Open Multiparty Interactions in the link-calculus

Roberto Bruni
(Pisa)

joint work with
Chiara Bodei (Pisa)
Linda Brodo (Sassari)

Roadmap

- Problem statement: intro and motivation
- A new kind of interaction
- Handling message content
- Encoding mobile ambients
- Conclusion and future work

Let’s begin

(but feel free to interrupt)

Setting

Modelling concurrent communicating systems

Process calculi approach

(some basic knowledge of CCS and pi assumed,
some details omitted)

Interaction

An interaction is an action by which (communicating) processes can influence each other

Milner's CCS interaction

co-action prefix
(output?)

$$a.P \quad | \quad \overline{a}.Q$$

action prefix
(input?)

$$\Big\downarrow {\color{blue} a \bullet \overline{a} = \tau} \text{ silent action}$$

$$P \mid Q$$

Milner's pi interaction

$$\overline{a}x.P \;\mid\; a(y).Q \xrightarrow{\tau} P \mid Q[x/y]$$

Any better abstraction?

Internet
Biology
Social networks
Autonomic systems
...

I/O is the basic form of interaction
but "one size won't fit all"

(it is possibly misleading to think otherwise:
not all interactions are mutual/reciprocal)

Would you...?

...model piano playing using dyadic interaction

Open multiparty interactions are like playing piano
(either bad or good, it does not matter)

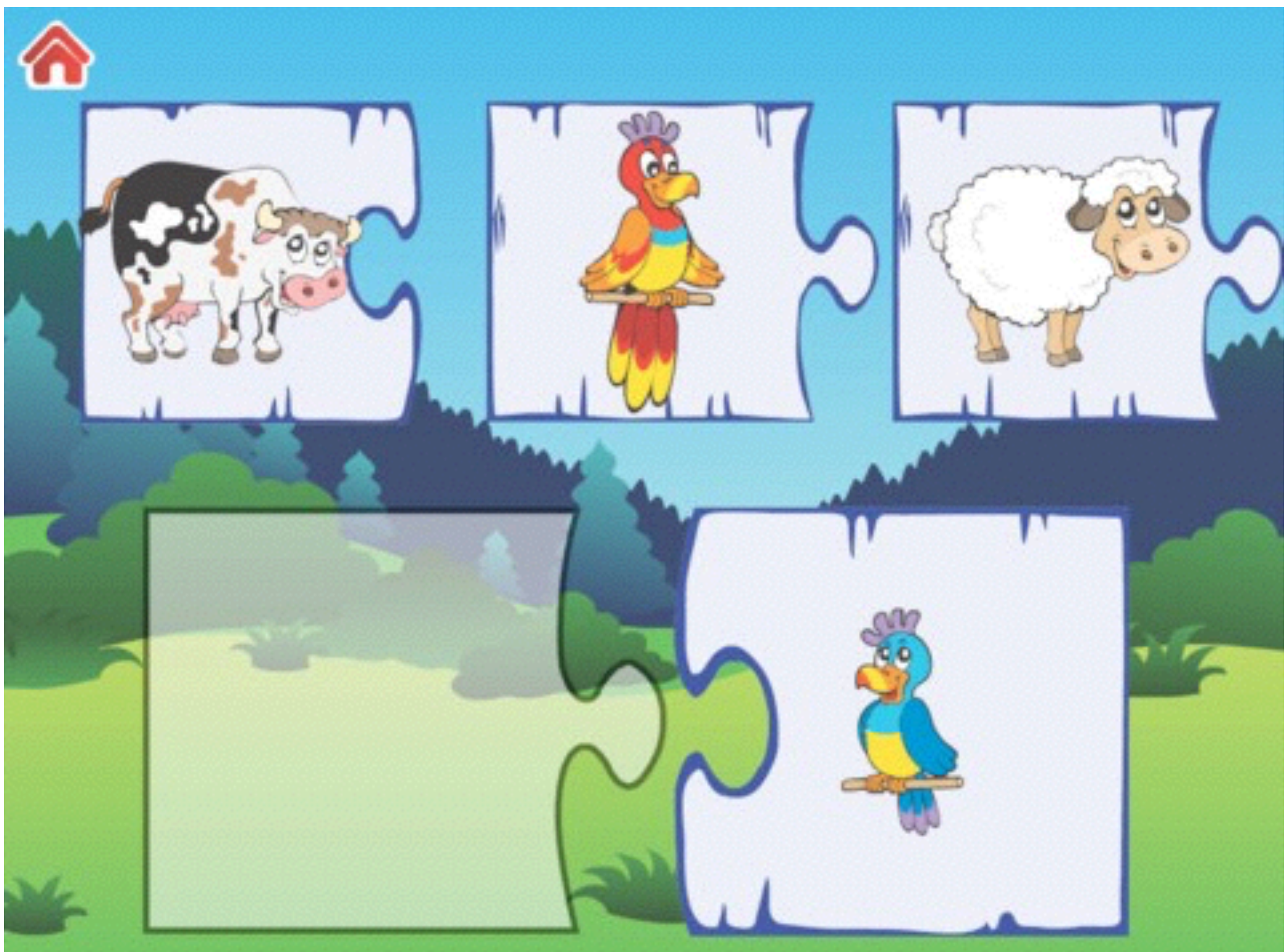
Vision

Interaction is like a puzzle:

it requires different pieces to fit together

Bold claim #1

Mutual (I/O-like) interaction is like a kid's puzzle



Multiparty interaction

An interaction is multiparty when
it involves two or more processes

Open interaction

An interaction is open when
the number of involved processes is not fixed

Our aim

Extend the theory of dyadic interactions
as little as possible
as well as possible
to deal with open multiparty interaction

Motivating example

How to encode Cardelli&Gordon's mobile ambients
(in ordinary process calculi)?

CCS/CSP:
immutable connectivity

pi:
channel mobility

mobile ambients:
mobility of nested processes
(barrier crossing)

HOpi:
flat process mobility

Process algebra ops

$\mathbf{0}$	nil
$\mu.P$	action prefix
$P + Q$	sum
$P \mid Q$	parallel
$(\nu a)P$	restriction
$!P$	replication
X	process variable
$\mathsf{rec}\ X.P$	recursive process
$P[\phi]$	renaming

Named, mobile, active, hierarchical ambients

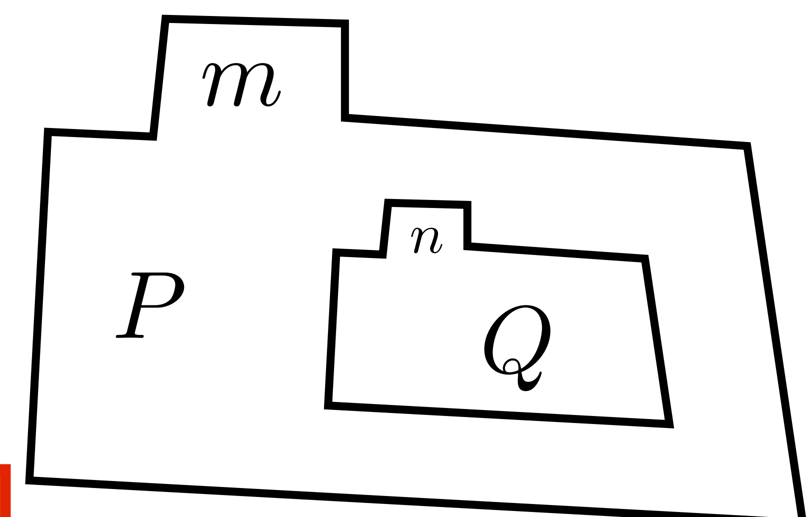
An ambient is a place where computation happens
An ambient defines some sort of boundary

An ambient has a name
An ambient has a collection of local processes
An ambient has a collection of sub-ambients

Ambients are subject to capabilities:
Ambients can move in/out of other ambients
Ambients can dissolve

(Pure) Ambient calculus

$$
\begin{array}{lrl}
P ::= & \mathbf{0} & \text{nil} \\
 & \boxed{m[P]} & \text{ambient} \\
 & \boxed{M.P} & \text{exercise a capability} \\
 & P \mid Q & \text{parallel} \\
 & (\nu a)P & \text{restriction} \\
 & !P & \text{replication} \\
 & & \\
M ::= & \mathsf{in}\ m & \text{entry capability} \\
 & \mathsf{out}\ m & \text{exit capability} \\
 & \mathsf{open}\ m & \text{open capability}
\end{array}
$$



Ambient calculus: semantics

Structural congruence

$P \equiv P$	$Q \equiv P \Rightarrow P \equiv Q$	$P \equiv Q, Q \equiv R \Rightarrow P \equiv R$
$P \mid \mathbf{0} \equiv P$	$P \mid Q \equiv Q \mid P$	$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$
$(\nu n)\mathbf{0} \equiv \mathbf{0}$	$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$
	$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q, \text{ if } n \notin fn(P)$	$P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q$
$!P \equiv P \mid !P$	$(\nu n)(m[P]) \equiv m[(\nu n)P], \text{ if } n \neq m$	$P \equiv Q \Rightarrow n[P] \equiv n[Q]$

Ambient calculus: semantics

Structural congruence

$$
\begin{array}{lll}
P \equiv P & Q \equiv P \Rightarrow P \equiv Q & P \equiv Q, Q \equiv R \Rightarrow P \equiv R \\
P \mid \mathbf{0} \equiv P & P \mid Q \equiv Q \mid P & (P \mid Q) \mid R \equiv P \mid (Q \mid R) \\
(\nu n)\mathbf{0} \equiv \mathbf{0} & (\nu n)(\nu m)P \equiv (\nu m)(\nu n)P & P \equiv Q \Rightarrow P|R \equiv Q|R \\
 & (\nu n)(P \mid Q) \equiv P \mid (\nu n)Q, \text{ if } n \notin fn(P) & P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q \\
!P \equiv P \mid !P & (\nu n)(m[P]) \equiv m[(\nu n)P], \text{ if } n \neq m & P \equiv Q \Rightarrow n[P] \equiv n[Q]
\end{array}
$$

