History Dependent Automata: a Co-Algebraic definition, a Partitioning Algorithm and its Implementation

Roberto Raggi & Emilio Tuosto

joint work with

Gianluigi Ferrari, Ugo Montanari and Marco Pistore

Plan of the talk

Plan of the talk

- Motivations

Plan of the talk

- Motivations
- HD approach

Plan of the talk

- Motivations
- HD approach
- Co-algebraic definition of HD

Plan of the talk

- Motivations
- HD approach
- Co-algebraic definition of HD
- HD-automata for π-agents

Plan of the talk

- Motivations
- HD approach
- Co-algebraic definition of HD
- HD-automata for π-agents
- The partitioning algorithm

Plan of the talk

- Motivations
- HD approach
- Co-algebraic definition of HD
- HD-automata for π-agents
- The partitioning algorithm
- Principal data structures

Plan of the talk

- Motivations
- HD approach
- Co-algebraic definition of HD
- HD-automata for π-agents
- The partitioning algorithm
- Principal data structures
- The main cicle

Plan of the talk

- Motivations
- HD approach
- Co-algebraic definition of HD
- HD-automata for π-agents
- The partitioning algorithm
- Principal data structures
- The main cicle
- An example

Plan of the talk

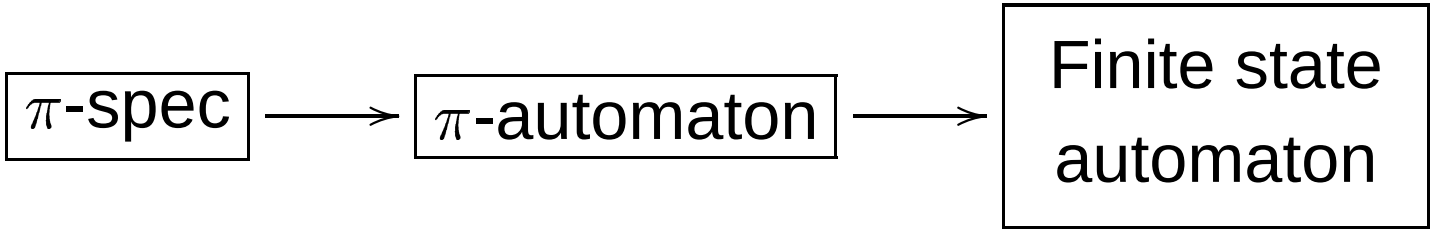
- Motivations
- HD approach
- Co-algebraic definition of HD
- HD-automata for π-agents
- The partitioning algorithm
- Principal data structures
- The main cicle
- An example
- Final considerations

Motivations

CAV98: π-spec of Handover protocol

Using HAL:

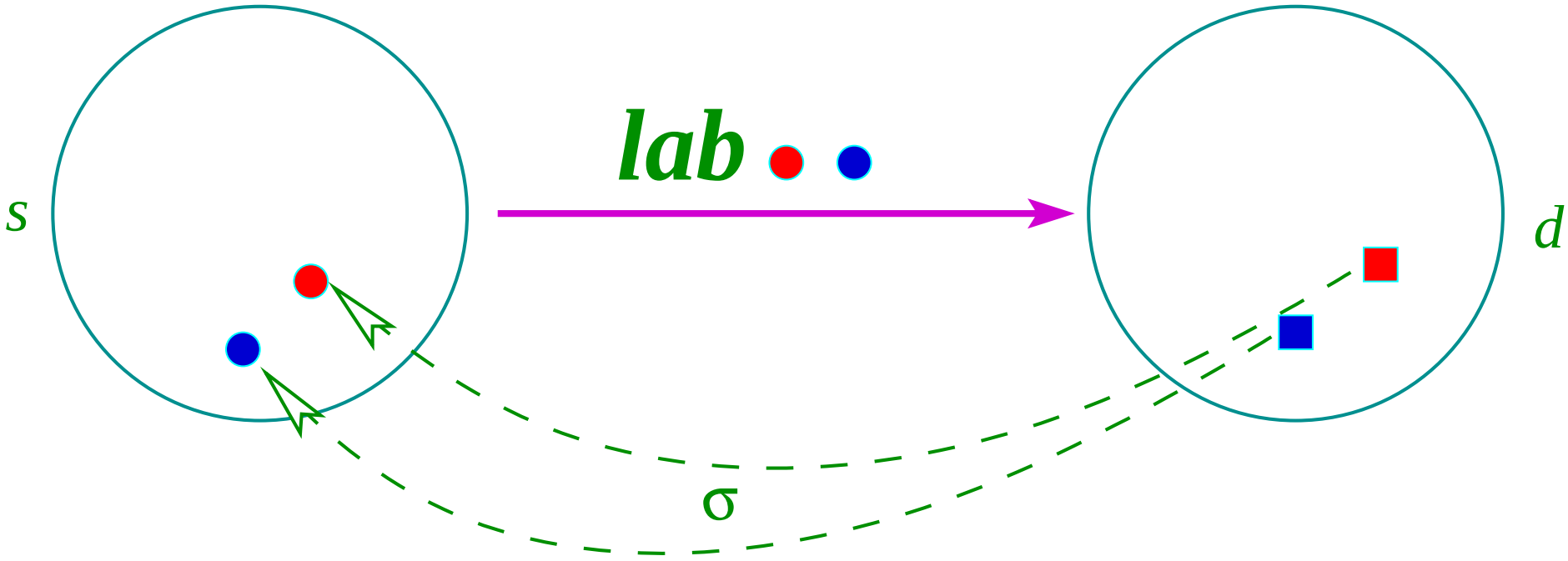
- 37199 states and 47958 Transitions
- Verification takes 15 min.



The approach

- HD as a model for name passing Calculi [Montanari & Pistore]
  - Specifically designed for verification purposes
  - Dynamic name allocation
  - Garbage collection of non-active names
  - Name symmetries
- Finite state representation of finite control π-agents
- Verification Techniques for HD-automata
- Semantic Minimization via Partition Refinement

HD: graphically



Basic Definitions

Transition System

$$T = (S, L, \rightarrow) \qquad \rightarrow \subseteq S \times L \times S$$

Notation

$$q \xrightarrow{l} q' \iff (q, l, q') \in \rightarrow$$

Basic Definitions

Transition System

$T = (S, L, \rightarrow) \quad \rightarrow \subseteq S \times L \times S$

Notation

$q \xrightarrow{l} q' \iff (q, l, q') \in \rightarrow$

T co-algebraically: $K : S \rightarrow \wp(L \times S)$

$K : q \mapsto \{(l, q') : q \xrightarrow{l} q'\}$

Basic Definitions

Transition System

$T = (S, L, \rightarrow) \quad \rightarrow \subseteq S \times L \times S$

Notation

$q \xrightarrow{l} q' \iff (q, l, q') \in \rightarrow$

T co-algebraically: $K : S \rightarrow \wp(L \times S)$

$K : q \mapsto \{(l, q') : q \xrightarrow{l} q'\}$

(step of a) bundle

Basic Definitions

Transition System

$T = (S, L, \rightarrow) \quad \rightarrow \subseteq S \times L \times S$

Notation

$q \xrightarrow{l} q' \iff (q, l, q') \in \rightarrow$

T co-algebraically: $K : S \rightarrow \wp(L \times S)$

$K : q \mapsto \{(l, q') : q \xrightarrow{l} q'\}$

(step of a) bundle $\beta = \langle D : \textbf{Set}, Step : \wp(L \times D) \rangle$

B collection of bundles

Basic Definitions

Transition System

$T = (S, L, \rightarrow) \quad \rightarrow \subseteq S \times L \times S$

Notation

$q \xrightarrow{l} q' \iff (q, l, q') \in \rightarrow$

T co-algebraically: $K : S \rightarrow \wp(L \times S)$

$K : q \mapsto \{(l, q') : q \xrightarrow{l} q'\}$

(step of a) bundle $\beta = \langle D : \textbf{Set}, Step : \wp(L \times D) \rangle$

B collection of bundles

HD-automata are co-algebras defi ned on top of a permutation algebra [MFCS2000]

Basic Definitions

Transition System

$T = (S, L, \rightarrow) \quad \rightarrow \subseteq S \times L \times S$

Notation

$q \xrightarrow{l} q' \iff (q, l, q') \in \rightarrow$

T co-algebraically: $K : S \rightarrow \wp(L \times S)$

$K : q \mapsto \{(l, q') : q \xrightarrow{l} q'\}$

(step of a) bundle $\beta = \langle D : \textbf{Set}, Step : \wp(L \times D) \rangle$

B collection of bundles

HD-automata are co-algebras defi ned on top of a permutation algebra [MFCS2000]

$\mathcal{H}$ is a HD-automata $\Rightarrow \exists \hat{\mathcal{H}} : \mathcal{H} \sim \hat{\mathcal{H}}$ and $\hat{\mathcal{H}}$ is minimal

Some notations

Set collection of sets

$: \triangleq \in$

Fun collection of arrows

$H = \langle S : \textbf{Set}, D : \textbf{Set}, h : S \to D \rangle$

$S_H \triangleq S, \quad D_H \triangleq D, \quad h_H \triangleq h$

$H; K$ composition of H, K : **Fun**

$S_{H;K} = S_H \quad D_{H;K} = D_K \quad h_{H;K} = h_K \circ h_H$

where $D_H = S_K$

A functor for transition systems

$T(Q) = \{\beta : B | D_\beta = Q\}$

Let H : **Fun** we define $T(H)$: **Fun** s.t.

