On Hierarchical Communication Topologies in the π-calculus

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

[1]TU Kaiserslautern
[2]University of Oxford

CPDS 2016

The Problem

Goal: Automated analysis of *concurrent systems*.

Challenges:

- Unbounded process creation + message passing
- Dynamic reconfiguration of communication topology
- Turing completeness

The Client/Server example

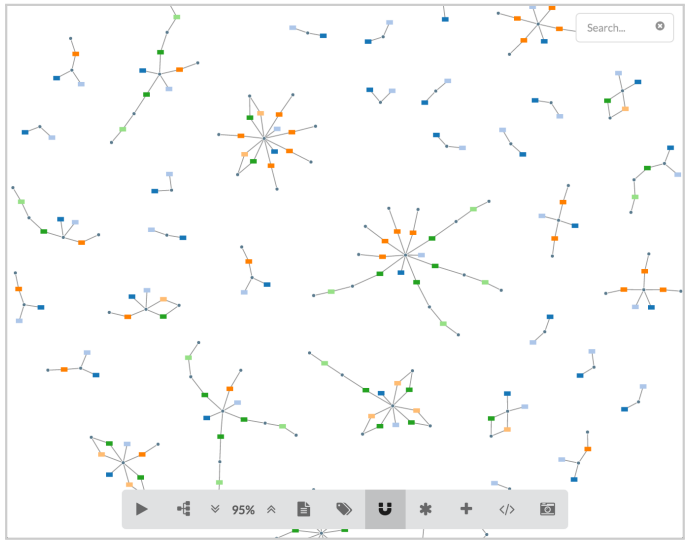
$$\mathsf{S}[s] := !s(x).(\boldsymbol{\nu} d.\overline{x}\langle d\rangle)$$

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

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

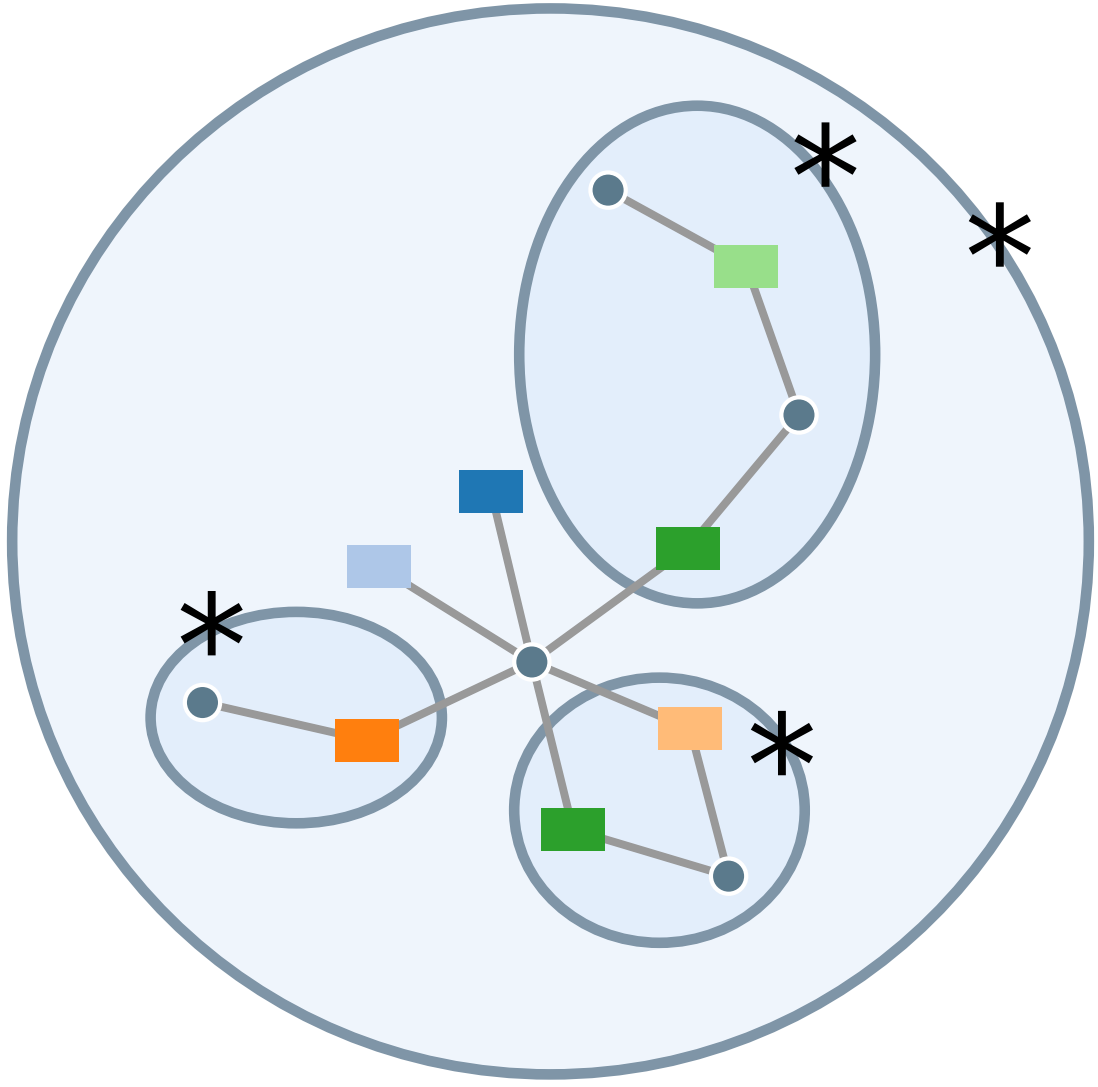
$$\boldsymbol{\nu} s.(\mathsf{S}[s] \parallel \mathsf{E}[s])$$

[Illustration of evolution of topology in the simulator]



- Property of interest: mailboxes are bounded by 1
- Typical abstractions ignore the topology: the property cannot be proven
- Alternatively we can prove the property using suitable inductive invariants

An inductive invariant



- The picture represents a set of configurations: each bubble can be cloned any number of times
- The invariant contains the initial configuration: instantiate once the outer bubble and zero times each inner bubble
- The invariant is closed under reductions
- The invariant satisfies the property: there is at most one message in each mailbox

Problem: such invariants do not always exist for arbitrary π-terms

Solution: there is a fragment of π-calculus for which such invariants always exist

Depth boundedness

If the simple paths of the reachable terms have bounded length, the initial term is **Depth Bounded**

If a system is Depth Bounded then some semantic properties are decidable

- termination
- coverability

One of the most expressive fragments of π-calculus to date

On Boundedness in Depth in the π-calculus
R. Meyer, 2008

Undecidability

Depth boundedness is undecidable!

And checking if a term is bounded in depth by a given k has *non primitive recursive* complexity

Undecidability

Depth boundedness is undecidable!

And checking if a term is bounded in depth by a given k has *non primitive recursive* complexity

We need more structure: Hierarchical systems.

Undecidability

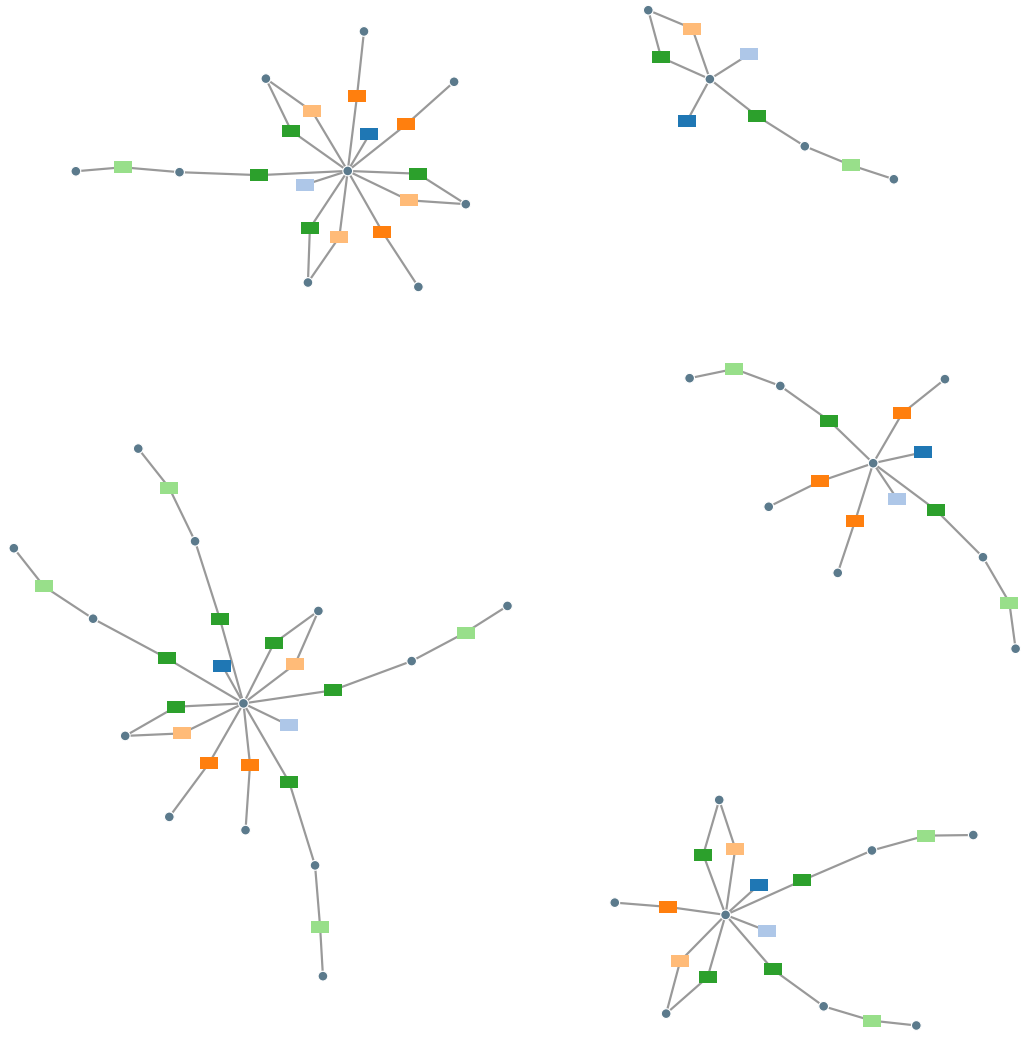
Depth boundedness is undecidable!

And checking if a term is bounded in depth by a given k has *non primitive recursive* complexity

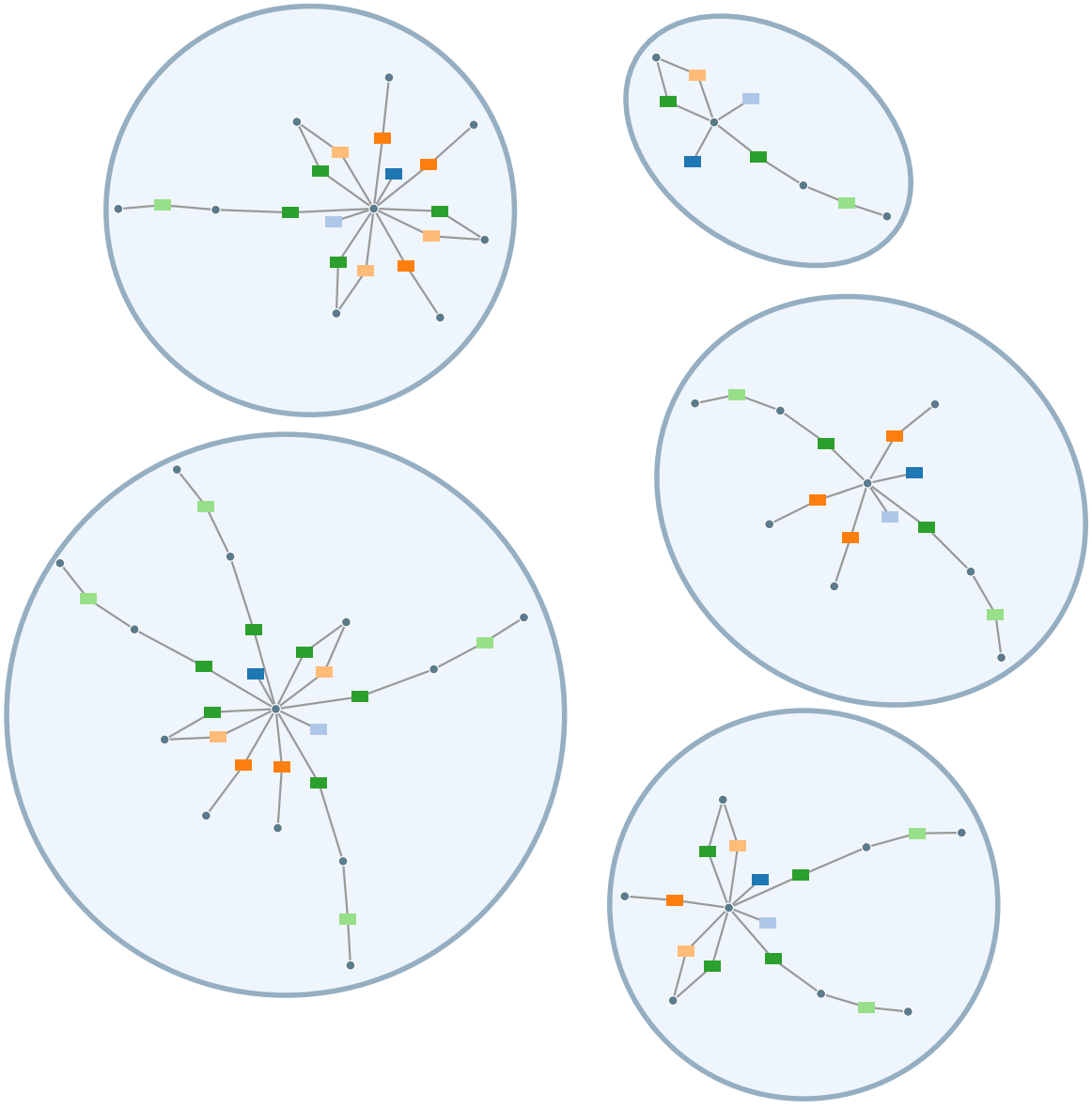
We need more structure: Hierarchical systems.

