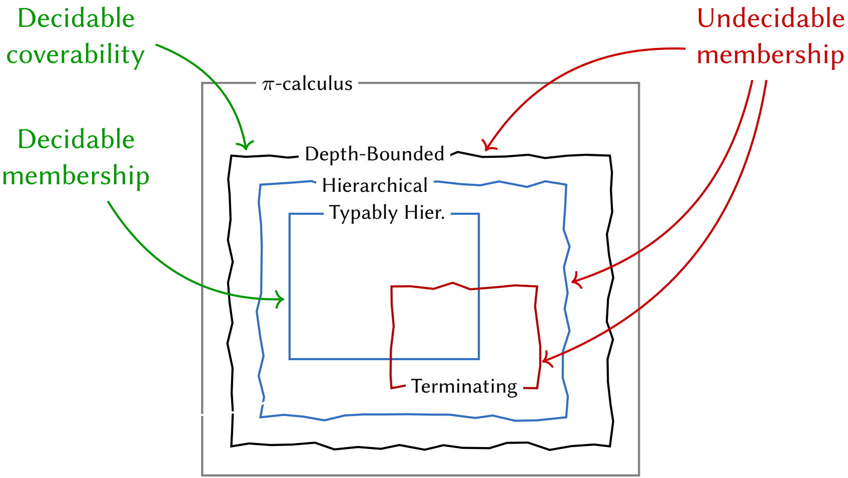
Expressivity



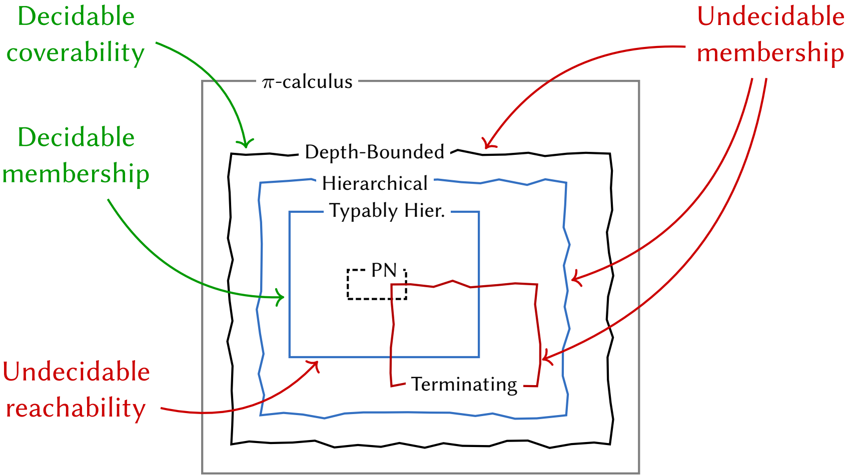
Expressivity



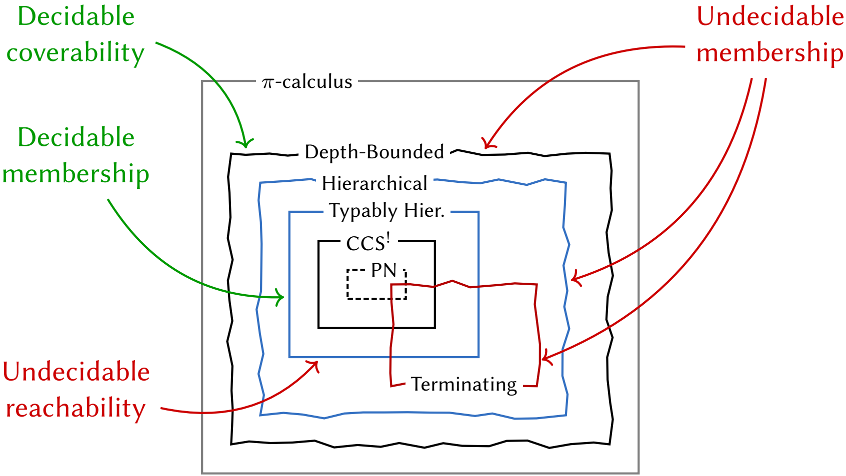
Expressivity



Expressivity



Expressivity



For details: see the ESOP 2016 paper

On Hierarchical Communication Topologies in the π-calculus

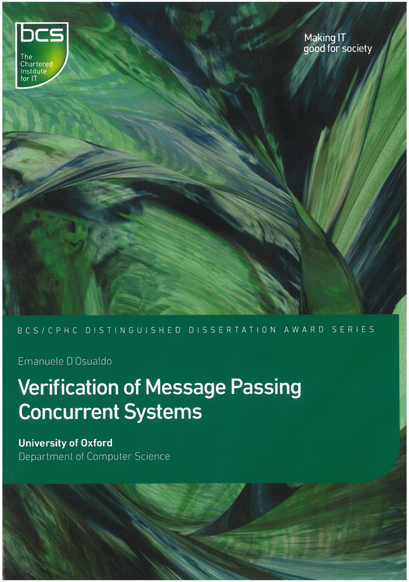
Emanuele D’Osualdo[1] and C.-H. Luke Ong[2]

[1] TU Kaiserslautern `dosualdo@cs.uni-kl.de`
[2] University of Oxford `lo@cs.ox.ac.uk`

Abstract. This paper is concerned with the shape invariants satisfied by the communication topology of π-terms, and the automatic inference of these invariants. A π-term P is *hierarchical* if there is a finite forest $\mathcal{T}$ such that the communication topology of every term reachable from P satisfies a $\mathcal{T}$-shaped invariant. We design a static analysis to prove a term hierarchical by means of a novel type system that enjoys decidable inference. The soundness proof of the type system employs a non-standard view of π-calculus reactions. The coverability problem for hierarchical terms is decidable. This is proved by showing that every hierarchical term is depth-bounded, an undecidable property introduced by R. Meyer. We thus obtain an expressive static fragment of the π-calculus with decidable safety verification problems.

Winner of 2016 BCS Distinguished Dissertation Award

For further details: see my coauthor/former student's Oxford DPhil dissertation:



Application to cryptographic protocol verification

See CSF17 paper:

Deciding Secrecy of Security Protocols for an Unbounded Number of Sessions: The Case of Depth-bounded Processes

Emanuele D'Osualdo
University of Kaiserslautern, Germany
dosualdo@cs.uni-kl.de

Luke Ong
University of Oxford, UK
lo@cs.ox.ac.uk

Alwen Tiu
Nanyang Technological University, Singapore
atiu@ntu.edu.sg