- $S_{T(H)} = T(S_H)$
- $D_{T(H)} = T(D_H)$
- $h_{T(H)} : \beta \mapsto \langle D_H, \{\langle l, h_H(q)\rangle | \langle l, q\rangle \in Step_\beta\rangle$

HD definitions: Named Sets

NSet

$$A = \langle Q : \textbf{Set},$$
$$| \ | : Q \to \omega,$$
$$\leq : \ Q \times Q \to Bool,$$
$$G : \prod_{q:Q} \wp(\{q\}_A) \stackrel{bij}{\to} \{q\}_A \rangle$$

where $\{q\}_A = v_1, ..., v_{|q|}$

HD definitions: Named Sets

NSet

$$A = \langle Q : \textbf{Set},$$
$$| \ | : Q \to \omega,$$
$$\leq : \ Q \times Q \to Bool,$$
$$G : \prod_{q:Q} \wp(\{q\}_A) \stackrel{bij}{\to} \{q\}_A \rangle$$

where $\{q\}_A = v_1, ..., v_{|q|}$

Let $p(v_1, ..., v_n)$ be a π-agent

- $Q_A = \{q : p(v_1, ..., v_n) \stackrel{l_1 ... l_t}{\to} q\}$
- $|p(v_1, ..., v_n)|_A = n,$
- a total order
- $G_A(q) = \{\rho : \{q\}_A \to \{q\}_A\}$

HD definitions: Named Functions

NFun

$$H = \langle S : \textbf{NSet},$$
$$D : \textbf{NSet},$$
$$h : Q_S \to Q_D,$$
$$\Sigma : \prod_{q:Q_S} \wp(\{h(q)\}_D \stackrel{inj}{\to} \{q\}_S)\rangle$$

HD definitions: Named Functions

NFun

$$H = \langle S : \textbf{NSet},$$
$$D : \textbf{NSet},$$
$$h : Q_S \to Q_D,$$
$$\Sigma : \prod_{q:Q_S} \wp(\{h(q)\}_D \overset{inj}{\to} \{q\}_S)\rangle$$

Properties:

$\forall \sigma : \Sigma_H(q)$

- $G_{D_H}(h_H(q)); \sigma = \Sigma_H(q)$
- $\sigma; G_{S_H}(q) \subseteq \Sigma_H(q)$
- composition is trivially defined

HD definitions: Named Functions

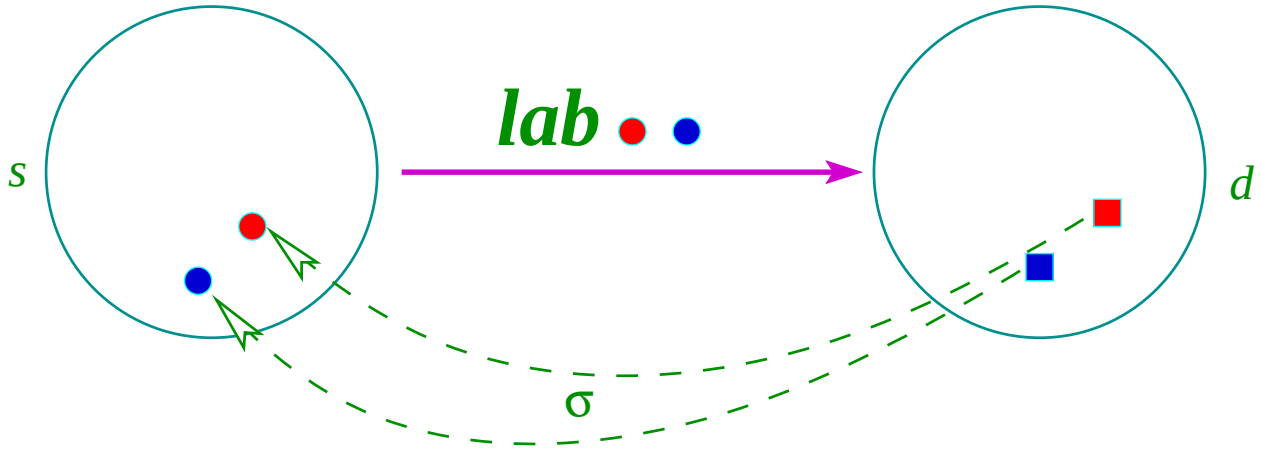
NFun

$$H = \langle S : \textbf{NSet},$$
$$D : \textbf{NSet},$$
$$h : Q_S \to Q_D,$$
$$\Sigma : \prod_{q:Q_S} \wp(\{h(q)\}_D \stackrel{inj}{\to} \{q\}_S)\rangle$$

Properties:

$\forall \sigma : \Sigma_H(q)$

- $G_{D_H}(h_H(q)); \sigma = \Sigma_H(q)$
- $\sigma; G_{S_H}(q) \subseteq \Sigma_H(q)$
- composition is trivially defined



HD definitions: Named Functions

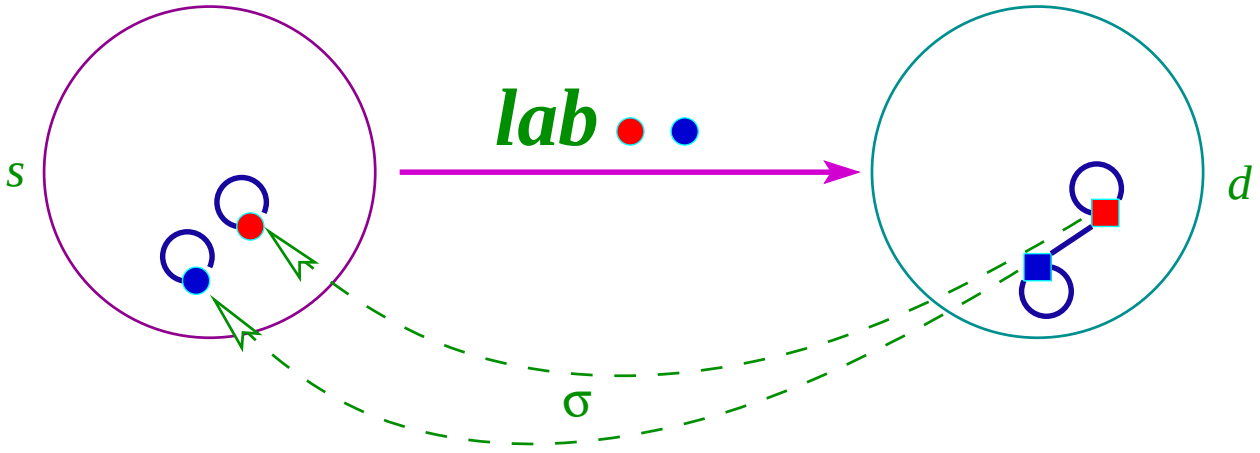
NFun

$$H = \langle S : \textbf{NSet},$$
$$D : \textbf{NSet},$$
$$h : Q_S \to Q_D,$$
$$\Sigma : \prod_{q:Q_S} \wp(\{h(q)\}_D \overset{inj}{\to} \{q\}_S)\rangle$$

Properties:

$\forall \sigma : \Sigma_H(q)$

- $G_{D_H}(h_H(q)); \sigma = \Sigma_H(q)$
- $\sigma; G_{S_H}(q) \subseteq \Sigma_H(q)$
- composition is trivially defined



HD-automata for π-agents

	TAU	IN	OUT	BIN	$BOUT$
$\|l\|$	$\emptyset$	$\{1,2\}$	$\{1,2\}$	$\{1\}$	$\{1\}$

HD-automata for π-agents

Bundle $\beta = \langle D : \mathbf{NSet},\ Step : \wp(qd\ D)\rangle$

$Step = \{..., \langle l,\ \pi,\ \sigma,\ q\rangle, ...\}$

HD-automata for π-agents

Bundle $\beta = \langle D : \mathbf{NSet},\ Step : \wp(qd\ D)\rangle$

$Step = \{..., \langle l,\ \pi,\ \sigma,\ q\rangle, ...\}$

π-calculus label

HD-automata for π-agents

Bundle $\beta = \langle D : \textbf{NSet},\ Step : \wp(qd\ D)\rangle$

$Step = \{..., \langle l,\ \pi,\ \sigma,\ q\rangle, ...\}$

: $|l| \rightarrow N$ observable names of the transition

HD-automata for π-agents

Bundle $\beta = \langle D : \textbf{NSet},\ Step : \wp(qd\ D)\rangle$

$Step = \{..., \langle l,\ \pi,\ \sigma,\ q\rangle, ...\}$

: $\{q\}_\bullet \to N$ meaning of the names of q

HD-automata for π-agents

Bundle $\beta = \langle D : \mathbf{NSet},\ Step : \wp(qd\ D)\rangle$

$Step = \{..., \langle l,\ \pi,\ \sigma,\ q\rangle, ...\}$

destination state

HD-automata for π-agents

Bundle $\beta = \langle D : \mathbf{NSet},\ Step : \wp(qd\ D)\rangle$

$Step = \{..., \langle l,\ \pi,\ \sigma,\ q\rangle, ...\}$

$S_q = \{\langle l, \pi, \sigma, q\rangle \in Step_\beta\}$

$\boxed{G_{D_\beta}(q); S_q = S_q}$

$\rho; \langle l, \pi, \sigma, q\rangle \ = \ \langle l, \pi, \rho; \sigma, q\rangle$

HD-automata for π-agents

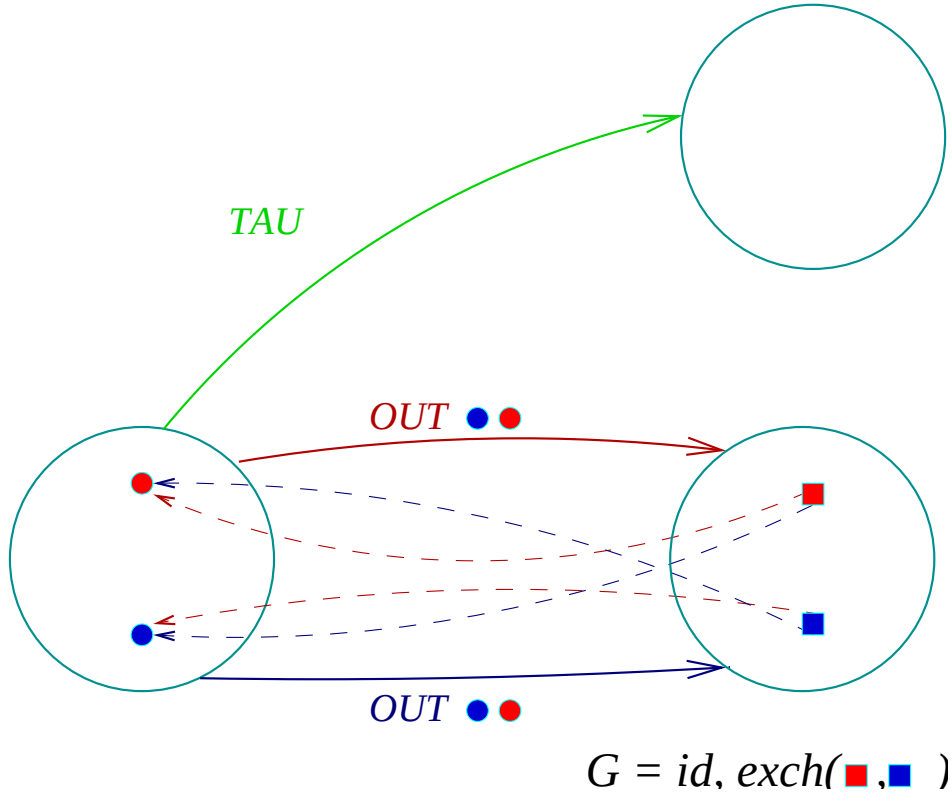
Bundle $\beta = \langle D : \mathbf{NSet},\ Step : \wp(qd\ D)\rangle$

