ESOP'02

Dependency Analysis of Mobile Systems

Jérôme Feret
École normale supérieure
http://www.di.ens.fr/~feret

April, 2002

Overview

1. Mobile systems:
   - intuitions,
   - the π-calculus;
2. Non-standard semantics:
   - marker allocation,
   - operational semantics;
3. Abstract semantics:
   - the abstract interpretation framework,
   - generic abstract transition system,
   - several analyses.

Mobile systems

Mobile system

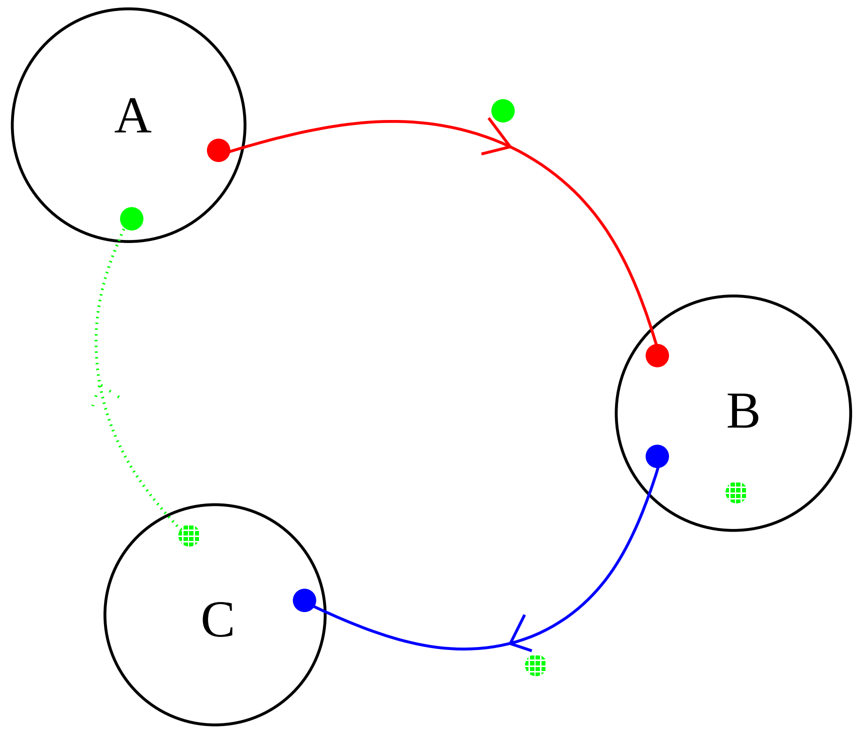
A pool of processes which interact and communicate:

Interactions

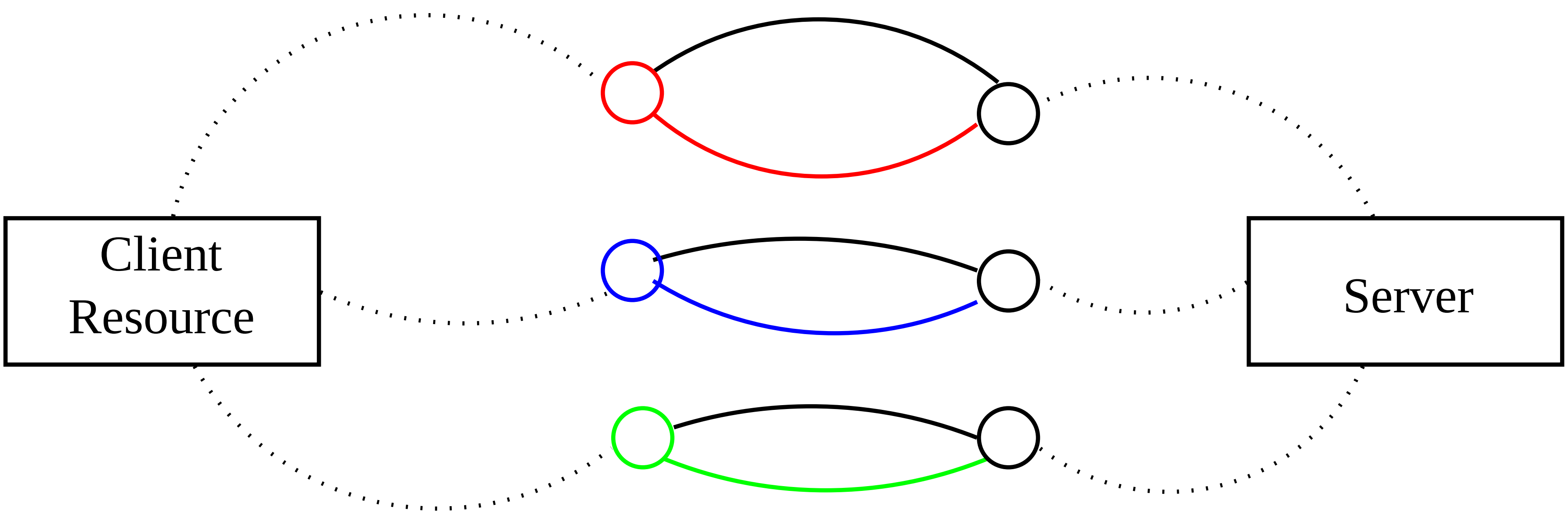
- synchronize process computation;
- change process structure (communication, migration);
- change communication links;
- create new processes.