Key contribution: a type system to check/infer if a system is hierarchical.

Example Hierarchy

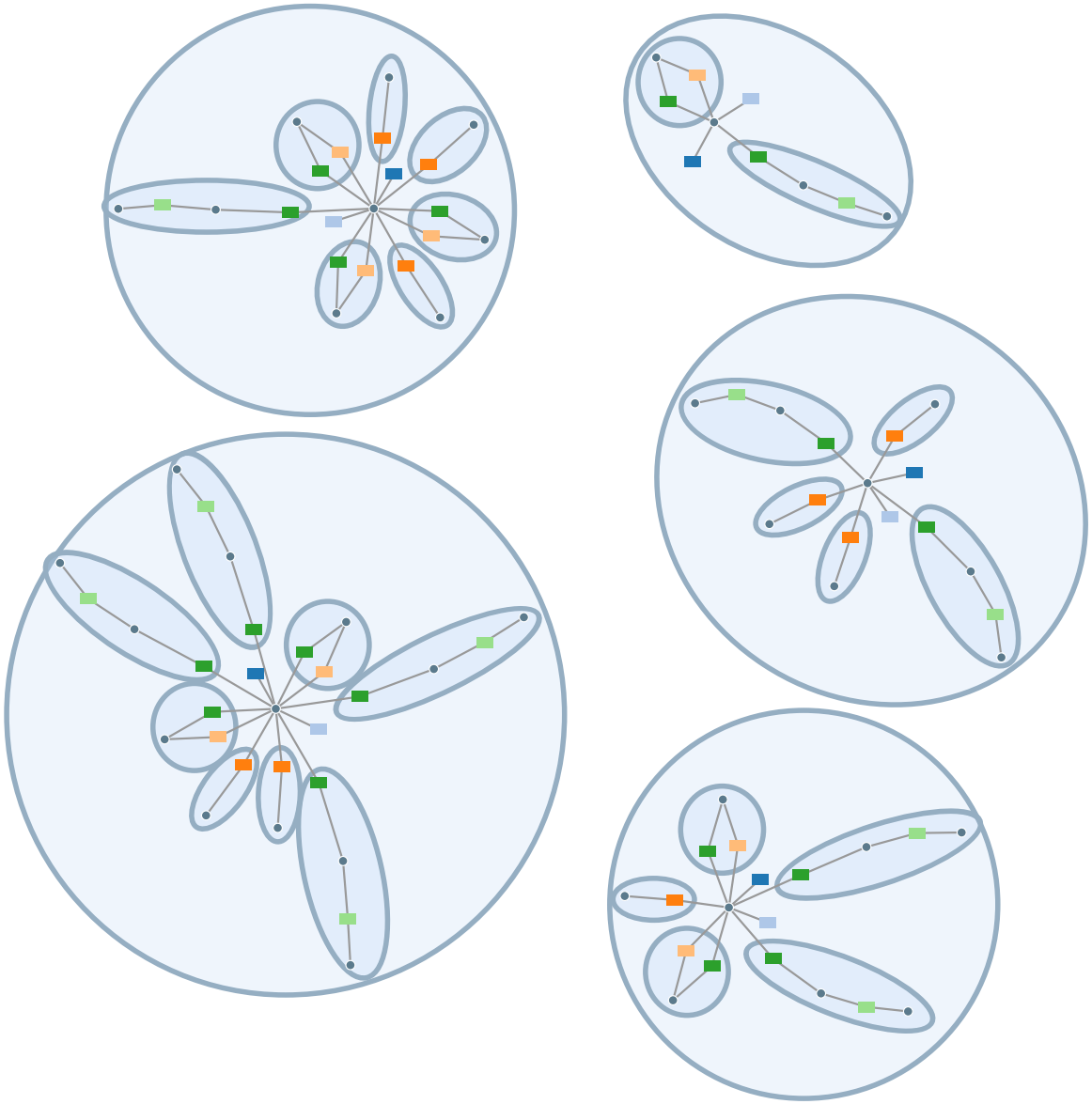


Example Hierarchy



server

Example Hierarchy

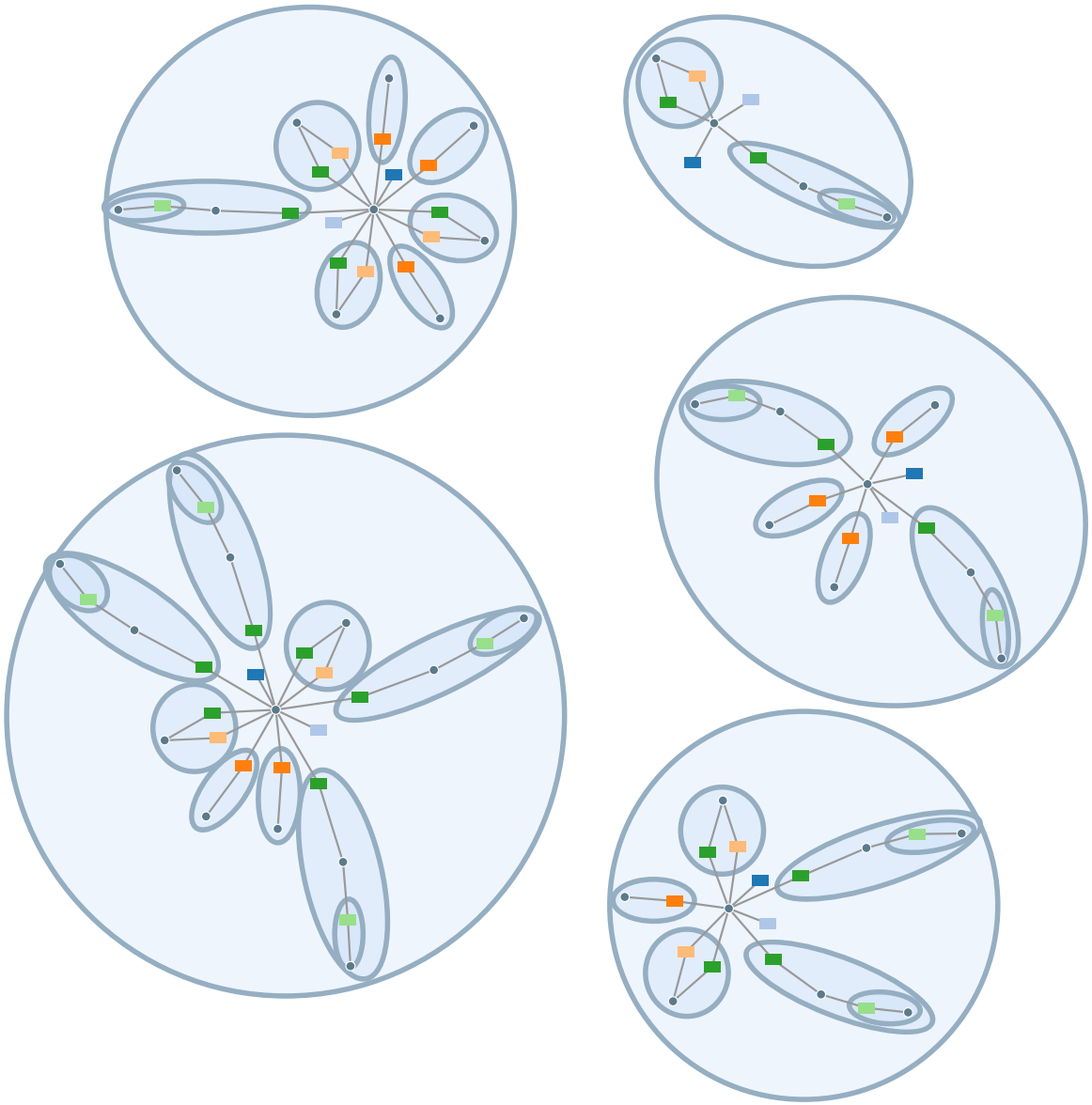


server

|

mailb

Example Hierarchy

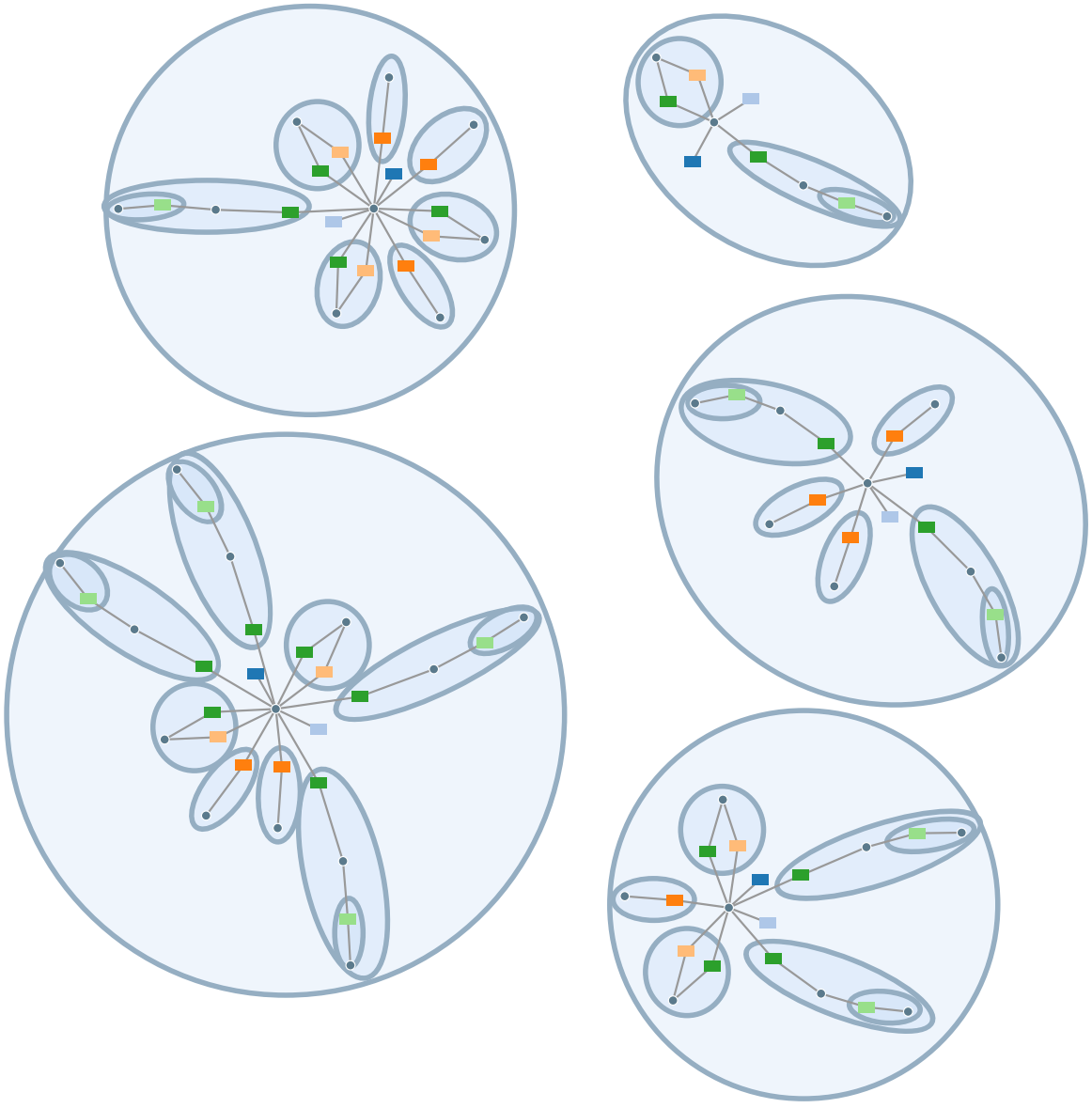


server

mailb

data

Example Hierarchy



$\mathcal{T}$

server

mailb

data

Types

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

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

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

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

Types

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

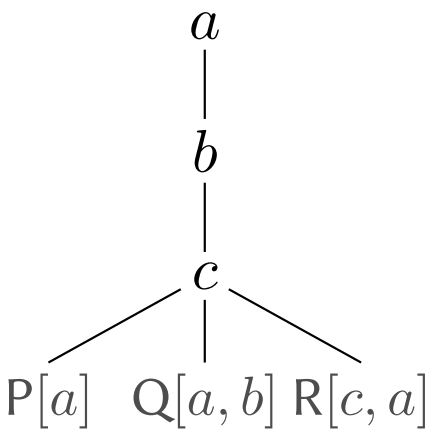
$$\mathsf{C}[s, m] := \overline{s}\langle m\rangle \parallel m(x).\mathsf{C}[s, m]$$

