

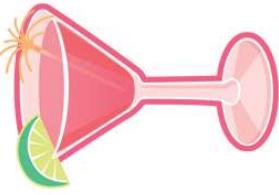
Outline

- Motivation
- Key notion
- CCS
- From CCS to Pi calculus
- **Pi calculus: syntax and semantics**
- The role of restriction for security
- Modeling protocols in Pi calculus
- Conclusions and what next

Pi Calculus

Milner:

*It will appear as though we reduce all concurrent computation to something like a **cocktail party**, in which the only purpose of communication is to transmit (or to receive) a name which will admit further communications. Surprisingly, this meagre basis is enough to encode computation over an arbitrary data types... We tentatively call our new calculus the **pi calculus**, since it aims for universality (at an elementary level) for concurrent computation, just as the **lambda calculus** is universal for functional computation.*



Ha il grande merito di aver enucleato un po', quali sono le primitive essenziali per rappresentare i linguaggi di programmazione concorrente, in particolare, quindi, siccome si presuppone che in questi sistemi sia possibile comunicare tra le varie entità, ci sono delle primitive appunto, per mandare e ricevere messaggi quindi si può pensare a vedere il passaggio di messaggi, si può avere la mobilità ed è anche possibile la creazione dinamica di canali, cioè? ogni processo può creare un nuovo canale, questo sempre per mettervi la pulce nell'orecchio, avrà a che fare anche con la modellazione dei protocolli, perché potrò creare nuovi canali privati o chiavi.

Pi Calculus or π calculus

- Introduced by Milner, Parrow, Walker in 1989. It extends CCS [1980]: channel names can be passed between processes
- Small but expressive programming language
- Core language for concurrent programming:
 - **Send** and **receive** primitives (point-to-point communication between concurrent processes)
 - **Message passing**
- **Dynamic creation of channels:** a channel is a transferable capability of communication
 - Creation of **new** tags (e.g., keys)
 - **Mobility** and network reconfiguration:
 - interconnections change with interaction

Pi Calculus

Warning: many, many variants! E.g.

- what is a message?
 - an atomic name
 - ...
 - a generic term (e.g., $f(g(x), y)$)
 - many ways to define the semantics
 - We focus on the main concepts rather than on the details of the particular variant we present here.

Names and processes

- An infinite set of names (channels, links, ports), with no structure: $a, b, \dots, p, q, r, \dots, x, y, \dots$
- A set of entities, called processes: A, B, \dots, P, Q, \dots
- Semantics through transitions
 - $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow \dots$
- Sometimes, we shall annotate the transitions with some label, to make some event observable

Syntax

$P ::= 0$

| $\tau.P$

| $\bar{x}y.P$

| $x(y).P$

| $[x = y] P$

| $(x) P$ [or (νx)]

| $P \mid P$

| $P + P$

| $! P$

inactive process

silent action

output

input

match

channel creation or restriction

parallel composition

non deterministic choice

replication

- Prefix ":" imposes an order upon actions

We take processes up to
alpha-conversion:
a bound names can be
renamed with a fresh name

Remarks

- Occurrences of the inactive process **0** will sometimes be omitted, thus, e.g., $\bar{x} < y >. 0$ will be written $\bar{x} < y >$
- τ indicates an internal action
- $x(y)$ indicates input, while $\bar{x}y$ indicates output
- $[x = y]P$ is known as **name matching**: it is equivalent to **if $x = y$ then P** , otherwise it is stuck
- $(x)P$ is alike CCS **restriction** $P \backslash x$
- $!P$ models **replication** (and hence recursion) and denotes the parallel composition of an arbitrary number of copies of P , i.e., it behaves like $P \parallel P \parallel \dots$

Two forms of binding

- $\text{INPUT } x(y).P$
 - Binding occurrence
 - Scope
- y is a *placeholder* for any name which may be received on the channel x . It is a bound name
 - Binding occurrence
 - Scope
- **RESTRICTION** $P \backslash y$ or $(y)P$ or $(u\ y)\ P\ y$
 - y creates a **new name (private to P)** and binds it to y in P : y cannot be confused with any other name in P
 - Unlike for CCS, the scope y can be dynamically extended, e.g., the process $(y)(\bar{x}y) \mid x(z).Q$ can extrude the **scope** of the restricted name y and then wait to receive some data on it
 - These names are **bound**, the others are **free**

Remarks (cont.)

- In $x(y).P \in (y)P$, the name y is **bound** in P (i.e., P is the **scope** of such name). A name that is **not bound** is called **free**
 - $\text{fn}(P)$ e $\text{bn}(P)$ are the sets of all **free**, resp. **bound**, names of P
- $\text{bn}(x(y).p) \stackrel{\text{def}}{=} \{y\} \cup \text{bn}(p)$

$\text{bn}((y).p) \stackrel{\text{def}}{=} \{y\} \cup \text{bn}(p)$
- $n(P) = \text{fn}(P) \cup \text{bn}(P)$
 - We take processes up to **alpha-conversion**, denoted by $=_{\alpha}$, which permits renaming of a bound name with a fresh name that is not already used

Substitution

- A substitution $\sigma : N \rightarrow N$ is a function on names that is the identity except on a finite set of names
- We write $\{y_1, \dots, y_n/x_1, \dots, x_n\}^*$ for the substitution σ such that $x_i = y_i$ for $i = 1, \dots, n$ and $x = x$ otherwise
- When applying σ to a process P we want to rename only the **free occurrences** of names x in P , **not** the **bound ones**
- **unintended capture of names x by binders of P must be avoided**
- How to define $P\{z/x\}$ if $P = x(y).\bar{w}x$?
- $P\{z/x\} =$

*also $\{y_1 | \dots > x_1, \dots, y_n | \dots > x_n\}$

Substitution

- A substitution $\sigma : N \rightarrow N$ is a function on names that is the identity except on a finite set of names
- We write $\{y_1, \dots, y_n/x_1, \dots, x_n\}^*$ for the substitution σ such that $x_i = y_i$ for $i = 1, \dots, n$ and $x = x$ otherwise
- When applying σ to a process P we want to rename only the free occurrences of names x in P , **not** the bound ones
- **unintended capture of names x by binders of P must be avoided**
- How to define $P\{z/x\}$ if $P = x(y).\bar{w}x$?
- $P\{z/x\} = z(y).\bar{w}z$