Reduction semantics

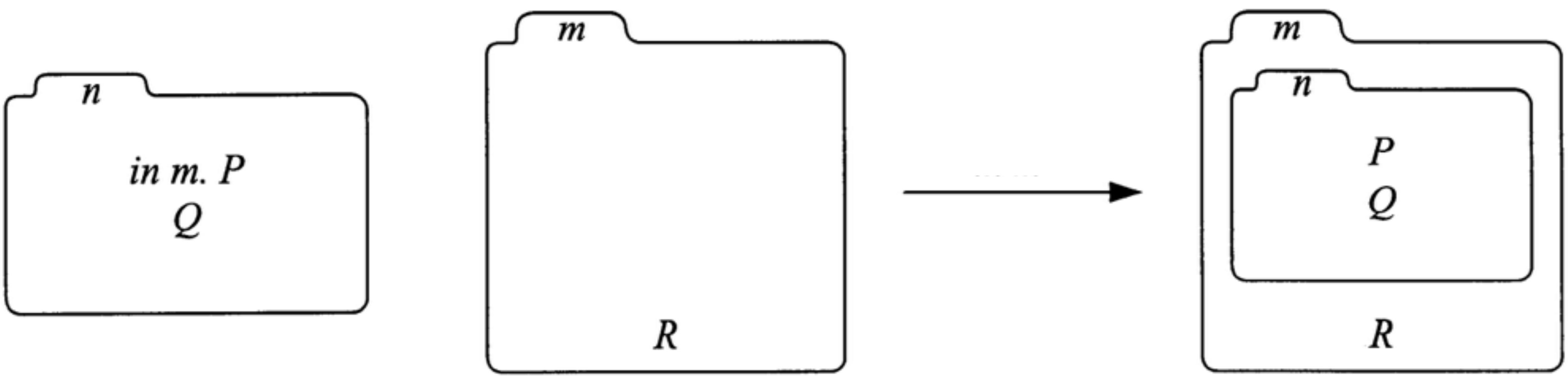
$$\frac{}{n[\mathsf{in}\, m.P \mid Q] \mid m[R] \rightarrow m[n[P \mid Q] \mid R]}\ \boxed{\text{(In)}} \qquad \frac{}{m[n[\mathsf{out}\, m.P \mid Q] \mid R] \rightarrow n[P \mid Q] \mid m[R]}\ \boxed{\text{(Out)}}$$

$$\frac{}{\mathsf{open}\, n.P \mid n[Q] \rightarrow P \mid Q}\ \boxed{\text{(Open)}} \qquad \frac{P \rightarrow Q}{(\nu n)P \rightarrow (\nu n)Q}\ \text{(Res)} \qquad \frac{P \rightarrow Q}{n[P] \rightarrow n[Q]}\ \text{(Amb)}$$

$$\frac{P \rightarrow Q}{P \mid R \rightarrow Q \mid R}\ \text{(Par)} \qquad \frac{P' \equiv P \qquad P \rightarrow Q \qquad Q \equiv Q'}{P' \rightarrow Q'}\ \text{(Cong)}$$

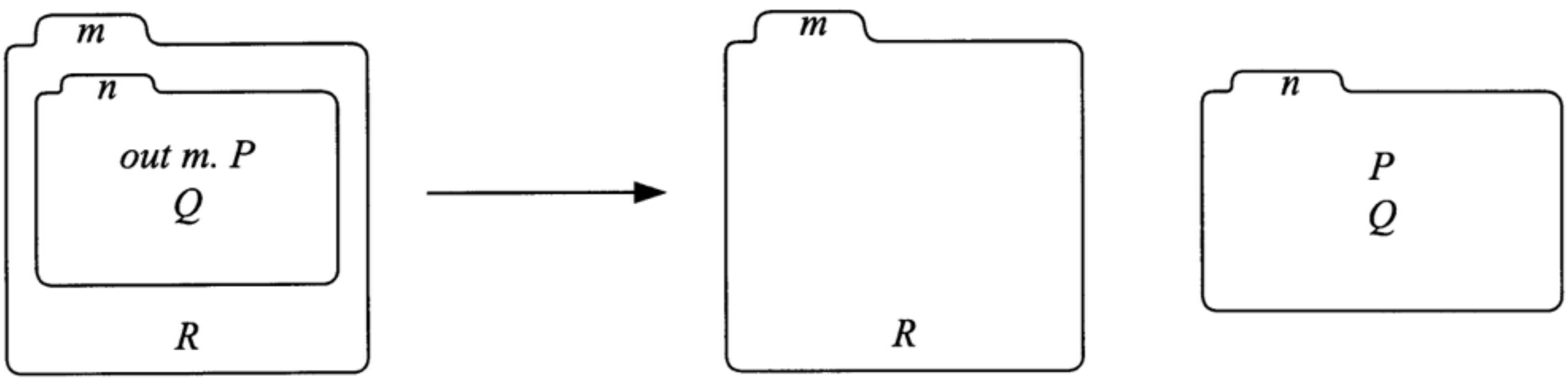
(In)

$$n[\mathsf{in}\, m.P \mid Q] \mid m[R] \rightarrow m[n[P \mid Q] \mid R]$$



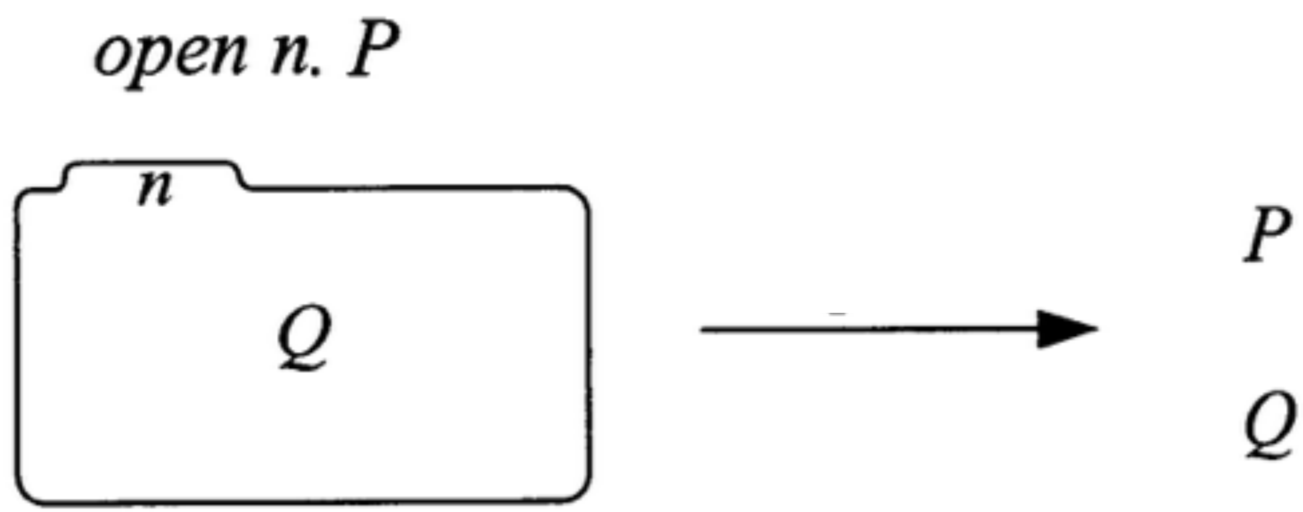
(Out)

$$m[n[\mathsf{out}\, m.P \mid Q] \mid R] \rightarrow n[P \mid Q] \mid m[R]$$



(Open)