$Step = \{..., \langle l,\ \pi,\ \sigma,\ q\rangle, ...\}$

$S_q = \{\langle l, \pi, \sigma, q\rangle \in Step_\beta\}$

$$\boxed{G_{D_\beta}(q); S_q = S_q}$$

$\rho; \langle l, \pi, \sigma, q\rangle \ = \ \langle l, \pi, \rho; \sigma, q\rangle$

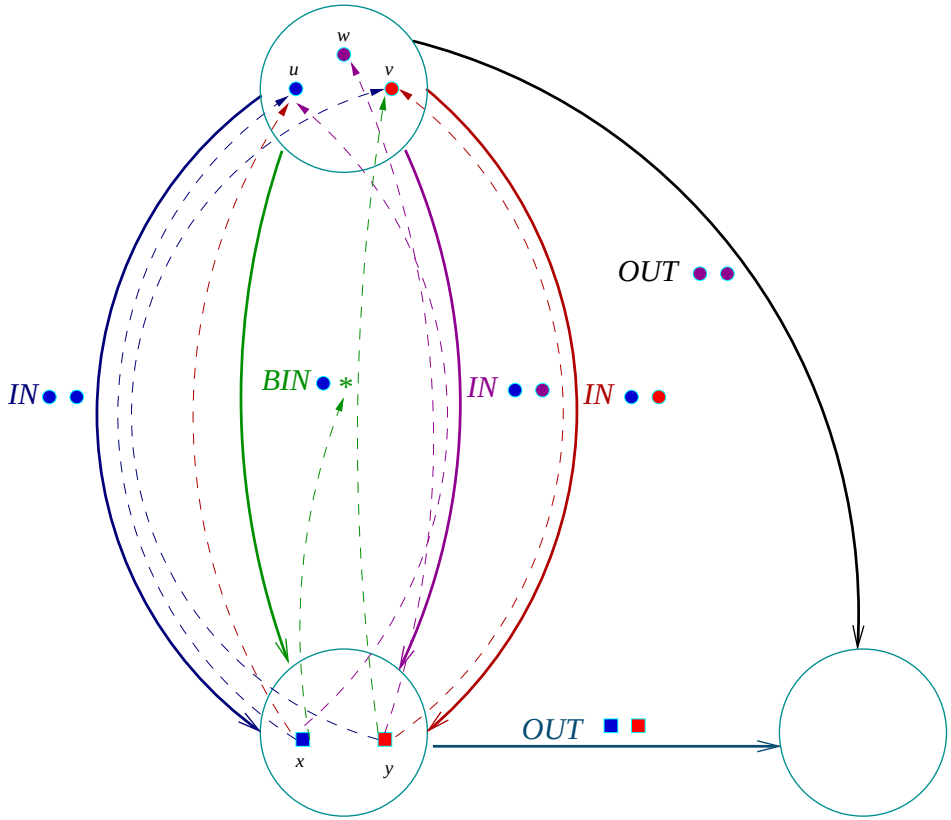


HD-automata and name creation

$$A(u,v,w) \;=\; u(x).\overline{x}v.nil \;+\; \overline{w}w.nil$$

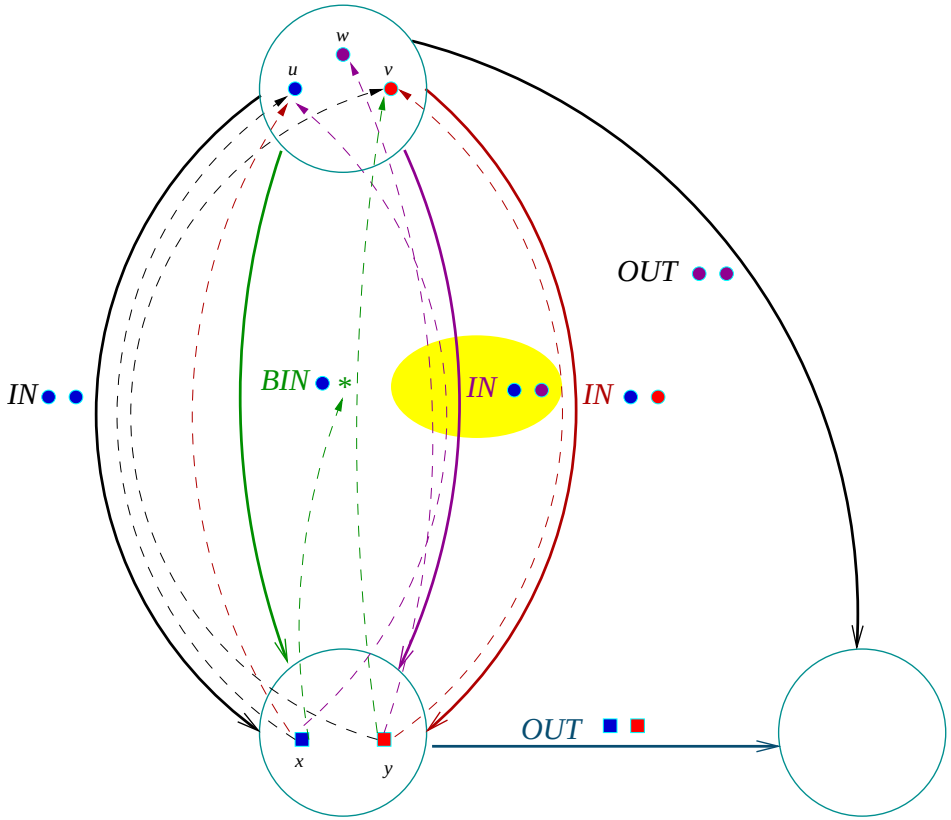
HD-automata and name creation

$$A(u,v,w) \;=\; u(x).\overline{x}v.nil \;+\; \overline{w}w.nil$$



HD-automata and name creation

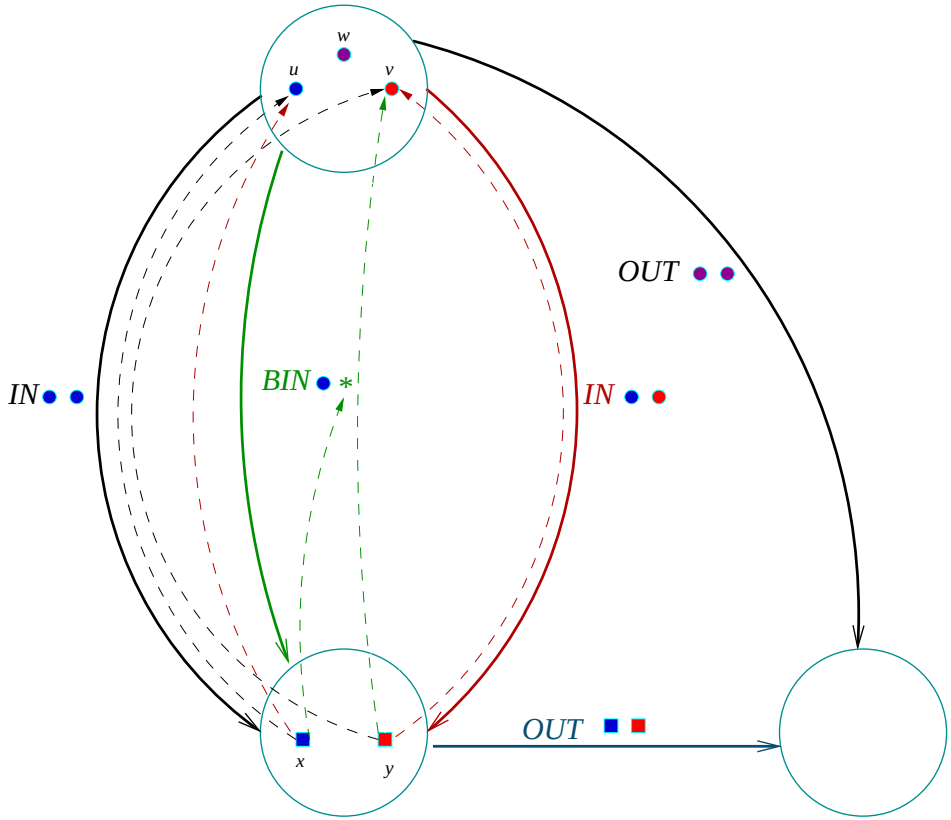
$$A(u,v,w) \;=\; u(x).\overline{x}v.nil \;+\; \overline{w}w.nil$$



$$an_\beta = \bigcup_{\langle l,\pi,\sigma,q\rangle\in\beta} rng\ \pi \cup rng\ \sigma \;=\; \{u,v\}$$

HD-automata and name creation

$$A(u,v,w) \;=\; u(x).\overline{x}v.nil \;+\; \overline{w}w.nil$$



$$an_\beta = \bigcup_{\langle l,\pi,\sigma,q\rangle\in\beta} rng\ \pi \cup rng\ \sigma \;=\; \{u,v\}$$

Bundle normalization

Bundle normalization

- compute the redundant transitions

Bundle normalization

- compute the redundant transitions
- compute the active names of a bundle

Bundle normalization

- compute the redundant transitions
- compute the active names of a bundle
- remove dominated transitions

