Distributed Delta

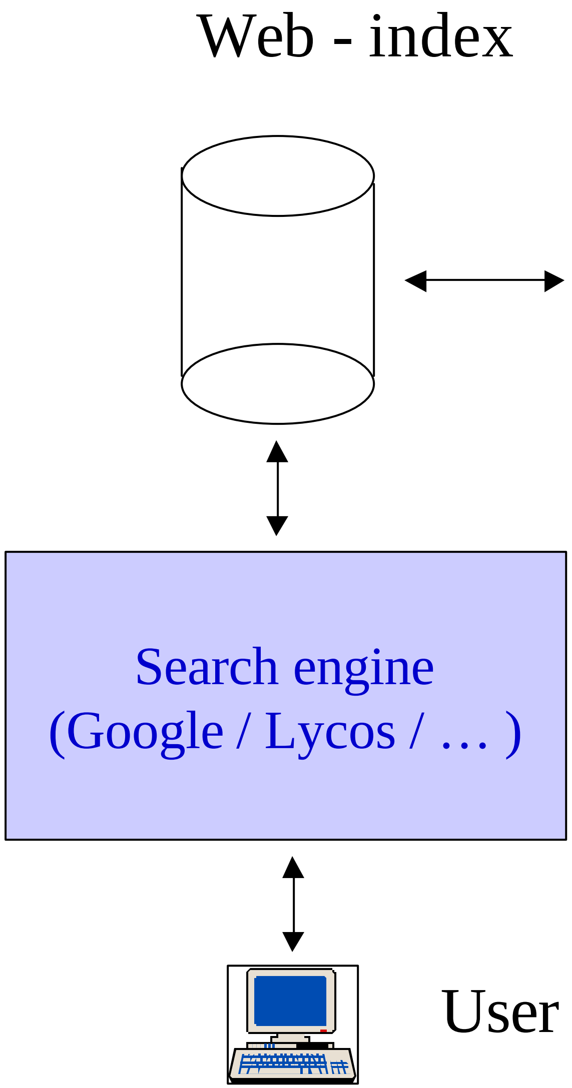
Distributed querying of Web by using dynamically created mobile agents

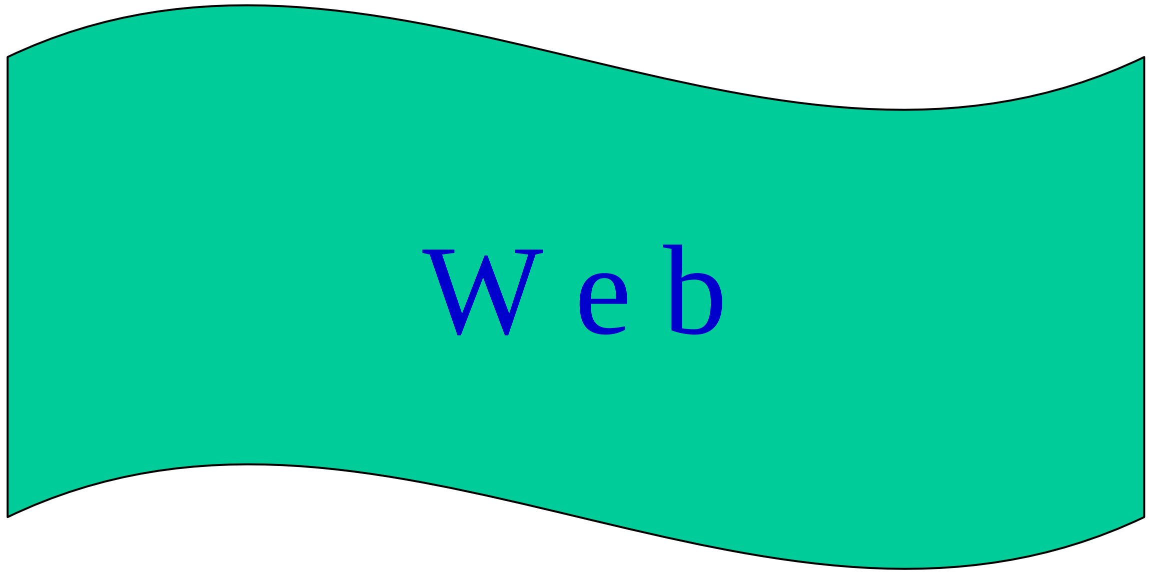
Vadim Guzev, University of Pereslavl-Zalessky
Vladimir Sazonov, University of Liverpool
Yuri Serdyuk, Program Systems Institute of the RAS, Pereslavl-Zalessky

Content

1. Introduction : the Web querying by using mobile agents
2. A set-theoretic language Delta for Web querying
3. The distributed calculi of mobile agents
4. Distributed mobile algorithms for evaluation of Delta-constructs
5. Demonstration : the developed application based on the Grasshopper agent platform

Traditional Web querying



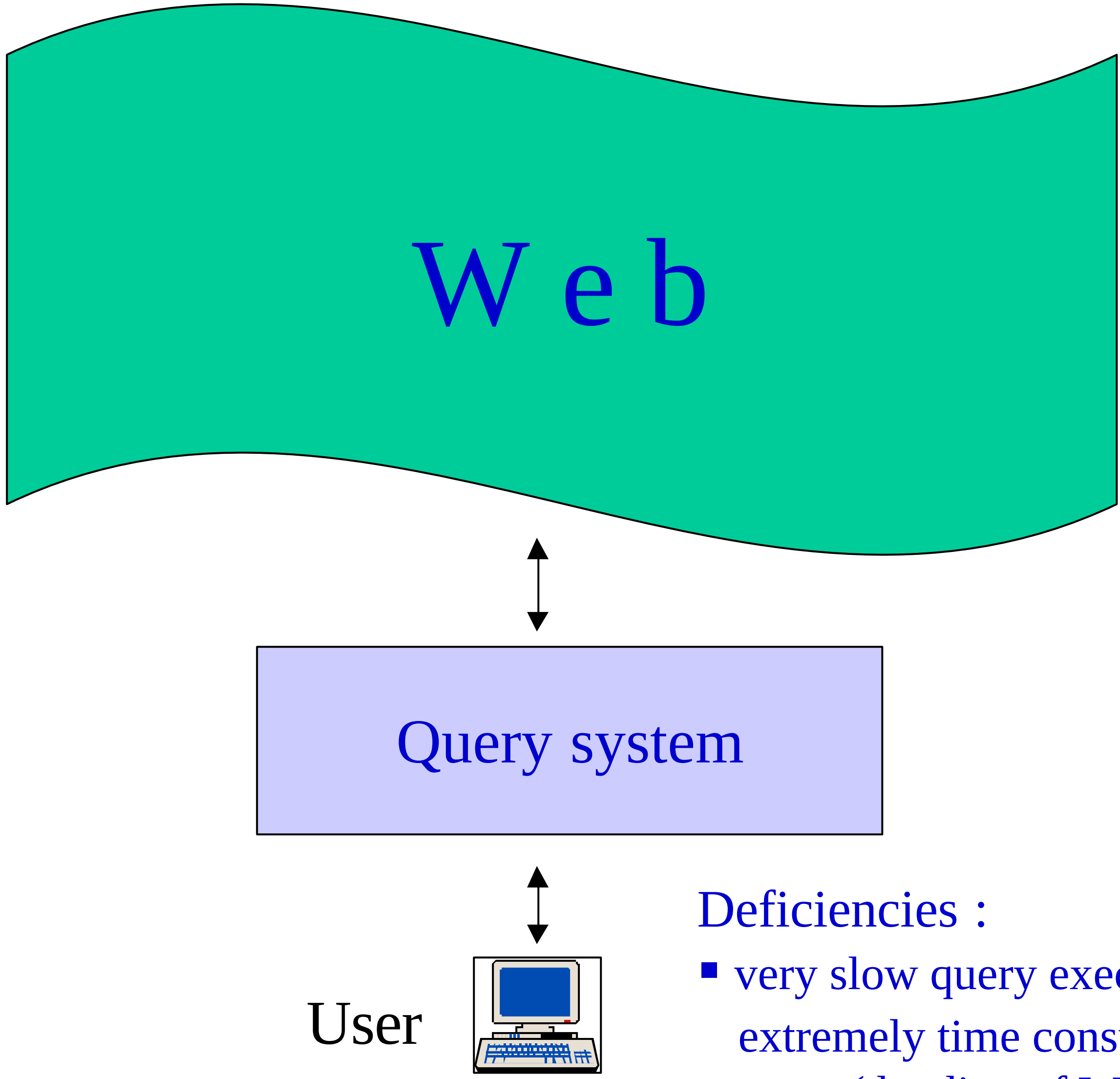


Deficiencies :

- obsolete information in the Web-index;
- impossibility to pose complex (structural) queries;
- impossibility to create automatically (by query execution) new Web-pages from existing ones

. . .

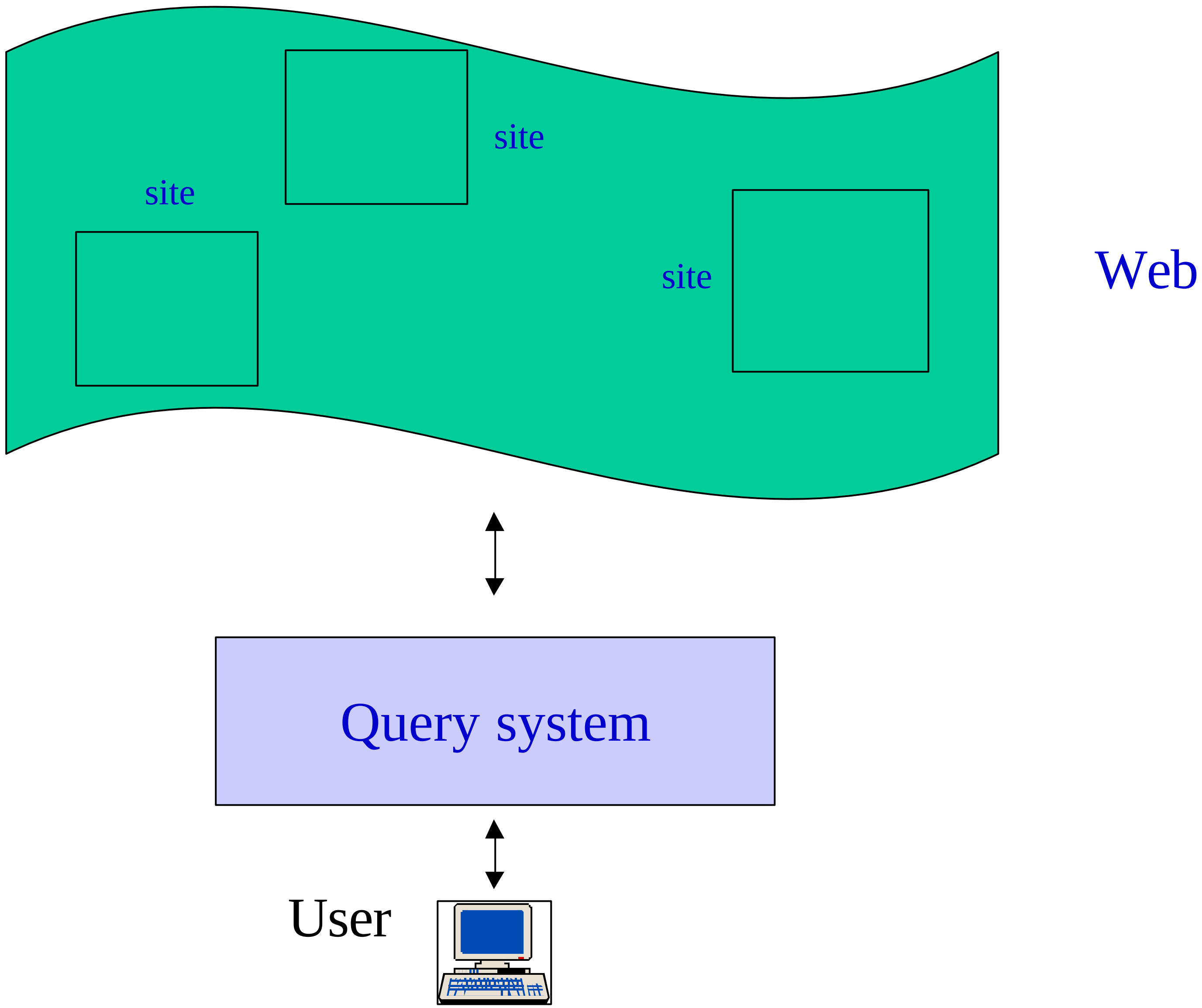
Direct Web querying



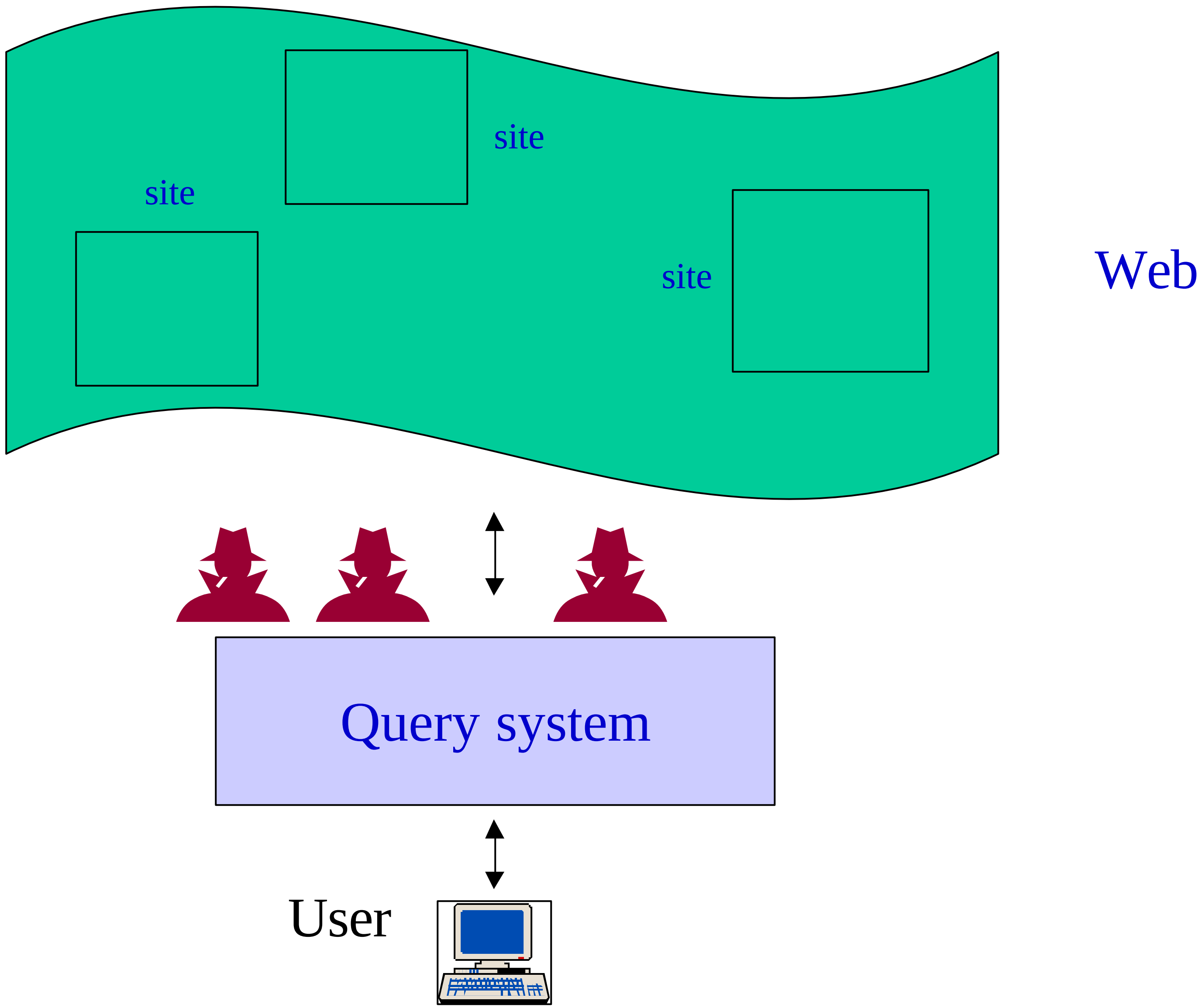
Deficiencies :

- very slow query execution due to extremely time consuming browsing steps (loading of Web pages)

