Groupe de travail

Analysis of Mobile Systems by Abstract Interpretation

Jérôme Feret

École Normale Supérieure

http://www.di.ens.fr/~feret

31/03/2005

Introduction I

We propose a unifying framework to design

- automatic,
- sound,
- approximate,
- decidable,

semantics to abstract the properties of mobile systems.

Our framework is model-independent:
$\Longrightarrow$ we use a META-language to encode mobility models,
$\Longrightarrow$ we design analyses at the META-language level.

We use the Abstract Interpretation theory.

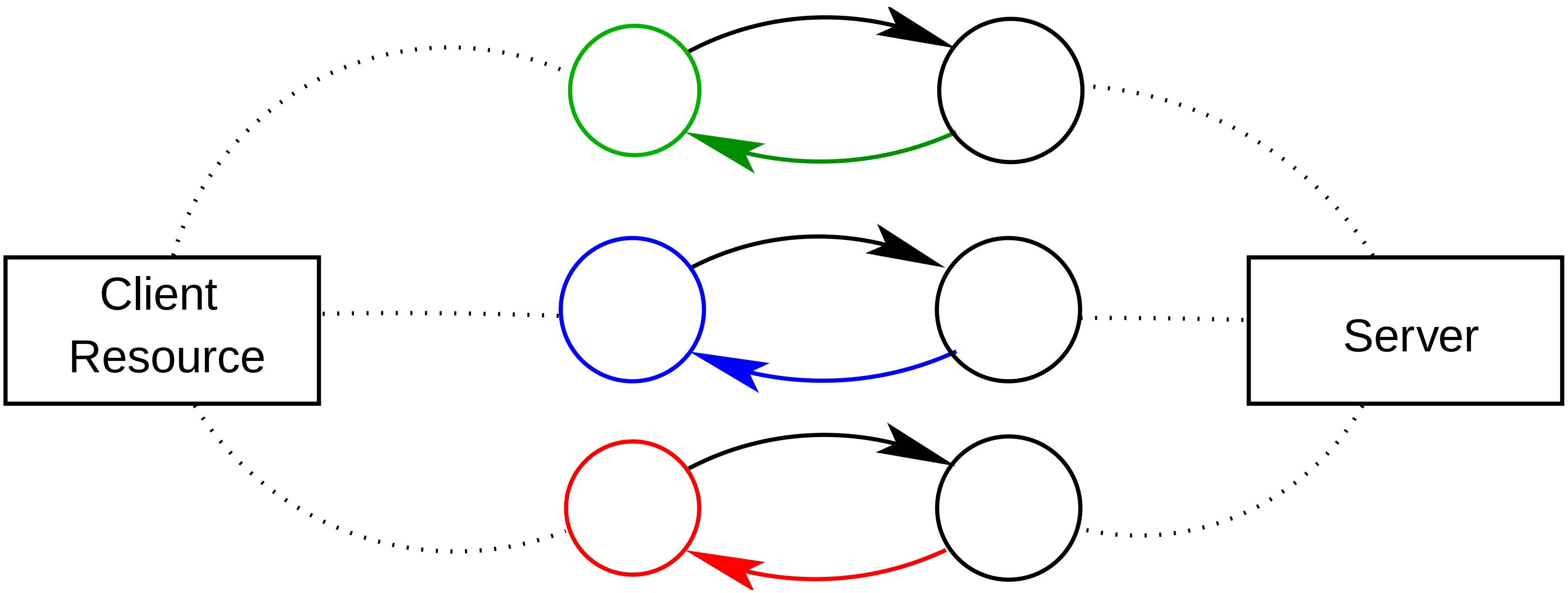
Introduction II

We focus on reachability properties.
We distinguish between recursive instances of components.

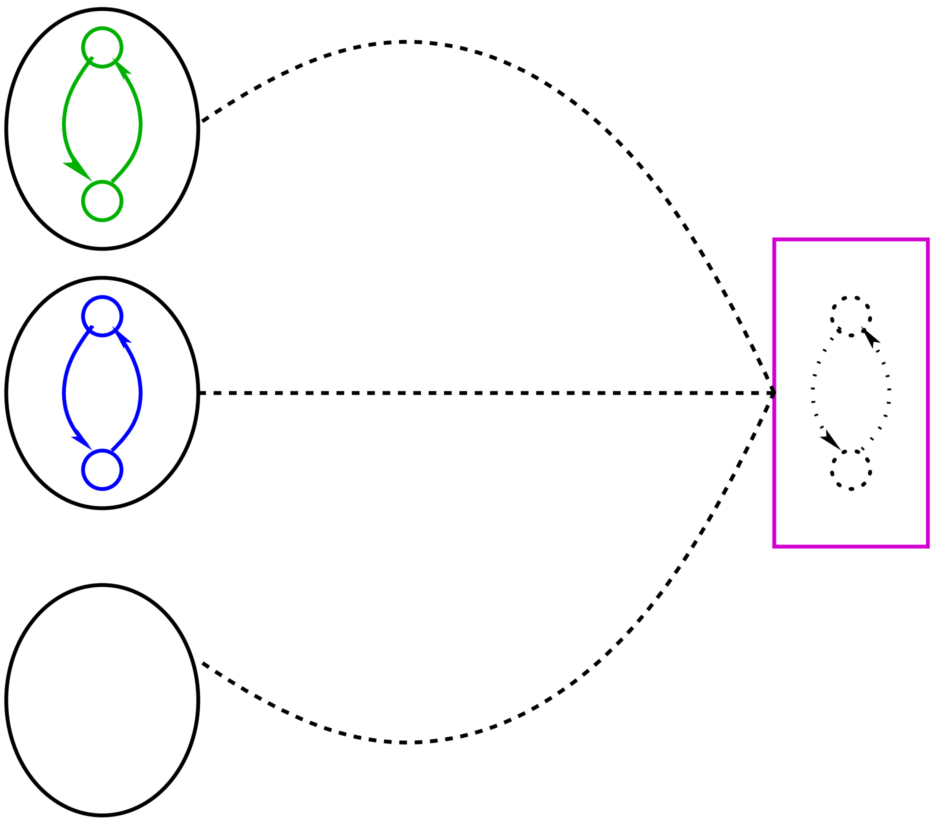
We design three families of analyses:

1. environment analyses capture dynamic topology properties
   (non-uniform control flow analysis, secrecy, confinement, . . .)
2. occurrence counting captures concurrency properties
   (mutual exclusion, non exhaustion of resources)
3. thread partitioning mixes both dynamic topology and concurrency properties
   (absence of race conditions, authentication, . . .).

A network



Example: a 3-port server



Example: a shared-memory

We consider the implementation of a shared-memory where:

- agents may allocate new cells,
- authorized agents may read the content of a cell,
- authorized agents may write inside a cell, overwriting the former content.

The content of a cell is encoded by an output on a channel. We want to prove that the value of each cell never becomes ambiguous (i.e. there never are two outputs over the same channel).

π-calculus: syntax

Name: infinite set of channel names,
Label : infinite set of labels,

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

$$\text{action} ::= c!^{i}[x_1, ..., x_n] \mid c?^{i}[x_1, ..., x_n] \mid *c?^{i}[x_1, ..., x_n]$$

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

ν and $?$ are the only name binders.
$fv(P)$: free variables in P,
$bn(P)$: bound names in P.

Non-standard semantics

A refined semantics where:

- each recursive instance of processes is identified with an unambiguous marker;
- each name of channel is stamped with the marker of the process which has opened this channel.

Example: non-standard configuration

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

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

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.

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

Transition system

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

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

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

where $\mathcal{N}$ is the tree constructor.

Overview

1. Abstract Interpretation
2. Environment analysis
3. Occurrence counting analysis
4. Thread partitioning
5. Conclusion

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 that 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 fixpoint of the following $\cup$-complete endomorphism $\mathbb{F}$:

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

This fixpoint is usually not computable automatically.

Abstract domain

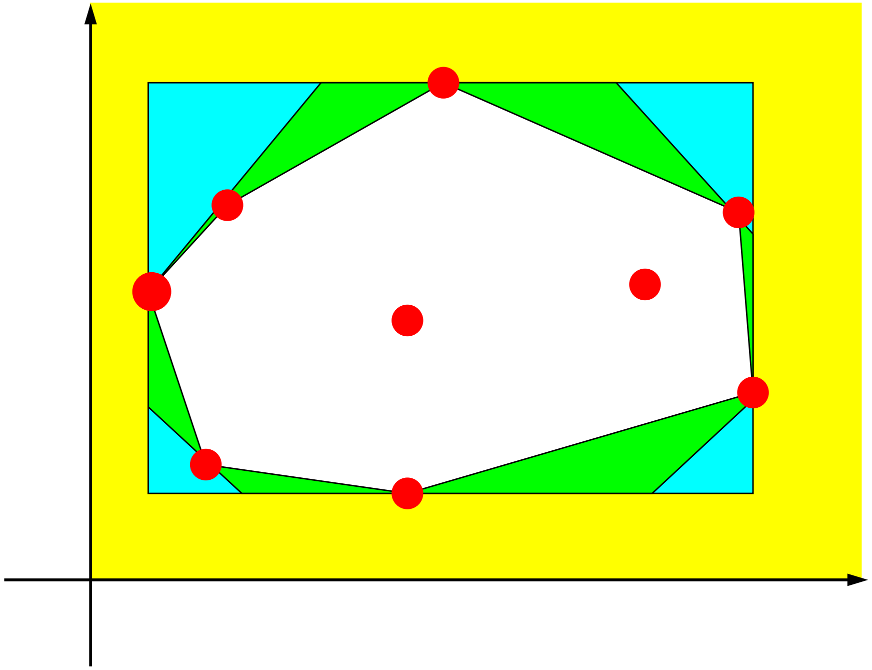
We introduce an abstract domain of properties:

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

This domain is often a 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 configurations which satisfy 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 transition relation, which satisfies $C_0 \subseteq \gamma(C_0^\sharp)$ and the following diagram:

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

Then, $\mathcal{S} \subseteq \bigcup_{n\in\mathbb{N}} \gamma(\mathbb{F}^{\sharp^n}(C_0^\sharp))$,

where $\mathbb{F}^\sharp(C^\sharp) = C_0^\sharp \sqcup \left(\bigsqcup\{\overline{C^\sharp} \mid C^\sharp \rightsquigarrow \overline{C^\sharp}\}\right)$.