Bundle normalization

- compute the redundant transitions
- compute the active names of a bundle
- remove dominated transitions
- select the canonical bundle according to the order relation

The functor on NSet

T is an endo-functor on **NSet**:

- $Q_{T(A)} = \{\beta | D_\beta = A \wedge \beta$ normalized$\}$
- $|\beta|_{T(A)} =$ number of names of β
- $\beta_1 \leq_{T(A)} \beta_2$ iff $Step_{\beta_1} \leq Step_{\beta_2}$
- $G_{T(A)}(\beta) =$ group of β

The functor on NFun

...while on named functions:

- $S_{T(H)} = T(S_H)$
- $D_{T(H)} = T(D_H)$
- $h_{T(H)}(\beta) = norm(\beta')$
  $\beta' = \langle D_H, \{\langle l, \pi, \sigma'; \sigma, h_H(q)\rangle | \langle l, \pi, \sigma, q\rangle : Step_\beta \wedge \sigma' : \Sigma_H(q)\}\rangle$
- $\Sigma_{T(H)}(\beta) = Gr\ norm(\beta'); perm^{-1}(\beta')$

The functor on NFun

...while on named functions:

- $S_{T(H)} = T(S_H)$
- $D_{T(H)} = T(D_H)$
- $h_{T(H)}(\beta) = norm(\beta')$
  $\beta' = \langle D_H, \{\langle l, \pi, \sigma'; \sigma, h_H(q)\rangle | \langle l, \pi, \sigma, q\rangle : Step_\beta \wedge \sigma' : \Sigma_H(q)\}\rangle$
- $\Sigma_{T(H)}(\beta) = Gr\ norm(\beta'); perm^{-1}(\beta')$

A transition system over **NSet** and π-actions is a named function K such that $D_K = T(S_K)$

HD-automata as T-coalgebras

Let $p(v_1, ..., v_n)$ be a π-agent

- $Q_A = \{q : p(v_1, ..., v_n) \stackrel{l_1...l_t}{\rightarrow} q\}$
- $|p(v_1, ..., v_n)|_A = n,$
- $G_A(q) = \{id : \{q\}_A \rightarrow \{q\}_A\}$

HD-automata as T-coalgebras

Let $p(v_1, ..., v_n)$ be a π-agent

- $Q_A = \{q : p(v_1, ..., v_n) \stackrel{l_1 ... l_t}{\rightarrow} q\}$
- $|p(v_1, ..., v_n)|_A = n,$
- $G_A(q) = \{id : \{q\}_A \rightarrow \{q\}_A\}$

$\alpha : Q_A \rightarrow \{\beta | D_\beta = A\}$

HD-automata as T-coalgebras

Let $p(v_1, ..., v_n)$ be a π-agent

- $Q_A = \{q : p(v_1, ..., v_n) \stackrel{l_1...l_t}{\rightarrow} q\}$
- $|p(v_1, ..., v_n)|_A = n,$
- $G_A(q) = \{id : \{q\}_A \rightarrow \{q\}_A\}$

$\alpha : Q_A \rightarrow \{\beta | D_\beta = A\}$

$K = \langle A,\ T(A),\ \Sigma \rangle,$ where $\Sigma(q) = Gr(h_K(q)); perm^{-1}(\alpha(q))$

Partition Refinement Algorithm

Initial approximation H_0:

- $S_{H_0} = S_K$
- $D_{H_0} = \perp, \quad Q_\perp = \{\star\}, |\star|_\perp = 0$
- $G_\perp \star = \emptyset$
- $h_{H_0}(q) = \star$
- $\Sigma_{H_0}\, q = \{\emptyset\}$

Partition Refinement Algorithm

Initial approximation H_0:

- $S_{H_0} = S_K$
- $D_{H_0} = \perp, \quad Q_\perp = \{\star\}, |\star|_\perp = 0$
- $G_\perp \star = \emptyset$
- $h_{H_0}(q) = \star$
- $\Sigma_{H_0}\, q = \{\emptyset\}$

$$H_{n+1} = K; T(H_n)$$

Partition Refinement Algorithm

Initial approximation H_0:

- $S_{H_0} = S_K$
- $D_{H_0} = \perp, \quad Q_\perp = \{\star\}, |\star|_\perp = 0$
- $G_\perp \star = \emptyset$
- $h_{H_0}(q) = \star$
- $\Sigma_{H_0}\, q = \{\emptyset\}$

$$H_{n+1} = K; T(H_n)$$

Theorem If K is a finite HD,

- $\exists \bar{n} : D_{H_{\bar{n}+1}} \equiv D_{H_{\bar{n}}}$
- The isomorphism $F : D_{H_{\bar{n}}} \to D_{H_{\bar{n}+1}}$ yields the minimal realization of K up to strong early bisimilarity

Partition Refinement Algorithm (2)

At each step:

- a block is splitted: $h_{H_n}(q) = h_{H_n}(q')$ and $h_{H_{n+1}}(q) \neq h_{H_{n+1}}(q')$
- new names may be introduced

Partition Refinement Algorithm (2)

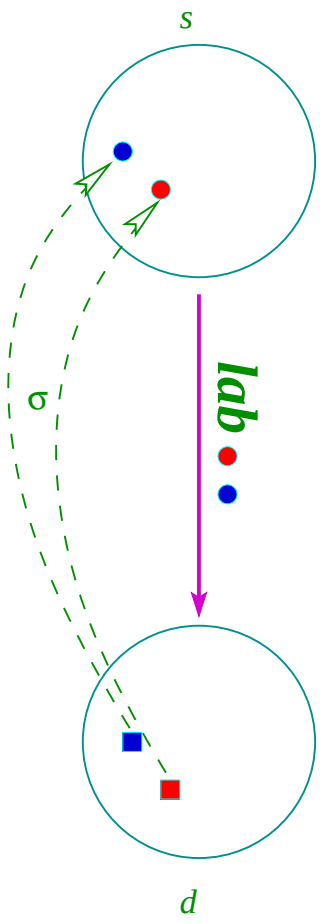
At each step:

- a block is splitted: $h_{H_n}(q) = h_{H_n}(q')$ and $h_{H_{n+1}}(q) \neq h_{H_{n+1}}(q')$
- new names may be introduced

The iteration step:

$$h_{H_{n+1}} = norm\langle D_{H_n},\ \{\langle l, \pi, \sigma'; \sigma, h_{H_n}(q')\rangle \mid q \xrightarrow{l,\pi,\sigma} q' \wedge \sigma' : \Sigma_{H_n}(q')\}$$

States, labels & arrows



type α state =

| Star

| State of $\underbrace{\text{string}}_{\text{id}} * \underbrace{\alpha\ \text{list}}_{\text{names}} * \underbrace{(\alpha\ \text{list})\ \text{list}}_{\text{group}}$

type α label =

| Tau of α list

| BIn of α list $*$ α list

| BOut of α list $*$ α list

| In of α list $*$ α list

| Out of $\underbrace{\alpha\ \text{list}}_{\pi} * \underbrace{\alpha\ \text{list}}_{\sigma}$

type α arrow =

Arrow of $\underbrace{\alpha\ \text{state}}_{\text{source}} * \underbrace{\alpha\ \text{state}}_{\text{dest}} * \underbrace{\alpha\ \text{label}}_{\text{lab}}$

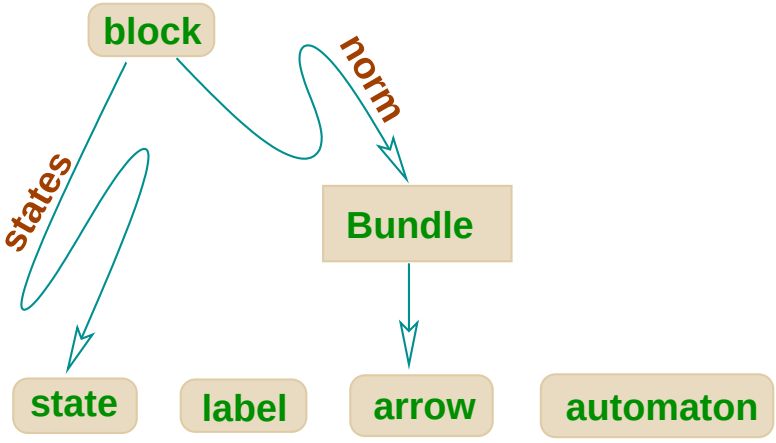
Bundle & Automata

bundle : $\underbrace{(\alpha \text{ arrow}) \quad \text{list}}_{\text{with the same source}}$

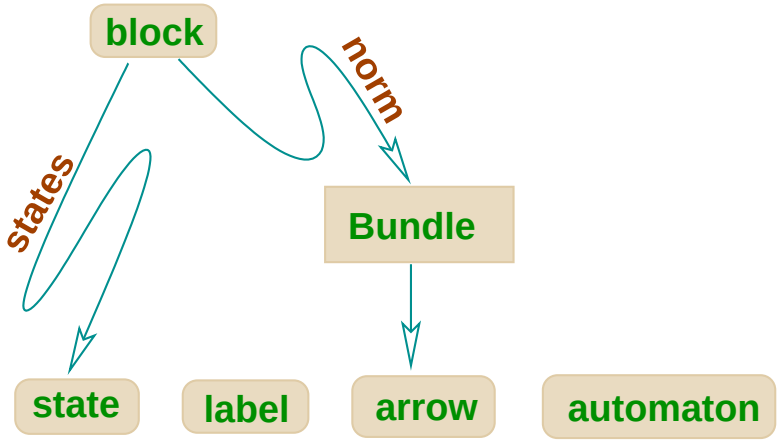
type α automaton =

HDAutoma of $\underbrace{\alpha \text{ state}}_{\text{start}} * \underbrace{(\alpha \text{ state}) \text{ list}}_{\text{states}} * \underbrace{(\alpha \text{ arrow}) \text{ list}}_{\text{arrows}}$