Topology of interaction may be unbounded !

Dynamic linkage of agents



A network



Motivation

We want to compute a sound approximation of the relations between the names communicated to the variables of agents:
Are the variables x and y of an instance of the agent P

- always linked to the same name ?
- always linked to distinct names ?
- always linked to names created by the same recursive instance of an agent ?
- never linked to names created by the same recursive instance of an agent ?

(We will use a wider class of properties to infere them.)

π-calculus: syntax

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

$$P ::= \text{action}.P \mid (P \mid P) \mid (P + P) \mid \emptyset$$

$$\text{action} ::= c!^i[x_1, ..., x_n] \mid c?^i[x_1, ..., x_n] \mid *c?^i[x_1, ..., x_n] \mid (\nu\ x) \mid [x \diamond^i y]$$

where $n \geqslant 0$, c, x_1, ..., x_n, x, $\in$ *Name* ,$i \in$ *Label*, $\diamond \in \{=; \neq\}$;
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.

Transition semantics

A reduction relation and a congruence relation give the semantics of the π-calculus:

- the reduction relation specifies the result of process computations:

$$
\begin{array}{rcll}
c?^i[\overline{y}]Q \mid c!^j[\overline{x}]P & \xrightarrow{i,j} & Q[\overline{y} \leftarrow \overline{x}] \mid P & \\
*c?^i[\overline{y}]Q \mid c!^j[\overline{x}]P & \xrightarrow{i,j} & Q[\overline{y} \leftarrow \overline{x}] \mid *c?^i[\overline{y}]Q \mid P & \\
P+Q & \xrightarrow{\varepsilon} & P & \\
P+Q & \xrightarrow{\varepsilon} & Q & \\
[x \diamond^i y].P & \xrightarrow{\varepsilon} & P & \text{when } x \diamond y, \diamond \in \{=, \neq\}
\end{array}
$$

- the congruence relation reveals redexes:
  - solves conflicts between names (α-conversion, extrusion,...);
  - allows structural modifications (commutativity and associativity of $\mid$).

Example: syntax

$$\mathcal{S} := (\nu\ \text{port})(\nu\ \text{gen})$$
$$(\textbf{Server} \mid \textbf{Customer} \mid \text{gen}!^{0}[])$$

where

$$\textbf{Server} := *\text{port}?^{1}[\mathit{info}, \mathit{add}](\mathit{add}!^{2}[\mathit{info}])$$

$$\textbf{Customer} := *\text{gen}?^{3}[]\ ((\nu\ \mathit{data})\ (\nu\ \mathit{email})$$
$$(\text{port}!^{4}[\mathit{data}, \mathit{email}] \mid \text{gen}!^{5}[]))$$