***Abstract*—We introduce a new class of security protocols with an unbounded number of sessions and unlimited fresh data for which the problem of secrecy is decidable. The only constraint we place on the class is a notion of *depth-boundedness*. Precisely we prove that, restricted to messages of up to a given size, secrecy is decidable for all depth-bounded processes. This decidable fragment of security protocols captures many real-world symmetric key protocols, including Needham-Schroeder Symmetric Key, Otway-Rees, and Yahalom.**

I. Introduction

Security protocols are distributed programs that are designed to achieve secure communications using cryptography. They are extensively deployed today to improve the security of

Several decidability results have been obtained by restricting the three sources of infinity identified above. Durgin et al. [2] proved that secrecy is DEXPTIME-complete when both the number of nonces and the size of messages are bounded. Rusinowitch and Turuani [5] and Comon-Lundh et al. [6] proved that nonsecrecy is NP-complete when the number of sessions is bounded. Of course, analysing a protocol for a fixed finite number of sessions does not prove secrecy.

A direction of investigation which has proved fruitful does not constrain the above sources of infinity *a priori*, but restricts the format of messages, so that the encrypted messages become *context explicit*. In a pioneering paper [7], Lowe considered

Outline

1. Automatic analysis of concurrency: depth-bounded pi-calculus
2. Hierarchical systems and a decidable type system
3. Results: algorithmics and expressivity
4. Application 1: verification of cryptographic protocols
5. Conclusions and ongoing/future work

Pi-calculus variants (Spi-Calculus and Applied Pi-Calculus) are widely used in reasoning about cryptographic protocols.