Blocks



Blocks



At the end of each iteration,

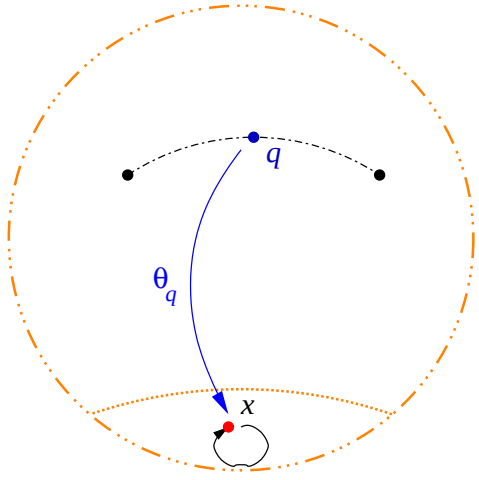
- blocks represent the states of the n-th approximation of the minimal automaton while
- their norm components are the arrows of the approximation

Blocks (2)

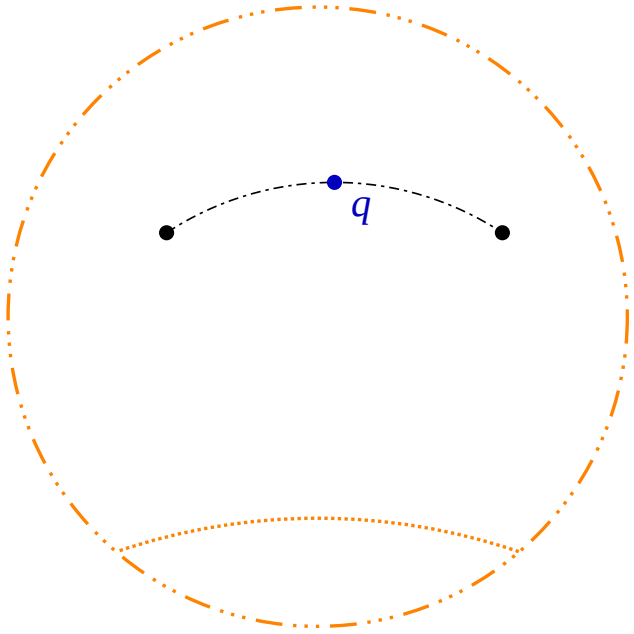
type α blocks =
  Block of
    states : α state list $*$
    norm : α arrow list $*$
    active_names : α list $*$
    group : α list list $*$
    Σ : (α state $\rightarrow$ (α $*$ α) list list) $*$
    Θ^{-1} : (α state $\rightarrow$ (α $*$ α) list)

Blocks (2)

type α blocks =
  Block of
    states : α state list $*$
    norm : α arrow list $*$
    active_names : α list $*$
    group : α list list $*$
    Σ : (α state $\rightarrow$ (α $*$ α) list list) $*$
    Θ^{-1} : (α state $\rightarrow$ (α $*$ α) list)



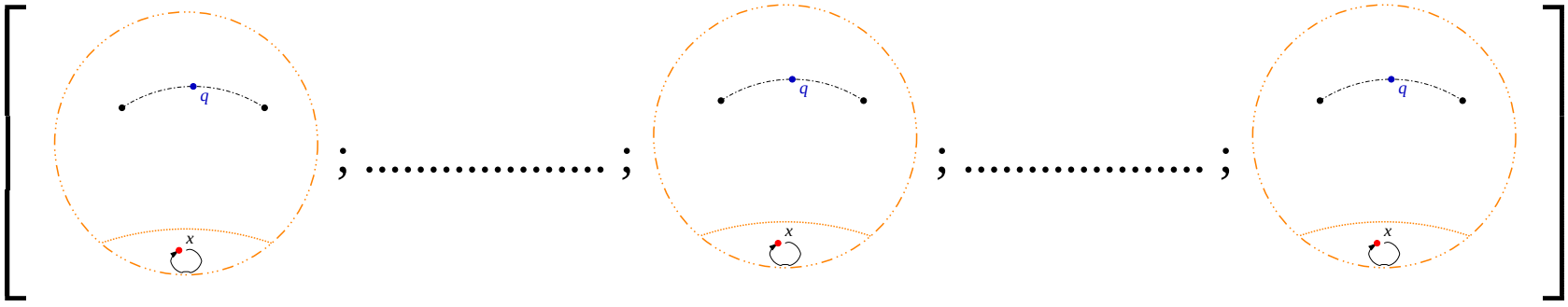
Initially...



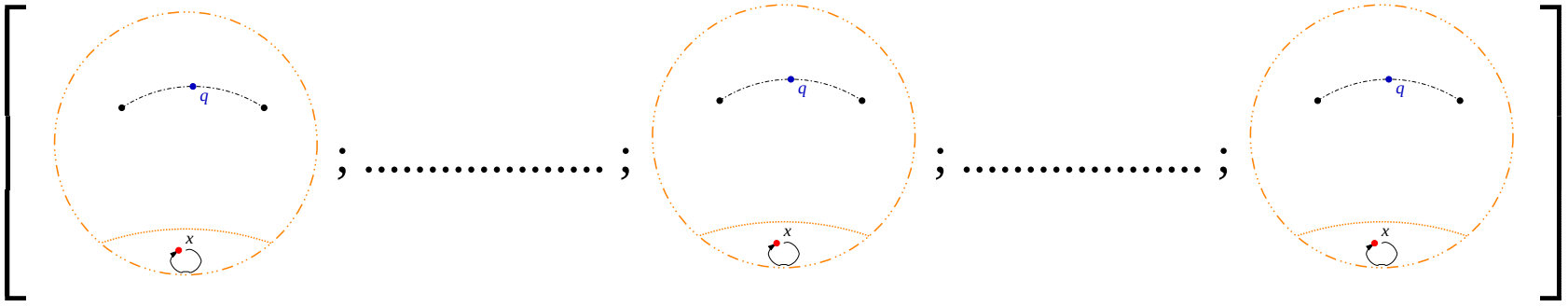
- All the states are (considered) bisimilar
- No norm, group or θ is given

[**Block**(states, [ ], [ ], [ ], (fun q $\rightarrow$ [[ ]]), (fun q $\rightarrow$ [ ]))]

