Cours de D.E.A

Abstract Interpretation of Mobile Systems

Jérôme Feret
École Normale Supérieure
http://www.di.ens.fr/~feret

February, 2004

Overview

1. Mobile systems

2. Non-standard semantics

3. Approximate the collecting semantics

4. Trace-based analysis

Mobile systems

Mobile system

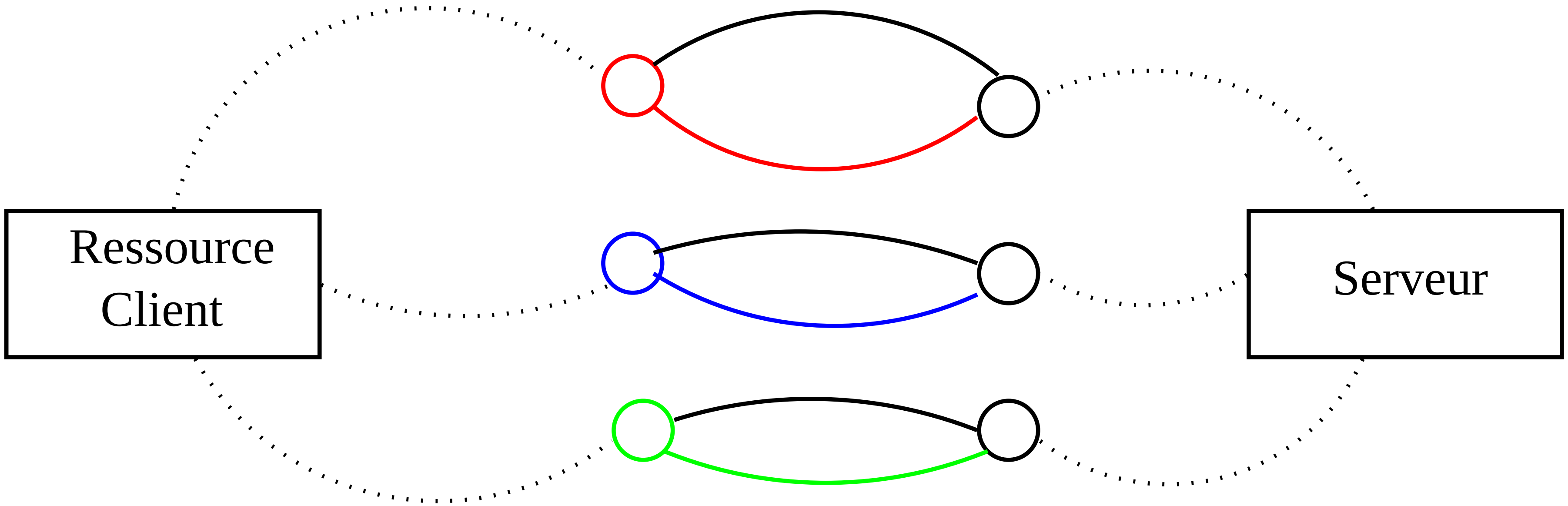
A pool of processes which interact and communicate:

Interactions

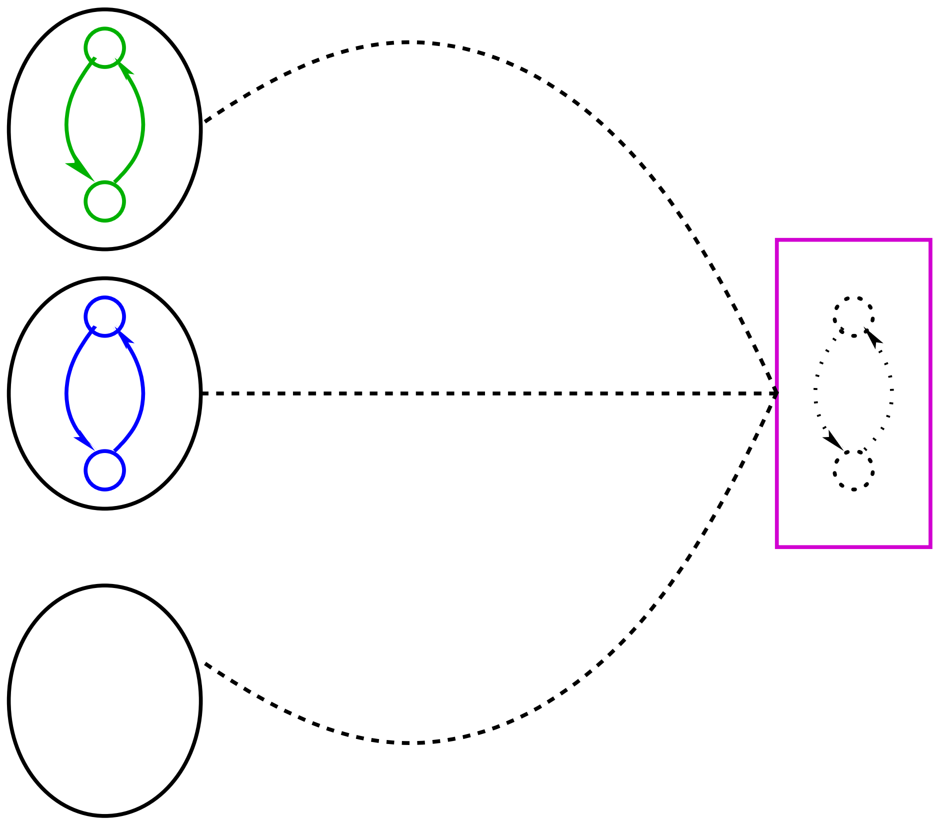
- synchronize process computation;
- change process structure (communication,migration);
- change communication links;
- create new processes;

Topology of interaction may be unbounded !

A network



Example: a server



Motivation

We want to compute a sound approximation of:

- the interactions between the processes,

  ⇒ to prove that private information cannot be passed to forbidden processes.

  This approximation should be non-uniform and context-free.

- the number of processes which occur during computation sequences:

  ⇒ to detect

  - mutual exclusion between processes;
  - non-exhaustion of resources;

  ⇒ to provide a good criterion of partitioning.

We propose a polynomial solution.

π-calculus: syntax

Let Name be an infinite set of channel names, and Label an infinite set of labels,

$$
\begin{array}{rcll}
P & ::= & \text{action}.P & \text{(Action)} \\
 & | & (P \mid P) & \text{(Parallel composition)} \\
 & | & (P + P) & \text{(Non deterministic choice)} \\
 & | & \emptyset & \text{(End of a process)}
\end{array}
\qquad
\begin{array}{rcll}
\text{action} & ::= & c!^i[x_1, ..., x_n] & \text{(Message)} \\
 & | & c?^i[x_1, ..., x_n] & \text{(Input guard)} \\
 & | & *c?^i[x_1, ..., x_n] & \text{(Replication guard)} \\
 & | & (\nu\ x) & \text{(Channel creation)}
\end{array}
$$

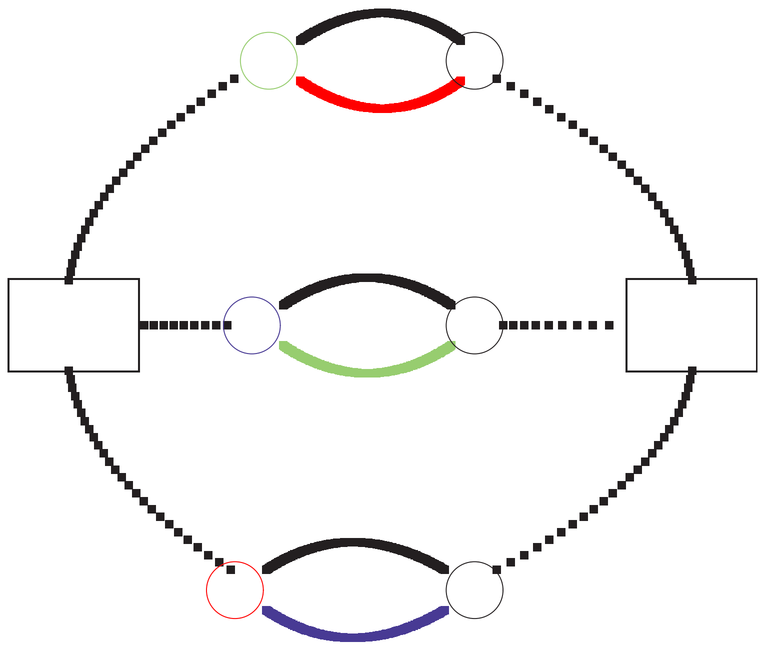
where $n \geqslant 0$, c, x_1, ..., x_n, x, $\in$ Name and $i \in$ Label.

ν and $?$ are the only name binders.
We denote by $fn(P)$ the set of free names in P, and by $bn(P)$ the set of bound names in P.

α-conversion

α-conversion destroys the link between channel names and processes which have declared them:

$$
\begin{array}{l}
(\nu\ \mathrm{port})(\nu\ \mathrm{gen})(\nu\ \mathrm{data}_1)(\nu\ \mathrm{email}_1)\\
(\nu\ \mathrm{data}_2)(\nu\ \mathrm{email}_2)\\
\ (\mathrm{Server}\ \mid\ \mathrm{Customer} \mid\ \mathrm{gen}!^5[]\\
\ \mid\ \mathrm{email}_1!^4[\mathrm{data}_1]\ \mid\ \mathrm{email}_2!^4[\mathrm{data}_2])\\
\sim_\alpha\\
(\nu\ \mathrm{port})(\nu\ \mathrm{gen})(\nu\ \mathrm{data}_2)\\
(\nu\ \mathrm{email}_1)(\nu\ \mathrm{data}_1)(\nu\ \mathrm{email}_2)\\
\ (\mathrm{Server}\ \mid\ \mathrm{Customer} \mid\ \mathrm{gen}!^5[]\\
\ \mid\ \mathrm{email}_1!^4[\mathrm{data}_2]\ \mid\ \mathrm{email}_2!^4[\mathrm{data}_1])
\end{array}
$$



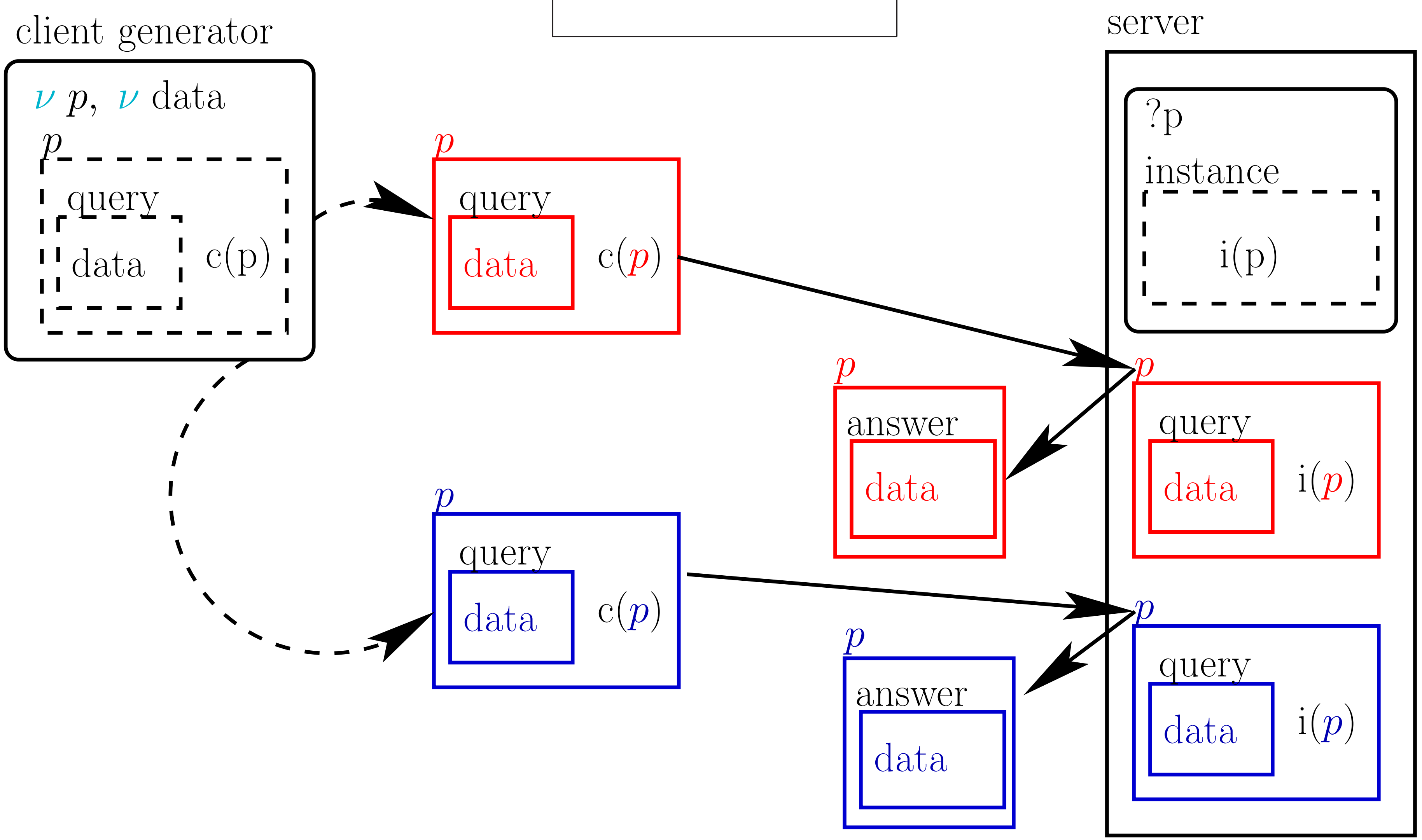
Mobile Ambients

Ambients are named boxes containing other ambients (and/or) some agents.