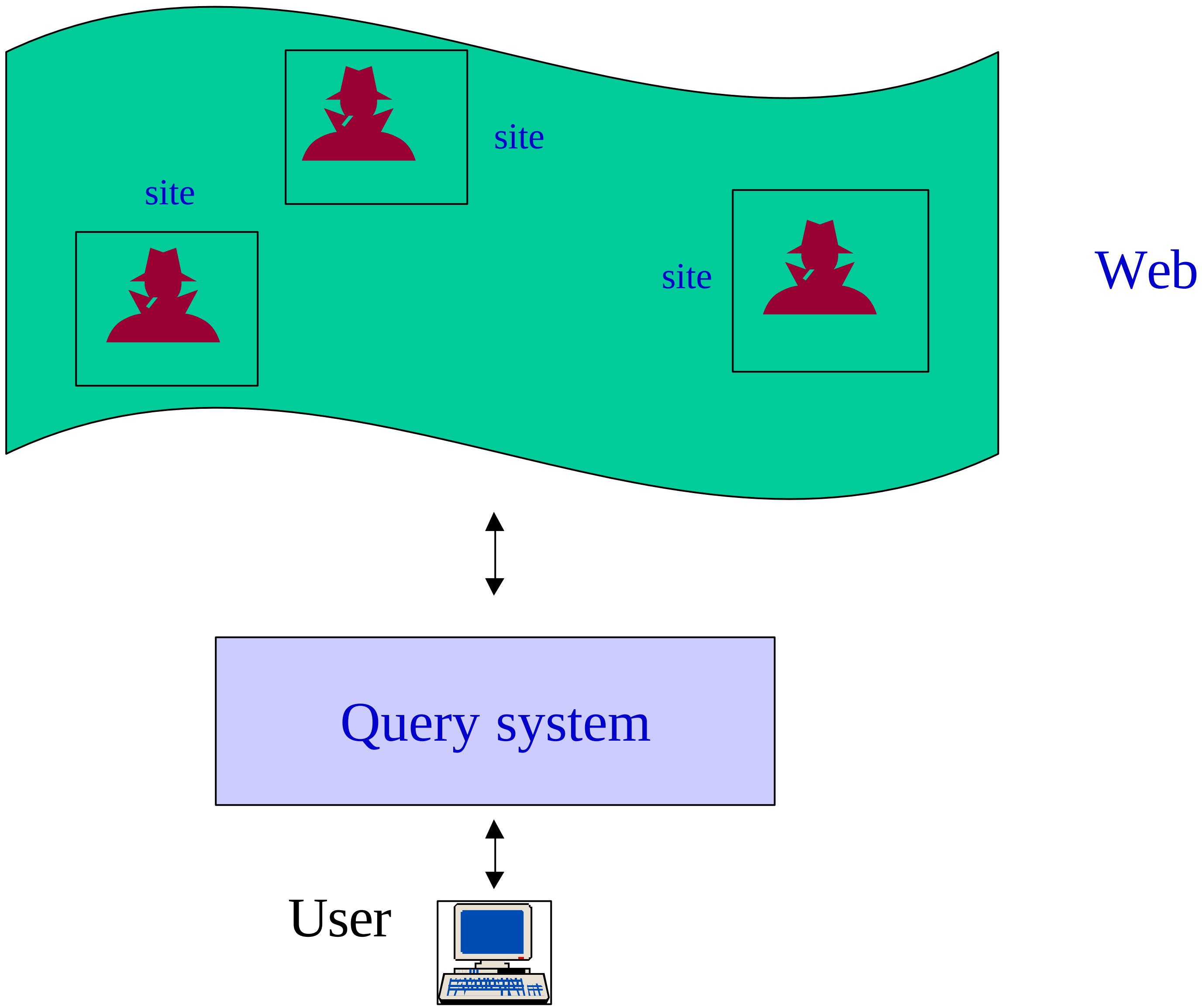
Web querying by using mobile agents



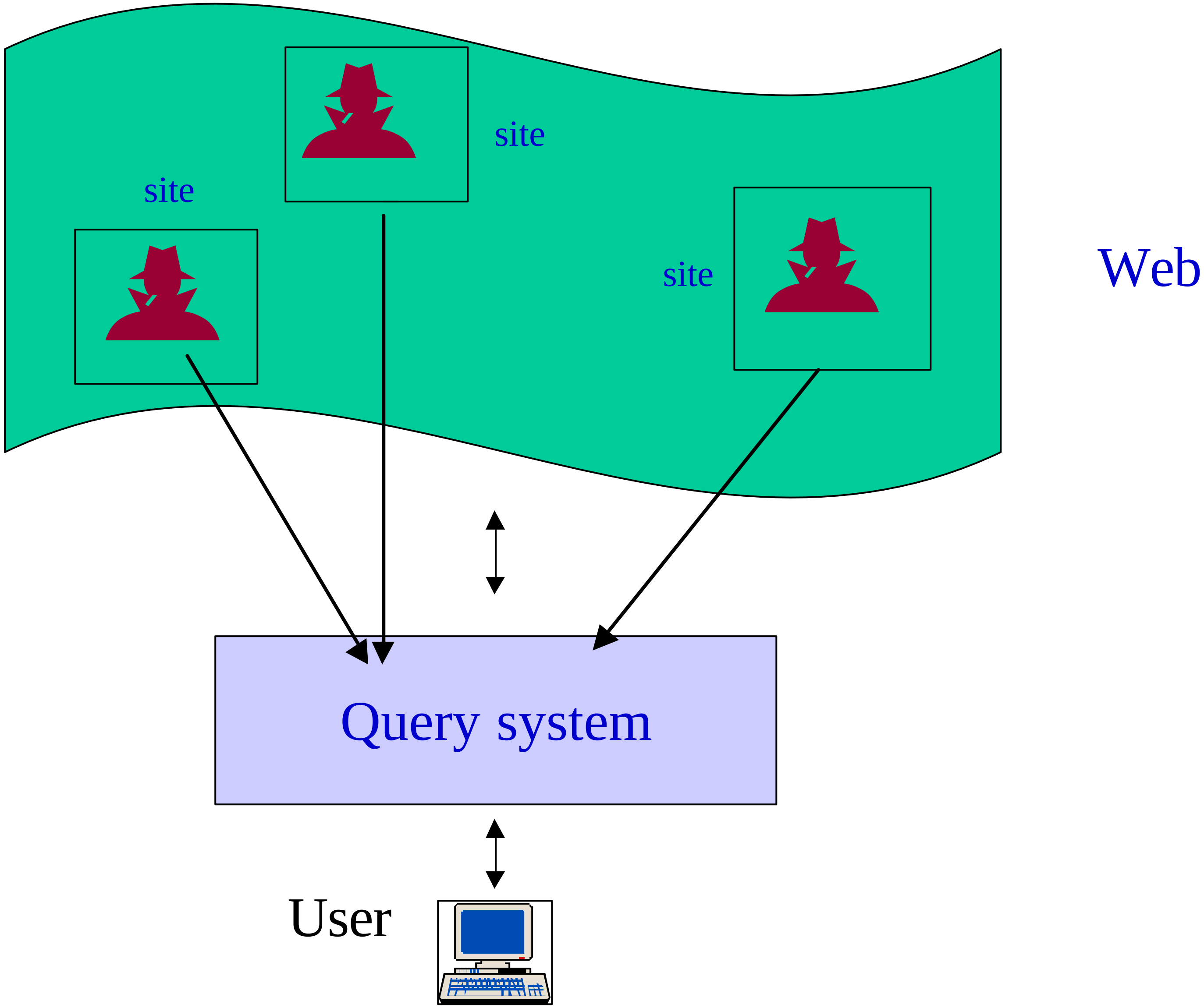
Web querying by using mobile agents



Web querying by using mobile agents

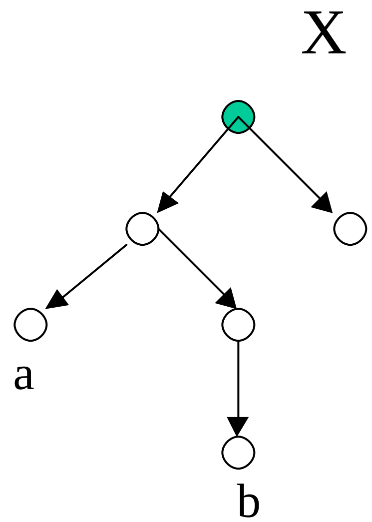


Web querying by using mobile agents

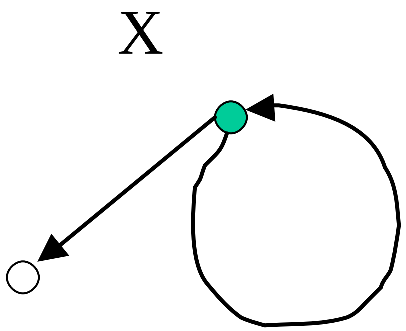


Set-theoretical language Delta for Web querying

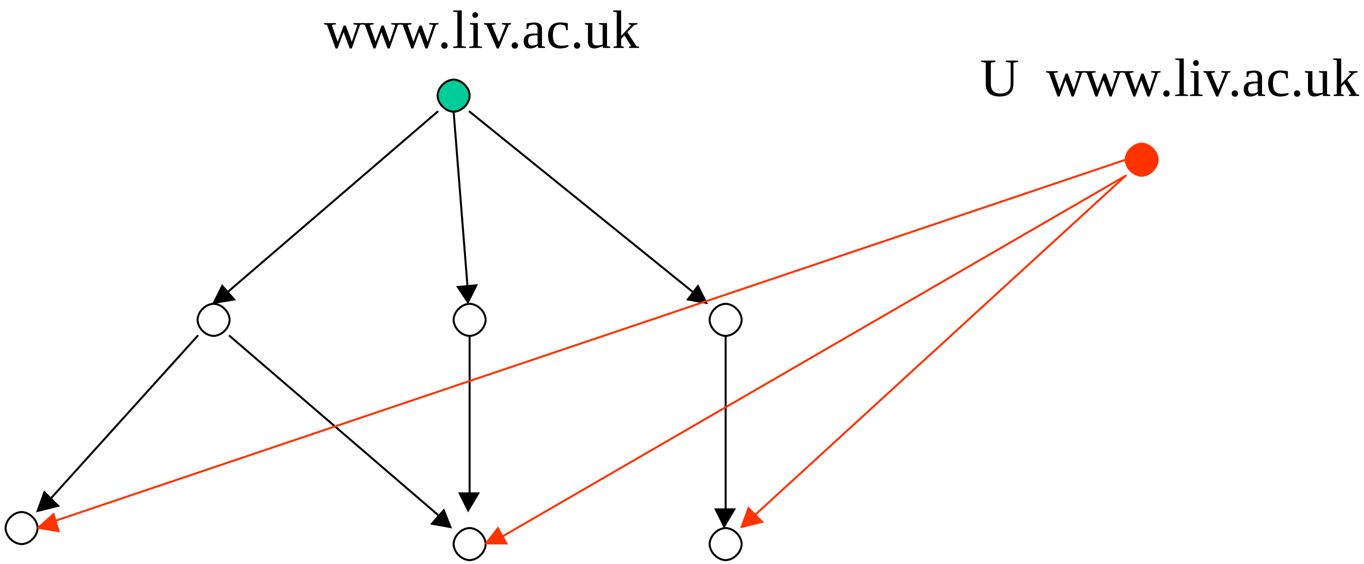
X = { { a, { b } }, {} }



X = { a, X }

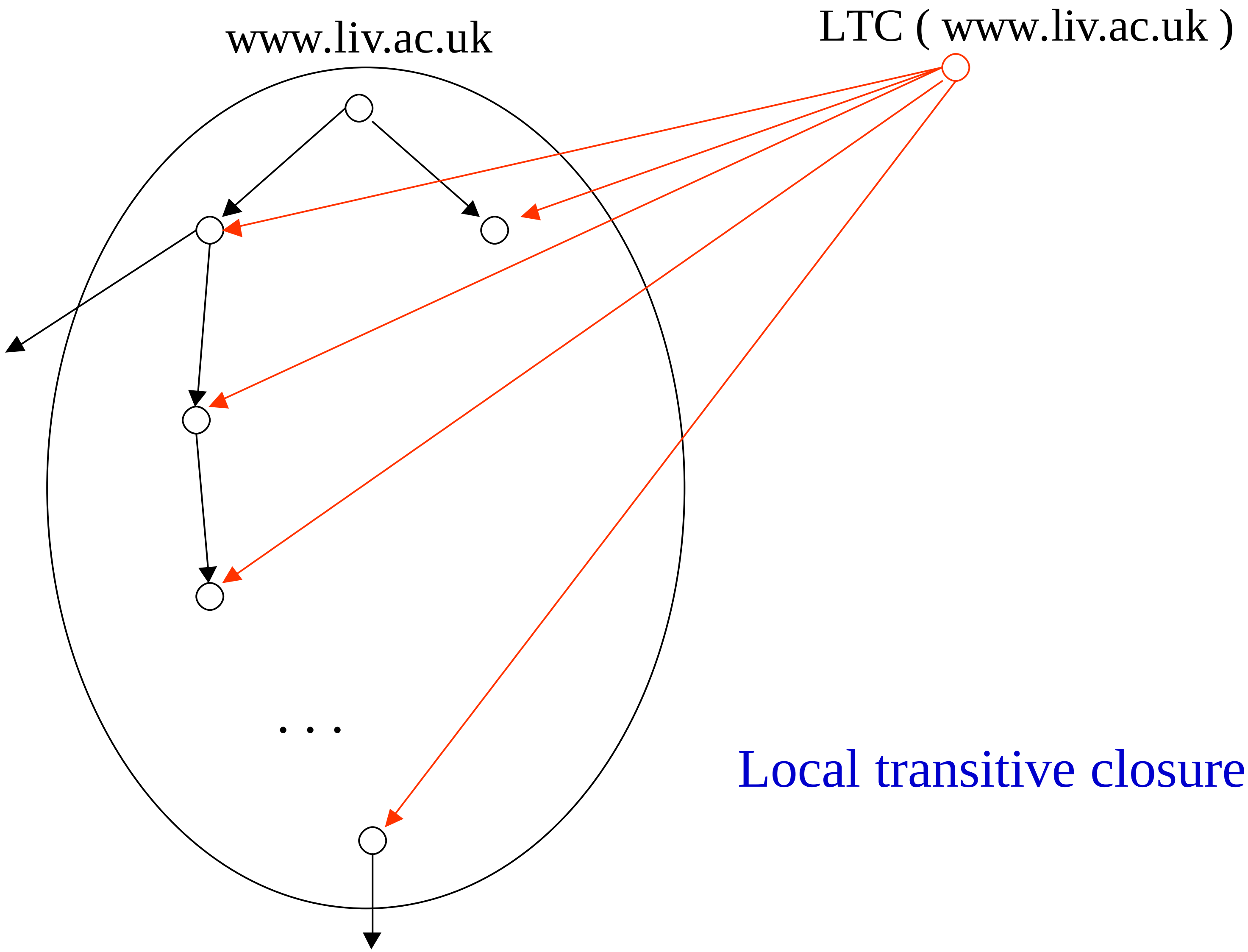


Set-theoretical language Delta for Web querying



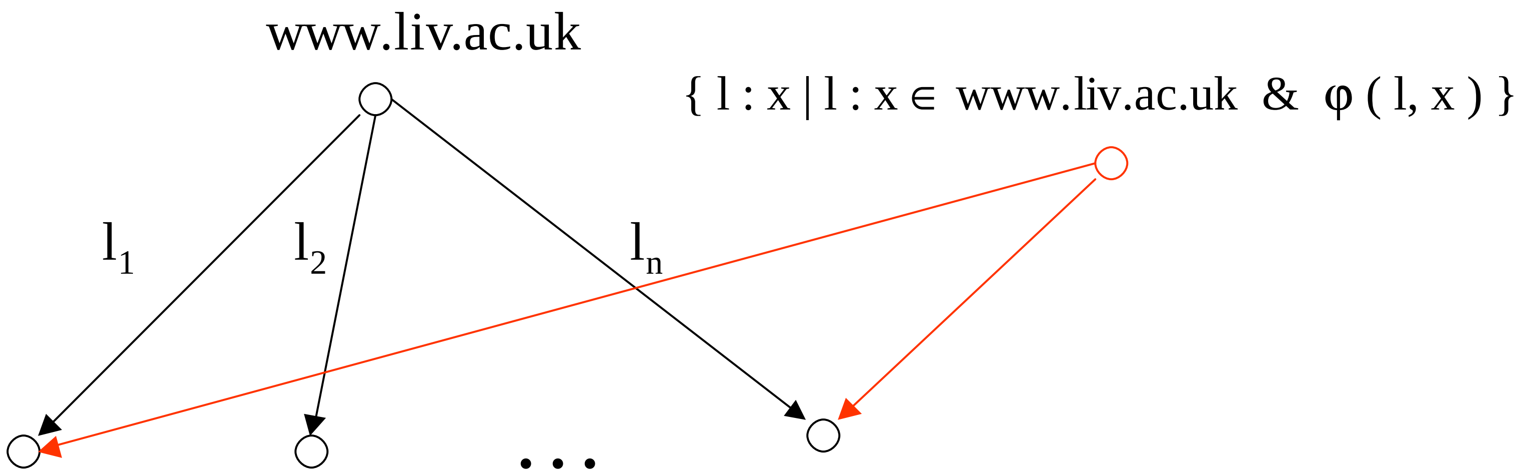
Union

Set-theoretical language Delta for Web querying



Local transitive closure

Set-theoretical language Delta for Web querying



Selection

Set-theoretical language Delta for Web querying

Syntax :

Δ-terms ::= $c \mid x \mid \varnothing \mid \bigcup t \mid LTC(t) \mid TC(t) \mid \{ l_1 : t_1 , \ldots , l_n : t_n \} \mid$

$\{ l : t(l, x) \mid l : x \in s \;\&\; \varphi(l, x) \} \mid$

$fix\; q.(q = q \cup \{ l : x \in s \;\&\; \varphi(q, l, x) \} \mid$

$D(s, t)$

Δ-formulas ::= $l = m \mid s = t \mid l : s \in t \mid P(t_1 , \ldots , t_n) \mid$

$\varphi \;\&\; \psi \mid \varphi \vee \psi \mid \neg \varphi \mid$

$\forall l : x \in s . \varphi \mid \exists l : x \in s . \varphi$

Examples of queries.