La sostituzione non è altro che quella funzione che mi permette di sovrascrivere i legami, cioè per cui quello che veniva chiamato x adesso viene chiamato y e quindi una sostituzione non è altro che una funzione sui nomi che mi permette di sostituire un nome con un'altro, quindi quando io lo applico a un processo P , noi vogliamo rinominare solo le occorrenze di quel nome, quindi quando io ricevo un certo nome, per esempio in input io andrò a sostituire tutte le occorrenze, supponiamo che sia x di y , io andrò a sostituire tutte le occorrenze di y con il nome che ricevo quindi in particolare vediamo l'esempio, quindi, supponiamo di avere un processo che aspetta sul canale x un valore per y e poi lo fa andare avanti. Che cosa succede se devo fare una sostituzione di x con z ? Non si creano problemi perché io vado a sostituire un nome che è libero, se voi ci pensate qui x è un nome libero, w è un nome libero di nuovo x un'altro occorrenza sempre libero è, mentre non sarebbe libero quale nome? y quindi se ci fosse invece un'occorrenza di y , quel nome non sarebbe libero.

Substitution (cont.)

- How to define $P\{x/z\}$ if $P = y(x).\bar{x}z$?

Substitution (cont.)

- How to define $P\{x/z\}$ if $P = y(x).\bar{x}z$?
- $y(x).\bar{x}x$ NO: name **confusion here**

Se io invece avessi P dove sostituisco zeta con x , cosa succederrebbe? Qui invece si crea il problema, perché? perché z è già definito e li ce l'abbiamo già, quindi avremo uno scontro con la nuova definizione, non sappiamo più distinguergli sì, perché io zeta ce l'ho già, ma anche la x e la x qua è legata quindi quello che succede è che se io facessi così avrei confusione di nomi perché io mescolerei mele con le pere perché la x legata ha un significato diverso da zeta, hanno due ruoli diversi? x sta per tutto ciò che deve andare e quindi a questo punto io mi scollerò il ruolo di chi deve fare la spedizione in questo caso con il ruolo di ciò che viene spedito e sono due cose diverse perché il canale su cui faccio l'operazione è quella che mi viene spedita dall' output, quindi io prima non la conosco e quindi è nuova, mentre z è diverso, quindi come devo fare?

Substitution (cont.)

A bound object
is a reference

- How to define $P\{x/z\}$ if $P = y(x).\bar{x}z$?
- **$y(x).\bar{x}x$ NO: name confusion must be avoided**
- Bound names cannot be substituted... but can be alpha-converted first

devo stare attenti a non sostituire i canali legati, i nomi legati, ma devo prima di generare confusione, fare l'alfa conversione, abbiamo detto che che io chiami ciò che ricevo x o lo chiami pippo per essere più chiari non conta, cioè io ho bisogno semplicemente di un placeholder, io ho bisogno che qui possa mettere tutto ciò che mi lega nel prosegui,

Substitution (cont.)

A bound object
is a reference

- How to define $P\{x/z\}$ if $P = y(x).\bar{x}z$?
- **$y(x).\bar{x}x$ NO: name confusion must be avoided**
- Bound names cannot be substituted... but can be alpha-converted first



$y(x).\bar{x}z \underset{\alpha}{=} y(w).\bar{w}z$ where w is a **fresh** name

- Bound names are a sort of placeholders for actual values
- $y(x).\bar{x}z$: whatever (x or w , who cares!) is received on y is the channel used to send z
- Hence, we first alpha-convert and then substitute
- $y(x).\bar{x}z\{x/z\} \underset{\alpha}{=} y(w).\bar{w}z\{x/z\} = y(w).\bar{w}x$ OK

Semantics: actions

- ora la semantica, come vi potete immaginare è fatta in maniera analoga a quella che abbiamo visto il CCS c'è un sistema di transizione etichettato, le transazioni qui sono un pochino più complesse perché come vi dicevo devono ottemperare la questione della possibile estrusione dei nomi quindi in particolare, io ho azioni sempre nella forma π -alfa-> Q ho naturalmente l'evoluzione con l'azione tau, poi ho l'azione che mi rappresenta l'output e l'azione che mi rappresenta l'input. Ora come vedete si parla non solo di output ma di quello che viene chiamato output libero e input libero, qui va messa questa distinzione perché come esiste quello libero esiste quello bound. Si distingue dal fatto che nel momento in cui faccio un bound output tutto quello che sto facendo è una estrusione, cioè lo sto emettendo un nome privato sul canale x quindi P in questo momento fa un bound output, quando il nome che rimanda non è un nome qualsiasi, ma un nome ristretto.

- A transition in the pi calculus is of the form $P \xrightarrow{\alpha} Q$
- Intuitively, it means that P can evolve into Q, and in doing so perform the action α
- The possible forms are:
 - $P \xrightarrow{\tau} Q$ P can evolve into Q, with no interaction with the environment [silent action]
 - $P \xrightarrow{\overline{w}y} Q$ P permits the free name y on the channel x [free output]
 - $P \xrightarrow{x(y)} Q$ P can receive any name w on x, becoming $Q\{w/y\}$ [free in]
 - $P \xrightarrow{\overline{x}(y)} Q$ P emits a private name y on x [bound output] Bound outputs arise from free outputs which carry names out of their scope.

Semantics (cont.)