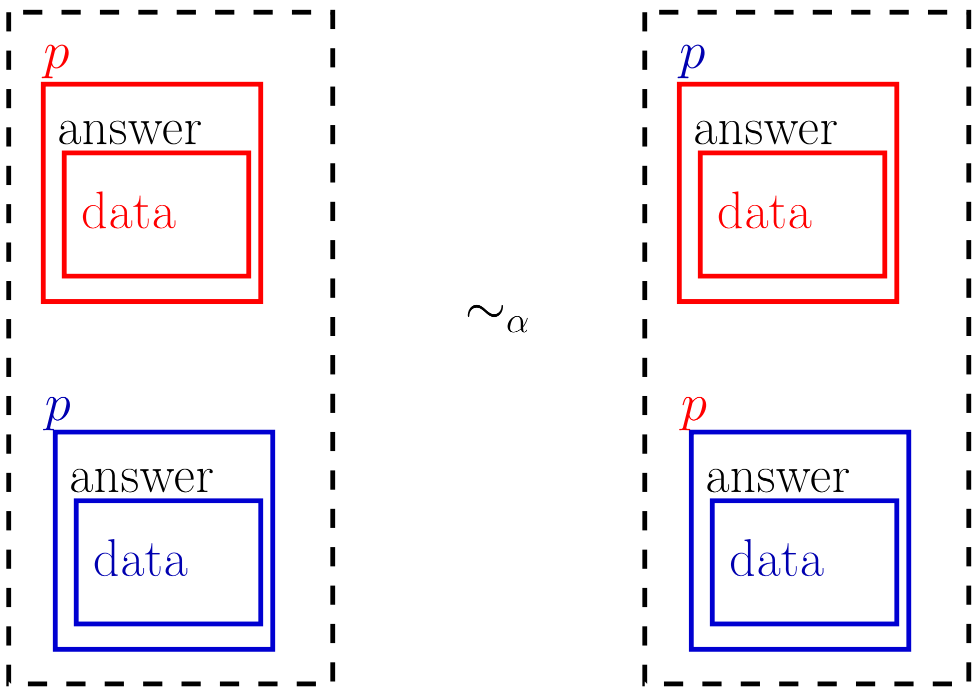
Agents:

- provide capabilities to their surrounding ambients for local migration and other ambient dissolution;
- dynamically create new ambients, names and agents;
- communicate names to each others.

An ftp-server



α-conversion



Non-standard Semantics

Non-standard semantics

A refined semantics in where

- recursive instances of processes are identified with unambiguous markers;
- channel names are stamped with the marker of the process which has declared them.

Example: non-standard configuration

$$(\mathrm{Server} \mid \mathrm{Customer} \mid \mathrm{gen}!^{5}[] \mid \mathrm{email}_1!^{2}[\mathrm{data}_1] \mid \mathrm{email}_2!^{2}[\mathrm{data}_2])$$

$$\left\{\begin{array}{l}
\left(1, \varepsilon, \left\{\mathrm{port} \mapsto (\mathrm{port}, \varepsilon)\right.\right) \\
\left(3, \varepsilon, \left\{\begin{array}{ll}\mathrm{gen} & \mapsto (\mathrm{gen}, \varepsilon) \\ \mathrm{port} & \mapsto (\mathrm{port}, \varepsilon)\end{array}\right.\right) \\
\left(2, \mathrm{id}'_1, \left\{\begin{array}{ll}\mathrm{add} & \mapsto (\mathrm{email}, \mathrm{id}_1) \\ \mathrm{info} & \mapsto (\mathrm{data}, \mathrm{id}_1)\end{array}\right.\right) \\
\left(2, \mathrm{id}'_2, \left\{\begin{array}{ll}\mathrm{add} & \mapsto (\mathrm{email}, \mathrm{id}_2) \\ \mathrm{info} & \mapsto (\mathrm{data}, \mathrm{id}_2)\end{array}\right.\right) \\
\left(5, \mathrm{id}_2, \left\{\mathrm{gen} \mapsto (\mathrm{gen}, \varepsilon)\right.\right)
\end{array}\right\}$$

Marker properties

1. Marker allocation must be consistent:

   Two instances of the same process cannot be associated the same marker during a computation sequence.

2. Marker allocation should be robust:

   Marker allocation should not depend on the interleaving order.

Marker allocation

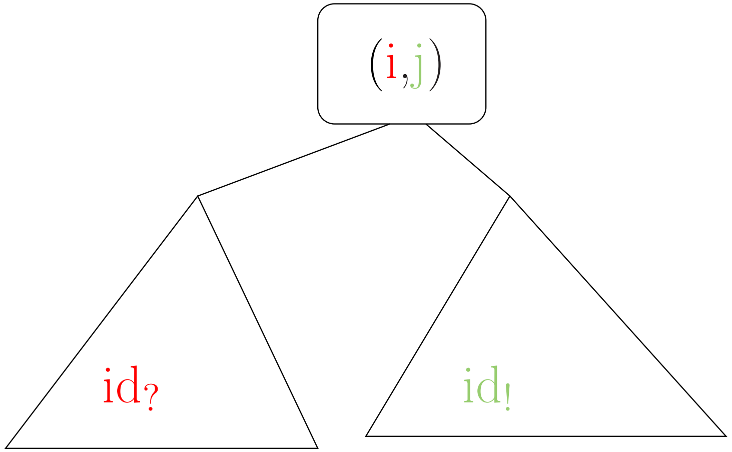
Markers describe the history of the replications which have led to the creation of the threads.

They are binary trees:

- leaves are not labeled;
- nodes are labeled with a pair $(i, j) \in \text{Label}^2$.

They are recursively calculated when fetching resources as follows:

id_* :



Small step semantics

Small step semantics is given by a transition system:

- an initial configuration;
- three structural reduction rules which simulate the congruence relation;
- four action reduction rules which simulate the transition relation.

Initial configuration

$$C_0(\mathcal{S}) = (\mathcal{S}, \varepsilon, \emptyset)$$

Structural rules

$$C \cup \{(P \mid Q, \mathrm{id}, E)\} \xrightarrow{\varepsilon} C \cup \{(P, \mathrm{id}, E_{|fn(P)}); (Q, \mathrm{id}, E_{|fn(Q)})\}$$

$$C \cup \{((\nu\ x)P, \mathrm{id}, E)\} \xrightarrow{\varepsilon} C \cup \{(P, \mathrm{id}, E[x \longrightarrow (x, \mathrm{id})]_{|fn(P)})\}$$

$$C \cup \{(\emptyset, \mathrm{id}, E)\} \xrightarrow{\varepsilon} C$$

Communication rules

$$\frac{E_?(y) = E_!(x)}{C \cup \left\{ \begin{array}{l} (y?^i[\overline{y}]P, \mathrm{id}_?, E_?); \\ (x!^j[\overline{x}]Q, \mathrm{id}_!, E_!) \end{array} \right\} \xrightarrow{(i,j)} C \cup \left\{ \begin{array}{l} (P, \mathrm{id}_?, E_?[\overline{y} \rightarrow E_![\overline{x}]]_{|fn(P)}); \\ (Q, \mathrm{id}_!, E_{!|fn(Q)}) \end{array} \right\}}$$

$$\frac{E_?(y) = E_!(x)}{C \cup \left\{ \begin{array}{l} (*y?^i[\overline{y}]P, \mathrm{id}_?, E_?); \\ (x!^j[\overline{x}]Q, \mathrm{id}_!, E_!) \end{array} \right\} \xrightarrow{(i,j)} C \cup \left\{ \begin{array}{l} (*y?^i[\overline{y}]P, \mathrm{id}_?, E_?); \\ (P, \mathrm{N}((i,j), \mathrm{id}_?, \mathrm{id}_!), E_?[\overline{y} \rightarrow E_!(\overline{x})]_{|fn(P)}); \\ (Q, \mathrm{id}_!, E_{!|fn(Q)}) \end{array} \right\}}$$

Choice rules

$$C \cup \{P+Q, \mathrm{id}, E\} \xrightarrow{\varepsilon} C \cup \{(P, \mathrm{id}, E_{|fn(P)}\}$$
$$C \cup \{P+Q, \mathrm{id}, E\} \xrightarrow{\varepsilon} C \cup \{(Q, \mathrm{id}, E_{|fn(Q)}\}$$

Coherence

Theorem 1 Standard semantics and small step non-standard semantics are weakly bisimilar.

The main point is to prove that there are no conflicts between markers.

Marker allocation consistency

We denote by $\text{father}(P)$ the father of P, when it exists, in the syntactic tree of $\mathcal{S}$.

1. the thread $(\mathcal{S}, \varepsilon, \emptyset)$ can only be created at the start of the system computation;
2. a thread $(P, \text{id}, \_)$ such that $\text{father}(P)$ is not a resource, can only be created by making a thread $(\text{father}(P), \text{id}, \_)$ react;
3. a thread $(P, \text{N}((i, j), \text{id}_?, \text{id}_!), \_)$ can only be created by making a thread $(P_j, \text{id}_!, \_)$ react (when P_j denote the syntactic process begining with the syntactic component labeled with j).

This proves marker allocation consistency.

Simplifying markers

We can simplify the shape of the marker without any loss of consistency:

1. replacing each tree by its right comb:

$$\begin{cases} \phi_1(\mathrm{N}((i,j),\mathrm{id}_1,\mathrm{id}_2)) & = \phi_1(\mathrm{id}_2).(i,j) \\ \phi_1(\varepsilon) & = \varepsilon \end{cases}$$

2. replacing pairs by their second component:

$$\begin{cases} \phi_2(\mathrm{N}((i,j),\mathrm{id}_1,\mathrm{id}_2)) = \phi_2(\mathrm{id}_2).j \\ \phi_2(\varepsilon) = \varepsilon \end{cases}$$

Those simplifications can be seen as an abstraction, they do not loose semantics consistency, but they may abstract away information, in the case of nested resources, by merging information about distinct computation sequences.

Middle semantics

Small step semantics can be analyzed but:

- there are too many transition rules;
- it uses too many kinds of processes.

$\Longrightarrow$ We design a new semantics with only active rules.

(Structural rules are included inside active rules)

Definition

Structural rules:

$$C \cup \{(P \mid Q, \mathrm{id}, E)\} \xrightarrow{\varepsilon} C \cup \{(P, \mathrm{id}, E_{|fn(P)}); (Q, \mathrm{id}, E_{|fn(Q)})\}$$

$$C \cup \{((\nu\ x)P, \mathrm{id}, E)\} \xrightarrow{\varepsilon} C \cup \{(P, \mathrm{id}, E[x \rightarrow (x, \mathrm{id})]_{|fn(P)})\}$$

$$C \cup \{(\emptyset, \mathrm{id}, E)\} \xrightarrow{\varepsilon} C$$

are a confluent and well-founded transition system,
we denote by $\Longrightarrow$ its limit:

$$a \Longrightarrow b \text{ ssi } \begin{cases} a \rightarrow^* b \\ \forall c, b \not\rightarrow c. \end{cases}$$

and we define our new transition system by $\xrightarrow{\lambda}{}' = \xrightarrow{\lambda} \circ \Longrightarrow$.

Extraction function

An extraction function calculates the set of the thread instances spawned at the beginning of the system execution or after a computation step.

$$
\begin{array}{rcl}
\beta((\nu\ n)P, \mathrm{id}, E) & = & \beta(P, \mathrm{id}, (E[n \mapsto (n, \mathrm{id})])) \\
\beta(\emptyset, \mathrm{id}, E) & = & \emptyset \\
\beta(P \mid Q, \mathrm{id}, E) & = & \beta(P, \mathrm{id}, E) \cup \beta(Q, \mathrm{id}, E) \\
\beta(P{+}Q, \mathrm{id}, E) & = & \{(P{+}Q, \mathrm{id}, E_{|fn(P+Q)})\} \\
\beta(y?^{i}[\overline{y}].P, \mathrm{id}, E) & = & \{(y?^{i}[\overline{y}].P, \mathrm{id}, E_{|fn(y?^{i}[\overline{y}].P)})\} \\
\beta(*y?^{i}[\overline{y}].P, \mathrm{id}, E) & = & \{(*y?^{i}[\overline{y}].P, \mathrm{id}, E_{|fn(*y?^{i}[\overline{y}].P)})\} \\
\beta(x!^{j}[\overline{x}].P, \mathrm{id}, E) & = & \{(x!^{j}[\overline{x}].P, \mathrm{id}, E_{|fn(x!^{j}[\overline{x}]P)})\}
\end{array}
$$

Transition system

$$C_0(\mathcal{S}) = \beta(\mathcal{S}, \varepsilon, \emptyset)$$

$$C \cup \{(P+Q, id, E)\} \xrightarrow{\varepsilon} (C \cup \beta(P, id, E))$$
$$C \cup \{(P+Q, id, E)\} \xrightarrow{\varepsilon} (C \cup \beta(Q, id, E))$$

Communication rules

$$\frac{E_?(y) = E_!(x)}{C \cup \left\{ \begin{array}{l} (y?^i[\overline{y}]P, id_?, E_?), \\ (x!^j[\overline{x}]Q, id_!, E_!) \end{array} \right\} \xrightarrow{(i,j)} (C \cup \beta(P, id_?, E_?[y_i \mapsto E_!(x_i)]) \cup \beta(Q, id_!, E_!))}$$

$$\frac{E_*(y) = E_!(x)}{C \cup \left\{ \begin{array}{l} (*y?^i[\overline{y}]P, id_*, E_*), \\ (x!^j[\overline{x}]Q, id_!, E_!) \end{array} \right\} \xrightarrow{(i,j)} \left(\begin{array}{l} C \cup \{(*y?^i[\overline{y}]P, id_*, E_*)\} \\ \quad \cup \beta(P, \mathrm{N}((i,j), id_*, id_!), E_*[y_i \mapsto E_!(x_i)]) \\ \quad \cup \beta(Q, id_!, E_!) \end{array} \right)}$$

Coherence

Middle semantics and standard semantics are strongly bisimilar, but we still consider too much process: we can also factor choice operations.