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

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

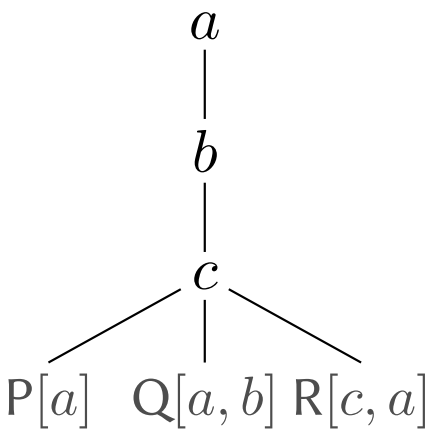
$\mathcal{T}$-shapedness

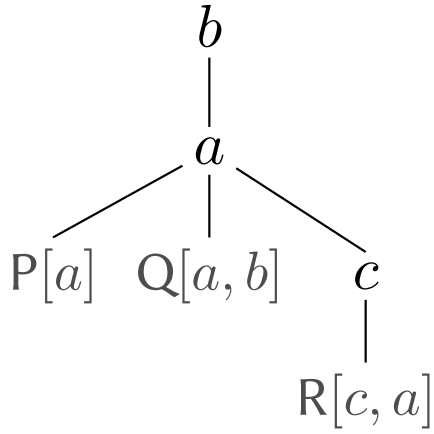
$$\nu a.\nu b.\nu c.(\mathsf{P}[a] \parallel \mathsf{Q}[a,b] \parallel \mathsf{R}[c,a])$$



$\mathcal{T}$-shapedness

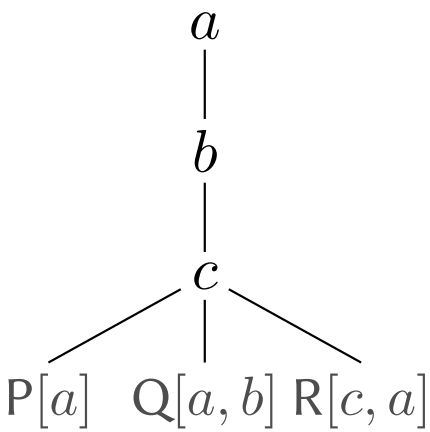
$$\nu a.\nu b.\nu c.(\mathsf{P}[a] \parallel \mathsf{Q}[a,b] \parallel \mathsf{R}[c,a]) \equiv \nu b.\nu a.(\mathsf{P}[a] \parallel \mathsf{Q}[a,b] \parallel \nu c.\mathsf{R}[c,a])$$

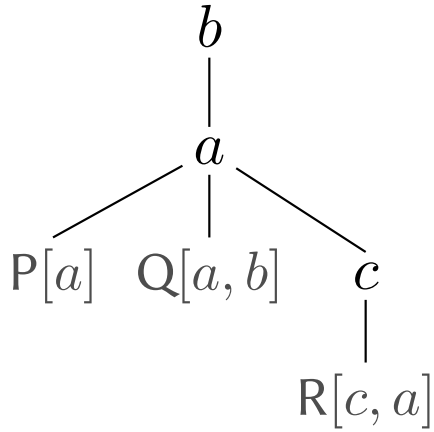


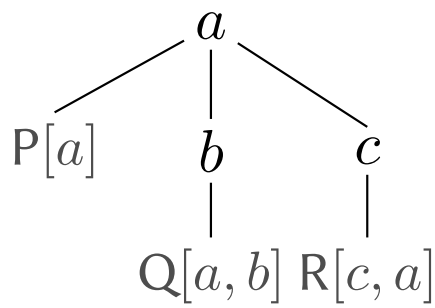


$\mathcal{T}$-shapedness

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

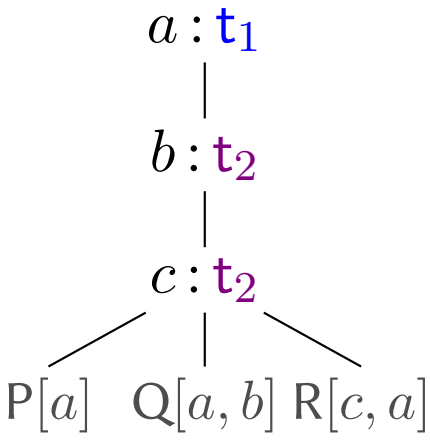


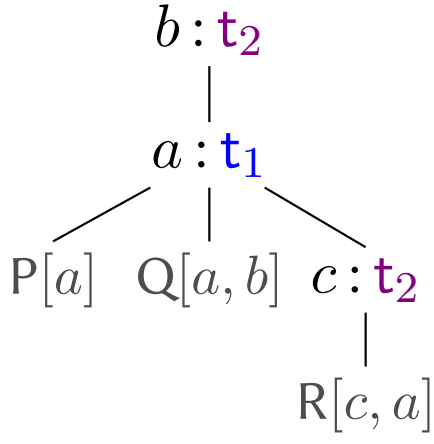


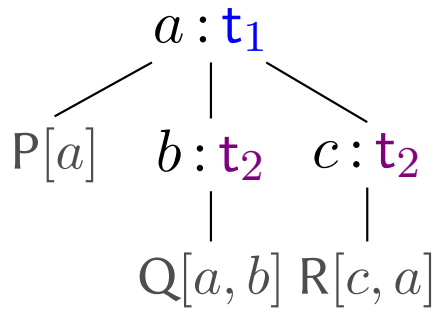


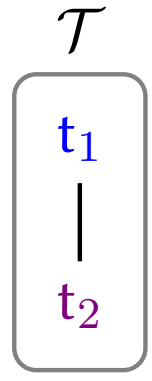
$\mathcal{T}$-shapedness

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



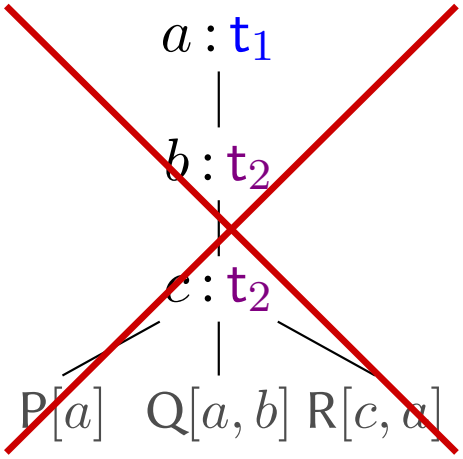


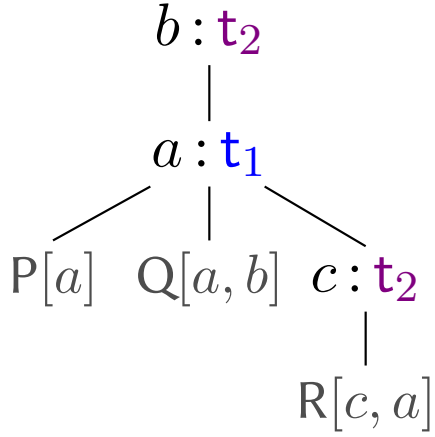


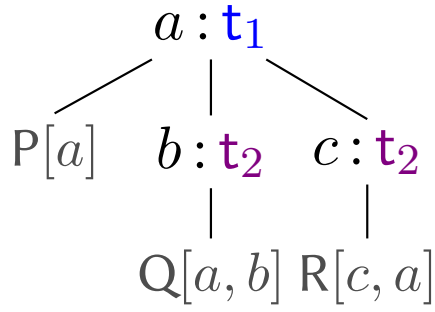


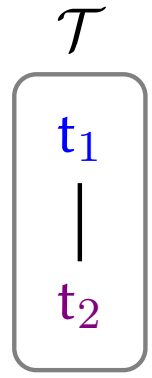
$\mathcal{T}$-shapedness

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



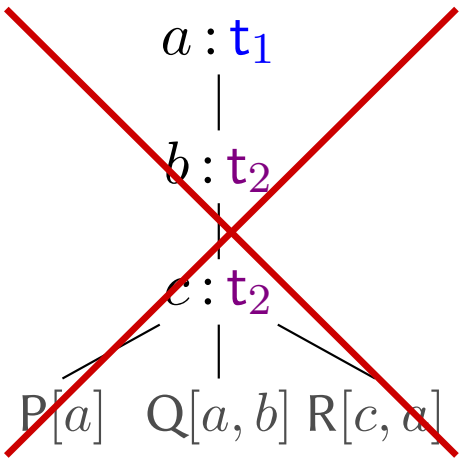


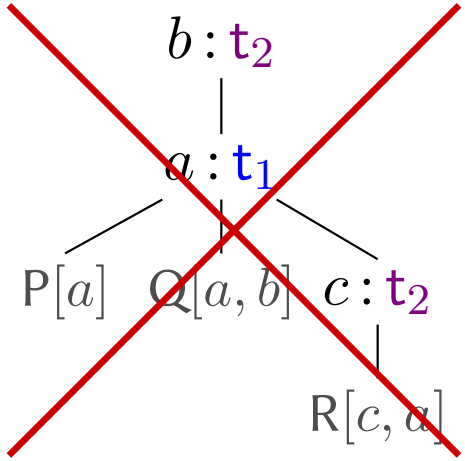


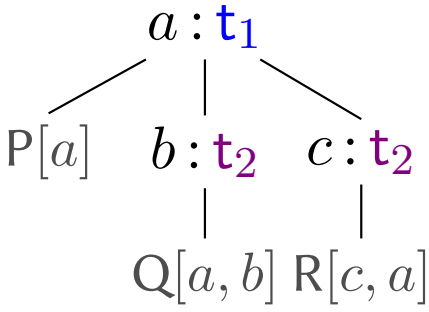


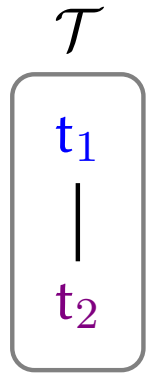
$\mathcal{T}$-shapedness

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



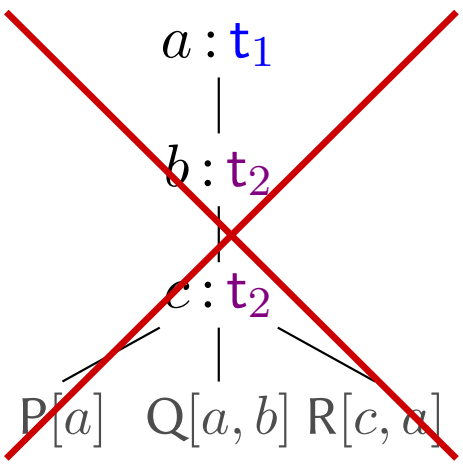


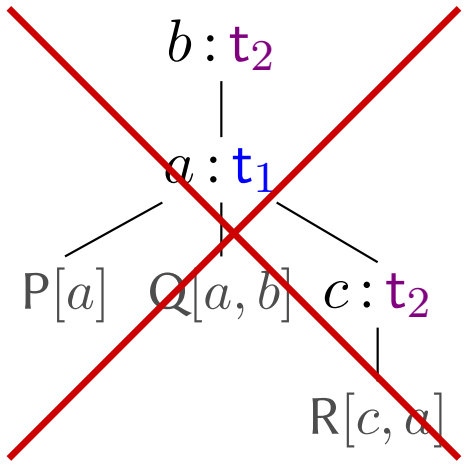


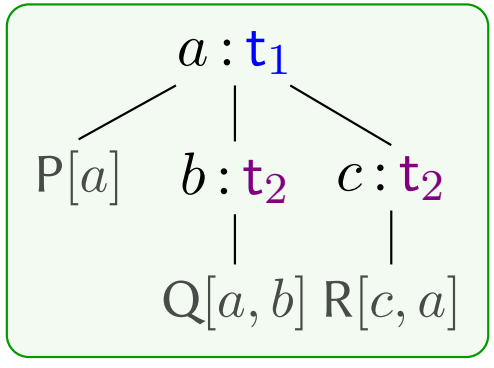


$\mathcal{T}$-shapedness

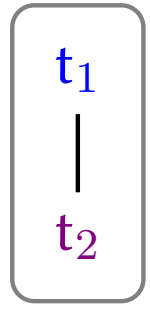
$$\nu(a : \mathtt{t}_1).\nu(b : \mathtt{t}_2).\nu(c : \mathtt{t}_2).(\mathsf{P}[a] \parallel \mathsf{Q}[a, b] \parallel \mathsf{R}[c, a])$$