$$\text{open}\, n.P \mid n[Q] \rightarrow P \mid Q$$

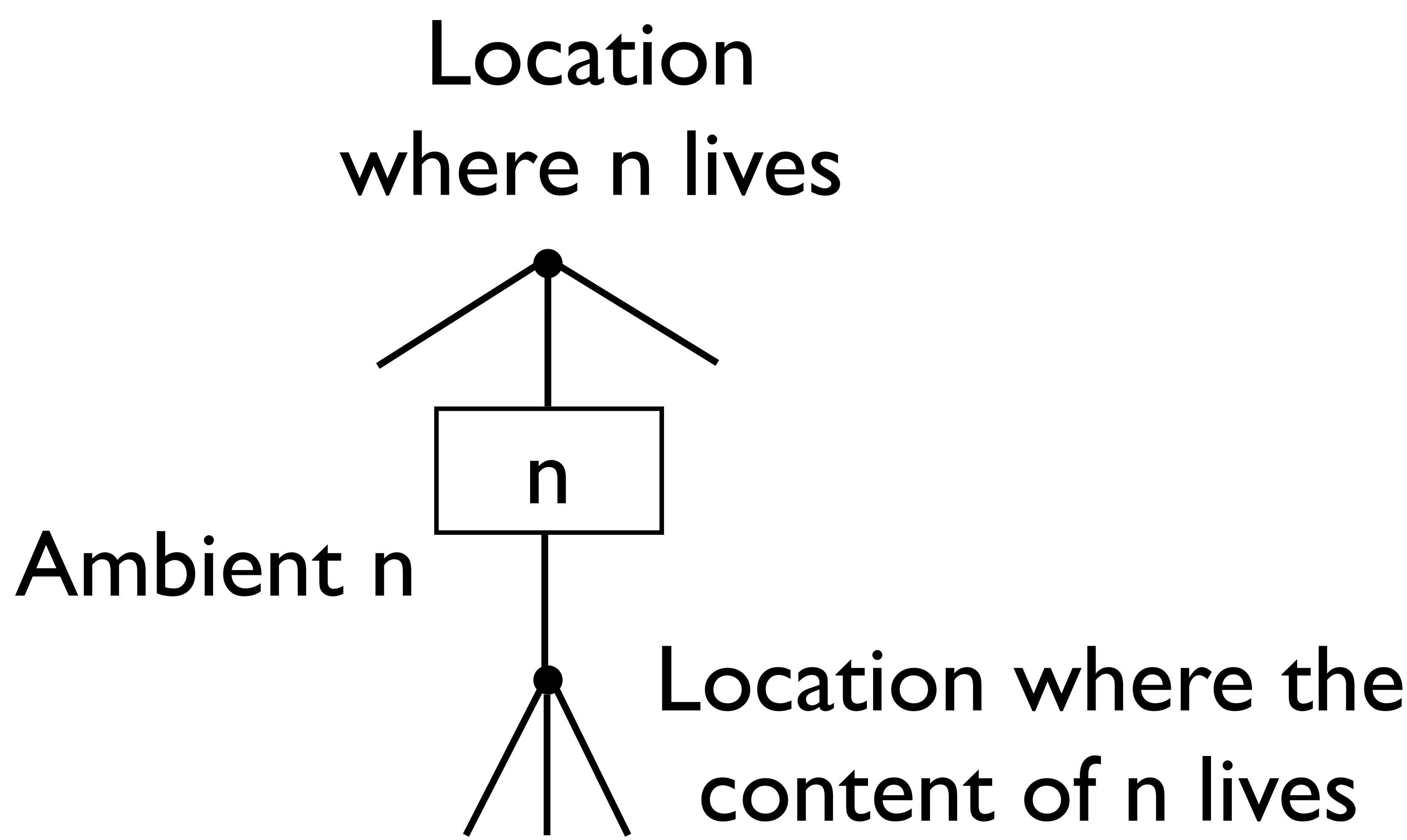


A challenge for the audience

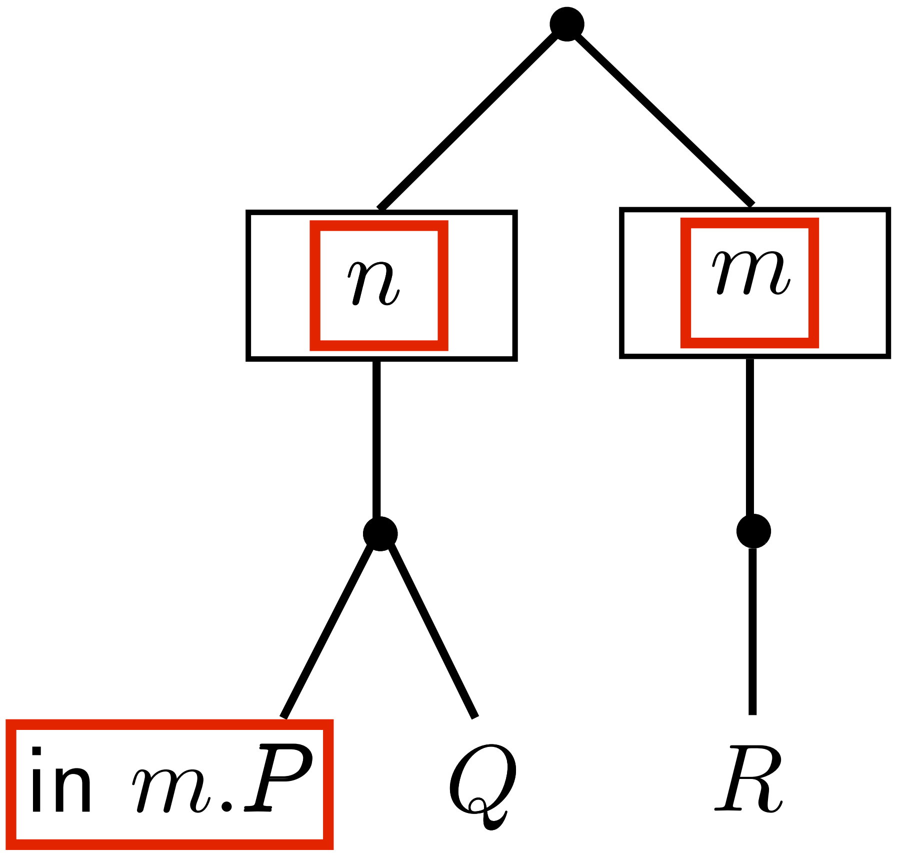
Why is it difficult to encode ambients into pi?
(How would you proceed?)

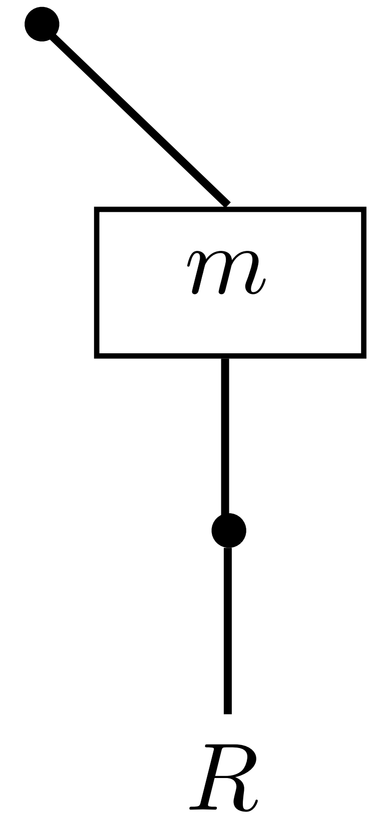
Personal guess:
it is just because ambient-like interaction
is inherently non-dyadic!

Ambients as graphs



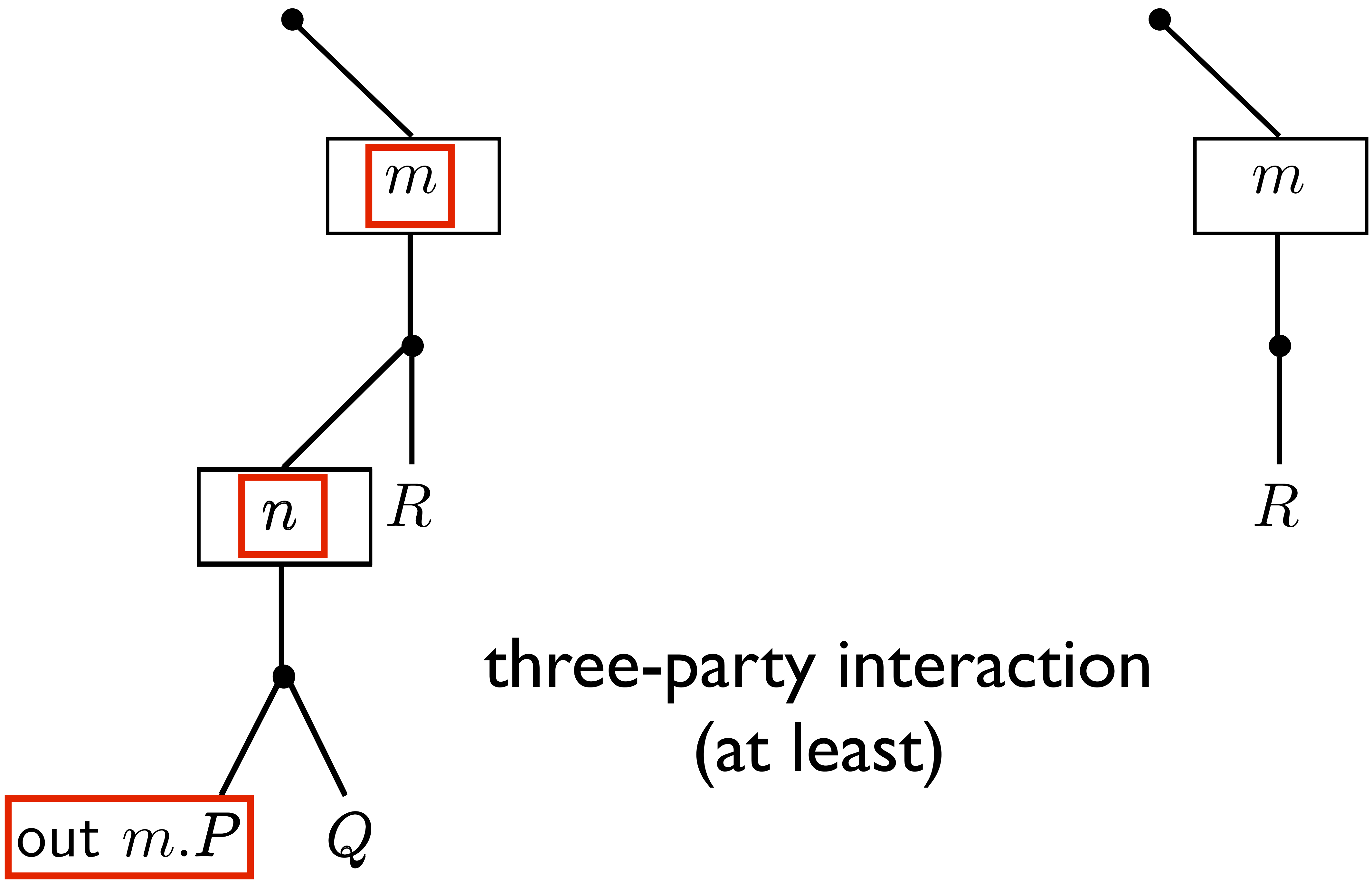
(In), again





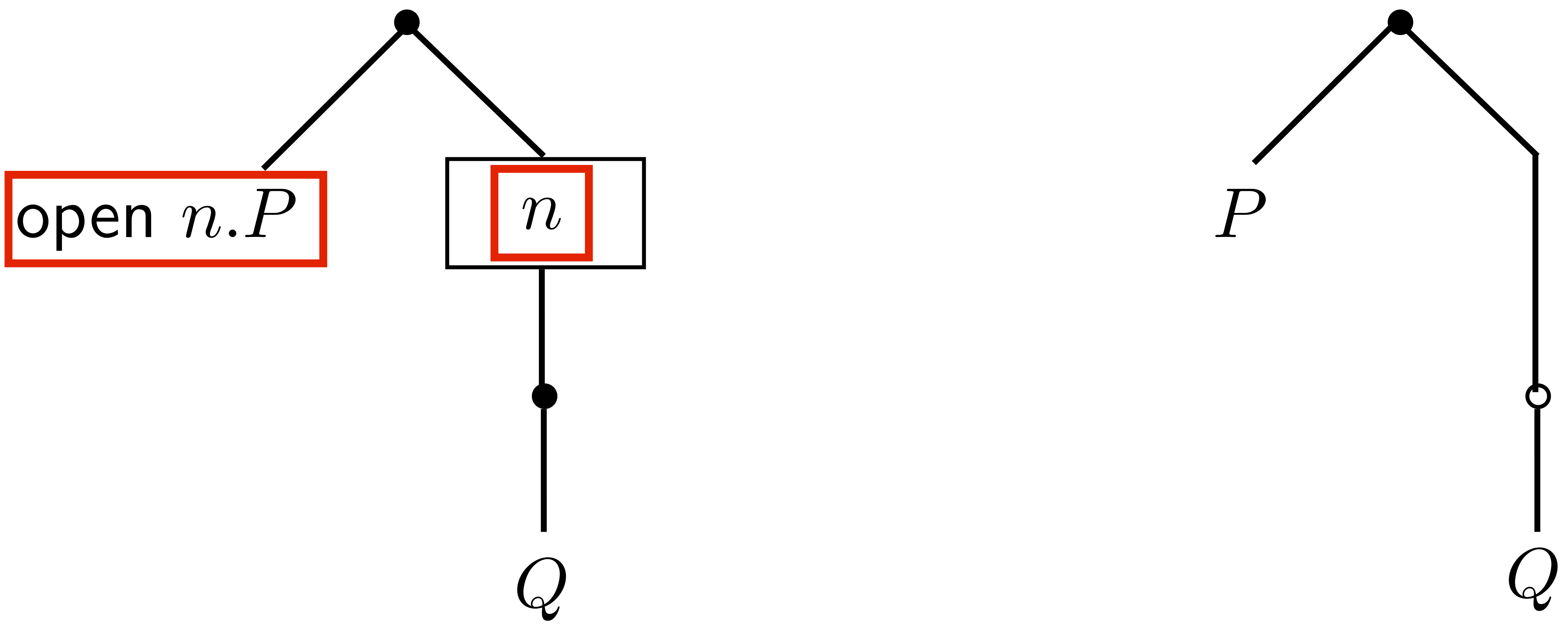
three-party interaction
(at least)