For that purpose we restrict our study to the computation sequences in where communication are only made when there are no choice thread instance at top level, and factor choices with communication rules.

Definition

Choice rules are a well-founded transition system,

we denote by $\Longrightarrow$ its non-deterministic limit:

$$a \Longrightarrow b \text{ ssi } \begin{cases} a \rightarrow^{*} b \\ \forall c, b \not\rightarrow c. \end{cases}$$

and we define our new transition system by $\xrightarrow{\lambda}{}' = \xrightarrow{\lambda} \circ \Longrightarrow$.

Extraction function

An extraction function calculates the set of all choices for the set of the thread instances spawned at the beginning of the system execution or after a communication.

$$
\begin{array}{rcl}
\beta((\nu\ n)P, \mathrm{id}, E) & = & \beta(P, \mathrm{id}, (E[n \mapsto (n, \mathrm{id})])) \\
\beta(\emptyset, \mathrm{id}, E) & = & \{\emptyset\} \\
\beta(P+Q, \mathrm{id}, E) & = & \beta(P, \mathrm{id}, E) \cup \beta(Q, \mathrm{id}, E) \\
\beta(P \mid Q, \mathrm{id}, E) & = & \{A \cup B \mid A \in \beta(P, \mathrm{id}, E),\ B \in \beta(Q, \mathrm{id}, E)\} \\
\beta(y?^i[\overline{y}].P, \mathrm{id}, E) & = & \{\{(y?^i[\overline{y}].P, \mathrm{id}, E_{|fn(y?^i[\overline{y}].P)})\}\} \\
\beta(*y?^i[\overline{y}].P, \mathrm{id}, E) & = & \{\{(*y?^i[\overline{y}].P, \mathrm{id}, E_{|fn(*y?^i[\overline{y}].P)})\}\} \\
\beta(x!^j[\overline{x}].P, \mathrm{id}, E) & = & \{\{(x!^j[\overline{x}].P, \mathrm{id}, E_{|fn(x!^j[\overline{x}]P)})\}\}
\end{array}
$$

Transition system

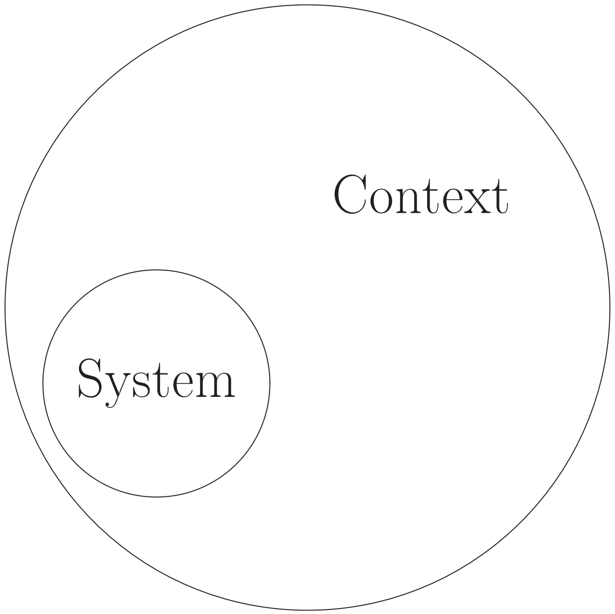
$$C_0(\mathcal{S}) = \beta(\mathcal{S}, \varepsilon, \emptyset)$$

$$\frac{E_?(y) = E_!(x),\ \ \mathrm{Cont}_P{\in}\beta(P, id_?, E_?[y_i \mapsto E_!(x_i)]), \mathrm{Cont}_Q{\in}\beta(Q, id_!, E_!)}{C \cup \{(y?^i[\overline{y}]P, id_?, E_?), (x!^j[\overline{x}]Q, id_!, E_!)\} \xrightarrow{(i,j)} (C \cup \mathrm{Cont}_P \cup \mathrm{Cont}_Q)}$$

$$\frac{E_*(y) = E_!(x),\ \ \mathrm{Cont}_P{\in}\beta(P, \mathrm{N}((i,j), id_*, id_!), E_*[y_i \mapsto E_!(x_i)]), \mathrm{Cont}_Q{\in}\beta(Q, id_!, E_!)}{C \cup \left\{ \begin{array}{l} (*y?^i[\overline{y}]P, id_*, E_*), \\ (x!^j[\overline{x}]Q, id_!, E_!) \end{array} \right\} \xrightarrow{(i,j)} \left(C \cup \{(*y?^i[\overline{y}]P, id_*, E_*)\} \cup \mathrm{Cont}_P \cup \mathrm{Cont}_Q\right)}$$

Context-free analysis

Analyzing interaction between a system and its unknown context.



The context may

- spy the system, by listening to message on unsafe channel names;
- spoil the system, by sending message via unsafe channel names.

Nasty context

$$\text{Context} := (\nu\ \text{unsafe})\ (\text{new} \mid \text{spy}_0 \mid \ldots \mid \text{spy}_n \mid \text{spoil}_0 \mid \ldots \mid \text{spoil}_n\ )$$

where

$$\text{new} := (*(\nu\ \text{channel}) * \text{unsafe}![\text{channel}])$$

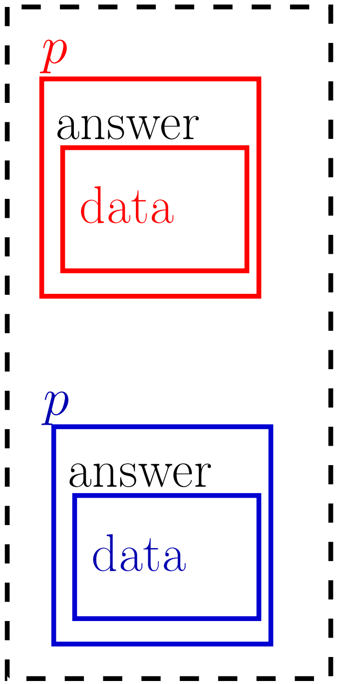
$$\text{spoil}_k := (*\text{unsafe}?[c]\text{unsafe}?[x_1] \ldots \text{unsafe}?[x_k]c![x_1,\ldots,x_k])$$

$$\text{spy}_k := (*\text{unsafe}?[c]\text{c}?[x_1,\ldots,x_k]\ ((*\text{unsafe}![x_1]) \mid \ldots \mid (*\text{unsafe}![x_k])))$$

Non-Standard Configuration

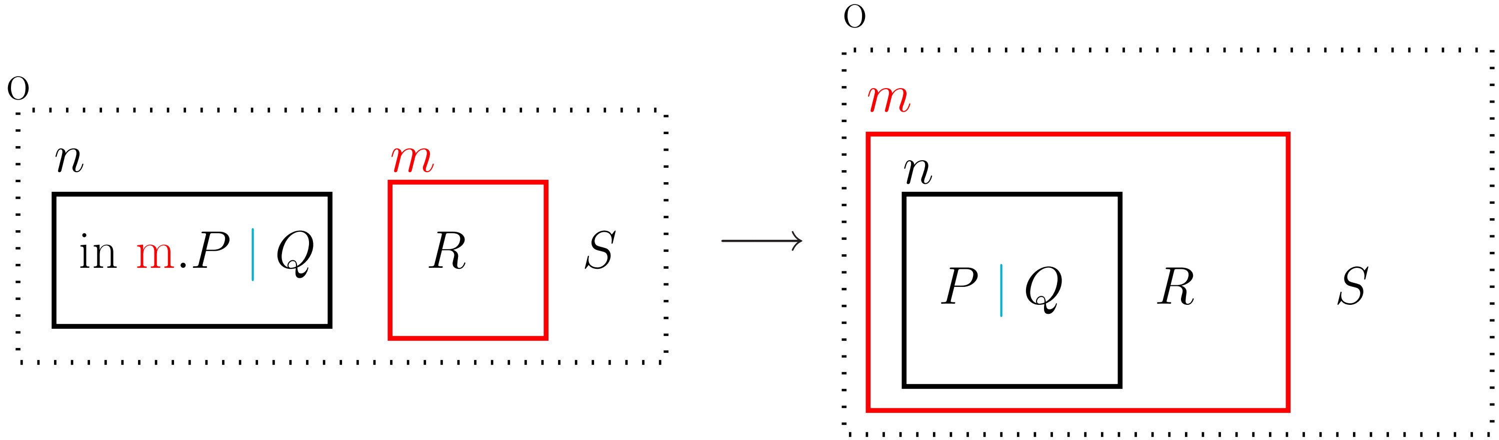
We flatly represent system configurations:

$$\begin{cases}
(p^{12}[\bullet], \mathrm{id}_0, (\mathrm{top}, \varepsilon), [p \mapsto (p, \mathrm{id}_0)]) \\
(p^{12}[\bullet], \mathrm{id}_1, (\mathrm{top}, \varepsilon), [p \mapsto (p, \mathrm{id}_1)]) \\
(\mathrm{answer}^{8}[\bullet], \mathrm{id}'_0, (12, \mathrm{id}_0), \emptyset) \\
(\mathrm{answer}^{8}[\bullet], \mathrm{id}'_1, (12, \mathrm{id}_1), \emptyset) \\
\hline
(\langle \mathrm{rep} \rangle^{9}, \mathrm{id}'_0, (8, \mathrm{id}'_0), [\mathrm{rep} \mapsto (\mathrm{data}, \mathrm{id}_0)]) \\
(\langle \mathrm{rep} \rangle^{9}, \mathrm{id}'_1, (8, \mathrm{id}'_1), [\mathrm{rep} \mapsto (\mathrm{data}, \mathrm{id}_1)])
\end{cases}$$



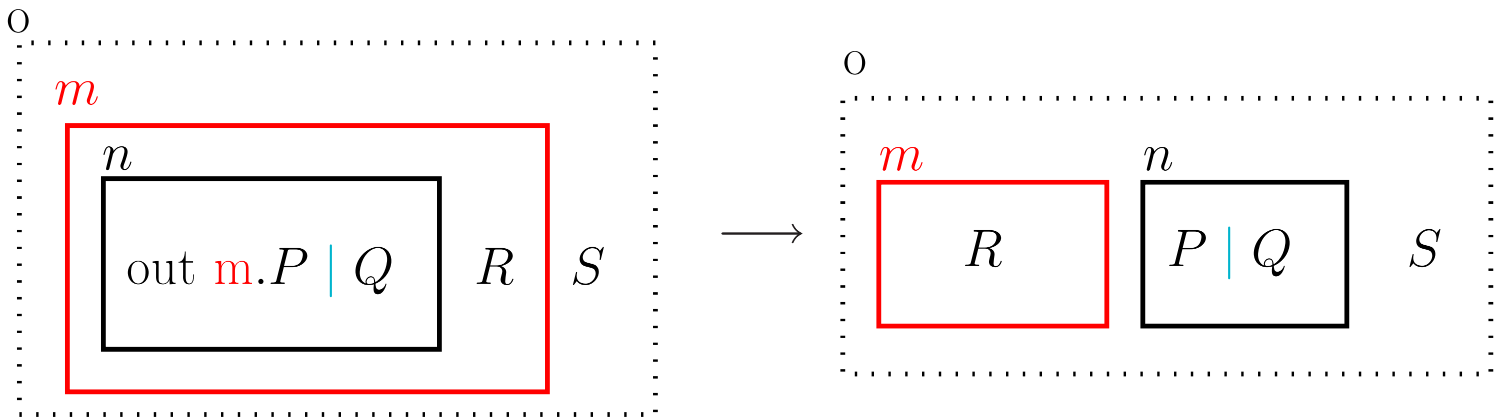
In migration

$$\frac{\begin{cases}\lambda = \left(n^i[\bullet], \mathrm{id}_1, \mathrm{loc}_1, E_1\right), \\ \mu = \left(m^j[\bullet], \mathrm{id}_2, \mathrm{loc}_2, E_2\right), \\ \psi = \left(\mathrm{in}^k o.P, \mathrm{id}_3, \mathrm{loc}_3, E_3\right), \\ \mathrm{loc}_1 = \mathrm{loc}_2,\ \ \mathrm{loc}_3 = (i, \mathrm{id}_1),\ \ E_2(m) = E_3(o),\ \ \lambda \neq \mu. \end{cases}}{C \cup \{\lambda; \mu; \psi\} \stackrel{\mathrm{in}(i,j,k)}{\longrightarrow} (C \cup \{\mu\}) \cup (n^i[\bullet], \mathrm{id}_1, (j, \mathrm{id}_2), E_1) \cup \beta\left(P, \mathrm{id}_3, \mathrm{loc}_3, E_{3|fn(P)}\right).}$$



