

BACI

Boxed Ambients with Communication Interfaces

joint work with Eduardo Bonelli, Adriana Compagnoni and
Pablo Garralda

drawings by L. Cardelli and P. Garralda

MFCS'04

29th International Symposium on Mathematical
Foundations of Computer Science

Thursday, 26 August 2004

Prague, Czech Republic



MIRADO

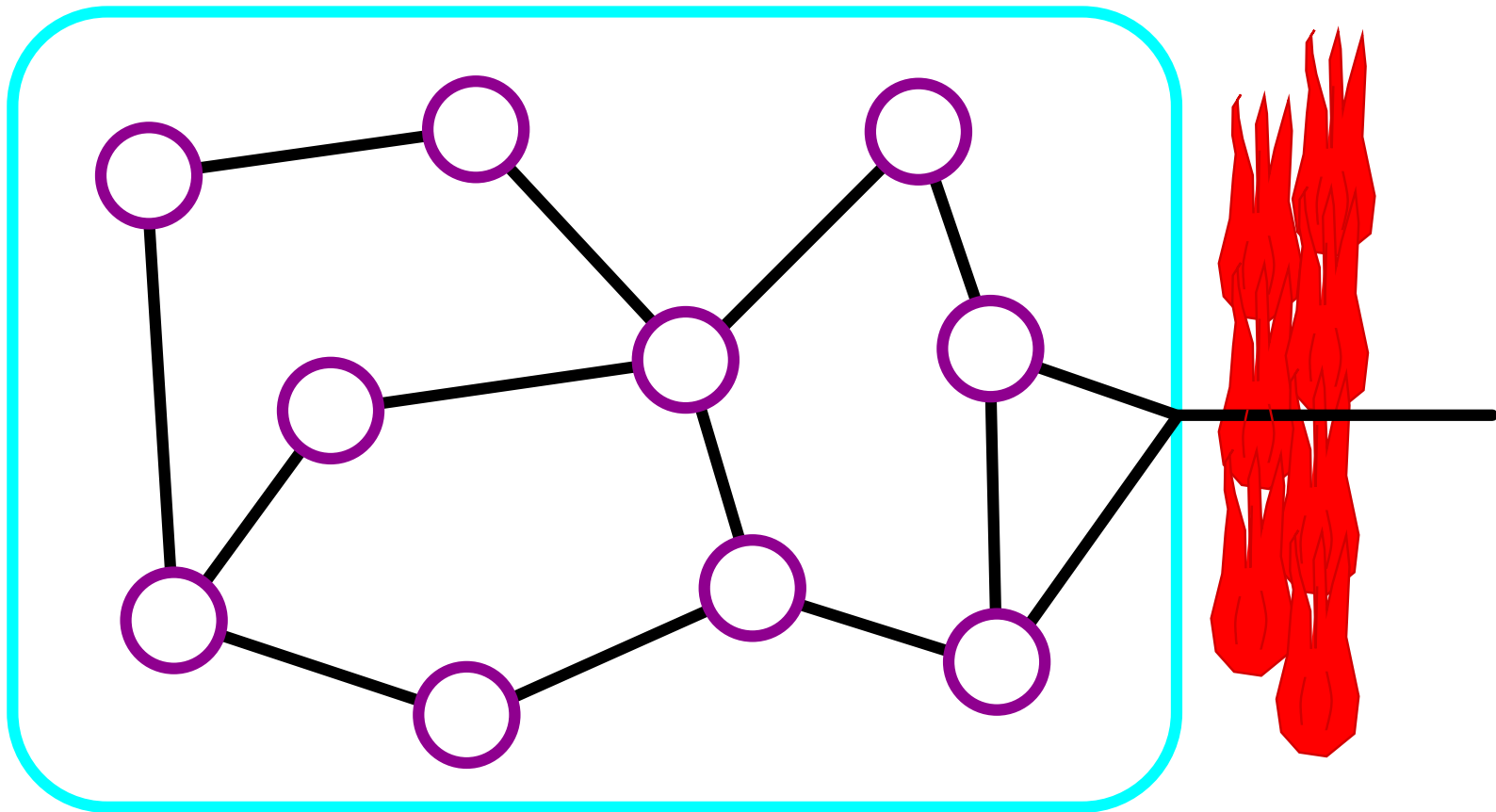
Global Computing

one computer



Global Computing

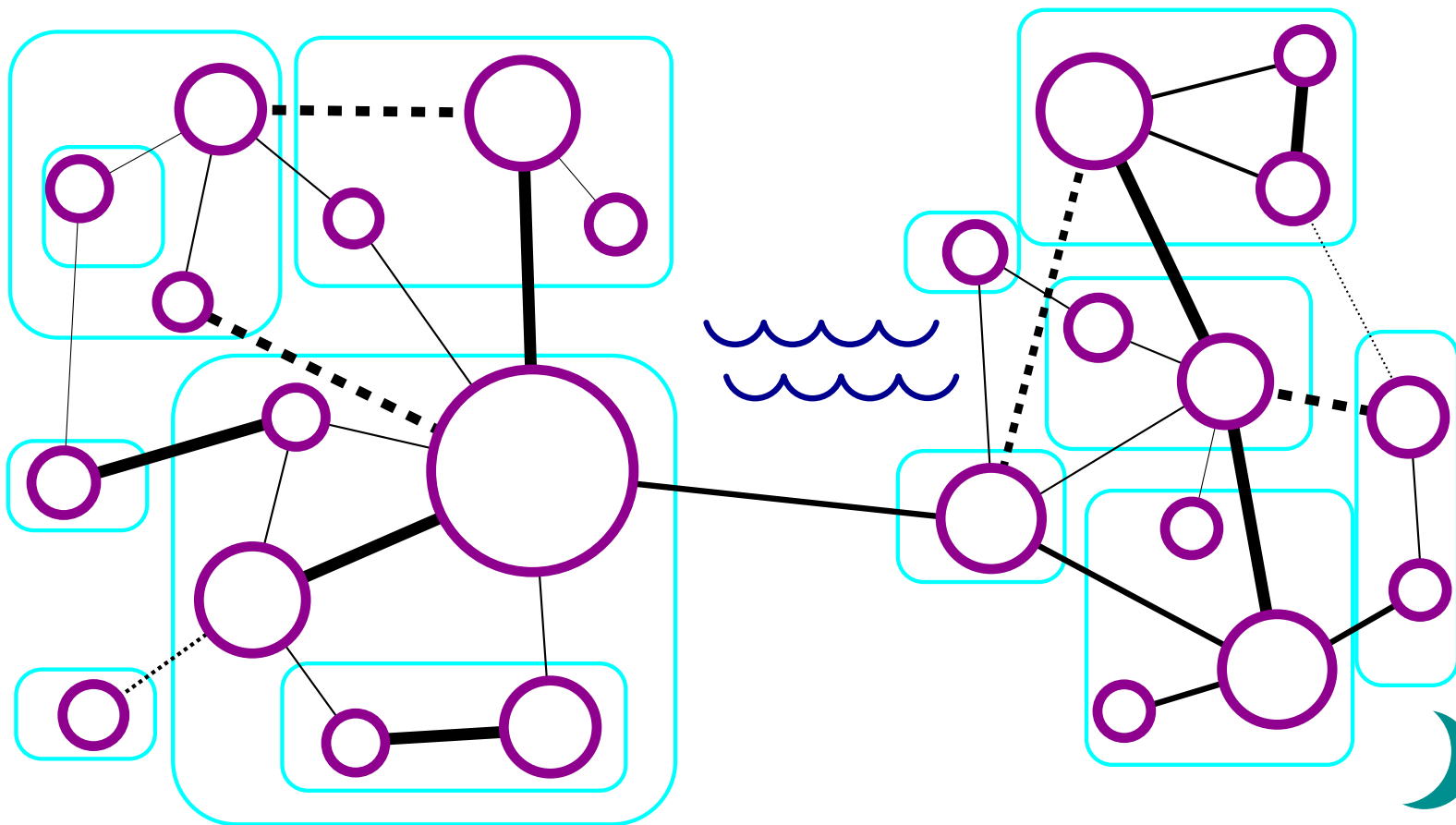
a local area network of computers



Global Computing



a wide area network of computers



Global Computing

a new dimension of computing:
localities and movement between
localities

New Models of Computing

- localities and domains

New Models of Computing

- localities and domains
- process movements

New Models of Computing

- localities and domains
- process movements
- partial knowledge

New Models of Computing

- localities and domains
- process movements
- partial knowledge
- security and trust

New Models of Computing

- localities and domains
- process movements
- partial knowledge
- security and trust
- dynamic changes

New Models of Computing

- localities and domains
- process movements
- partial knowledge
- security and trust
- dynamic changes
 - knowledge acquisition

New Models of Computing

- localities and domains
- process movements
- partial knowledge
- security and trust
- dynamic changes
 - knowledge acquisition
 - a trusted agent becomes untrusted or vice versa

New Typing Disciplines

- **aim**: to control movement and communication

New Typing Disciplines

- **aim**: to control movement and communication
- **features**:

New Typing Disciplines

- **aim**: to control movement and communication
- **features**:
 - local types: no global assumption

New Typing Disciplines

- **aim**: to control movement and communication
- **features**:
 - local types: no global assumption
 - dynamic typing: type information evolves

New Typing Disciplines

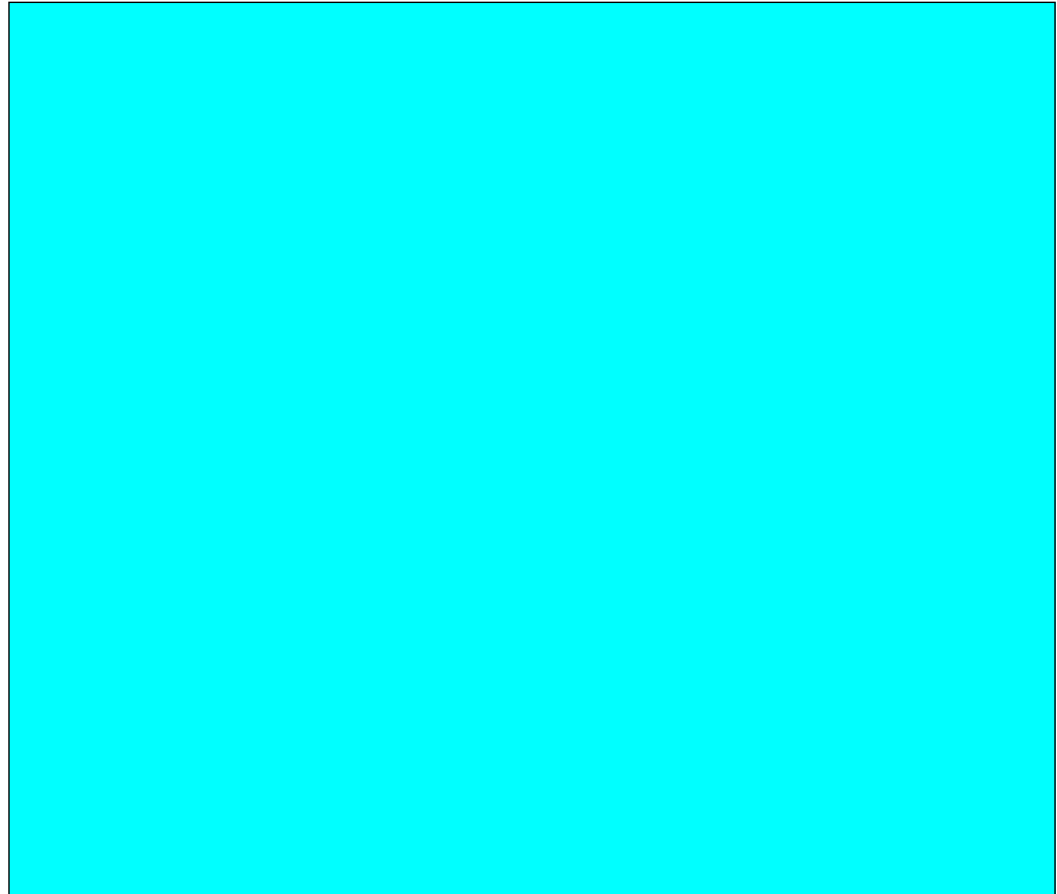
- **aim**: to control movement and communication
- **features**:
 - local types: no global assumption
 - dynamic typing: type information evolves
 - the operational semantics checks types

Boxed Ambients (BA)

ambient

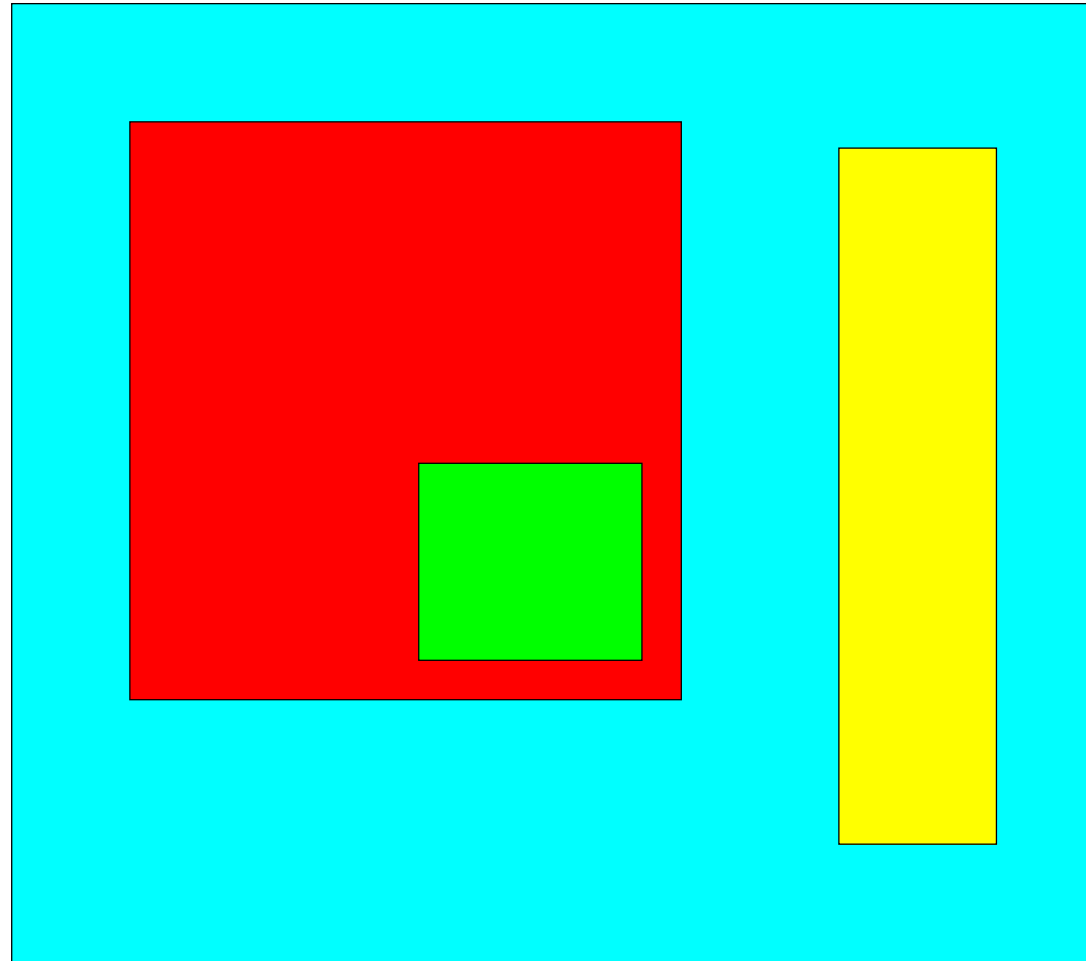
or

domain

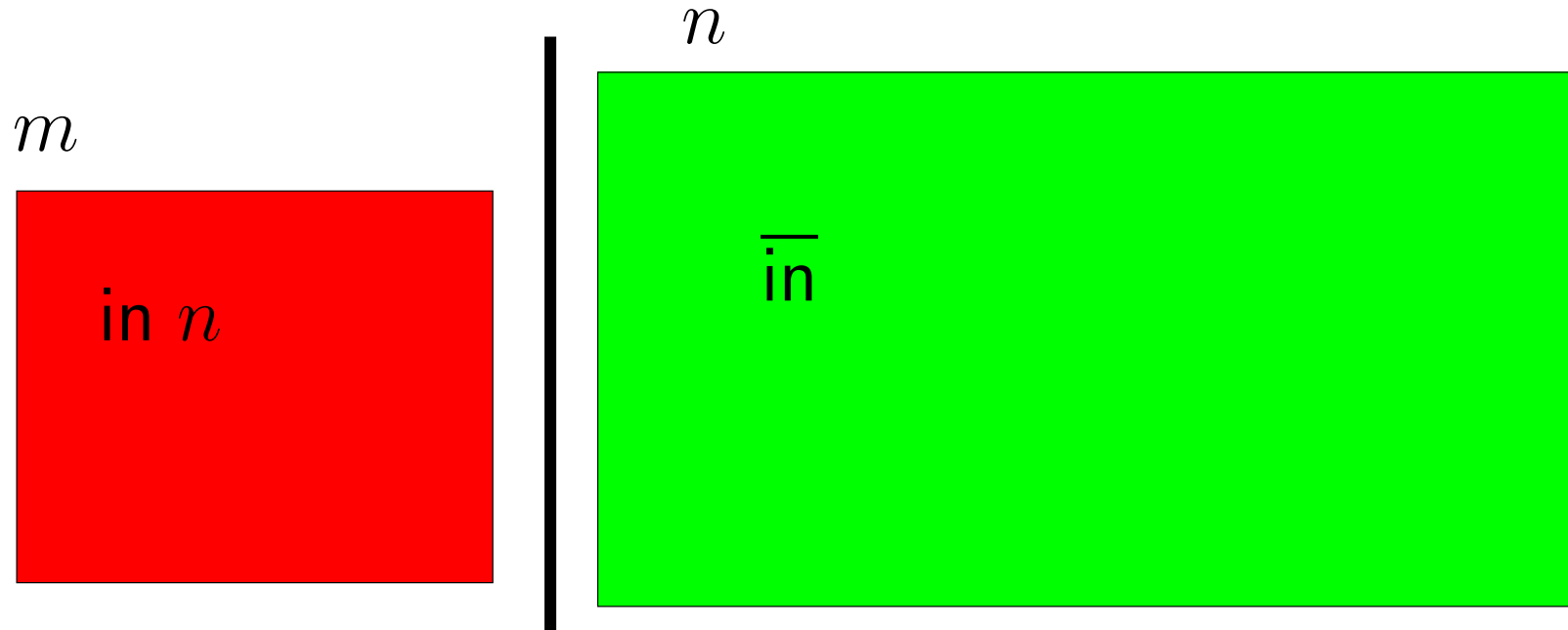


Boxed Ambients (BA)

ambients
are
possibly
nested

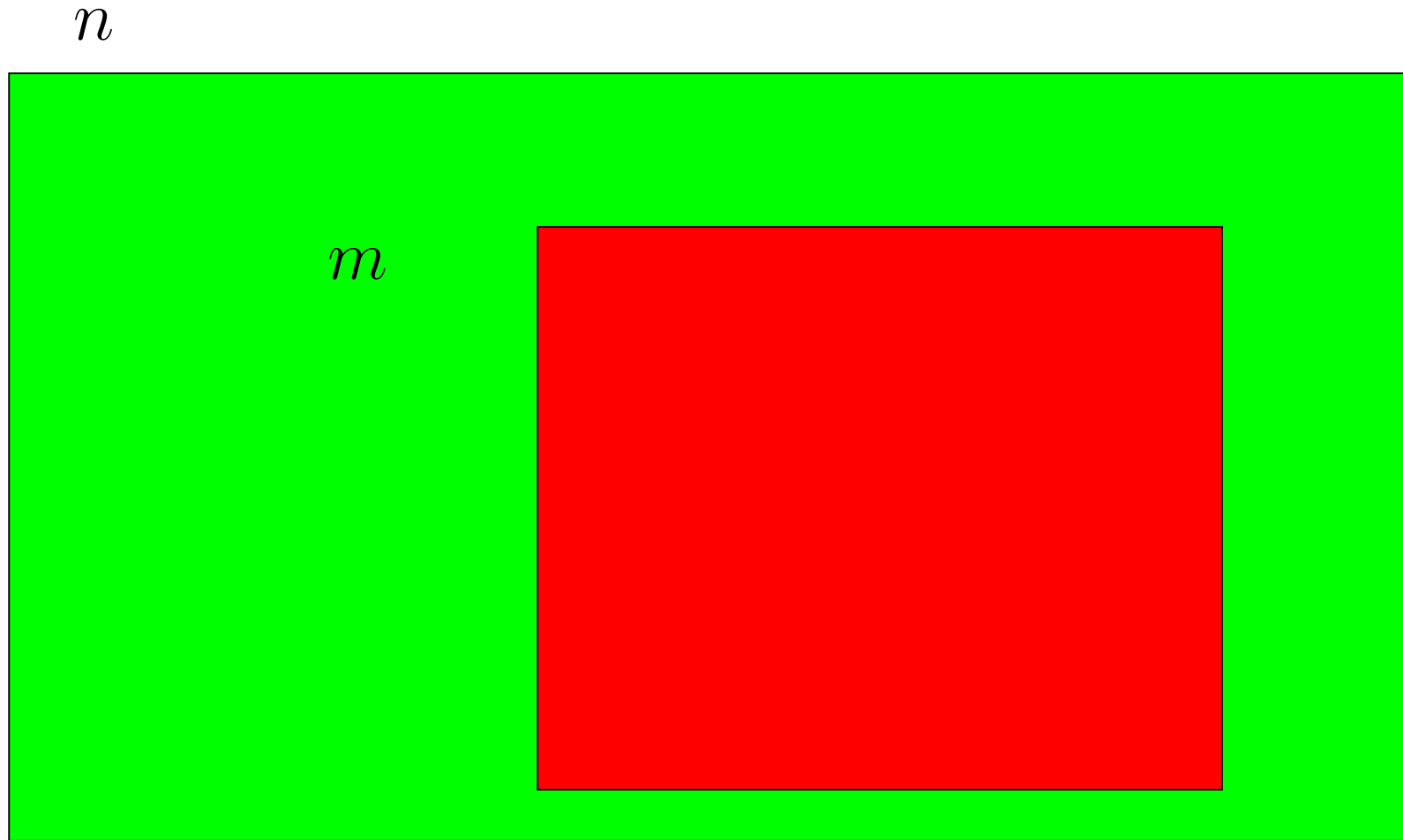


Movements: IN



m wants to enter n

Movements: IN



m is now inside n

Movements: IN

$m[\text{in } n \dots] \mid n[\overline{\text{in}} \dots]$

Movements: IN

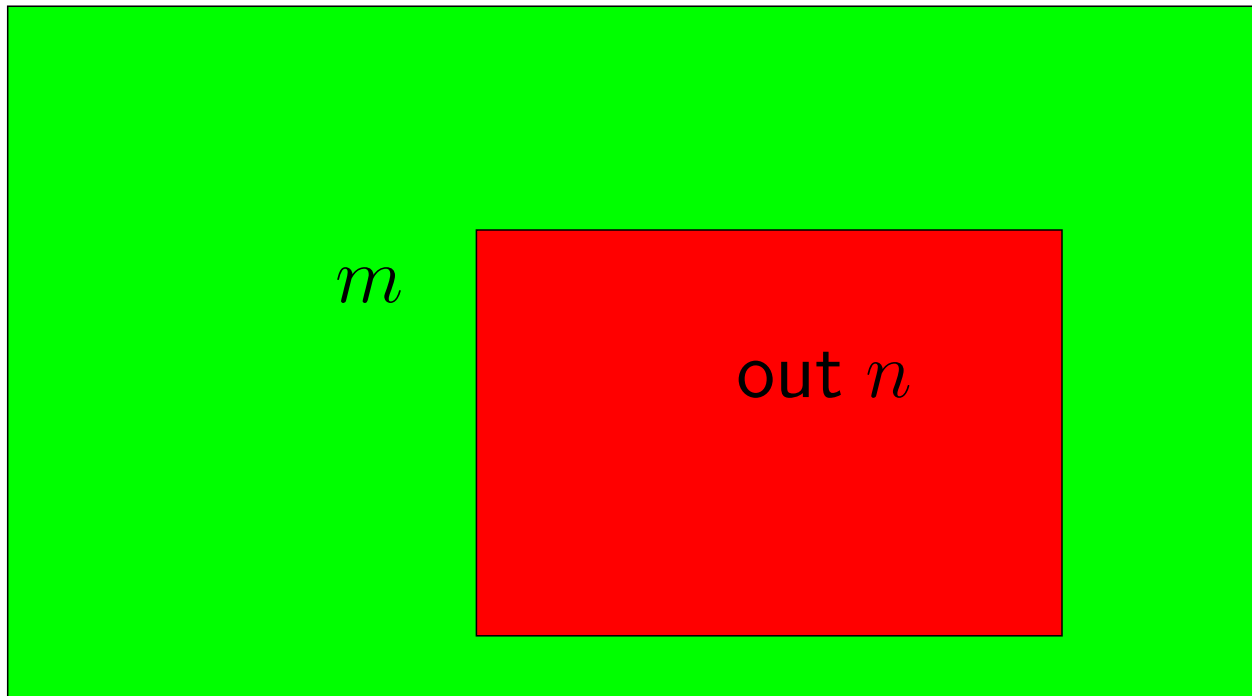
$$m[\text{in } n \dots] \mid n[\overline{\text{in}} \dots]$$