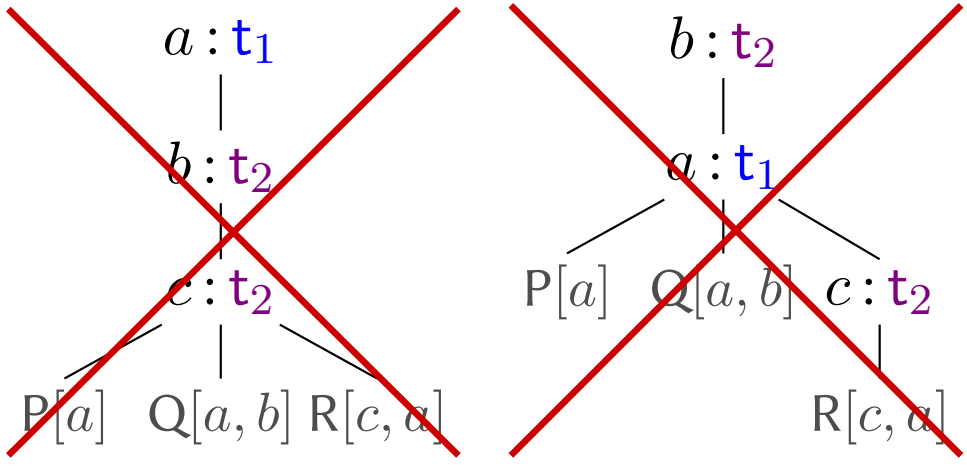
$\mathcal{T}$

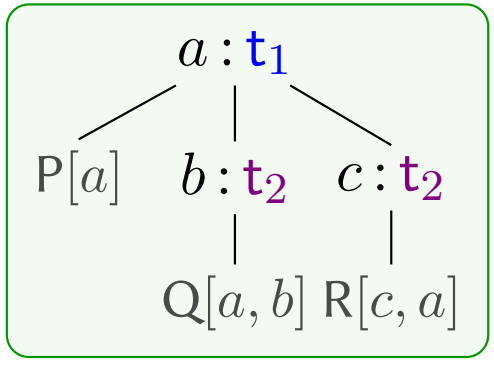


$\mathcal{T}$-shapedness

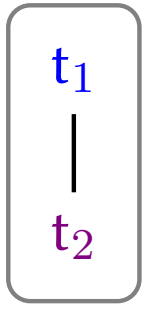
$\nu(a : \mathtt{t}_1).\nu(b : \mathtt{t}_2).\nu(c : \mathtt{t}_2).(\mathrm{P}[a] \parallel \mathrm{Q}[a, b] \parallel \mathrm{R}[c, a])$ is $\mathcal{T}$-shaped

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

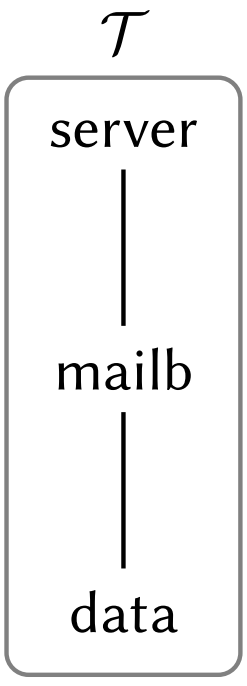


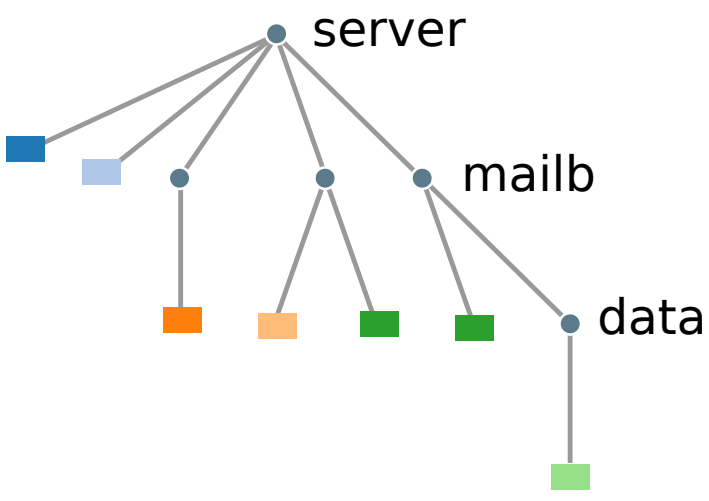


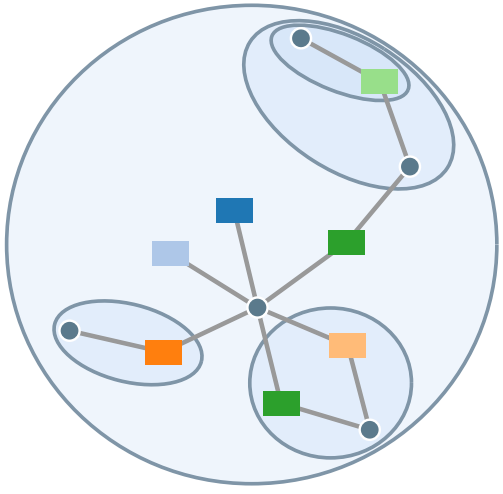
$\mathcal{T}$



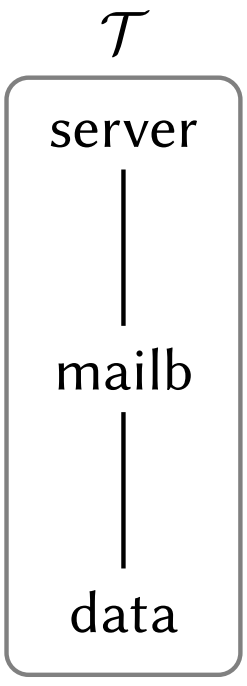
The Client/Server example is $\mathcal{T}$-shaped

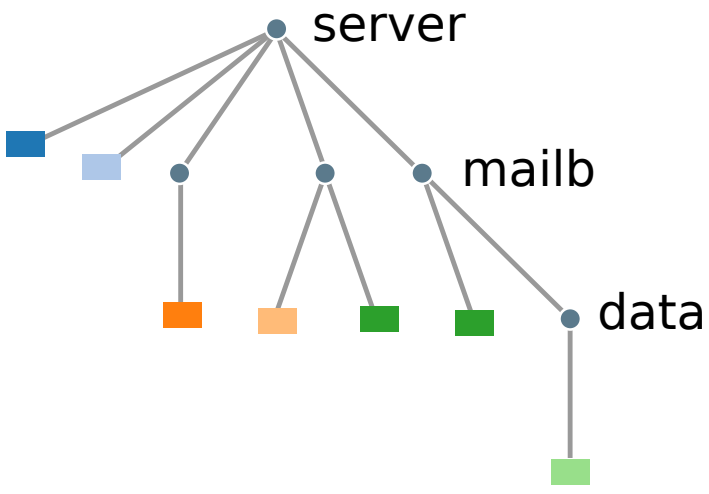


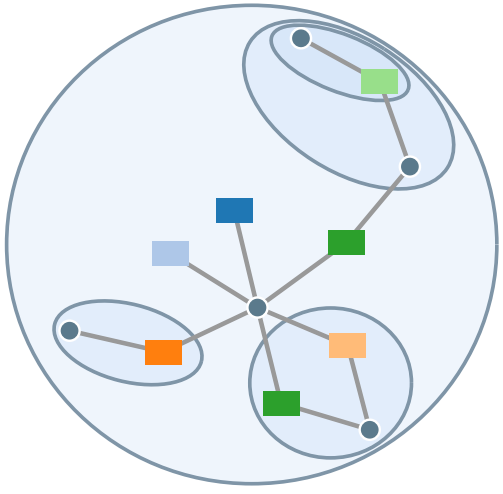




The Client/Server example is $\mathcal{T}$-shaped







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

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

Hierarchical systems

Definition

P is hierarchical

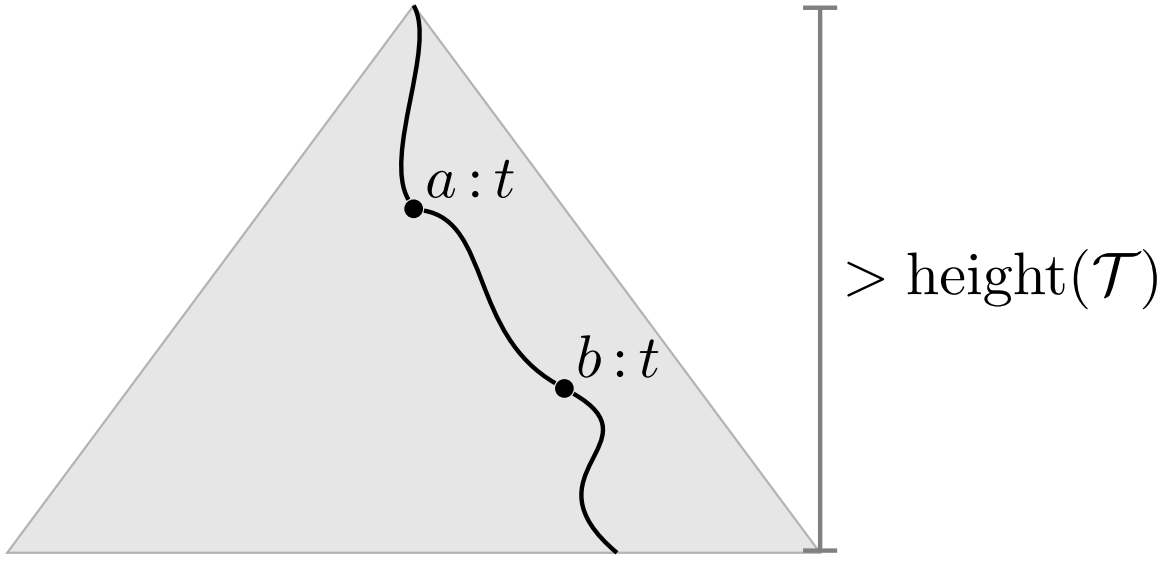
iif

$\exists \mathcal{T}$ finite . $\forall Q.\ P \rightarrow^* Q \implies Q$ is $\mathcal{T}$-shaped

(Hierarchical = $\mathcal{T}$-shapedness is 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



Designing a type system

Proposition

$$\text{Hierarchical} \implies \text{Depth-bounded}$$

Designing a type system

Proposition

$$\text{Hierarchical} \implies \text{Depth-bounded}$$

Problem: Being hierarchical is still an undecidable property.

Designing a type system

Proposition

$$\text{Hierarchical} \implies \text{Depth-bounded}$$

Problem: Being hierarchical is still an undecidable property.

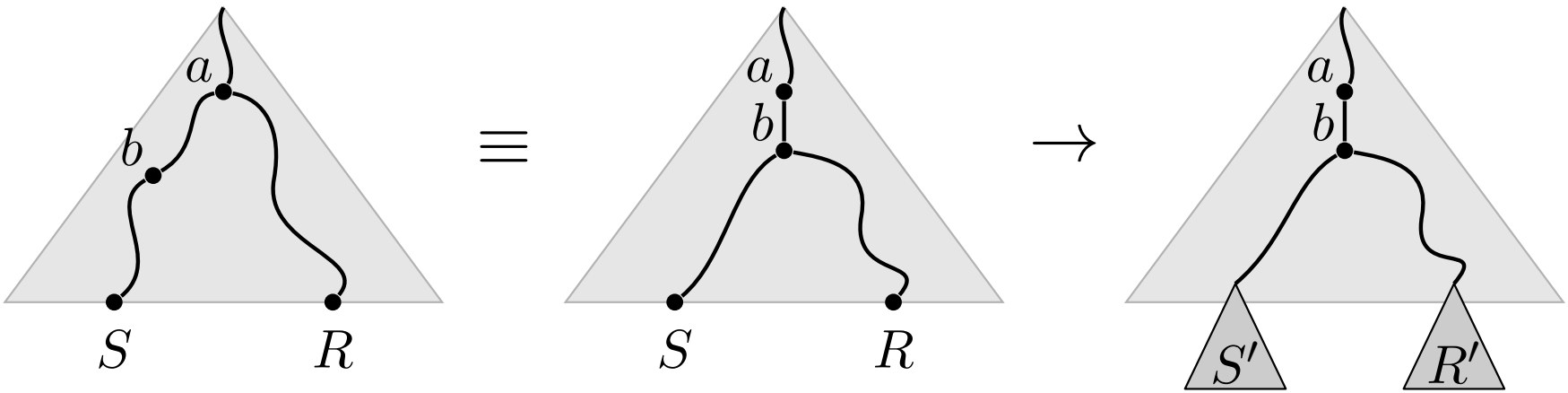
Solution: But now we have more structure, which we exploit to design a type system such that

$$\text{If} \quad \Gamma \vdash_{\mathcal{T}} P \quad \text{then}$$

$$P \text{ is } \mathcal{T}\text{-shaped and } P \to Q \implies Q \text{ is } \mathcal{T}\text{-shaped}$$