out migration

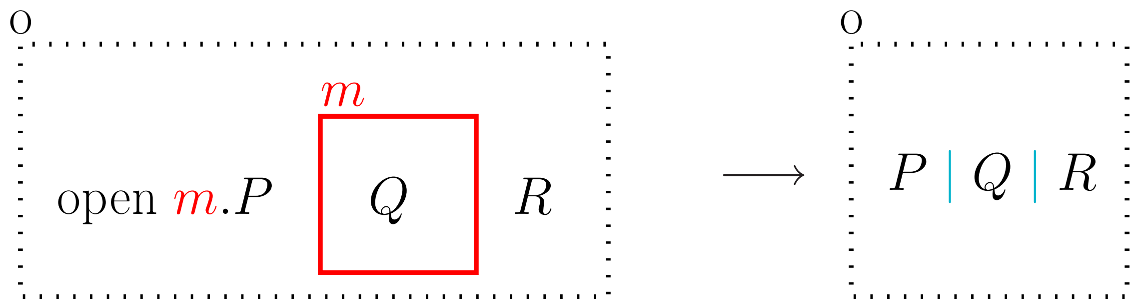
$$\frac{\begin{cases}\lambda = \left(m^i[\bullet], \mathrm{id}_1, \mathrm{loc}_1, E_1\right), \\ \mu = \left(n^j[\bullet], \mathrm{id}_2, \mathrm{loc}_2, E_2\right), \\ \psi = \left(\mathrm{out}^k o.P, \mathrm{id}_3, \mathrm{loc}_3, E_3\right), \\ \mathrm{loc}_2 = (i, \mathrm{id}_1),\ \ \mathrm{loc}_3 = (j, \mathrm{id}_2),\ \ E_1(m) = E_3(o)\end{cases}}{C \cup \{\lambda; \mu; \psi\} \overset{\mathrm{out}(i,j,k)}{\longrightarrow} (C \cup \{\lambda\}) \cup (n^j[\bullet], \mathrm{id}_2, \mathrm{loc}_1, E_2) \cup \beta\left(P, \mathrm{id}_3, \mathrm{loc}_3, E_{3|fn(P)}\right).}$$



Dissolution

$$\frac{\begin{cases}\lambda = \left(\mathrm{open}^i m.P, \mathrm{id}_1, \mathrm{loc}_1, E_1\right)\\ \mu = \left(n^j[\bullet], \mathrm{id}_2, \mathrm{loc}_2, E_2\right),\\ \mathrm{loc}_1 = \mathrm{loc}_2,\ E_1(m) = E_2(n),\end{cases}}{C \cup \{\lambda;\mu\} \overset{\mathrm{open}(i,j)}{\longrightarrow} (C \setminus A) \cup A' \cup \beta\left(P, \mathrm{id}_1, \mathrm{loc}_1, E_{1|fn(P)}\right)}$$

$$\text{where } \begin{cases} A = \{(a, \mathrm{id}, \mathrm{loc}, E) \in C \mid \mathrm{loc} = (j, \mathrm{id}_2)\}\\ A' = \{(a, \mathrm{id}, \mathrm{loc}_2, E) \mid (a, \mathrm{id}, (j, \mathrm{id}_2), E) \in C\}\,. \end{cases}$$



Abstraction

Collecting semantics

$(\mathcal{C}, C_0, \longrightarrow)$ is a transition system,
We restrict our study to its collecting semantics:
this is the set of the states which are reachable within a finite transition sequence.

$$\mathcal{S} = \{C \mid \exists i \in C_0,\ i \longrightarrow^* C\}$$

It is also given by the least fix-point of the following $\cup$-complete endomorphism $\mathbb{F}$:

$$\mathbb{F} = \begin{cases} \wp(\mathcal{C}) & \longrightarrow \wp(\mathcal{C}) \\ X & \longmapsto C_0 \cup \{C' \mid \exists C \in X,\ C \rightarrow C'\} \end{cases}$$

The calculus of this fix point is not usually decidable.

Abstract domain

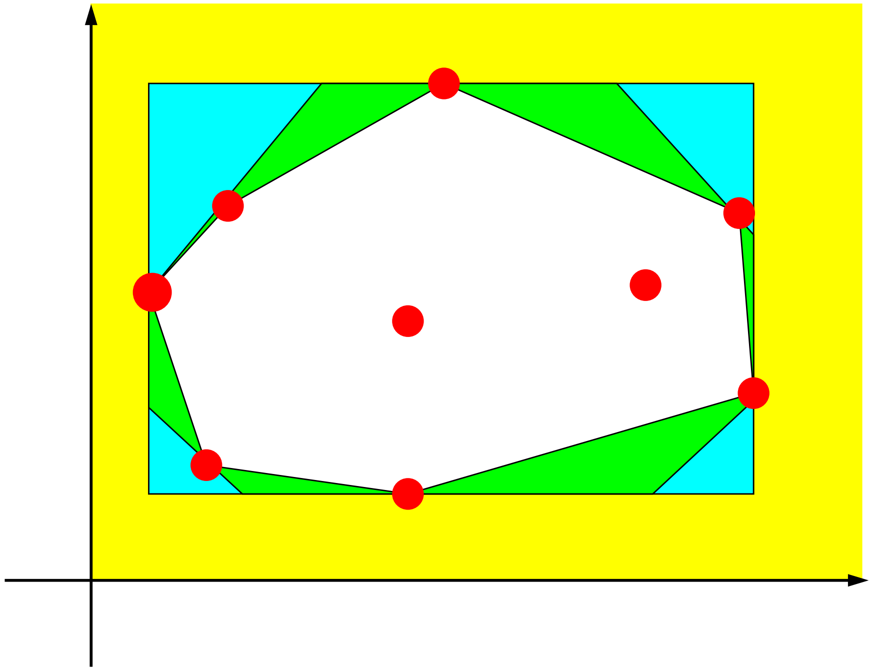
We introduce an abstract domain of properties:

- properties of interest;
- more complex properties used in calculating them.

This domain is often a binary lattice: $(\mathcal{D}^\sharp, \sqsubseteq, \sqcup, \bot, \sqcap, \top)$ and is related to the concrete domain $\wp(\mathcal{C})$ by a monotonic concretization function γ.

$\forall A \in \mathcal{D}^\sharp$, $\gamma(A)$ is the set of the elements which satisfies the property A.

Numerical domains



- sign approximation;
- interval approximation;
- octagonal approximation;
- polyhedra approximation;
- concrete domain.

Abstract transition system

Let $C_0^\sharp$ be an abstraction of the initial states and $\rightsquigarrow$ be an abstract relation of transition, which satisfy $C_0 \subseteq \gamma(C_0^\sharp)$ and the following diagram:

$$\begin{tikzcd} C^\sharp \arrow[r, "\overset{\lambda}{\rightsquigarrow}"] & \overline{C}^\sharp \\ C \arrow[u, "\gamma"] \arrow[r, "\overset{\lambda}{\rightarrow}"'] & \overline{C} \arrow[u, "\gamma"'] \end{tikzcd}$$

Then, $\mathcal{S} \subseteq \bigcup_{n\in\mathbb{N}} \gamma({\mathbb{F}^\sharp}^n(C_0^\sharp))$ where $\mathbb{F}^\sharp(C^\sharp) = C_0^\sharp \sqcup C^\sharp \sqcup \left(\bigsqcup_{\text{finite}}\{\overline{C^\sharp} \mid C^\sharp \rightsquigarrow \overline{C^\sharp}\}\right)$.

Widening operator

We require a widening operator to ensure the convergence of the analysis:

$$\nabla \ : \ D^\sharp \times D^\sharp \rightarrow D^\sharp$$

such that:

- $\forall X_1^\sharp,\ X_2^\sharp \in D^\sharp,\ X_1^\sharp \sqcup X_2^\sharp \sqsubseteq X_1^\sharp \nabla X_2^\sharp$
- for all increasing sequence $(X_n^\sharp) \in \left(D^\sharp\right)^{\mathbb{N}}$, the sequence (X_n^∇) defined as

$$\begin{cases} X_0^\nabla = X_0^\sharp \\ X_{n+1}^\nabla = X_n^\nabla \ \nabla \ X_{n+1}^\sharp \end{cases}$$

  is ultimately stationary.

Abstract iteration

The abstract iteration (C_n^{∇}) of $\mathbb{F}^{\sharp}$ defined as follows

$$\begin{cases} C_0^{\nabla} = C_0^{\sharp} \\ C_{n+1}^{\nabla} = \begin{cases} C_n^{\nabla} & \text{if } \mathbb{F}^{\sharp}(C_n^{\nabla}) \sqsubseteq C_n^{\nabla} \\ C_n^{\nabla} \; \nabla \; \mathbb{F}^{\sharp}(C_n^{\nabla}) & \text{otherwise} \end{cases} \end{cases}$$

is ultimately stationary and its limit C^{∇} satisfies $\mathrm{lfp}_{\emptyset}\mathbb{F} \subseteq \gamma(C^{\nabla})$.

Example: Interval widening

We consider the complete $\mathcal{I}$ lattice of the natural number intervals.

$\mathcal{I}$ does not satisfy the increasing chain condition.

Given n a natural number, we use the following widening operator to ensure the convergence of the analyses based on the use of $\mathcal{I}$:

$$\begin{cases} [|a;b|] \ \nabla \ [|c;d|] = [|\min\{a;c\};\infty|[& \text{if } d > \max\{n;b\} \\ I \ \nabla \ J = I \sqcup J & \text{otherwise} \end{cases}$$

Composing two abstractions

Given two abstractions $(\mathcal{D}^\sharp, \gamma, C_0^\sharp, \rightsquigarrow, \nabla)$ and $(\mathcal{D}^\sharp, \gamma, C_0^\sharp, \rightsquigarrow, \nabla)$, and a reduction $\rho : \mathcal{D}^\sharp \times \mathcal{D}^\sharp \rightarrow \mathcal{D}^\sharp \times \mathcal{D}^\sharp$ which satisfy:

$$\forall (A, A) \in \mathcal{D}^\sharp \times \mathcal{D}^\sharp,\ \gamma(A) \cap \gamma(A) \subseteq \gamma(a) \cap \gamma(a) \text{ where } (a, a) = \rho(A, A).$$

Then $(\mathcal{D}^\sharp, \gamma, C_0^\sharp, \rightsquigarrow, \nabla)$ where:

- $\mathcal{D}^\sharp = \mathcal{D}^\sharp \times \mathcal{D}^\sharp$;
- ∇ is pair-wisely defined;
- $\gamma(A, A) = \gamma(A) \cap \gamma(A)$;
- $C_0^\sharp = \rho(C_0^\sharp, C_0^\sharp)$;
- $(A, A) \rightsquigarrow \rho(C, C)$
  if $B \rightsquigarrow C$ and $B \rightsquigarrow C$ and $(B, B) = \rho(A, A)$

is also an abstraction.