1. Find on the Web-site of Mathematics department of Indian University all pages with labels which occur words “Seminar” or “Colloquium”.

$$\{\, l : x \mid l : x \in \mathrm{LTC}(\text{“www.math.indiana.edu”}) \;\&\; (\; \mathrm{Occur}(\text{“Seminar”}, l) \vee \mathrm{Occur}(\text{“Colloquium”}, l) \;) \,\}$$

Examples of queries.

2. Find all external pages for a given site
(i.e., find all pages to which there is a reference from some page on given site and which doesn't belong to this site).

$$\bigcup \{ \, l : \{ \, m : y \mid m : y \in x \;\&\; \neg \, \mathrm{Local}\,(x, y) \, \} \mid$$

$$l : x \in \mathrm{LTC}\,(\text{“www.math.indiana.edu”})$$

$$\}$$

Examples of queries.

3. On given site find all pages from which the main (home) page of this site is reachable (with some lightened syntax without labels).

$$\text{fix } q \,.\, (q = q \cup \{ \; x \mid x \in \; \text{LTC}(\text{“www.math.indiana.edu”}) \;\; \& $$

$$(\;\; (\;\; \text{“www.math.indiana.edu”} \in x) \;\; \vee$$

$$(\;\; \exists z \in q \,.\, z \in x)$$

$$)$$

$$)$$

Examples of queries.

4. Copy remote site to local computer
   (more precisely, copy it’s topological (hyperlink) structure).

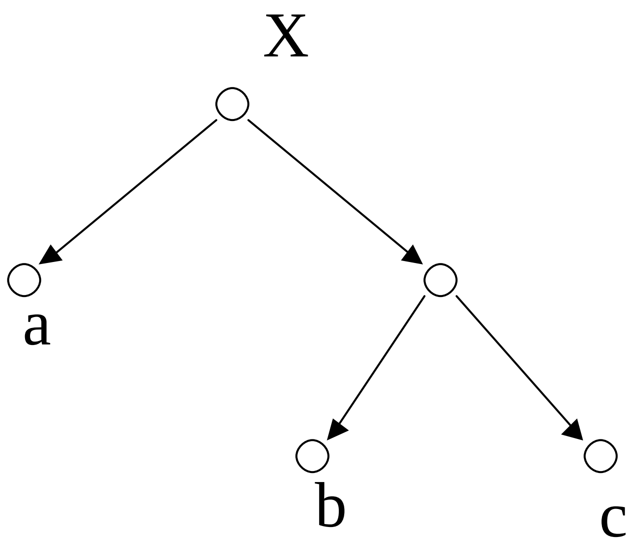
Decoration

$X = \{ a, \{ b, c \} \}$

$X_h = \{ \{ <x,a> , <x,y>, <y,b>, <y,c> \} , x \}$

where $<a,b> = \{ \{a\}, \{a, b\} \}$

$D (X_h , x) = X$



Examples of queries.

4. Copy remote site to local computer
(more precisely, copy remote site's topological (hyperlink) structure).

$$V = \{ l : x \mid l : x \in LTC(\{ \text{“www.math.indiana.edu”} \}) \}$$

$$E = \bigcup \{ l : \{ m : \langle x, y \rangle \mid m : y \in x \;\&\; Local(x, y) \} \mid l : x \in V \}$$

$$D(E, \text{“www.math.indiana.edu”})$$

π - calculus (Robin Milner, 1991)

N – (an infinite) set of names

P , Q := 0	nil
P | Q	parallel composition
c ! v	output
c ? x. P	input
new c in P	new channel name creation

π - calculus : examples of reductions

1. x ! a | x ? u. y ! u

π - calculus : examples of reductions

1. **x ! a | x ? u. y ! u** $\longrightarrow$ **y ! a**

π - calculus : examples of reductions

1. **x ! a | x ? u. y ! u** $\longrightarrow$ **y ! a**

2. **x ! a | new x in (x ! b | x ? u. y ! u)**

π - calculus : examples of reductions

1. **x ! a | x ? u. y ! u** $\longrightarrow$ **y ! a**

2. **x ! a | new x in (x ! b | x ? u. y ! u)** $\longrightarrow$ **x ! a | new x in y ! b**

π - calculus : examples of reductions

1. x ! a | x ? u. y ! u $\longrightarrow$ **y ! a**

2. x ! a | new x in (x ! b | x ? u. y ! u) $\longrightarrow$ **x ! a | new x in y ! b**

new x in P [x] o **new x' in P [x / x']**

! P ≡ P | ! P

π - calculus : structural congruence

P | 0 º P

P | Q º Q | P

P | (Q | R) º (P | Q) | R

new x in new y in P º new y in new x in P

P | new x in Q º new x in (P | Q) , if x Ï fn (P)

π - calculus : reduction semantics

Com

$$\frac{}{c\ !\ v \mid c\ ?\ w.\ P \longrightarrow P\ \{\ w/v\ \}}$$

Par

$$\frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}$$

Res

$$\frac{P \longrightarrow P'}{\textbf{new}\ x\ \textbf{in}\ P \longrightarrow \textbf{new}\ x\ \textbf{in}\ P'}$$

Struct

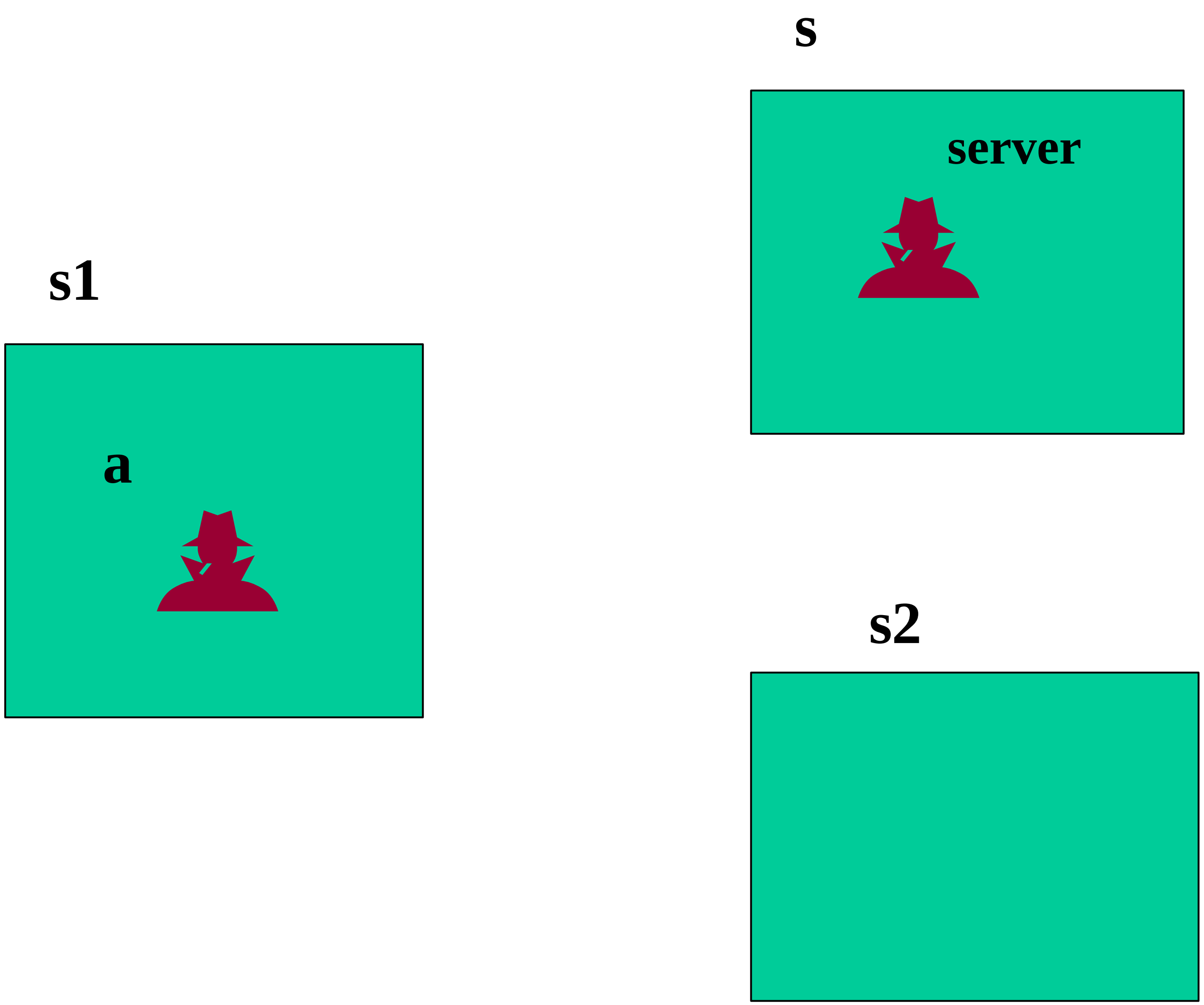
$$\frac{P\ {}^{\circ}\ Q\ ,\ Q \longrightarrow R\ ,\ R\ {}^{\circ}\ T}{P \longrightarrow T}$$

Distributed π - calculus
(Peter Seweel et al., 1998)

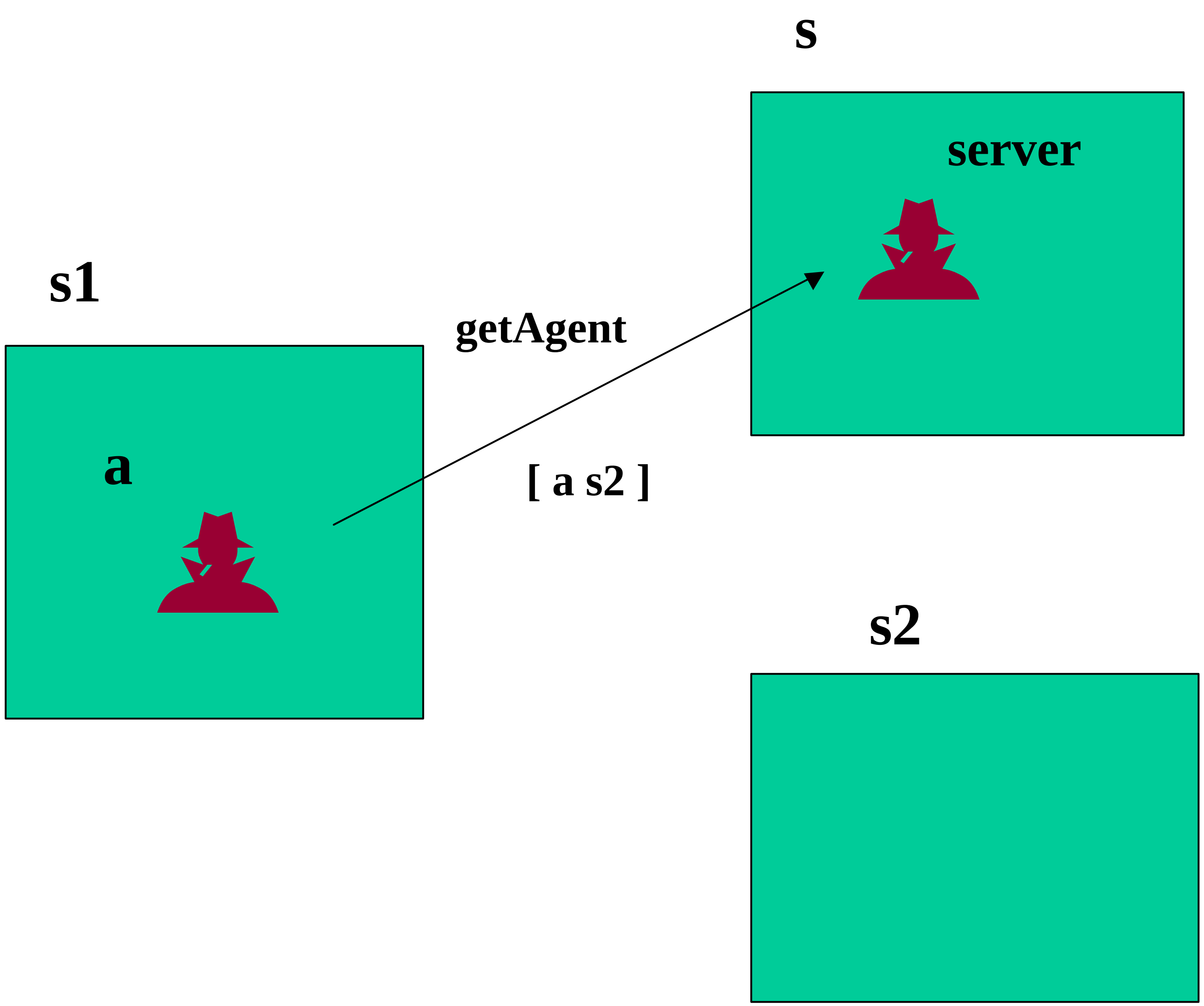
Syntax : *N* - (an infinite) set of names
a , b , … - agents c - channel s - site

P , Q ::=	0	nil
	P | Q	parallel composition
	new c in P	new channel name creation
	c ! v	output v on channel c in the current agent
	c ? p → P	input from channel c
	* c ? p → P	replicated input from channel c
	agent a = P in Q	agent creation
	migrate to s → P	agent migration
	<a@s> c ! v	output to agent a on site s
	if u = v then P else Q	value equality testing