Tau $\tau.P \xrightarrow{\tau} P$

Inp $x(y).P \xrightarrow{x(w)} P\{w/y\}$ $w \notin \text{fn}(\nu y P)$ Out $\bar{x}y.P \xrightarrow{\bar{x}y} P$

- y is a placeholder and w is fresh, i.e., it does not occur in P: w=z or w not in fn(P)

dal punto di vista formale la semantica mi è regolata nella stessa maniera del ccs, cioè vediamo proprio regole di inferenza logica, premesse e conclusioni, che naturalmente iniziano con quelli che vengono chiamati gli assiomi, cioè le regole che non hanno bisogno di premesse perché funzionano sempre, cioè in tutti i casi in cui abbiamo dei prefissi, non c'ho bisogno di premesse, quindi io l'azione a top level la posso eseguire e arrivare nel processo continuazione quindi tau.P --tau-->P, l'input fa l'input e diventa la prosecuzione e l' output altrettanto. La particolarità di questo input e che può essere reso anche in un'altra maniera e che qui, ma non voglio entrare troppo nel dettaglio perché sennò diventa un corso monografico sulle algebre di processo, vi faccio vedere quella che viene chiamata versione early in cui si presuppone che in qualche modo nell'azione di input si indovini qual è il nome che viene passato dall' output, quindi io vado già a sostituire w su y sempre che w non appartenga a nomi liberi di y ristretti, ma questo lo vedremo dopo il motivo.

Semantics (cont.)

Tau $\tau.P \xrightarrow{\tau} P$

Inp $x(y).P \xrightarrow{x(w)} P\{w/y\}$ $w \notin \text{fn}(\nu y P)$

- y is a placeholder and
- w is fresh, i.e., it does not occur in P: w=z or w not in fn(P)

$$\begin{array}{c} \text{Match} \quad \frac{}{[x=x]P \xrightarrow{\alpha} P'} \\ \text{Sum} \quad \frac{}{P + Q \xrightarrow{\alpha} P'} \\ \text{Par} \quad \frac{}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \end{array} \quad \begin{array}{c} \text{Bang} \quad \frac{P \mid !P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'} \\ \text{To avoid name captures} \end{array}$$

Semantics (cont.)

$$\text{Com} \quad \frac{P \xrightarrow{\bar{x}y} P' \quad Q \xrightarrow{x(y)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\text{Close} \quad \frac{P \xrightarrow{\bar{x}(w)} P' \quad Q \xrightarrow{x(w)} Q'}{P \mid Q \xrightarrow{\tau} \nu w (P' \mid Q')}$$

It **closes** the scope of restriction enclosing Q :
 w becomes private btw P' and Q'

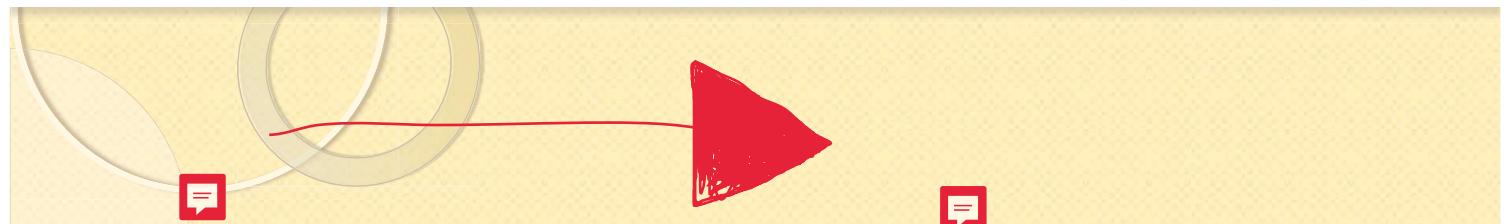
It **opens** the scope of restriction
 of restriction

$$\text{Open} \quad \frac{P \xrightarrow{\bar{x}y} P'}{\nu y P \xrightarrow{\bar{x}(w)} P' \{w/y\}}$$

Bound output label

$$\text{Res} \quad \frac{P \xrightarrow{\alpha} P'}{\nu y P \xrightarrow{\alpha} \nu y P'}$$

Free and bound names



Structural Congruence

The syntax of processes is to some extent too concrete:

- The order processes are composed in parallel should not matter.
- The order processes "summed" should not matter.
- The order names are restricted should not matter.

Processes differing only for the above behave more or less the same

$$(ALPHA) \quad \frac{P =_a Q \quad Q \xrightarrow{\alpha} Q'}{P \xrightarrow{\alpha} Q'}$$

Parliamo di congruenza strutturale, abbiamo già visto che a volte la sintassi dei processi è troppo concreta e abbiamo visto che sostanzialmente non è così importante distinguere l'ordine in cui i processi vengono composti in parallelo o messi in somma per il non determinismo e persino l'ordine in cui nomi sono ristretti non dovrebbe importare, cioè se io ho come ristretti i nomi xy o yx non cambia molto quindi, laddove i processi differiscono per questi tipi di aspetti, proprio che si possono rendere precisi in termini algebrici posso usare un'ulteriore regola di semantica che mi dice che se i processi sono congruenti, allora si comportano nello stesso modo, quindi non ho bisogno di avere le due regole per la somma oppure le due regole per il parallelo, allo stesso modo, anche quando i processi differiscono per alfa conversion posso assimilare i due comportamenti, tanto che posso avere una regola che si chiama alfa che mi dice che se P è lo stesso processo di Q a meno di alfa conversione e Q si evolve in Q' allora nello stesso modo, P si può evolvere in Q'.

Structural Congruence (cont.)

$(P_{\cong}, +, 0)$ and $(P_{\cong}, |, 0)$ are commutative monoids

$$\begin{array}{ll} p + \text{nil} \equiv p & p + q \equiv q + p \\ p | \text{nil} \equiv p & p | q \equiv q | p \\ (x) \text{nil} \equiv \text{nil} & (y)(x)p \equiv (x)(y)p \\ [x=y]p \equiv p & [x=x]p \equiv p \end{array}$$

$$\begin{array}{ll} p + q \equiv q + p & (p + q) + r \equiv p + (q + r) \\ p | q \equiv q | p & (p | q) | r \equiv p | (q | r) \\ (x)(p | q) \equiv p | (x)q & (x)(p | q) \equiv p | (x)q \text{ if } x \notin \text{fn}(p) \\ p | !p \equiv !p & \end{array}$$