Generic step

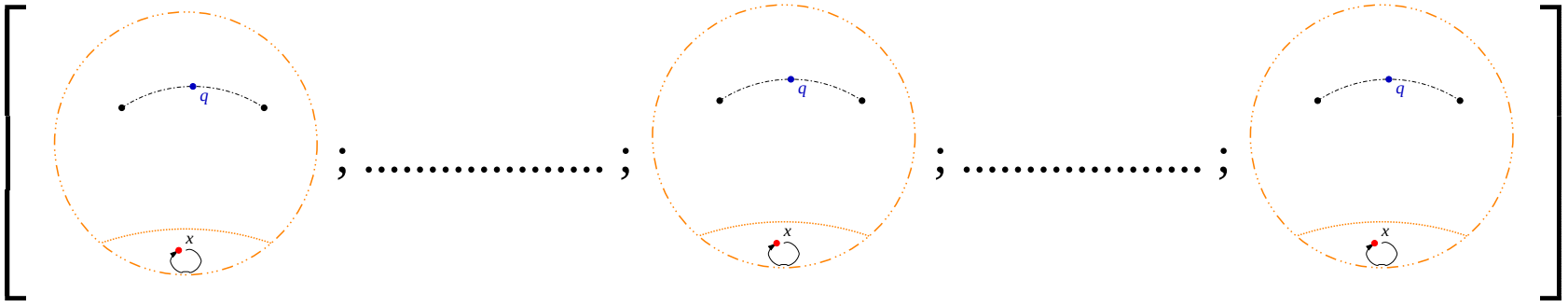


Generic step

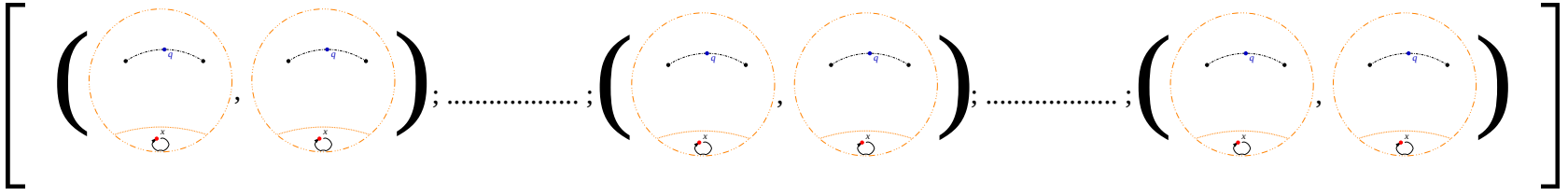


↓ split

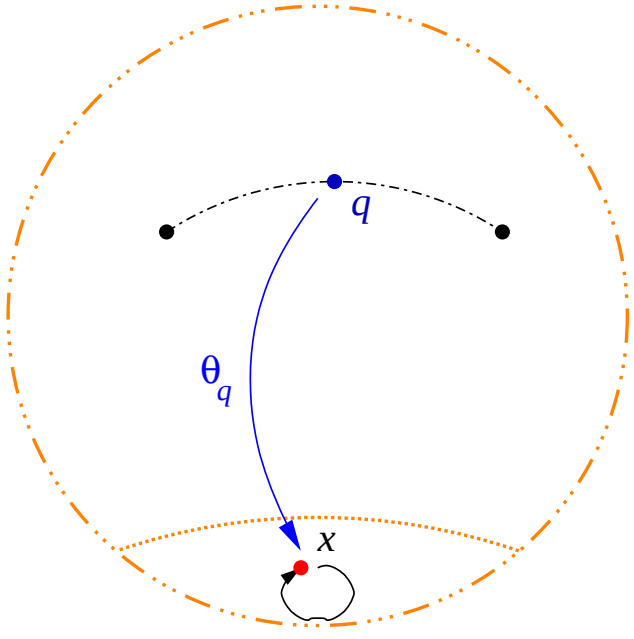
Generic step



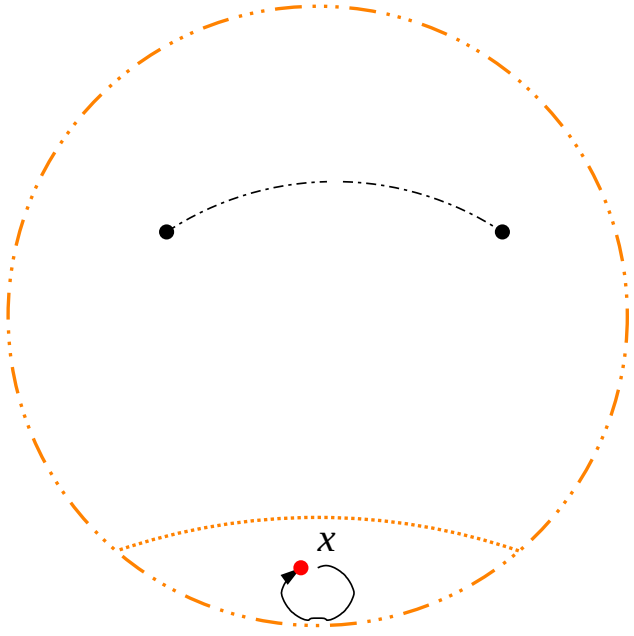
↓ split

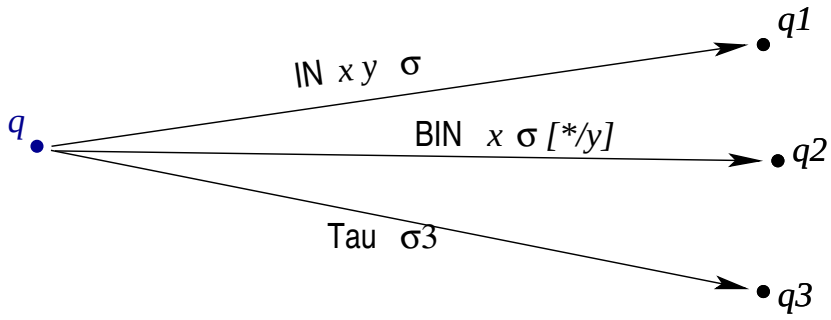


Splitting: a closer look



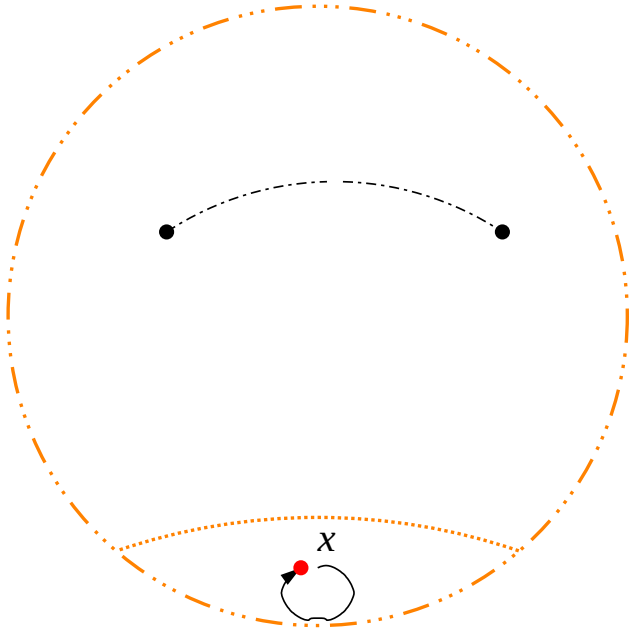
Splitting: a closer look

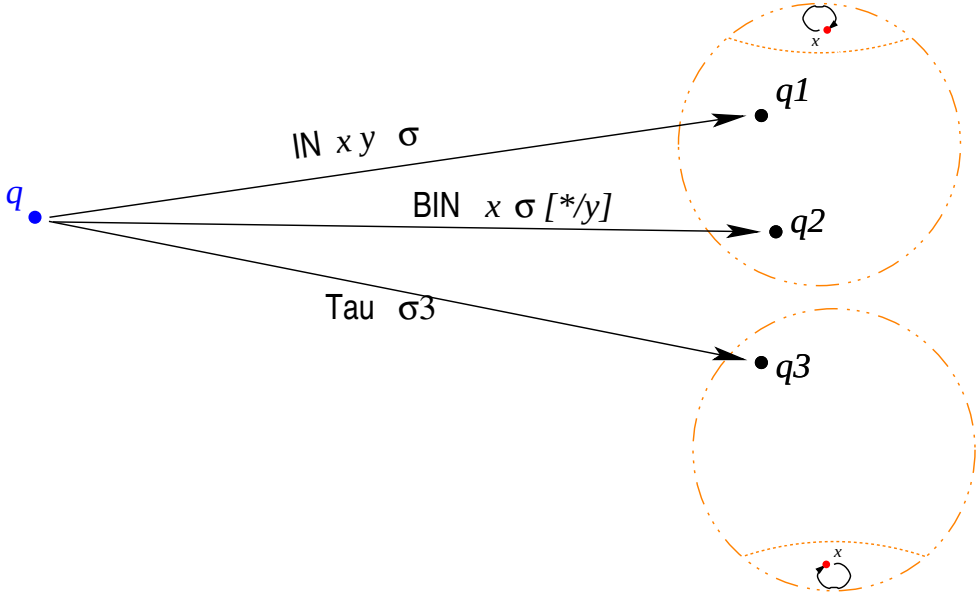





```
let bundle hd q =
  List.sort compare
    (List.filter (fun h → (Arrow.source h) = q) (arrows hd))
```