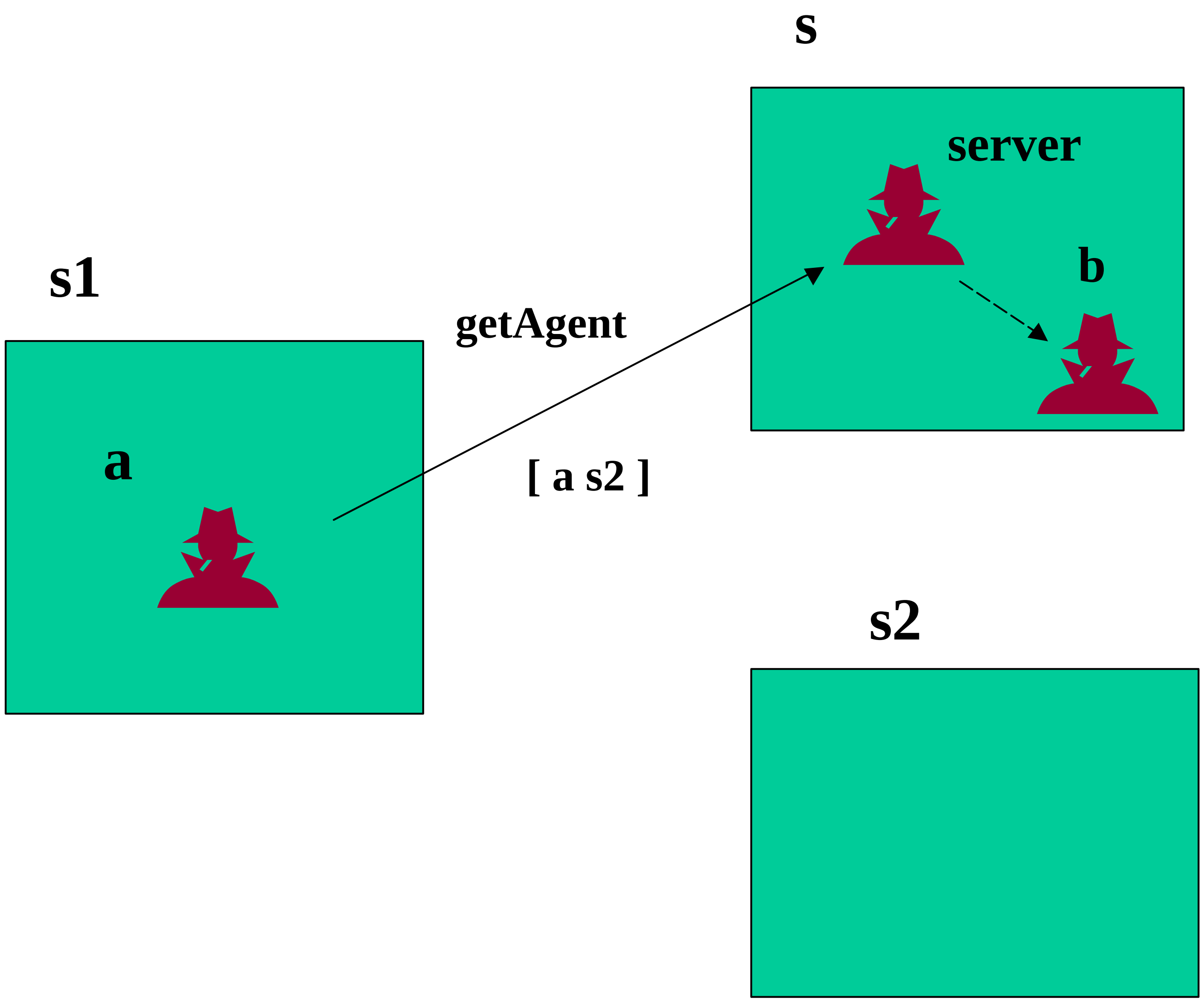
Distributed π - calculus : example



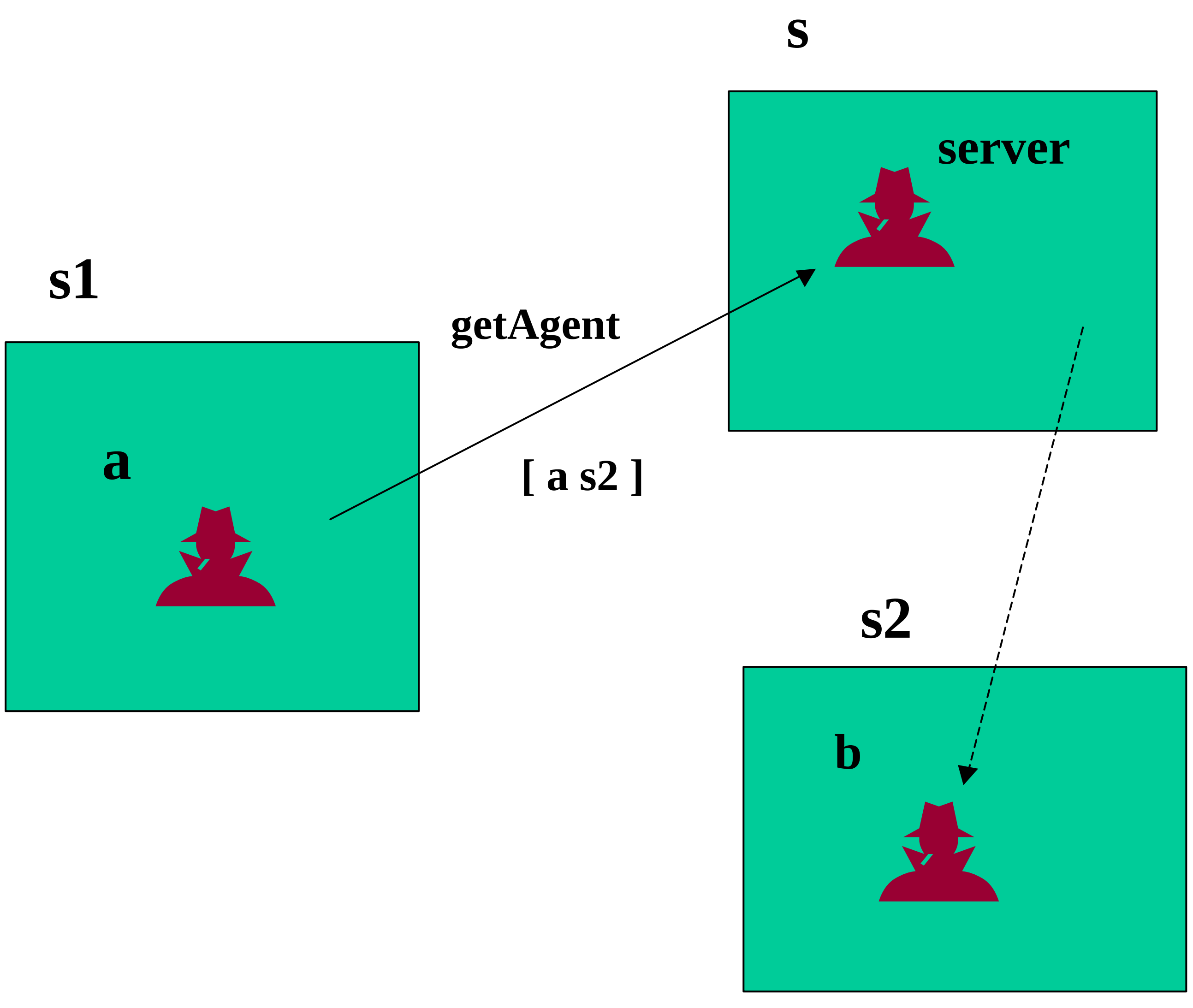
Distributed π - calculus : example



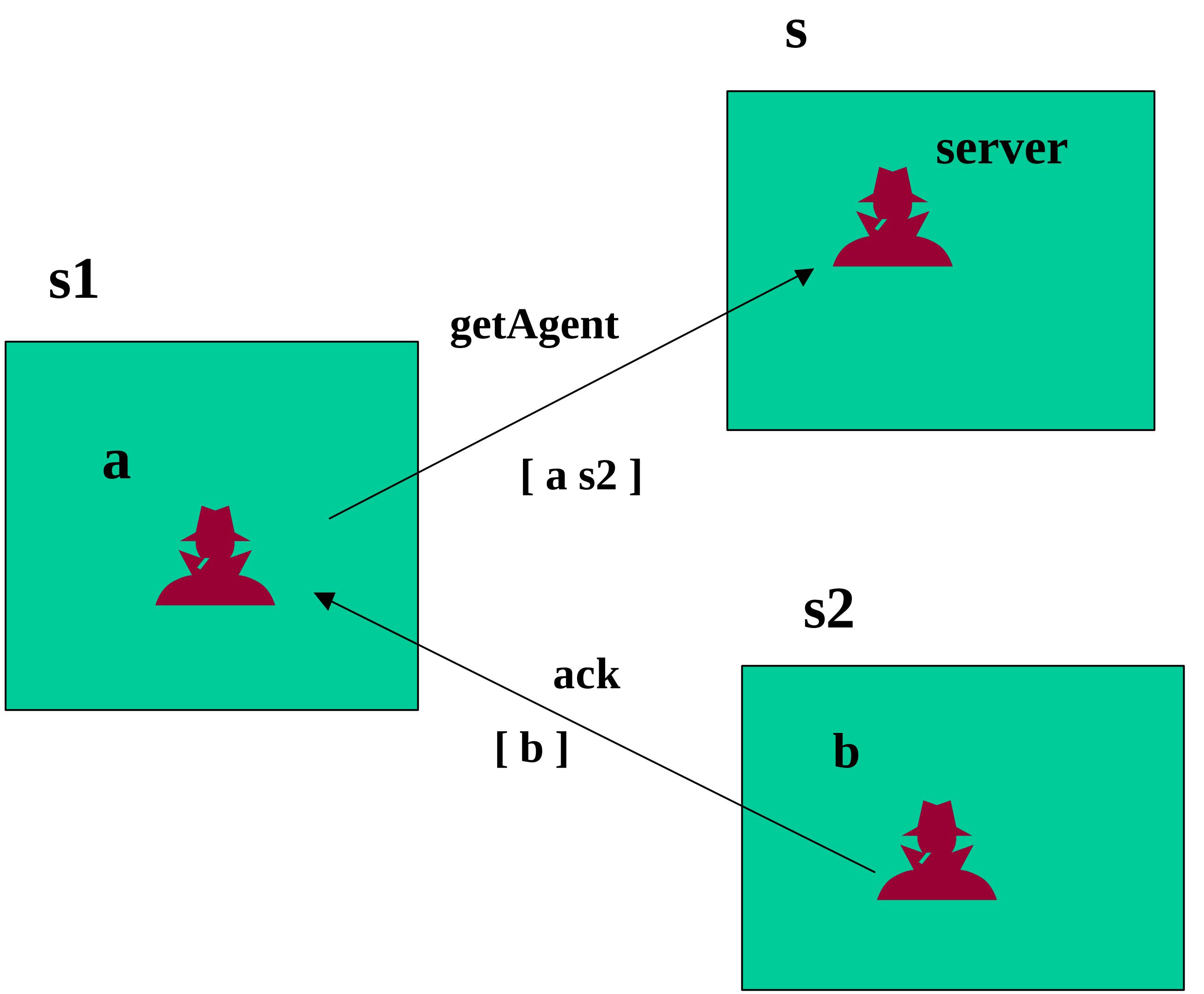
Distributed π - calculus : example



Distributed π - calculus : example



Distributed π - calculus : example



Distributed π - calculus : example

*** getAgent ? (a s2) →**

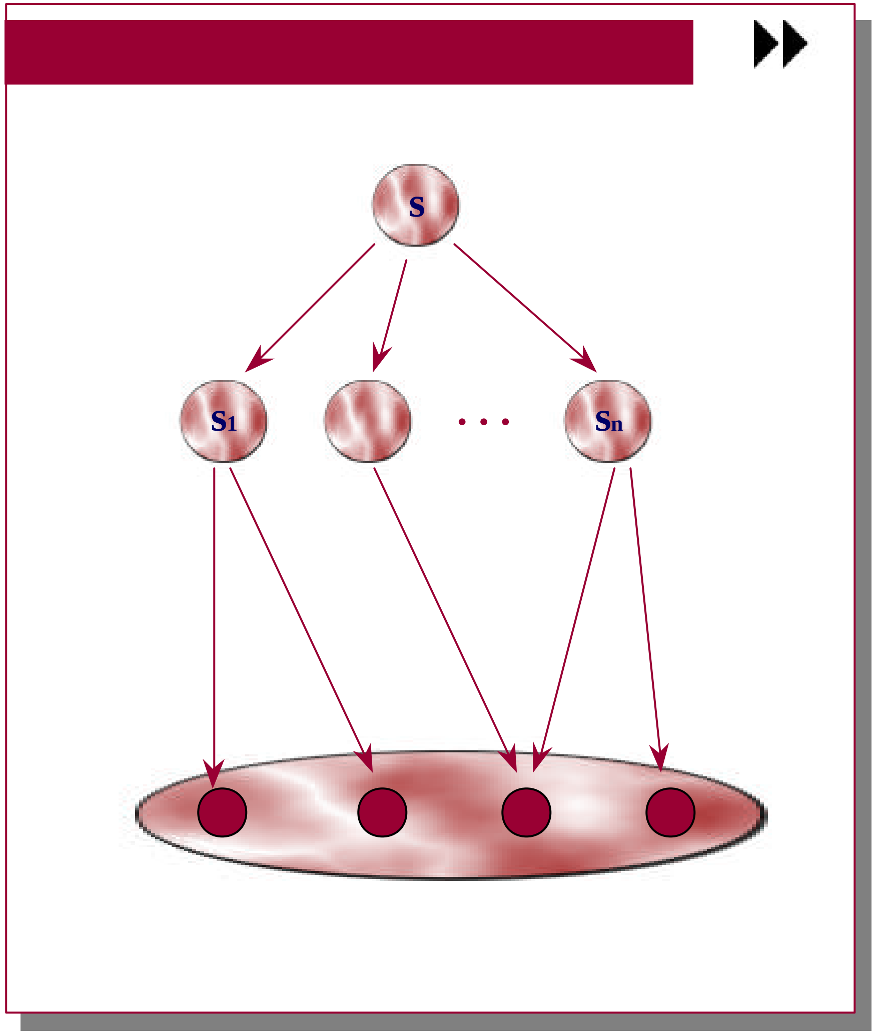
agent b =

migrate to s2 → (<a @ s1> ack ! b | B)