→

$$n[m[\dots] \dots]$$

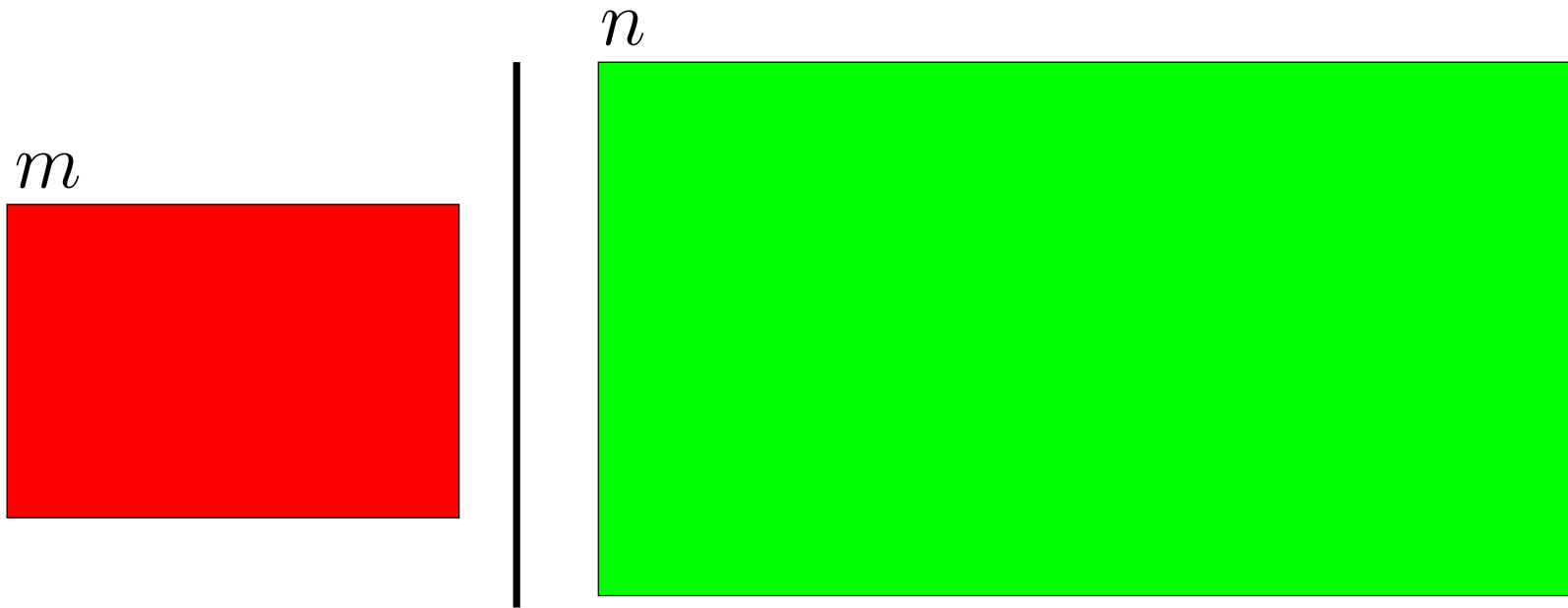
Movements: OUT

n



m wants to exit n

Movements: OUT



m is now out of n

Movements: OUT

$$n[m[out\ n \dots] \dots] \mid \overline{out}$$

Movements: OUT

$$n[m[\text{out } n \dots] \dots] \mid \overline{\text{out}}$$

$$m[\dots] \mid n[\dots]$$

Communication: local

n


$$(x).P \mid \langle 3 \rangle$$

an input $(x).P$ and an output $\langle 3 \rangle$ in the same ambient

Communication: local

n

$$P\{x := 3\}$$

communicate producing $P\{x := 3\}$

Communication: local

$$n[(x).P \mid \langle 3 \rangle \cdots]$$

Communication: local

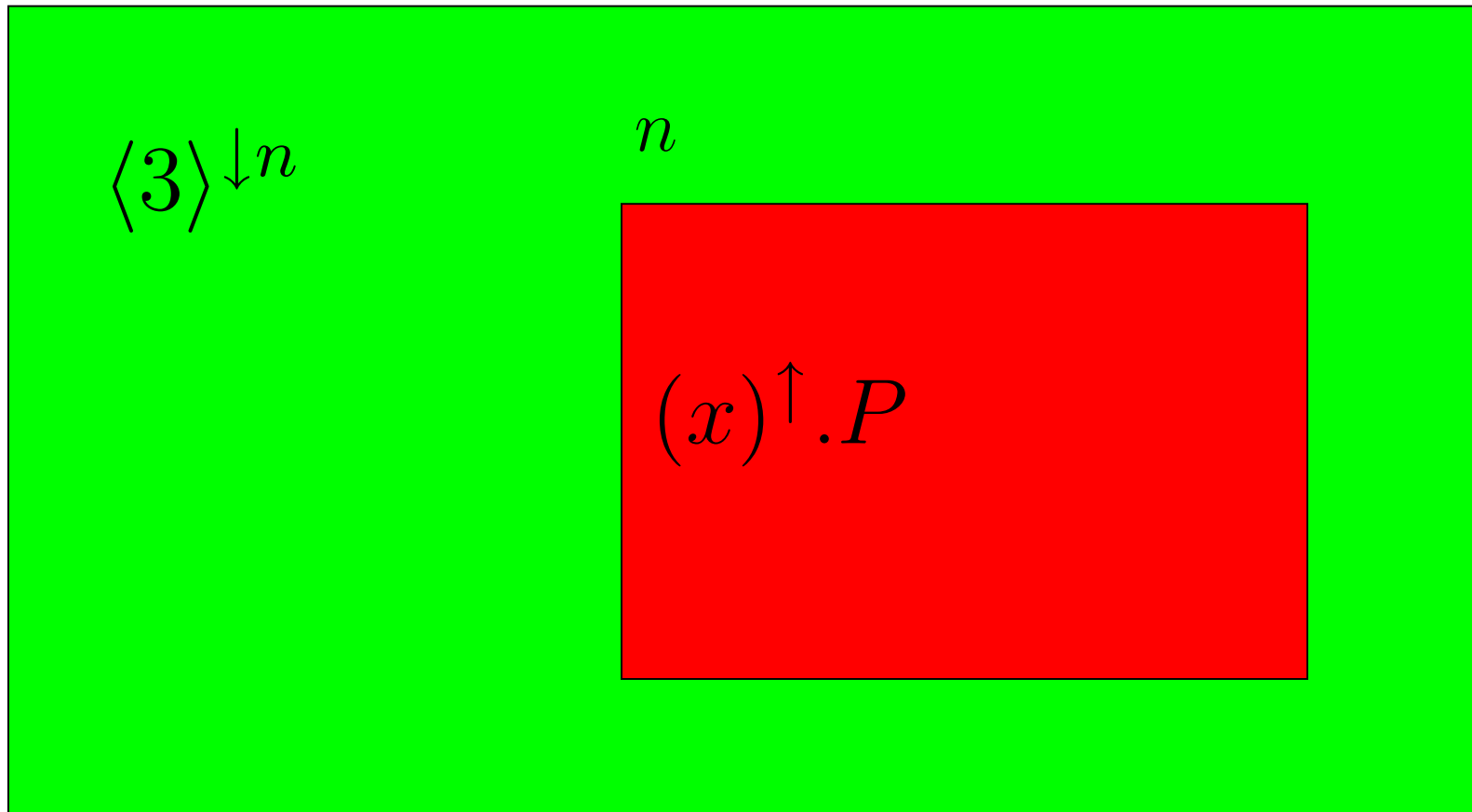
$$n[(x).P \mid \langle 3 \rangle \cdots]$$



$$n[P\{x := 3\} \cdots]$$

Communication: parent-child

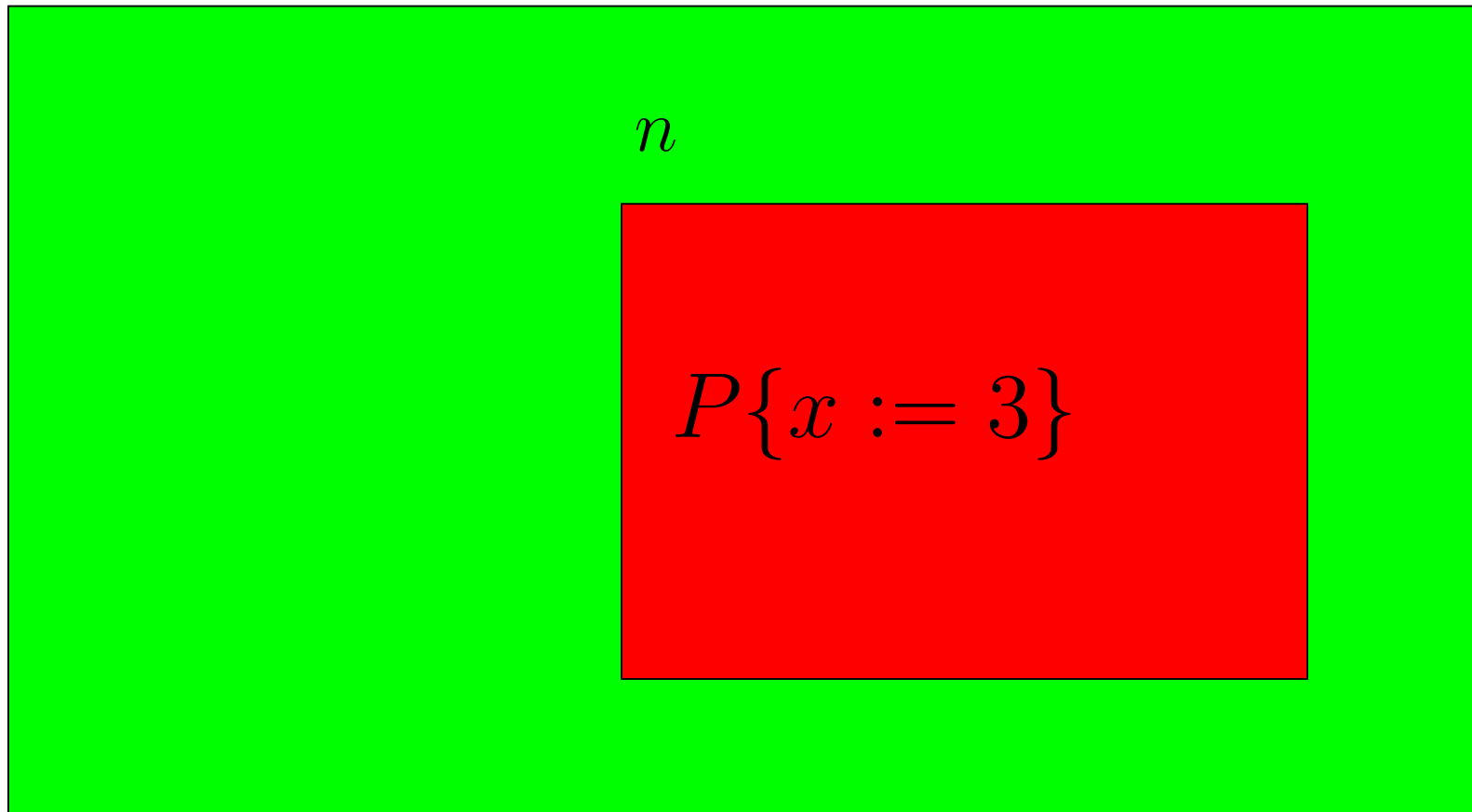
m



an input $(x)^{\uparrow} \cdot P$ in the child n
and an output $\langle 3 \rangle^{\downarrow n}$ in the parent

Communication: parent-child

m



communicate producing $P\{x := 3\}$ in the child n

Communication: parent-child

$$\dots \langle 3 \rangle^{\downarrow n} \mid n[(x)^{\uparrow} . P \dots]$$

Communication: parent-child

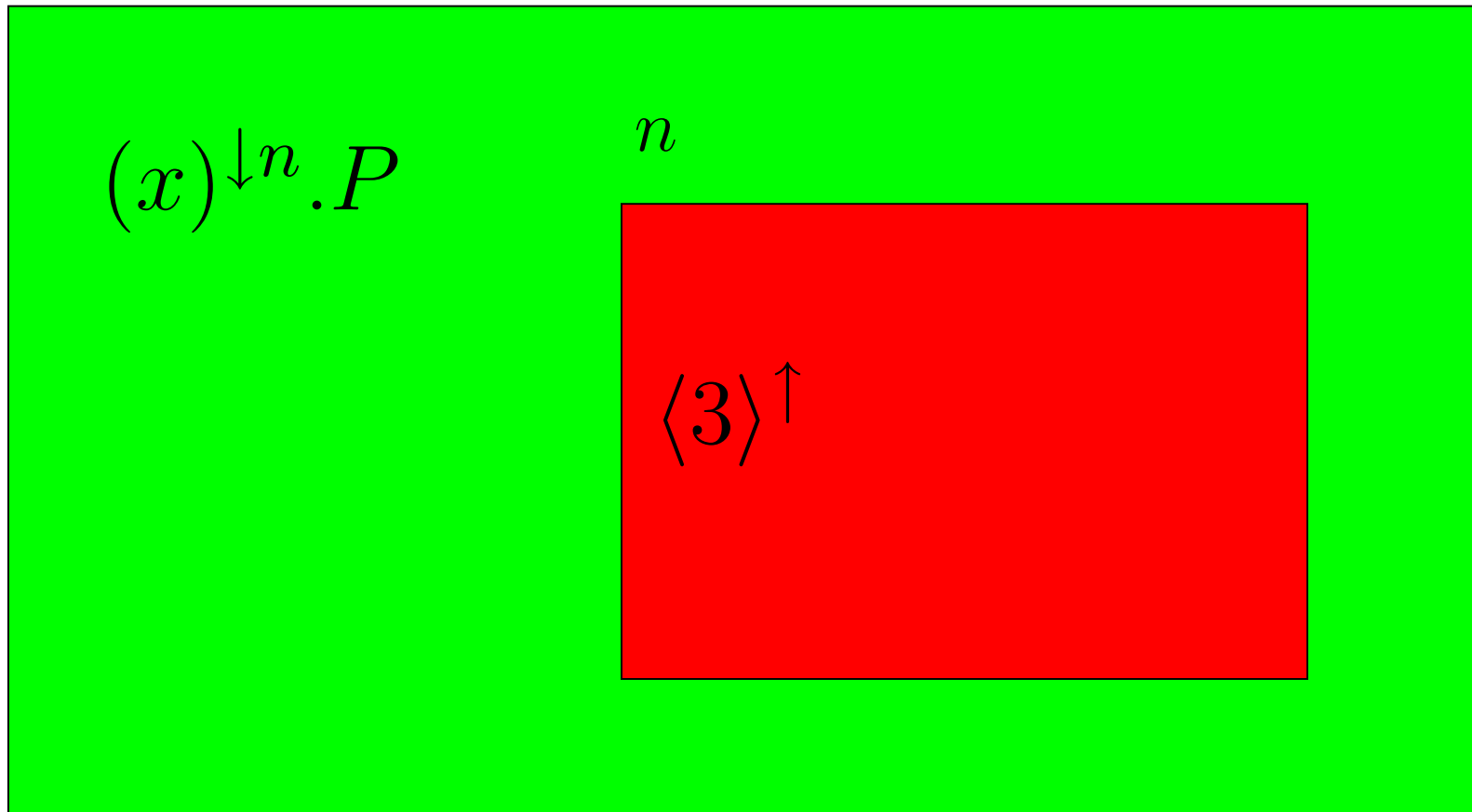
$$\dots \langle 3 \rangle^{\downarrow n} \mid n[(x)^{\uparrow}.P \dots]$$

→

$$\dots n[P\{x := 3\} \dots]$$

Communication: child-parent

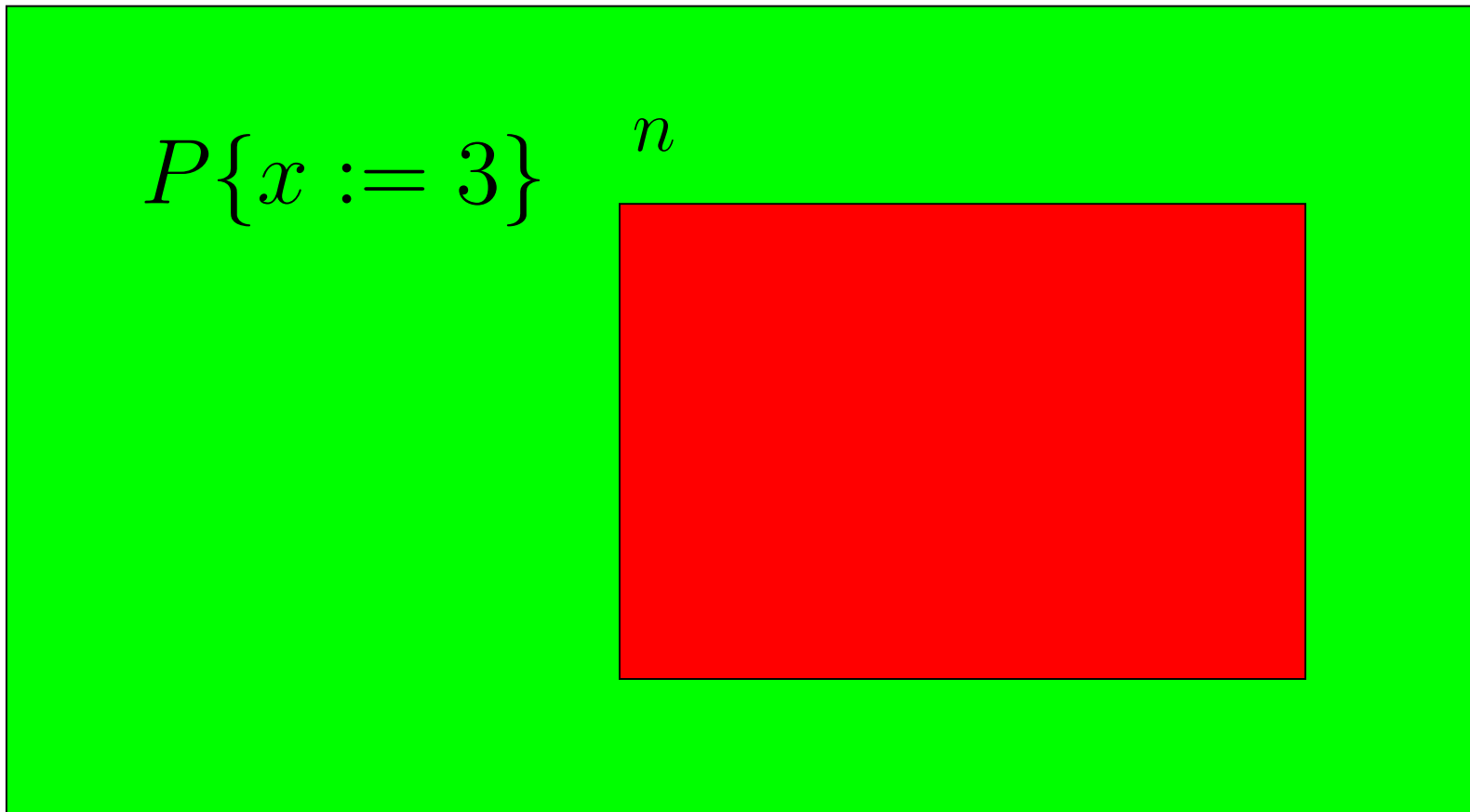
m



an input $(x) \downarrow^n . P$ in the parent
and an output $\langle 3 \rangle \uparrow$ in the child n

Communication: child-parent

m



communicate producing $P\{x := 3\}$ in the parent

Communication: child-parent

$$\dots (x)^{\downarrow n} .P \mid n[\langle 3 \rangle^{\uparrow} \dots]$$

Communication: child-parent

$$\dots (x) \downarrow^n .P \mid n[\langle 3 \rangle \uparrow \dots]$$

→

$$\dots P\{x := 3\}n[\dots]$$

Types for BA

types assure the correctness of communication: they specify the topic of conversation (ToC)

Types for BA

types assure the correctness of communication: they specify the topic of conversation (ToC)

each ambient name has associated:

Types for BA

types assure the correctness of communication: they specify the topic of conversation (ToC)

each ambient name has associated:

- a local ToC

Types for BA

types assure the correctness of communication: they specify the topic of conversation (ToC)

each ambient name has associated:

- a local ToC
- a fixed ToC with all its parents

Types for BA

types assure the correctness of communication: they specify the topic of conversation (ToC)

each ambient name has associated:

- a local ToC
- a fixed ToC with all its parents
- a different ToC with each child (according to child's name)

Types for BA

types assure the correctness of communication: they specify the topic of conversation (ToC)

each ambient name has associated:

- a local ToC
- a fixed ToC with all its parents
- a different ToC with each child (according to child's name)

This association is:

Types for BA

types assure the correctness of communication: they specify the topic of conversation (ToC)

each ambient name has associated:

- a local ToC
- a fixed ToC with all its parents
- a different ToC with each child (according to child's name)

This association is:

- global:

Types for BA

types assure the correctness of communication: they specify the topic of conversation (ToC)

each ambient name has associated:

- a local ToC
- a fixed ToC with all its parents
- a different ToC with each child (according to child's name)

This association is:

- global: every ambient knows the ToC's of every other ambient

Types for BA

types assure the correctness of communication: they specify the topic of conversation (ToC)

each ambient name has associated:

- a local ToC
- a fixed ToC with all its parents
- a different ToC with each child (according to child's name)

This association is:

- global: different ambients with the same name share the same ToC's

Types for BA

types assure the correctness of communication: they specify the topic of conversation (ToC)

each ambient name has associated:

- a local ToC
- a fixed ToC with all its parents
- a different ToC with each child (according to child's name)

This association is:

- global:
- static:

Types for BA

types assure the correctness of communication: they specify the topic of conversation (ToC)

each ambient name has associated:

- a local ToC
- a fixed ToC with all its parents
- a different ToC with each child (according to child's name)

This association is:

- global:
- static: the computation does not change the ToC's

Types for BA

types assure the correctness of communication: they specify the topic of conversation (ToC)

each ambient name has associated:

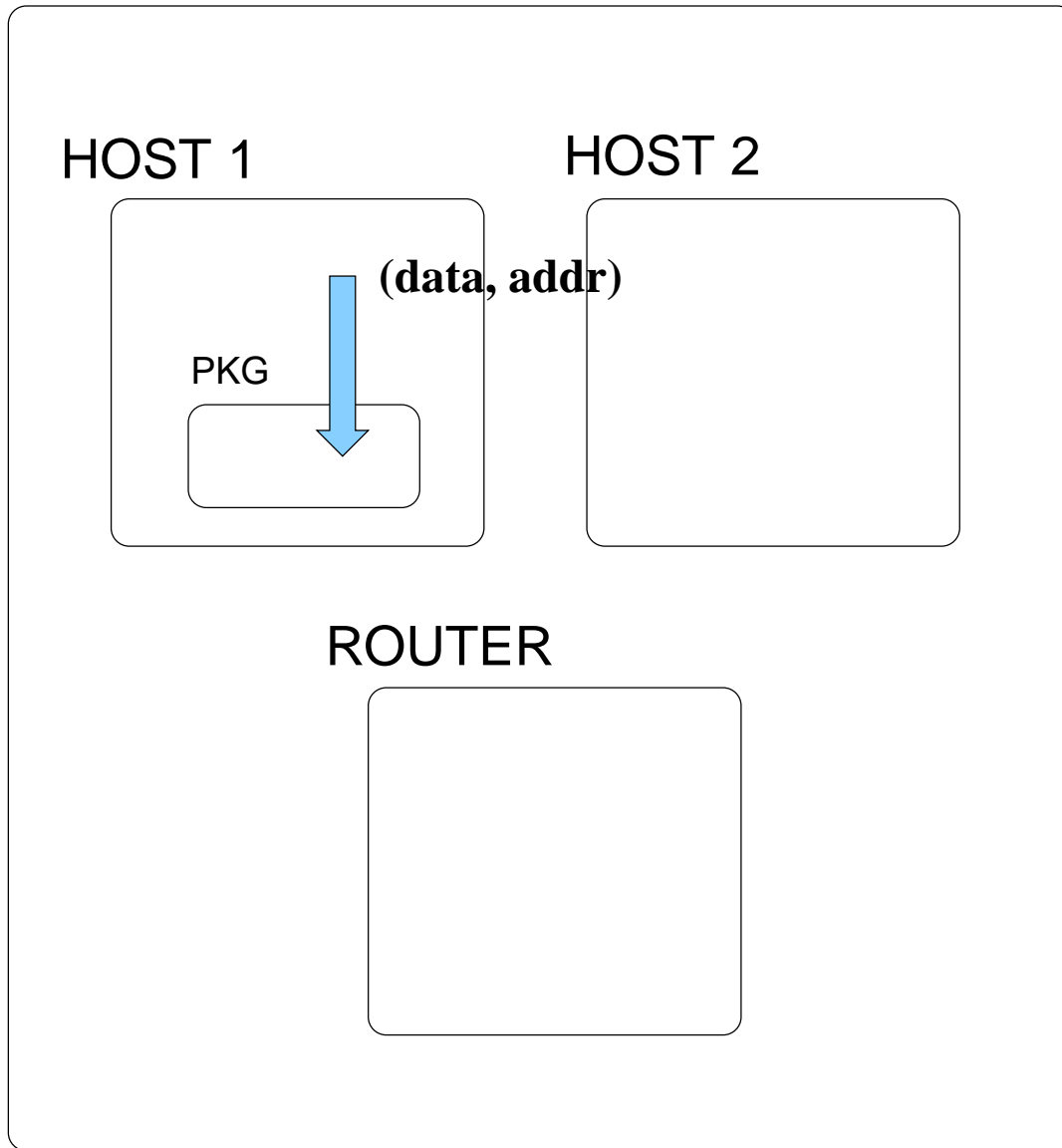
- a local ToC
- a fixed ToC with all its parents
- a different ToC with each child (according to child's name)

This association is:

- global:
- static:

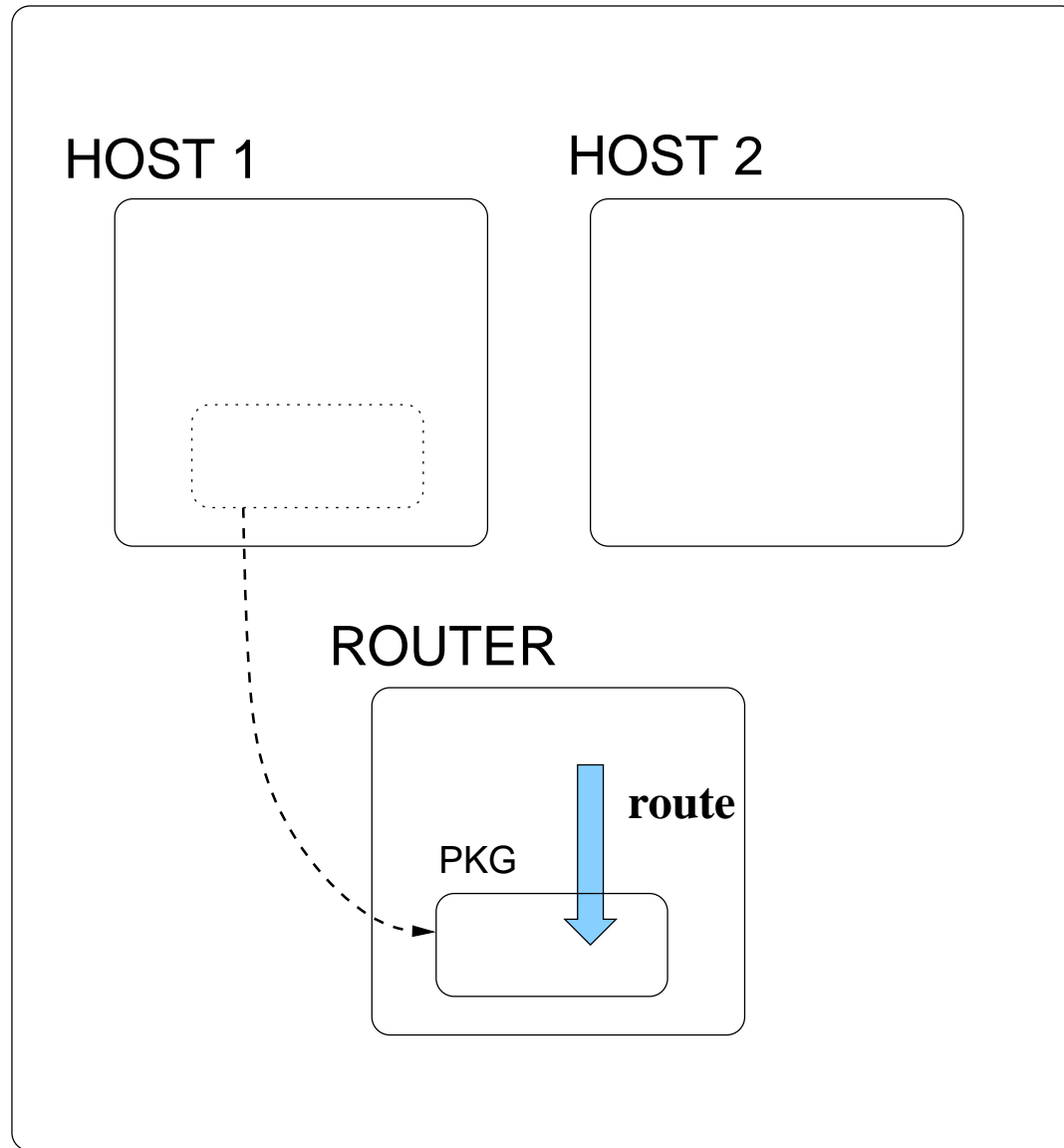
A motivating example

NETWORK



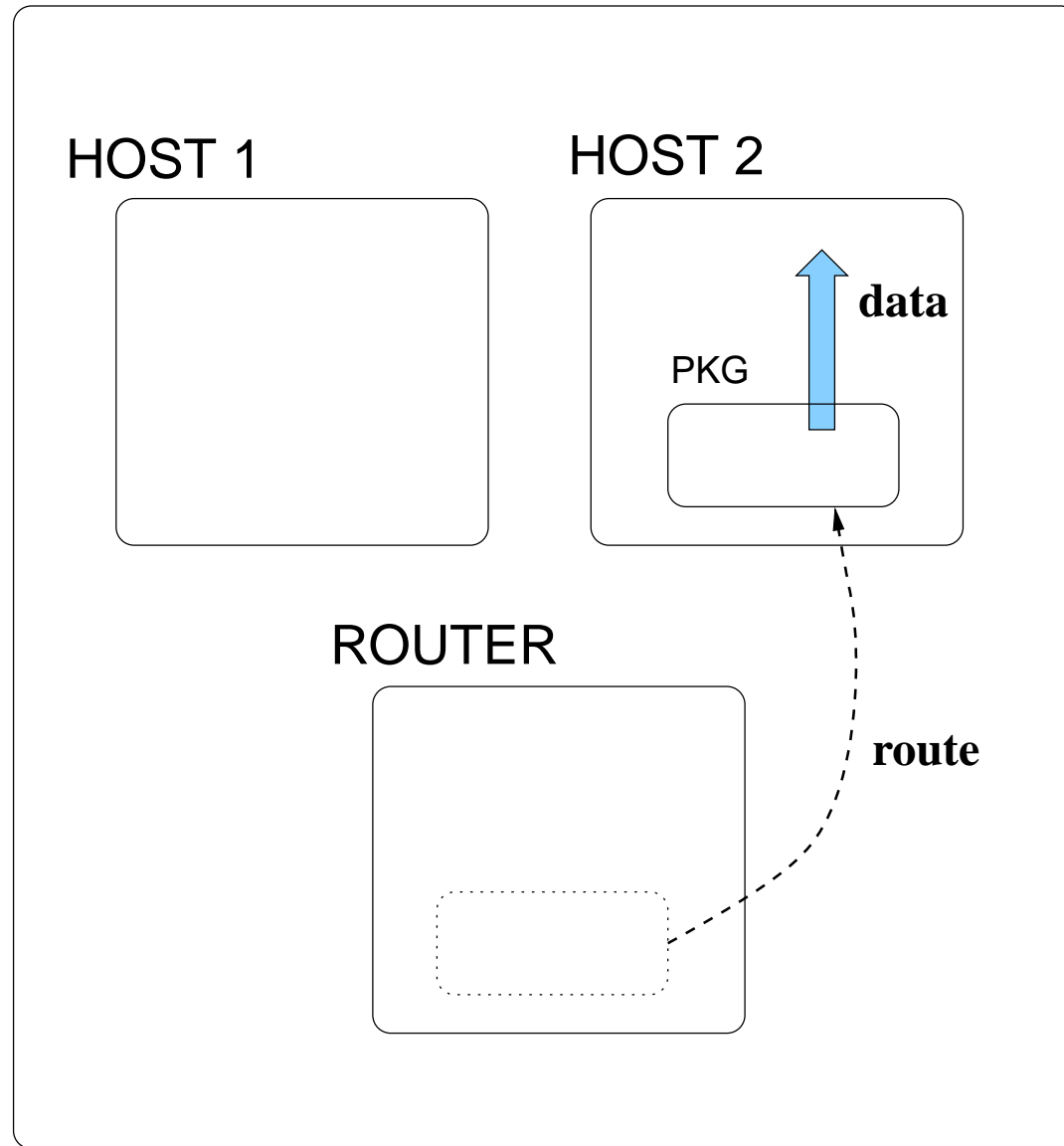
A motivating example

NETWORK



A motivating example

NETWORK



BACI

Boxed Ambients with Communication Interfaces

BACI

Boxed Ambients with Communication Interfaces

- each ambient comes equipped with a communication interface:

BACI

Boxed Ambients with Communication Interfaces

- each ambient comes equipped with a communication interface:
 - a communication port c ;

BACI

Boxed Ambients with Communication Interfaces

- each ambient comes equipped with a communication interface:
 - a communication port c ;
 - a local view:

BACI

Boxed Ambients with Communication Interfaces

- each ambient comes equipped with a communication interface:
 - a communication port c ;
 - a local view:
 - giving the local ToC;

BACI

Boxed Ambients with Communication Interfaces

- each ambient comes equipped with a communication interface:
 - a communication port c ;
 - a local view:
 - giving the local ToC;
 - associating ToC's with parent and child ports;

BACI

Boxed Ambients with Communication Interfaces

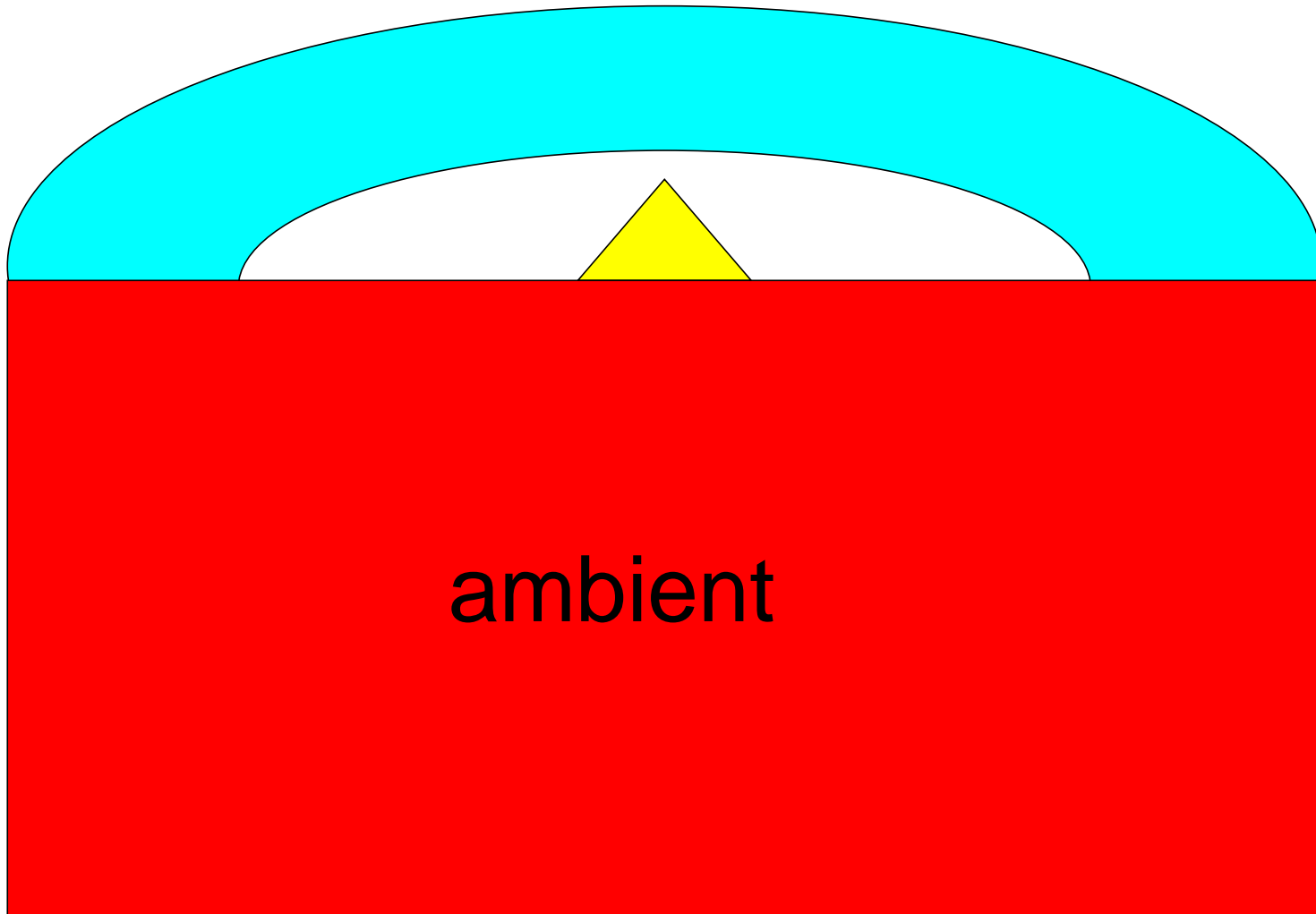
- each ambient comes equipped with a communication interface:
 - a communication port c ;
 - a local view:
 - giving the local ToC;
 - associating ToC's with parent and child ports;
- all associations are local;

BACI

Boxed Ambients with Communication Interfaces

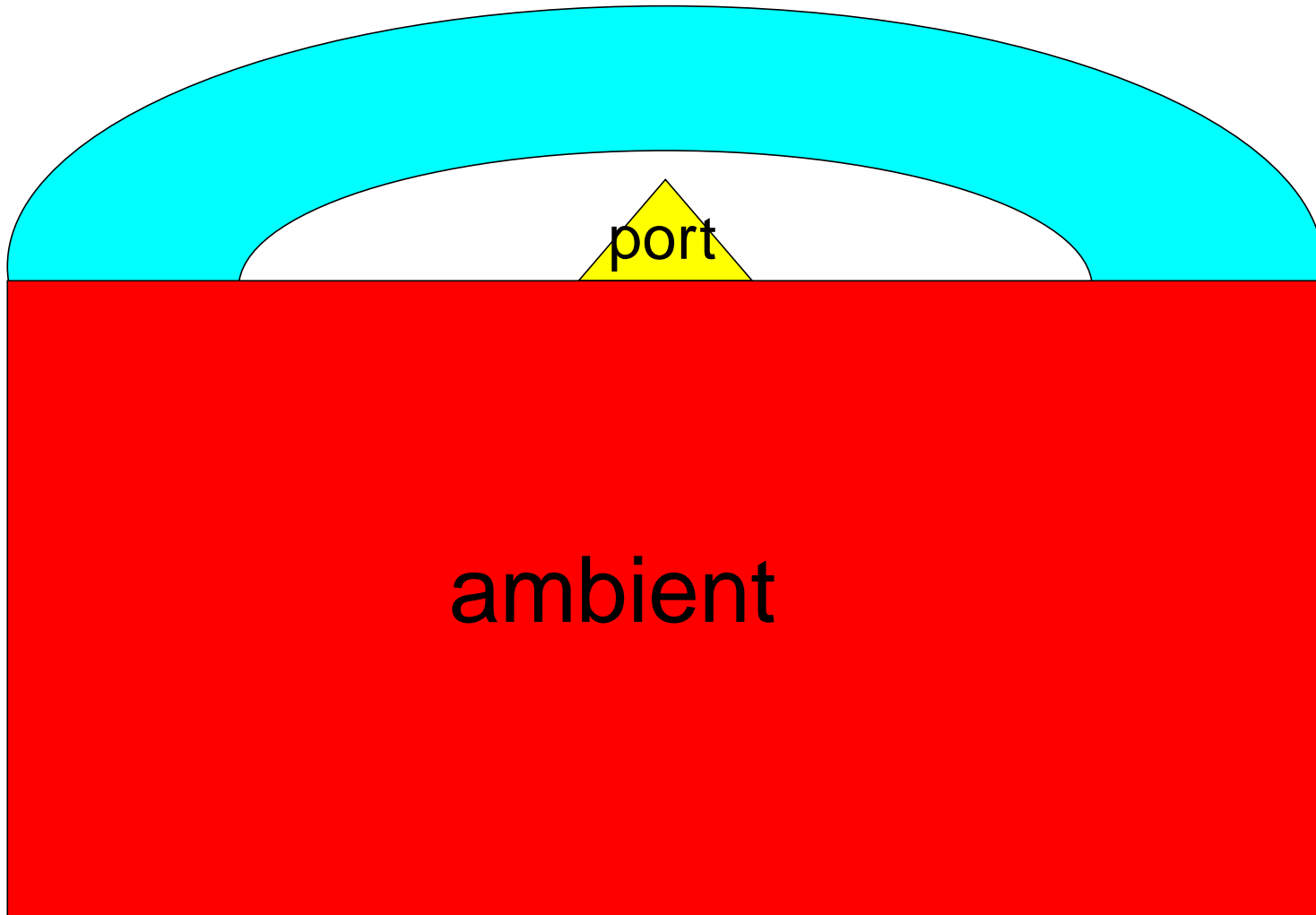
- each ambient comes equipped with a communication interface:
 - a communication port c ;
 - a local view:
 - giving the local ToC;
 - associating ToC's with parent and child ports;
- all associations are local;
- the local views are dynamic, i.e. they increase with ambient movements.

BACI



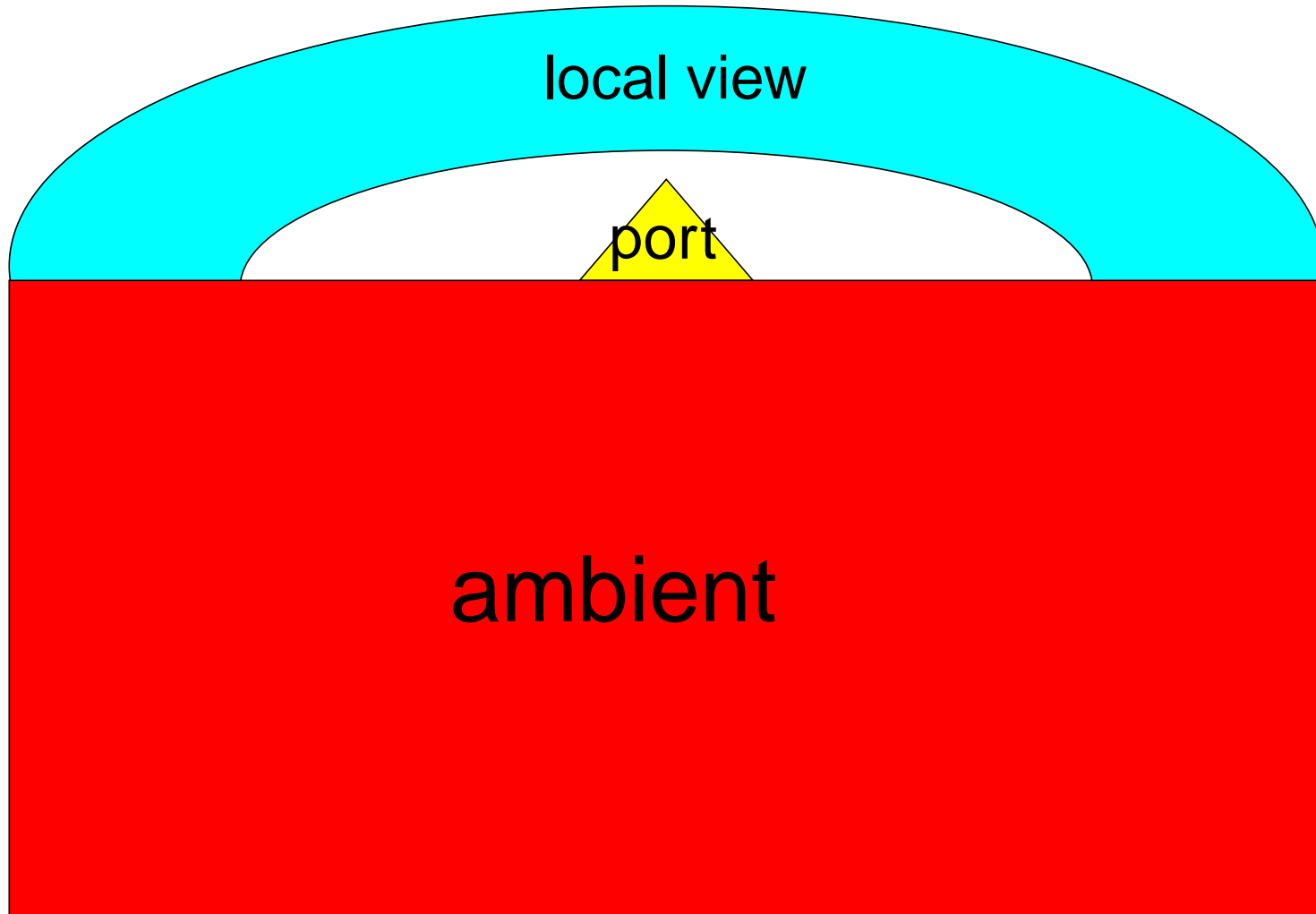
a BACI ambient comes equipped with

BACI



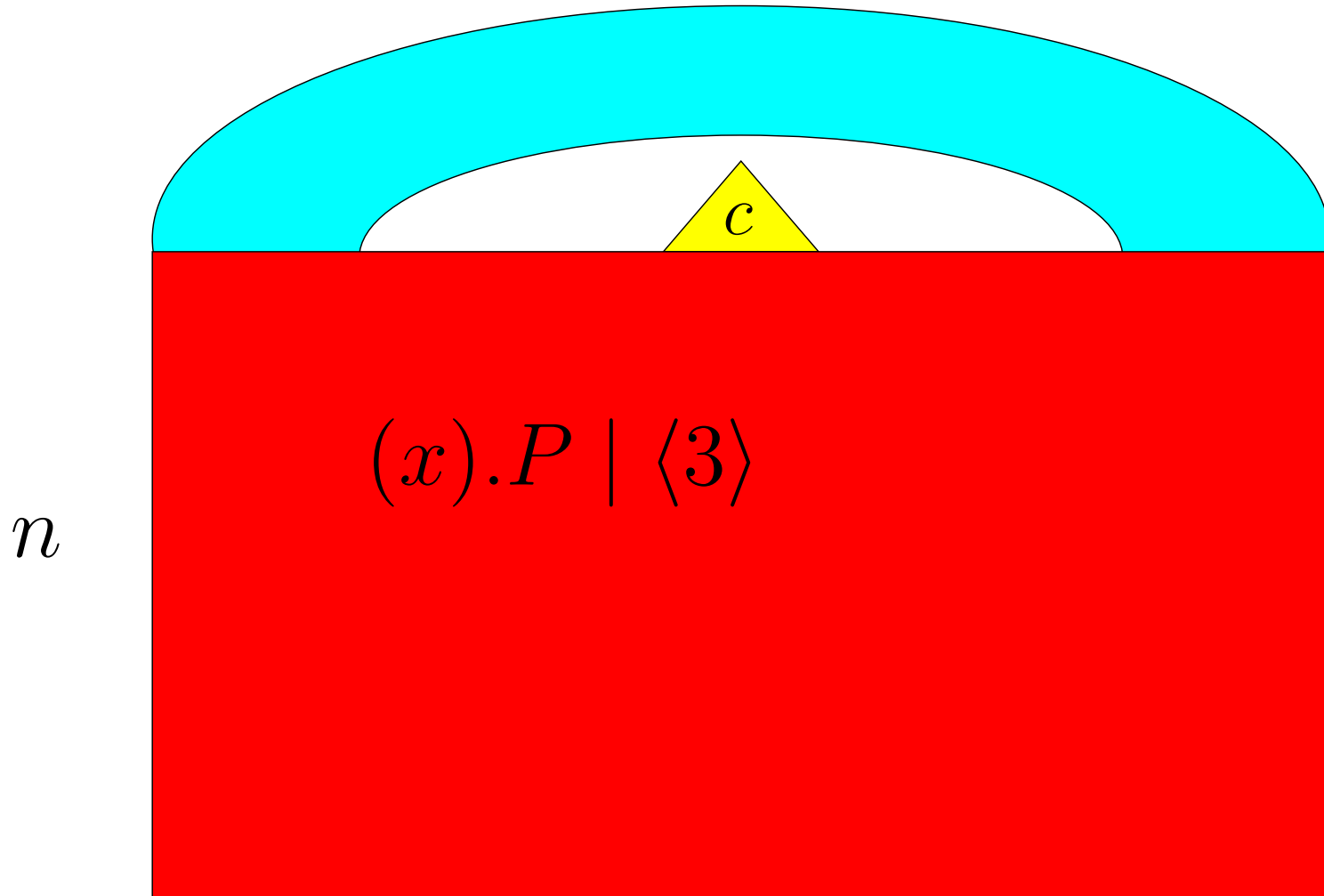
a BACI ambient comes equipped with a port