Widening operator

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

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

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 *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 $\mathcal{I}$:

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

Approximate reduced product

The Abstract Interpretation framework provides tools for making the product of several abstractions.

Abstract properties may refine each other to get better results.

An abstract computation step is enabled if and only if it is enabled in all abstractions.

Approximate reduced product

Given two abstractions $(\mathcal{D}^\sharp, \gamma, C_0^\sharp, \rightsquigarrow)$ and $(\mathcal{D}^\sharp, \gamma, C_0^\sharp, \rightsquigarrow)$, and a reduction $\rho\ :\ \mathcal{D}^\sharp \times \mathcal{D}^\sharp \longrightarrow \mathcal{D}^\sharp \times \mathcal{D}^\sharp$ 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 $(\mathcal{D}^\sharp, \gamma, C_0^\sharp, \rightsquigarrow)$ 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.

Overview

1. Abstract Interpretation
2. Environment analysis
3. Occurrence counting analysis
4. Thread partitioning
5. Conclusion

Generic environment analysis

$\Longrightarrow$ Abstract the relations among the marker and the names of threads at each program point.

For any finite subset $V \subseteq \mathcal{V}$,

$$\wp(\mathit{Id} \times (V \to (\mathit{Label} \times \mathit{Id}))) \xleftarrow{\gamma_V} \mathit{Atom}^\sharp_V.$$

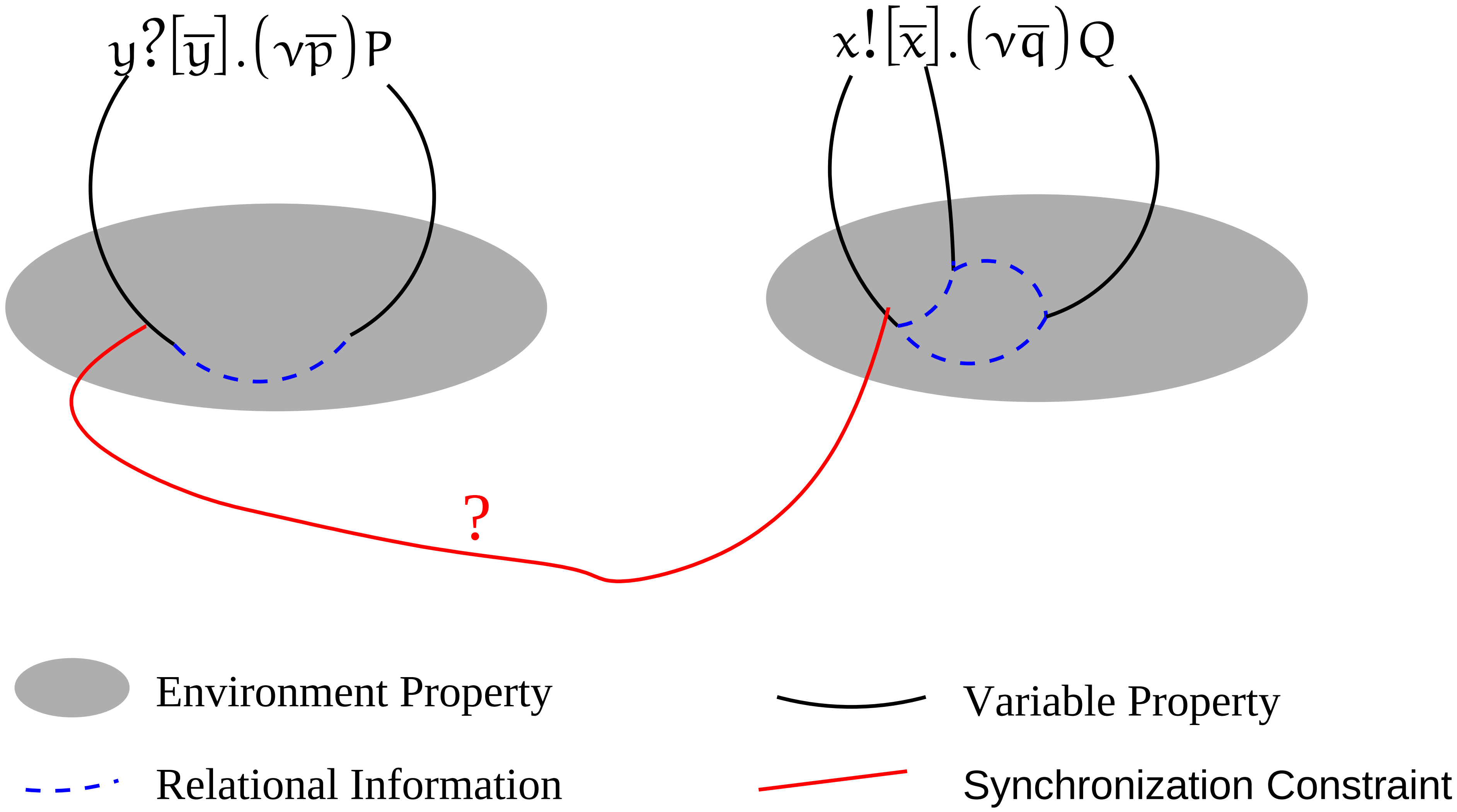
The abstract domain $C^\sharp$ is then the set:

$$C^\sharp = \prod_{p \in \mathcal{P}} \mathit{Atom}^\sharp_{I(p)}$$

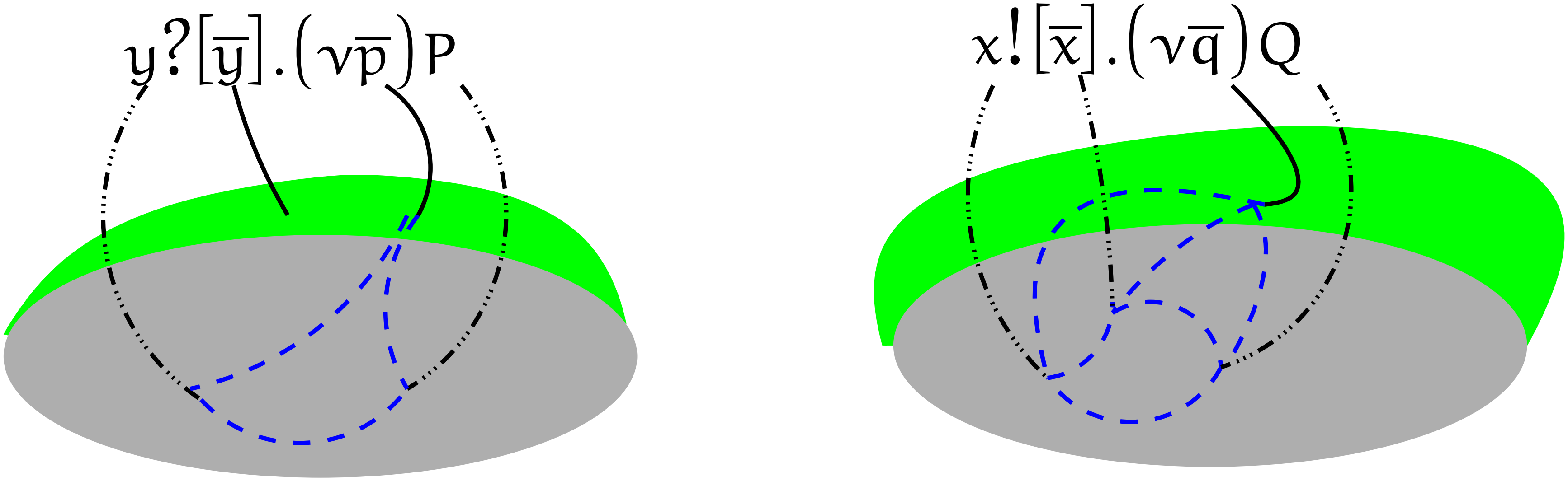
related to $\wp(C)$ by the concretization γ:

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

Abstract communication



Extending environments



Environment Property

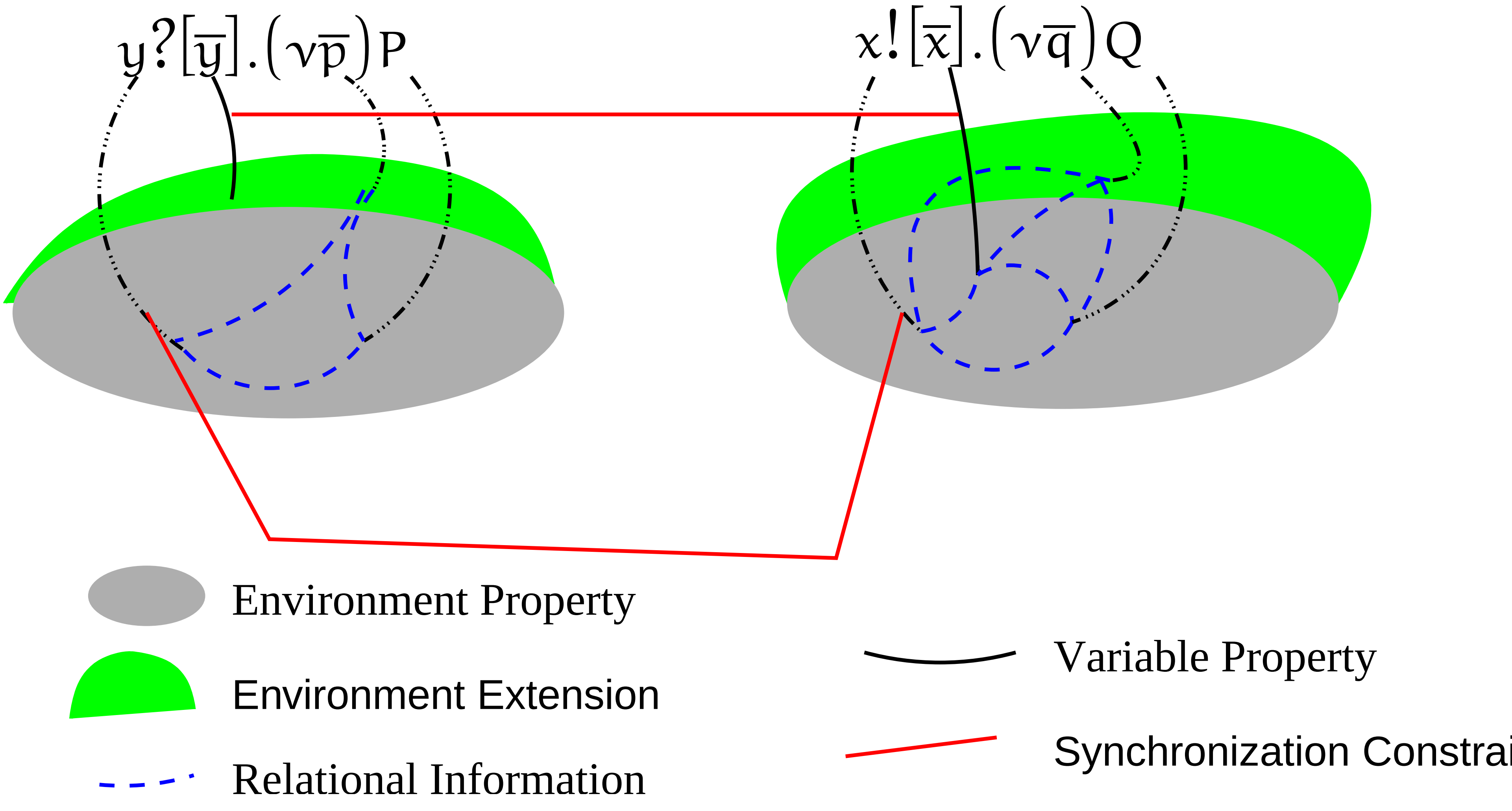
Environment Extension

Relational Information

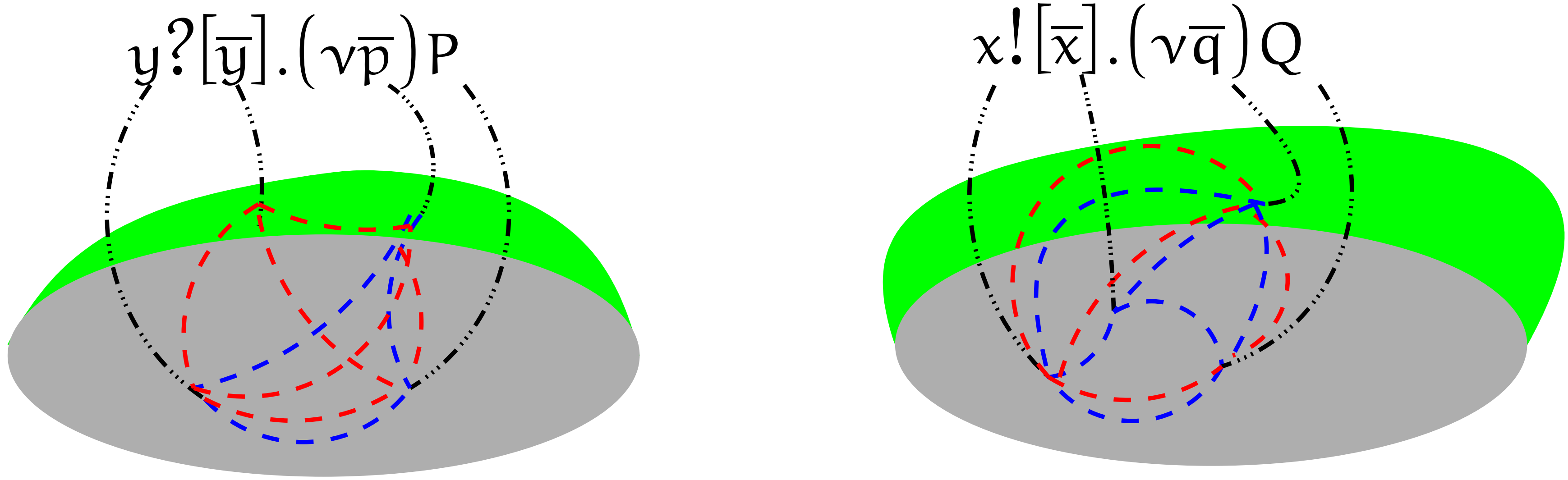
Variable Property

Synchronization Constraint

Synchronizing environments



Propagating information



Environment Property

Environment Extension

Relational Information

Variable Property

Information closure

Generic primitives

We only require abstract primitives to:

1. extend the domain of the environments,
2. gather the description of the linkage of the syntactic agents,
3. synchronize variables,
4. compute information closure,
5. separate the descriptions,
6. restrict the domain of the environments.