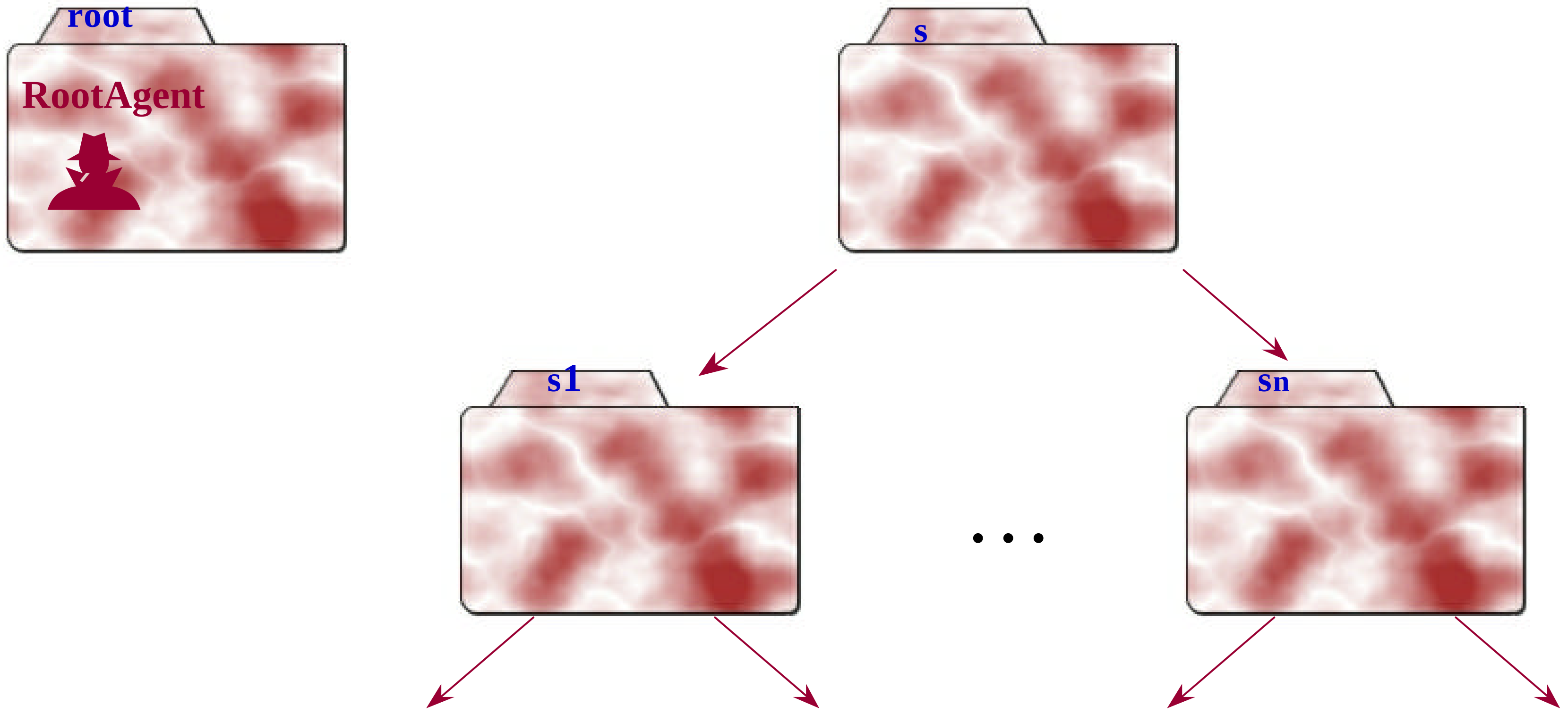
in

0

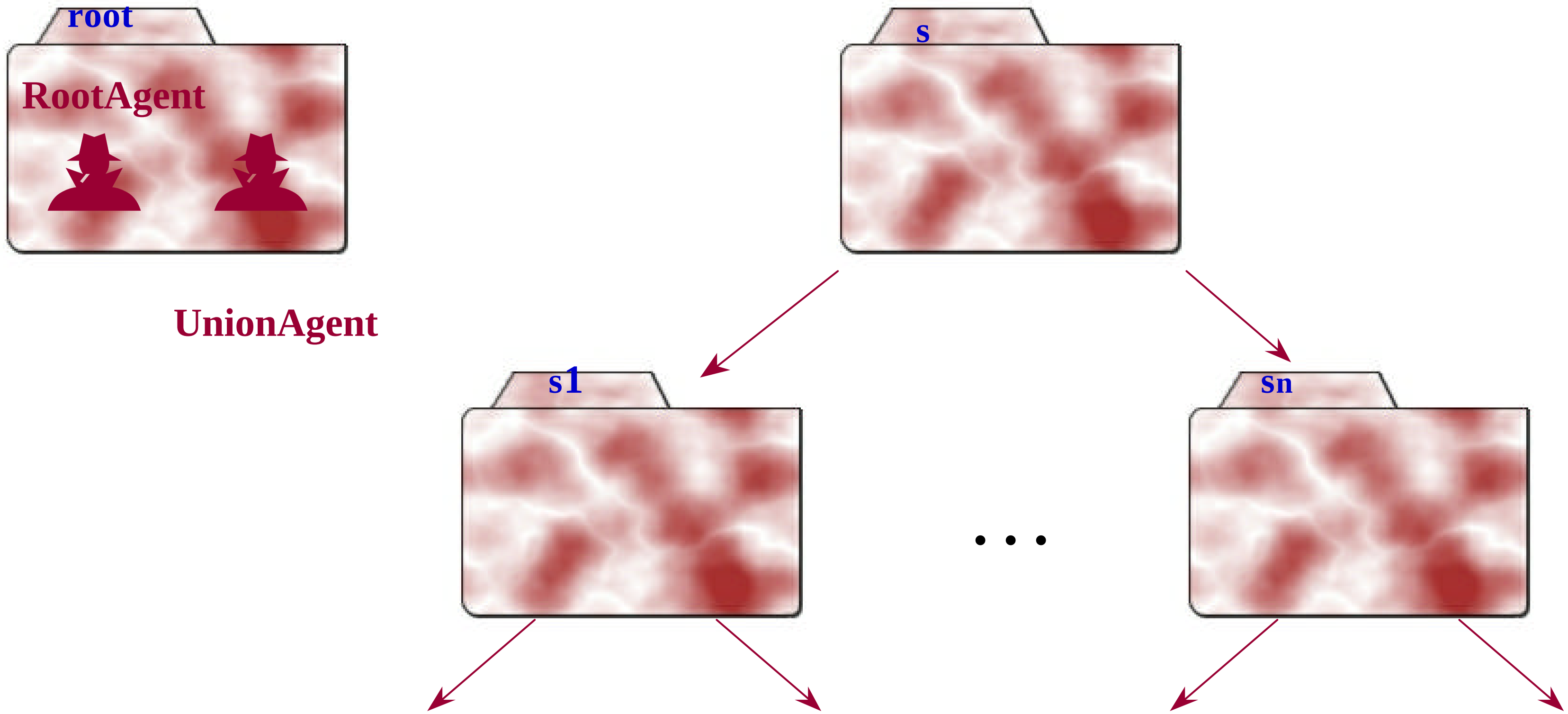
Mobile algorithm for Union-construct



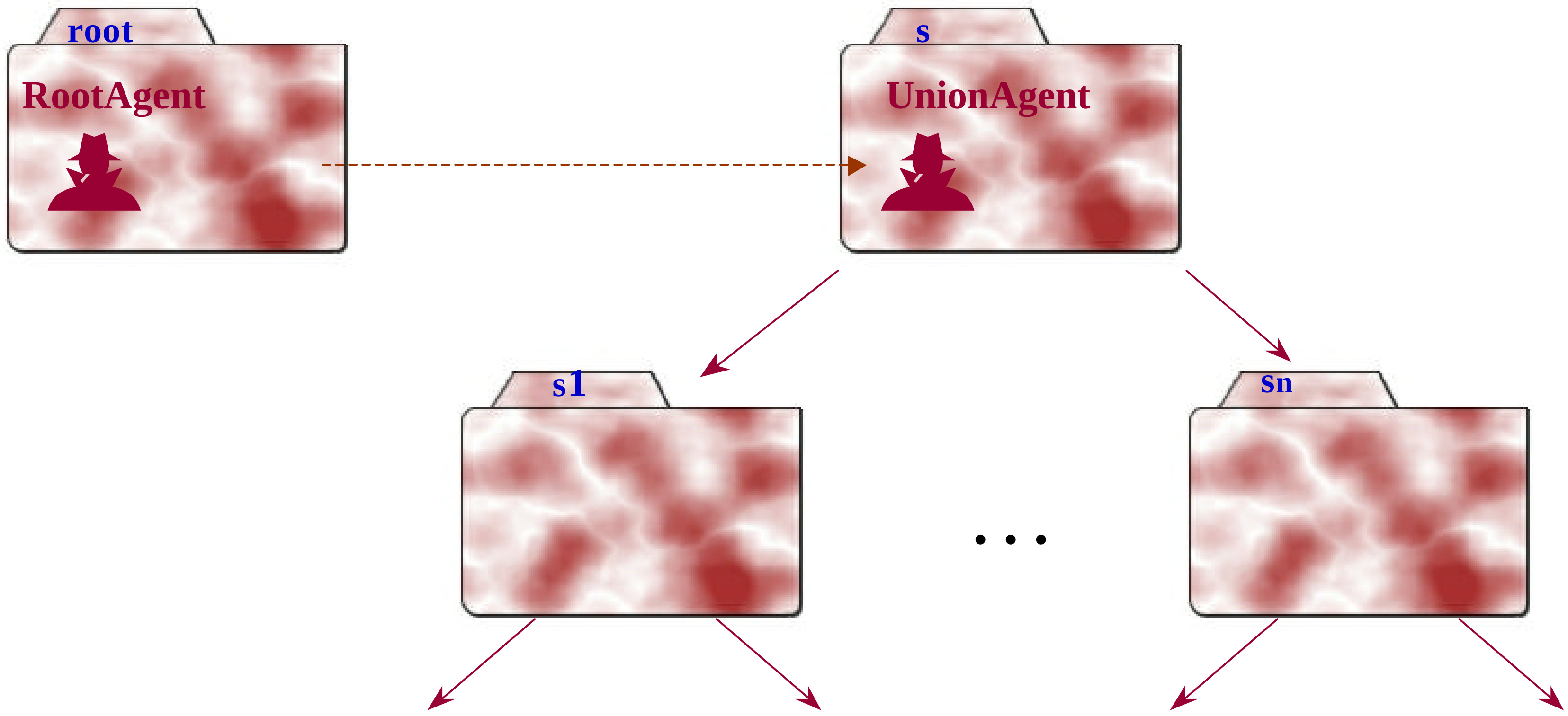
Mobile algorithm for Union-construct



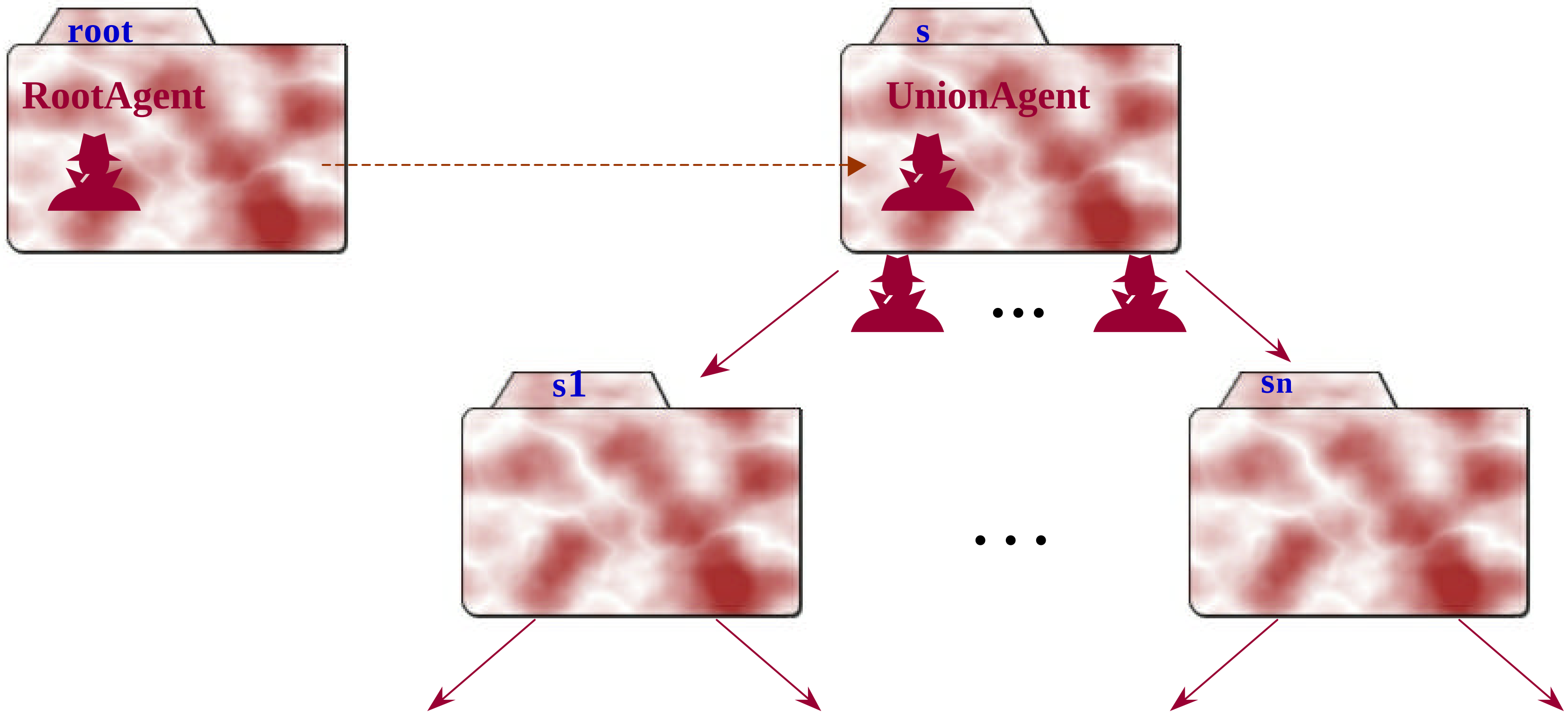
Mobile algorithm for Union-construct



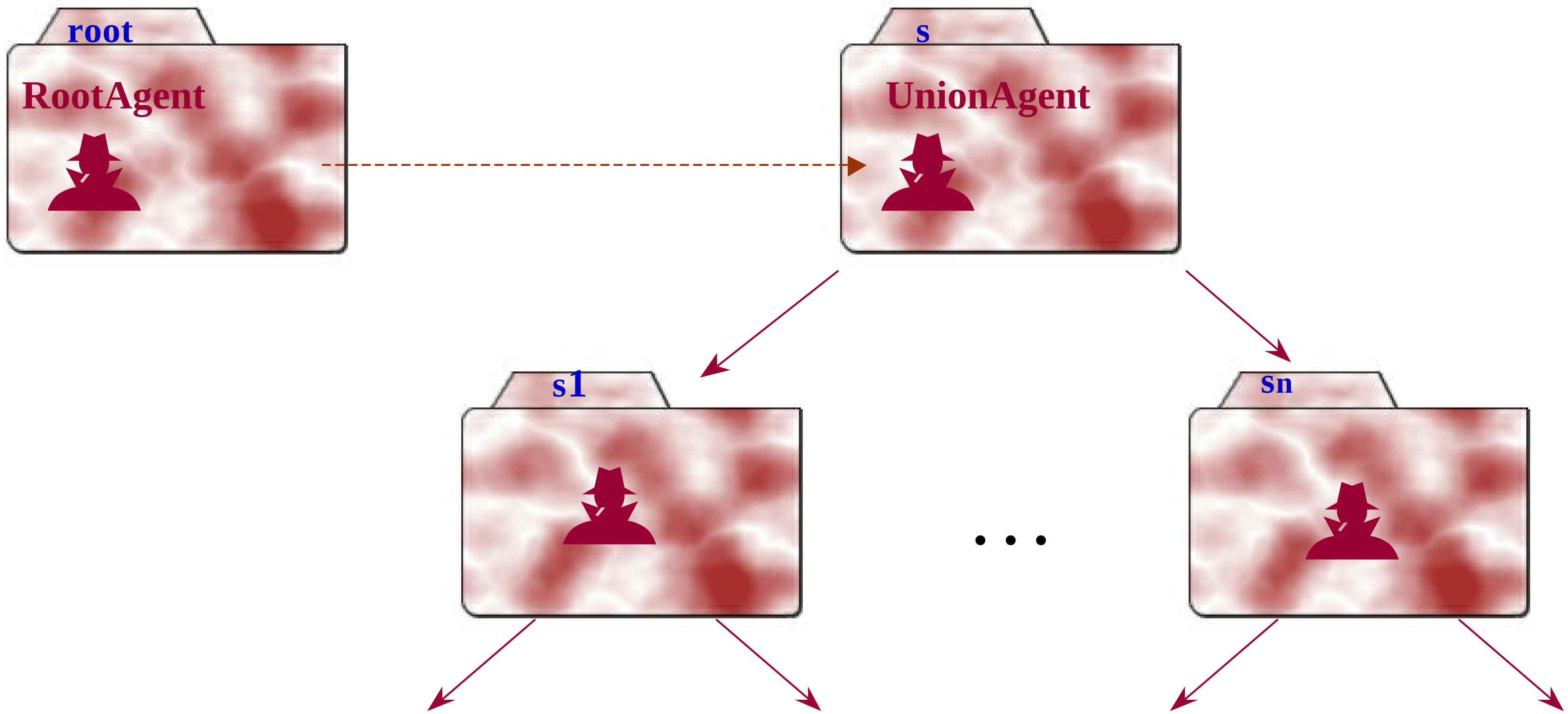
Mobile algorithm for Union-construct



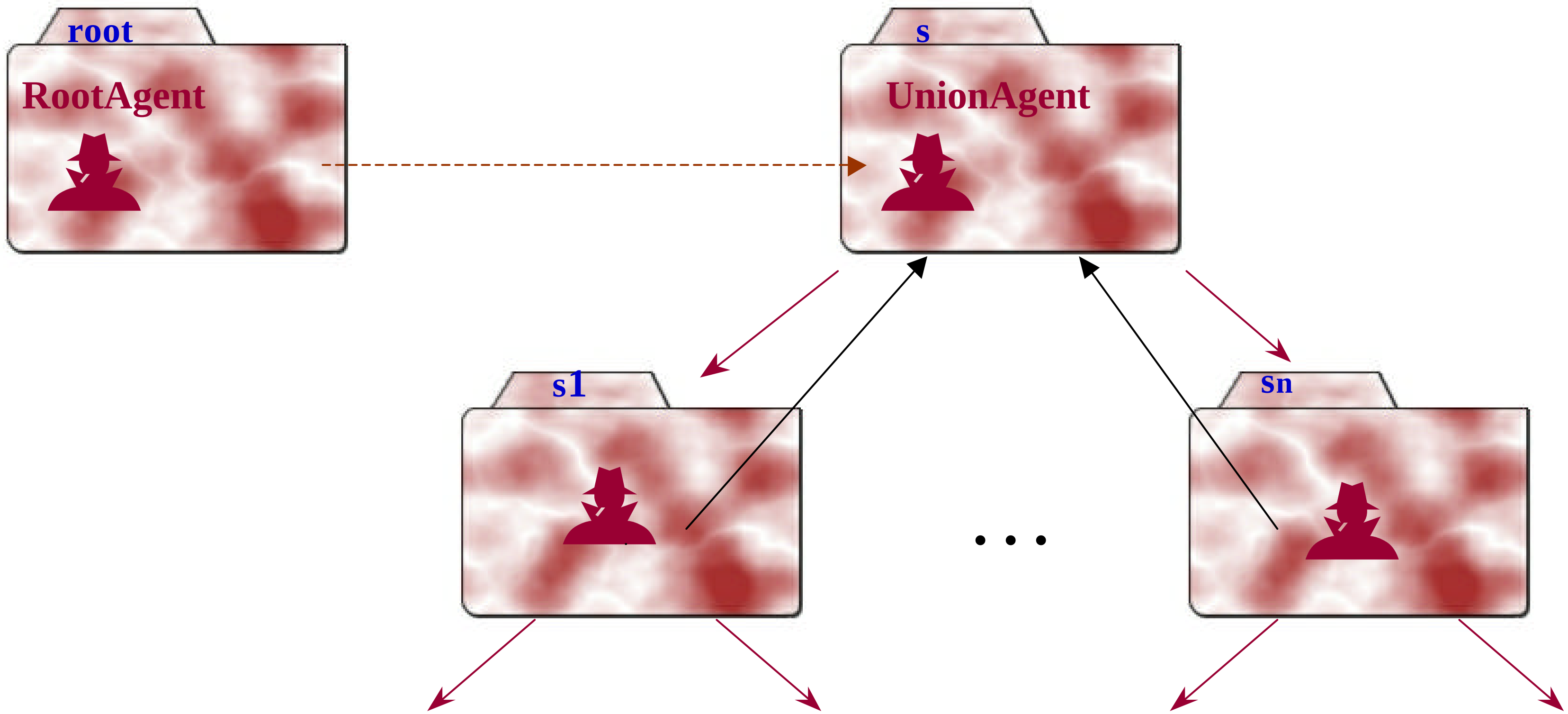
Mobile algorithm for Union-construct



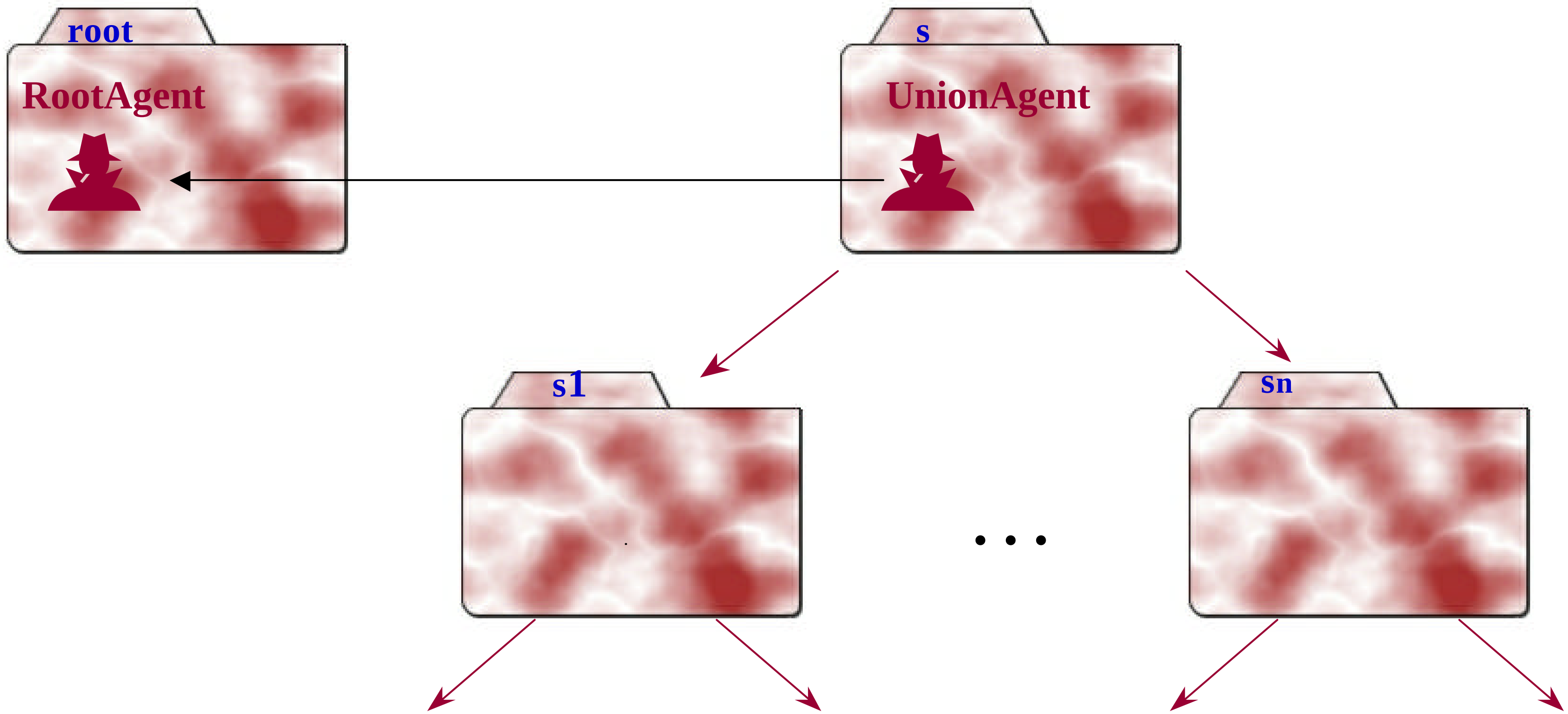
Mobile algorithm for Union-construct



Mobile algorithm for Union-construct