BACI

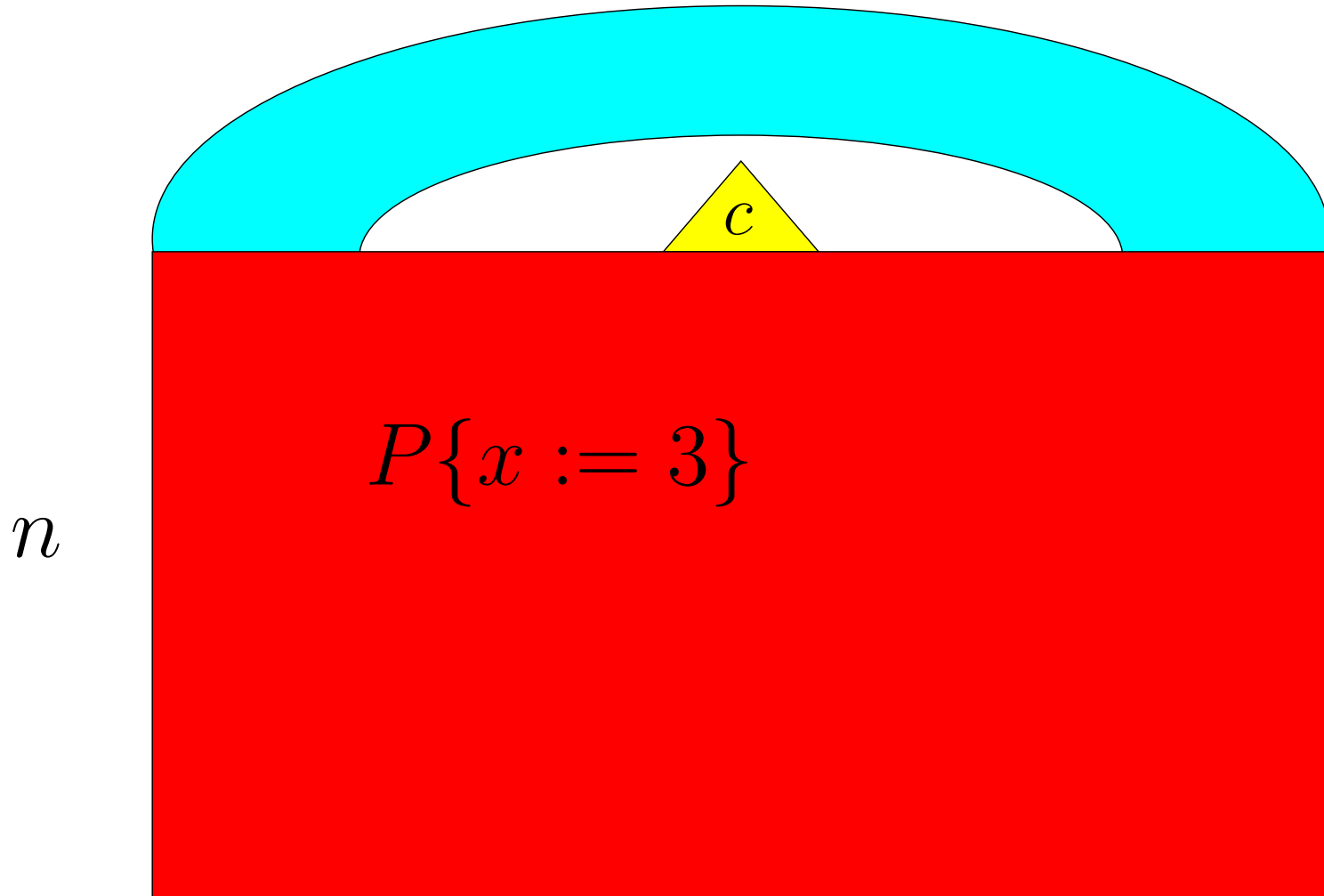


a BACI ambient comes equipped with a port and a local view

Communication with ports: local



Communication with ports: local



Communication with ports: local

$$n[||c||(x).P \mid \langle 3 \rangle \cdots]$$

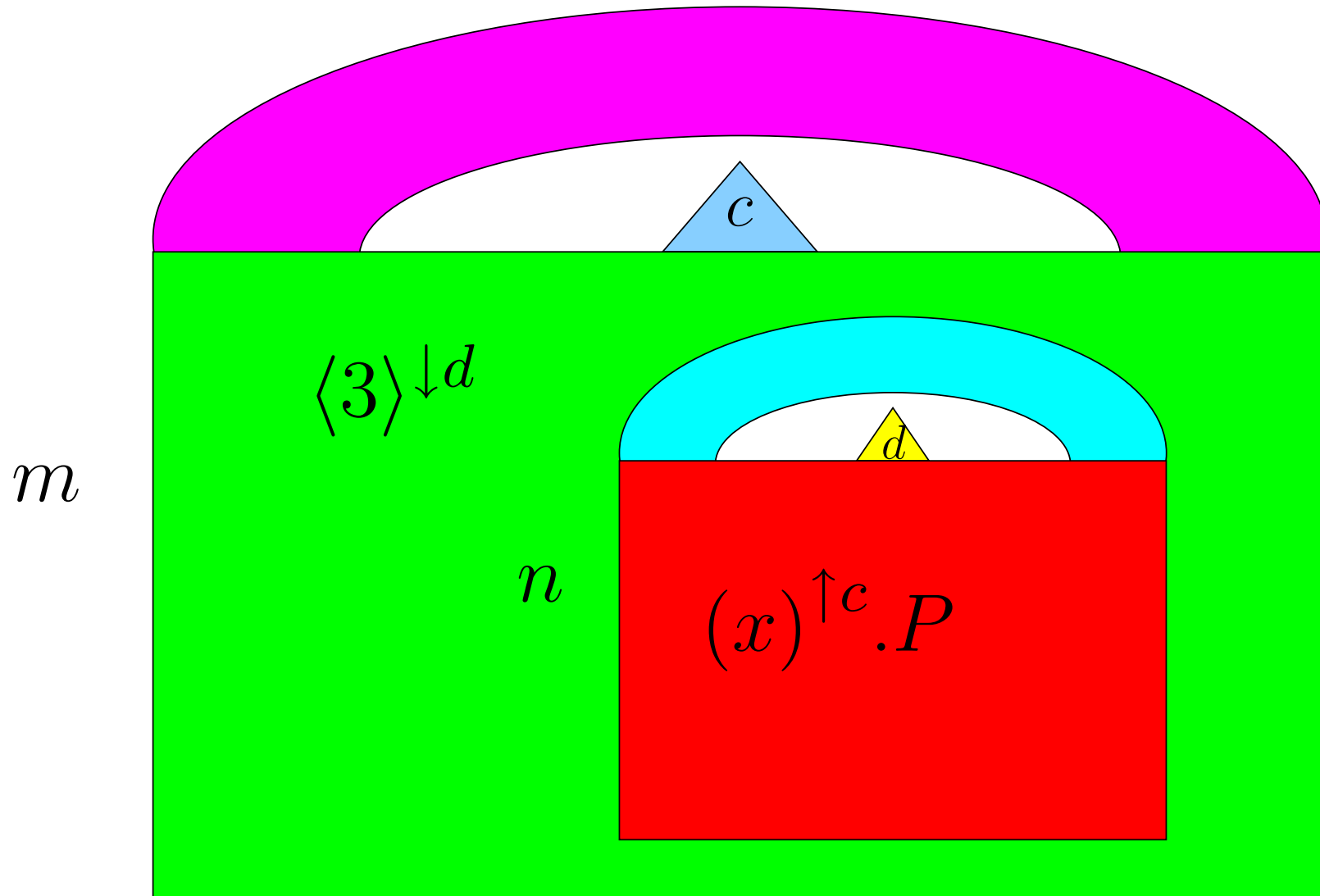
Communication with ports: local

$$n[||c||(x).P \mid \langle 3 \rangle \cdots]$$

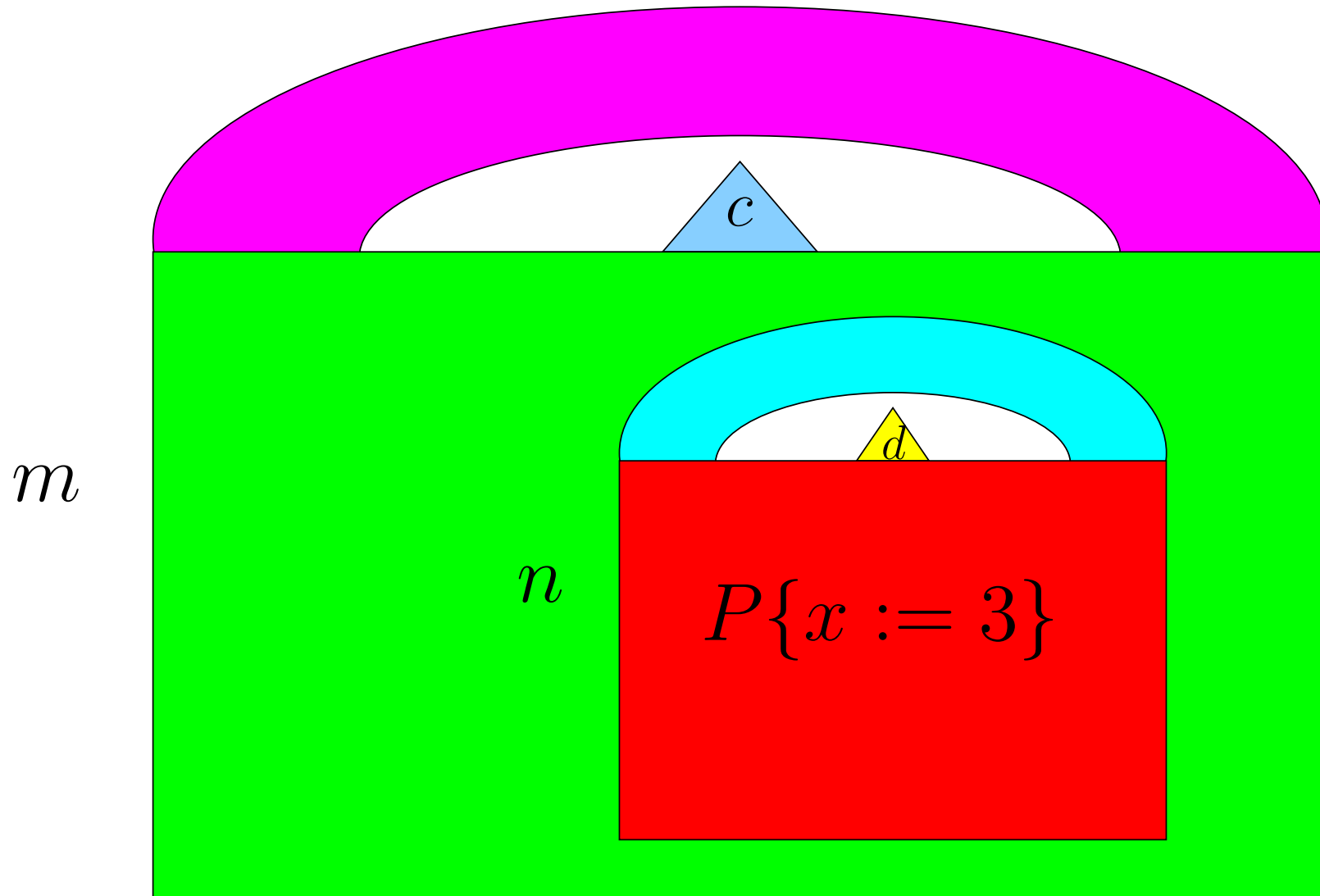
→

$$n[||c||P\{x := 3\} \cdots]$$

Communication with ports: parent-child



Communication with ports: parent-child



Communication with ports: parent-child

$$m[||c|| \cdot \cdot \cdot \langle 3 \rangle \downarrow^d \mid n[||d|| (x) \uparrow^c . P \cdot \cdot \cdot]]$$

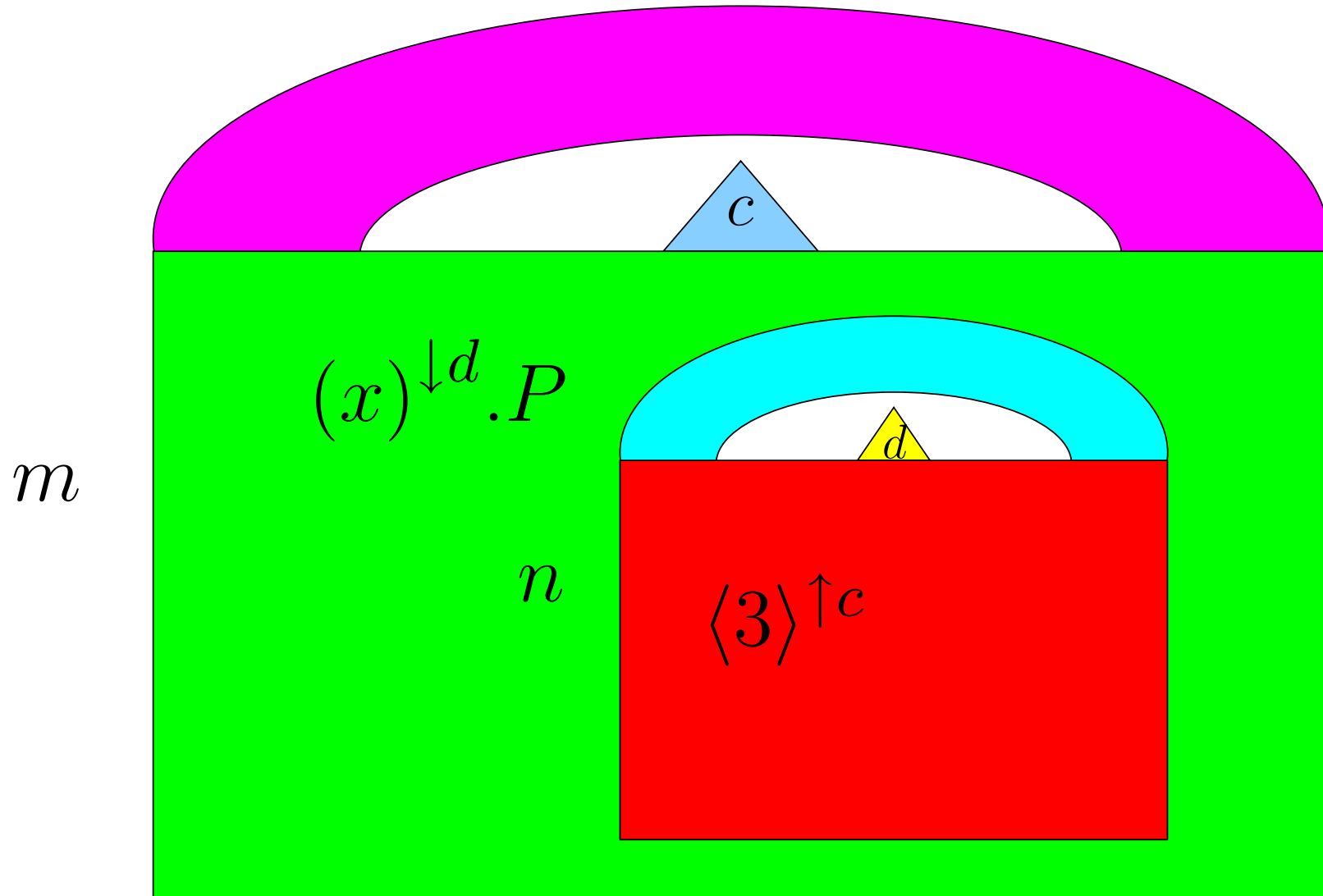
Communication with ports: parent-child

$$m[||c|| \cdot \cdot \cdot \langle 3 \rangle \downarrow^d \mid n[||d|| (x) \uparrow^c . P \cdot \cdot \cdot]]$$

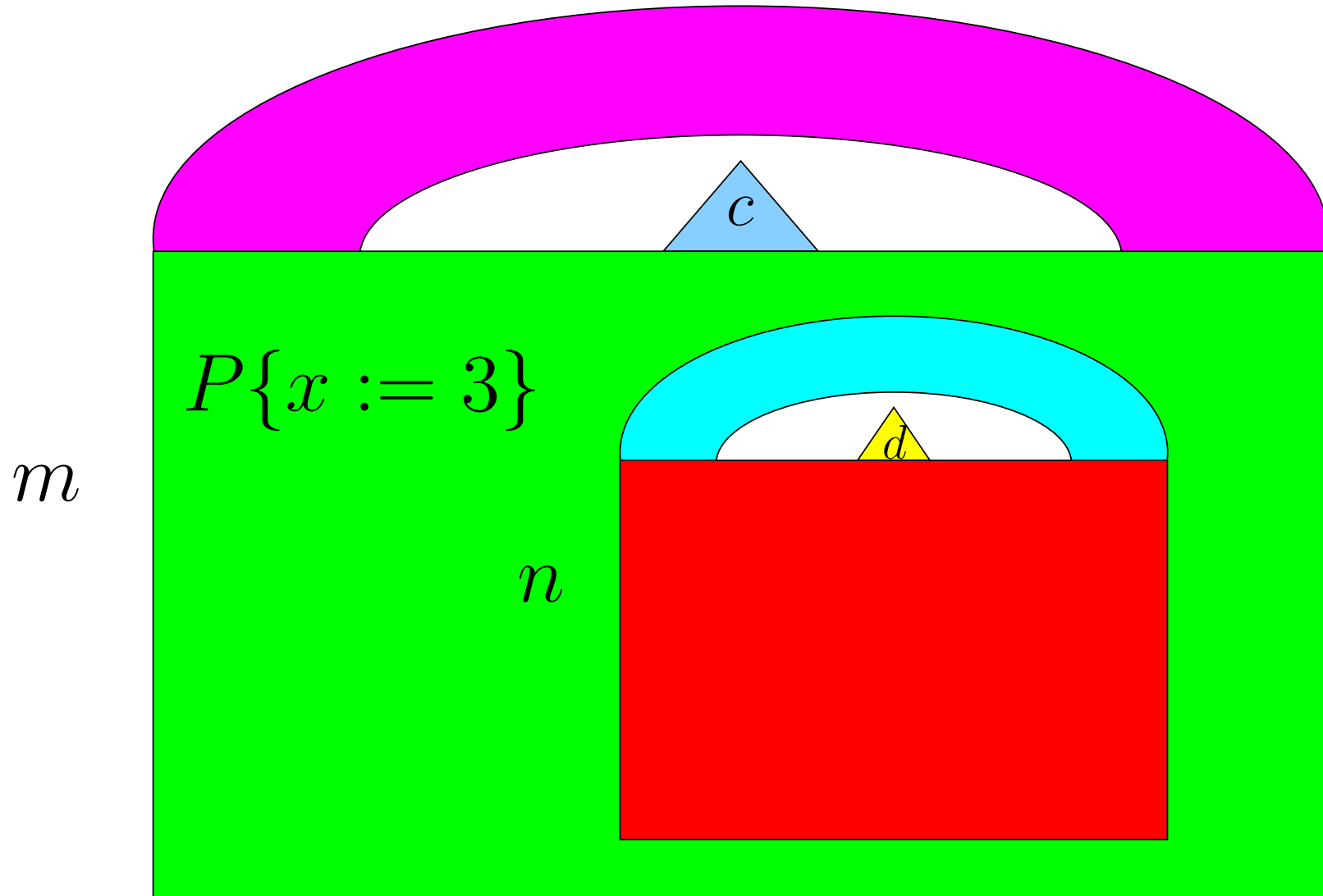
→

$$m[||c|| \cdot \cdot \cdot n[||d|| P \{x := 3\} \cdot \cdot \cdot]]$$

Communication with ports: child-parent



Communication with ports: child-parent



Communication with ports: child-parent

$$m[||c|| \cdot \cdot \cdot (x) \downarrow^d . P \mid n[||d|| \langle 3 \rangle \uparrow^c \cdot \cdot \cdot]]$$

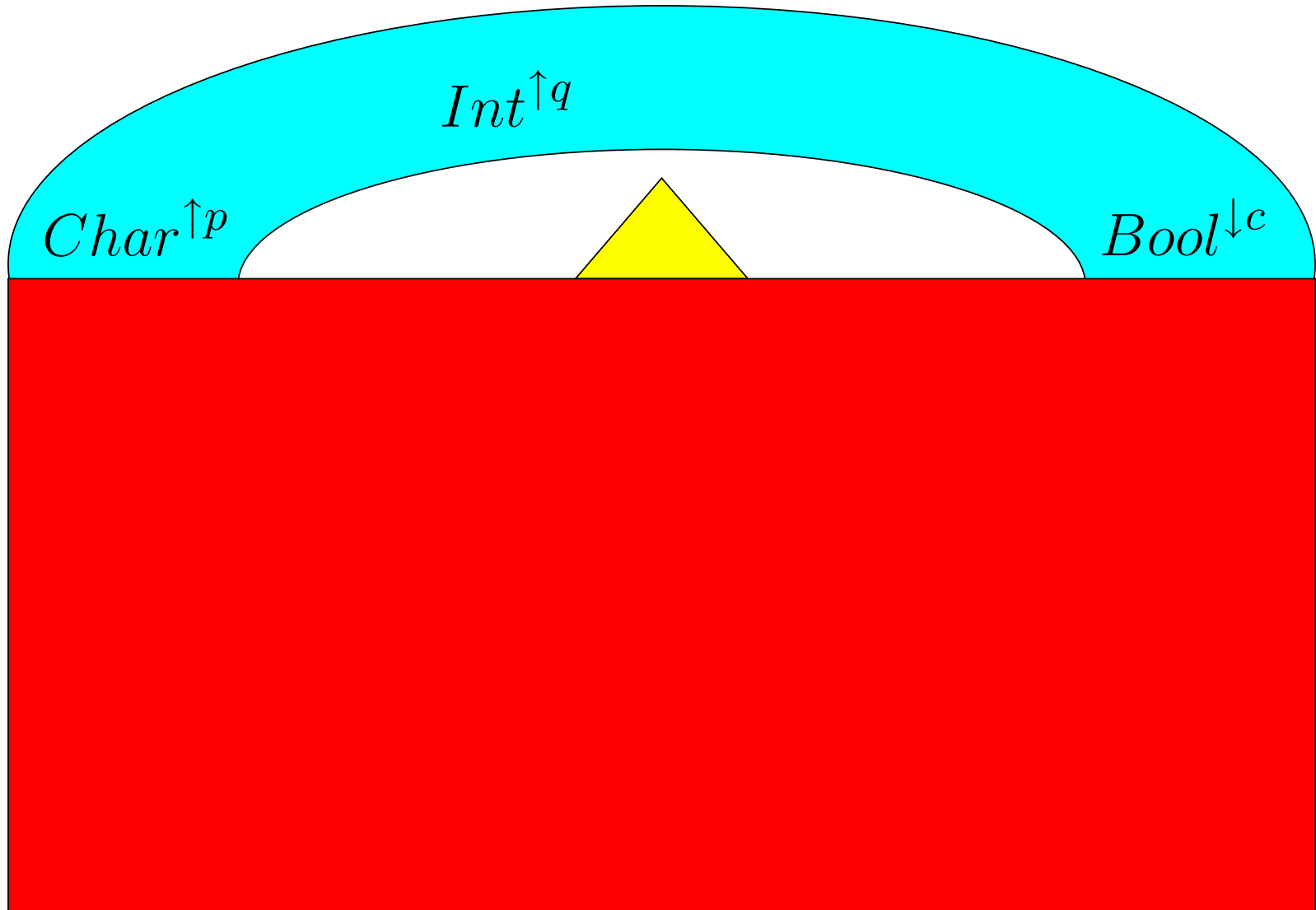
Communication with ports: child-parent

$$m[||c|| \cdot \cdot \cdot (x) \downarrow^d . P \mid n[||d|| \langle 3 \rangle \uparrow^c \cdot \cdot \cdot]]$$

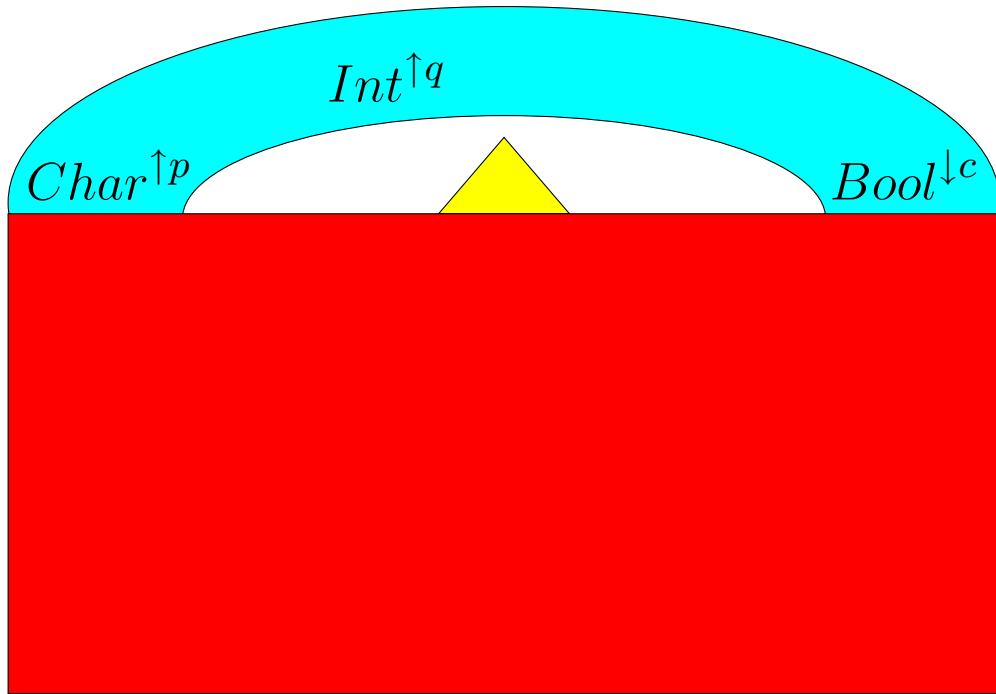
→

$$m[||c|| \cdot \cdot \cdot P\{x := 3\} \mid n[||d|| \cdot \cdot \cdot]]$$

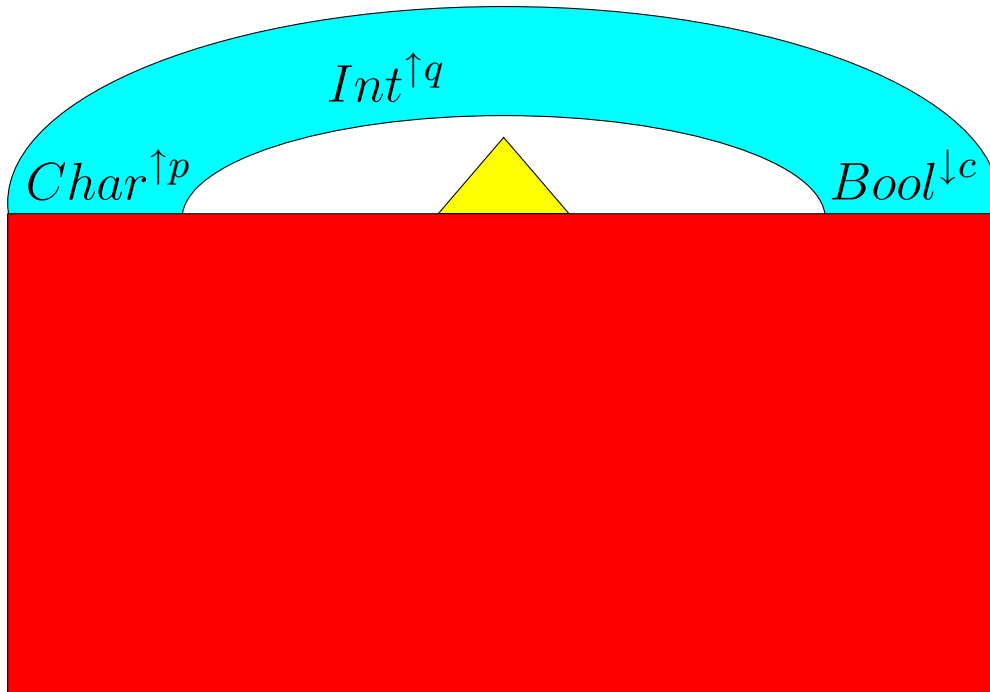
Local Views



Local Views



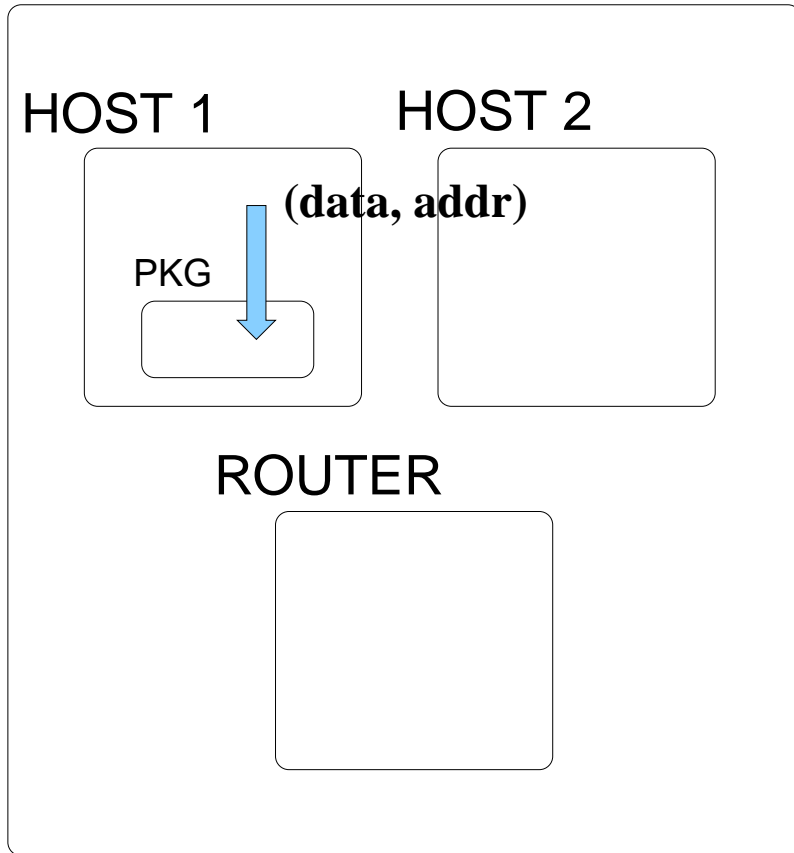
Local Views



$[\{Char^{\uparrow p}, Int^{\uparrow q}, Bool^{\downarrow c}\} ||| \dots]$

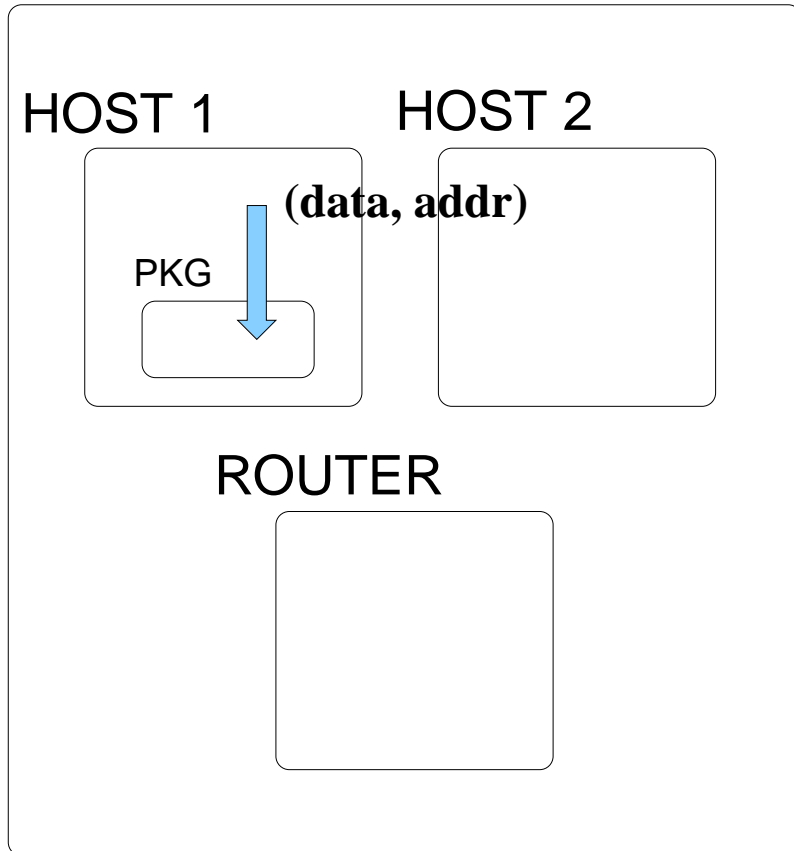
the example revisited

NETWORK



the example revisited

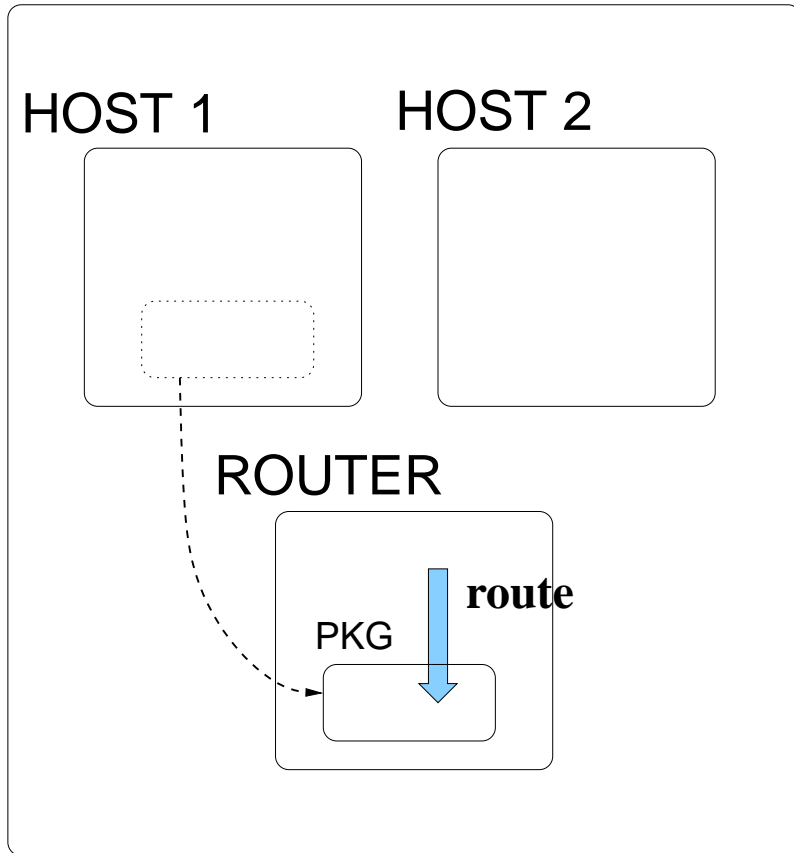
NETWORK



$\text{PKG}[\{(\text{data, addr})^{\uparrow_{\text{CHOST1}}},$
 $\text{route}^{\uparrow_{\text{CROUTER}}}, \text{data}^{\uparrow_{\text{CHOST2}}}\} ||| \dots]$