Generic environment analysis

For each subset V of variables, we introduce a generic abstract domain $\mathcal{G}_V$ to describe the markers and the environments which may be associated to a syntactic component the free name of which is V:

$$\wp(\mathrm{Id} \times (V \to (\mathrm{Name} \times \mathrm{Id}))) \xleftarrow{\gamma_V} \mathcal{G}_V.$$

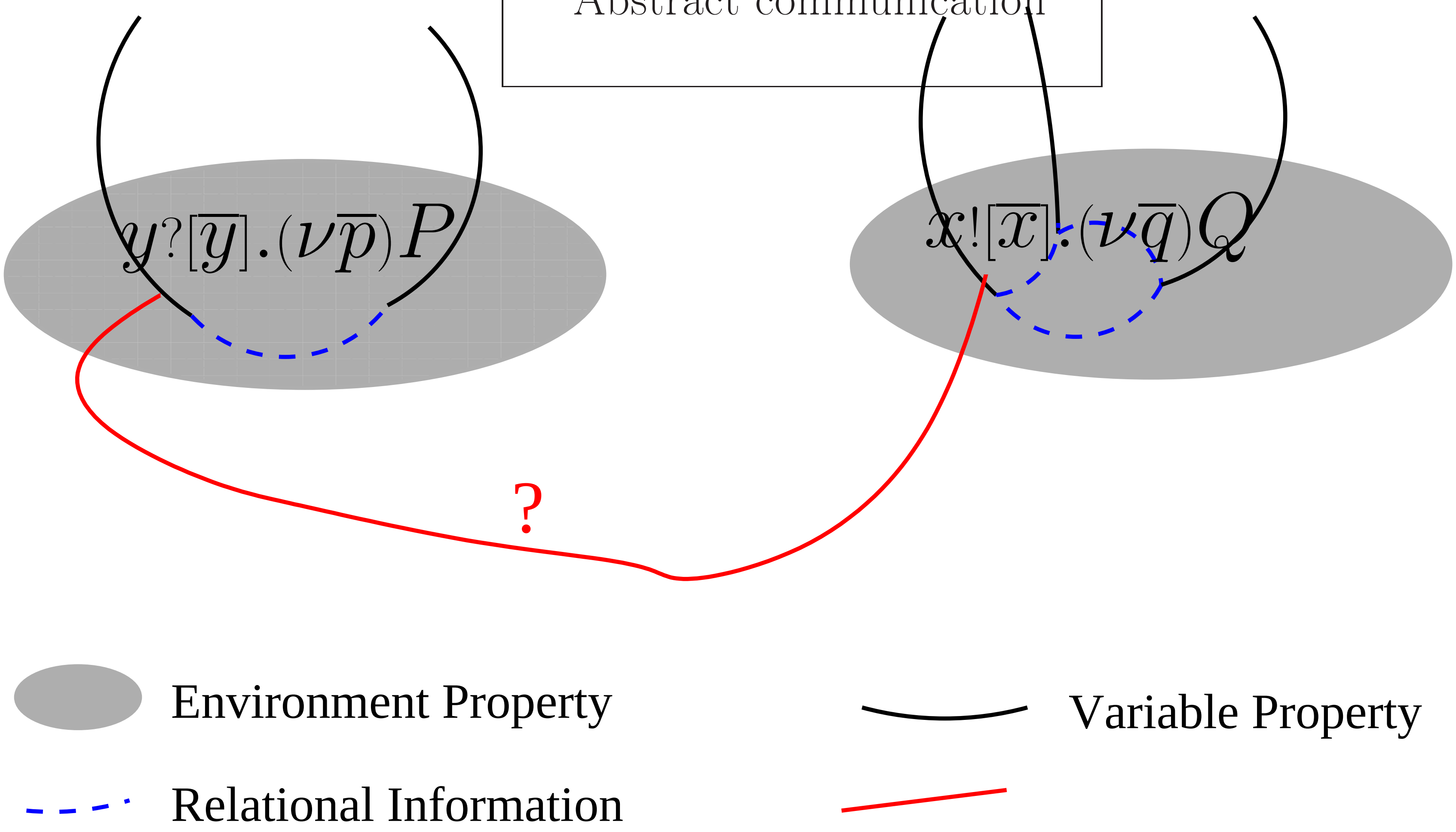
The abstract domain $C^\sharp$ is then the set:

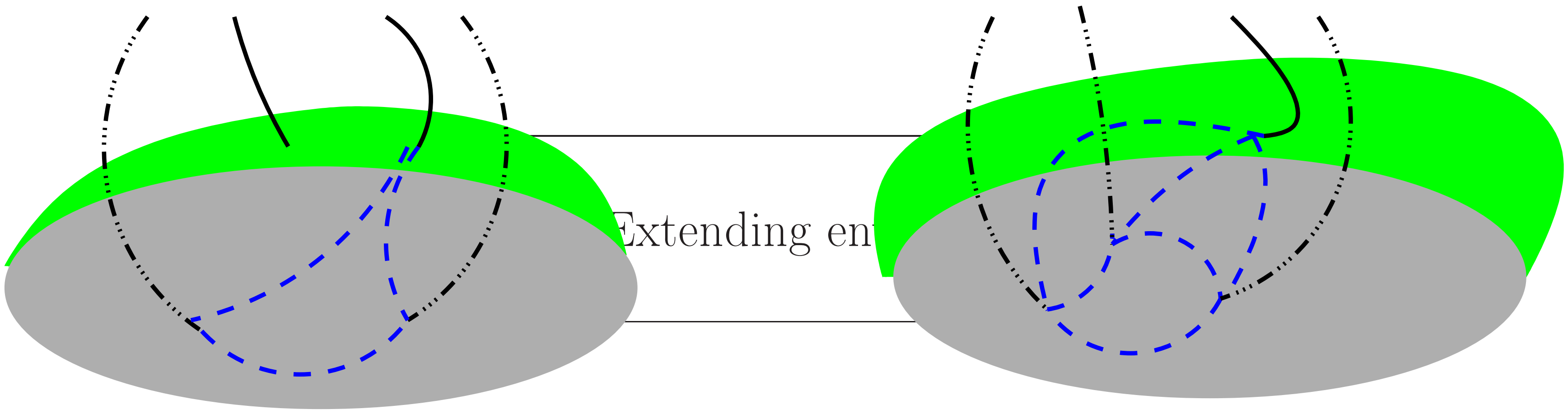
$$C^\sharp = \prod_{p \in \mathcal{P}} \mathcal{G}_{fn(p)}$$

related to $\wp(C)$ by the concretization γ:

$$\gamma(f) = \{C \mid (p, \mathrm{id}, E) \in C \implies (\mathrm{id}, E) \in \gamma_{fn(p)}(f_p)\}.$$

Abstract communication





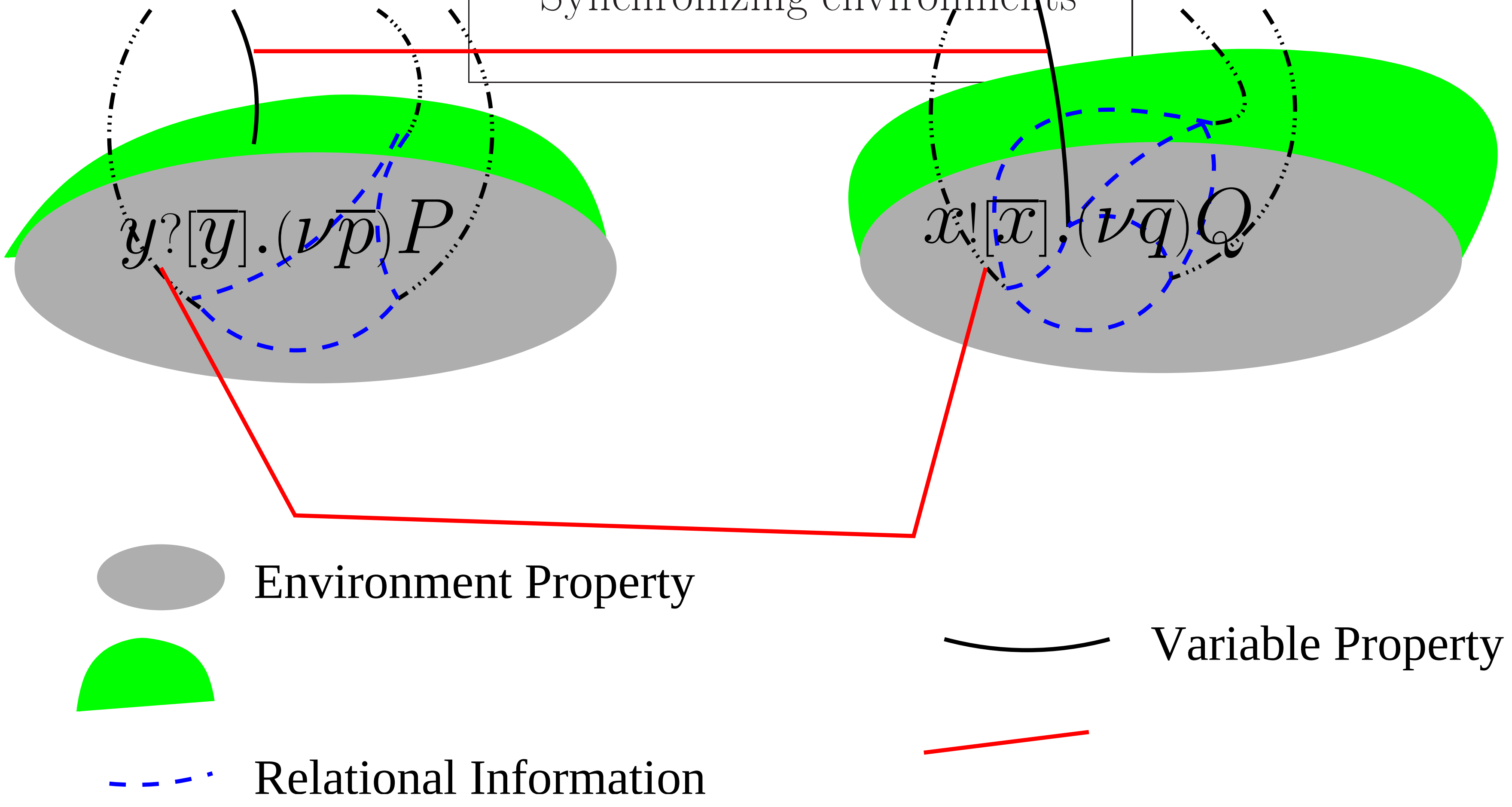
$y?[\overline{y}].(\nu\overline{p})P$ $\qquad$ $x![\overline{x}].(\nu\overline{q})Q$