Secrecy Problem for Crytographic Protocol P

Given a secret M, can protocol P leak M?

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

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

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

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)

We (CSF 2017) give the first class of security protocols with an unbounded sessions and unlimited fresh data for which the problem of secrecy is decidable. The key constraint we place on the class is depth boundedness.

Outline

1. Automatic analysis of concurrency: depth-bounded pi-calculus
2. Hierarchical systems and a decidable type system
3. Results: algorithmics and expressivity
4. Application 1: verification of cryptographic protocols
5. Conclusions and ongoing/future work

Conclusions

- We define hierarchical systems.
- Hierarchical systems are expressive yet have decidable semantic properties (coverability & termination).
- We introduce a novel decidable type system that can prove a term hierarchical, in a feasible and sound (but incomplete) way.
- We give the first automatic inference of shape invariants of communication topologies; prototype implementation available.

Ongoing / future work:

- use typing failures to do smart abstractions (think: abstraction refinement)
- tune precision of the type system
- applications to
  - cryptographic protocol verification
  - concurrent heap-manipulating programs verification

Thank you for your attention!

Luke Ong
lo@cs.ox.ac.uk

Appendix

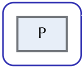
Outline

6 Coverability

7 Basic definitions

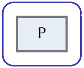
8 Soundness

Coverability

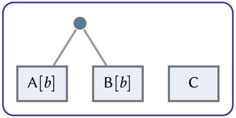


Init

Coverability

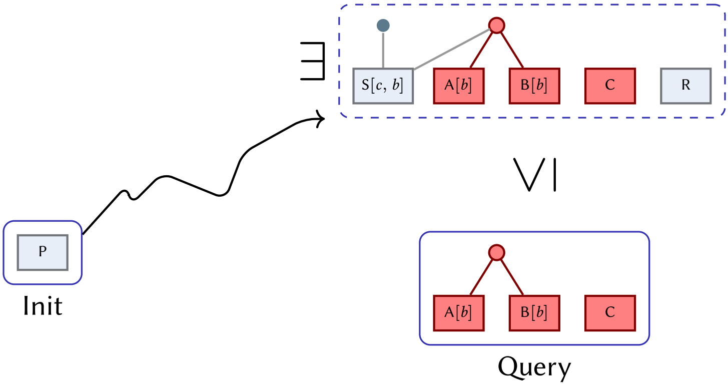


Init

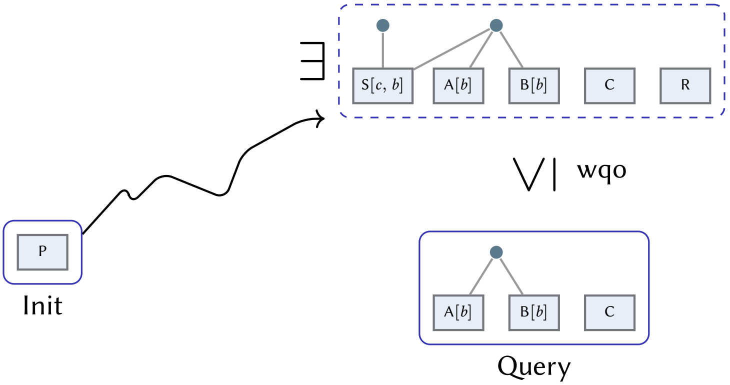


Query

Coverability



Coverability Decidable for depth bounded systems via WSTS



Outline

6 Coverability

7 Basic definitions

8 Soundness

Syntax:

$$
\begin{array}{rlr}
\mathcal{P} \ni P, Q ::= & \mathbf{0} \;\mid\; \nu x.P \;\mid\; P_1 \parallel P_2 \;\mid\; M \;\mid\; !M & \text{process} \\
M ::= & M + M \;\mid\; \pi.P & \text{choice} \\
\pi ::= & a(x) \;\mid\; \overline{a}\langle b \rangle \;\mid\; \boldsymbol{\tau} & \text{prefix}
\end{array}
$$