We need fewer semantic rules

queste sono le regole di congruenza strutturale invece che come vedete hanno una chiara struttura algebrica sono proprio monoidi commutativi proprio perché ho un elemento neutro, ho la proprietà commutativa e poi ho anche una proprietà associativa. Per quanto riguarda invece le restrizioni, come dicevamo effettivamente posso controllare, è la terza riga, e la cosa che abbiamo già detto è che se ho ristretto y e poi x o viceversa non fa differenza, la regola invece a sinistra mi dice che il processo nil con la restrizione su x posso anche toglierla la restrizione perché la x non appare dentro nil e quindi quella è equivalente in tutto e per tutto a nil. La regola sulla destra merita un attimo in più di riflessione ed è quella che mi dice che laddove la x è un nome ristretto tra P e Q ma in P parallelo Q ma in P la x non appare mai, allora questo processo è congruente strutturalmente al processo in cui P viene tolto dallo scope della x. Infine è quello su cui si basa poi la regola che abbiamo visto per la replication, il processo P, in parallelo con !P è congruente strutturalmente a !P cioè, e questa è la regola che vi permette di fare l'unfolding, cioè da questo punto di vista lo potete leggere da destra verso sinistra !P è considerato equivalente a P !P ma siccome a sua volta !P è equivalente a P!P, !P è equivalente a P parallelo P parallelo !P e quindi possiamo fare l'unfolding all'infinito.

Derivations

Derive

per la
spiegazione
della
derivazione
andare alla fine

$$(((\bar{y})\bar{x}y.P) \mid Q \mid x(z).R) \xrightarrow{\tau} (\bar{v})(((P\{v/y\} \mid Q) \mid R\{v/z\})$$

Derivations

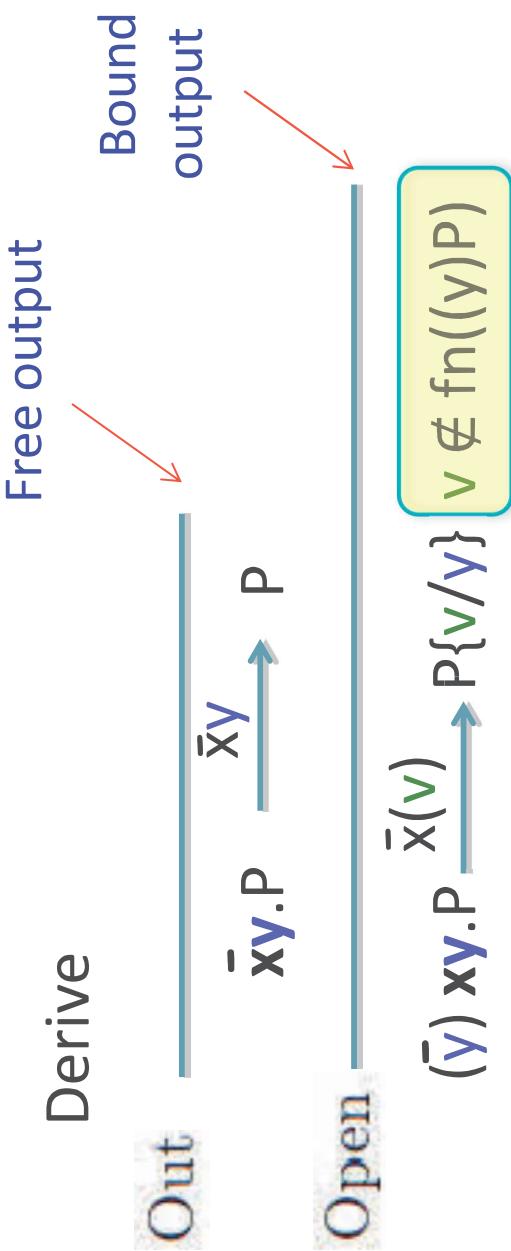
Derive

Free output



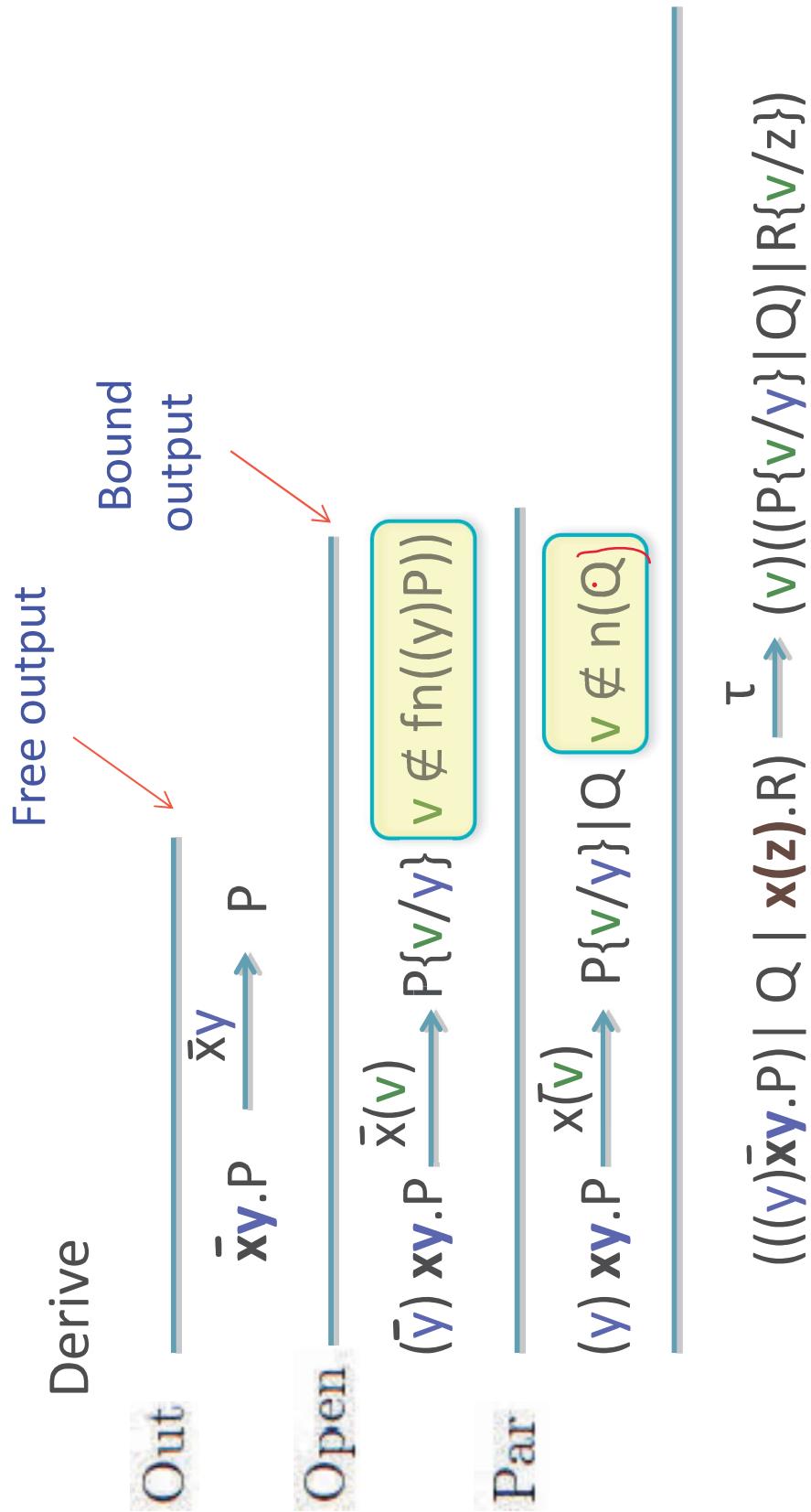
$$(((y)\bar{xy}.P) \mid Q \mid x(z).R) \xrightarrow{\tau} (v)((P\{v/y\} \mid Q) \mid R\{v/z\})$$