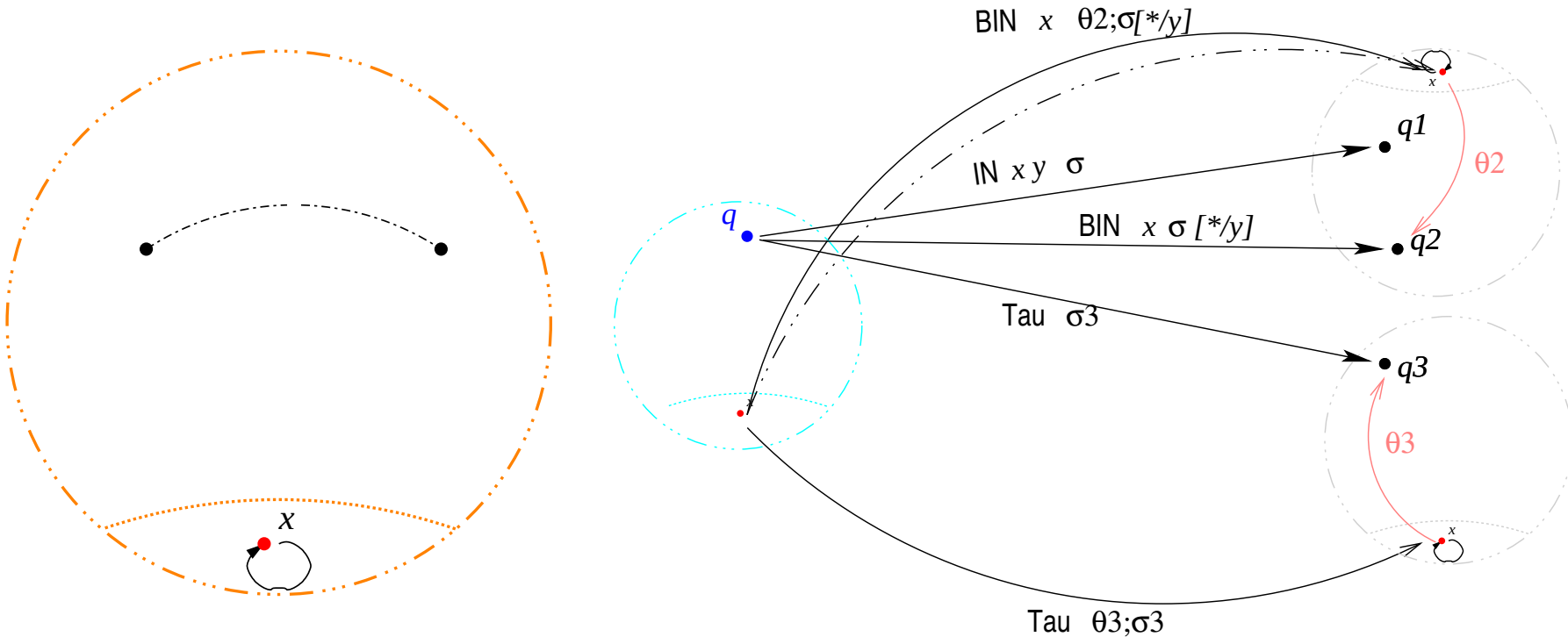
Splitting: a closer look





List.map h_n bundle

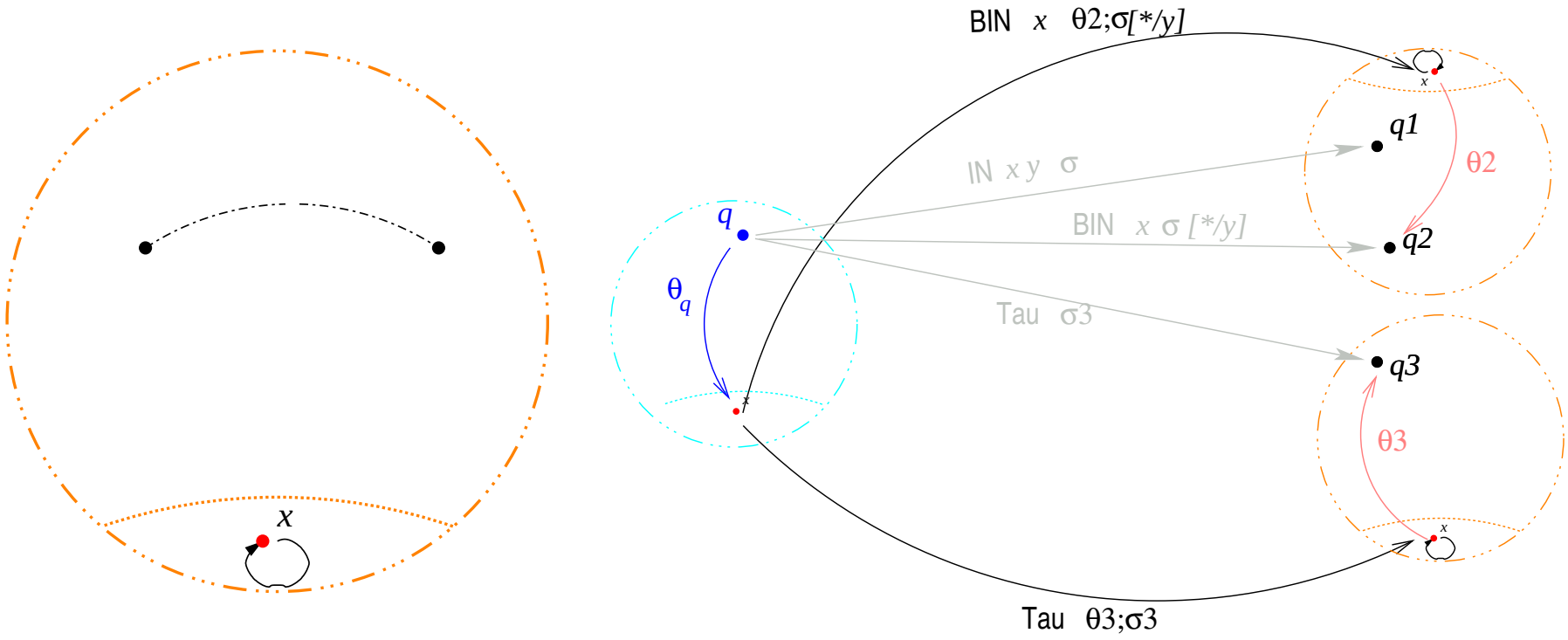
Splitting: a closer look



$$h_{n+1} = norm\langle \text{states}, \{\langle \ell, \pi, h_n(q'), \sigma'; \sigma\rangle | q \xrightarrow{\ell\ \pi\ \sigma} q' \wedge \sigma' \in \Sigma_n(q')\}\rangle$$

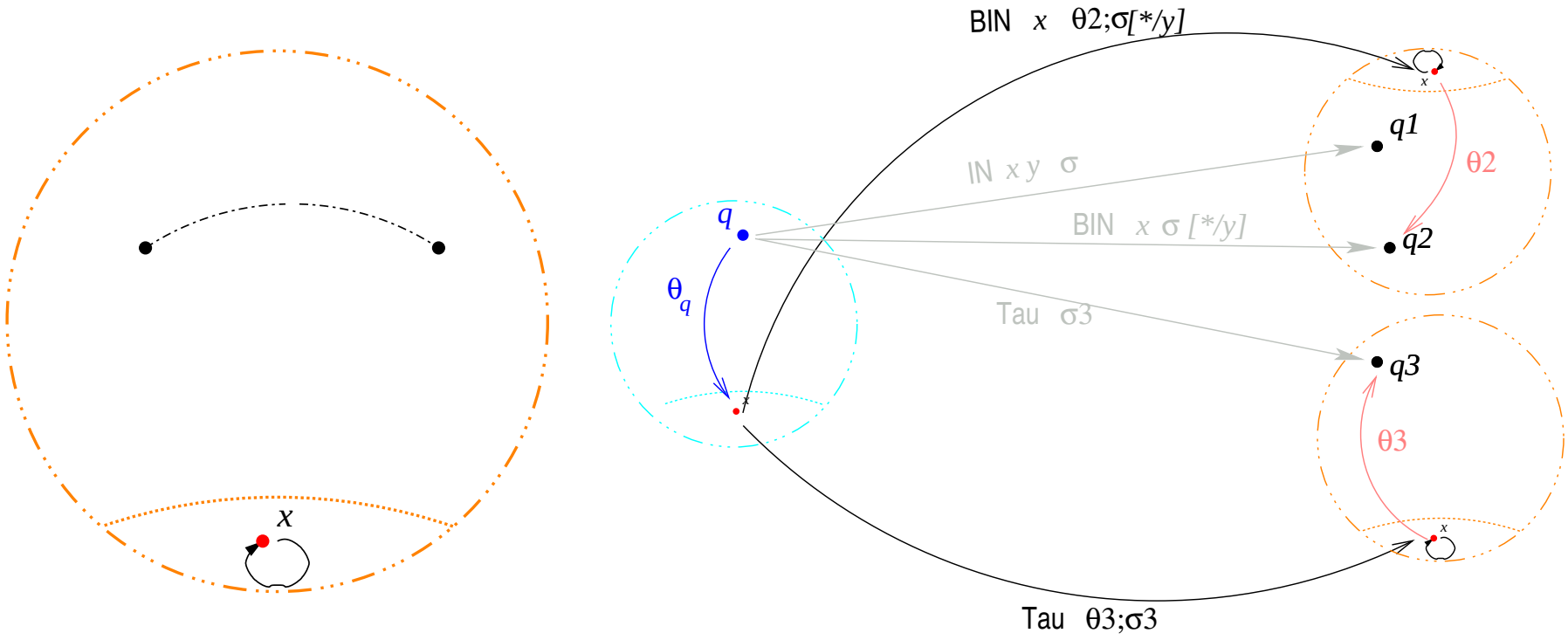
```
let red bl = .....
  let bl_in = List.fi lter covered_inbl
  in list_diff bl bl_in
```

Splitting: a closer look




```
let an = active_names_bundle (red bundle) in
let remove_in ar = match ar with
   | Arrow(_,_,In(_,_)) → not (List.mem (obj ar) an)
   | _ → false in
     list_diff bundle (List.filter remove_in bundle)
```

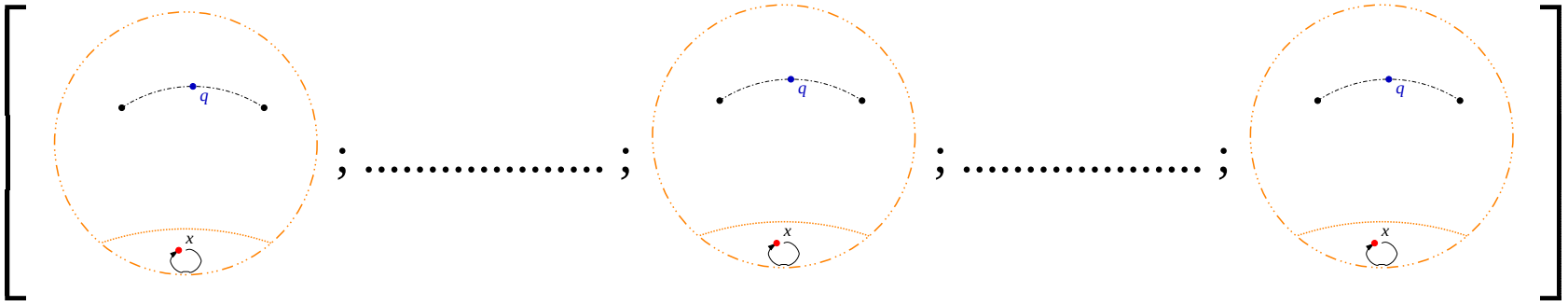
Splitting: a closer look



$$\Sigma_{n+1}(q) = (\text{compute\_group}\ \ (\text{norm}\ \ \text{bundle}))\ \ ;\ \ \theta_q^{-1}$$