Mobile algorithm for Union-construct



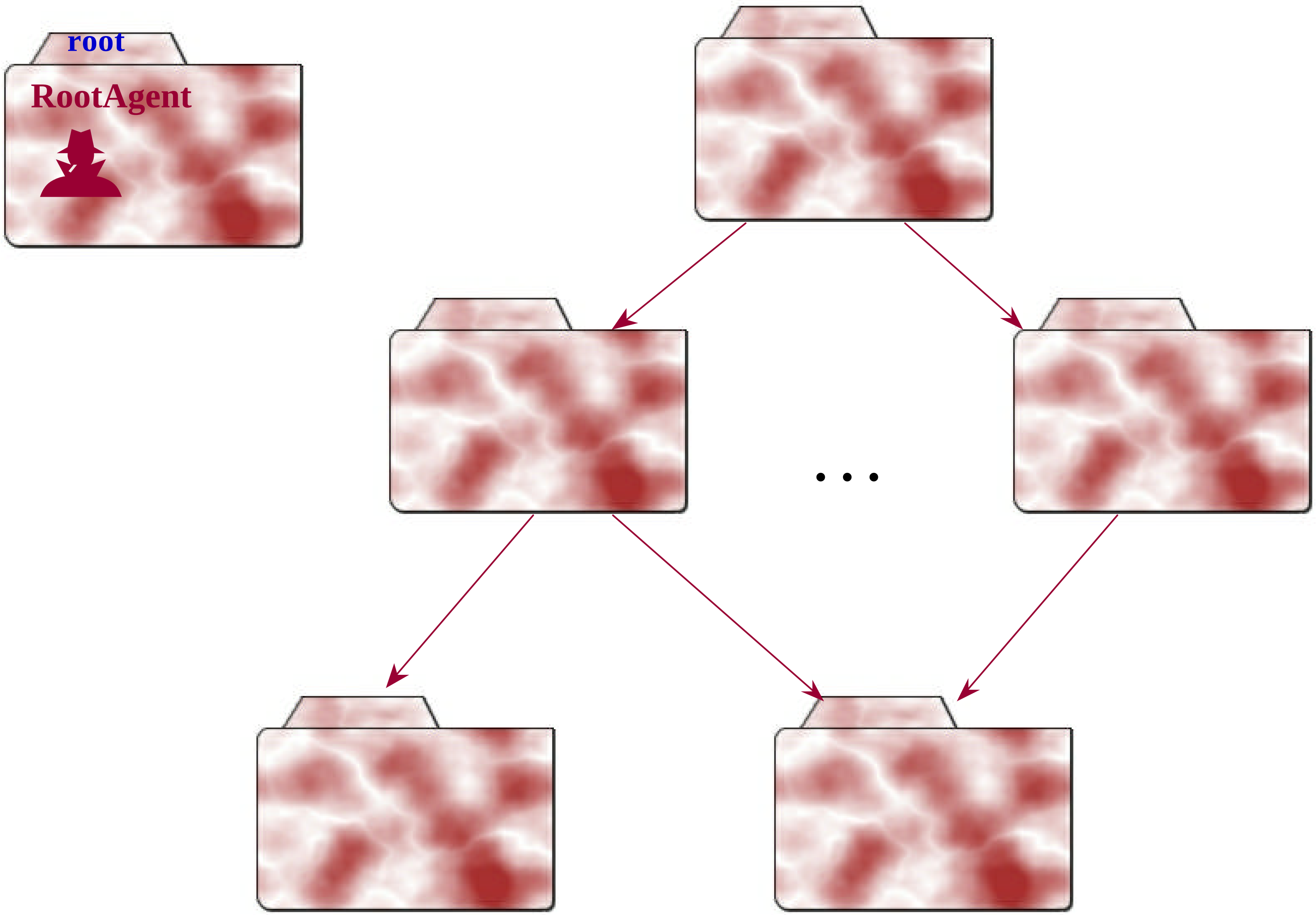
Mobile algorithm for Union-construct

UnionAgent (s) = migrate to s →
links = getLinks( );

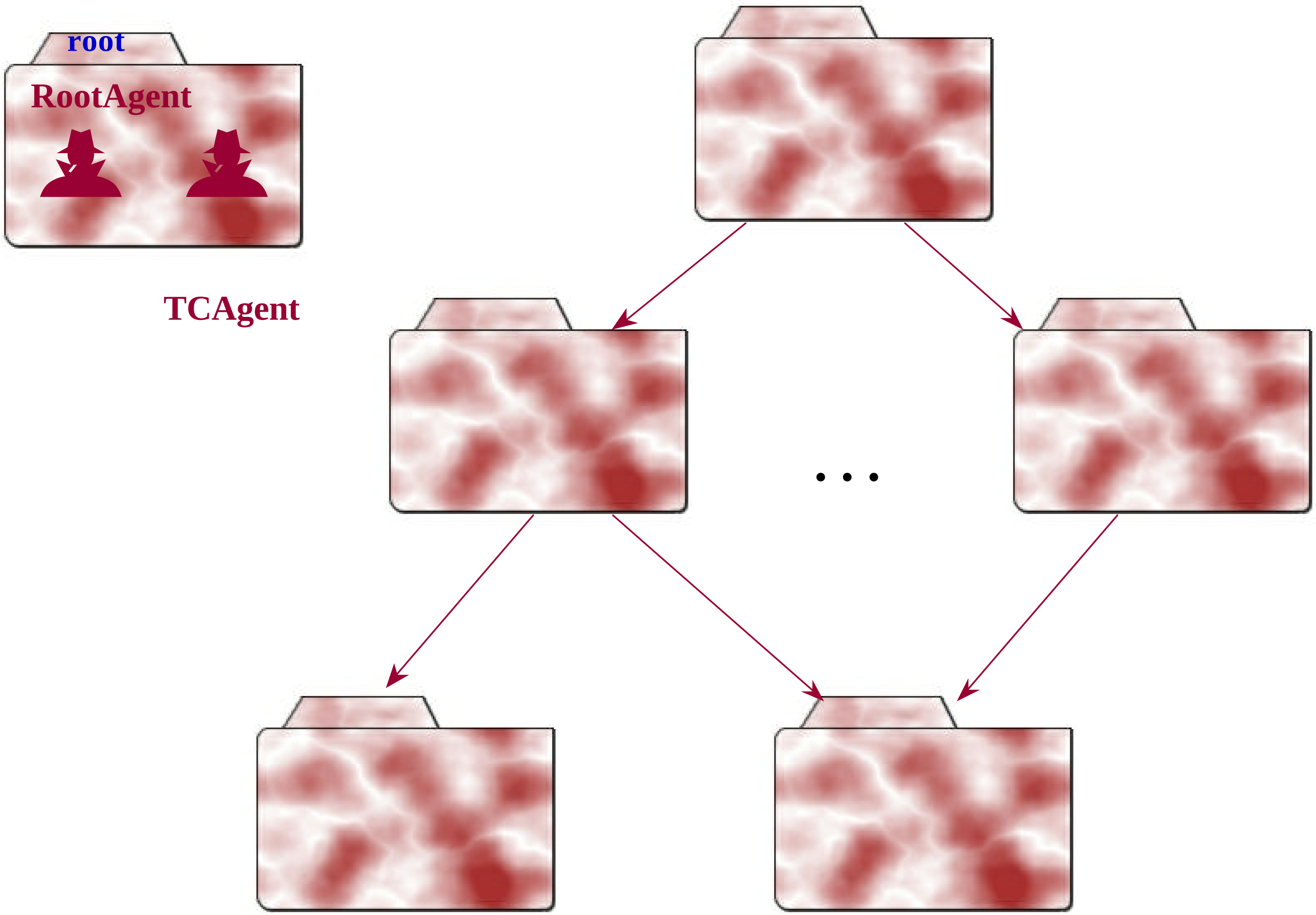
forall (s_i Î links)
create SubAgent (s_i)
|
(u = getResultsFromSubAgents();
sendToParent (u);
)

SubAgent (v) = migrate to v →
links = getLinks();
sendToParent (links);

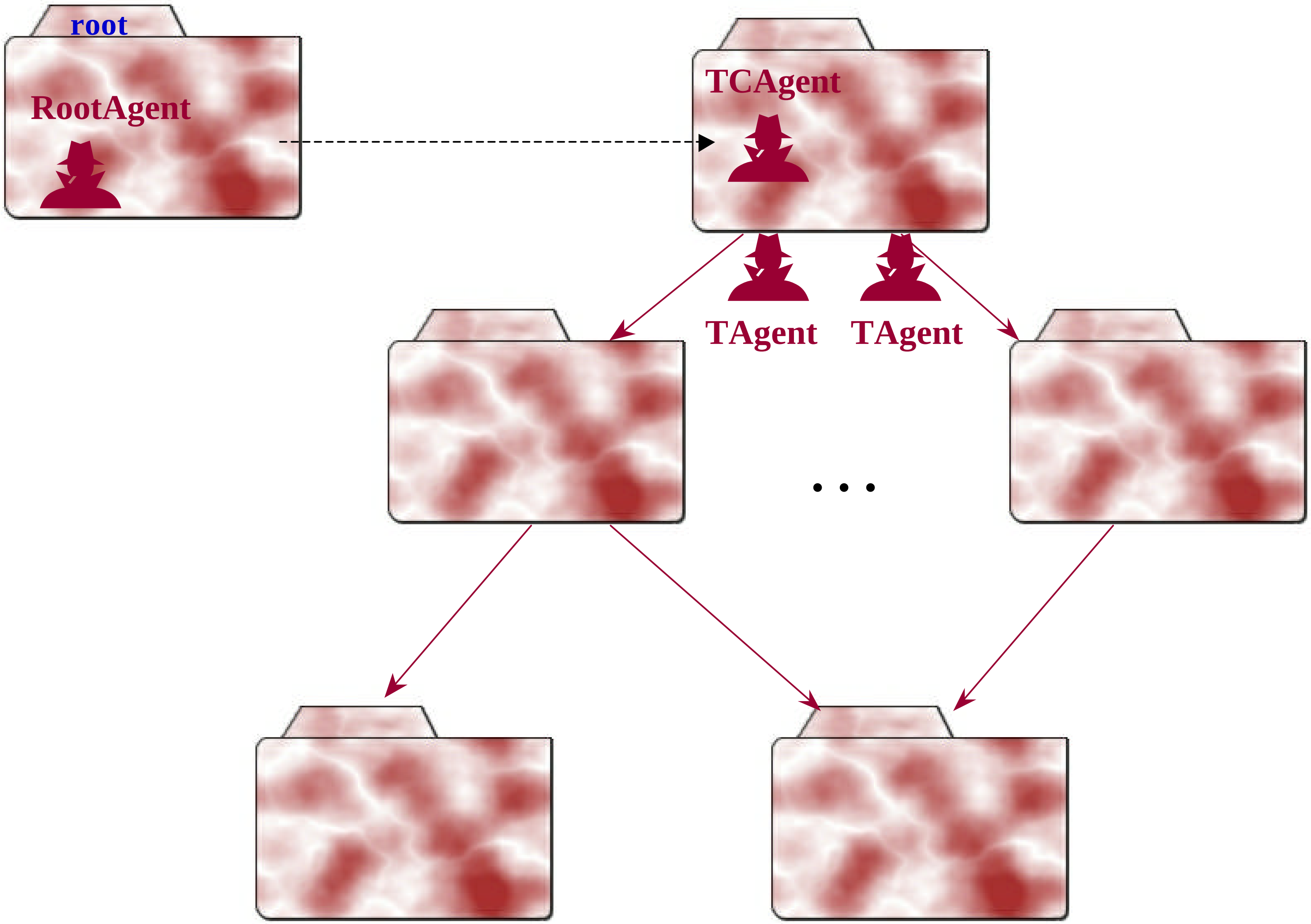
Mobile algorithm for Transitive Closure-construct



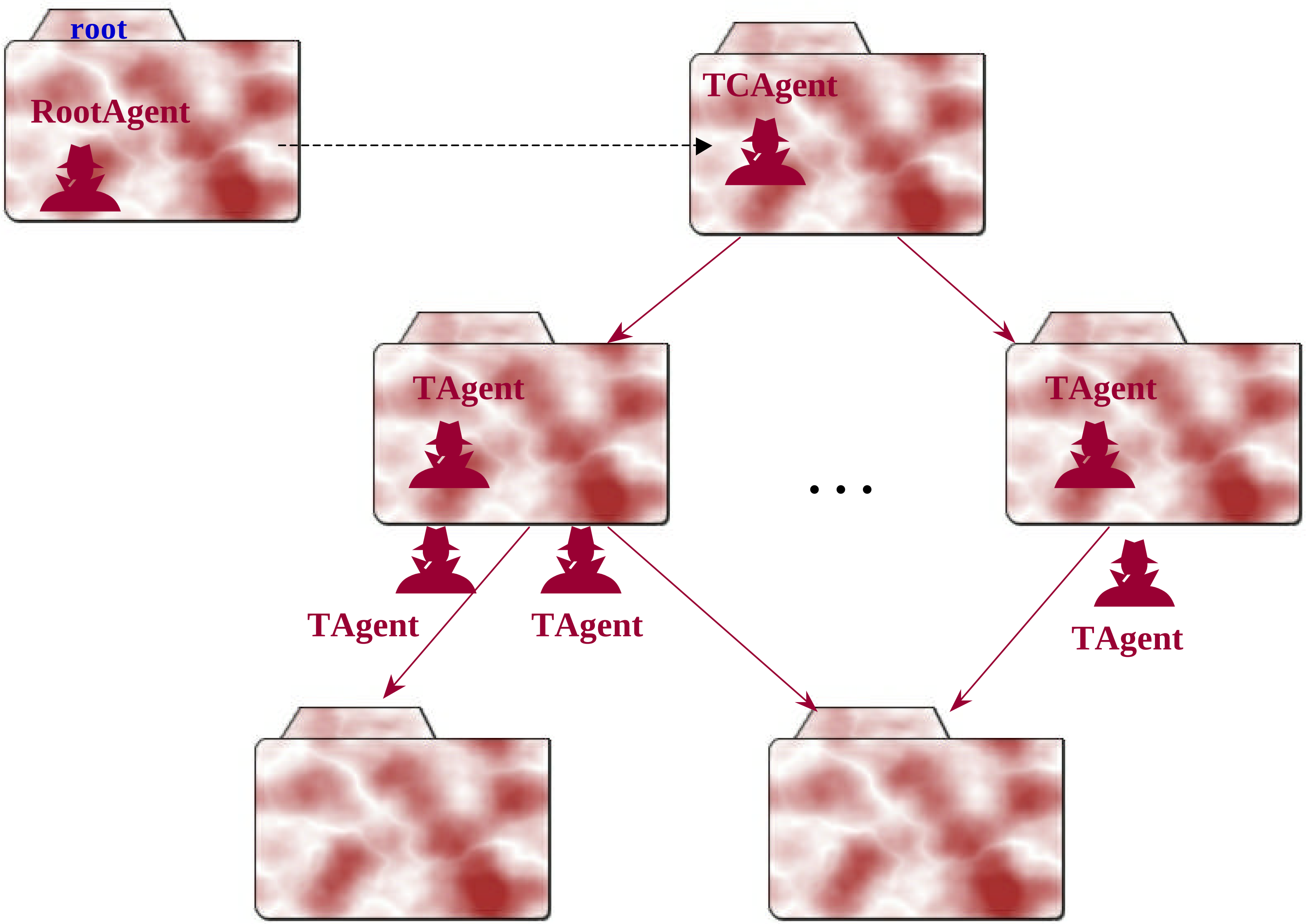
Mobile algorithm for Transitive Closure-construct