Standard reductions

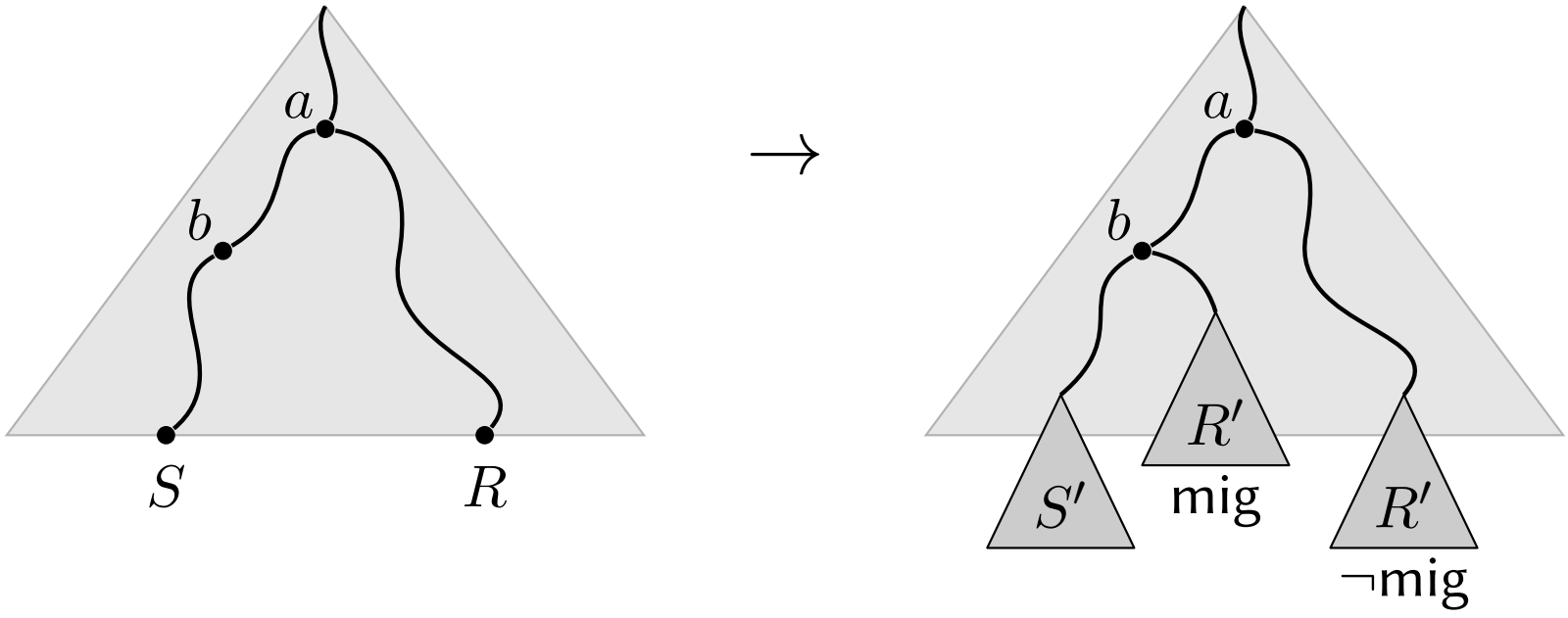
$$S = \overline{a}\langle b\rangle.S' \qquad R = a(x).R'$$



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

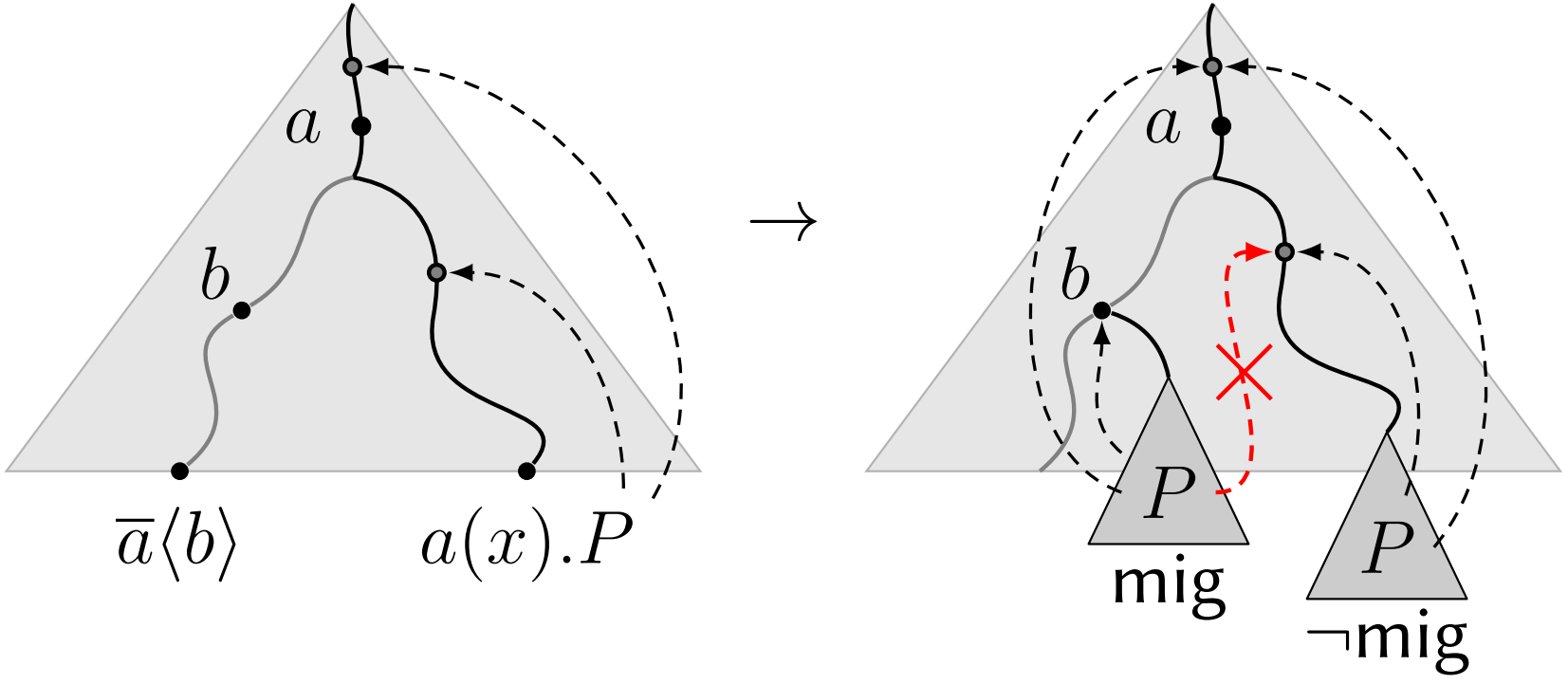
$\mathcal{T}$-shaped reductions

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



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

$\mathcal{T}$-shaped reductions are special



The Type System

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

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

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

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

Soundness

Subject reduction

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

Soundness

Subject reduction

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

Theorem

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

Soundness

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

When $\Gamma \vdash_{\mathcal{T}} P$ and P is $\mathcal{T}$-shaped,
we say P is typably hierarchical.

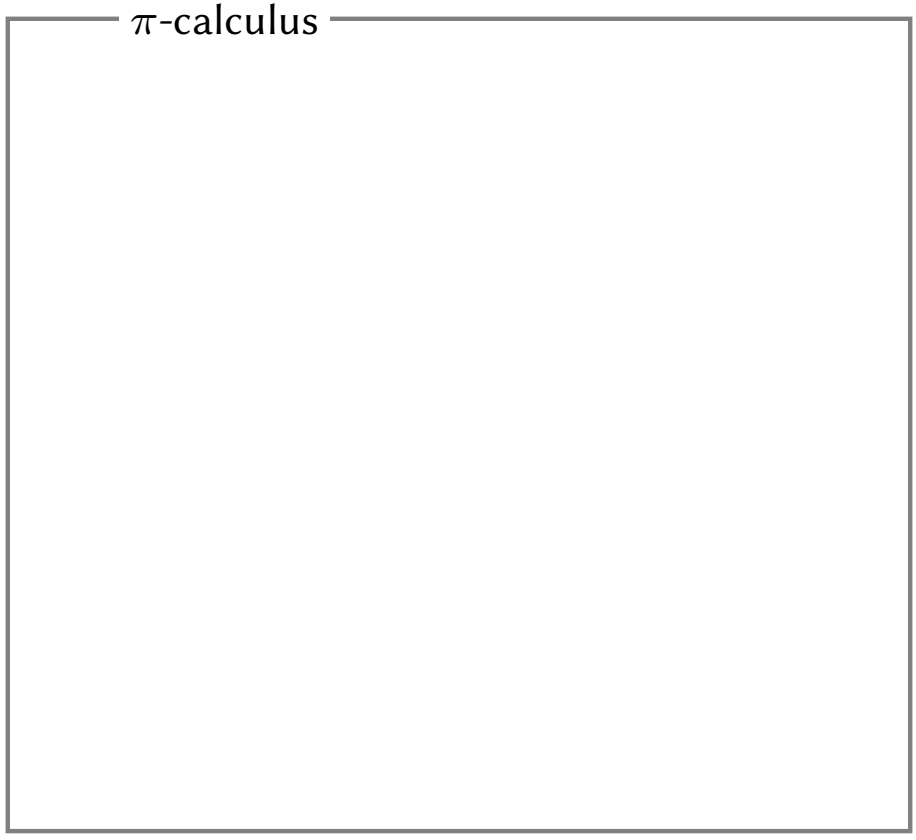
Type Inference

The type system:

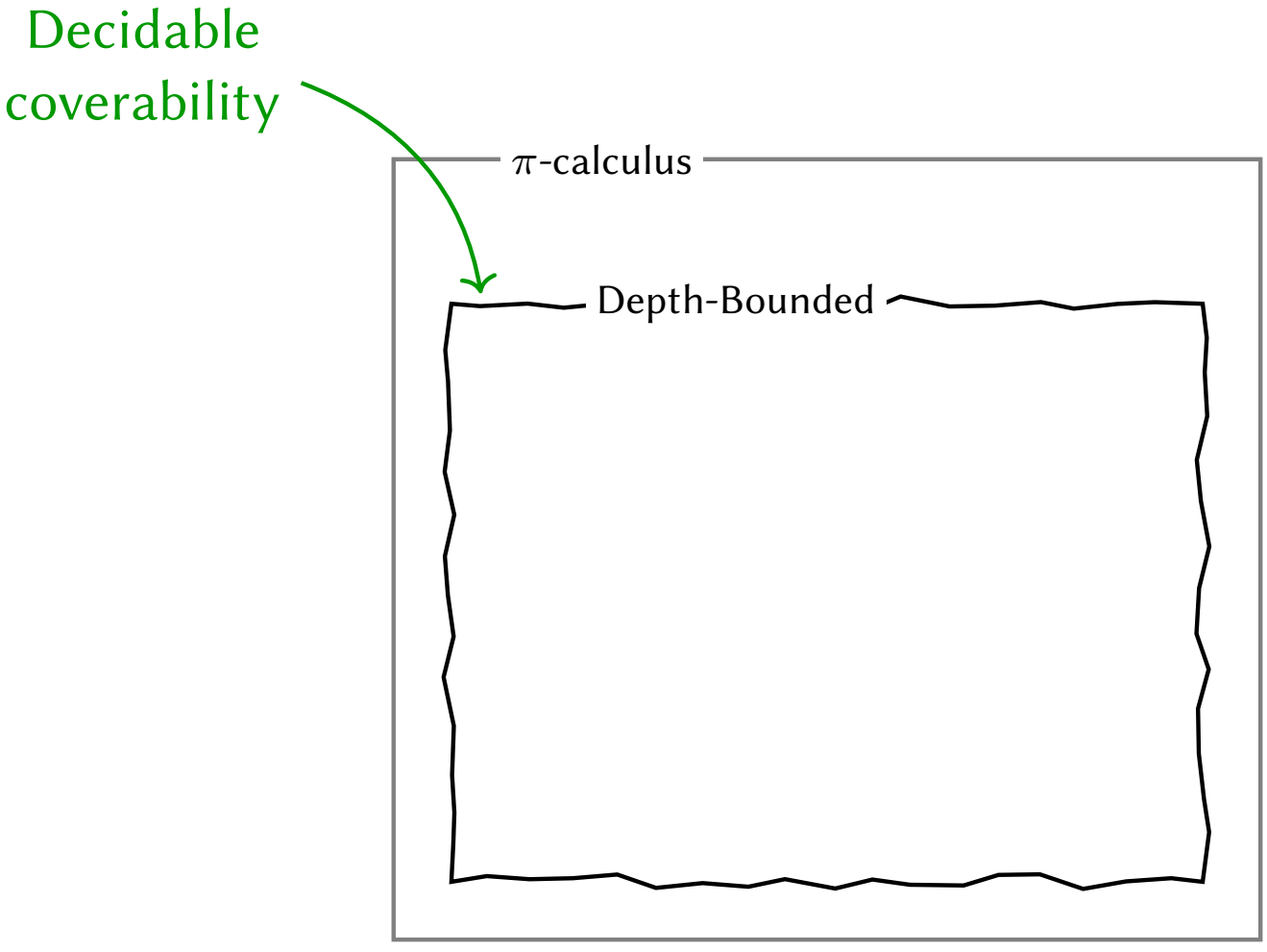
- **type checking:** decidable in P
- **type inference:** decidable in NP
- first type system inferring topological properties