(Out), again



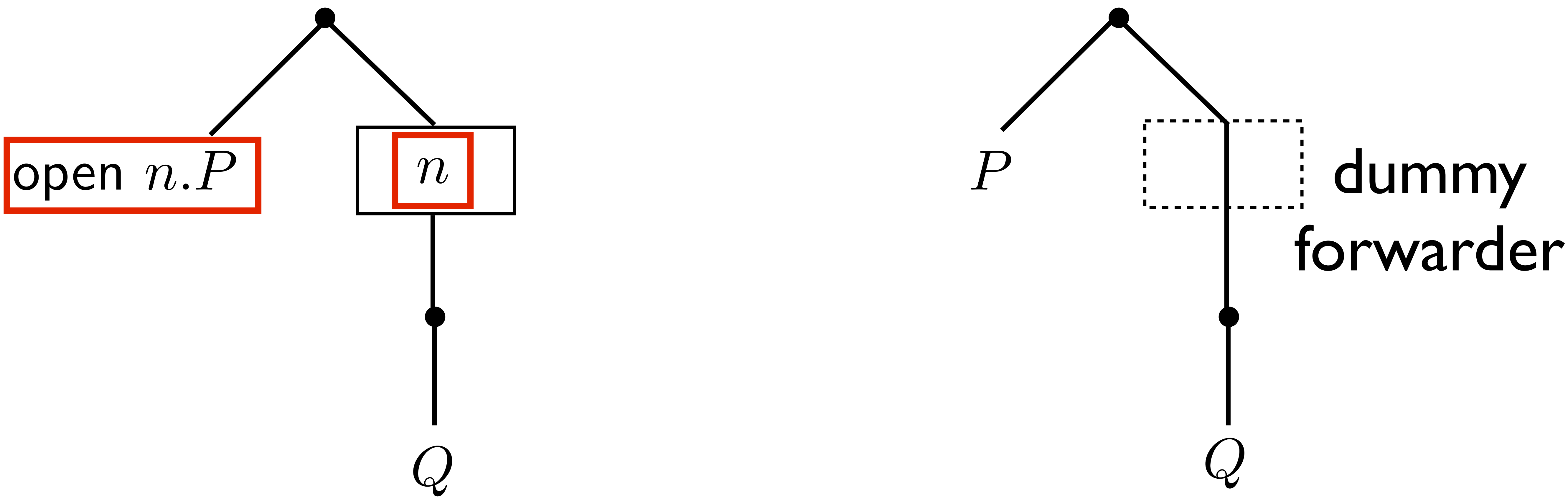
three-party interaction
(at least)

(Open), again



looks like a two-party interaction, but it is not!
It is open! (accident of fate):
many processes (Q) change location at once

(Open), yet another



ok, now it is a two-party interaction

But (In) and (Out) become open!

they must involve as many fwd-ers as needed

Some consequences

Proposed encoding are either quite involved
or centralized (unnecessary bottle-necks)

LTS semantics for ambients are ad-hoc
(to say the least)
and based on HO labels

Some references

- Fabio Gadducci, Giacoma Valentina Monreale: A decentralised graphical implementation of mobile ambients. J. Log. Algebr. Program. 80(2): 113-136 (2011)
- Linda Brodo: On the Expressiveness of the pi-Calculus and the Mobile Ambients. AMAST 2010: 44-59
- Gabriel Ciobanu, Vladimir A. Zakharov: Encoding Mobile Ambients into the pi -Calculus. Ershov Memorial Conference 2006: 148-165
- Linda Brodo, Pierpaolo Degano, Corrado Priami: Reflecting Mobile Ambients into the p-Calculus. Global Computing 2003: 25-56
- Cédric Fournet, Jean-Jacques Lévy, Alan Schmitt: An Asynchronous, Distributed Implementation of Mobile Ambients. IFIP TCS 2000: 348-364

Roadmap

- Problem statement: intro and motivation
- A new kind of interaction
- Handling message content
- Encoding mobile ambients
- Conclusion and future work

(Recall our aim)

Extend the theory of dyadic interactions
as little as possible
as well as possible
to deal with open multiparty interaction

and to encode mobile ambients

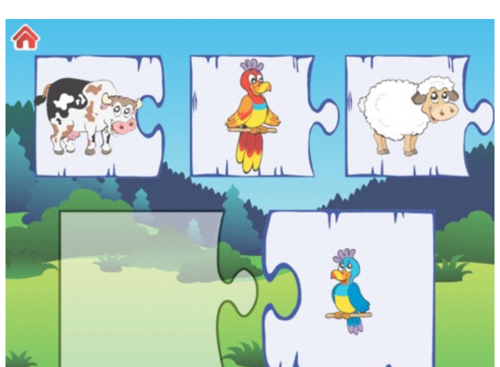
Guidelines

Keep the syntax simple
Do not move the complexity to SOS rules

All we need is just a proper synchronization algebra

Linked interaction

We regard an interaction as a chain of links
(still a kid's puzzle after all)



Process algebra ops

$\mathbf{0}$	nil
$\boxed{\mu.P}$	$\boxed{\text{action prefix}}$
$P + Q$	sum
$P \mid Q$	parallel
$(\nu a)P$	restriction
$!P$	replication
X	process variable
rec $X.P$	recursive process
$P[\phi]$	renaming

We take as action the offering of a link

Notation

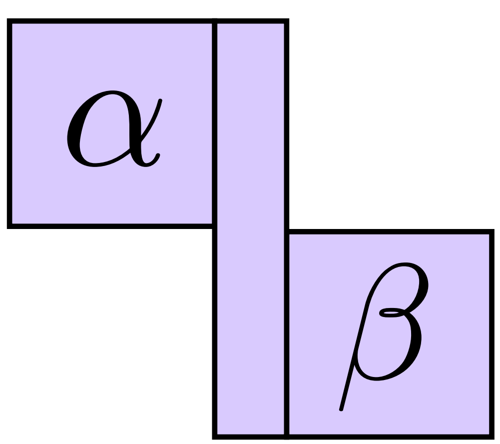
a interaction over a

τ silent interaction

$*$ any interaction (only in labels)

Link

${}^{\alpha}\diagdown_{\beta}$ From α to β



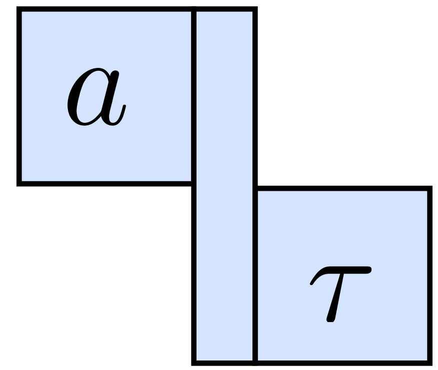
Valid:

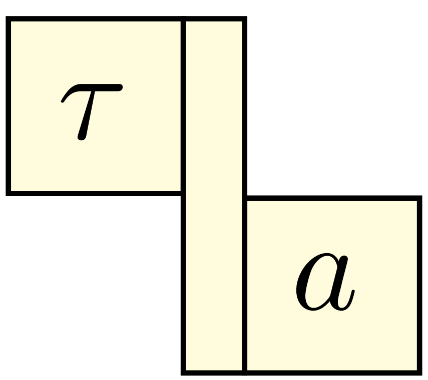
$$\alpha = \beta = * \text{ or } \alpha, \beta \neq *$$

Virtual if ${}^{*}\diagdown_{*}$

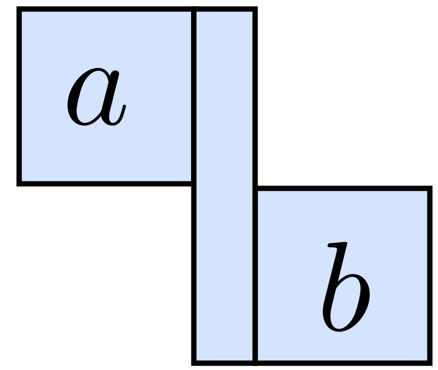
Solid (otherwise)