Thread abstraction: Structure

For any finite $V \subseteq \mathcal{V}$,

1. $(\mathit{Atom}^\sharp_V, \sqsubseteq_V)$ is a pre-order;
2. $\mathit{Env}_V \stackrel{\Delta}{=} \mathit{Id} \times (V \to (\mathit{Label} \times \mathit{Id}))$;
3. $\gamma_V : \mathit{Atom}^\sharp_V \to \wp(\mathit{Env}_V)$, $a \sqsubseteq_V b \implies \gamma_V(a) \subseteq \gamma_V(b)$;
4. $\bigsqcup_V : \wp(\mathit{Atom}^\sharp_V) \to \mathit{Atom}^\sharp_V$, $\forall a \in A,\ a \sqsubseteq_V \bigsqcup_V(A)$;
5. $\bot_V \in \mathit{Atom}^\sharp_V$, $\gamma(\bot_V) = \emptyset$;
6. ∇_V is a widening operator;

Thread abstraction: Abstract primitives

1. initial environment abstraction:
$\varepsilon^\sharp \in \mathit{Atom}^\sharp_\emptyset$,

$$\{(\varepsilon,\emptyset)\} \subseteq \gamma_\emptyset(\varepsilon^\sharp);$$

2. abstract name restriction:
$\forall x \in \mathcal{V} \setminus V, l \in \mathit{Label}, A \in \mathit{Atom}^\sharp_V, \nu^\sharp(x,l,A) \in \mathit{Atom}^\sharp_{V\cup\{x\}}$,

$$\left\{ (id, E) \in \mathit{Env}_{V\cup\{x\}} \;\middle|\; \begin{array}{l} (id, E_{|V}) \in \gamma_V(A), \\ E(x) = (l, id), \\ \forall y \in V,\ E(y) \neq (x, id) \end{array} \right\} \subseteq \gamma_{V\cup\{x\}}(\nu^\sharp(x,l,A));$$

3. abstract garbage collection:
$\forall X \subseteq \mathcal{V}, A \in \mathit{Atom}^\sharp_V, \mathrm{GC}^\sharp(X,A) \in \mathit{Atom}^\sharp_X$,

$$\{(id, E_{|X}) \in \mathit{Env}_X \mid (id, E) \in \gamma_V(A)\} \subseteq \gamma_X(\mathrm{GC}^\sharp(X,A)).$$

Thread tuple abstraction: Domain

For any tuple $(V_i)_{1\leq i\leq n}$ of finite sets of variables,

$$\textit{Molecule}^{\sharp}_{(V_i)_{1\leq i\leq n}}$$

is an abstract domain.

No structure is required (no extrapolation);

$$\gamma_{(V_i)_{1\leq i\leq n}} : \textit{Molecule}^{\sharp}_{(V_i)_{1\leq i\leq n}} \to \wp(\Pi_{1\leq i\leq n}(\textit{Env}_{V_i})).$$

Thread tuple abstraction: Conversion

1. abstract injection: $\forall V \subseteq \mathcal{V}, A \in \textit{Atom}^{\sharp}_{V}$, $\textsc{inj}^{\sharp}(A) \in \textit{Molecule}^{\sharp}_{(V)}$

$$\gamma_V(A) \subseteq \gamma_{(V)}(\textsc{inj}^{\sharp}(A)).$$

2. abstract product: $\forall A \in \textit{Molecule}^{\sharp}_{(U_i)_{1\leq i\leq m}}, B \in \textit{Molecule}^{\sharp}_{(V_i)_{1\leq i\leq n}}$, $A \bullet B \in \textit{Molecule}^{\sharp}_{(U_i).(V_i)}$

$$\left\{ (e_i)_{i\in[\![1;m+n]\!]} \;\middle|\; \begin{array}{l} (e_i)_{1\leq i\leq m} \in \gamma_{(U_i)}(A) \\ (e_{i+m})_{1\leq i\leq n} \in \gamma_{(V_i)}(B) \end{array} \right\} \subseteq \gamma_{(U_i).(V_i)}(A \bullet B).$$

3. abstract projections: $\forall A \in \textit{Molecule}^{\sharp}_{(V_i)_{1\leq i\leq n}}, k \in [\![1;n]\!]$, $\textsc{proj}^{\sharp}(k, A) \in \textit{Atom}$

$$\left\{ (id_k, E_k) \;\middle|\; \exists\, (id_i, E_i)_i \in \gamma_{(V_i)_i}(A) \right\} \subseteq \gamma_{V_k}(\textsc{proj}^{\sharp}(k, A)).$$

Thread tuple abstraction:
Abstract extension

Given:

- $(V_i)_{1 \le i \le n} \in (\wp(\mathcal{V}))^n$;
- $X \subseteq \mathcal{V} \times [\![1;n]\!]$;
- $A \in \textit{Molecule}^{\sharp}_{(V_i)}$;

we define $\begin{cases} U_i = V_i \setminus \{x \mid (x,i) \in X\} & \text{(unchanged variables)} \\ W_i = V_i \cup \{x \mid (x,i) \in X\} & \text{(extended interface)}; \end{cases}$

then:

$$\text{NEW}^{\sharp}_{\top}(X, A) \in \textit{Molecule}^{\sharp}_{(W_i)}$$

$$\left\{ (\textit{id}_i, E_i) \in \Pi(\textit{Env}_{W_i}) \;\middle|\; \begin{array}{l} \exists (\textit{id}_i, E'_i) \in \gamma_{(V_i)}(A), \\ \forall i \in [\![1;n]\!], E'_{i|U_i} = E_{i|U_i} \end{array} \right\} \subseteq \gamma_{(W_i)}(\text{NEW}^{\sharp}_{\top}(X, A)).$$

Thread tuple abstraction: Abstract synchronization

Given:

- $A \in \textit{Molecule}^{\sharp}_{(V_i)_{1\leq i\leq n}}$,
- $(p_i) \in \mathcal{L}_p^n$,
- $S \subseteq \{(x,k) \diamond (y,l) \mid \diamond \in \{=;\neq\}, x \in V_k \cup \{I\}, y \in V_l \cup \{I\}\}$;

then:

$$\text{SYNC}^{\sharp}(S,(p_i),A) \in \textit{Molecule}^{\sharp}_{(V_i)},$$

$$\{(\textit{id}_i, E_i) \in \gamma_{(V_i)}(A) \mid \forall (a \diamond b) \in S,\ \rho(a) \diamond \rho(b)\} \subseteq \gamma_{(V_i)}(\text{SYNC}^{\sharp}(S,(p_i),A)),$$

where $\rho((x,i)) = E_i(x)$ when $x \in V_i$ and $\rho((I,i)) = (p_i, \textit{id})$.

thread tuple abstraction: Abstract marker allocation

Given:

- $(p_i) \in \mathcal{L}_p^n$;
- $A \in \textit{Molecule}^\sharp_{(V_i)}$,

Then: $\text{FETCH}^\sharp((p_i), A) \in \textit{Molecule}^\sharp_{(V_i)}$

$$\left\{ (\overline{id}_i, E_i) \;\middle|\; \begin{array}{l} \forall i \in [\![1;n]\!],\ (id_i, E_i) \in \gamma_{(V_i)}(A) \\ id_i \neq \overline{id}_1 \\ \forall x \in V_i, y \in \textit{Label},\ (y, \overline{id}_1) \neq E_i(x) \end{array} \right\} \subseteq \gamma_{(V_i)}(\text{FETCH}^\sharp((p_i), A))$$

where $\overline{id}_i = \begin{cases} \mathcal{N}((p_i)_{1 \leq i \leq n}, id_1, ..., id_n) & \text{if } i = 1 \\ id_i & \text{otherwise.} \end{cases}$

Abstract operational semantics: Reactive molecule

Given:

- $\mathcal{R} = (n, \mathit{components}, \mathit{compatibility}, \mathit{v\text{-}passing}, \mathit{broadcast})$ a formal rule,
- $(p_k)_{1 \le k \le n} \in (\mathcal{L}_p)^n$, $\mathit{param}_l^k \in (\mathcal{V}^*)^n$,
- $\mathit{constraints}^k \in \wp(\{x \diamond y \mid \diamond \in \{=; \neq\}, x, y \in \mathcal{V}\})^n$,
- $C^\sharp \in \mathcal{C}^\sharp_{\mathit{env}}$;

We compute $\textsc{sync}^\sharp(R_0 \cup \bigcup_{1 \le k \le n} R_k, (p^k), \mathit{mol})$ where:

- $\mathit{mol} \stackrel{\Delta}{=} \textsc{inj}^\sharp(C^\sharp(p_1)) \bullet \ldots \bullet \textsc{inj}^\sharp(C^\sharp(p_n))$;
- $R_0 \stackrel{\Delta}{=} \{\sigma(X) = \sigma(Y) \mid (X, Y) \in \mathit{compatibility}\}$,
  with $\sigma(I^k) = (I, k)$ and $\sigma(X_l^k) = (\mathit{param}_l^k, k)$;
- $\forall k \in [\![1; n]\!],\ R_k = \{(x, k) \diamond (y, k) \mid x \diamond y \in \mathit{constraints}^k\}$;

Abstract operational semantics: Marker computation and value passing

We perform name passing and marker allocation in the same time:

1. we introduce ghost variables;
2. we pass values to these ghost variables;
3. we compute markers;
4. we synchronize updated variable with their ghost twin.

Given:

- $\mathcal{R} = (n, \textit{components}, \textit{compatibility}, \textit{v-passing}, \textit{broadcast})$,
- $(p_k)_{1 \le k \le n} \in (\mathcal{L}_p)^n$, $\textit{param}_l^k \in (\mathcal{V}^*)^n$, $\textit{bd}_l^k \in (\mathcal{V}^*)^n$,
- $\textit{molecule}^\sharp \in \textit{Molecule}^\sharp_{(I(p^i))_{1 \le i \le n}}$,
- $(Z_l)_{l \in \mathbb{N}}$ be a family of fresh distinct ghost variables in $\mathcal{V}$;

We

- locate the variables that are bound by the communication:

  $\mathcal{A} \stackrel{\Delta}{=} \{(k, l) \mid 1 \le k \le n,\ 1 \le l \le \textit{n-vars}(\textit{components}(a))\}$;
- create ghost variables:

  $C_1 \stackrel{\Delta}{=} \mathrm{NEW}^\sharp_\top(\{(Z_l, k) \mid (k, l) \in \mathcal{A}\}, \textit{molecule}^\sharp)$;

- bind them to their values:

$$C_2 \stackrel{\Delta}{=} \text{SYNC}^\sharp(\mathit{cons}_1 \cup \mathit{cons}_2, (p^k), C_1), \text{ where:}$$

$$\begin{cases} \mathit{cons}_1 = \{(Z_l, k) = (\mathit{param}_{l'}^{k'}, k') \mid \exists k, k', l, l' \in \mathbb{N},\ \mathit{v\text{-}passing}(Y_l^k) = X_{l'}^{k'}\}, \\ \mathit{cons}_2 = \{(Z_l, k) = (I, k') \mid \exists k, k', l \in \mathbb{N},\ \mathit{v\text{-}passing}(Y_l^k) = I^{k'}\}, \end{cases}$$

- allocate markers:

$$C_3 \stackrel{\Delta}{=} \begin{cases} \text{FETCH}^\sharp((p^k), C_2) & \text{if } \mathit{type}(\mathit{components}(1)) = \mathit{replication} \\ C_2 & \text{otherwise.} \end{cases};$$

- forget useless information:

$$C_4 \stackrel{\Delta}{=} \text{NEW}_\top^\sharp(\{(\mathit{bd}_l^k, k) \mid (k, l) \in \mathcal{A}\}, C_3);$$

- copy ghost variable in their correct twin:

$$C_5 \stackrel{\Delta}{=} \text{SYNC}^\sharp(\{(Z_l, k) = (\mathit{bd}_l^k, k) \mid (k, l) \in \mathcal{A}\}, (p^k), C_4).$$

Abstract operational semantics: Fresh value allocation

Given:

- $E_s \in V_s \rightarrow$ *Label*,
  we can write *Dom*$(E_s) = \{x_i \mid 1 \leq i \leq q\}$;
- $A \in$ *Atom*$^{\sharp}_{V}$;

We compute C_n where:

- $C_0 \stackrel{\Delta}{=} \mathrm{GC}^{\sharp}(V \setminus \{\mathit{Dom}(E_s)\}, C)$,
- $C_{k+1} \stackrel{\Delta}{=} \nu^{\sharp}(x_{k+1}, E_s(x_{k+1}), C_k), \ \forall k \in [\![0; n[\![$.

Abstract operational semantics:
Continuation launching

Given:

- $\mathcal{R} = (n, \mathit{components}, \mathit{compatibility}, \mathit{v\text{-}passing}, \mathit{broadcast})$,
- $(p^k) \in \mathcal{L}_p^n$,
- $\mathit{mol} \in \mathit{Molecule}^\sharp_{(fv(p^k))_{1 \leq k \leq n}}$,
- $(\mathit{ct}^k) \in \wp(\wp(\mathcal{L}_p \times (\mathcal{V} \rightharpoonup \mathit{Label})))^n$;

We compute:

$$\left[p' \mapsto \bigsqcup_{I(p')} \left\{ \mathrm{GC}^\sharp(I(p'), \mathit{update}^\sharp(E_s, \mathrm{PROJ}^\sharp(k', \mathit{mol}))) \;\middle|\; \exists k',\ (p', E_s) \in \bigcup \mathit{ct}_{k'} \right\}\right.$$

Abstract operational semantics:
Broadcast value passing I

Given:

- *mol* the reactive molecule,
- $C^\sharp \in \mathcal{C}^\sharp_{\mathit{env}}$ the abstraction of the system before broacast value passing,
- $\mathcal{R} = (n, \mathit{components}, \mathit{compatibility}, \mathit{v\text{-}passing}, \mathit{broadcast})$,
- $\mathit{param}_l^k \in (\mathcal{V}^*)^n$;
- $p \in \mathcal{L}_p$;

We want to update the abstraction of threads of program point p.
First, we compute $\mathit{mol} \bullet \mathrm{INJ}^\sharp(C^\sharp(p))$.

Abstract operational semantics: Broadcast value passing II

Then, we condider all cases $\rho \in I(p) \to \{0\} \cup Dom(broadcast)$,
such that the value x of each variable v at program point p:

1. does not match with the value of any formal variable in $Dom(broadcast)$, and we write $(\rho(v) = 0)$,

   in such a case, the value x remain unchanged;

2. is substituted by the value of the formal variable $broadcast(X)$

   in such a case, x matches the value of X, and we write $(\rho(v) = X)$.

We partition according to all potentitial ρ.

Abstract operational semantics: Broadcast value passing III

For any ρ,

1. we take into account partitioning constraints;
2. we introduce a ghost variable for each variable x such that $\rho(x) \neq \emptyset$;
3. we synchronize these ghost variables with the correct value;
4. we forget any information about any variable x such that $\rho(x) \neq \emptyset$;
5. we synchronize replaced variables with their ghost twin.