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

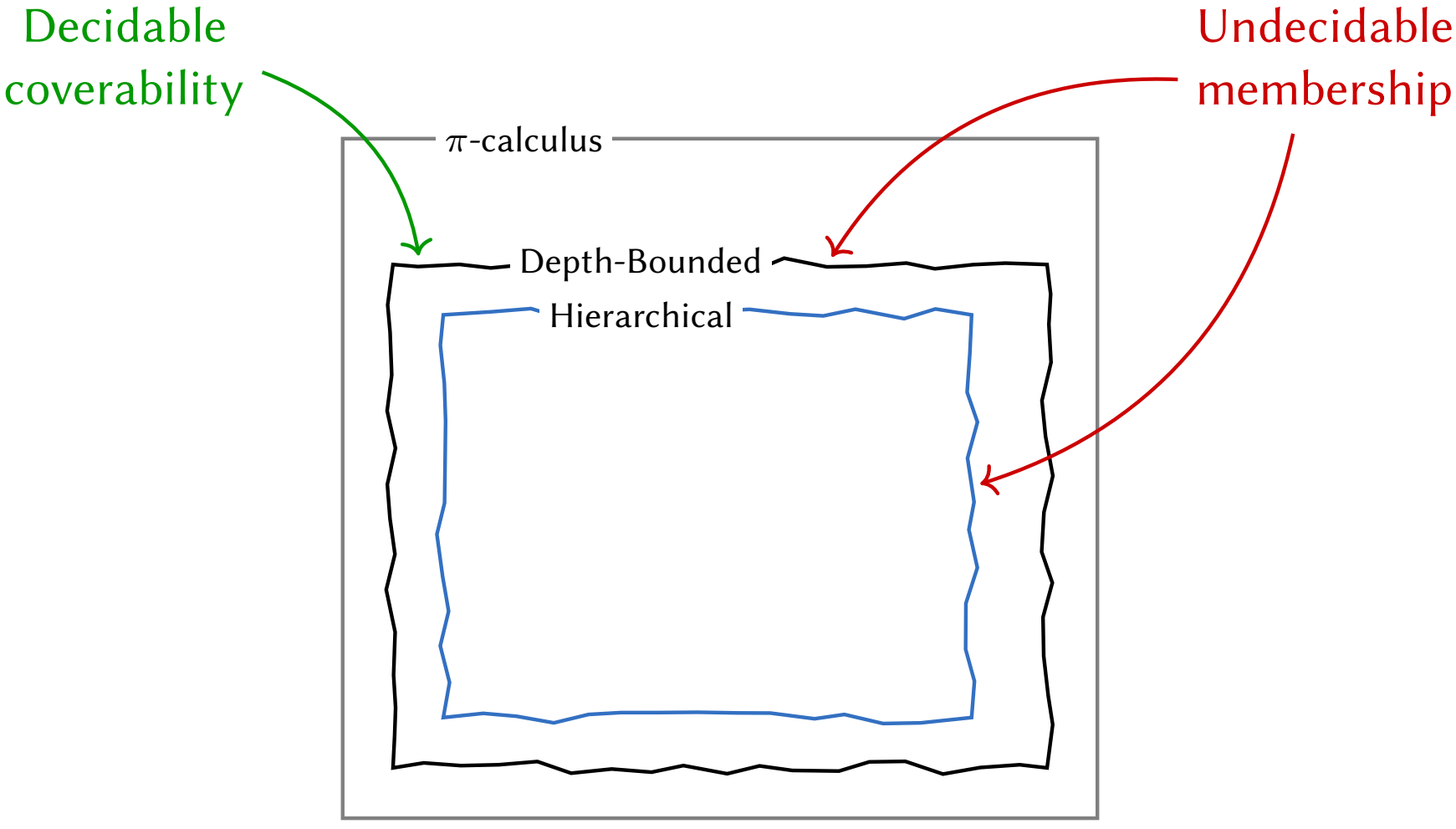
Expressivity



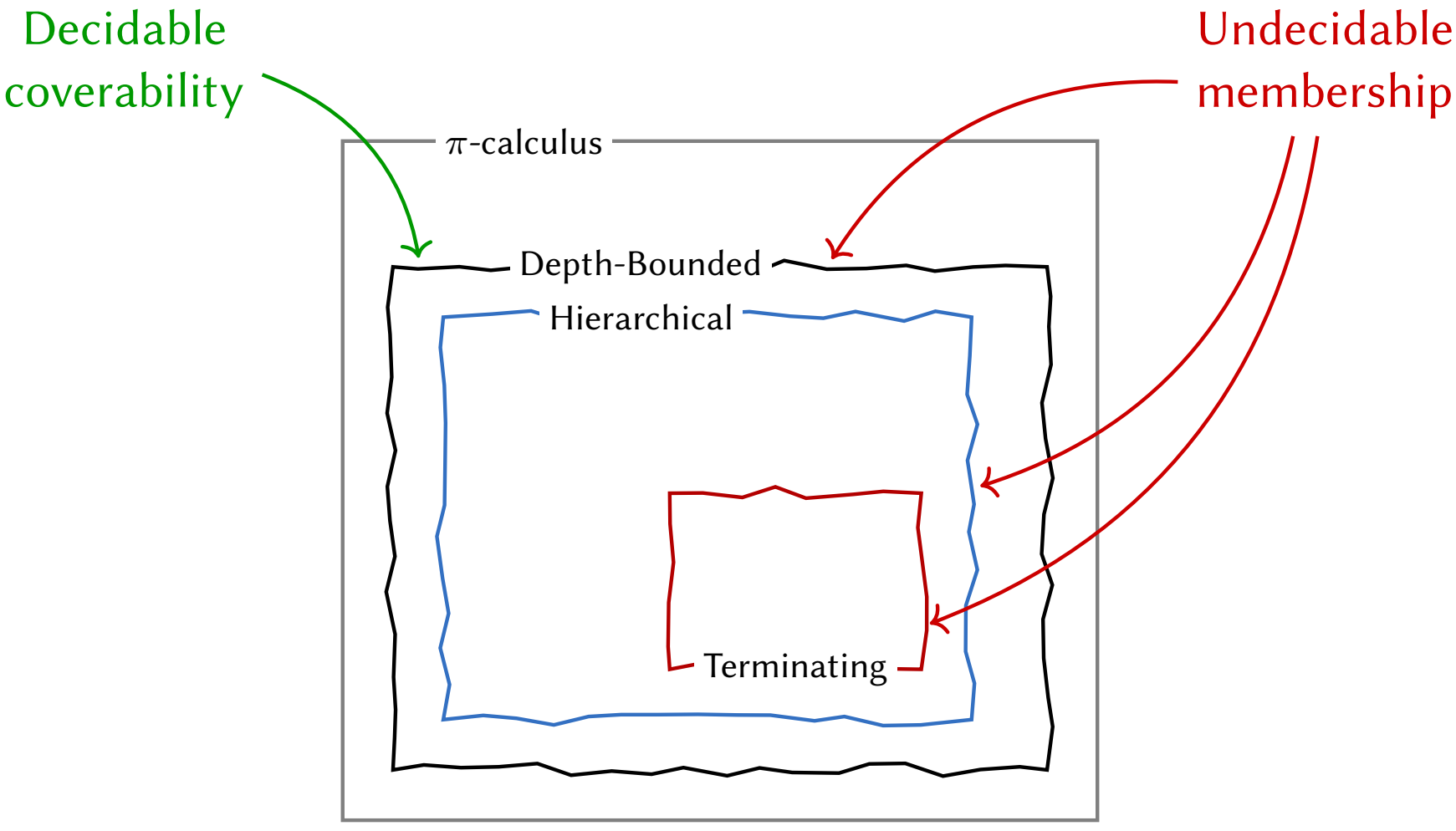
Expressivity



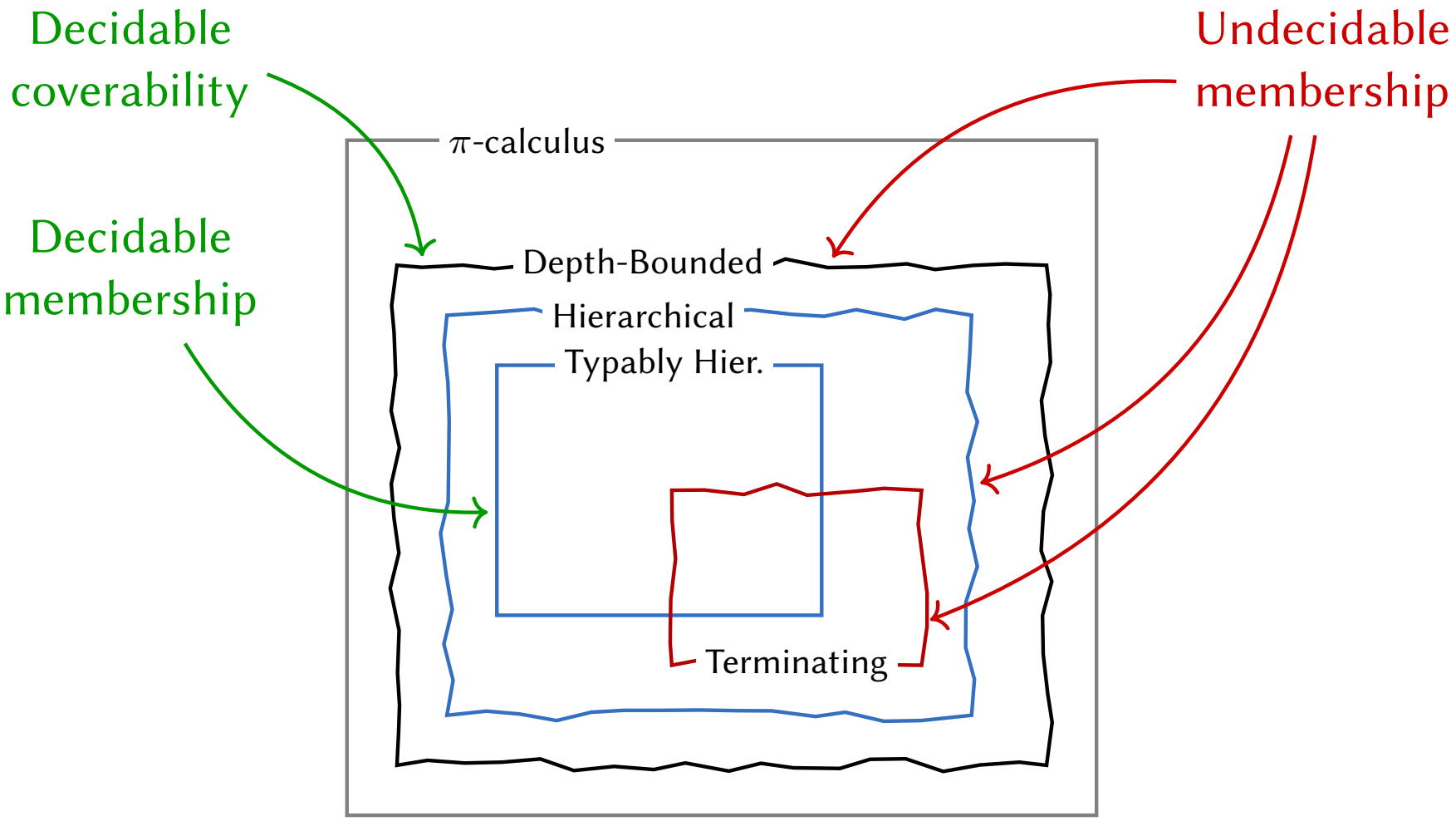
Expressivity



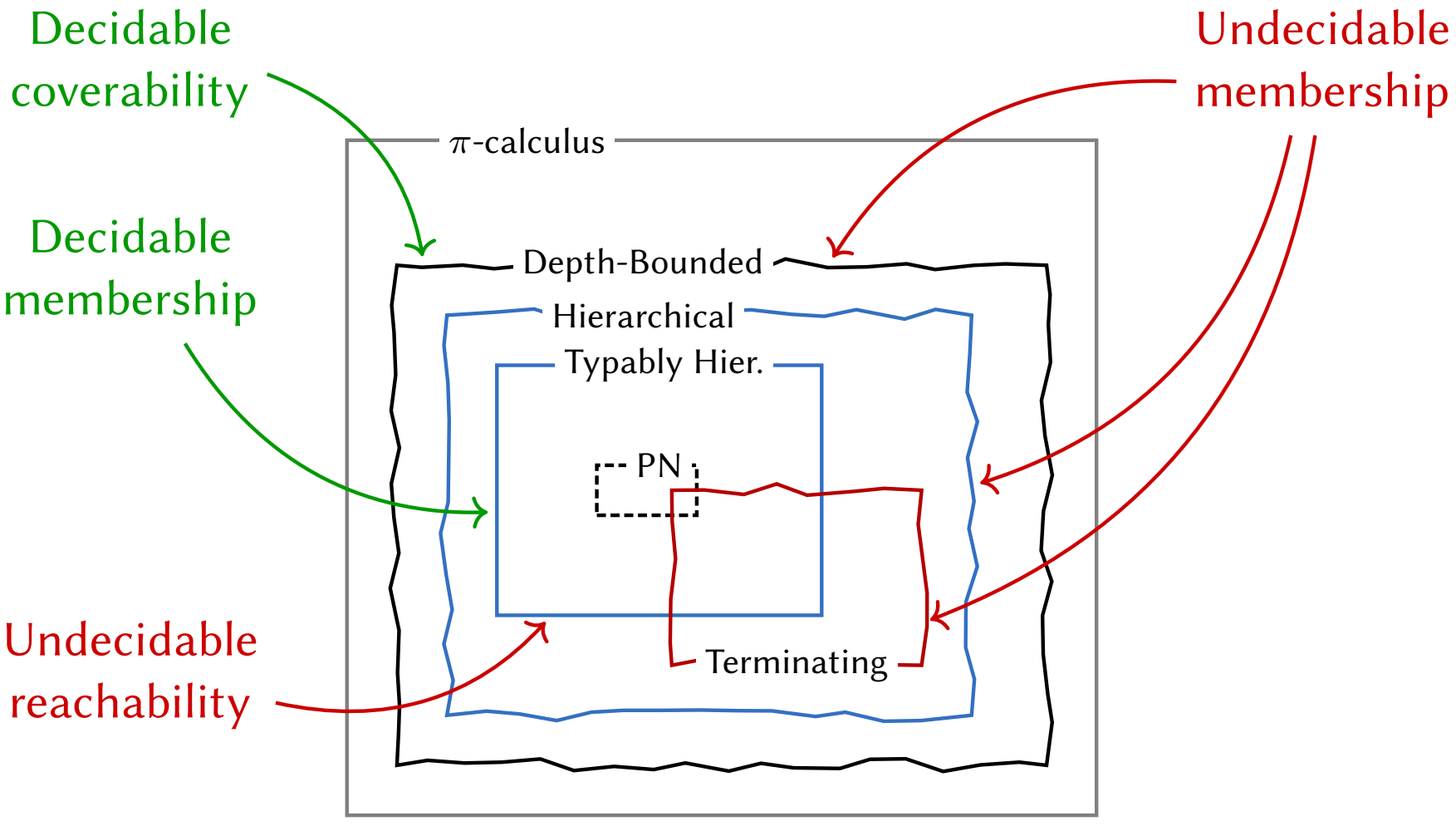
Expressivity



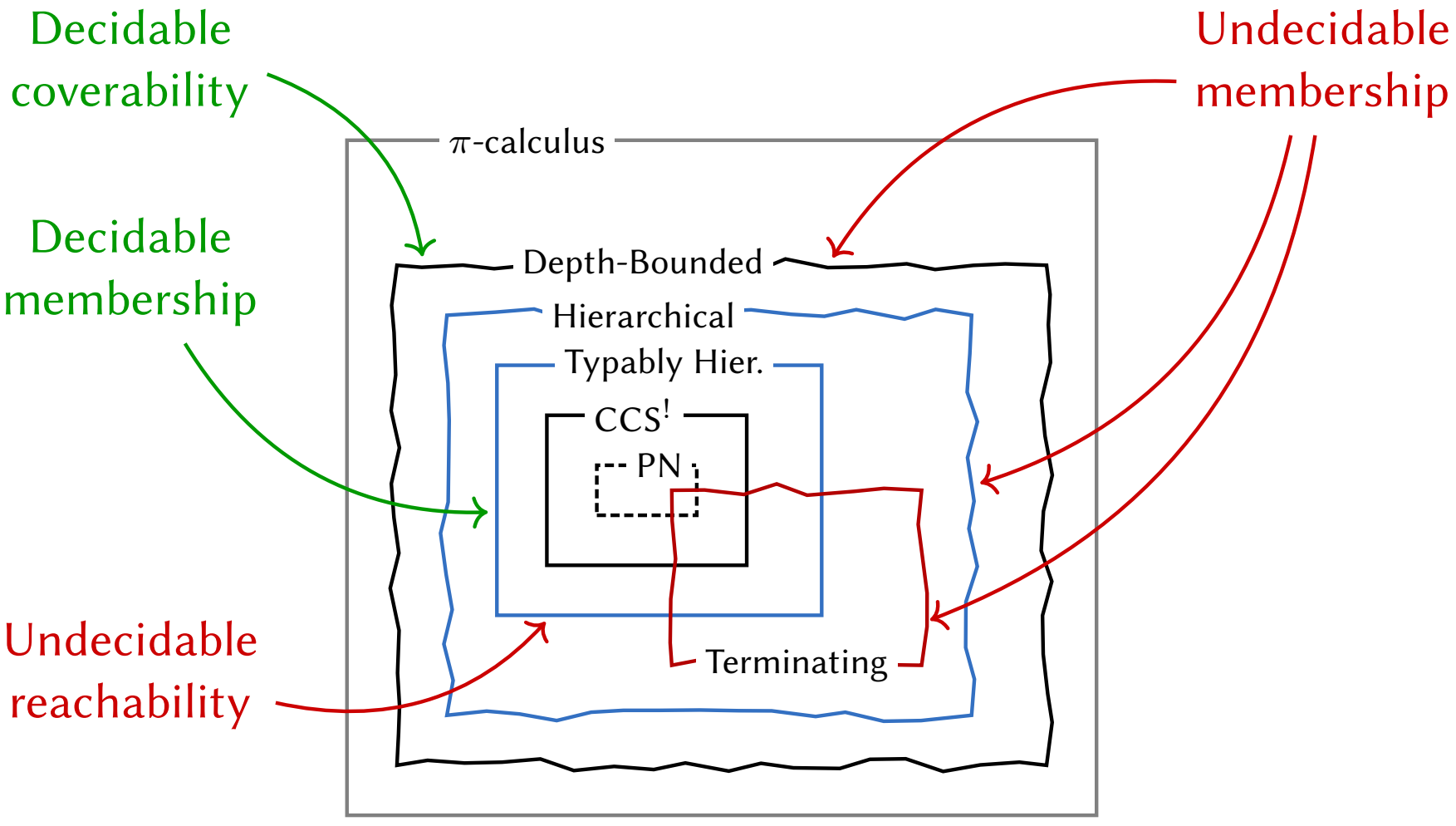
Expressivity



Expressivity



Expressivity



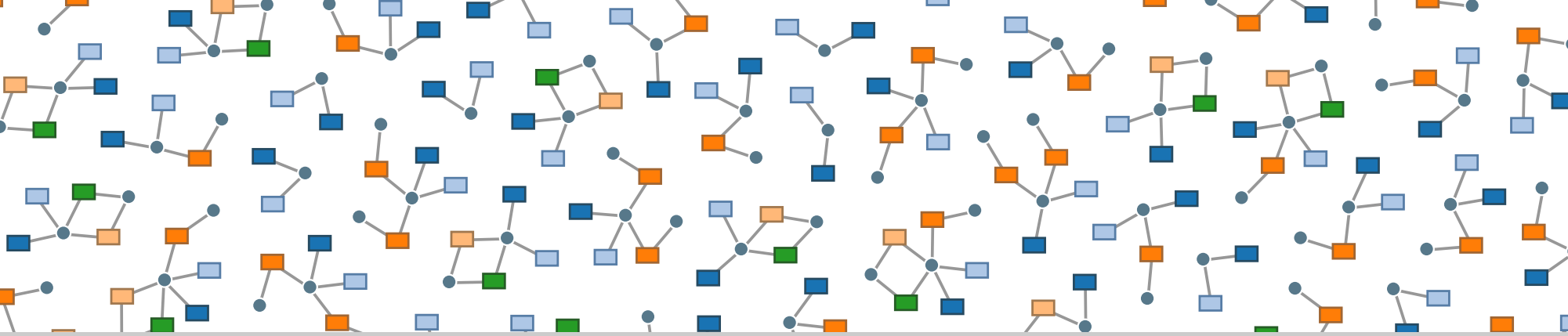
Summary and Future