Example: computation

$$(\nu\ \text{port})(\nu\ \text{gen})$$
$$(\textbf{Server} \mid \textbf{Customer} \mid \text{gen}!^{0}[])$$

$$\xrightarrow{3,0} (\nu\ \text{port})(\nu\ \text{gen})(\nu\ data_1)(\nu\ email_1)$$
$$(\textbf{Server} \mid \textbf{Customer} \mid \text{gen}!^{5}[] \mid \text{port}!^{4}[data_1, email_1])$$

$$\xrightarrow{1,4} (\nu\ \text{port})(\nu\ \text{gen})(\nu\ data_1)(\nu\ email_1)$$
$$(\textbf{Server} \mid \textbf{Customer} \mid \text{gen}!^{5}[] \mid email_1!^{2}[data_1])$$

$$\xrightarrow{3,5} (\nu\ \text{port})(\nu\ \text{gen})(\nu\ data_1)(\nu\ email_1)(\nu\ data_2)(\nu\ email_2)$$
$$(\textbf{Server} \mid \textbf{Customer} \mid \text{gen}!^{5}[] \mid email_1!^{2}[data_1] \mid \text{port}!^{4}[data_2, email_2])$$

$$\xrightarrow{1,4} (\nu\ \text{port})(\nu\ \text{gen})(\nu\ data_1)(\nu\ email_1)(\nu\ data_2)(\nu\ email_2)$$
$$(\textbf{Server} \mid \textbf{Customer} \mid \text{gen}!^{5}[] \mid email_1!^{2}[data_1] \mid email_2!^{2}[data_2])$$

Non-standard Semantics

Example: non-standard configuration

$$(\textbf{Server}^{1} \mid \textbf{Customer}^{3} \mid \mathrm{gen}!^{5}[] \mid \mathit{email}_1!^{2}[\mathit{data}_1] \mid \mathit{email}_2!^{2}[\mathit{data}_2])$$

$$\left\{\begin{array}{l}
\left(1, \varepsilon, \left\{\mathrm{port} \mapsto (\mathrm{port}, \varepsilon)\right.\right) \\
\left(3, \varepsilon, \left\{\begin{array}{ll}\mathrm{gen} & \mapsto (\mathrm{gen}, \varepsilon) \\ \mathrm{port} & \mapsto (\mathrm{port}, \varepsilon)\end{array}\right.\right) \\
\left(2, id'_1, \left\{\begin{array}{ll}\mathit{add} & \mapsto (\mathit{email}, id_1) \\ \mathit{info} & \mapsto (\mathit{data}, id_1)\end{array}\right.\right) \\
\left(2, id'_2, \left\{\begin{array}{ll}\mathit{add} & \mapsto (\mathit{email}, id_2) \\ \mathit{info} & \mapsto (\mathit{data}, id_2)\end{array}\right.\right) \\
\left(5, id_2, \left\{\mathrm{gen} \mapsto (\mathrm{gen}, \varepsilon)\right.\right)
\end{array}\right\}$$

Marker properties

1. Marker allocation must be consistent:

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

2. Marker allocation should be robust:

   Marker allocation should not depend on the interleaving.

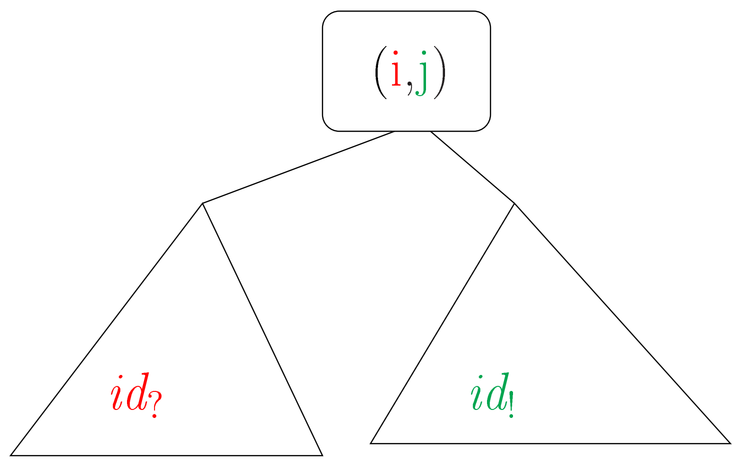
Marker allocation

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 $(i, j) \in \mathit{Label}^2$.

They are recursively calculated when fetching resources as follows:

id_* :



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.

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

Transition system

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

$$\frac{\diamond \in \{=;\neq\},\ E(y) \diamond E(x),\ Cont \in \beta(P, id, E)}{C \cup \{([x \diamond^i y].P, id, E)\} \xrightarrow{\varepsilon} (C \cup Cont)}$$

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

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

Abstraction

Collecting semantics

$(\mathcal{C}, C_0, \rightarrow)$ 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.