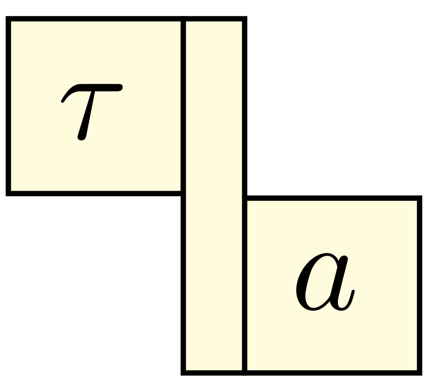
Examples: CCS-like

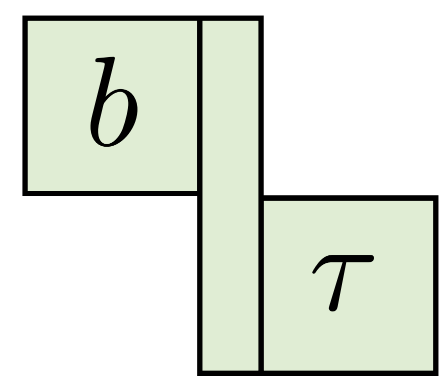




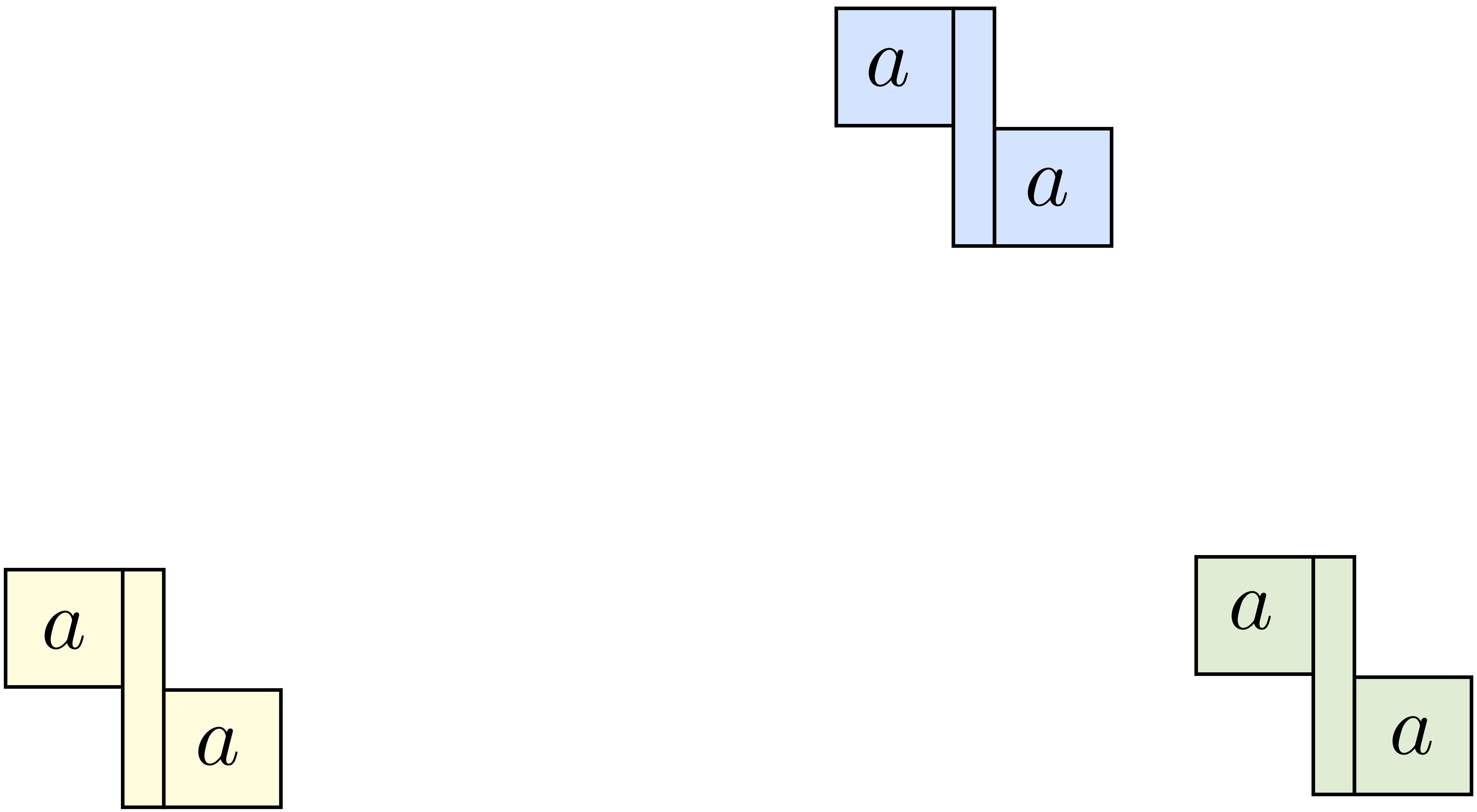
Examples: three party







Examples: CSP



Link chain

$$^{\alpha_1}\backslash_{\beta_1} \ ^{\alpha_2}\backslash_{\beta_2} \ \ldots \ ^{\alpha_n}\backslash_{\beta_n}$$

such that:

$$\beta_i, \alpha_{i+1} \notin \{\tau, *\} \text{ implies } \beta_i = \alpha_{i+1}$$

$$\beta_i = \tau \text{ iff } \alpha_{i+1} = \tau$$

$$\forall i. \alpha_i, \beta_i \in \{\tau, *\} \text{ implies } \forall i. \alpha_i = \beta_i = \tau$$

Link chain: terminology

$$ {}^{\alpha_1}\backslash_{\beta_1} \; {}^{\alpha_2}\backslash_{\beta_2} \; \cdots \; {}^{\alpha_n}\backslash_{\beta_n} $$

Solid:

if all its links are so

Simple:

if it contains exactly one solid link

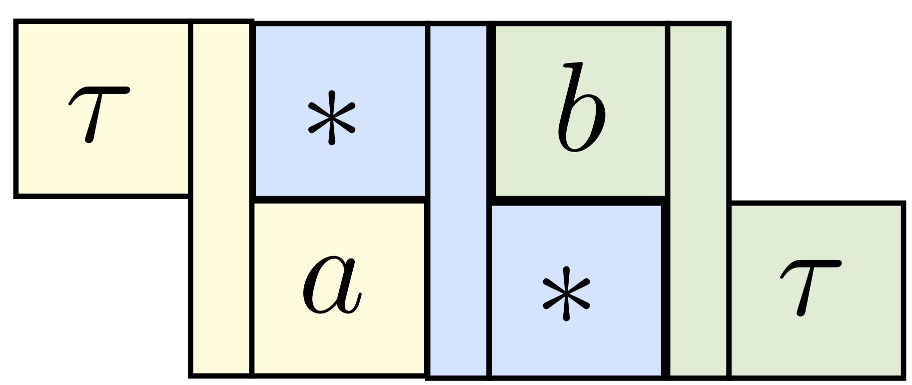
$\ell \in s$:

s is simple and ℓ is the only solid link in s

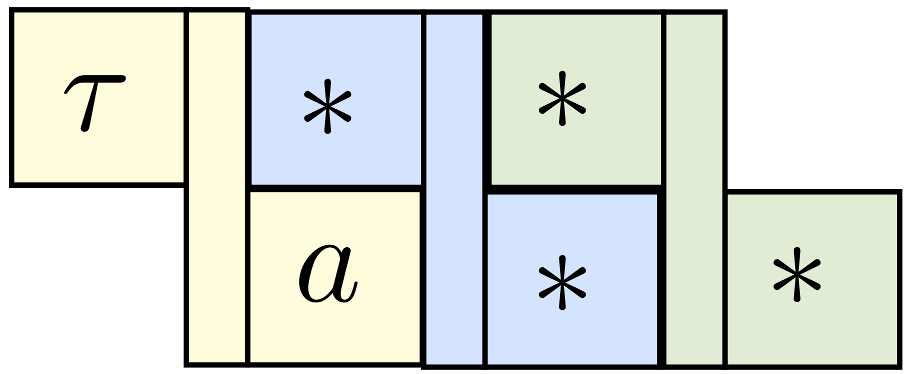
Examples: non solid

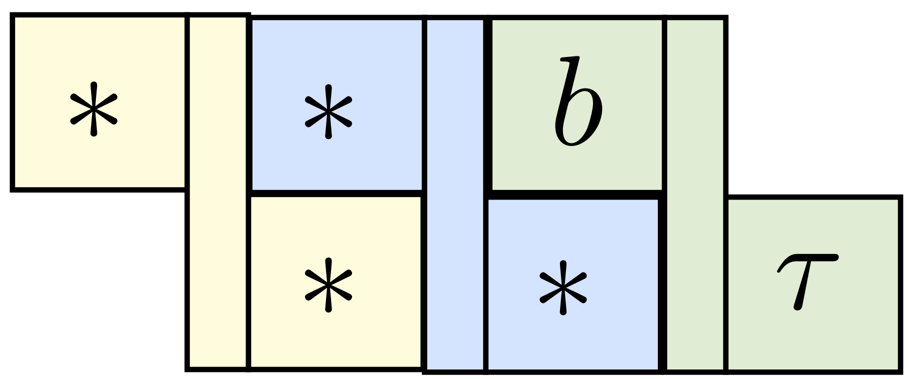
Virtual links ${}^{*}\backslash_{*}$

can be read as missing pieces of the puzzle

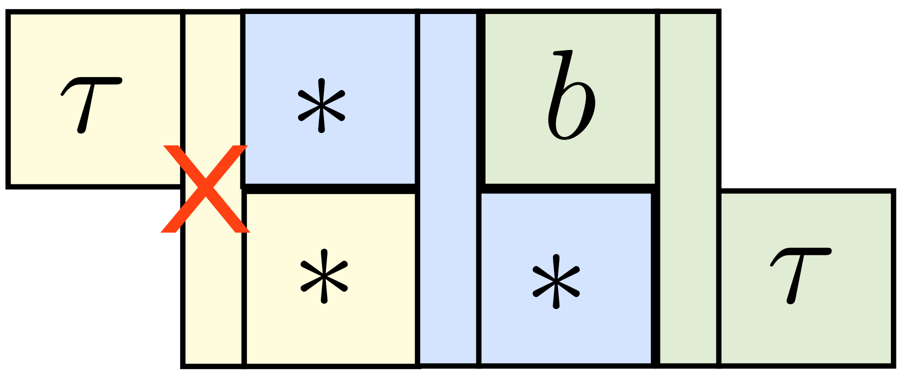


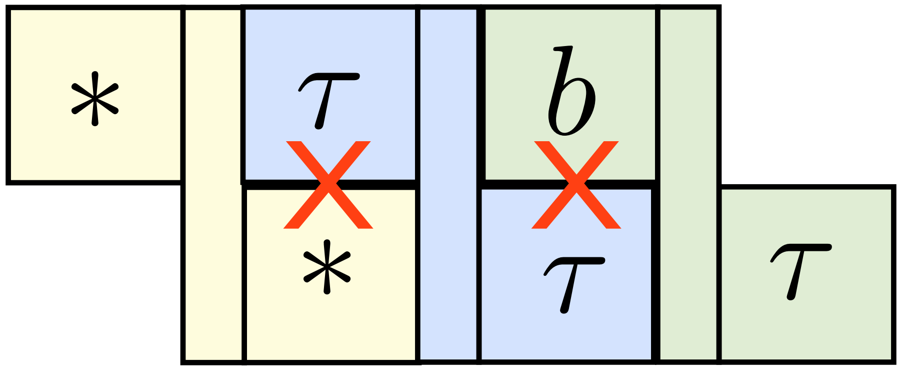
Examples: simple





Counter-examples





Merge

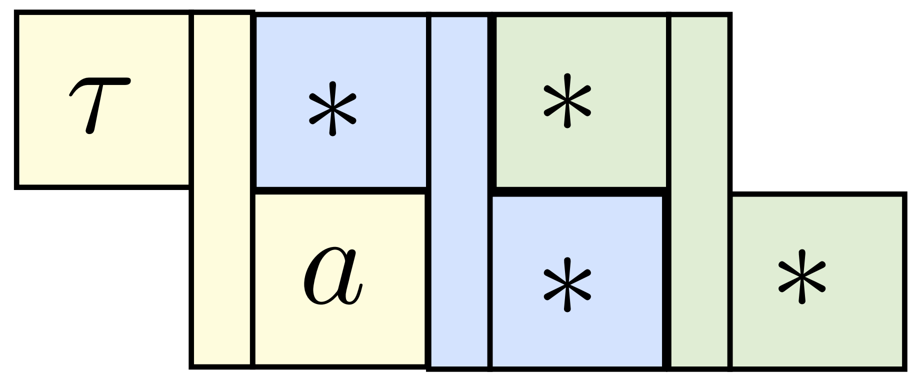
(All the ops we show are strict)

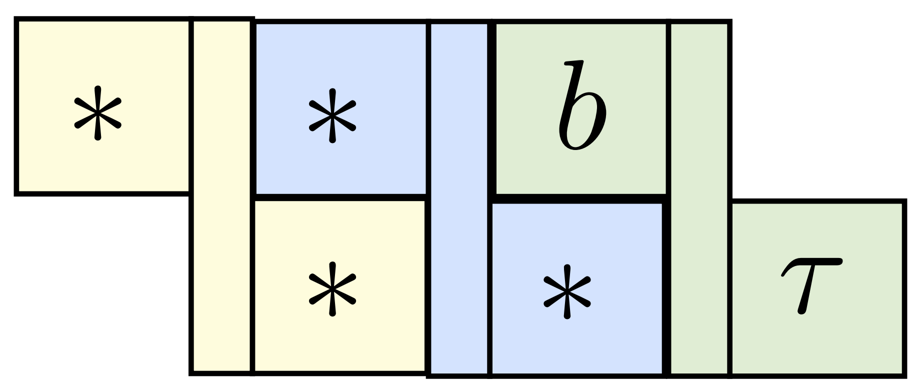
$$\alpha \bullet \beta \quad \triangleq \quad \begin{cases} \alpha & \text{if } \beta = * \\ \beta & \text{if } \alpha = * \\ \bot & \text{otherwise} \end{cases}$$

$$^{\alpha}\backslash_{\beta} \bullet {}^{\alpha'}\backslash_{\beta'} \quad \triangleq \quad \begin{cases} ^{(\alpha \bullet \alpha')}\backslash_{(\beta \bullet \beta')} & \text{if } \alpha \bullet \alpha', \beta \bullet \beta' \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

The definition extends to chains element-wise
(the result is undefined if the outcome is not valid)

Examples: merge





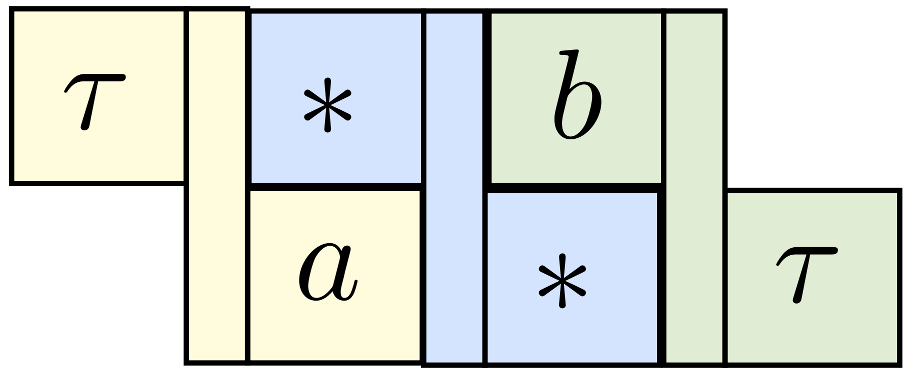
Restriction

$$(\nu a)({}_{\beta}{}^{\alpha}) \triangleq \begin{cases} {}_{\beta}{}^{\alpha} & \text{if } \alpha, \beta \neq a \\ {}_{\tau}{}^{\tau} & \text{if } \alpha = \beta = a \\ \perp & \text{otherwise} \end{cases}$$

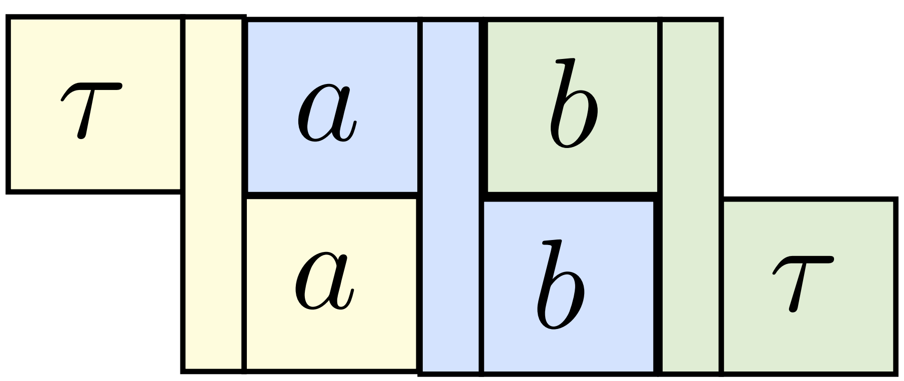
$$(\nu a)({}^{\alpha_1}\backslash_{\beta_1}\ {}^{\alpha_2}\backslash_{\beta_2}\ \ldots\ {}^{\alpha_n}\backslash_{\beta_n}\ ) \triangleq$$

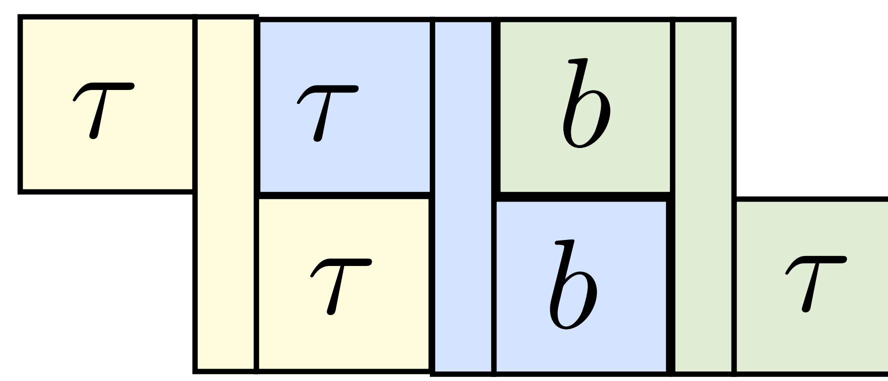
$$\begin{cases} {}^{\alpha_1}\backslash(\nu a)({}_{\beta_1}{}^{\alpha_2})\backslash\ldots\backslash(\nu a)({}_{\beta_{n-1}}{}^{\alpha_n})\backslash_{\beta_n} & \text{if } \alpha_1, \beta_n \neq a \\ \\ \perp & \text{otherwise} \end{cases}$$

Examples: restriction

(νa) 

$= \bot$

(νa) 

$=$ 

(Relevant) SOS rules

(solid) (simple)

$$\frac{\ell \in s}{\ell.P \xrightarrow{s} P}\ (\mathrm{Act}) \qquad \frac{P \xrightarrow{s} P'}{(\nu a)P \xrightarrow{(\nu a)s} (\nu a)P'}\ (\mathrm{Res})$$

$$\frac{P \xrightarrow{s} P'}{P|Q \xrightarrow{s} P'|Q}\ (\mathrm{Lpar}) \qquad \frac{P \xrightarrow{s} P' \qquad Q \xrightarrow{s'} Q'}{P|Q \xrightarrow{s \bullet s'} P'|Q'}\ (\mathrm{Com})$$

(look as ordinary CCS rules)

Example

$$(\nu a)({}^{\tau}\backslash_{a}.P \mid {}^{a}\backslash_{b}.Q \mid {}^{b}\backslash_{\tau}.R)$$

$$\dfrac{\dfrac{\dfrac{{}^{\tau}\backslash_{a}.P \xrightarrow{{}^{\tau}\backslash_{a}^{*}\backslash_{*}^{*}\backslash_{*}} P \qquad {}^{a}\backslash_{b}.Q \xrightarrow{{}^{*}\backslash_{*}^{a}\backslash_{b}^{*}\backslash_{*}} Q}{{}^{\tau}\backslash_{a}.P \mid {}^{a}\backslash_{b}.Q \xrightarrow{{}^{\tau}\backslash_{a}^{a}\backslash_{b}^{*}\backslash_{*}} P \mid Q} \qquad {}^{b}\backslash_{\tau}.R \xrightarrow{{}^{*}\backslash_{*}^{*}\backslash_{*}^{b}\backslash_{\tau}} R}{{}^{\tau}\backslash_{a}.P \mid {}^{a}\backslash_{b}.Q \mid {}^{b}\backslash_{\tau}.R \xrightarrow{{}^{\tau}\backslash_{a}^{a}\backslash_{b}^{b}\backslash_{\tau}} P \mid Q \mid R}}{(\nu a)({}^{\tau}\backslash_{a}.P \mid {}^{a}\backslash_{b}.Q \mid {}^{b}\backslash_{\tau}.R) \xrightarrow{{}^{\tau}\backslash_{\tau}^{\tau}\backslash_{b}^{b}\backslash_{\tau}} (\nu a)(P \mid Q \mid R)}$$

Fact

The process algebra of linked interactions
is a straightforward extension of CCS
It includes CCS as a sub-calculus

Finer (bisimilarity over the) LTS wrt CCS:
three kinds of meaningful observables

$${}^{\tau}\backslash_{a} \qquad\qquad {}^{\tau}\backslash^{*}_{a}\backslash^{b}_{*}\backslash_{\tau} \qquad\qquad {}^{b}\backslash_{\tau}$$

$${}^{\tau}\backslash_{a}.{}^{b}\backslash_{\tau} \;+\; {}^{b}\backslash_{\tau}.{}^{\tau}\backslash_{a} \;\not\sim\; {}^{\tau}\backslash_{a} \mid {}^{b}\backslash_{\tau}$$

$${}^{\tau}\backslash_{a}.{}^{\tau}\backslash_{b} \;+\; {}^{\tau}\backslash_{b}.{}^{\tau}\backslash_{a} \;\sim\; {}^{\tau}\backslash_{a} \mid {}^{\tau}\backslash_{b}$$

Some references

- U. Montanari and M. Sammartino. Network conscious pi-calculus. Technical Report TR-12-01, Computer Science Department, University of Pisa, 2012.