Contributions:

- Definition of *hierarchical system*
- A *type system* for hierarchical systems
- Hierarchical systems are expressive
  but have *decidable coverability & termination*

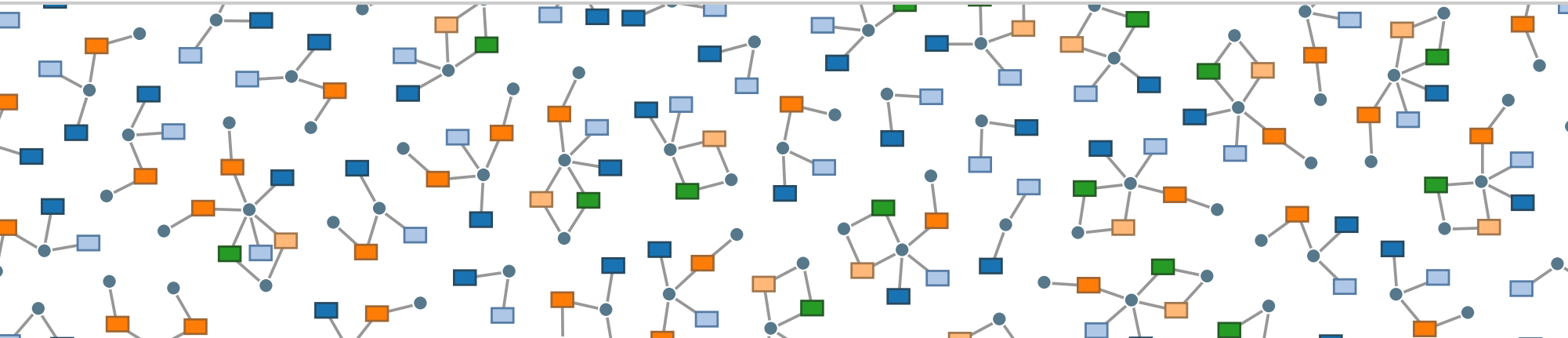
Future work:

- use typing failures to do smart abstractions
- make the type system more precise
- applications to
  - protocol verification
  - concurrent heap manipulating programs verification



Thank you!