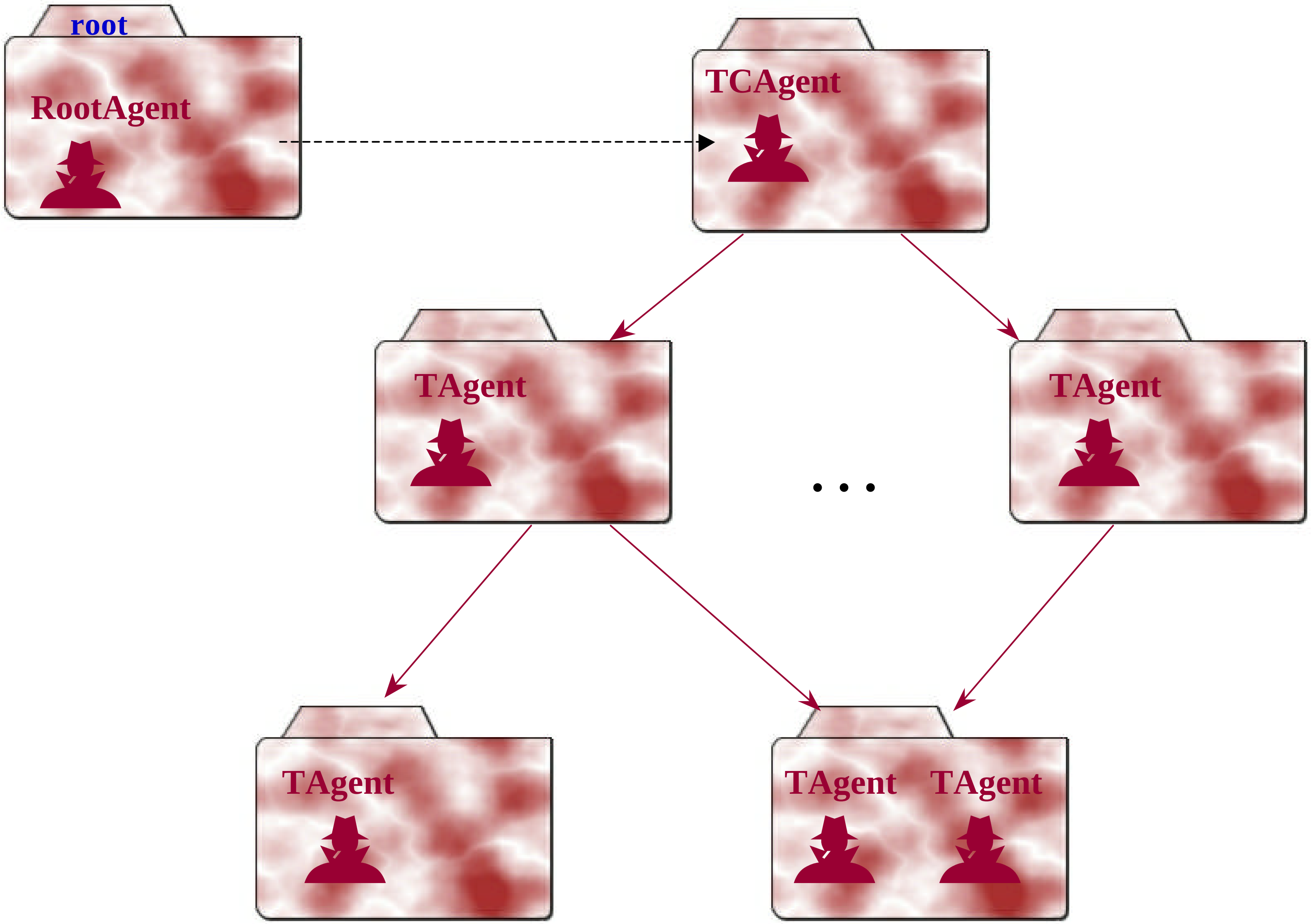
Mobile algorithm for Transitive Closure - construct



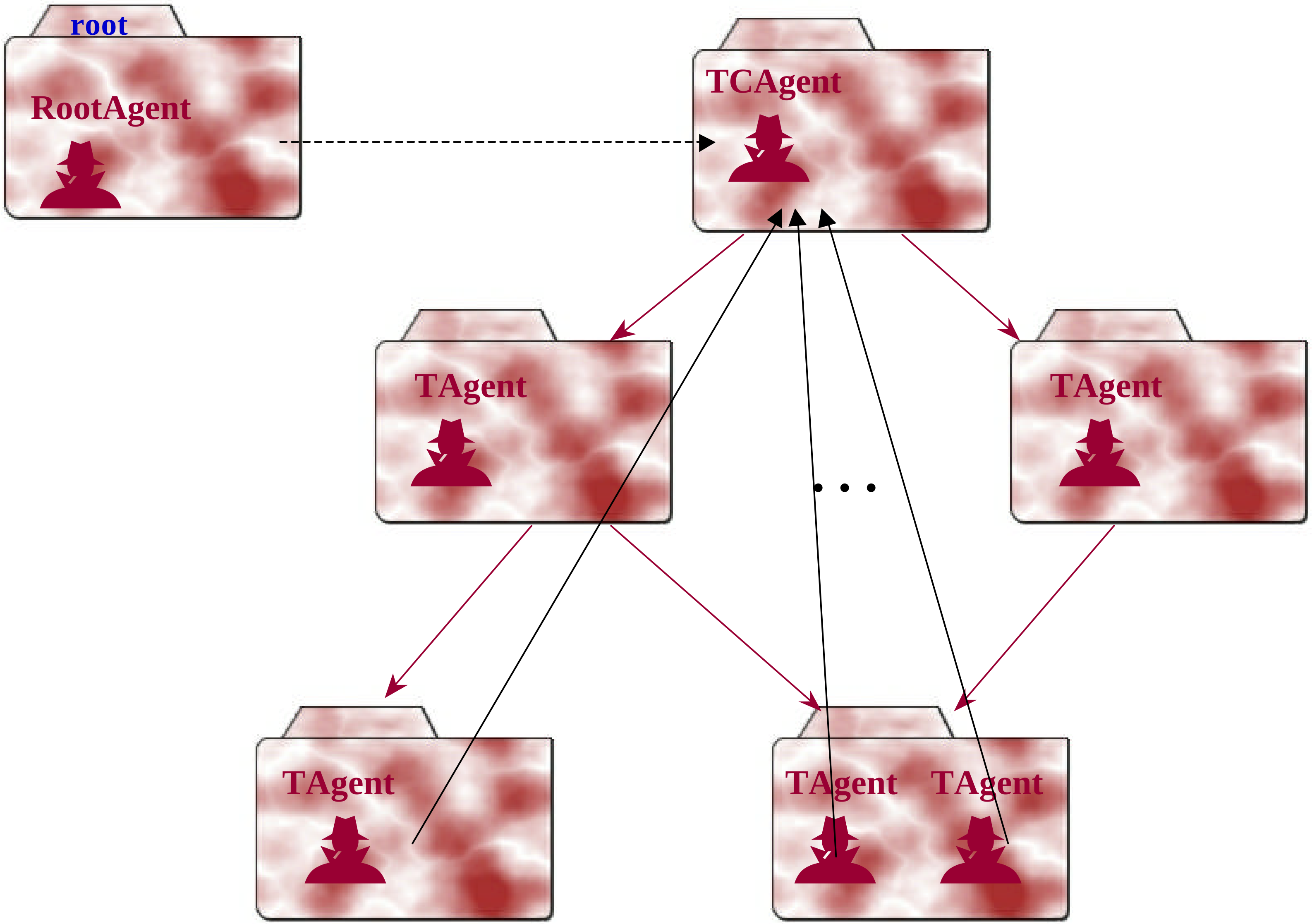
Mobile algorithm for Transitive Closure - construct



Mobile algorithm for Transitive Closure - construct



Mobile algorithm for Transitive Closure - construct



Mobile algorithm for Transitive Closure - construct

```
TCAgent ( s ) =  migrate to  s →
                        links = getLinks ( );
                        (  forall ( s_i Î  links  )
                              create  TAgent ( s_i )
                         | u = getResultsFromAllTAgents ();
                         | getSignalsFromChilds();
                          sendToParent ( u )
                        )
TAgent ( v )  =  if  (  v Î  result_set_from_TCAgent )
                 then  sendSignalToParent();
                 else
                      migrate to  →
                          links = getLinks ( );
                         (  forall ( u Î  links  )
                               create  TAgent ( u )
                           | getSignalsFromChilds ();
                            sendSignalToParent();
                         )
```

Mobile algorithm for equality - construct

Examples of queries.

1. Compare a given site S with it's mirror M_S.

   $\{ \; \{ \varnothing \} \mid S = M_S \; \}$, or , more formally , in precise Delta-syntax :

   $\{ \, l : \{ \varnothing \} \mid l : x \in \{ \varnothing \} \; \& \; S = M_S \; \}$

2. On given site S find all identical pages .

   $\{ < x, y > \mid x, y \in LTC (\{ S \}) \; \& \; x = y \; \}$

Mobile algorithm for equality - construct

Def. Let G = (V, $\rightarrow$) be a (finite-branching) graph,
and R is a binary relation on $V \times V$.
R is a **bisimulation** relation if
$\forall$ (E, F) $\in$ R
(E $\rightarrow$ E' $\Rightarrow$ F $\rightarrow$ F' & (E', F') $\in$ R) &
(F $\rightarrow$ F' $\Rightarrow$ E $\rightarrow$ E' & (E', F') $\in$ R)

R is a **non-bisimulation** relation if
$\forall$ (E, F) $\in$ R
$\exists$ E $\rightarrow$ E' $\forall$ F $\rightarrow$ F' (E', F') $\in$ R $\vee$
$\exists$ F $\rightarrow$ F' $\forall$ E $\rightarrow$ E' (E', F') $\in$ R

Mobile algorithm for equality - construct

Expansion trees [Y. Hirshfeld, 1994, P.Jancar, F.Moller,1999]

Def. Given two sets $B \neq \varnothing$ and A of pairs of vertices, A is called an **expansion** of B iff it is a minimal set (wrt inclusion) satisfying the property

$\forall$ (E, F) $\in$ B

$\exists\, E \rightarrow E' \;\forall\; F \rightarrow F' \;\; (E', F') \in A \quad \vee$

$\exists\, F \rightarrow F' \;\forall\; E \rightarrow E' \;\; (E', F') \in A$

Properties.

1. A non-empty set B **fails** to have an expansion iff
   $\exists$ (E, F) $\in$ B (out_degree (E) = out_degree (F) = 0)

2. A non-empty set B has (the single) expansion $\varnothing$ iff
   $\forall$ (E, F) $\in$ B $\;E \not\sim_1 F$
   (where $E \not\sim_1 F$, if out_degree (E) = 0 & out_degree (F) $\neq$ 0 or symmetrically

Mobile algorithm for equality - construct

Expansion trees [Y. Hirshfeld, 1994, P.Jancar, F.Moller,1999]

Properties.

3. An any (non-empty) finite set of pairs of vertices has only finitely many expansions, all of which are finite.

4. Let R be a non-bisimulation relation, and let B ($\neq \emptyset$) $\subseteq$ R.
Then there exists a some expansion A of B such that

$$A \subseteq R.$$

(Proof is based on dual variant of Zorn's Lemma for maximal elements in sets).

5. Let B $\neq \emptyset$ and A is an expansion of B, such that

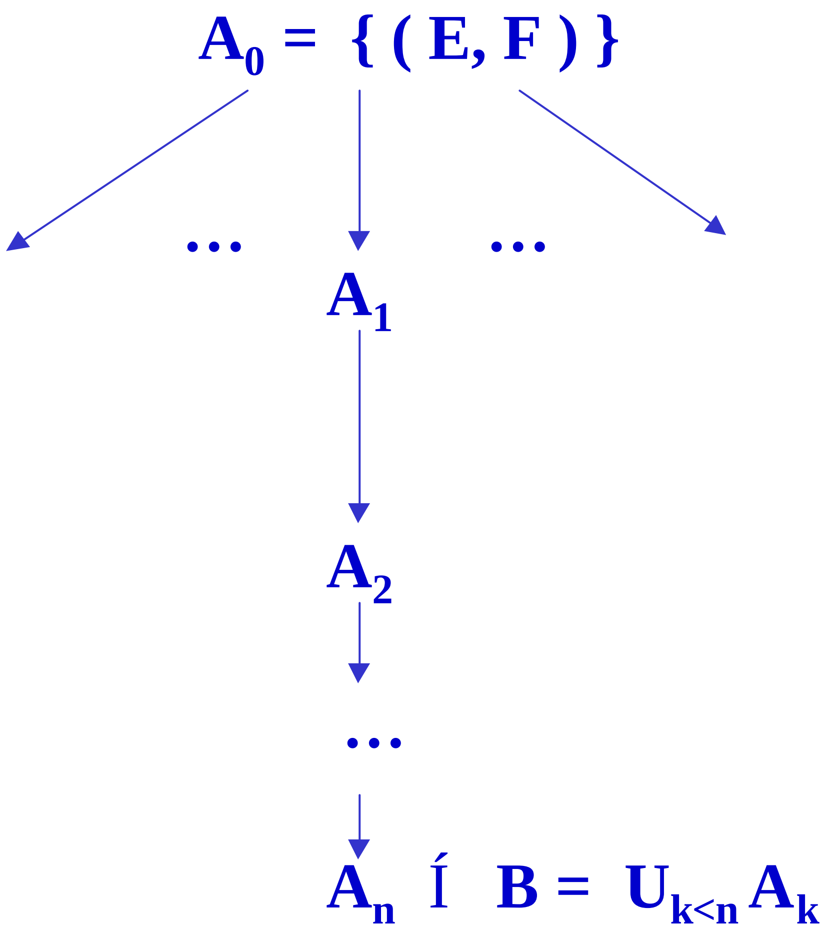
$$A \subseteq B$$

Then B is a non-bisimulation relation.

Proof. (E, F) $\in$ B $\Rightarrow$
$\exists$ E $\to$ E' $\forall$ F $\to$ F' (E', F') $\in$ A $\subseteq$ B or symmetrically.
So (E', F') $\in$ B.