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

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

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

The calculus of this fix point is not usually decidable.

Abstract domain

We introduce an abstract domain of properties:

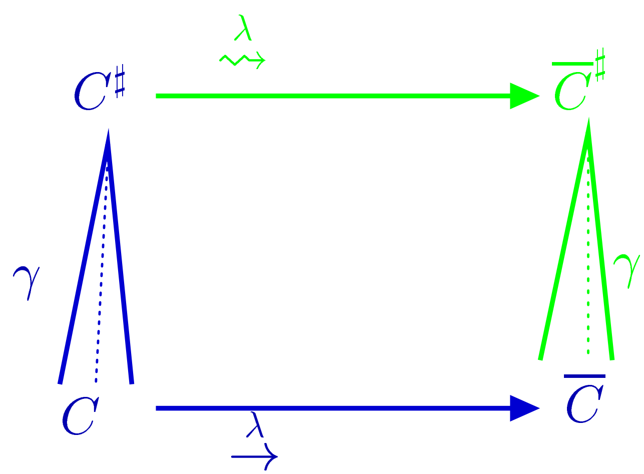
- properties of interest;
- more complex properties used in calculating them.

The 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 satisfy the property A.

Abstract transition system

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



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

Generic environment analysis

For each finite 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:

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

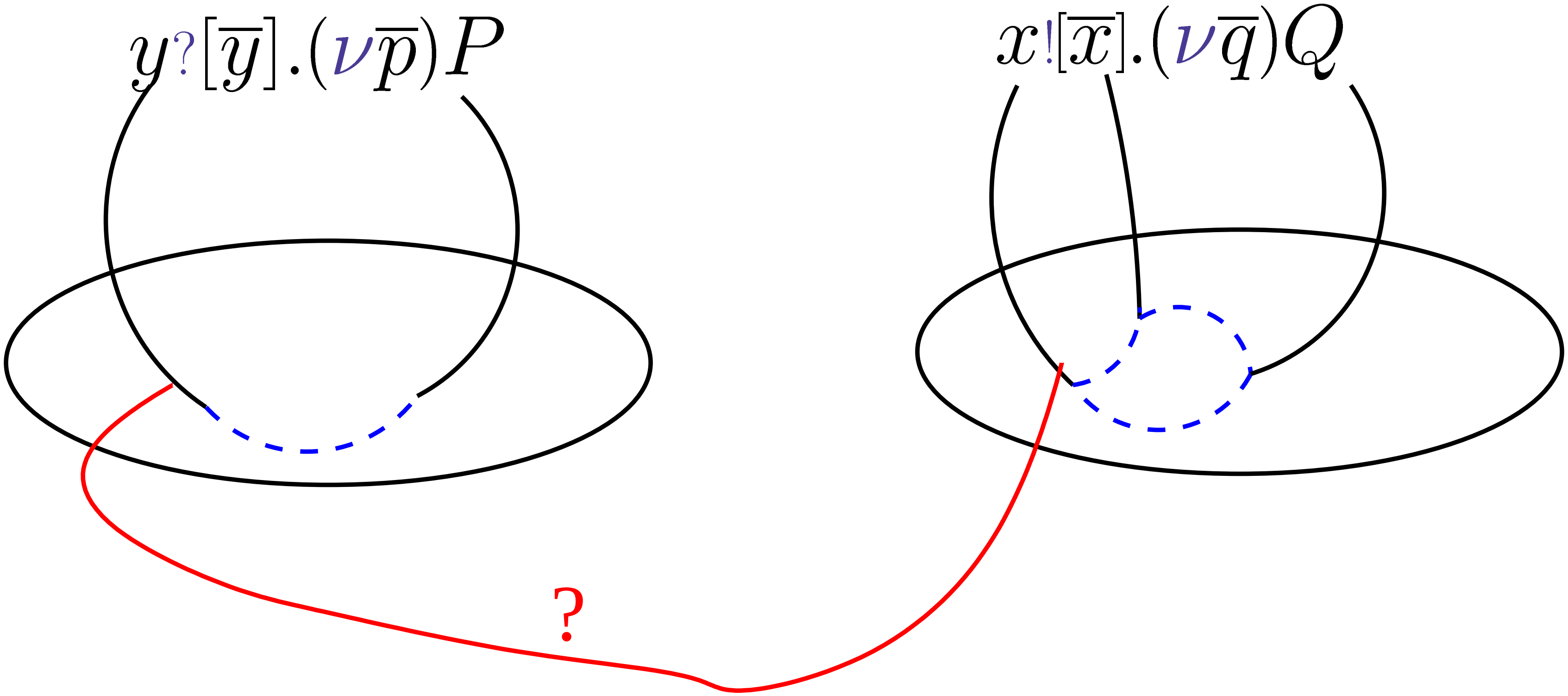
The abstract domain $C^\sharp$ is then the set:

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

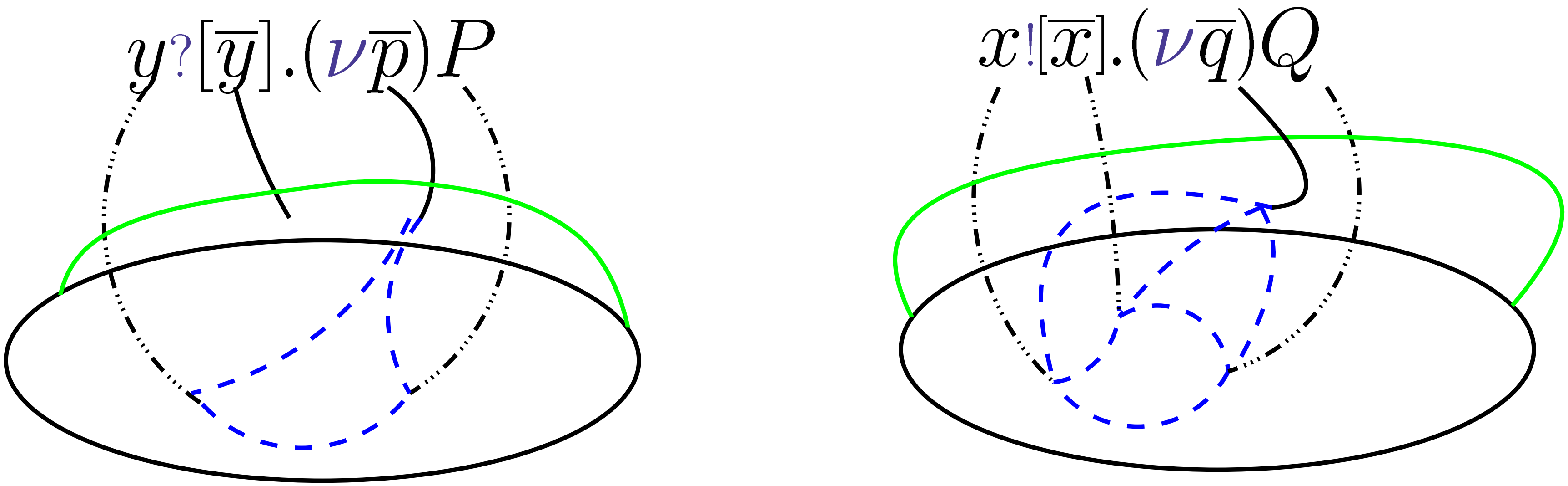
related to $\wp(C)$ by the concretization γ:

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

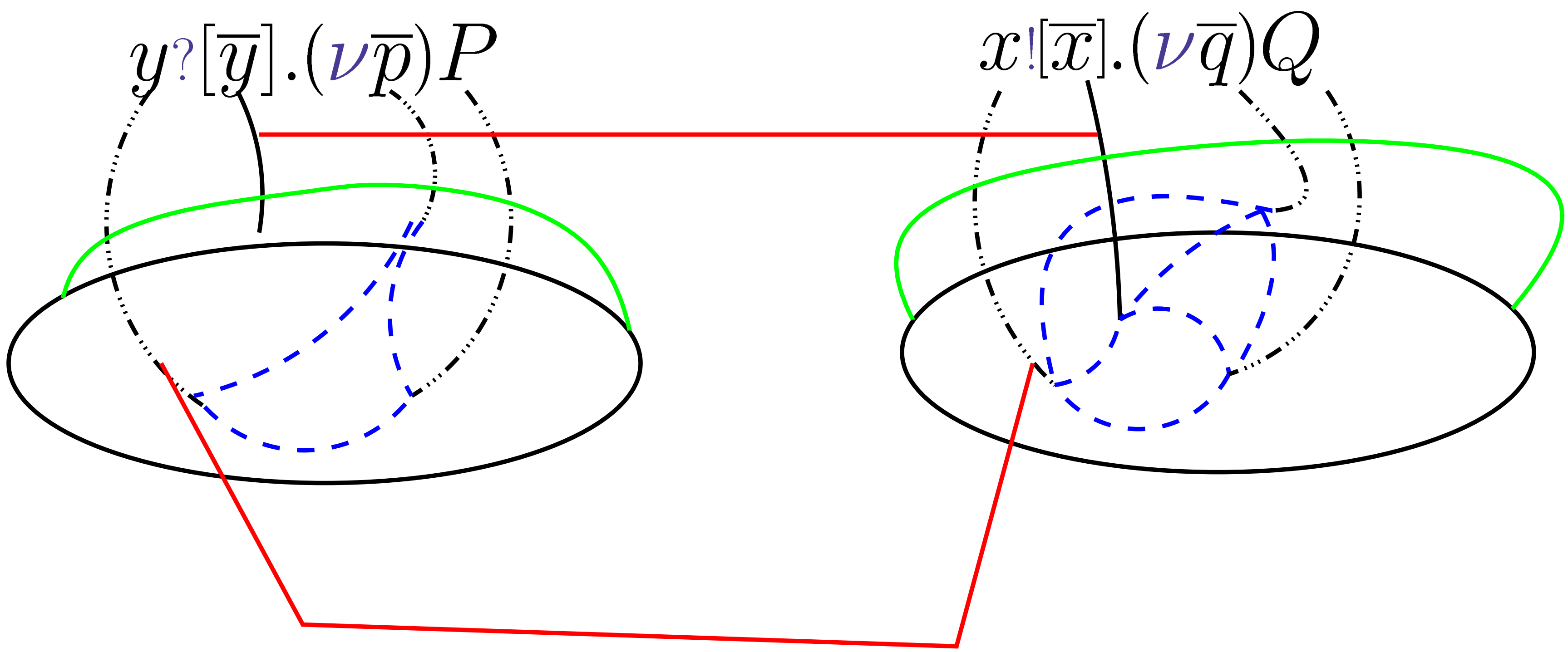
Abstract communication



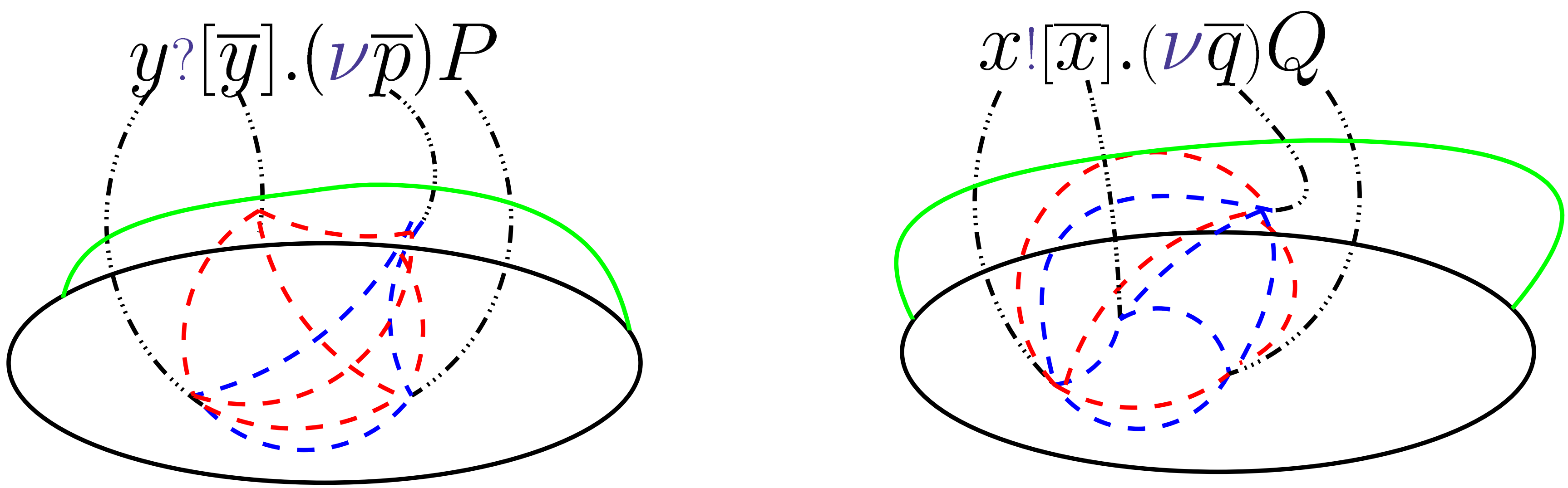
Extending environments



Synchronizing environments



Propagating information



A centered-relationnal abstraction

We capture the relations between the marker of an agent and the markers of the names linked to its variables:

- we abstract the shape of the markers by using a regular domain for describing sets of markers:

$$\mathcal{G}_V = (\{Ag\} + (V \times Name)) \to Reg_\Sigma,$$
$$\gamma_V(f) = \left\{ (id, E) \;\middle|\; \begin{cases} id \in \gamma_{Reg_\Sigma}(f(Ag)) \\ E(y) = (x, id_x) \implies id_x \in \gamma_{Reg_\Sigma}(f(y,x)) \end{cases} \right\},$$

- we abstract relations between markers by using a numerical domain:

$$\mathcal{G}_V = (V \times Name) \to N_\Sigma$$
$$\gamma(f) = \{(id, E) \mid E(y) = (x, id_x) \implies (id, id_y) \in \gamma_{N_\Sigma}\}.$$

Non-uniform result

Netscape: Pi-s.a. 3: Pi static analyser

```
(# port)(# gen)
 ( *gen?^1[](#email)(#data)(port!^2[data,email] | gen!^3[])
 | *port?^4[info,add]add!^5[info]
 | gen!^6[]
 )
```

Start --> (1,6)A
A --> (1,3)A + (4,2)B
B --> END

Start --> (1,6)A
A --> END + (1,3)A

(1,3) = (1,3)
(4,2) = (4,2) + 1
(1,6) = (1,6)

---

main menu – control flow analysis – (#email)

Pi–s.a. Version 3.16, last Modified Tue November 27 2001