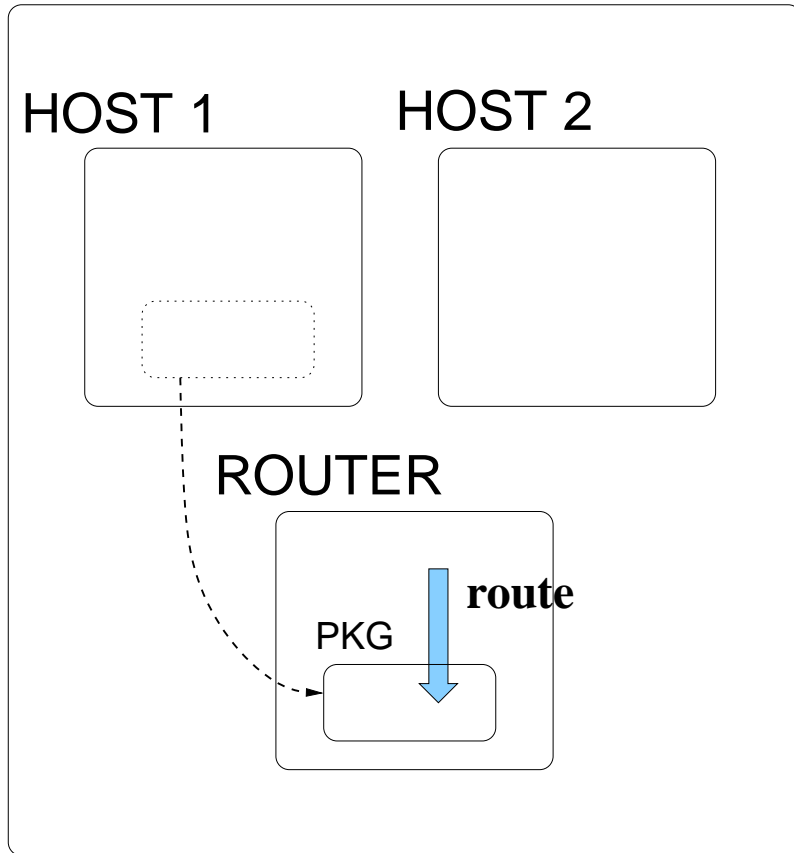
the example revisited

NETWORK



the example revisited

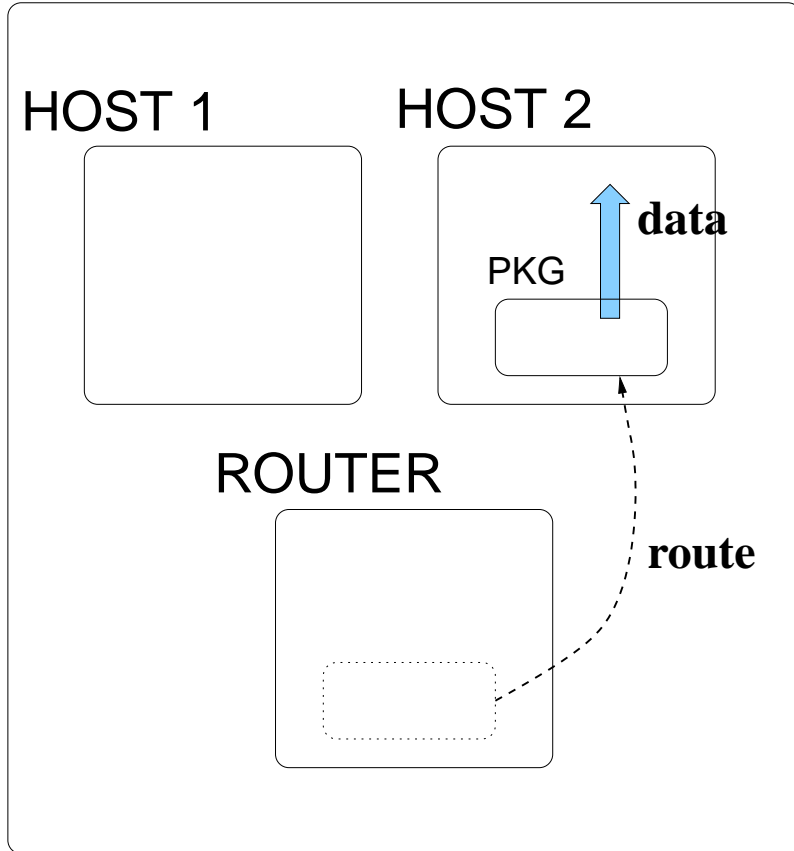
NETWORK



$\text{PKG}[\{(\text{data}, \text{addr})^{\uparrow_{\text{CHOST1}}},$
 $\text{route}^{\uparrow_{\text{CROUTER}}}, \text{data}^{\uparrow_{\text{CHOST2}}}\} ||| \dots]$

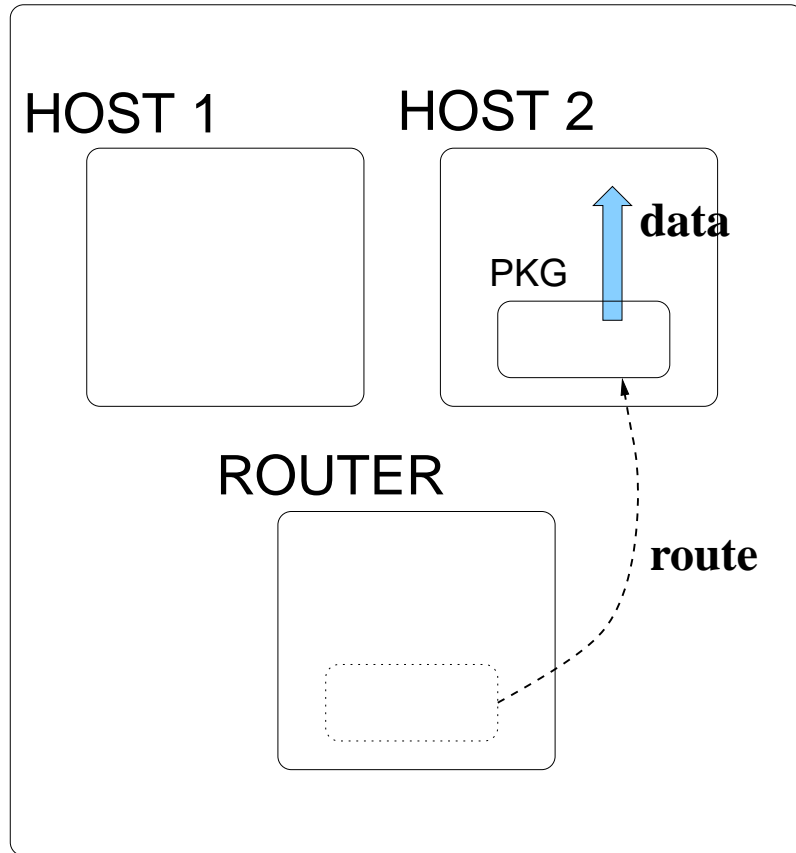
the example revisited

NETWORK



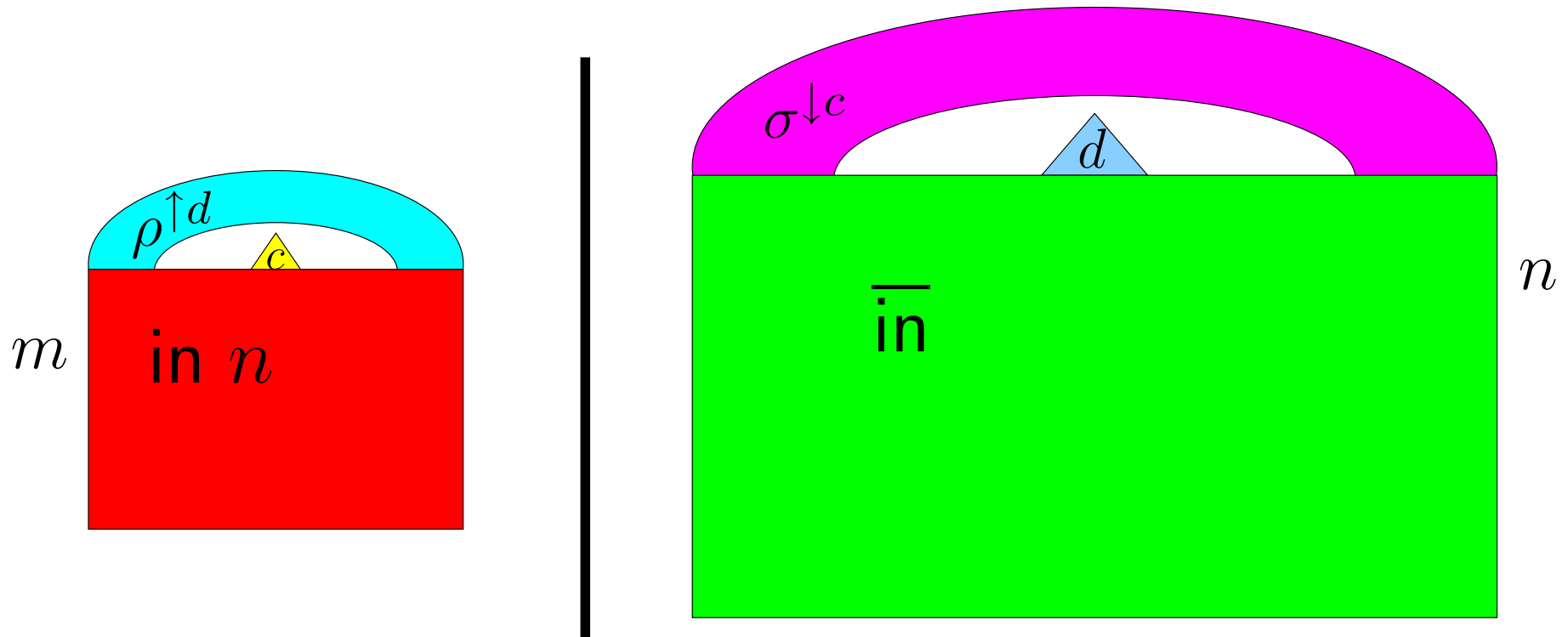
the example revisited

NETWORK



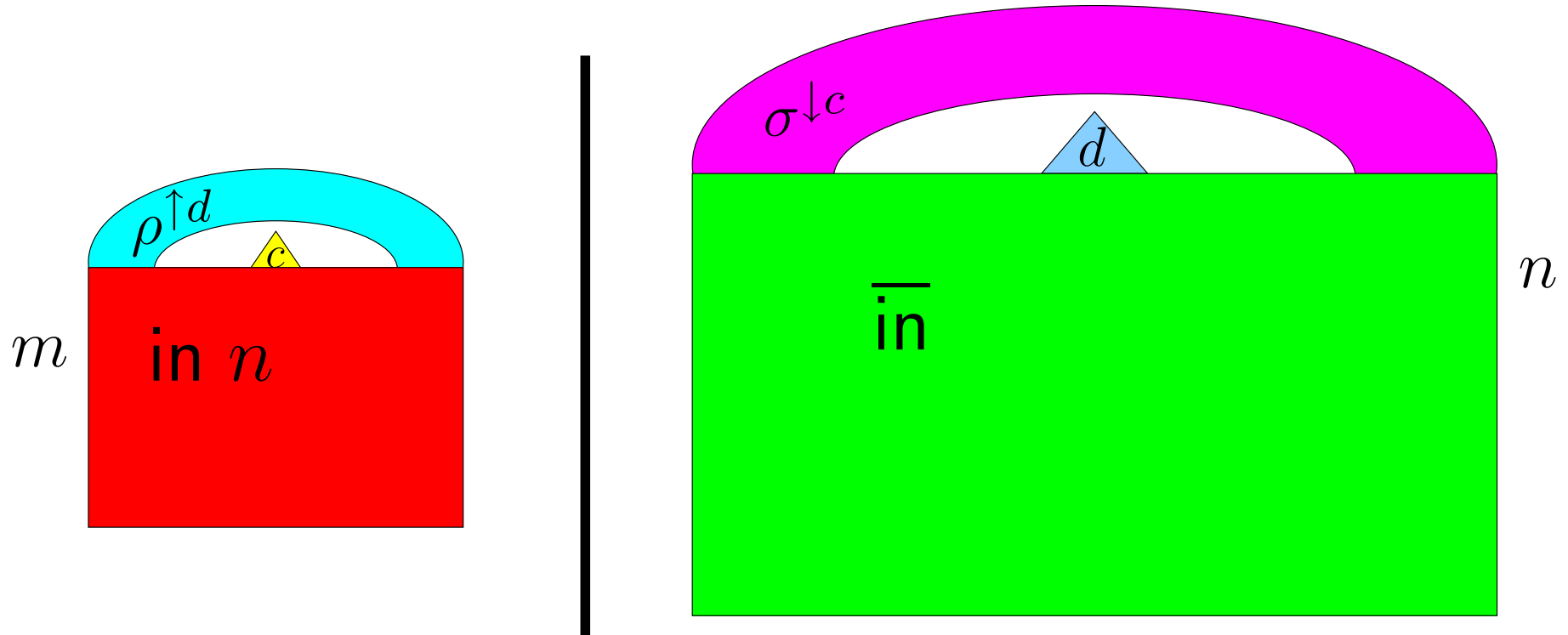
$\text{PKG}[\{(\mathbf{data}, \mathbf{addr})^{\uparrow_{\text{CHOST1}}},$
 $\mathbf{route}^{\uparrow_{\text{CROUTER}}}, \mathbf{data}^{\uparrow_{\text{CHOST2}}}\} ||| \dots]$

Local views control movement: IN



if either $\rho = \text{shh}$ or $\rho = \sigma$

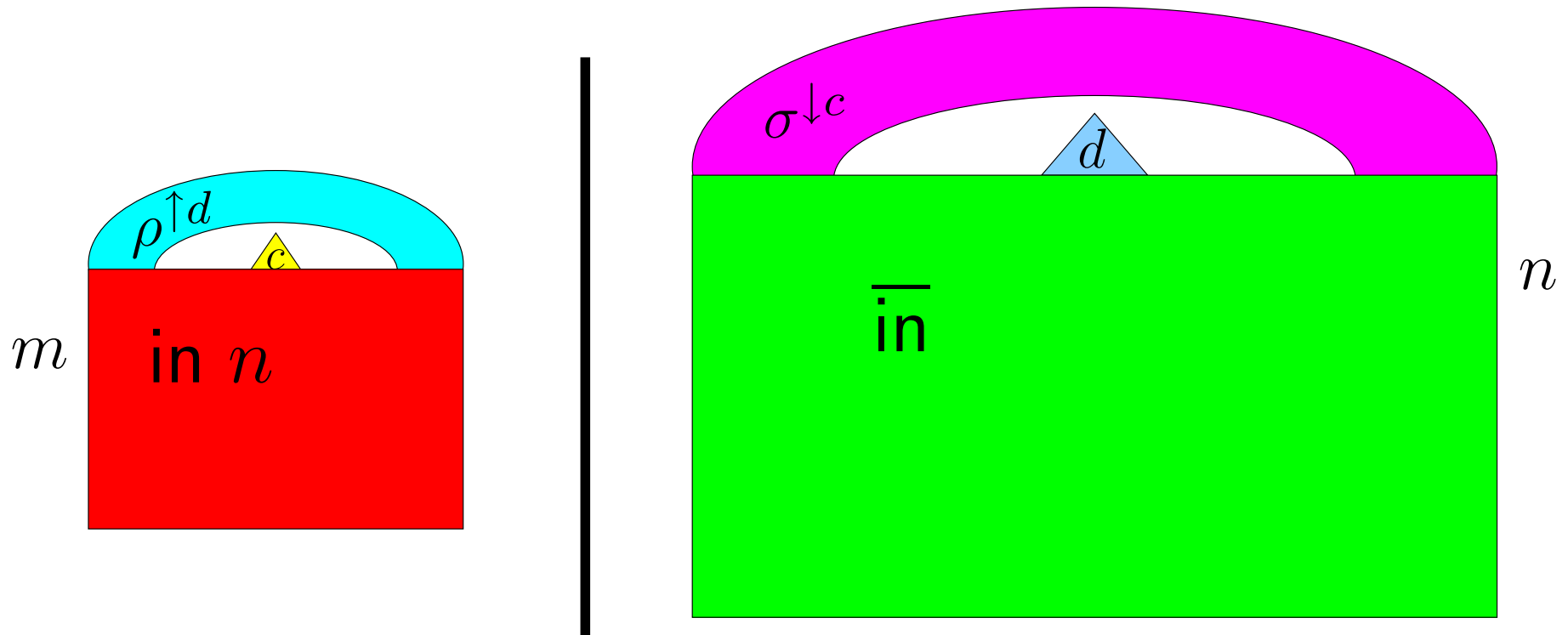
Local views control movement: IN



if either $\rho = \text{shh}$ or $\rho = \sigma$

a silent child can always enter

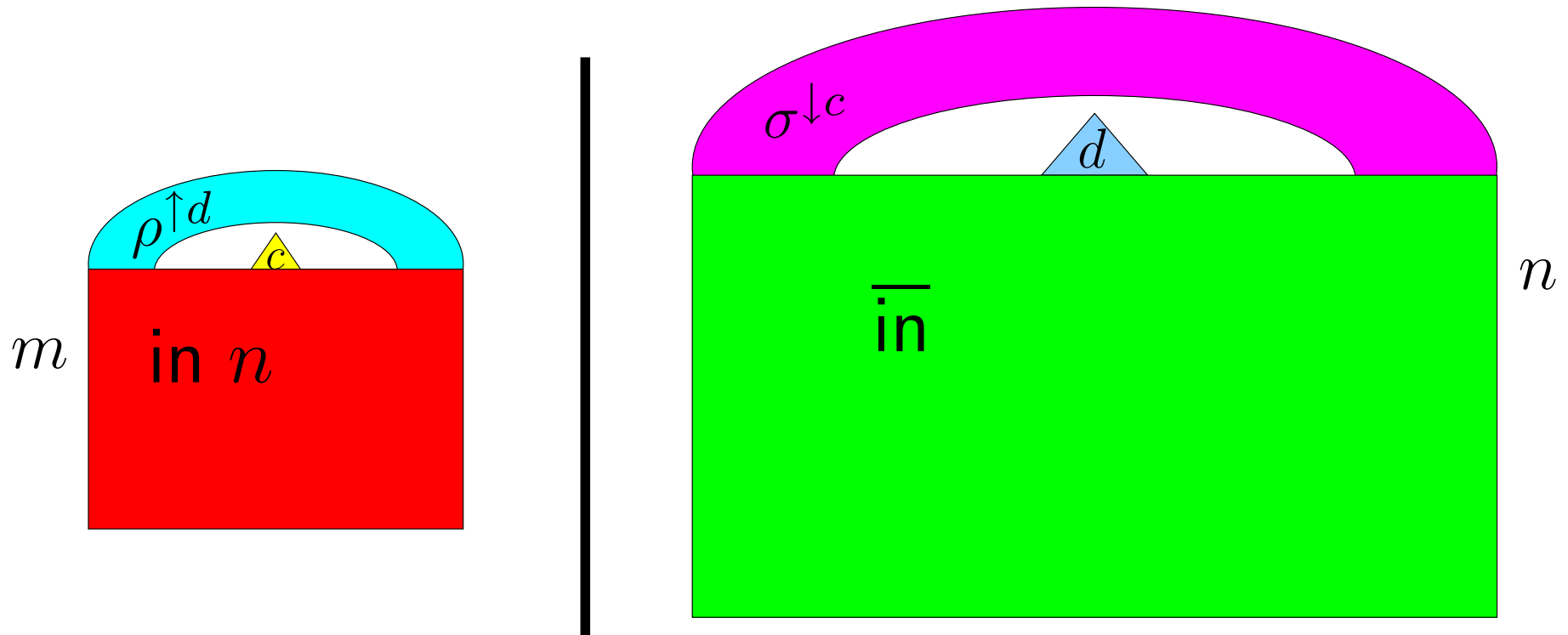
Local views control movement: IN



if either $\rho = \text{shh}$ or $\rho = \sigma$

parent and child agree on the ToC

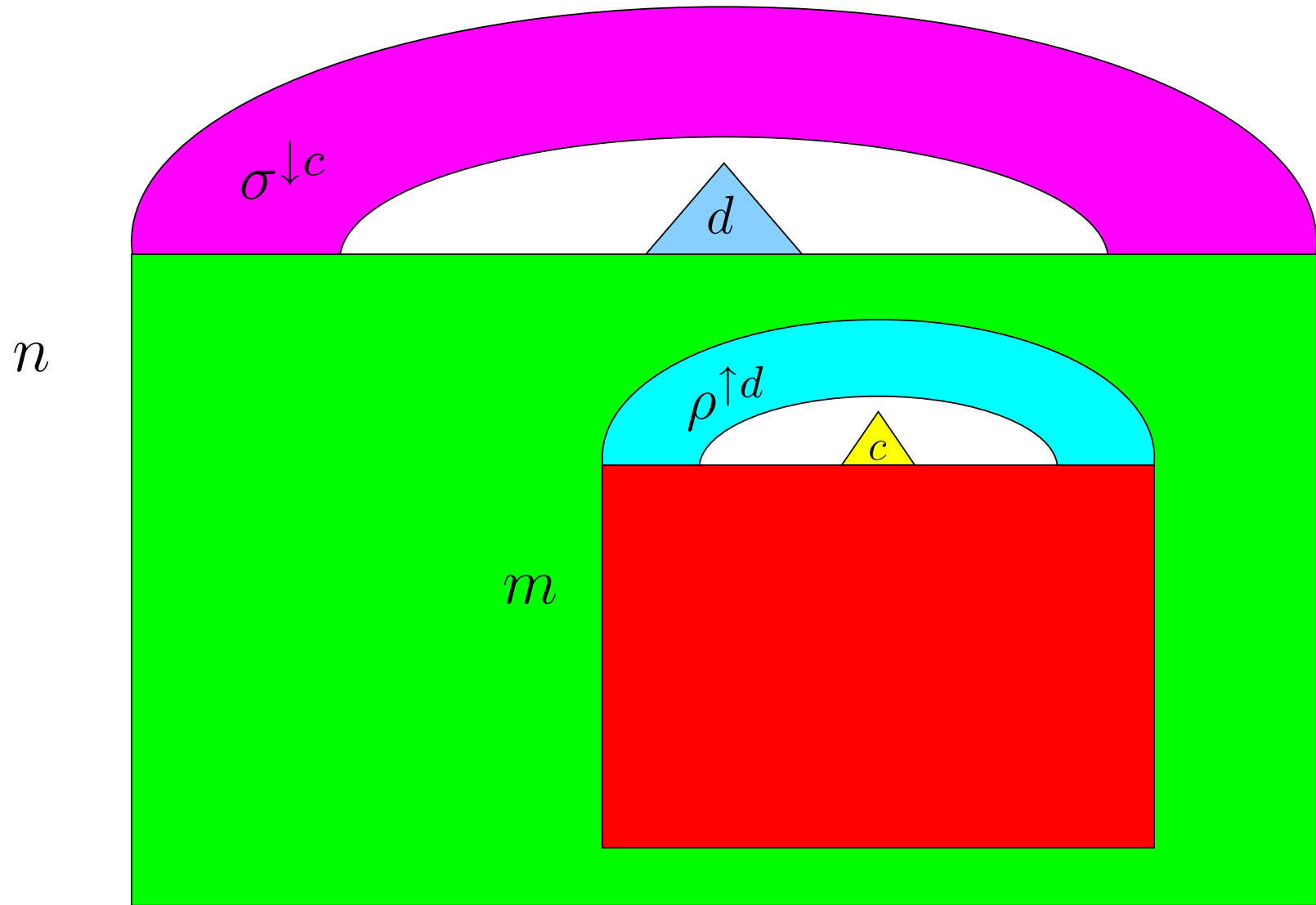
Local views control movement: IN



if either $\rho = shh$ or $\rho = \sigma$

parents must listen to children (not vice versa)