Roadmap

- Problem statement: intro and motivation
- A new kind of interaction
- Handling message content
- Encoding mobile ambients
- Conclusion and future work

Name mobility

Ready to handle mobile ambients interactions

but we need to update locations of processes
when ambient moves

some form of name mobility is needed

Handling name mobility

Aim: introduce polyadic communication and reuse/rely on pi as much as possible

One possibility: ${}^{a(\widetilde{x})}\backslash_{b\widetilde{y}}.P$

each link receive some arguments and send some names... too complex

Another possibility: ${}^{a}\backslash_{b}\widetilde{x}.P$

each link in the chain carry the same list of arguments... but with different (send/receive) capabilities

Separation of concerns

$$P, Q, R ::= \cdots \mid \ell t.P$$

This way we separate
the interaction mechanism ℓ
from
the name passing mechanism t

(We formalize them separately and
then fit them together)

No need to reinvent the wheel

We can easily borrow from pi
the name handling machinery
(and free it from dyadic interaction legacy)

$P \mid a(x).Q$	(waits input from P)	$P' \mid Q[b/x]$
$P \mid \overline{a}x.Q$	(outputs to P)	$P' \mid Q$
$P \mid (\nu x)\overline{a}x.Q$	(extrudes to P)	$(\nu y)P' \mid Q[y/x]$

Tuple

$$t = \langle \widetilde{w} \rangle \qquad w ::= \begin{array}{ll} x & \text{value (output)} \\ \underline{x} & \text{variable (input)} \end{array}$$

variables are instantiated by values

values are used for matching arguments

$$\begin{array}{ccc} \langle n, & m, & \underline{x} \rangle \\ \downarrow & = & \uparrow \\ \langle \underline{y}, & m, & k \rangle \end{array}$$

Assigns n to y
Matches m with m
Assigns k to x

Extrusion

an argument in a tuple can be extruded if it is not already annotated

extruded arguments are overlined

$$(\nu a)w \triangleq \begin{cases} \perp & \text{if } w = \overline{a} \vee w = \underline{a} \\ \overline{a} & \text{if } w = a \\ w & \text{otherwise} \end{cases}$$

$$(\nu a)\langle w_1, \ldots, w_n \rangle \triangleq \begin{cases} \langle (\nu a)w_1, \ldots, (\nu a)w_n \rangle & \text{if } \forall i \in [1,n].(\nu a)w_i \neq \perp \\ \perp & \text{otherwise} \end{cases}$$

$$(\nu a)(st) \triangleq \begin{cases} ((\nu a)s)((\nu a)t) & \text{if } (\nu a)s \neq \perp \wedge (\nu a)t \neq \perp \\ \perp & \text{otherwise} \end{cases}$$

Merge

$$w \bullet w' \triangleq \begin{cases} w & \text{if } (w = w' = v) \vee (w = w' = \underline{v}) \\ v & \text{if } (w = v \wedge w' = \underline{v}) \vee (w = \underline{v} \wedge w' = v) \\ \overline{v} & \text{if } (w = \overline{v} \wedge w' = \underline{v}) \vee (w = \underline{v} \wedge w' = \overline{v}) \\ \bot & \text{otherwise} \end{cases}$$

$$\langle w_1, \ldots, w_n \rangle \bullet \langle w'_1, \ldots, w'_n \rangle \triangleq \begin{cases} \langle w_1 \bullet w'_1, \ldots, w_n \bullet w'_n \rangle & \text{if } \forall i \in [1, n]. w_i \bullet w'_i \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

$$st \bullet s't' \triangleq \begin{cases} (s \bullet s')(t \bullet t') & \text{if } s \bullet s' \neq \bot \wedge t \bullet t' \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

(Relevant) SOS rules

$$\frac{\ell \in s \qquad g = t\rho}{\ell t.P \xrightarrow{sg} P\rho}\;(\mathrm{Act})$$

(a appears in g)

$$\frac{P \xrightarrow{sg} P' \qquad a \notin g}{(\nu a)P \xrightarrow{(\nu a)sg} (\nu a)P'}\;(\mathrm{Res}) \qquad \frac{P \xrightarrow{sg} P' \qquad a \in g}{(\nu a)P \xrightarrow{(\nu a)sg} P'}\;(\mathrm{Open})$$

(analogous to (early) pi rules)

(Relevant) SOS rules

(extruded names of g)

$$\frac{P \xrightarrow{sg} P' \qquad ex(g) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{sg} P'|Q} \text{(Lpar)}$$

$$\frac{P \xrightarrow{sg} P' \qquad Q \xrightarrow{s'g'} Q' \qquad \overset{ex(g)\cap fn(Q) = ex(g')\cap fn(P) = \emptyset}{s \bullet s' \text{ is not solid}}}{P|Q \xrightarrow{sg \bullet s'g'} P'|Q'} \text{(Com)}$$

$$\frac{P \xrightarrow{sg} P' \qquad Q \xrightarrow{s'g'} Q' \qquad vars(g \bullet g') = \emptyset \quad \overset{ex(g)\cap fn(Q) = ex(g')\cap fn(P) = \emptyset}{s \bullet s' \text{ is solid}}}{P|Q \xrightarrow{s \bullet s'} (\nu\, ex(g \bullet g'))(P'|Q')} \text{(Close)}$$

(analogous to (early) pi rules)

Fact

The process calculus of linked interactions with name mobility is a straightforward extension of pi
It includes pi as a sub-calculus

Finer (bisimilarity over the) LTS wrt pi
(but it is a congruence)

Some references

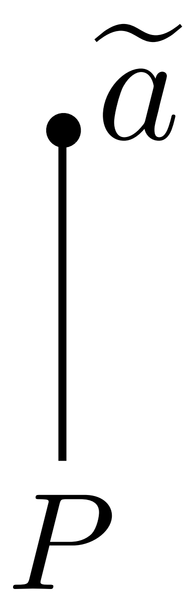
- Roberto Bruni, Ivan Lanese: Parametric synchronizations in mobile nominal calculi. Theor. Comput. Sci. 402(2-3): 102-119 (2008)
- Marco Carbone, Sergio Maffeis: On the Expressive Power of Polyadic Synchronisation in pi-calculus. Nord. J. Comput. 10(2): 70-98 (2003)

Roadmap

- Problem statement: intro and motivation
- A new kind of interaction
- Handling message content
- Encoding mobile ambients
- Conclusion and future work

Encoding mobile ambients

$[\![P]\!]_{\tilde{a}}$



a_{in} requests from in capability

$a_{[in]}$ requests from an ambient with in capability inside

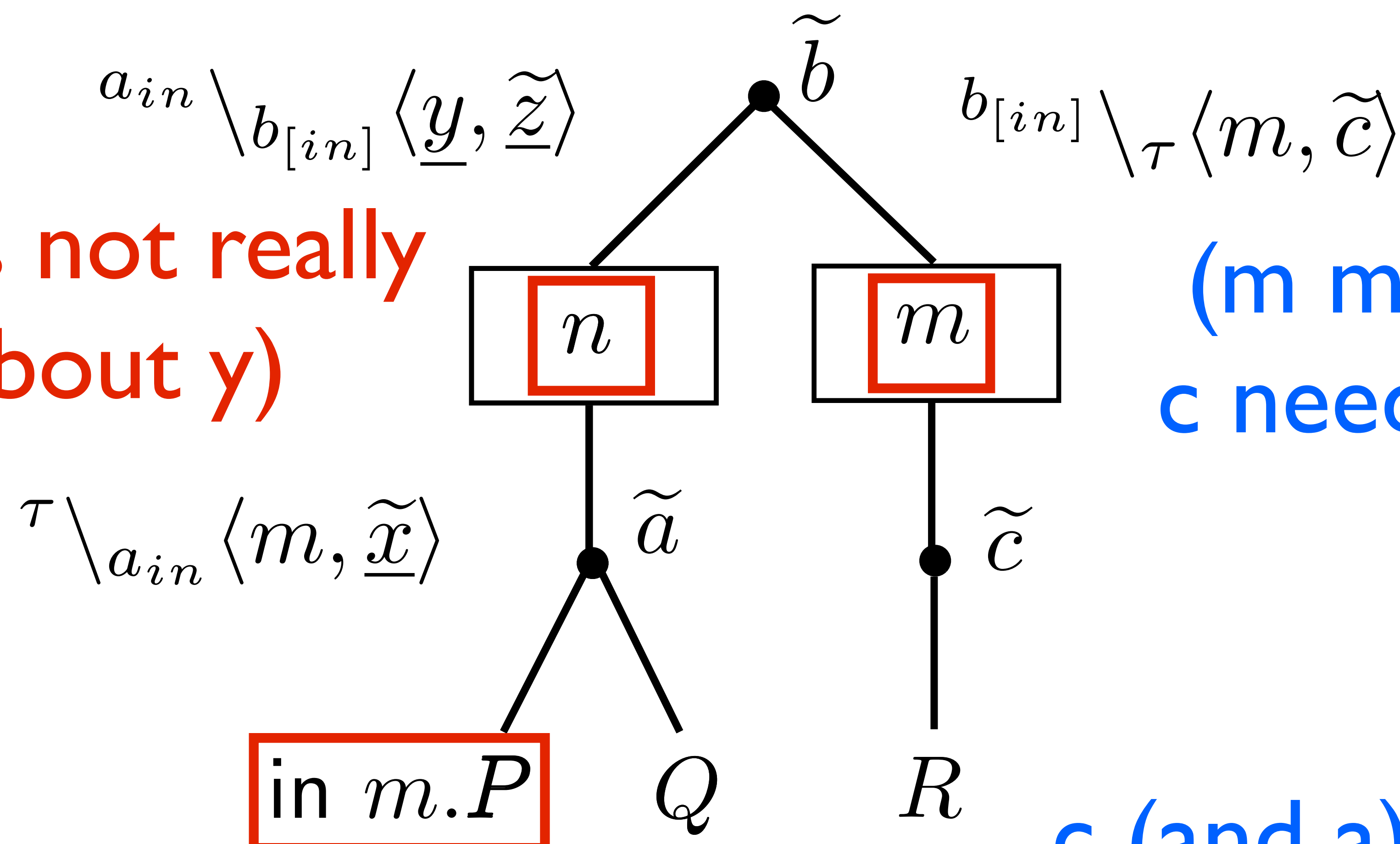
a_{out} requests from out capability

$a_{[out]}$ requests from an ambient with out capability inside

a_{opn} requests from open capability

Sketch of the idea

$$ {}^{\tau}\backslash^{a_{in}}_{a_{in}}\backslash^{b_{[in]}}_{b_{[in]}}\backslash_{\tau}\langle m, \widetilde{c}\rangle $$