Pi–s.a. is an experimental prototype for an academic use only.

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 name and agent markers:

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

we cannot infer that 11 is unreachable.

Abstracting equivalence relation

We introduce a compact abstract domain to describe sets of equivalence relations over a set of variables $\mathcal{V}$:

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

For any set I, $\mathcal{T}_{\mathcal{V}}$ is related to the powerset $\wp(\mathcal{V} \rightarrow I)$ via the following concretization function:

$$\gamma_{\mathcal{V}}^{I}((A, R)) = \left\{ f \;\middle|\; \begin{array}{l} \forall \mathcal{X} \in A,\ \{x, y\} \subseteq \mathcal{X} \implies f(x) = f(y) \\ (\mathcal{X}, \mathcal{Y}) \in R \implies \forall x \in \mathcal{X}, y \in \mathcal{Y},\ f(x) \neq f(y) \end{array} \right\}$$

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

Dependency analysis

We abstract the relation between the names linked to the variables of an agent:

$$\mathcal{G}_V = \mathcal{T}_V,$$
$$\gamma_V = \{(id, E) \mid E \in \gamma_V^{Name \times Id}\};$$

and the relation between the markers of an agent and its names:

$$\mathcal{G}_V = \mathcal{T}_{\{Ag\} \cup V},$$
$$\gamma_V = \left\{(id, E) \;\middle|\; [Ag \mapsto id; y \mapsto snd(E(x))] \in \gamma_{\{Ag\} \cup V}^{Id}\right\}.$$