Derivations

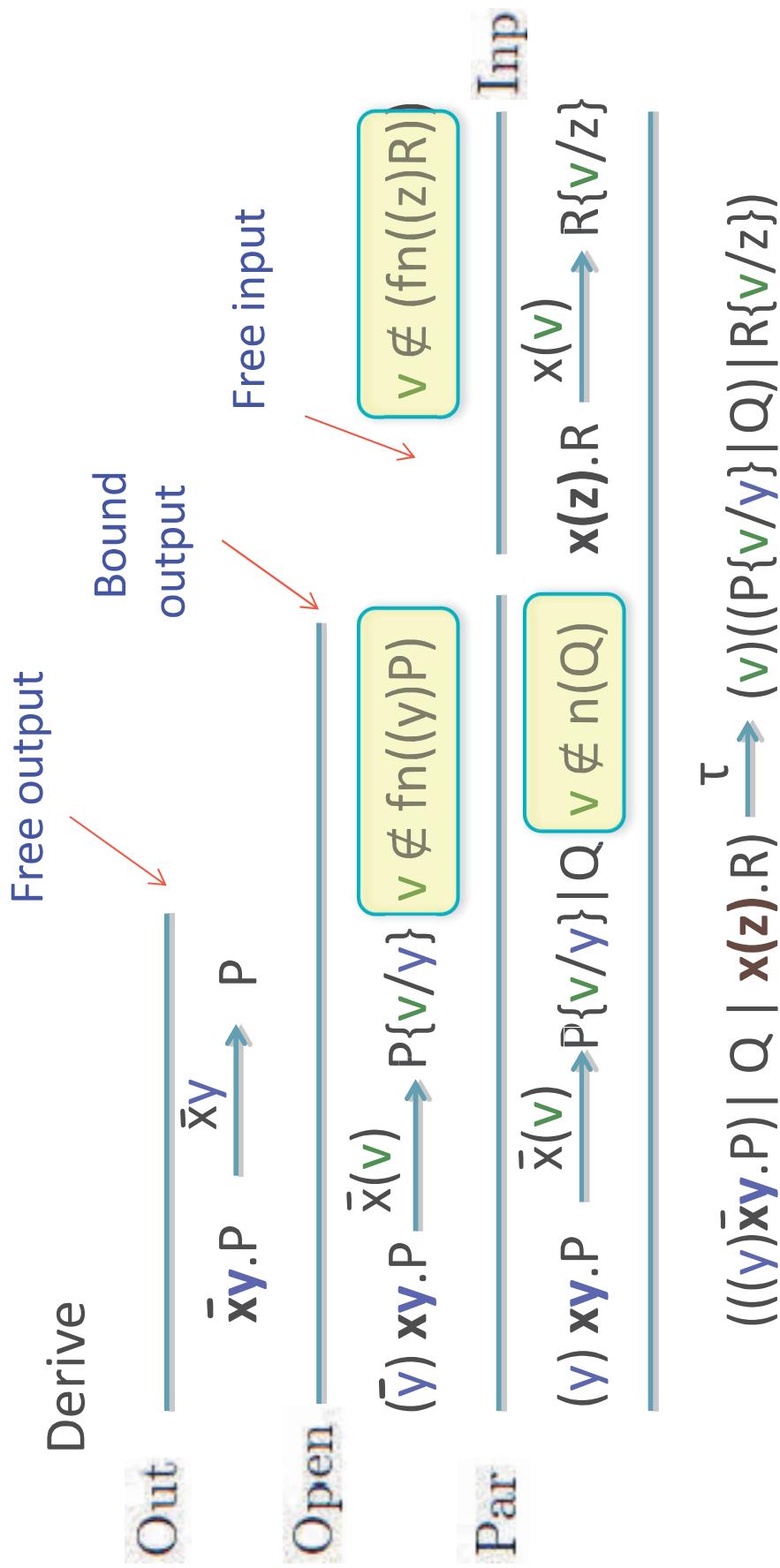


$$(((y)\bar{x}\bar{y}.P) \mid Q \mid x(z).R) \xrightarrow{\tau} (v)((P\{v/y\} \mid Q) \mid R\{v/z\})$$