Local views control movement: IN



Local views control movement: IN

$$m[\{\rho^{\uparrow d}, \dots\} || c || \text{in } n \dots] \mid n[\{\sigma^{\downarrow c}, \dots\} || d || \overline{\text{in}} \dots]$$

Local views control movement: IN

$$m[\{\rho^{\uparrow d}, \dots\} \parallel c \parallel \text{in } n \dots] \mid n[\{\sigma^{\downarrow c}, \dots\} \parallel d \parallel \overline{\text{in}} \dots]$$

if either $\rho = \text{shh}$ or $\rho = \sigma$

Local views control movement: IN

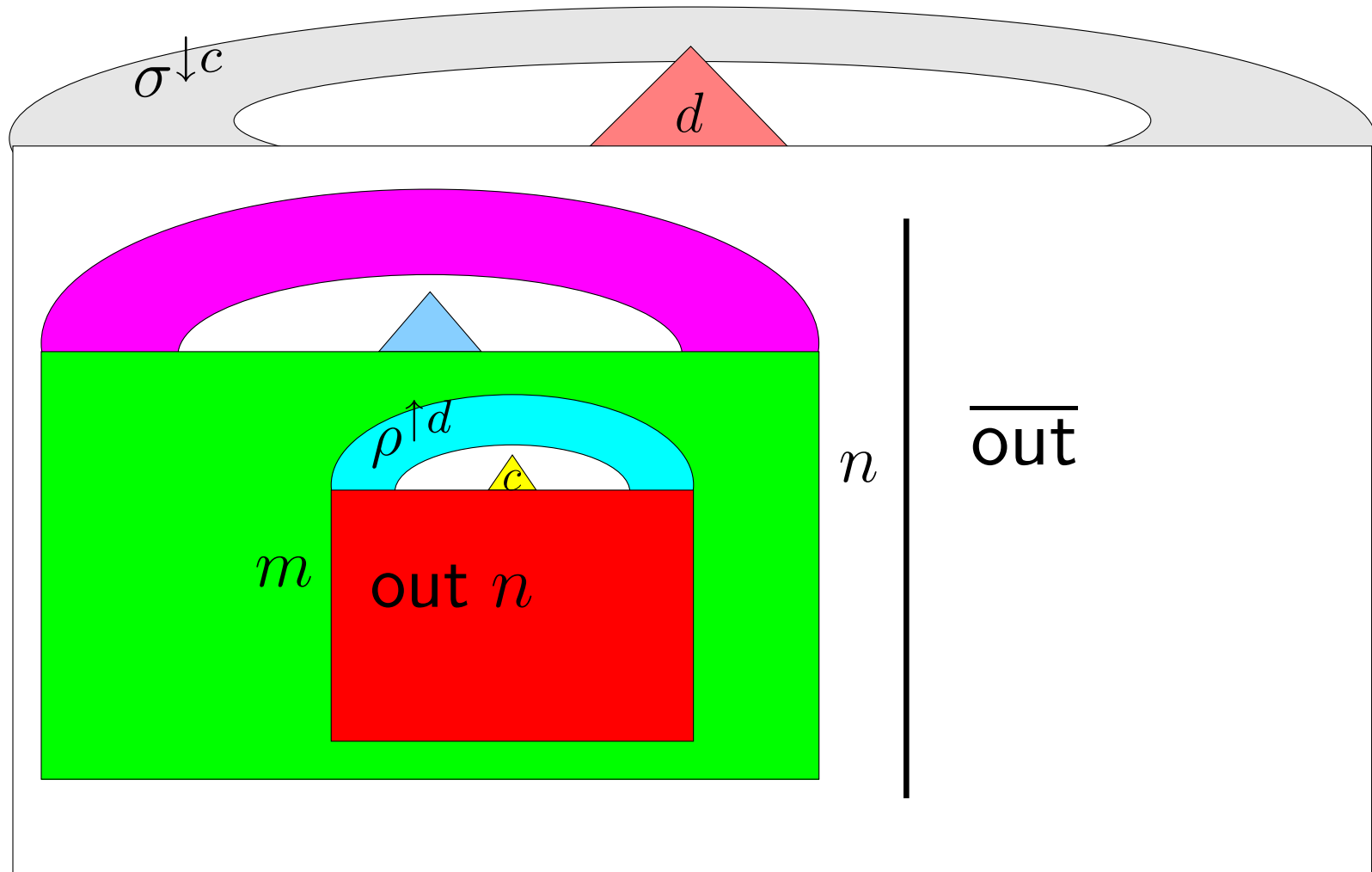
$$m[\{\rho^{\uparrow d}, \dots\} \parallel c \parallel \text{in } n \dots] \mid n[\{\sigma^{\downarrow c}, \dots\} \parallel d \parallel \overline{\text{in}} \dots]$$

if either $\rho = \text{shh}$ or $\rho = \sigma$

→

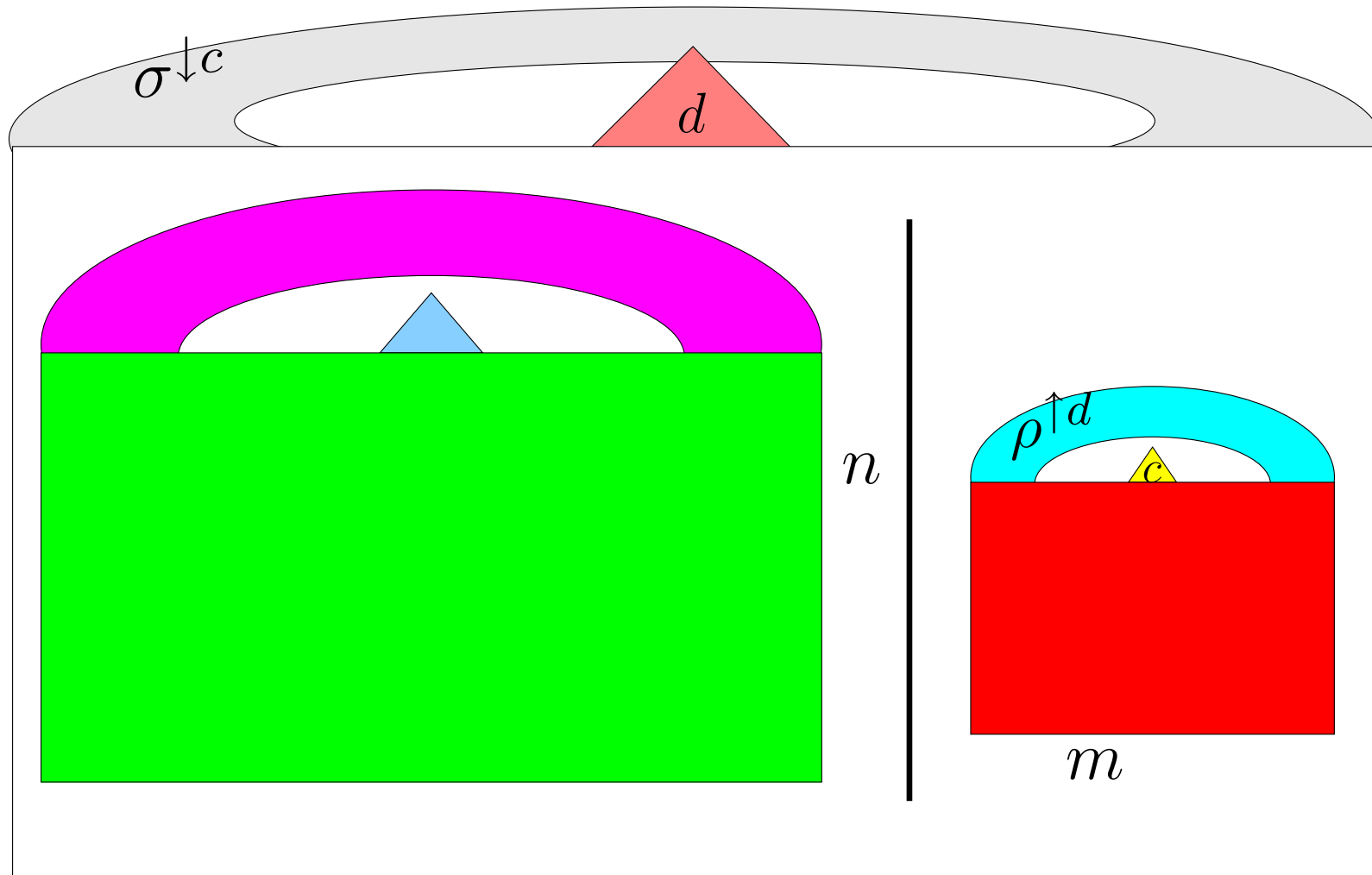
$$n[\{\sigma^{\downarrow c}, \dots\} \parallel d \parallel m[\{\rho^{\uparrow d}, \dots\} \parallel c \parallel \dots] \dots]$$

Local views control movement: OUT



if either $\rho = shh$ or $\rho = \sigma$

Local views control movement: OUT



Local views control movement: OUT

$$\{\sigma^{\downarrow c}, \dots\} \parallel d \parallel n[\dots m[\{\rho^{\uparrow d}, \dots\} \parallel c \parallel \text{out } n \dots] \dots] \mid \overline{\text{out}}$$

Local views control movement: OUT

$\{\sigma^{\downarrow c}, \dots\} \parallel d \parallel n[\dots m[\{\rho^{\uparrow d}, \dots\} \parallel c \parallel \text{out } n \dots] \dots] \mid \overline{\text{out}}$
if either $\rho = \text{shh}$ or $\rho = \sigma$

Local views control movement: OUT

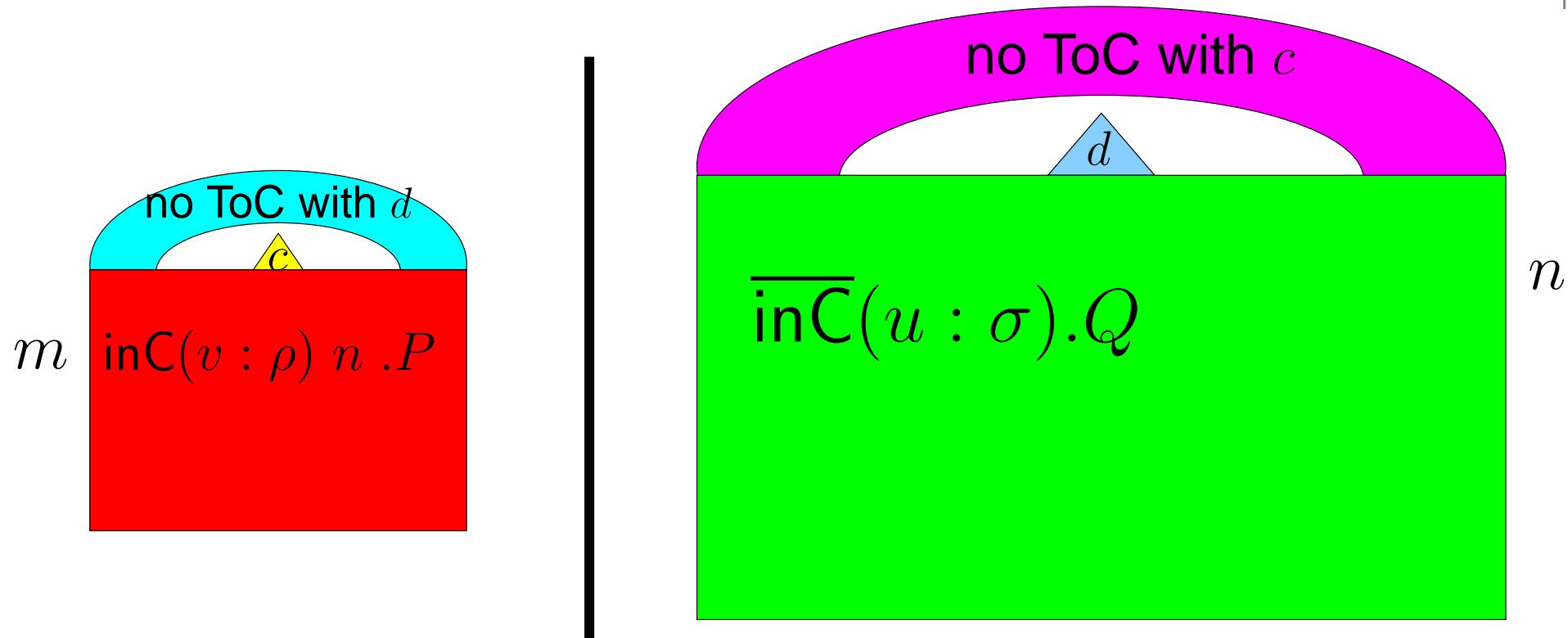
$$\{\sigma^{\downarrow c}, \dots\} \| d \| n[\dots m[\{\rho^{\uparrow d}, \dots\} \| c \| \text{out } n \dots] \dots] \mid \overline{\text{out}}$$

if either $\rho = \text{shh}$ or $\rho = \sigma$

→

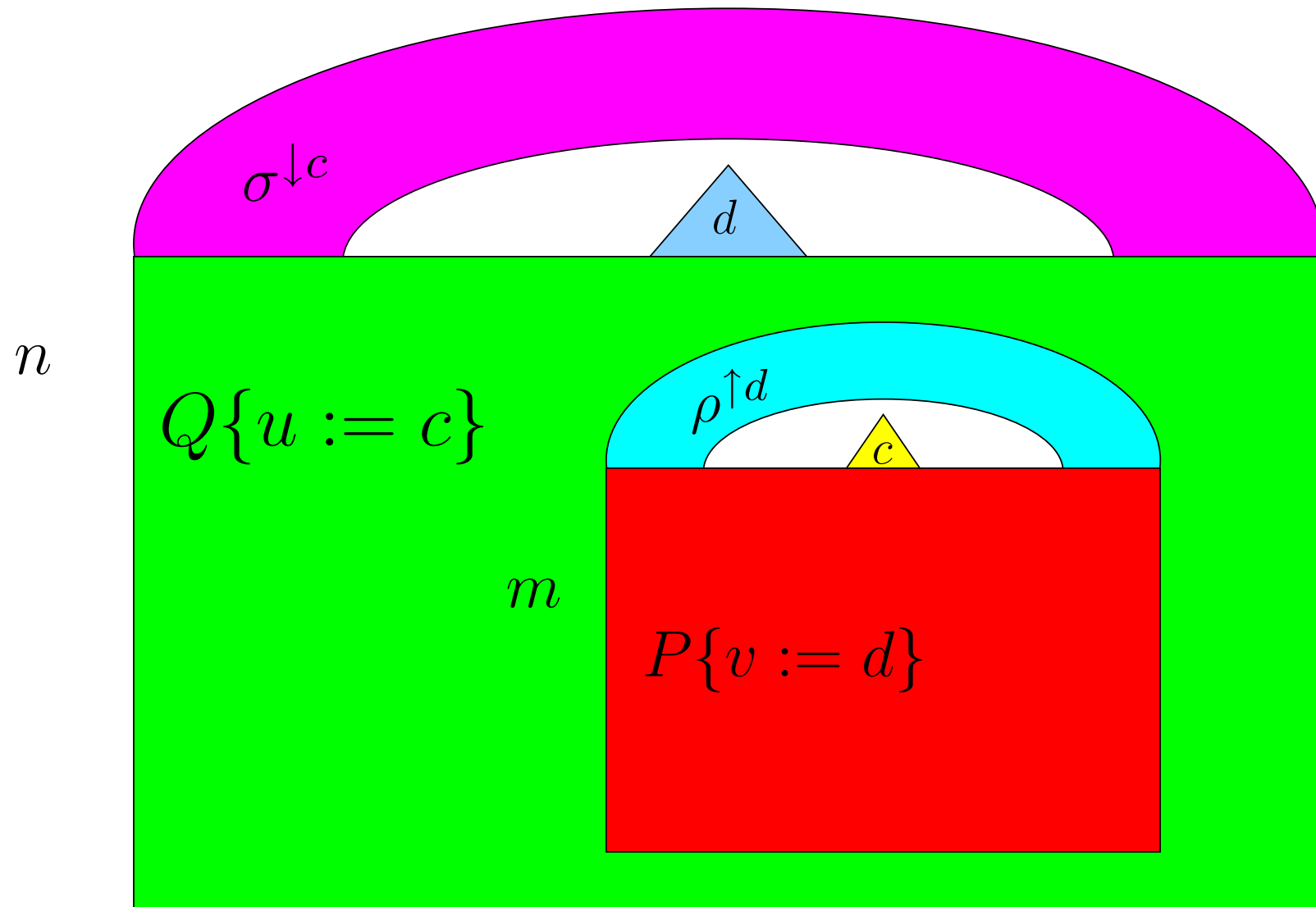
$$\{\sigma^{\downarrow c}, \dots\} \| d \| m[\{\rho^{\uparrow d}, \dots\} \| c \| \dots] \mid n[\dots]$$

Local views dynamically increase: IN



if either $\rho = \text{shh}$ or $\rho = \sigma$

Local views dynamically increase: IN



Local views dynamically increase: IN

$$m[\{\dots\} \parallel c \parallel \text{inC}(v : \rho) \ n \ .P \ \dots] \mid n[\{\dots\} \parallel d \parallel \overline{\text{inC}}(u : \sigma) \ .Q \ \dots]$$

Local views dynamically increase: IN

$m[\{\dots\} \| c \| \text{inC}(v : \rho) n . P \dots] \mid n[\{\dots\} \| d \| \overline{\text{inC}}(u : \sigma) . Q \dots]$

if either $\rho = \text{shh}$ or $\rho = \sigma$

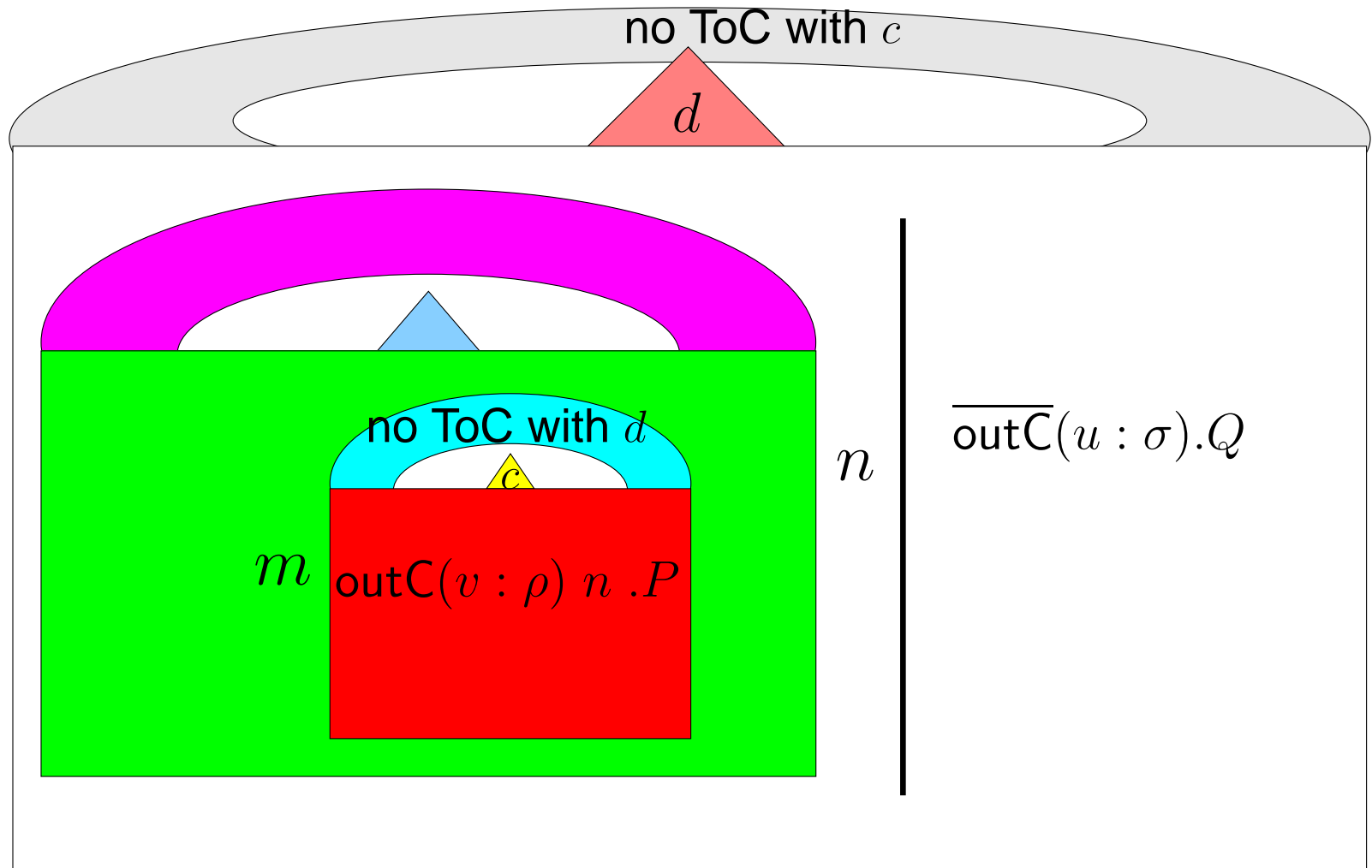
Local views dynamically increase: IN

$$m[\{\dots\} \parallel c \parallel \text{inC}(v : \rho) \ n \ .P \dots] \mid n[\{\dots\} \parallel d \parallel \overline{\text{inC}}(u : \sigma).Q \dots]$$

if either $\rho = \text{shh}$ or $\rho = \sigma$

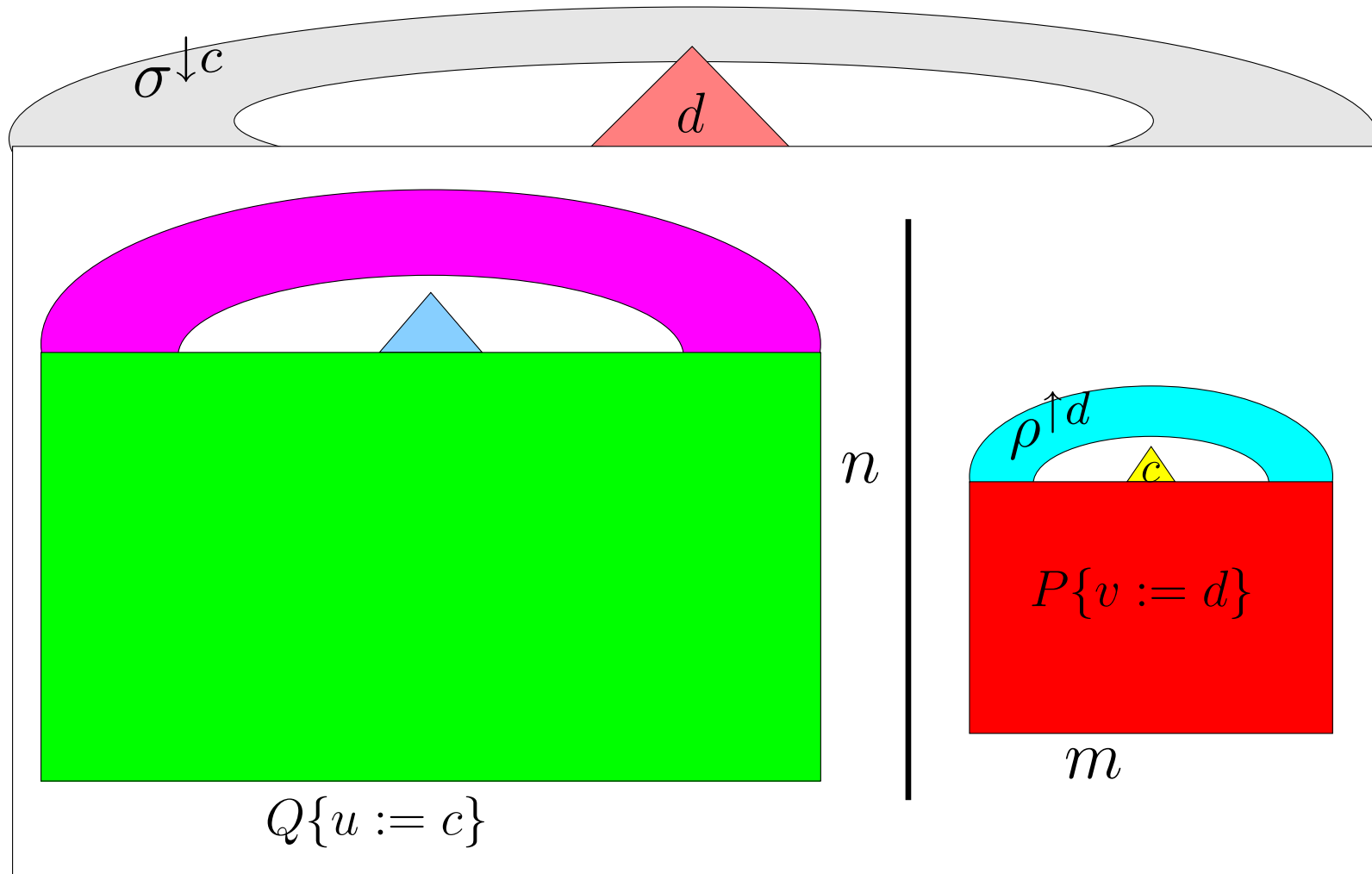
$$\longrightarrow n[\{\sigma \downarrow^c, \dots\} \parallel d \parallel m[\{\rho \uparrow^d, \dots\} \parallel c \parallel P\{v := d\} \dots] \mid Q\{u := c\} \dots]$$