Global numerical analysis

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

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

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

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

Pair-wise numerical analysis

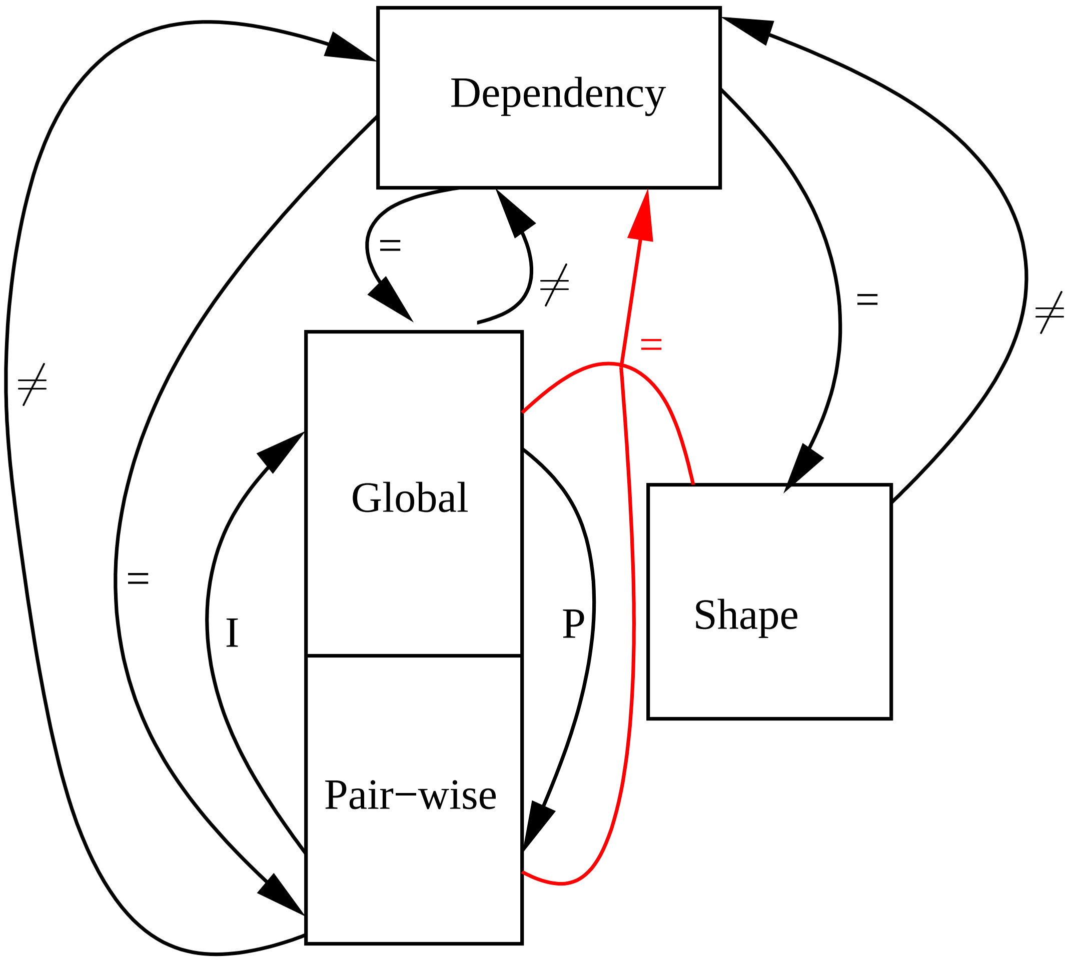
We compare pair-wisely markers, having partitioned in accordance with the name creations having created the names.
Let Φ be a linear form defined on $\mathbb{R}^{\Sigma}$, for each $V \subseteq Name$, the domain $\mathcal{G}_V$ is a pair of function (f, g):

$$\begin{array}{rcl} f & : & V \times Name \rightarrow \{\ \textit{affine subspace of}\ \mathbb{R}^2\}, \\ g & : & (V \times Name)^2 \rightarrow \{\ \textit{affine subspace of}\ \mathbb{R}^2\}, \end{array}$$

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

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

Reduction



Example

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

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

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

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

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

we first discover in global abstraction domain that:

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

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

Example (continued)

We then prove in the pair-wise relation domain that in process 9, x and y are respectively linked to names created by some instance of the restrictions :

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

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

Conclusion

We have proposed a generic framework for analyzing the invariants of mobile systems:

- it can be applied to many formalisms such as the π-calculus, mobile ambients, the join-calculus, (etc),
- it can be used in designing several analyzes, such as $0-\text{CFA}$ and non-uniform CFA.

Future Works

Design a behavioral analysis which dinstinguishes between several instances of the same agent.