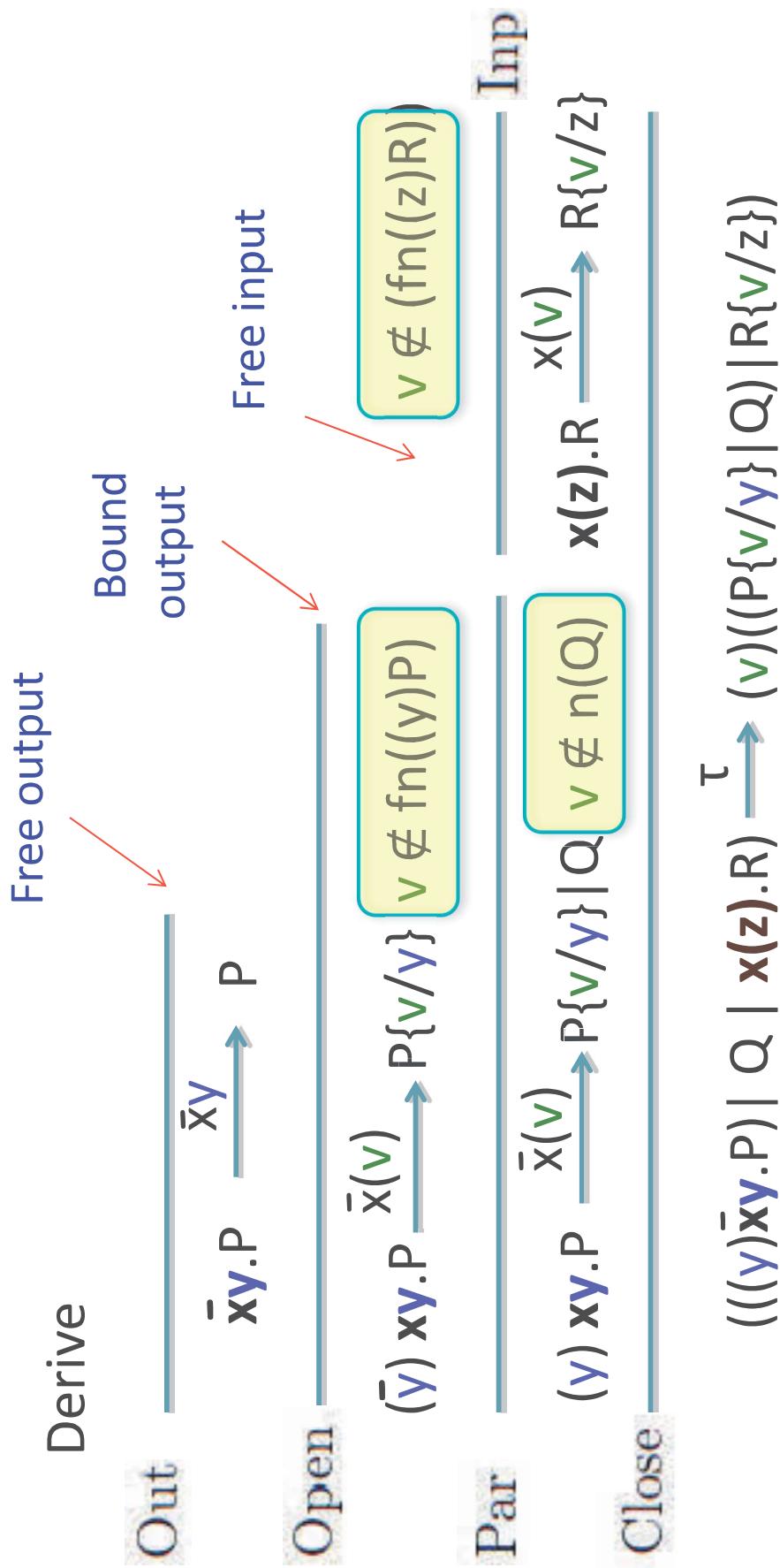
Derivations



Derivations

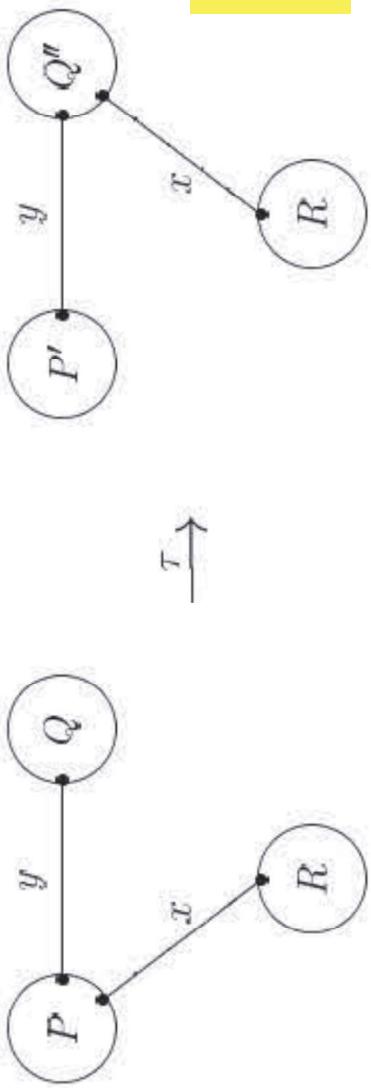


Derivations



Restriction

The agent P has a link x to R, and wishes to pass x along its link y to Q. Q is willing to receive it.

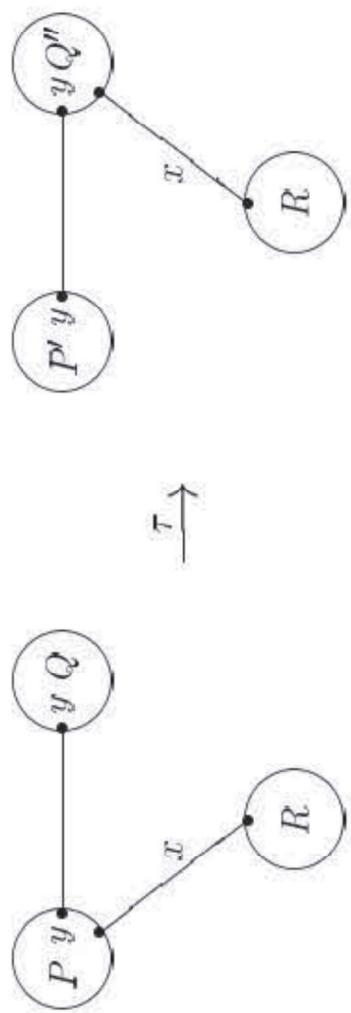


$$\bar{y}x.P' \mid y(z).Q' \mid R \xrightarrow{\tau} P' \mid Q'\{x/z\} \mid R$$

il processo τ per esempio, questa è un output normale manda il nome, il canale x all'altro processo e in questo caso sono tutti pubblici, ora pensiamo già in termini di sicurezza, sono tutti pubblici, quindi lui può mandare il che vuol dire che manda un indirizzo di un canale che comunque è pubblico. Cosa diversa quando il link è privato ed è il caso che abbiamo visto prima, quindi noi abbiamo che su P voglio spedire il canale x , qui poi il canale resta con lo stesso nome, l'effetto dell'estruzione è quello per cui il canale x alla fine viene posseduto anche da Q.