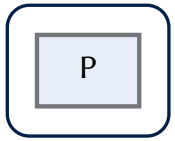
@bordaigorl
emanueledosualdo.com



Appendix

Verification of Depth Bounded systems

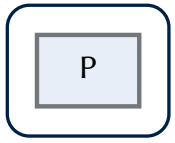
Coverability



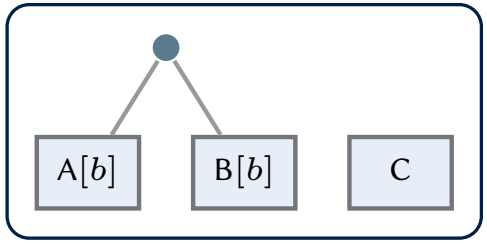
Init

Verification of Depth Bounded systems

Coverability



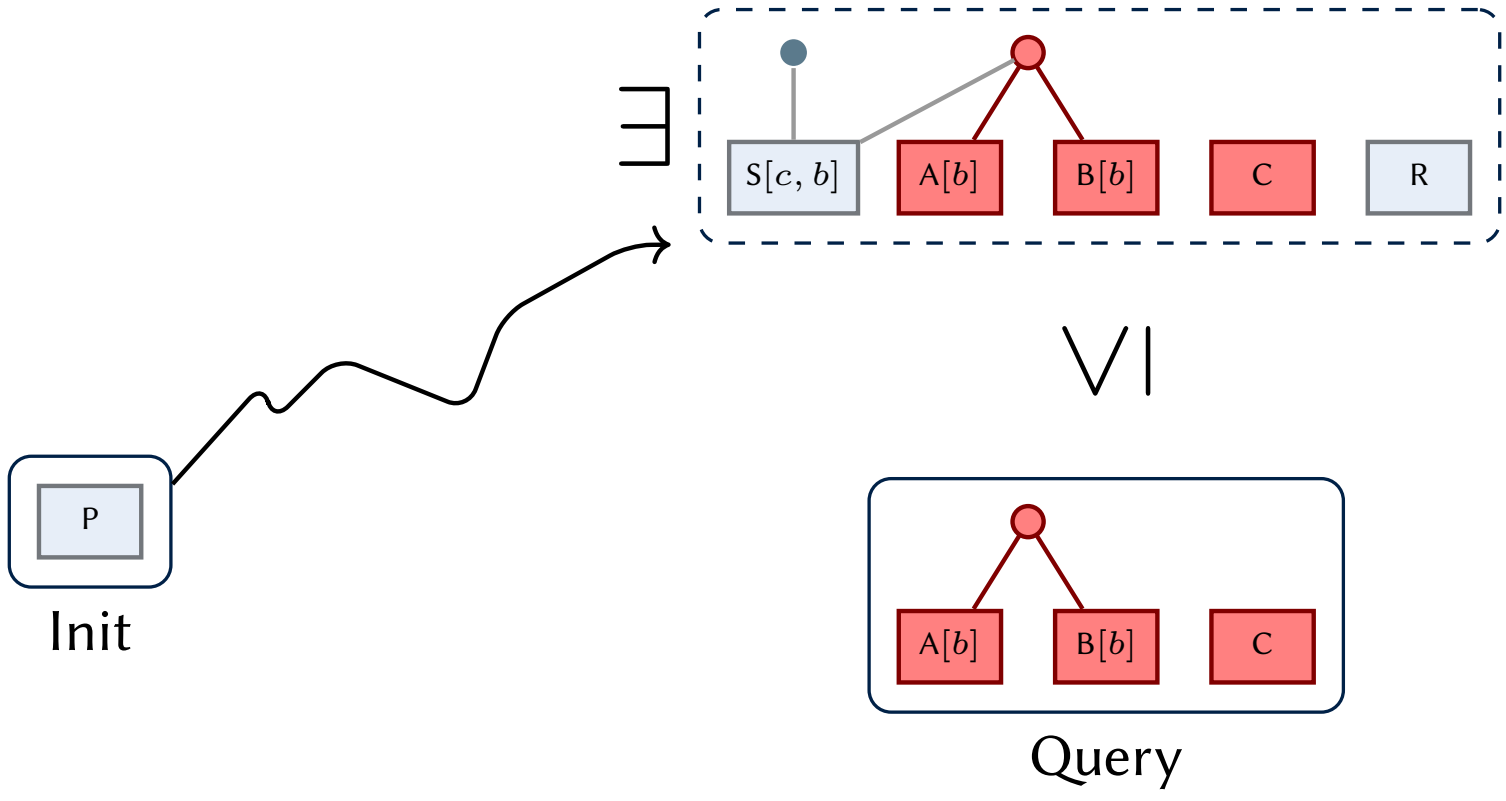
Init



Query

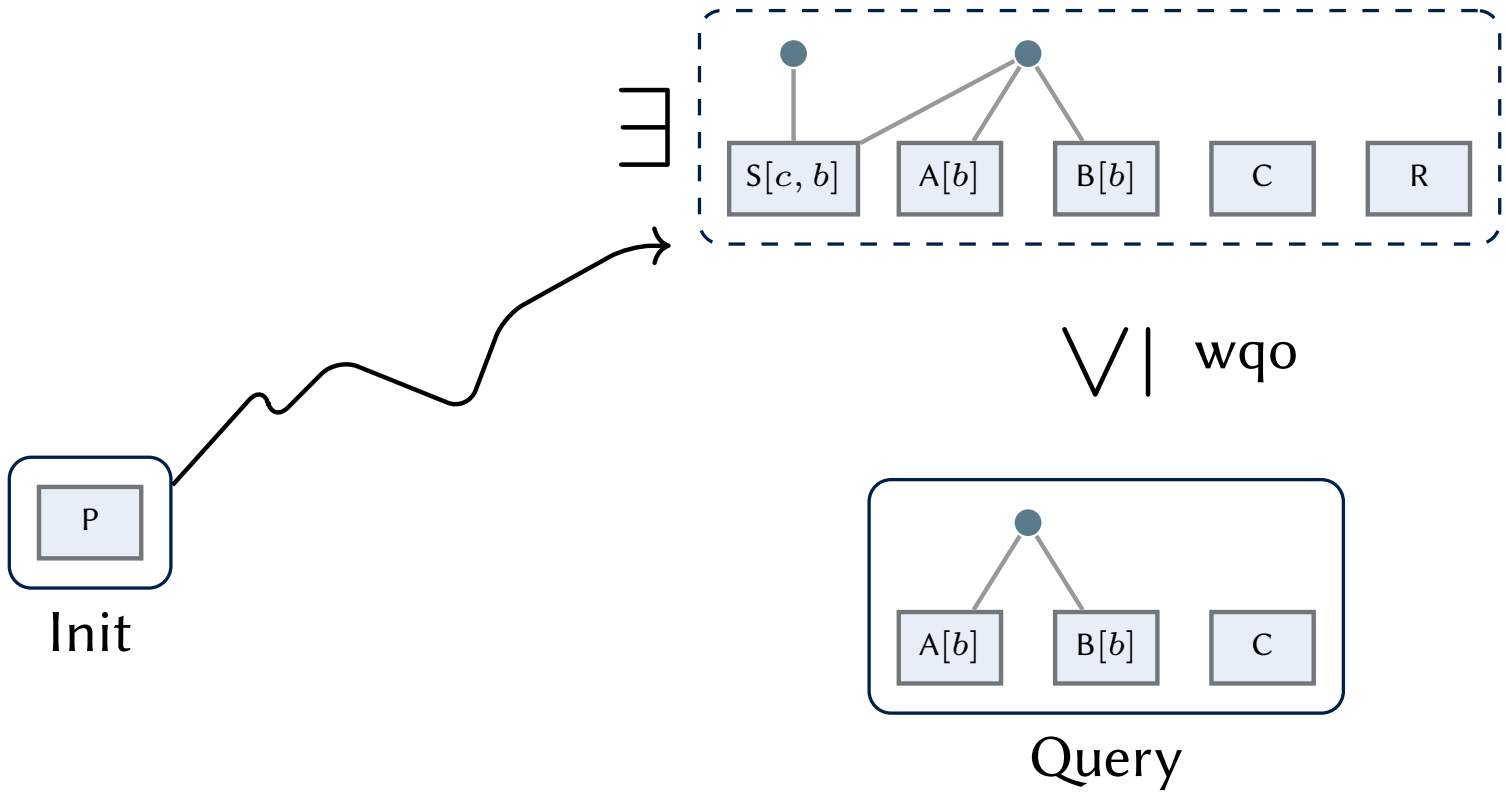
Verification of Depth Bounded systems

Coverability



Verification of Depth Bounded systems

Coverability Decidable for depth bounded systems via WSTS



π-calculus

Syntax:

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

Normal form:

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

Depth

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

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

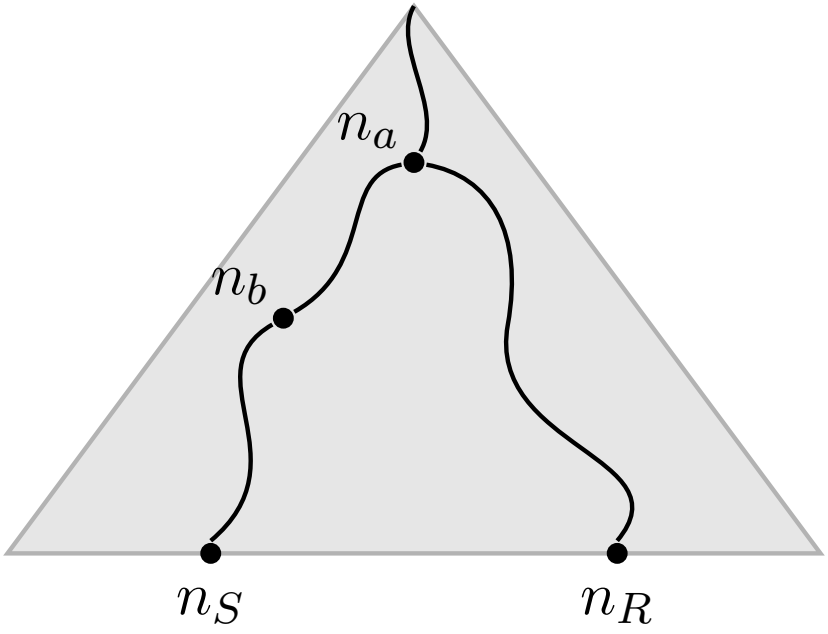
The *depth* of a term is defined as the minimal nesting of restrictions in its congruence class:

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

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

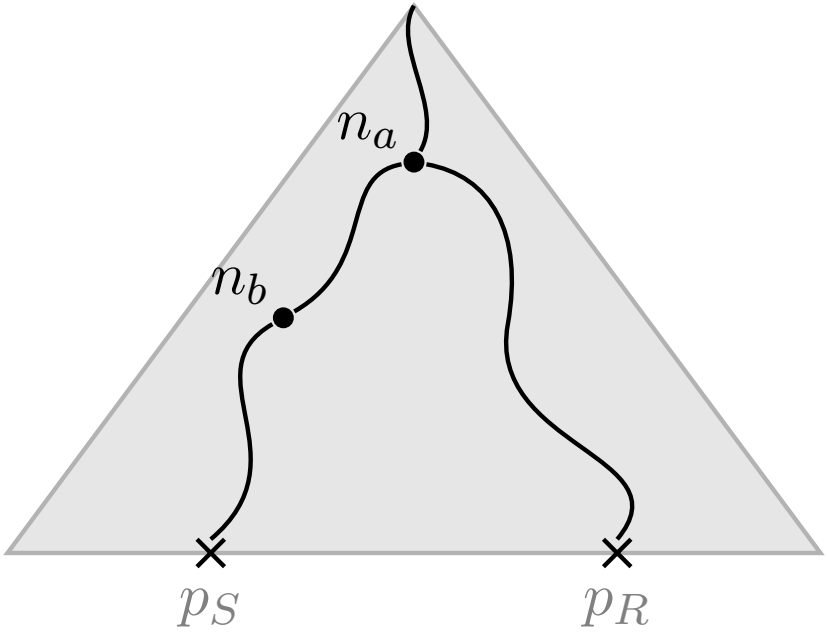
Soundness argument

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



Soundness argument

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



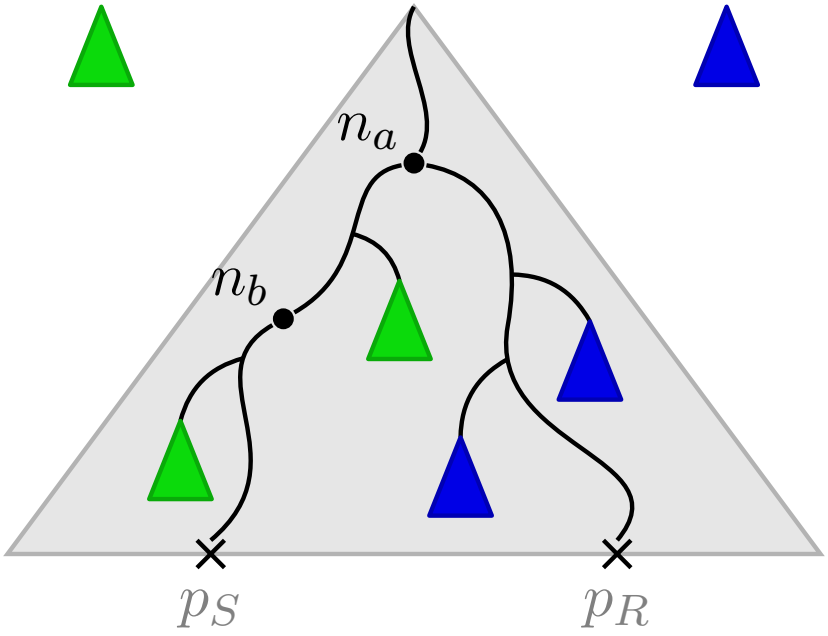
$\in S'$

$\in R'_{\mathsf{mig}}$

$\in R'_{\neg\mathsf{mig}}$

Soundness argument

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



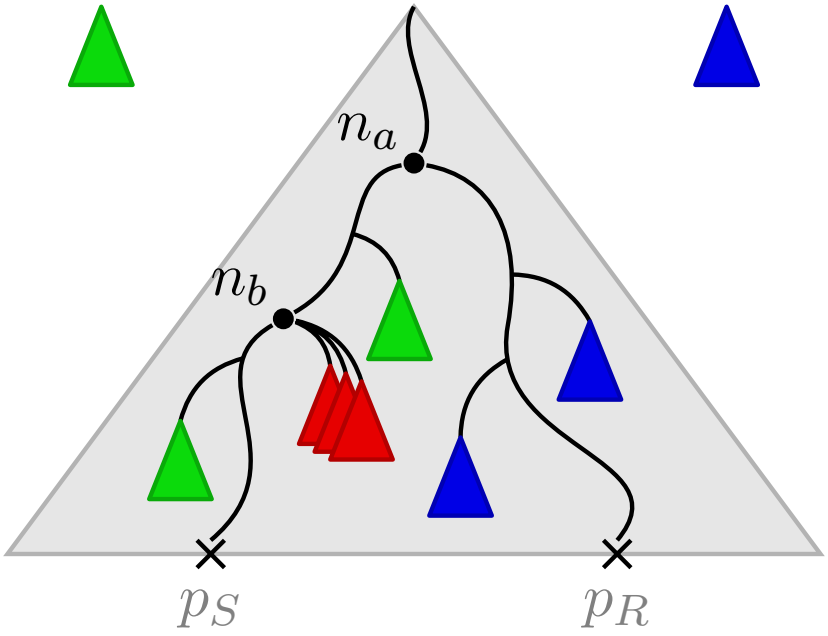
$\in S'$

$\in R'_{\mathsf{mig}}$

$\in R'_{\neg\mathsf{mig}}$

Soundness argument

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



$\in S'$

$\in R'_{\mathsf{mig}}$

$\in R'_{\neg\mathsf{mig}}$