(n[ ] does not really care about y)

(m must match, c needed by n[ ])

(P does not really care about x)

c (and a) are typically restricted: c must be extruded

Desiderata

$$P \to P' \quad \text{implies} \quad [\![P]\!]_{\tilde{a}} \to [\![P']\!]_{\tilde{a}}$$

$$[\![P]\!]_{\tilde{a}} \to Q \ \text{ implies } \exists P' \quad Q = [\![P']\!]_{\tilde{a}} \quad P \to P'$$

But both statements fail because of forwarders!

Roundabout

Extend ambients with parentheses

$$P \quad ::= \quad \cdots \mid (\!| P |\!)$$

They are introduced when an ambient is dissolved

The encoding

$$
\begin{aligned}
[\![\mathbf{0}]\!]_{\tilde{a}} &\triangleq \mathbf{0} \\
[\![n[P]]\!]_{\tilde{a}} &\triangleq (\nu\tilde{b})(Amb(n,\tilde{b},\tilde{a})|[\![P]\!]_{\tilde{b}}) \\
[\![(\!|P|\!)]\!]_{\tilde{a}} &\triangleq (\nu\tilde{b})(Fwd(\tilde{b},\tilde{a})|[\![P]\!]_{\tilde{b}}) \\
[\![\mathsf{in}\, m.P]\!]_{\tilde{a}} &\triangleq {}^{\tau}\backslash_{a_{in}}\langle m,\underline{\tilde{x}}\rangle.[\![P]\!]_{\tilde{a}} \\
[\![\mathsf{out}\, m.P]\!]_{\tilde{a}} &\triangleq {}^{\tau}\backslash_{a_{out}}\langle m,\underline{\tilde{x}}\rangle.[\![P]\!]_{\tilde{a}} \\
[\![\mathsf{open}\, n.P]\!]_{\tilde{a}} &\triangleq {}^{\tau}\backslash_{a_{opn}}\langle n\rangle.[\![P]\!]_{\tilde{a}}
\end{aligned}
$$

$$
\begin{aligned}
[\![P|Q]\!]_{\tilde{a}} &\triangleq [\![P]\!]_{\tilde{a}}|[\![Q]\!]_{\tilde{a}} \\
[\![(\nu n)P]\!]_{\tilde{a}} &\triangleq (\nu n)[\![P]\!]_{\tilde{a}} \\
[\![!P]\!]_{\tilde{a}} &\triangleq \mathsf{rec}X.([\![P]\!]_{\tilde{a}}|X)
\end{aligned}
$$

$$
\begin{aligned}
Amb(n,\tilde{a},\tilde{f}) \triangleq\ & {}^{a_{in}}\backslash_{f_{[in]}}\langle \underline{m},\underline{\tilde{z}}\rangle.Amb(n,\tilde{a},\tilde{z}) + {}^{f_{[in]}}\backslash_{\tau}\langle n,\tilde{a}\rangle.Amb(n,\tilde{a},\tilde{f}) + \\
& {}^{a_{out}}\backslash_{f_{[out]}}\langle \underline{m},\underline{\tilde{z}}\rangle.Amb(n,\tilde{a},\tilde{z}) + {}^{a_{[out]}}\backslash_{\tau}\langle n,\tilde{f}\rangle.Amb(n,\tilde{a},\tilde{f}) + \\
& {}^{f_{opn}}\backslash_{\tau}\langle n\rangle.Fwd(\tilde{a},\tilde{f}) \\
Fwd(\tilde{a},\tilde{f}) \triangleq\ & {}^{a_{in}}\backslash_{f_{in}}\langle \underline{n},\underline{\tilde{x}}\rangle.Fwd(\tilde{a},\tilde{f}) + {}^{a_{[in]}}\backslash_{f_{[in]}}\langle \underline{n},\underline{\tilde{x}}\rangle.Fwd(\tilde{a},\tilde{f}) + {}^{f_{[in]}}\backslash_{a_{[in]}}\langle \underline{n},\underline{\tilde{x}}\rangle.Fwd(\tilde{a},\tilde{f}) + \\
& {}^{a_{out}}\backslash_{f_{out}}\langle \underline{n},\underline{\tilde{x}}\rangle.Fwd(\tilde{a},\tilde{f}) + {}^{a_{[out]}}\backslash_{f_{[out]}}\langle \underline{n},\underline{\tilde{x}}\rangle.Fwd(\tilde{a},\tilde{f}) + \\
& {}^{a_{opn}}\backslash_{f_{opn}}\langle \underline{n}\rangle.Fwd(\tilde{a},\tilde{f}) + {}^{f_{opn}}\backslash_{a_{opn}}\langle \underline{n}\rangle.Fwd(\tilde{a},\tilde{f})
\end{aligned}
$$

Some references

- Julian Rathke, Pawel Sobocinski: Deriving structural labelled transitions for mobile ambients. Inf. Comput. 208(10): 1221-1242 (2010)
- Filippo Bonchi, Fabio Gadducci, Giacoma Valentina Monreale: Reactive Systems, Barbed Semantics, and the Mobile Ambients. FOSSACS 2009: 272-287
- Massimo Merro, Francesco Zappa Nardelli: Behavioral theory for mobile ambients. J. ACM 52(6): 961-1023 (2005)
- Gian Luigi Ferrari, Ugo Montanari, Emilio Tuosto: A LTS Semantics of Ambients via Graph Synchronization with Mobility. ICTCS 2001: 1-16

Roadmap

- Problem statement: intro and motivation
- A new kind of interaction
- Handling message content
- Encoding mobile ambients
- Conclusion and future work

Conclusion

Envisage interaction like a puzzle

A theory of linked interactions

Derive standard first-order LTS semantics
(and suitable bisimilarities congruences)

Ongoing work

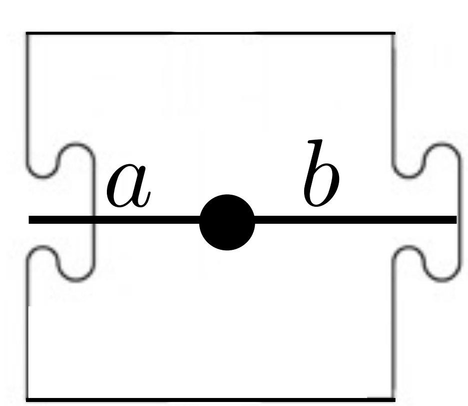
Relation with existing abstract semantics for mobile ambients (conjecture: slightly finer equivalence, but non ad hoc)

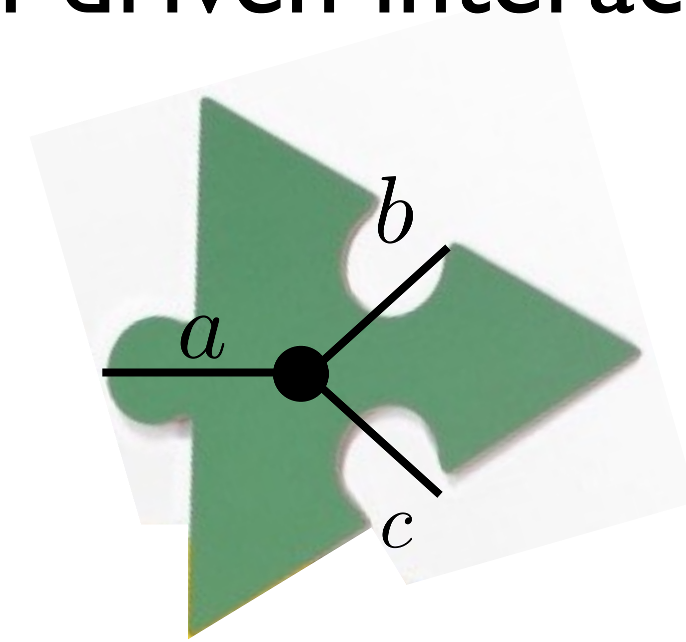
Future work

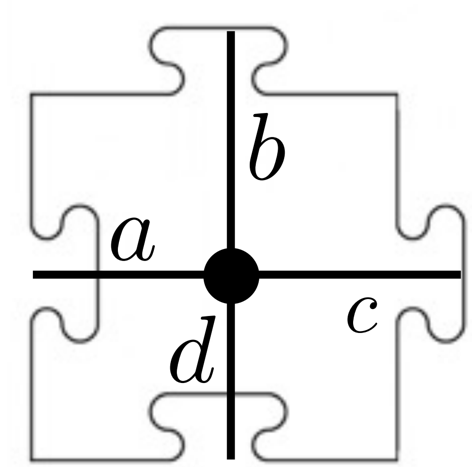
Expressiveness
(with/without name mobility)

Extensions: $\left({}^{\tau}\backslash^{*}_{a}\backslash^{b}_{*}\backslash^{*}_{b}\backslash^{c}_{*}\backslash_{\tau} \right).P$
non-simple prefixes

graph-driven interaction







THANKS FOR THE ATTENTION