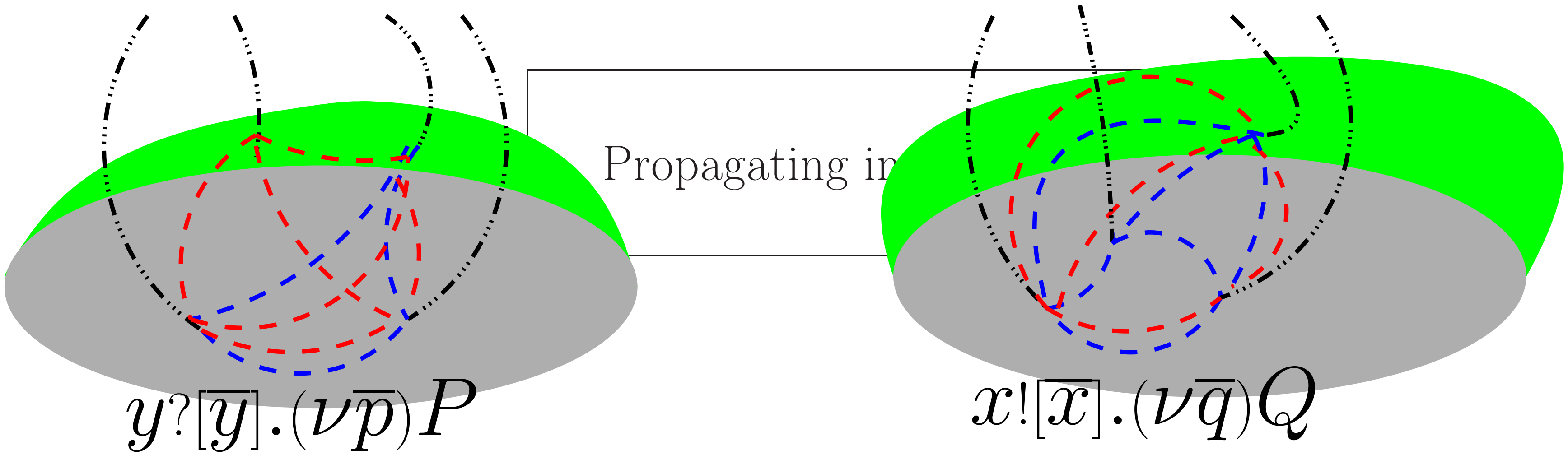
Environment Property

Variable Property

Relational Information

Synchronizing environments





Environment Property

Variable Property

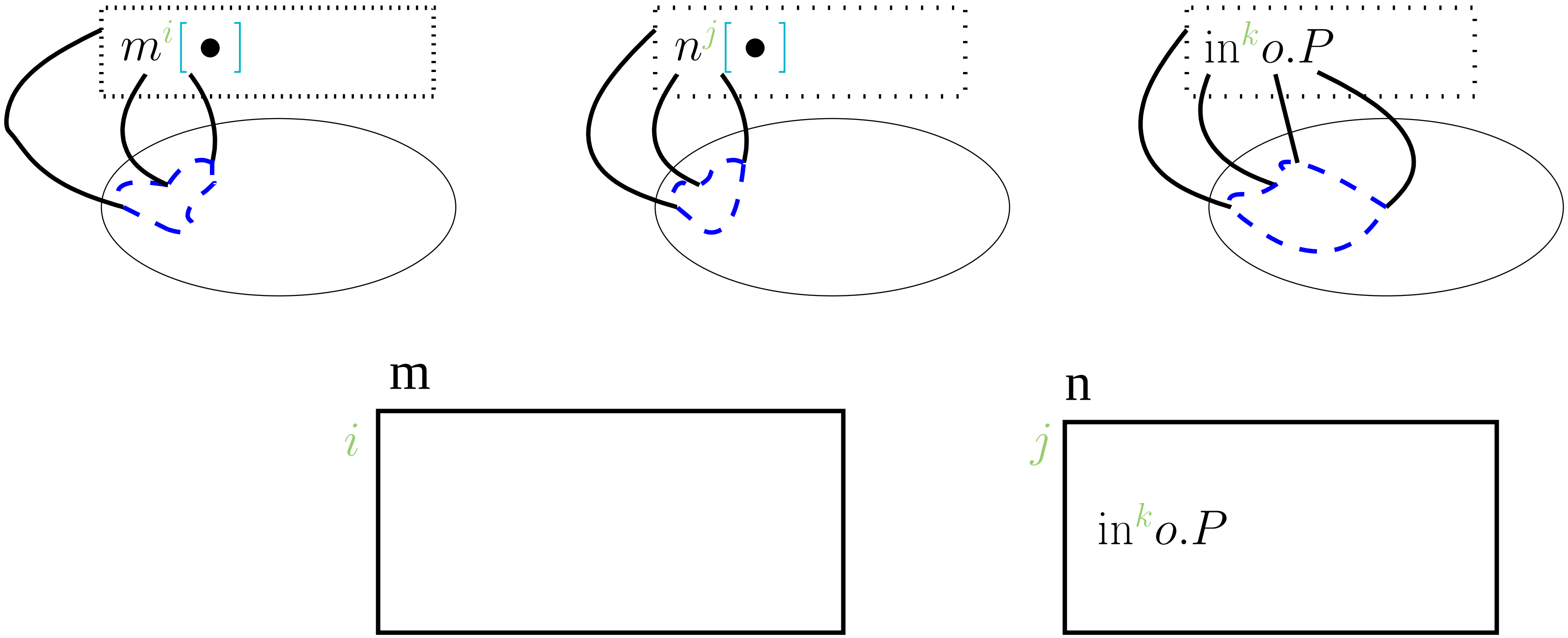
Relational Information

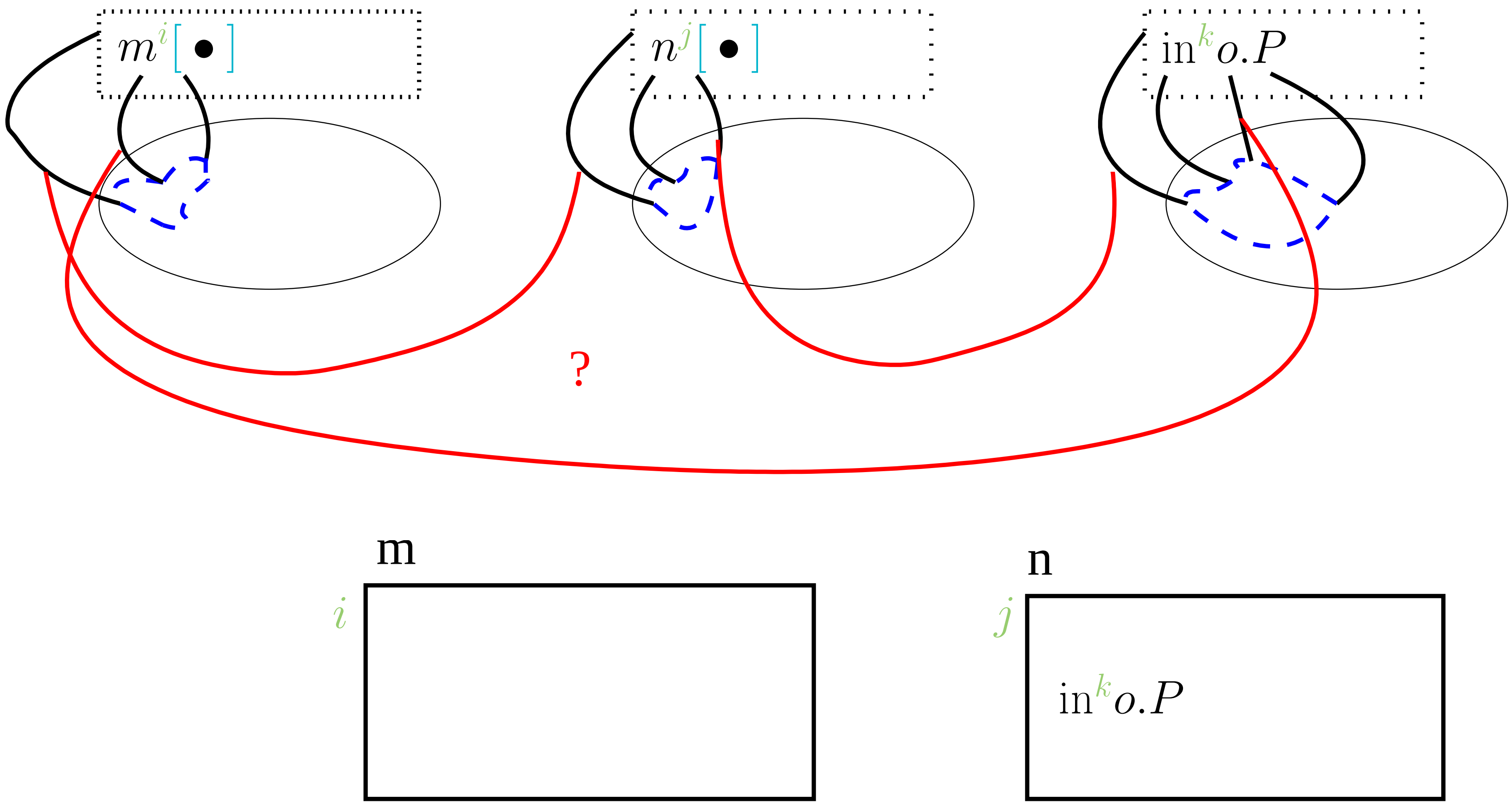
Generic primitives

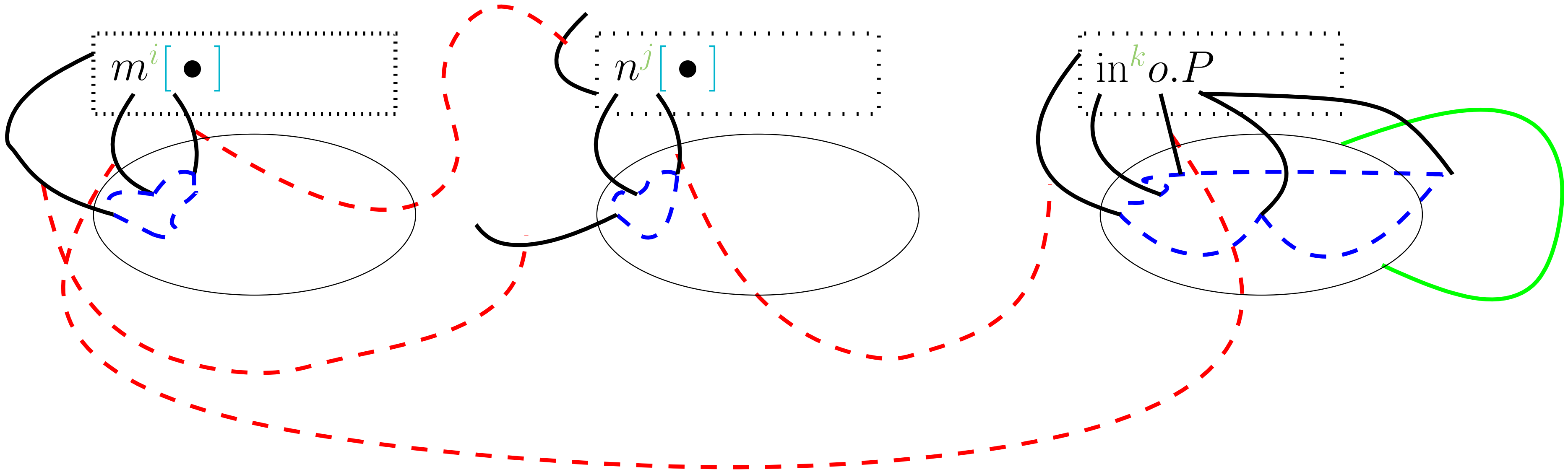
We only require abstract primitives to

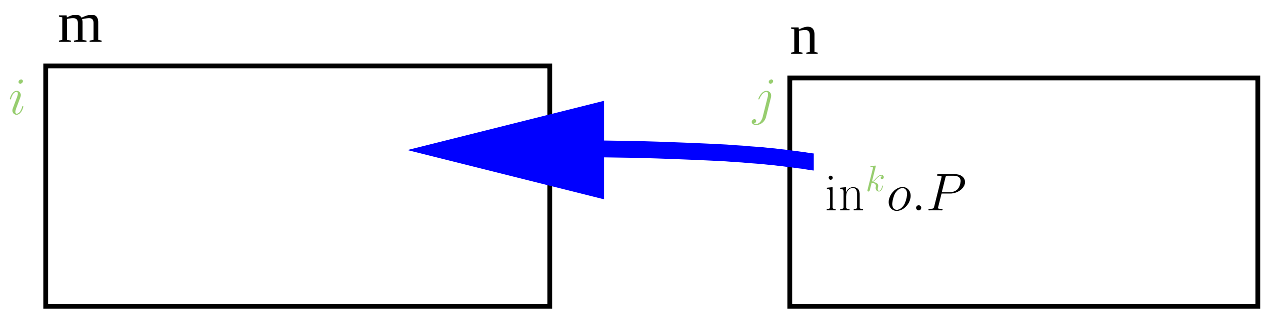
1. check whether the communication is enable,
2. extend an environment domain,
3. gather the description of the linkage of two syntactic agents,
4. synchronize variables,
5. separate two descriptions,
6. restrict an environment domain.

About mobile ambients




$m^i[\bullet]$
$n^j[\bullet]$
$\mathrm{in}^k o.P$
?
m
i
n
j
$\mathrm{in}^k o.P$


$m^i[\bullet]$
$n^j[\bullet]$
$\text{in}^k o.P$


m
i
n
j
$\text{in}^k o.P$

0-CFA

We set $\mathcal{G}_V = V \rightarrow \wp(\text{Name})$ related to $\wp(\text{Id} \times (V \rightarrow (\text{Name} \times \text{Id})))$ by the following concretization function:

$$\gamma_V(f) = \{(\text{id}, E) \mid \forall v \in V,\ \text{fst}(E(v)) \in f(v)\}$$

This analysis computes sound approximative answers to the question:

- may a name created by an instance of the name restriction $(\nu\ x)$ be communicated to the variable y of an instance of the thread P ?

Regular approximation

We approximate the shape of the markers which may be associated to channel names linked to variables, and syntactic components, without relations among them.
We use the following abstract domain:

$$\wp(\Sigma) \times \wp(\Sigma) \times \wp(\Sigma \times \Sigma) \times \{\text{true;false}\}.$$

$\gamma(I, F, T, b)$ is defined by $\gamma_1(I) \cap \gamma_2(F) \cap \gamma_3(T) \cap \gamma_4(b)$ where:

- $\gamma_1(I) = \{u \in \Sigma^* \mid |u| > 0 \Rightarrow u_1 \in I\}$,
- $\gamma_2(F) = \{u \in \Sigma^* \mid |u| > 0 \Rightarrow u_{|u|} \in F\}$,
- $\gamma_3(T) = \{u \in \Sigma^* \mid \forall a, b \in \Sigma^*, \lambda, \mu \in \Sigma,\ u = a.\lambda.\mu.b \Rightarrow (\lambda, \mu) \in T\}$,
- $\gamma_4(b) = \begin{cases} \Sigma^+ & \text{if } b = 0 \\ \Sigma^* & \text{otherwise.} \end{cases}$

Comparison between channel and agent markers

For each triplet (P, y, x), we approximate the relations between the markers of P and the markers of the names created by an instance of the restriction $(\nu\ x)$ and communicated to y.
We capture the difference between the occurrence number of letters in such two markers.

$$(\Sigma \rightarrow (\mathbb{N} \cup \{\top\})) \cup \{\bot\}$$

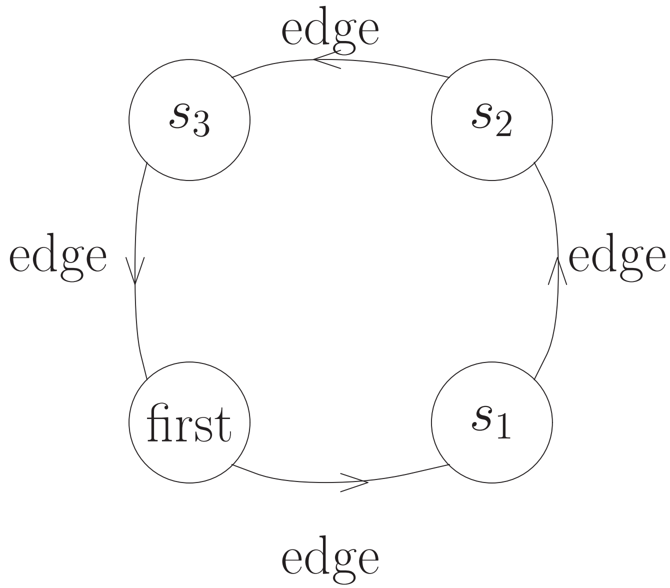
γ is defined as follows:
$\gamma(\bot) = \emptyset$
$\gamma(f) = \{(u, v) \in (\Sigma^*)^2 \mid \forall \lambda,\ f(\lambda) \in \mathbb{N} \implies |u|_\lambda - |v|_\lambda = f(n)\}.$