Example: private link passing

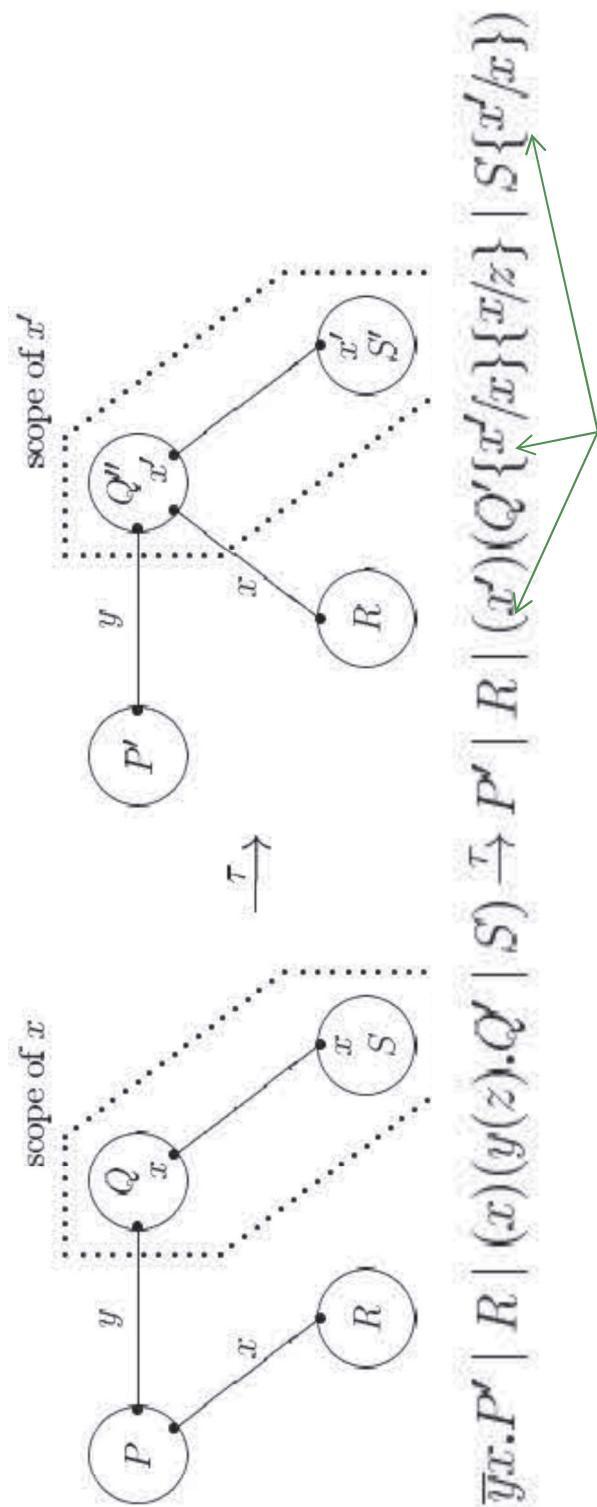
The link is private



$$(y)(\bar{y}x.P' \mid y(z).Q') \mid R \xrightarrow{\tau} (y)(P' \mid Q'\{x/z\}) \mid R$$

Example: scope intrusion

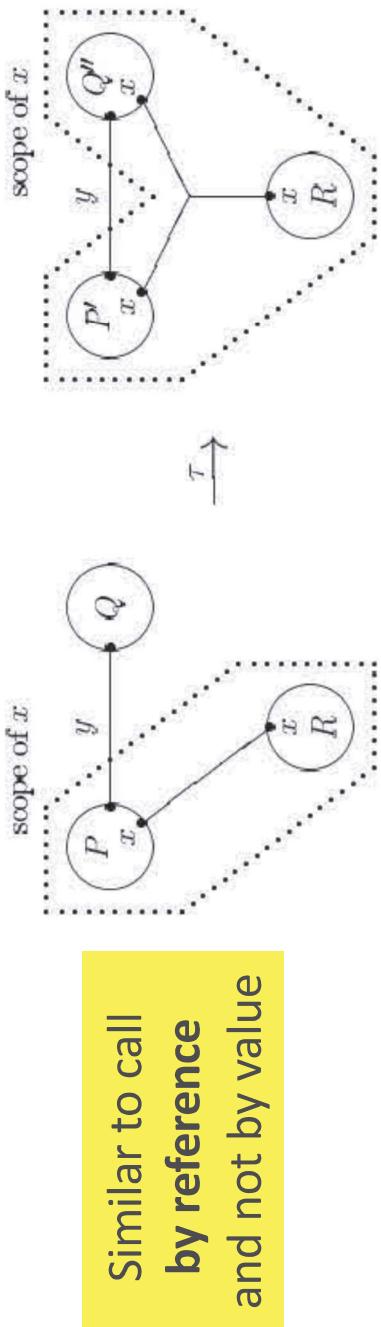
P has a link x to R and wishes to pass x along its link y to Q. Q is willing to receive it, but already possesses a private link x to S; the latter must be renamed to avoid confusion. P intrudes the scope of the private link x between Q and S.



Alpha conversion, i.e., change of bound names is needed

Example: scope extrusion

P has a link x to R, but we now suppose that this link is private. However, P wishes to pass x along its link y to Q. Q is willing to receive it, and possesses no x-link.