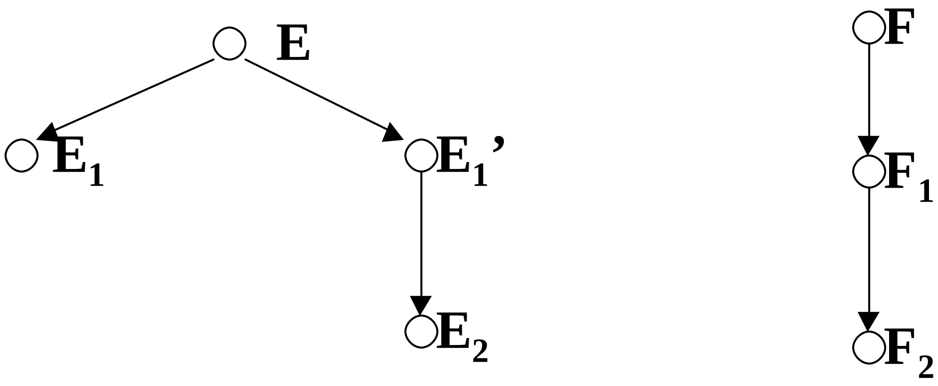
Mobile algorithm for equality - construct

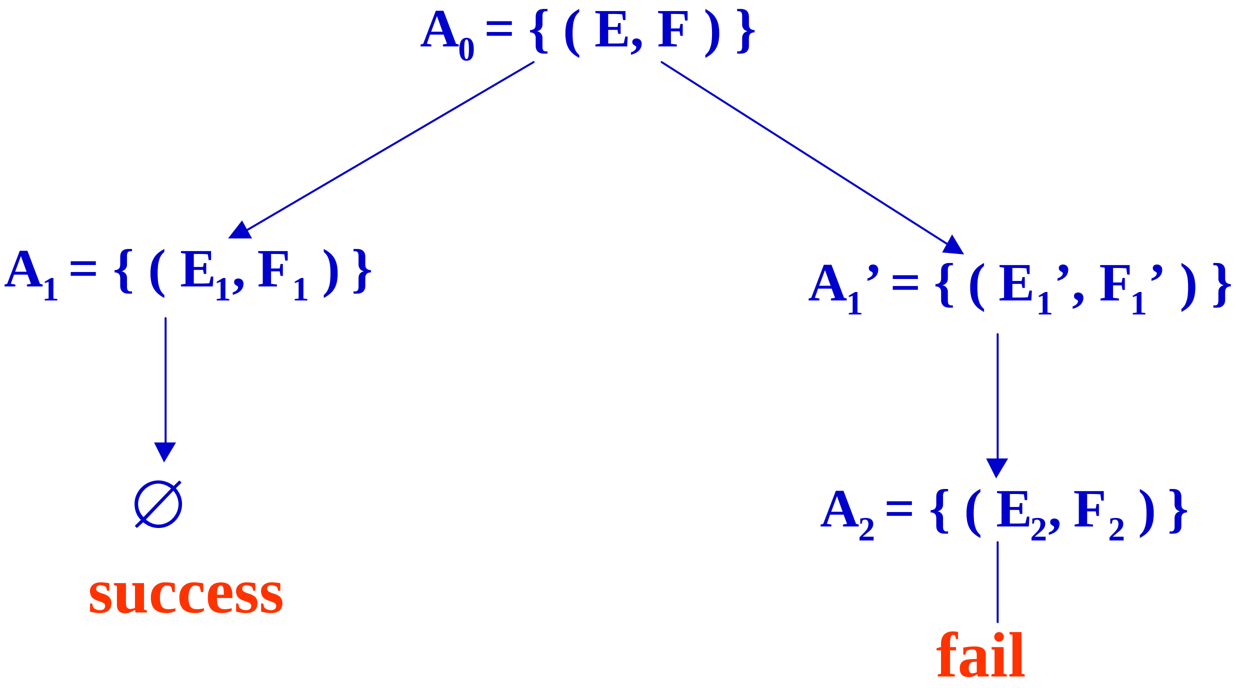
Algorithm for testing E $\not\sim$ F



Mobile algorithm for equality - construct

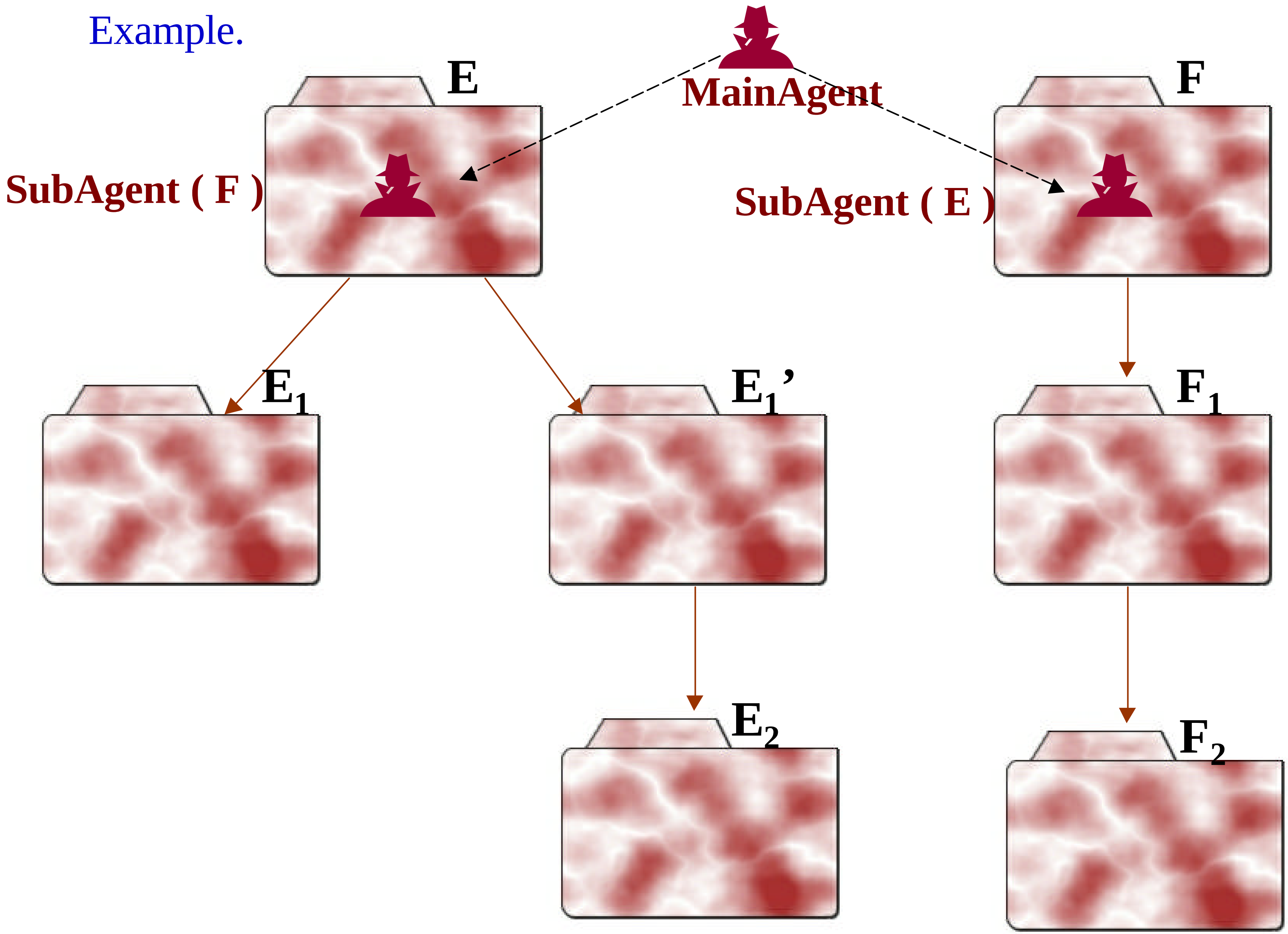
Example.





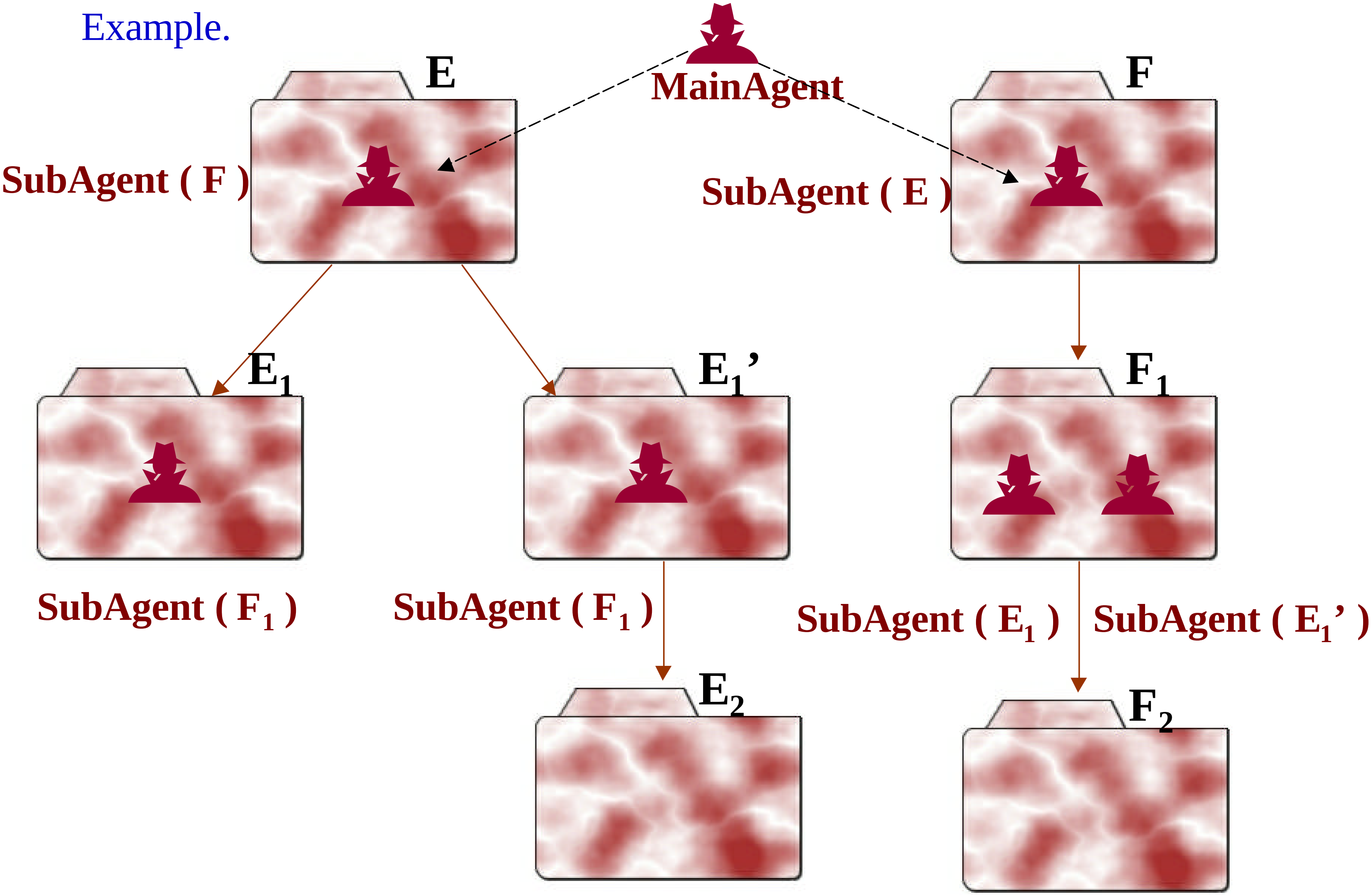
Mobile algorithm for equality - construct

Example.



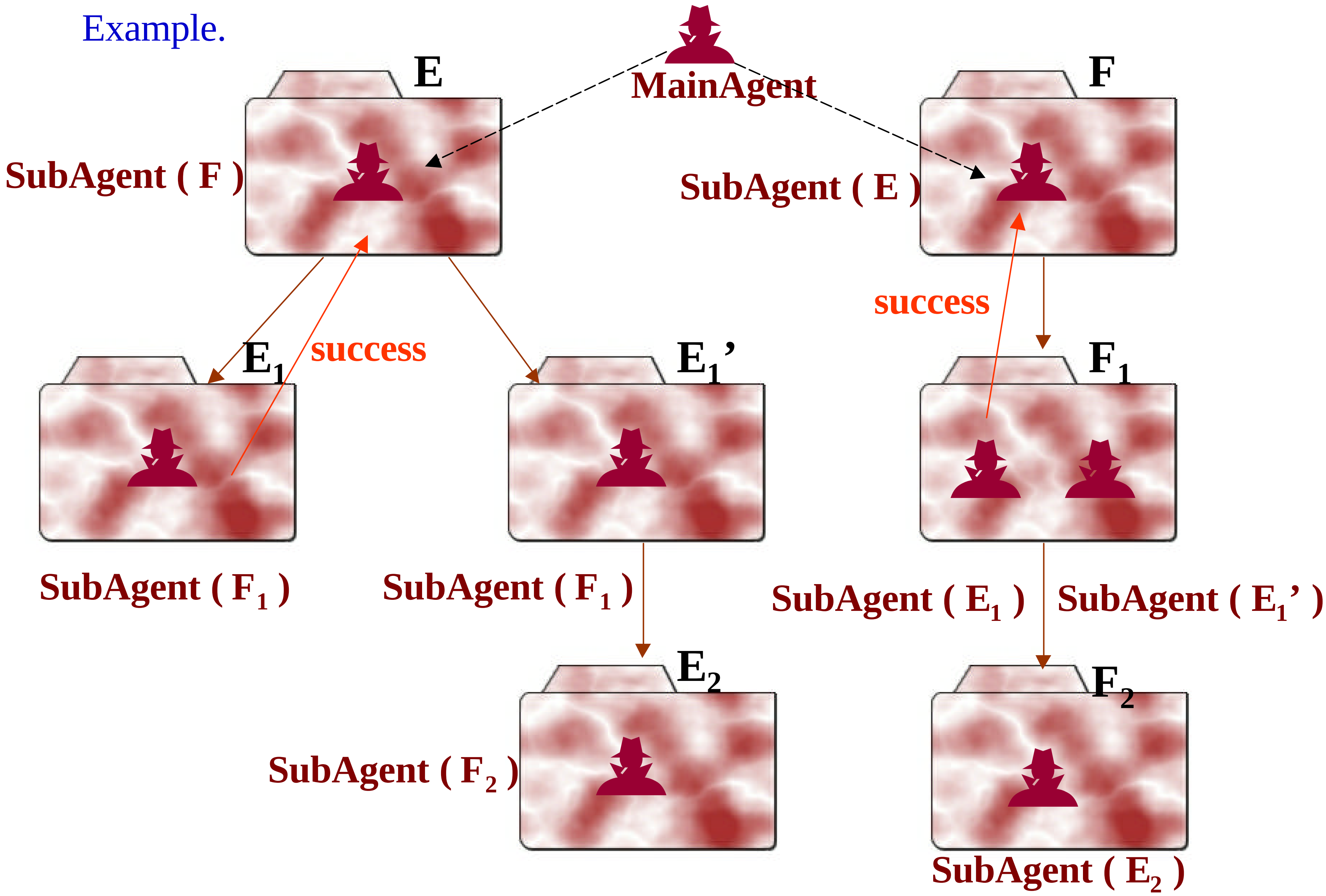
Mobile algorithm for equality - construct

Example.



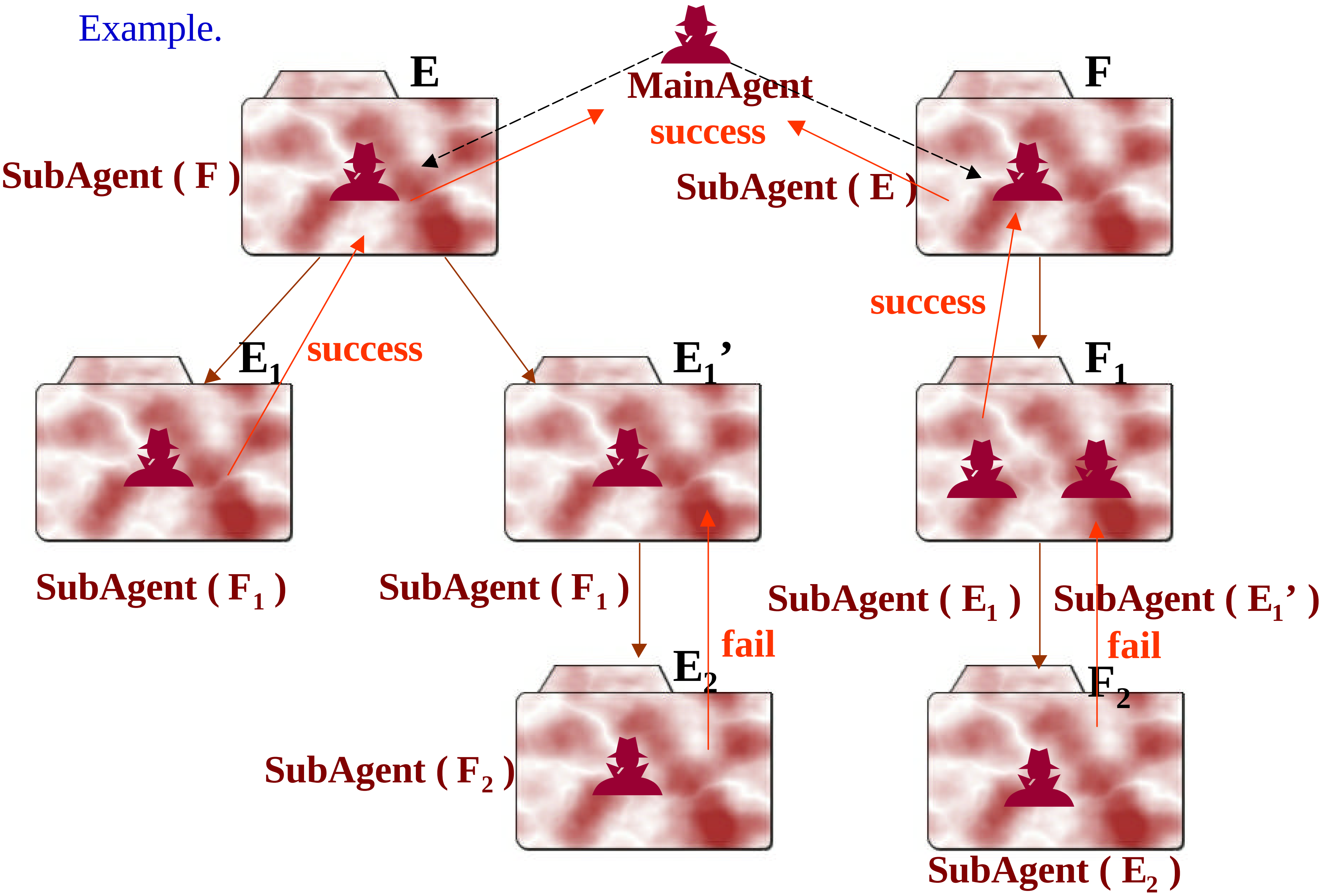
Mobile algorithm for equality - construct

Example.



Mobile algorithm for equality - construct

Example.



Application configuration