Local views dynamically increase: OUT



if either $\rho = \text{shh}$ or $\rho = \sigma$

Local views dynamically increase: OUT



Local views dynamically increase: OUT

$$\{\dots\} \parallel d \parallel n[\dots m[\{\dots\} \parallel c \parallel \text{outC}(v : \rho) \ n .P \dots] \dots] \mid \overline{\text{outC}}(u : \sigma).Q$$

Local views dynamically increase: OUT

$$\{\dots\} \parallel d \parallel n[\dots m[\{\dots\} \parallel c \parallel \text{outC}(v : \rho) \ n .P \dots] \dots] \mid \overline{\text{outC}}(u : \sigma).Q$$

if either $\rho = \text{shh}$ or $\rho = \sigma$

Local views dynamically increase: OUT

$$\{\dots\} \| d \| n[\dots m[\{\dots\} \| c \| \text{outC}(v : \rho) n . P \dots] \dots] \mid \overline{\text{outC}}(u : \sigma).Q$$

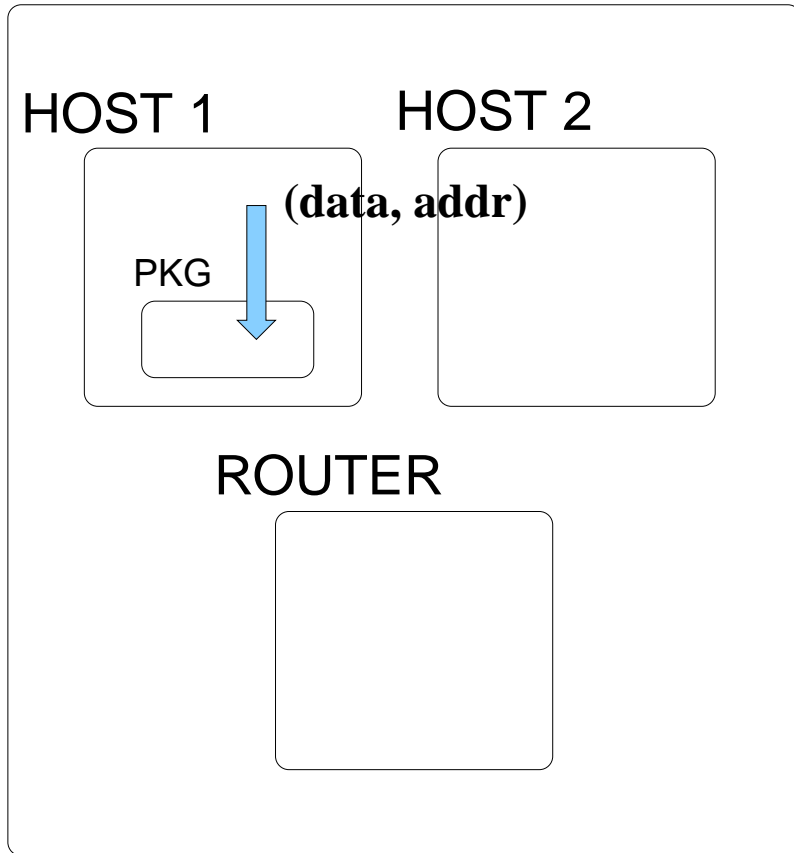
if either $\rho = \text{shh}$ or $\rho = \sigma$

→

$$\{\sigma \downarrow^c, \dots\} \| d \| m[\{\rho \uparrow^d, \dots\} \| c \| P\{v := d\} \dots] \mid n[\dots] \mid Q\{u := c\}$$

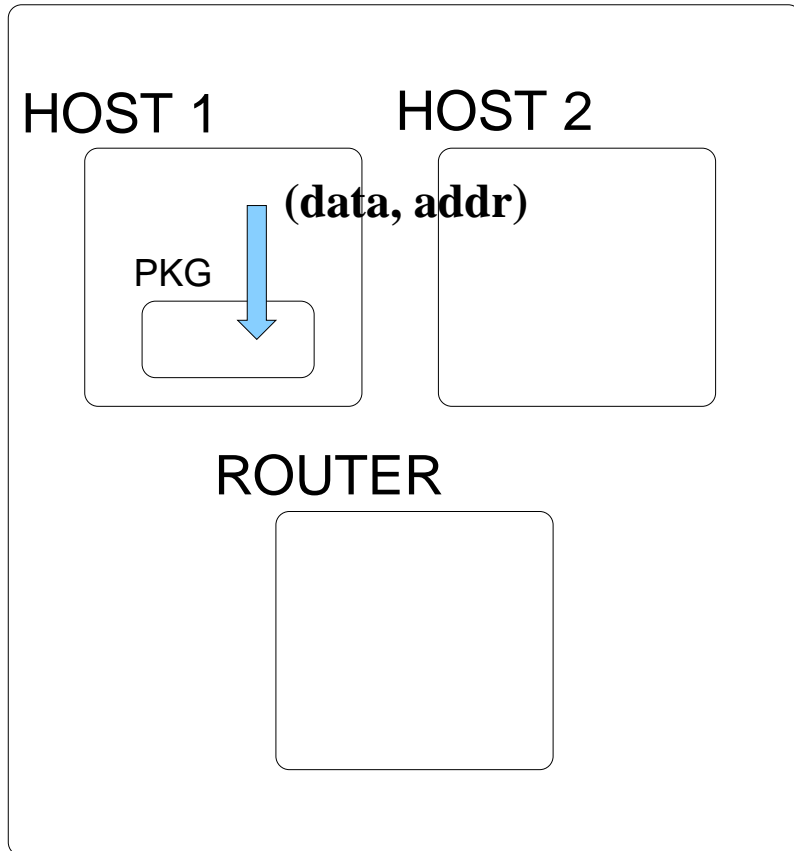
the example once more

NETWORK



the example once more

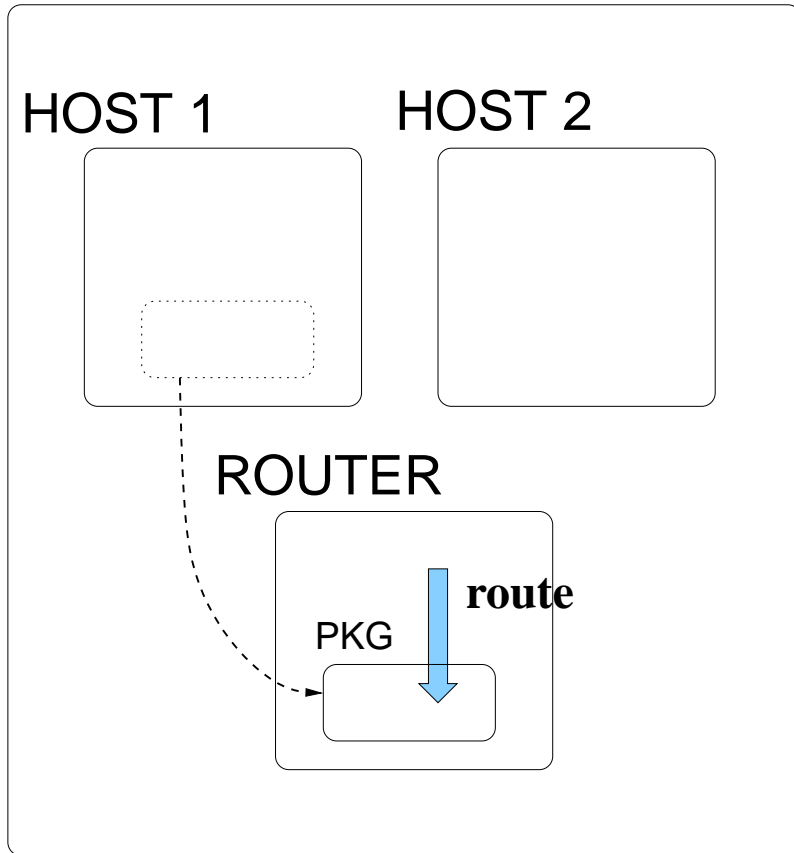
NETWORK



$\text{PKG}[\{(\text{data, addr})^{\uparrow_{\text{HOST1}}}\} ||| \dots]$

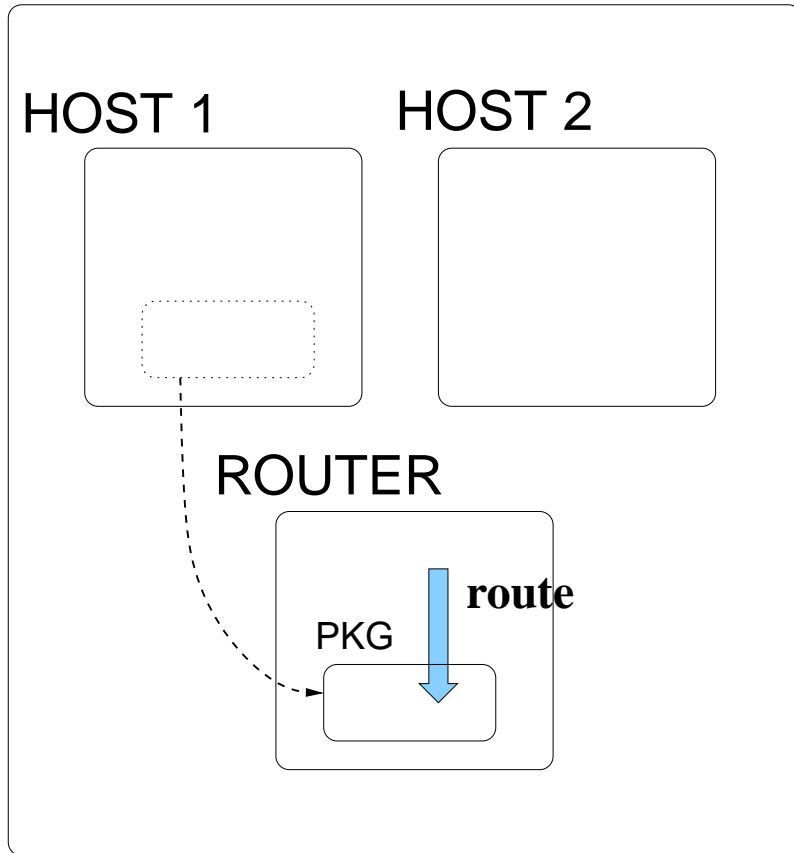
the example once more

NETWORK



the example once more

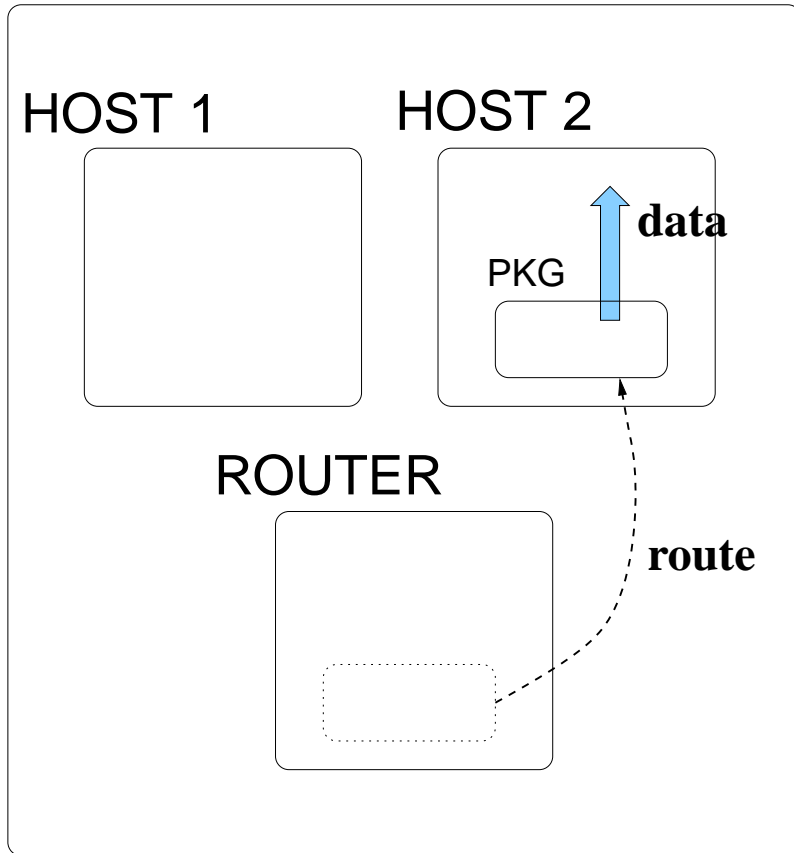
NETWORK



$\text{PKG}[\{(\mathbf{data}, \mathbf{addr})^{\uparrow_{\text{CHOST1}}},$
 $\mathbf{route}^{\uparrow_{\text{CROUTER}}}\} ||| \dots]$

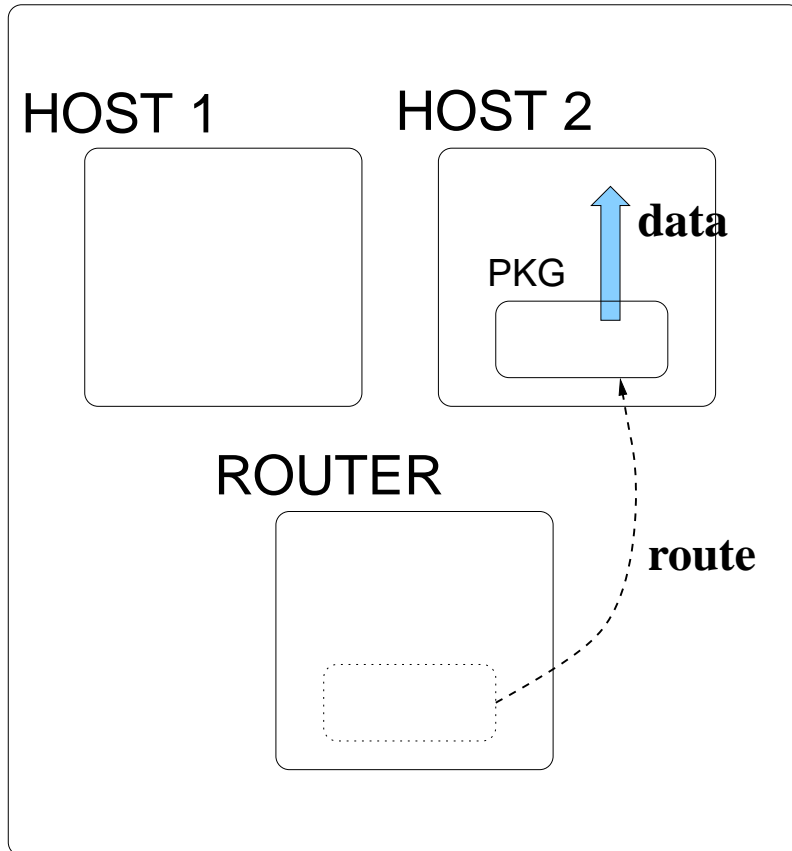
the example once more

NETWORK



the example once more

NETWORK



$\text{PKG}[\{(\text{data}, \text{addr})^{\uparrow_{\text{CHOST1}}},$
 $\text{route}^{\uparrow_{\text{CROUTER}}}, \text{data}^{\uparrow_{\text{CHOST2}}}\} ||| \dots]$

syntax of BACI

P	$::=$	0	nil process
		$P_1 \mid P_2$	composition
		$(\nu n)P$	restriction
		$!P$	replication
		$\pi.P$	prefixing
		$\alpha[\Gamma \parallel c \parallel P]$	ambient

syntax of BACI

P	$::=$	$\mathbf{0}$	nil process
		$P_1 \mid P_2$	composition
		$(\nu n)P$	restriction
		$!P$	replication
		$\pi.P$	prefixing
		$\alpha[\Gamma \parallel c \parallel P]$	ambient
π	$::=$	C	capabilities
		$(x_1 : \varphi_1, \dots, x_k : \varphi_k)^\eta$	input
		$\langle M_1, \dots, M_k \rangle^\eta$	output
		$\text{inC}(v : \rho) \alpha$	port enter
		$\text{outC}(v : \rho) \alpha$	port exit
		$\overline{\text{inC}}(v : \rho)$	allow port enter
		$\overline{\text{outC}}(v : \rho)$	allow port exit

syntax of BACI

$P ::= 0$ nil process
| $P_1 \mid P_2$ composition
| $(\nu n)P$ restriction
| $!P$ replication
| $\pi.P$ prefixing
| $\alpha[\Gamma \parallel c \parallel P]$ ambient

$C, D ::= \text{in } \alpha$ enter
| $\text{out } \alpha$ exit
| $\bar{\text{in}}$ allow enter
| $\bar{\text{out}}$ allow exit
| $C.D$ path
| x capability variable

syntax of BACI

P	$::=$	0	nil process
		$P_1 \mid P_2$	composition
		$(\nu n)P$	restriction
		$!P$	replication
		$\pi.P$	prefixing
		$\alpha[\Gamma \parallel c \parallel P]$	ambient

ambient name or variable



syntax of BACI

P	$::=$	0	nil process
		$P_1 \mid P_2$	composition
		$(\nu n)P$	restriction
		$!P$	replication
		$\pi.P$	prefixing
		$\alpha[\Gamma \parallel c \parallel P]$	ambient

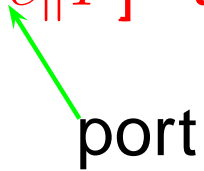
local view



syntax of BACI

P	$::=$	0	nil process
		$P_1 \mid P_2$	composition
		$(\nu n)P$	restriction
		$!P$	replication
		$\pi.P$	prefixing
		$\alpha[\Gamma \parallel c \parallel P]$	ambient

port



syntax of BACI

P	$::=$	0	nil process
		$P_1 \mid P_2$	composition
		$(\nu n)P$	restriction
		$!P$	replication
		$\pi.P$	prefixing
		$\alpha[\Gamma \parallel c \parallel P]$	ambient

↑
process

printer example

an ambient *printer* **allows ambients to enter** and receives data from them (the ports of the ambients are communicated on entering).

$$\mathit{printer}[\|c_{pr}\| \overline{\mathit{inC}}(v : \mathbf{data})(d : \mathbf{data})^{\downarrow v}]$$

printer example

an ambient *printer* allows enter ambients and **receives data from them** (the ports of the ambients are communicated on entering)

$$printer[||c_{pr}||\overline{inC}(v : \mathbf{data})(d : \mathbf{data})\downarrow v]$$

printer example

an ambient *printer* allows enter ambients and receives data from them (the ports of the ambients are communicated on entering)

$$\mathit{printer}[\|\|c_{pr}\|\|\overline{\text{inC}}(v : \mathbf{data})(d : \mathbf{data})\downarrow v]$$

printer example

an ambient *printer* allows enter ambients and receives data from them (the ports of the ambients are communicated on entering)

$$printer[||c_{pr}|| \overline{inC}(v : \mathbf{data})(d : \mathbf{data})^{\downarrow v}]$$

an ambient *job* (inside the ambient *client*) **receives the data and the printer name**, exits the client, enters the printer (exchanging port names), and communicates the data.

$$job[\{(\mathbf{data}, \mathbf{amb})^{\uparrow c_{cl}}\} ||c_j||$$

$$(\mathbf{d} : \mathbf{data}, p : \mathbf{amb})^{\uparrow c_{cl}}.out\ client.inC(v' : \mathbf{data}.\langle d \rangle^{\uparrow v'}) p]$$

printer example

an ambient *printer* allows enter ambients and receives data from them (the ports of the ambients are communicated on entering)

$$printer[||c_{pr}||!\overline{\text{inC}}(v : \mathbf{data})(d : \mathbf{data})^{\downarrow v}]$$

an ambient *job* (inside the ambient *client*) receives the data and the printer name, **exits the client**, enters the printer (exchanging port names) and communicates the data

$$job[\{(\mathbf{data}, \mathbf{amb})^{\uparrow c_{cl}}\}||c_j||$$

$$(d : \mathbf{data}, p : \mathbf{amb})^{\uparrow c_{cl}}.\mathbf{out} \mathit{client}.\text{inC}(v' : \mathbf{data}) p .\langle d \rangle^{\uparrow v'}]$$

printer example

an ambient *printer* allows enter ambients and receives data from them (the ports of the ambients are communicated on entering)

$$printer[\|c_{pr}\| \overline{\text{inC}}(v : \mathbf{data})(d : \mathbf{data})^{\downarrow v}]$$

an ambient *job* (inside the ambient *client*) receives the data and the printer name, exits the client, **enters the printer (exchanging port names)** and communicates the data

$$job[\{(\mathbf{data}, \mathbf{amb})^{\uparrow c_{cl}}\} \|c_j\|$$

$$(d : \mathbf{data}, p : \mathbf{amb})^{\uparrow c_{cl}}.out\ client.inC(v' : \mathbf{data})\ p.\langle d \rangle^{\uparrow v'}]$$

printer example

an ambient *printer* allows enter ambients and receives data from them (the ports of the ambients are communicated on entering)

$$printer[||c_{pr}||!\overline{inC}(v : \mathbf{data})(d : \mathbf{data})\downarrow v]$$

an ambient *job* (inside the ambient *client*) receives the data and the printer name, exits the client, enters the printer (exchanging port names) and **communicates the data**

$$job[\{(\mathbf{data}, \mathbf{amb})\uparrow^{c_{cl}}\}||c_j||]$$

$$(d : \mathbf{data}, p : \mathbf{amb})\uparrow^{c_{cl}}.out\ client.inC(v' : \mathbf{data})\ p.\langle d \rangle\uparrow^{v'}$$

printer example

an ambient *printer* allows enter ambients and receives data from them (the ports of the ambients are communicated on entering)

$$printer[||c_{pr}||!\overline{inC}(v : \mathbf{data})(d : \mathbf{data})\downarrow v]$$

an ambient *job* (inside the ambient *client*) receives the data and the printer name, exits the client, enters the printer (exchanging port names) and communicates the data

$$job[\{(\mathbf{data}, \mathbf{amb})\uparrow^{c_{cl}}\}||c_j||]$$

$$(d : \mathbf{data}, p : \mathbf{amb})\uparrow^{c_{cl}}.out\ client.inC(v' : \mathbf{data})\ p.\langle d \rangle\uparrow^{v'}$$

Operational Semantics I

structural congruence:

Operational Semantics I

structural congruence:

identifies the “capabilities” with the “port enter/exit”, “allow port enter/exit” when the communicated port name is cancelled:

Operational Semantics I

structural congruence:

identifies the “capabilities” with the “port enter/exit”, “allow port enter/exit” when the communicated port name is cancelled:

$$\text{in } n.P \equiv \text{inC}(v : \text{shh}) n .P, \text{ if } v \notin \text{fv}(P)$$

$$\text{out } n.P \equiv \text{outC}(v : \text{shh}) n .P, \text{ if } v \notin \text{fv}(P)$$

$$\overline{\text{in}}.P \equiv \overline{\text{inC}}(v : \text{shh}).P, \text{ if } v \notin \text{fv}(P)$$

$$\overline{\text{out}}.P \equiv \overline{\text{outC}}(v : \text{shh}).P, \text{ if } v \notin \text{fv}(P)$$

Operational Semantics I

structural congruence:

identifies the “capabilities” with the “port enter/exit”, “allow port enter/exit” when the communicated port name is cancelled:

$$\text{in } n.P \equiv \text{inC}(v : \text{shh}) n .P, \text{ if } v \notin \text{fv}(P)$$

$$\text{out } n.P \equiv \text{outC}(v : \text{shh}) n .P, \text{ if } v \notin \text{fv}(P)$$

$$\overline{\text{in}}.P \equiv \overline{\text{inC}}(v : \text{shh}).P, \text{ if } v \notin \text{fv}(P)$$

$$\overline{\text{out}}.P \equiv \overline{\text{outC}}(v : \text{shh}).P, \text{ if } v \notin \text{fv}(P)$$

why do we need capabilities?

Operational Semantics I

structural congruence:

identifies the “capabilities” with the “port enter/exit”, “allow port enter/exit” when the communicated port name is cancelled:

$$\text{in } n.P \equiv \text{inC}(v : \text{shh}) n .P, \text{ if } v \notin \text{fv}(P)$$

$$\text{out } n.P \equiv \text{outC}(v : \text{shh}) n .P, \text{ if } v \notin \text{fv}(P)$$

$$\overline{\text{in}}.P \equiv \overline{\text{inC}}(v : \text{shh}).P, \text{ if } v \notin \text{fv}(P)$$

$$\overline{\text{out}}.P \equiv \overline{\text{outC}}(v : \text{shh}).P, \text{ if } v \notin \text{fv}(P)$$

why do we need capabilities? since they can be sent as messages!