Example: 0-CFA

Example: Non-uniform result

Example: the ring of processes

$$
\begin{array}{l}
(\nu\ \text{make})(\nu\ \text{edge})(\nu\ \text{first}) \\
\quad (*\text{make}?^{1}[\text{last}](\nu \text{next}) \\
\qquad\qquad\qquad (\text{edge}!^{2}[\text{last},\text{next}] \\
\qquad\qquad\qquad \mid\ \ \text{make}!^{3}[\text{next}]) \\
\quad \mid\ \ *\text{make}?^{4}[\text{last}](\text{edge}!^{5}[\text{last},\text{first}]) \\
\quad \mid\ \ \text{make}!^{6}[\text{first}])
\end{array}
$$



$$\underline{\sharp}(1,3)+1 = \underline{\sharp}(1,3)$$

Example: Algebraic properties

Example

We detect that:

$$\begin{cases} (p^{12}[\bullet], (11,20)^m.(11,21), \_, [p \mapsto (p, (11,20)^m.(11,21))]) \\ (\text{answer}^8[\bullet], (3,19).(11,20)^n.(11,21), (12, (11,20)^n.(11,21)), \_) \\ (\langle \text{rep} \rangle^9, \_, (8, (3,19).(11,20)^p.(11,21)), [\text{rep} \mapsto (\text{data}, (11,20)^p.(11,21))]) \end{cases}$$

We deduce that each packet exiting the server has the following structure:

$(p.(11,20)^n.(11,21))$

$(11,20)^n.(11,21)$

answer

$(3,19).(11,20)^n.(11,21)$

$\boxed{(\text{data}, (11,20)^n.(11,21))}$

Limitations

Two main drawbacks:

1. we only prove equalities between Parrikh's vectors, some more work is needed in order to prove equalities of words;
2. we only capture properties involving comparison between channel name and agent markers:

$$
\begin{array}{l}
(\nu\ \mathrm{make})(\nu\ \mathrm{edge})(\nu\ \mathrm{first}) \\
\quad (*\mathrm{make}?^1[\mathrm{last}](\nu \mathrm{next}) \\
\qquad\qquad\qquad (\mathrm{edge}!^2[\mathrm{last},\mathrm{next}] \\
\qquad\qquad\qquad \mid\ \mathrm{make}!^3[\mathrm{next}]) \\
\quad \mid\ *\mathrm{make}?^6[\mathrm{last}](\mathrm{edge}!^7[\mathrm{last},\mathrm{first}]) \\
\quad \mid\ \mathrm{make}!^8[\mathrm{first}]) \\
\quad \mid\ \mathrm{edge}?[x,y][x =^9 y][x \neq^{10} \mathrm{first}]\mathrm{Ok}!^{11}[]
\end{array}
$$

we cannot infer that 11 is unreachable.

Dependency analysis

We describe equality and inequality relations between the names linked to variables.

$$\mathcal{G}_V = \left\{ (A,R) \;\middle|\; \begin{array}{l} A \text{ is a partition of } V \\ R \text{ is a symetric anti-reflexive relation on } A \end{array} \right\}.$$

$\mathcal{G}_V$ is related to $\wp(\mathrm{Id} \times (V \rightarrow (\mathrm{Name} \times \mathrm{Id})))$ by the following concretization function:

$$\gamma_V((A,R)) = \left\{ (\mathrm{id}, E) \;\middle|\; \begin{array}{l} \forall \mathcal{X} \in A,\ \{x,y\} \subseteq \mathcal{X} \implies E(x) = E(y) \\ (\mathcal{X},\mathcal{Y}) \in R \implies \forall x \in \mathcal{X}, y \in \mathcal{Y},\ E(x) \neq E(y) \end{array} \right\}$$

$\Longrightarrow$ implicit closure of relations and information propagation.

Global numerical analysis

We abstract relations between all the name markers and all the names linked to variables, and the thread markers:
For each $V \subseteq \mathrm{Name}$, we introduce the set

$$\mathcal{X}_V = \{p^\lambda \mid \lambda \in \Sigma\} \cup \{c^{(\lambda,v)} \mid \lambda \in \Sigma \cup \mathrm{Name},\ v \in V\}$$

The domain $\mathcal{G}_V$ is then the set of the affine relations system among $\mathcal{X}_V$ related to the concrete domain by the following concretization:

$$\gamma_V(\mathcal{K}) = \left\{ (\mathrm{id}, E) \;\middle|\; \begin{pmatrix} p^\lambda \longrightarrow |\mathrm{id}|_\lambda \\ x^{(y,v)} \longrightarrow (x = \mathrm{first}(E(v))) \\ x^{(\lambda,v)} \longrightarrow |\mathrm{snd}(E(v))|_\lambda \end{pmatrix} \text{ satisfies } \mathcal{K} \right\}.$$

Pair-wise numerical analysis

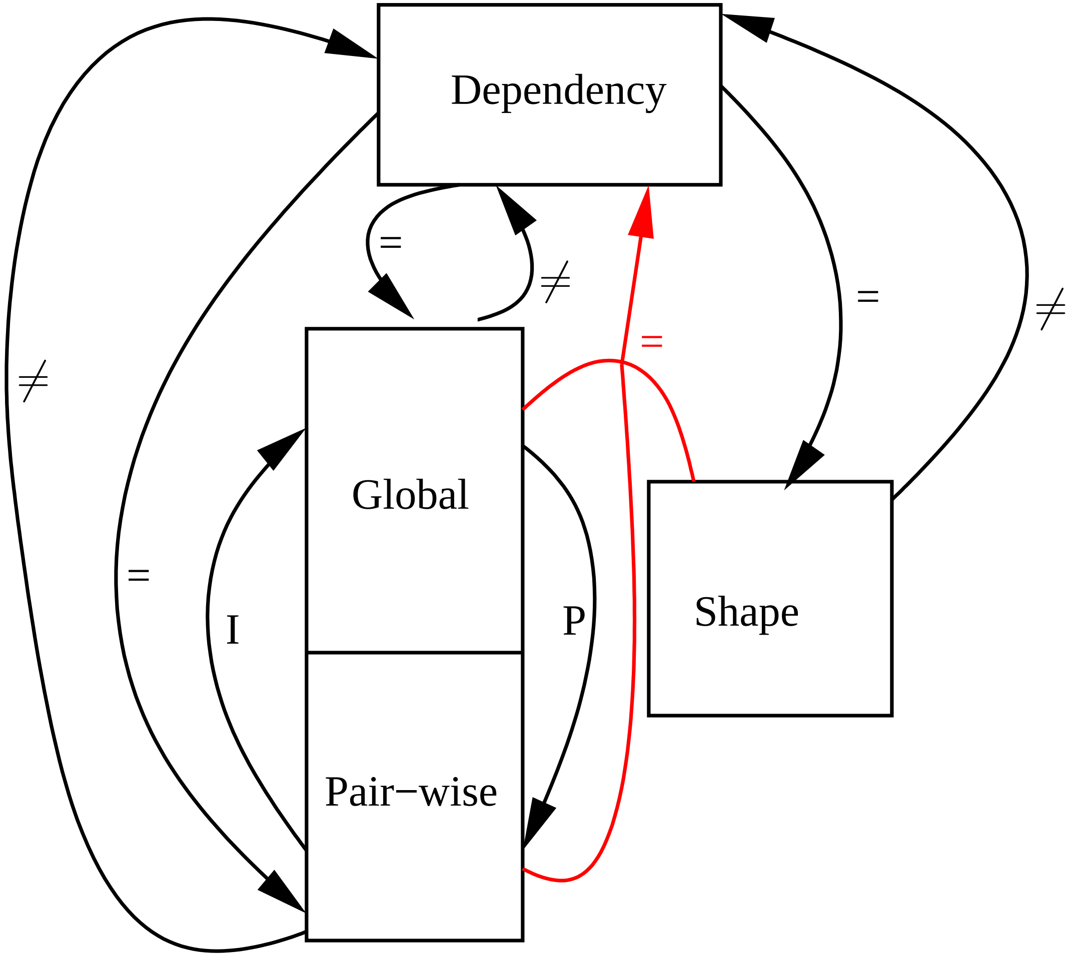
We compare pair-wisely markers, having partitioned in accordance with the name restrictions having declared the names.
Let Φ be a linear form defined on $\mathbb{R}^{\Sigma}$, for each $V \subseteq \mathrm{Name}$, the domain $\mathcal{G}_V$ is a pair of function (f,g):

$$\begin{array}{rcl} f & : & V \cup \mathrm{Name} \rightarrow \{\ \text{Affine subspace of } \mathbb{R}^2\}, \\ g & : & (V \cup \mathrm{Name})^2 \rightarrow \{\ \text{Affine subspace of } \mathbb{R}^2\}, \end{array}$$

the concretization $\gamma_V(f,g)$ is given by:

$$\left\{ (\mathrm{id}, E) \;\middle|\; \begin{array}{l} E(x) = (y, \mathrm{id}_y) \implies (\Phi((|\mathrm{id}|_\lambda)_{\lambda\in\Sigma}), \Phi((|\mathrm{id}_y|_\lambda)_{\lambda\in\Sigma})) \in f(x,y) \\ \begin{cases} E(x) = (y, \mathrm{id}_y) \\ E(x') = (y', \mathrm{id}_y') \end{cases} \implies (\Phi((|\mathrm{id}_y|_\lambda)_{\lambda\in\Sigma}), \Phi((|\mathrm{id}_y')|_\lambda)_{\lambda\in\Sigma}) \in g((x,y),(x',y')) \end{array} \right\}$$

Reduction



Example

$(\nu\ \mathrm{make})(\nu\ \mathrm{edge})(\nu\ \mathrm{first})(\nu\ \mathrm{first})$

$\quad (*\mathrm{make}?^{1}[\mathrm{last}](\nu \mathrm{next})\ (\mathrm{edge}!^{2}[\mathrm{last},\mathrm{next}]\ \mid\ \mathrm{make}!^{3}[\mathrm{next}])$

$\quad \mid\ *\mathrm{make}?^{6}[\mathrm{last}](\mathrm{edge}!^{7}[\mathrm{last},\mathrm{first}])$

$\quad \mid\ \mathrm{make}!^{8}[\mathrm{first}])$

$\quad \mid\ \mathrm{edge}?^{9}[x,y][x =^{10} y][x \neq^{11} \mathrm{first}]\mathrm{Ok}!^{12}[]$

we first prove in global abstraction that:

$$f(2) \text{ satisfies } \begin{cases} c^{(1,3),\text{next}} = c^{(1,3),\text{last}} + c^{\text{next},\text{last}} \\ c^{\text{first},\text{last}} + c^{\text{next},\text{last}} = 1 \\ c^{\text{next},\text{next}} = 1 \end{cases}$$

$$f(7) \text{ satisfies } \begin{cases} c^{\text{next},\text{last}} + c^{\text{first},\text{last}} = 1 \\ c^{\text{last},\text{last}} = 1 \end{cases}$$

Example

We then prove in pair-wise analysis that in process 10, x and y are respectively linked to names created by some instance of the restrictions :

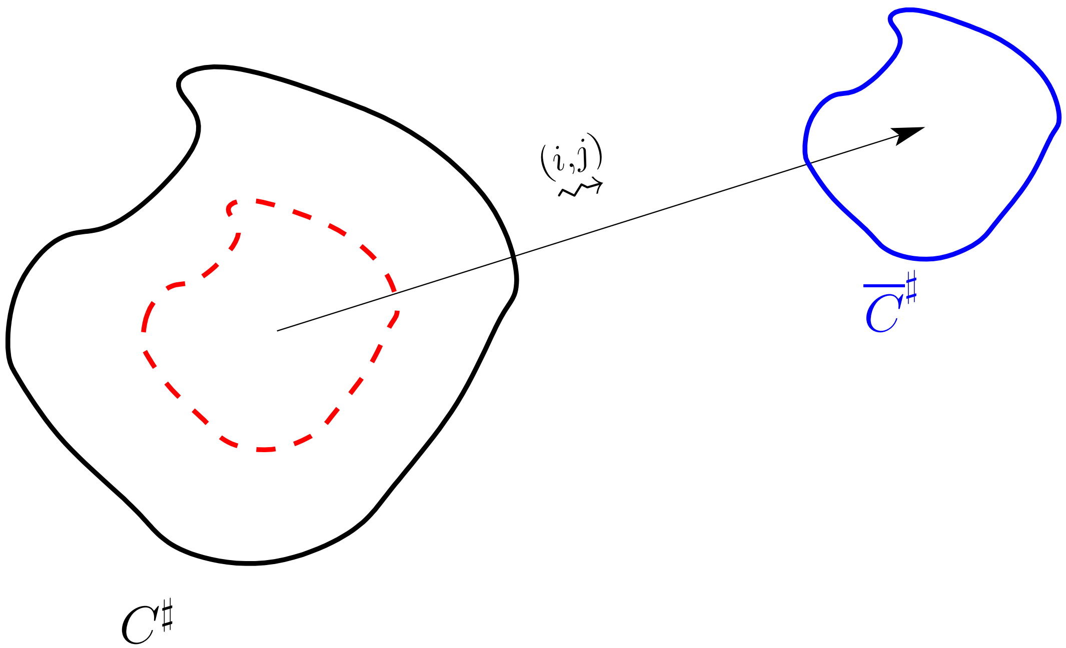
1. (ν first) and (ν first),
2. (ν first) and (ν next),
3. (ν next) and (ν next) but distinct instances,
4. (ν next) and (ν first).

so, the matching pattern $[x = y]$ is satisfiable only in the first case !!!