Normal form:

$$
\begin{array}{rl}
\mathcal{P}_{\mathsf{nf}} \ni N ::= & \nu x_1. \cdots \nu x_n.(A_1 \parallel \cdots \parallel A_m) \\
A ::= & \pi_1.N_1 + \cdots + \pi_n.N_n \;\mid\; !(\pi_1.N_1 + \cdots + \pi_n.N_n)
\end{array}
$$

Depth

The nesting of restrictions of a term is given by the function

$$\begin{aligned}
\mathrm{nest}_\nu(M) &:= \mathrm{nest}_\nu(!M) := \mathrm{nest}_\nu(\mathbf{0}) := 0 \\
\mathrm{nest}_\nu(\nu x.P) &:= 1 + \mathrm{nest}_\nu(P) \\
\mathrm{nest}_\nu(P \parallel Q) &:= \max(\mathrm{nest}_\nu(P), \mathrm{nest}_\nu(Q)).
\end{aligned}$$

The depth of a term is defined as the minimal nesting of restrictions in its congruence class:

$$\mathrm{depth}(P) := \min\{\mathrm{nest}_\nu(Q) \mid P \equiv Q\}$$

A term P is depth-bounded if there exists $k \geq 0$ such that for each $Q \in \mathrm{Reach}(P)$, $\mathrm{depth}(Q) \leq k$.

Outline

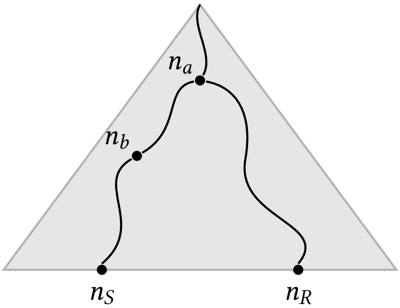
6 Coverability

7 Basic definitions

8 Soundness

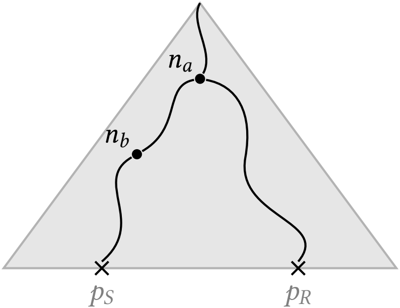
Soundness argument

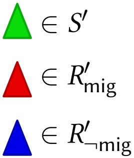
$$S = \overline{a}\langle b\rangle.S' \qquad R = a(x).(\underbrace{R'_{\mathrm{mig}}}_{\text{uses } x} \parallel R'_{\neg\mathrm{mig}})$$



Soundness argument

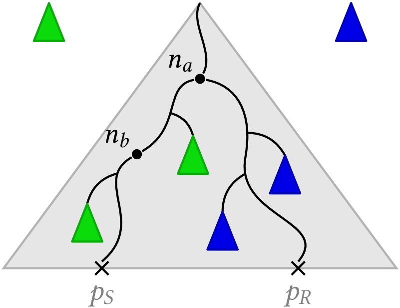
$$S = \overline{a}\langle b\rangle.S' \qquad R = a(x).(\underbrace{R'_{\mathrm{mig}}}_{\text{uses } x} \parallel R'_{\neg\mathrm{mig}})$$

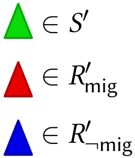




Soundness argument

$$S = \overline{a}\langle b\rangle.S' \qquad R = a(x).(\underbrace{R'_{\mathrm{mig}}}_{\text{uses } x} \parallel R'_{\neg\mathrm{mig}})$$





Soundness argument

$$S = \overline{a}\langle b \rangle . S' \qquad R = a(x).(\, \underbrace{R'_{\mathsf{mig}}}_{\text{uses } x} \parallel R'_{\neg\mathsf{mig}} \,)$$