Operational Semantics II

some reduction rules:

Operational Semantics II

(ENTER)

$$n[\Gamma_n \parallel c_n \parallel \text{inC}(v : \rho) m . P_1 \mid P_2] \mid m[\Gamma_m \parallel c_m \parallel \overline{\text{inC}}(v' : \rho') . Q_1 \mid Q_2]$$

Operational Semantics II

(ENTER)

$$n[\Gamma_n \| c_n \| \text{inC}(v : \rho) m . P_1 \mid P_2] \mid m[\Gamma_m \| c_m \| \overline{\text{inC}}(v' : \rho') . Q_1 \mid Q_2]$$

→

$$m[\Gamma_m \oplus \rho'^{\downarrow c_n} \| c_m \|$$

$$n[\Gamma_n \oplus \rho^{\uparrow c_m} \| c_n \| P_1 \{ \uparrow v := \uparrow c_m \} \mid P_2] \mid Q_1 \{ \downarrow v' := \downarrow c_n \} \mid Q_2]$$

$$\text{if } \uparrow c_m(\Gamma_n) \sqcup \rho \preceq \downarrow c_n(\Gamma_m) \sqcup \rho'$$

Operational Semantics II

(ENTER)

$$n[\Gamma_n \parallel c_n \parallel \text{inC}(v : \rho) \ m \ .P_1 \mid P_2] \mid m[\Gamma_m \parallel c_m \parallel \overline{\text{inC}}(v' : \rho').Q_1 \mid Q_2]$$

→

$$m[\Gamma_m \oplus \rho'^{\downarrow c_n} \parallel c_m \parallel$$

$$n[\Gamma_n \oplus \rho^{\uparrow c_m} \parallel c_n \parallel P_1 \{ \uparrow v := \uparrow c_m \} \mid P_2] \mid Q_1 \{ \downarrow v' := \downarrow c_n \} \mid Q_2]$$

$$\text{if } \uparrow c_m(\Gamma_n) \sqcup \rho \preceq \downarrow c_n(\Gamma_m) \sqcup \rho'$$

addition of located types to local views

$$\Gamma \oplus \rho^\eta = \begin{cases} \Gamma & \text{if } \rho = \text{shh}, \\ \Gamma, \rho^\eta & \text{otherwise.} \end{cases}$$

Operational Semantics II

(ENTER)

$$n[\Gamma_n \parallel c_n \parallel \text{inC}(v : \rho) \ m . P_1 \mid P_2] \mid m[\Gamma_m \parallel c_m \parallel \overline{\text{inC}}(v' : \rho'). Q_1 \mid Q_2]$$

→

$$m[\Gamma_m \oplus \rho'^{\downarrow c_n} \parallel c_m \parallel$$

$$n[\Gamma_n \oplus \rho^{\uparrow c_m} \parallel c_n \parallel P_1 \{ \uparrow v := \uparrow c_m \} \mid P_2] \mid Q_1 \{ \downarrow v' := \downarrow c_n \} \mid Q_2]$$

$$\text{if } \uparrow c_m(\Gamma_n) \sqcup \rho \preceq \downarrow c_n(\Gamma_m) \sqcup \rho'$$

join of communication types

$$\rho \sqcup \rho' = \begin{cases} \rho & \text{if } \rho = \rho' \text{ or } \rho' = \text{shh}, \\ \rho' & \text{if } \rho = \text{shh}, \\ \perp & \text{otherwise.} \end{cases}$$

Operational Semantics II

(ENTER)

$$n[\Gamma_n \| c_n \| \text{inC}(v : \rho) m . P_1 \mid P_2] \mid m[\Gamma_m \| c_m \| \overline{\text{inC}}(v' : \rho'). Q_1 \mid Q_2]$$

→

$$m[\Gamma_m \oplus \rho'^{\downarrow c_n} \| c_m \|$$

$$n[\Gamma_n \oplus \rho^{\uparrow c_m} \| c_n \| P_1 \{ \uparrow v := \uparrow c_m \} \mid P_2] \mid Q_1 \{ \downarrow v' := \downarrow c_n \} \mid Q_2]$$

$$\text{if } \uparrow c_m(\Gamma_n) \sqcup \rho \preceq \downarrow c_n(\Gamma_m) \sqcup \rho'$$

preorder on communication types

$$\rho \preceq \rho' \text{ iff } \rho \sqcup \rho' = \rho'$$

Operational Semantics II

(ENTER)

$$n[\Gamma_n \parallel c_n \parallel \text{inC}(v : \rho) \ m \ .P_1 \mid P_2] \mid m[\Gamma_m \parallel c_m \parallel \overline{\text{inC}}(v' : \rho').Q_1 \mid Q_2]$$

→

$$m[\Gamma_m \oplus \rho'^{\downarrow c_n} \parallel c_m \parallel$$

$$n[\Gamma_n \oplus \rho^{\uparrow c_m} \parallel c_n \parallel P_1 \{ \uparrow v := \uparrow c_m \} \mid P_2] \mid Q_1 \{ \downarrow v' := \downarrow c_n \} \mid Q_2]$$

$$\text{if } \uparrow c_m(\Gamma_n) \sqcup \rho \preceq \downarrow c_n(\Gamma_m) \sqcup \rho'$$

application of locations

to located types

to local views

$$\eta(\tau) = \begin{cases} \rho & \text{if } \tau = \rho^\eta, \\ \text{shh} & \text{otherwise} \end{cases} \quad \begin{aligned} \eta(\emptyset) &= \text{shh} \\ \eta(\Gamma) &= \sqcup_{\tau \in \Gamma} \eta(\tau) \end{aligned}$$

Operational Semantics III

(INPUT \downarrow - \uparrow)

$$m[\Gamma_m \parallel c_m \parallel (\tilde{x} : \tilde{\varphi})^{\downarrow c_n} . P \mid n[\Gamma_n \parallel c_n \parallel \langle \tilde{M} \rangle^{\uparrow c_m} . Q \mid R] \mid S]$$

Operational Semantics III

(INPUT \downarrow - \uparrow)

$$\begin{aligned} m[\Gamma_m \parallel c_m \parallel (\tilde{x} : \tilde{\varphi}) \downarrow^{c_n} . P \mid n[\Gamma_n \parallel c_n \parallel \langle \tilde{M} \rangle \uparrow^{c_m} . Q \mid R] \mid S] \\ \longrightarrow \\ m[\Gamma_m \parallel c_m \parallel P\{\tilde{x} := \tilde{M}\} \mid n[\Gamma_n \parallel c_n \parallel Q \mid R] \mid S] \end{aligned}$$

printer example cont.

PRINTER= $printer[\|c_{pr}\| \overline{!inC}(v : \mathbf{data}).(d : \mathbf{data})^{\downarrow v}]$

JOB= $job[\{(\mathbf{data}, \mathbf{amb})^{\uparrow c_{cl}}\} \|c_j\| (d : \mathbf{data}, p : \mathbf{amb})^{\uparrow c_{cl}}.out \textit{client}.$

$inC(v' : \mathbf{data}) p . \langle d \rangle^{\uparrow v'}$

printer example cont.

PRINTER= $printer[\|c_{pr}\| \overline{\text{inC}}(v : \mathbf{data}).(d : \mathbf{data})^{\downarrow v}]$

JOB= $job[\{(\mathbf{data}, \mathbf{amb})^{\uparrow c_{cl}}\} \|c_j\| (d : \mathbf{data}, p : \mathbf{amb})^{\uparrow c_{cl}}.out \textit{client}.$

$\textit{inC}(v' : \mathbf{data}) p . \langle d \rangle^{\uparrow v'}$]

CLIENT= $client[\{(\mathbf{data}, \mathbf{addr})^{\downarrow c_j}\} \|c_{cl}\| \langle (d_1, printer) \rangle^{\downarrow c_j} \mid \text{JOB}]$

printer example cont.

PRINTER= $printer[\|c_{pr}\| \overline{!inC}(v : \mathbf{data}).(d : \mathbf{data})^{\downarrow v}]$

JOB= $job[\{(\mathbf{data}, \mathbf{amb})^{\uparrow c_{cl}}\} \|c_j\| (d : \mathbf{data}, p : \mathbf{amb})^{\uparrow c_{cl}}.out \textit{client}.$

$inC(v' : \mathbf{data}) p . \langle d \rangle^{\uparrow v'}$]

CLIENT= $client[\{(\mathbf{data}, \mathbf{addr})^{\downarrow c_j}\} \|c_{cl}\| \langle (d_1, printer) \rangle^{\downarrow c_j} \mid \text{JOB}]$

CLIENT \mid PRINTER $\mid \overline{!out} \longrightarrow^* \text{CLIENT}' \mid \text{JOB}' \mid \text{PRINTER} \mid \overline{!out}$

printer example cont.

PRINTER= $printer[\|c_{pr}\| \overline{!inC}(v : \mathbf{data}).(d : \mathbf{data}) \downarrow v]$

JOB= $job[\{(\mathbf{data}, \mathbf{amb}) \uparrow^{c_{cl}}\} \|c_j\| (d : \mathbf{data}, p : \mathbf{amb}) \uparrow^{c_{cl}}.out \ client.$
 $inC(v' : \mathbf{data}) p .\langle d \rangle \uparrow^{v'}]$

CLIENT= $client[\{(\mathbf{data}, \mathbf{addr}) \downarrow^{c_j}\} \|c_{cl}\| \langle (d_1, printer) \rangle \downarrow^{c_j} \mid \text{JOB}]$

CLIENT \mid PRINTER $\mid \overline{!out} \longrightarrow^* \text{CLIENT}' \mid \text{JOB}' \mid \text{PRINTER} \mid \overline{!out}$

JOB' = $job[\{(\mathbf{data}, \mathbf{amb}) \uparrow^{c_{cl}}\} \|c_j\| inC(v' : \mathbf{data}) p .\langle d \rangle \uparrow^{v'}]$

CLIENT' = $client[\{(\mathbf{data}, \mathbf{addr}) \downarrow^{c_j}\} \|c_{cl}\|]$

printer example cont.

PRINTER= $printer[\|c_{pr}\| \overline{!inC}(v : \mathbf{data}).(d : \mathbf{data}) \downarrow v]$

JOB= $job[\{(\mathbf{data}, \mathbf{amb}) \uparrow^{c_{cl}}\} \|c_j\| (d : \mathbf{data}, p : \mathbf{amb}) \uparrow^{c_{cl}}.out \ client.$
 $inC(v' : \mathbf{data}) p . \langle d \rangle \uparrow^{v'}]$

CLIENT= $client[\{(\mathbf{data}, \mathbf{addr}) \downarrow^{c_j}\} \|c_{cl}\| \langle (d_1, printer) \rangle \downarrow^{c_j} \mid \text{JOB}]$

CLIENT \mid PRINTER $\mid \overline{!out} \longrightarrow^* \text{CLIENT}' \mid \text{JOB}' \mid \text{PRINTER} \mid \overline{!out}$

JOB' = $job[\{(\mathbf{data}, \mathbf{amb}) \uparrow^{c_{cl}}\} \|c_j\| inC(v' : \mathbf{data}) p . \langle d \rangle \uparrow^{v'}]$

CLIENT' = $client[\{(\mathbf{data}, \mathbf{addr}) \downarrow^{c_j}\} \|c_{cl}\|]$

CLIENT' \mid JOB' \mid PRINTER $\mid \overline{!out} \longrightarrow^* \text{CLIENT}' \mid \text{PRINTER}' \mid \overline{!out}$

printer example cont.

PRINTER= $printer[||c_{pr}||\overline{!inC}(v : \mathbf{data}).(d : \mathbf{data})\downarrow^v]$

JOB= $job[\{(\mathbf{data}, \mathbf{amb})\uparrow^{c_{cl}}\}||c_j||\langle(d : \mathbf{data}, p : \mathbf{amb})\uparrow^{c_{cl}}.out\ client.$
 $inC(v' : \mathbf{data}) p .\langle d \rangle\uparrow^{v'}]$

CLIENT= $client[\{(\mathbf{data}, \mathbf{addr})\downarrow^{c_j}\}||c_{cl}||\langle(d_1, printer)\rangle\downarrow^{c_j} | \text{JOB}]$

CLIENT | PRINTER | $\overline{!out} \longrightarrow^* \text{CLIENT}' | \text{JOB}' | \text{PRINTER} | \overline{!out}$

JOB' = $job[\{(\mathbf{data}, \mathbf{amb})\uparrow^{c_{cl}}\}||c_j||inC(v' : \mathbf{data}) p .\langle d \rangle\uparrow^{v'}]$

CLIENT' = $client[\{(\mathbf{data}, \mathbf{addr})\downarrow^{c_j}\}||c_{cl}||]$

CLIENT' | JOB' | PRINTER | $\overline{!out} \longrightarrow^* \text{CLIENT}' | \text{PRINTER}' | \overline{!out}$

PRINTER' = $printer[\{\mathbf{data}\downarrow^{c_j}\}||c_{pr}||\langle(d : \mathbf{data})\downarrow^{c_j} | \text{JOB}'']$

JOB'' = $job[\{(\mathbf{data}, \mathbf{amb})\uparrow^{c_{cl}}, \mathbf{data}\uparrow^{c_{pr}}\}||c_j||\langle d \rangle\uparrow^{c_{pr}}]$

typing judgment

$$\Sigma \vdash_c P : \Gamma$$


typing judgment

$$\Sigma \vdash_c P : \Gamma$$

the environment associates types to variables $\left\{ \begin{array}{l} x : \text{amb} \\ x : \text{cap} \end{array} \right.$

typing judgment

$$\Sigma \vdash_c P : \Gamma$$


the port

typing judgment

$$\Sigma \vdash_c P : \Gamma$$

the local view



typing judgment

$$\Sigma \vdash_c P : \Gamma$$

the process



typing judgment

$$\Sigma \vdash_c P : \Gamma$$

P is a well-formed process in the environment Σ assuming the local communication interface of its host consists of the communication port c and the local view Γ

some typing rules I

(PROC-INC)

$$\Sigma \vdash_c \text{inC}(v : \rho) \alpha . P : \Gamma$$

some typing rules I

(PROC-INC)

$$\frac{\Sigma \vdash_c P :}{\Sigma \vdash_c \text{inC}(v : \rho) \alpha . P : \Gamma}$$

some typing rules I

(PROC-INC)

$$\frac{\Sigma \vdash_c P : \Gamma \oplus \rho^{\uparrow v}}{\Sigma \vdash_c \text{inC}(v : \rho) \alpha . P : \Gamma}$$

some typing rules II

(PROC-AMB)

$$\Sigma \vdash_c \beta[\Gamma' || c' || P] : \Gamma$$

some typing rules II

(PROC-AMB)

$$\Sigma \vdash_{c'} P : \Gamma'$$

$$\Sigma \vdash_c \beta[\Gamma' || c' || P] : \Gamma$$

some typing rules II

(PROC-AMB)

$$\Sigma \vdash_{c'} P : \Gamma' \quad \Sigma \vdash \beta : \text{amb}$$

$$\Sigma \vdash_c \beta[\Gamma' || c' || P] : \Gamma$$

some typing rules II

(PROC-AMB)

$$\Sigma \vdash_{c'} P : \Gamma' \quad \Sigma \vdash \beta : \text{amb}$$

$$\uparrow c(\Gamma') \preceq \downarrow c'(\Gamma)$$

parents must listen to children, but not vice-versa

$$\Sigma \vdash_c \beta[\Gamma' || c' || P] : \Gamma$$

some typing rules II

(PROC-AMB)

$$\Sigma \vdash_{c'} P : \Gamma' \quad \Sigma \vdash \beta : \text{amb}$$

$$\uparrow c(\Gamma') \preceq \downarrow c'(\Gamma)$$

Γ' is closed does not contain port variables

$$\Sigma \vdash_c \beta[\Gamma' || c' || P] : \Gamma$$

some typing rules II

(PROC-AMB)

$$\Sigma \vdash_{c'} P : \Gamma' \quad \Sigma \vdash \beta : \text{amb}$$

$$\uparrow c(\Gamma') \preceq \downarrow c'(\Gamma)$$

Γ' is closed Γ is *ok*

$$\Sigma \vdash_c \beta[\Gamma' || c' || P] : \Gamma$$

$\eta(\Gamma) \neq \perp$ for all η

each port variable occurs at most once in Γ

some typing rules II

(PROC-AMB)

$$\Sigma \vdash_{c'} P : \Gamma' \quad \Sigma \vdash \beta : \text{amb}$$
$$\uparrow c(\Gamma') \preceq \downarrow c'(\Gamma)$$

Γ' is closed Γ is *ok*

$$\Sigma \vdash_c \beta[\Gamma' || c' || P] : \Gamma$$

if parents do not listen to children...

other children can enter producing a wrong communication

if parents do not listen to children...

other children can enter producing a wrong communication

we could type

$$p[\emptyset \parallel c \parallel \overline{\text{inC}}(v : \text{amb}). \langle q \rangle \downarrow^v \mid m[\text{cap}^{\uparrow c} \parallel c' \parallel (x : \text{cap})^{\uparrow c}. P]] \mid \\ n[\emptyset \parallel c' \parallel \text{inC}(v' : \text{amb}) p]$$

if parents do not listen to children...

other children can enter producing a wrong communication

we could type

$$p[\emptyset \parallel c \parallel \overline{\text{inC}}(v : \text{amb}).\langle q \rangle^{\downarrow v} \mid m[\text{cap}^{\uparrow c} \parallel c' \parallel (x : \text{cap})^{\uparrow c}.P]] \mid \\ n[\emptyset \parallel c' \parallel \text{inC}(v' : \text{amb}) p]$$

which reduces to

$$p[\text{amb}^{\downarrow c'} \parallel c \parallel \langle q \rangle^{\downarrow c'} \mid m[\text{cap}^{\uparrow c} \parallel c' \parallel (x : \text{cap})^{\uparrow c}.P] \mid n[\text{amb}^{\uparrow c} \parallel c' \parallel]]$$

if parents do not listen to children...

other children can enter producing a wrong communication

we could type

$$p[\emptyset \parallel c \parallel \overline{\text{inC}}(v : \text{amb}). \langle q \rangle \downarrow^v \mid m[\text{cap}^{\uparrow c} \parallel c' \parallel (x : \text{cap})^{\uparrow c}. P]] \mid \\ n[\emptyset \parallel c' \parallel \text{inC}(v' : \text{amb}) p]$$

which reduces to

$$p[\text{amb}^{\downarrow c'} \parallel c \parallel \langle q \rangle \downarrow^{c'} \mid m[\text{cap}^{\uparrow c} \parallel c' \parallel (x : \text{cap})^{\uparrow c}. P] \mid n[\text{amb}^{\uparrow c} \parallel c' \parallel]]$$

an ambient is sent to a process expecting a capability!

Subject Reduction

If $\Sigma \vdash_c P : \Gamma$ and $P \longrightarrow Q$,
then $\Sigma \vdash_c Q : \Gamma$

Behavioural Semantics I

different choices of barbs:

$$P \downarrow_n^1 \quad \triangleq \quad P \equiv (\nu \tilde{m})(n[\Gamma \| c \| \overline{\text{in}}.Q \mid R] \mid S)$$

$$P \downarrow_n^2 \quad \triangleq \quad P \equiv (\nu \tilde{m})(n[\Gamma \| c \| \overline{\text{in}}\overline{C}(v : \rho).Q \mid R] \mid S)$$

$$P \downarrow_{\langle c, c' \rangle}^3 \quad \triangleq \quad P \equiv (\nu \tilde{m})(n[\Gamma \| c \| (x : \varphi)^{\uparrow c'}.Q \mid R] \mid S)$$

$$P \downarrow_{\langle c, c' \rangle}^4 \quad \triangleq \quad P \equiv (\nu \tilde{m})(n[\Gamma \| c \| \langle M \rangle^{\uparrow c'}.Q \mid R] \mid S)$$

Behavioural Semantics I

different choices of barbs:

$$P \downarrow_n^1 \quad \triangleq \quad P \equiv (\nu \tilde{m})(n[\Gamma \| c \| \overline{\text{in}}.Q \mid R] \mid S)$$

$$P \downarrow_n^2 \quad \triangleq \quad P \equiv (\nu \tilde{m})(n[\Gamma \| c \| \overline{\text{in}}\overline{C}(v : \rho).Q \mid R] \mid S)$$

$$P \downarrow_{\langle c, c' \rangle}^3 \quad \triangleq \quad P \equiv (\nu \tilde{m})(n[\Gamma \| c \| (x : \varphi)^{\uparrow c'}.Q \mid R] \mid S)$$

$$P \downarrow_{\langle c, c' \rangle}^4 \quad \triangleq \quad P \equiv (\nu \tilde{m})(n[\Gamma \| c \| \langle M \rangle^{\uparrow c'}.Q \mid R] \mid S)$$

$$P \Downarrow_n^i (P \downarrow_{\langle c, c' \rangle}^i) \text{ if } P \longrightarrow^* P' \text{ and } P' \downarrow_n^i (P \downarrow_{\langle c, c' \rangle}^i)$$

Behavioural Semantics II

Reduction Barbed Congruence

\cong_i (with $i \in [1..4]$) is the largest equivalence relation that is preserved by contexts and, when restricted to closed processes

- is **reduction closed**: $P \cong_i Q$ and $P \longrightarrow P'$ imply the existence of some Q' such that $Q \Longrightarrow Q'$ and $P' \cong_i Q'$
- is **barb preserving**: $P \cong_i Q$ and $P \Downarrow_n^i (P \Downarrow_{\langle c, c' \rangle}^i)$ imply $Q \Downarrow_n^i (Q \Downarrow_{\langle c, c' \rangle}^i)$

Behavioural Semantics II

Reduction Barbed Congruence

\cong_i (with $i \in [1..4]$) is the largest equivalence relation that is preserved by contexts and, when restricted to closed processes

- is **reduction closed**: $P \cong_i Q$ and $P \longrightarrow P'$ imply the existence of some Q' such that $Q \Longrightarrow Q'$ and $P' \cong_i Q'$
- is **barb preserving**: $P \cong_i Q$ and $P \downarrow_n^i (P \downarrow_{\langle c, c' \rangle}^i)$ imply $Q \downarrow_n^i (Q \downarrow_{\langle c, c' \rangle}^i)$

Independence from barbs $\cong_i = \cong_j$ for all $i, j \in [1..4]$

Some Algebraic Laws

an ambient only willing to communicate with its father but using a “wrong” port name is dead:

$$n[\Gamma_n \parallel c_n \parallel m[\Gamma_m \parallel c_m \parallel (\tilde{x} : \tilde{\varphi})^{\uparrow c}.P] \mid Q] \cong n[\Gamma_n \parallel c_n \parallel Q]$$

$$n[\Gamma_n \parallel c_n \parallel m[\Gamma_m \parallel c_m \parallel \langle \tilde{M} \rangle^{\uparrow c}.P] \mid Q] \cong n[\Gamma_n \parallel c_n \parallel Q]$$

Some Algebraic Laws

communications can be forced also when other processes are active:

if c_m, m do not occur in S and c_n, n, m do not occur in R

$$(\nu n)(n[\Gamma_n \parallel c_n \parallel (\nu m)(m[\Gamma_m \parallel c_m \parallel \langle \tilde{M} \rangle^{\uparrow c_n}.P \mid R]) \mid (\tilde{x} : \tilde{\varphi})^{\downarrow c_m}.Q \mid S])$$

$$\cong$$

$$(\nu n)(\nu m)(n[\Gamma_n \parallel c_n \parallel m[\Gamma_m \parallel c_m \parallel P \mid R]) \mid Q\{\tilde{x} := \tilde{M}\} \mid S)$$

$$(\nu n)(n[\Gamma_n \parallel c_n \parallel (\nu m)(m[\Gamma_m \parallel c_m \parallel (\tilde{x} : \tilde{\varphi})^{\uparrow c_n}.P \mid R]) \mid \langle \tilde{M} \rangle^{\downarrow c_m}.Q \mid S])$$

$$\cong$$

$$(\nu n)(n[\Gamma_n \parallel c_n \parallel (\nu m)(m[\Gamma_m \parallel c_m \parallel P\{\tilde{x} := \tilde{M}\} \mid R]) \mid Q \mid S])$$

Labelled Transition System

$$P \xrightarrow{\xi} O$$

an outcome O may be

- a process

- a concretion $\left\{ \begin{array}{l} (\nu \tilde{p}) \langle P \rangle Q \\ (\nu \tilde{p}) \langle \tilde{M} \rangle Q \end{array} \right.$

Labelled Transition System

(OUTPUT)

$$\langle \tilde{M} \rangle^\eta . P \xrightarrow{\langle - \rangle^\eta} \langle\langle \tilde{M} \rangle\rangle P$$

(GET)

$$P \xrightarrow{(\tilde{M})^{\uparrow c}} P'$$

$$m[\Gamma_m \parallel c_m \parallel P] \xrightarrow{\tilde{M} \text{ get } (c_m, c)} m[\Gamma_m \parallel c_m \parallel P']$$

Labelled Transition System

Higher-order transitions

$$P \xrightarrow{\lambda} P'$$

(HO OUTPUT[↑])

$$P \xrightarrow{\langle - \rangle^{\uparrow c}} (\nu \tilde{p}) \langle \tilde{M} \rangle P'$$

$$P \xrightarrow{\langle - \rangle^{\uparrow c} m[\Gamma_m \parallel c_m \parallel R] \mid Q} (\nu \tilde{p}) (m[\Gamma_m \parallel c_m \parallel P' \mid R] \mid Q\{\tilde{x} := \tilde{M}\})$$

Labelled Transition System

\Longrightarrow denotes the reflexive and transitive closure of $\xrightarrow{\tau}$
 $\xRightarrow{\lambda}$ denotes $\Longrightarrow \xrightarrow{\lambda} \Longrightarrow$

Labelled Transition System

\Longrightarrow denotes the reflexive and transitive closure of $\xrightarrow{\tau}$

$\Longrightarrow^{\lambda}$ denotes $\Longrightarrow \xrightarrow{\lambda} \Longrightarrow$

bisimilarity between closed processes: $P \approx_c Q$ if there is a symmetric relation \mathcal{R} over closed processes such that $P\mathcal{R}Q$

and $P \xrightarrow{\lambda} P'$ imply there exists Q' such that

- $Q \xrightarrow{\lambda} Q'$ and

- $P'\mathcal{R}Q'$

Labelled Transition System

\Longrightarrow denotes the reflexive and transitive closure of $\xrightarrow{\tau}$

$\Longrightarrow^{\lambda}$ denotes $\Longrightarrow \xrightarrow{\lambda} \Longrightarrow$

bisimilarity between closed processes: $P \approx_c Q$ if there is a symmetric relation \mathcal{R} over closed processes such that $P\mathcal{R}Q$

and $P \xrightarrow{\lambda} P'$ imply there exists Q' such that

- $Q \xrightarrow{\lambda} Q'$ and

- $P'\mathcal{R}Q'$

fully bisimilarity: $P \approx Q$, if $Ps \approx_c Qs$ for every closing substitution s that respects types

Labelled Transition System

\Longrightarrow denotes the reflexive and transitive closure of $\xrightarrow{\tau}$

$\Longrightarrow^{\lambda}$ denotes $\Longrightarrow \xrightarrow{\lambda} \Longrightarrow$

bisimilarity between closed processes: $P \approx_c Q$ if there is a symmetric relation \mathcal{R} over closed processes such that $P \mathcal{R} Q$

and $P \xrightarrow{\lambda} P'$ imply there exists Q' such that

- $Q \Longrightarrow^{\lambda} Q'$ and

- $P' \mathcal{R} Q'$

fully bisimilarity: $P \approx Q$, if $Ps \approx_c Qs$ for every closing substitution s that respects types

Soundness of Full Bisimilarity

If $P \approx Q$ then $P \cong Q$

Features of BACI

- distinction between movement and communication (ambients for movements and ports for communication)

Features of BACI

- distinction between movement and communication
- finer control of non-determinism

Features of BACI

- distinction between movement and communication
- finer control of non-determinism
- named communications with parents (allows an ambient to communicate with different parents in different types)

Features of BACI

- distinction between movement and communication
- finer control of non-determinism
- named communications with parents
- local typing (each ambient has his local view of the type of information which can be exchanged over parent and child ports)

Features of BACI

- distinction between movement and communication
- finer control of non-determinism
- named communications with parents
- local typing
- dynamic typing (ambients increase their local knowledge of their surrounding by movements)

Features of BACI

- distinction between movement and communication
- finer control of non-determinism
- named communications with parents
- local typing
- dynamic typing

Future Work (PhD thesis of P. Garralda)

- local movement rights (finer control of movements)

Future Work (PhD thesis of P. Garralda)

- local movement rights
- private port names (to model private interfaces)

Future Work (PhD thesis of P. Garralda)

- local movement rights
- private port names
- multiple ports (to have more flexible communications)

Future Work (PhD thesis of P. Garralda)

- local movement rights
- private port names
- multiple ports
- information flow analysis

Future Work (PhD thesis of P. Garralda)

- local movement rights
- private port names
- multiple ports
- information flow analysis



Future Work (PhD thesis of P. Garralda)

- local movement rights
- private port names
- multiple ports
- information flow analysis



thank you for your attention