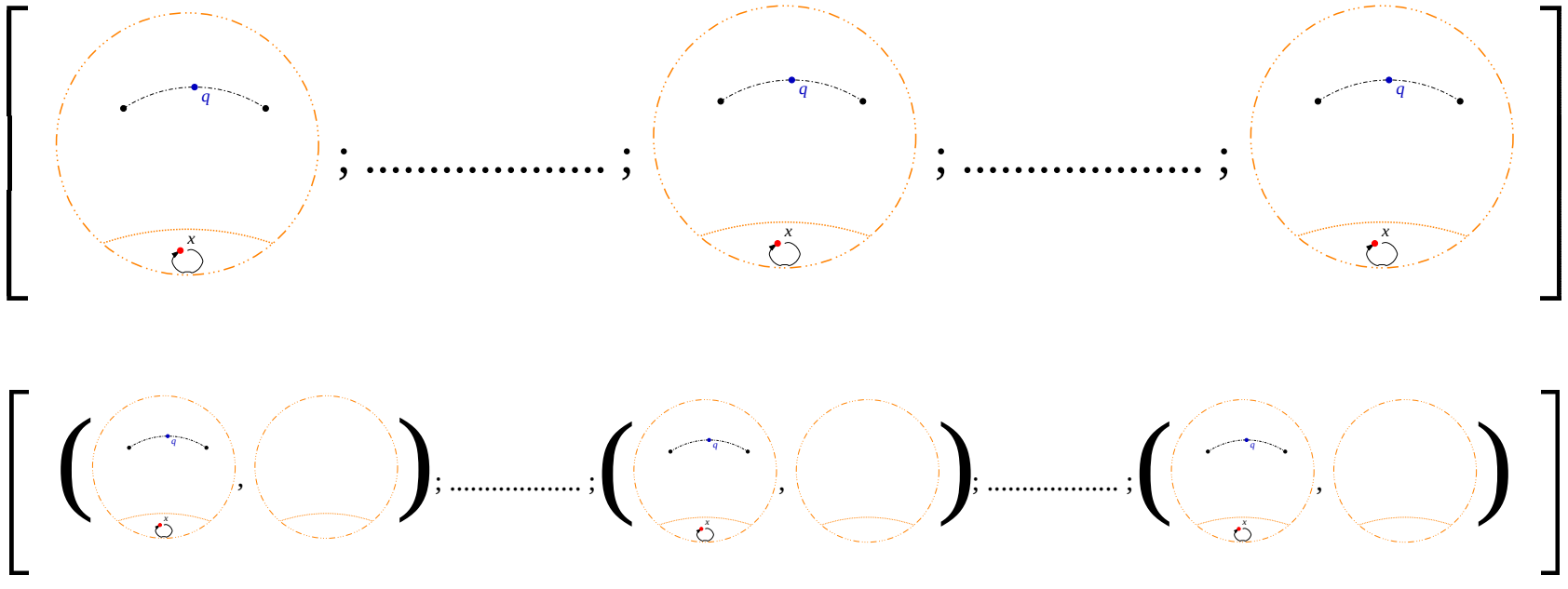
Termination

...informally, when H_{n+1} is isomorph to H_n



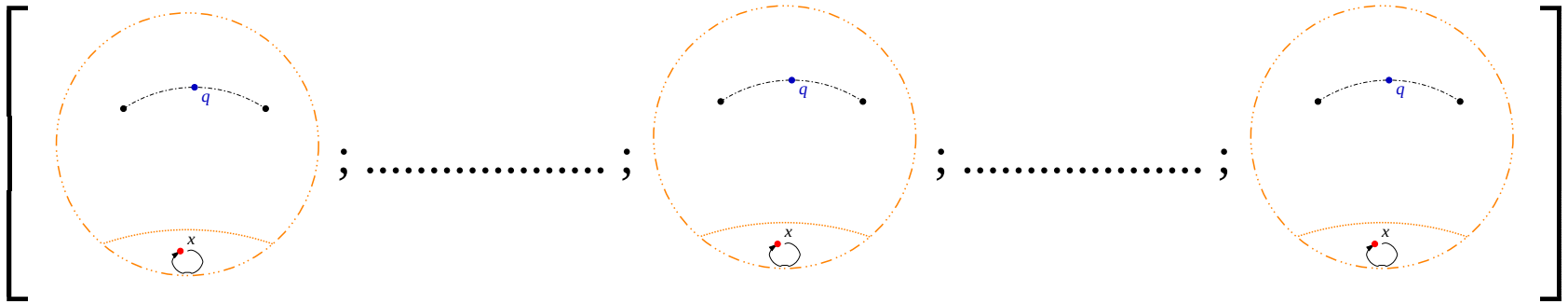
Termination

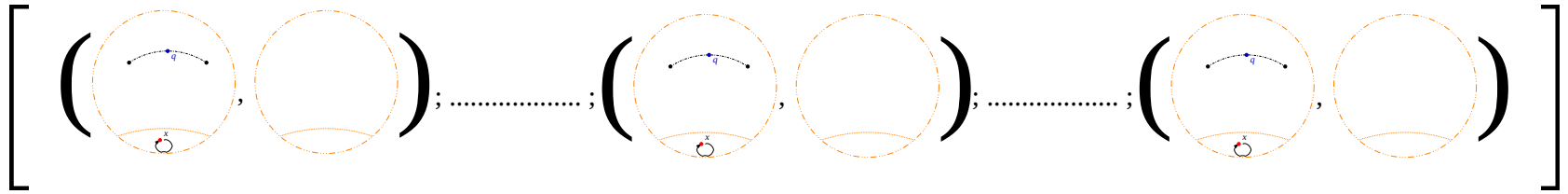
...informally, when H_{n+1} is isomorph to H_n



Termination

...informally, when H_{n+1} is isomorph to H_n

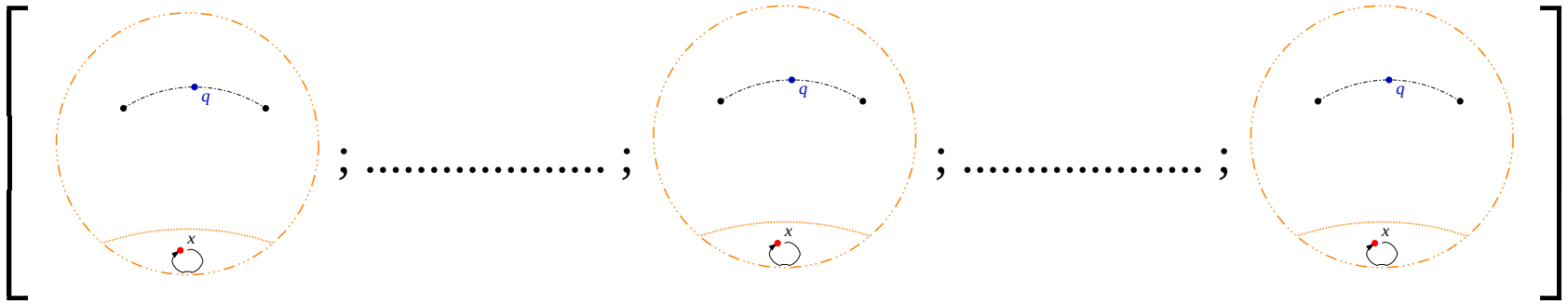




$\wedge$

Termination

...informally, when H_{n+1} is isomorph to H_n



$\wedge$

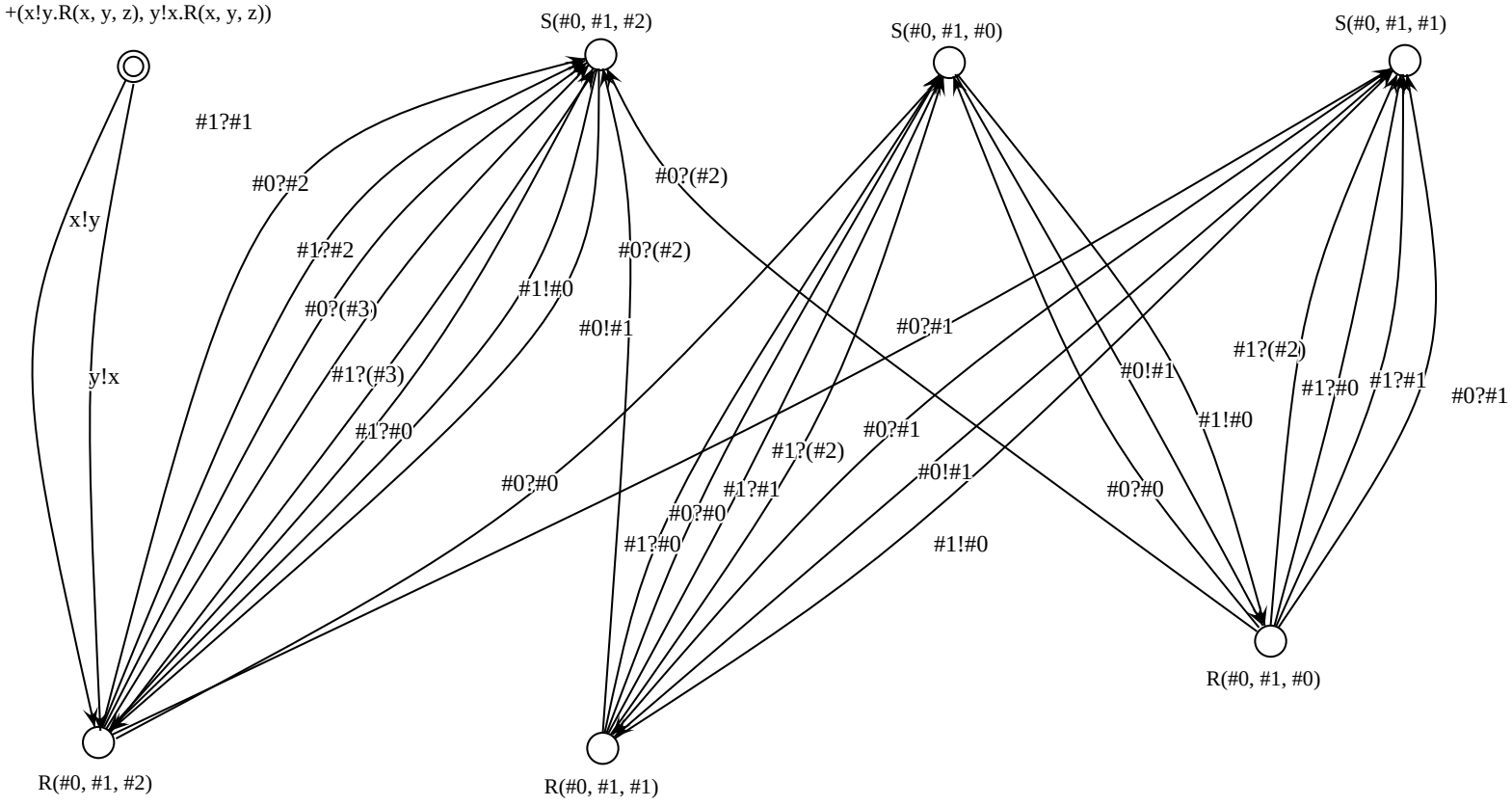
no further names are added

An example

$$S(x,y,z) = x!y.R(x,y,z) + y!x.R(x,y,z)$$
$$R(x,y,z) = x?(w).S(x,y,w) + y?(w).S(y,x,z)$$

An example

$$S(x,y,z) = x!y.R(x,y,z) + y!x.R(x,y,z)$$
$$R(x,y,z) = x?(w).S(x,y,w) + y?(w).S(y,x,z)$$



Minimal representation

state	b0		2	[[1 ; 2] ; [2 ; 1]]
state	b1		2	[[1 ; 2] ; [2 ; 1]]
b0	→	b1	out[1 ; 2]	[1 ; 2]
b0	→	b1	out[1 ; 2]	[2 ; 1]
b0	→	b1	out[2 ; 1]	[1 ; 2]
b0	→	b1	out[2 ; 1]	[2 ; 1]
b1	→	b0	bin[1]	[1 ; 2]
b1	→	b0	bin[1]	[2 ; 1]
b1	→	b0	bin[2]	[2 ; 1]
b1	→	b0	bin[2]	[1 ; 2]
b1	→	b0	in[1 ; 1]	[1 ; 2]
b1	→	b0	in[1 ; 1]	[2 ; 1]
b1	→	b0	in[1 ; 2]	[1 ; 2]
b1	→	b0	in[1 ; 2]	[2 ; 1]
b1	→	b0	in[2 ; 1]	[1 ; 2]
b1	→	b0	in[2 ; 1]	[2 ; 1]
b1	→	b0	in[2 ; 2]	[1 ; 2]
b1	→	b0	in[2 ; 2]	[2 ; 1]

Final remarks

- Handhover benchmarks are encouraging
- Extend to
  - open bisimilarity
  - asynchronous π-calculus
  - complex terms: application to verification of security protocols
- More experimental results
- Optimization: handling of permutations
- Integrations (HAL and Mobility workbench)
- Ocaml compiler
- Tool re-engineering (on-going work)