Occurrences counting analysis

$$\left\{\begin{array}{l}
\left(1, \varepsilon, \left\{\text{port} \quad \mapsto (\text{port}, \varepsilon)\right.\right) \\
\left(3, \varepsilon, \left\{\begin{array}{ll}\text{gen} & \mapsto (\text{gen}, \varepsilon) \\ \text{port} & \mapsto (\text{port}, \varepsilon)\end{array}\right.\right) \\
\left(2, \text{id}'_1, \left\{\begin{array}{ll}\text{add} & \mapsto (\text{email}, \text{id}_1) \\ \text{info} & \mapsto (\text{data}, \text{id}_1)\end{array}\right.\right) \\
\left(2, \text{id}'_2, \left\{\begin{array}{ll}\text{add} & \mapsto (\text{email}, \text{id}_2) \\ \text{info} & \mapsto (\text{data}, \text{id}_2)\end{array}\right.\right) \\
\left(5, \text{id}_2, \left\{\text{gen} \quad \mapsto (\text{gen}, \varepsilon)\right.\right)
\end{array}\right\}$$

Abstract transition



Abstract domains

We design a domain for representing numerical constrains between

- the number of occurrences of processes $\sharp(\mathrm{i})$;
- the number of performed transitions $\underline{\sharp}(\mathrm{i},\mathrm{j})$.

We use the product of

- a non-relational domain:
  $\Longrightarrow$ the interval lattice;
- a relational domain:
  $\Longrightarrow$ the lattice of affine relationships.

Interval narrowing

An exact reduction is exponential.
We use:

- Gaus reduction: $\begin{cases} x+y+z=1 \\ x+y+t=2 \end{cases} \Longrightarrow \begin{cases} x+y+z=1 \\ t-z=1 \end{cases}$
- Interval propagation: $\begin{cases} x+y+z=3 \\ x \in [|0;\infty|[\\ y \in [|0;\infty|[\\ z \in [|0;\infty|[\end{cases} \Longrightarrow \begin{cases} x+y+z=3 \\ x \in [|0;3|] \\ y \in [|0;\infty|[\\ z \in [|0;\infty|[\end{cases}$
- Redundancy introduction: $\begin{cases} x+y-z=3 \\ x \in [|1;2|[\end{cases} \Longrightarrow \begin{cases} x+y-z=3 \\ y-z \in [|1;2|] \\ x \in [|1;2|] \end{cases}$

to get a polynomial approximated reduction.

Example: non-exhaustion of resources

Example: exhaustion of resources

Example: mutual exclusion

Example: token ring

More example

$$
\begin{array}{l}
(\nu\ \mathrm{make})(\nu\ \mathrm{test})\\
\quad (*\mathrm{make}?^{1}[]((\nu\ a)(\nu\ b)\\
\qquad\qquad\qquad (a?^{2}[]b?^{3}[]\cancel{\mathrm{test}!^{4}[]}\\
\qquad\qquad\qquad \mid\ a?^{5}[]b!^{6}[]\\
\qquad\qquad\qquad \mid\ a!^{7}[])\\
\qquad\qquad\qquad \mid \mathrm{make}!^{8}[])\\
\quad \mid \mathrm{make}!^{9}[])
\end{array}
$$

Context-free analysis allows to prove that subprocess $\mathrm{test}!^{4}[]$ is unreachable.

Trace-based analysis

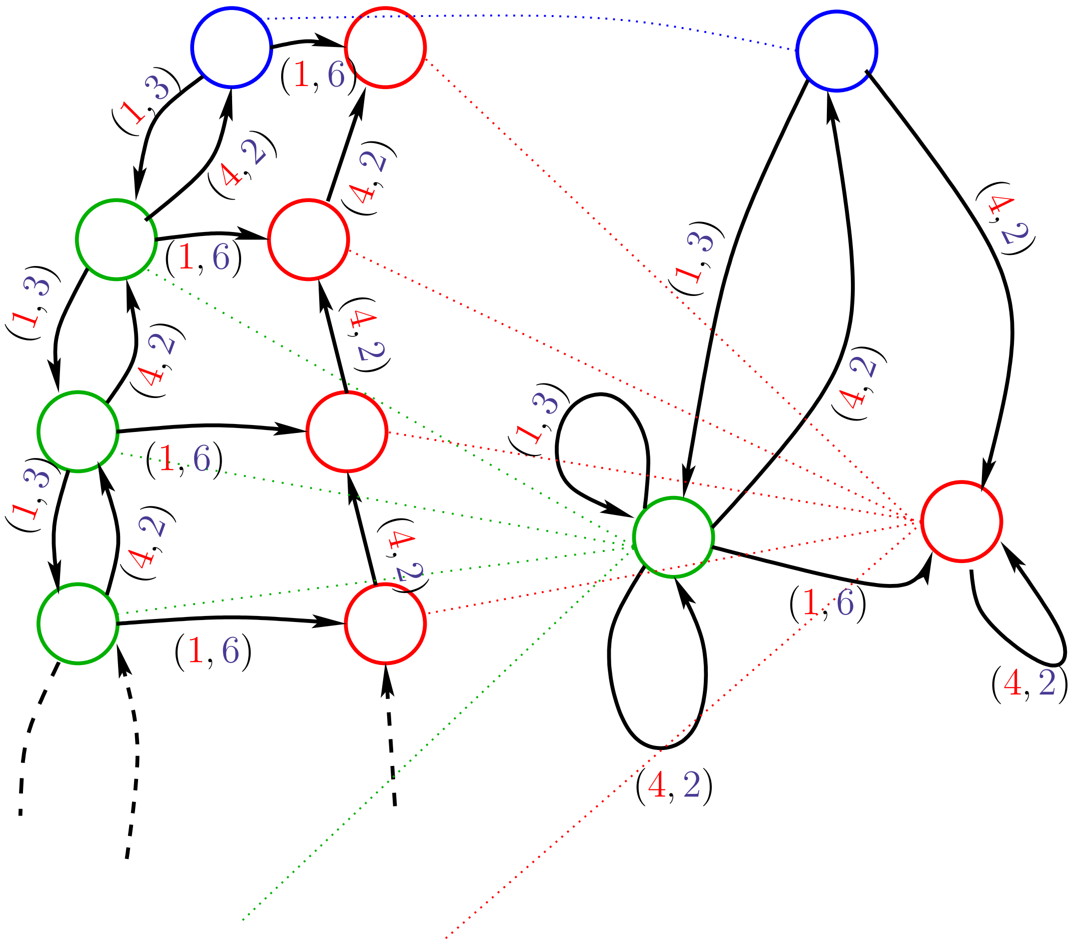
Main idea

We want to approximate the set of the configurations by which no infinite computation sequence can pass.

We propose to

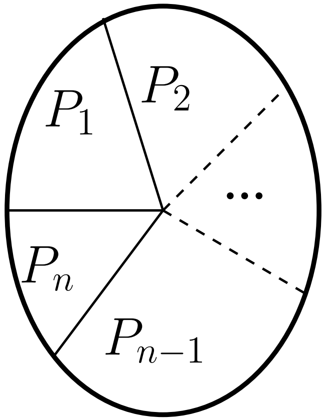
1. abstract the trace semantics of a mobile system;
2. for each configuration,
   - approximate the set of the transitions that may occur inside a computation sequence which stems from this configuration;
   - detect and prove whether this set defines a well-founded relation.

Quotient



Partitioning

We finitely partition $\mathcal{D}^\sharp$



$\mathcal{D}^\sharp$

$$\begin{cases} \forall k \in [|1;n|],\ P_k \neq \bot \\ \forall k,l \in [|1;n|],\ k \neq l \implies P_k \sqcap P_l = \bot \\ \bigsqcup (P_k)_{k \in [|1;n|]} = \top \end{cases}$$

by using our occurrences counting analysis.

Iteration

We iteratively construct both

- a transition system over (P_i),
- a representation function $f : [|1;n|] \to \mathcal{D}^\sharp$:

If $P_k \sqcap f(P_k) \overset{(i,j)}{\rightsquigarrow} \overline{C}^\sharp$ with $\overline{C}^\sharp \sqcap P_l \neq \bot$ then $\begin{cases} f(P_l) \leftarrow f(P_l) \nabla (\overline{C}^\sharp \sqcap P_l) \\ \text{the transition } P_k \overset{(i,j)}{\to} P_l \text{ is added.} \end{cases}$

Proof of termination

How to check that transition systems are well-founded ?

Abstracting environments away, transition rules look like chemical reactions.

$A|B \rightarrow A_1|A_2|...|B_1|B_2|...$ (communication)

$C|D \rightarrow C|C_1|C_2|...|D_1|D_2|...$ (resource fetching)

A_1, A_2,... is the continuation of A,
B_1, B_2,... is the continuation of B,
C_1, C_2,... is the continuation of C,
D_1, D_2,... is the continuation of D.

We decompose each transition into two half-transitions:

communication

$$A \rightarrow A_1|A_2|...$$
$$B \rightarrow B_1|B_2|...$$

resource fetching

$$C \rightarrow C$$
$$D \rightarrow C_1|C_2|...|D_1|D_2|...$$

Then we check if the following relation is well-founded:

communication

$$A > A_1,\ A > A_2,\ ...$$
$$B > B_1,\ B > B_2,\ ...$$

resource fetching

$$D > C_1,\ D > C_2,\ ...$$
$$D > D_1,\ D > D_2,\ ...$$

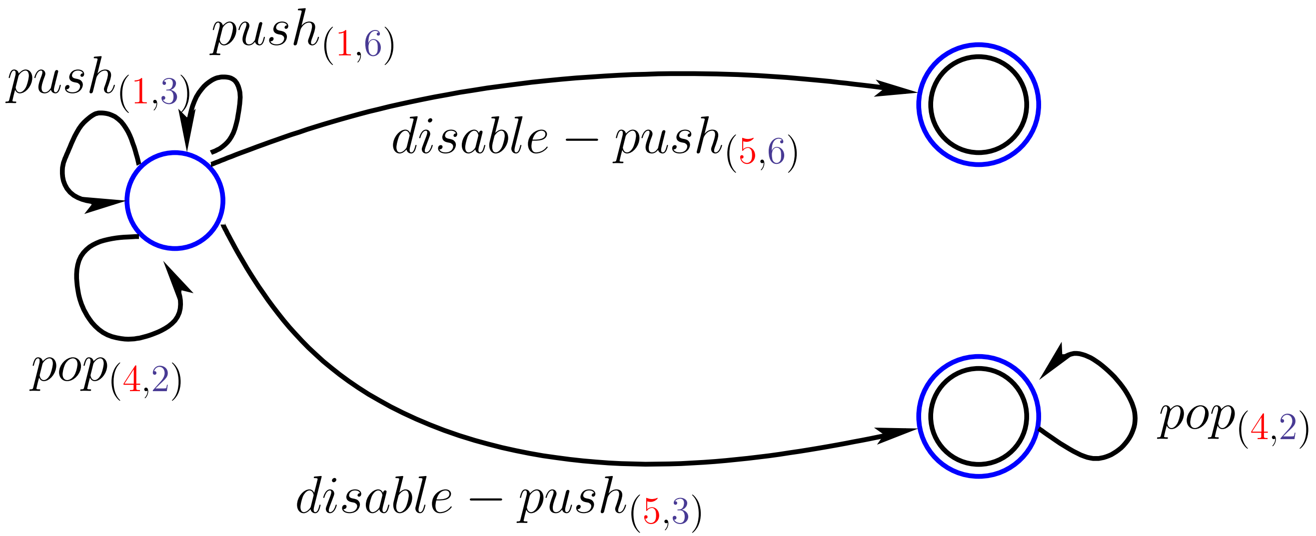
(i.e we check wether there are no cyclic production)

Example: a stack

$$
\begin{array}{rcl}
\mathcal{S} & := & (\nu\ \text{push})(\nu\ \text{pop}) \\
& & ((*\text{push}?^{1}[](\text{pop}!^{2}[] \mid \text{push}!^{3}[])) \\
& & \mid *\text{pop}?^{4}[] \\
& & \mid *\text{push}?^{5}[] \\
& & \mid \text{push}!^{6}[])
\end{array}
$$

$$
\begin{cases}
\pi(1) = 1,\ \pi(2) \in [|0; +\infty|[,\ \pi(3) \in [|0;1|], \\
\pi(4) = 1,\ \pi(5) = 1,\ \pi(6) \in [|0;1|], \\
\underline{\pi}(1,6) \in [|0;1|],\ \underline{\pi}(5,3) \in [|0;1|],\ \underline{\pi}(5,6) \in [|0;1|], \\
\underline{\pi}(1,3) \in [|0;\infty|[,\ \underline{\pi}(4,2) \in [|0;\infty|[.
\end{cases}
$$

Result



The analysis has proved that the computations of our system are bound to terminate as soon as a communication $(5,3)$ or a communication $(5,6)$ is performed.

Conclusion

- Our framework allows to infer a sound non-uniform description of mobile systems..
- It has succeeded in proving:
  - non-uniform confidentiality properties, in a polynomial time;
  - non-exhaustion, in a polynomial time;
  - mutual exclusion, in a polynomial time;
  - some dead locks, in an exponential time (in the case of the π-calculus).

Future Works

- Future Works:
  - Refine our initial partitioning,
  - Investigate other approximated algorithms,
  - Investigate relations with CFA and behvioral types,
  - Design a more generic analysis for analyzing liveness properties..

$\Longrightarrow$ To analyze big programs.