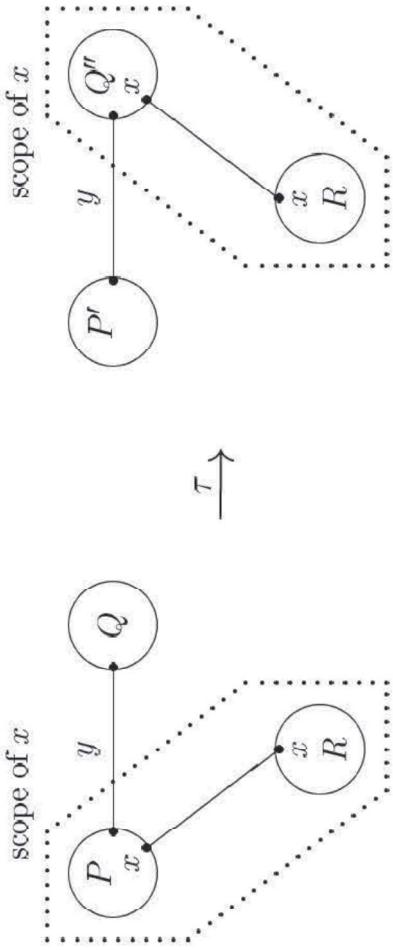


$$(x)(\bar{y}x.P' \mid R) \mid y(z).Q' \xrightarrow{T} (x)(P' \mid R \mid Q'\{x/z\})$$

When this link is exported to Q the scope of the restriction is extended, we say that P extrudes the scope of the private x-link. If Q possesses a public x-link than the extruded name must be renamed

Example: scope extrusion

If P' does not possesses the private x -link after the transition, we have a migration of scope



$$(x)(\bar{y}x.P' \mid R) \mid y(z).Q' \xrightarrow{\tau} (x)(P' \mid R \mid Q'\{x/z\})$$

since $(x)(P_1 \mid P_2) = P_1 \mid (x)P_2$ if $x \notin \text{fn}(P_1)$

we have $(x)(P' \mid R \mid Q'\{x/z\}) = P' \mid (x)(R \mid Q'\{x/z\})$

Outline

- Motivation
- Key notions
- CCS
- From CCS to Pi calculus
- Pi calculus: syntax and semantics
- **The role of restriction for security**
- Modeling protocols in Pi calculus
- Conclusions and what next

Role of restriction

- The restriction operator $(v\;x)P$ makes a fresh channel x for use within process P . Hence, restriction governs the visibility of names
- Nobody can monitor a restricted channel, outside the scope
- Scoping is the basis of security

per preparare il terreno rispetto alla modellazione di scenari di sicurezza potete immaginare che è questo operatore di restrizione che mi rende questo formalismo adatto alla modellazione. Questo operatore mi da la possibilità di creare canali freschi ad esempio nomi di chiavi o di canali ristretti che possono essere usati dai processi. Le regole della restrizione che abbiamo visto sono in grado di governare la visibilità di questi nomi, quindi posso decidere di triangolare e di mandare un nome ristretto da una parte alla altra e siccome questi nomi per definizione non sono visibili all'esterno questo mi dà proprio l'idea della sicurezza.

Restriction against intrusion

- The use of restricted channels can prevent intrusions*

Naive Campaigning	Vota Antonio
$Speaker$	$\triangleq \overline{air} \langle vota\;antonio \rangle$
$Microphone$	$\triangleq air(x).\overline{wire}(x)$
$Loudspeaker$	$\triangleq wire(y).\overline{highvolume}(y)$
Ad	$\triangleq Speaker \mid Microphone \mid Loudspeaker$

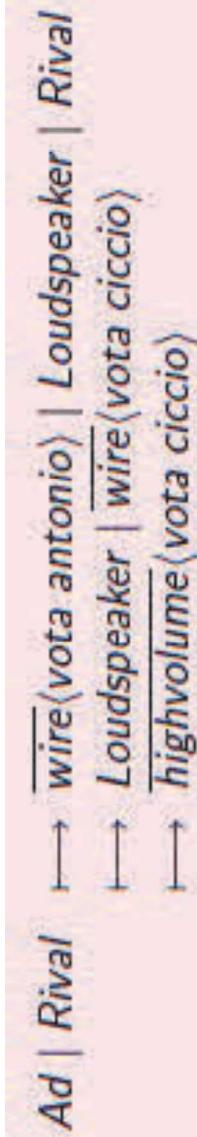
$Ad \longrightarrow \overline{wire} \langle vota\;antonio \rangle \mid Loudspeaker$
 $\longrightarrow \overline{highvolume} \langle vota\;antonio \rangle$

Questo è un esempio per spiegare l'uso dei canali ristretti nella sicurezza: pensate di avere una campagna politica in cui c'è un comizio in cui lo speaker deve incoraggiare l'uditore a votare per antonio, quindi lo speaker supponiamo che parli attraverso l'aria, vota antonio, la voce arriva attraverso un microfono e quindi voi vedete che questo sistema è composto da più entità, quindi c'è lo speaker che dice votantonio, il microfono che riceve la voce dello speaker e la rimanda sul filo, il filo arriva all'altoparlante e l'altoparlante manda con volume più alto di qui e quindi amplifica la voce dello speaker. L'evoluzione del mio sistema è quello che mi fa sì che la frase venga detta dallo speaker e quindi venga raccolta testualmente con tau dal microfono, dopodiché ci sarà una trasmissione tra il microfono e lo speaker e alla fine ci sarà l'emissione ad alto volume di ciò che ha detto lo speaker e questa è la situazione in assenza di restrizioni. Qui è uno schema in cui tutto è tranquillo e non succede niente. Allora questa è una campagna naïve perché si assume che tutti si comportino correttamente.

Restriction against intrusion (cont.)

- The use of restricted channels can prevent intrusions*

Naive Campaigning	Vota Antonio
Speaker	$\triangleq \overline{\text{air}}\langle \text{vota antonio} \rangle$
Microphone	$\triangleq \overline{\text{air}}(x).\overline{\text{wire}}\langle x \rangle$
Loudspeaker	$\triangleq \overline{\text{wire}}(y).\overline{\text{highvolume}}\langle y \rangle$
Ad	$\triangleq \text{Speaker} \mid \text{Microphone} \mid \text{Loudspeaker}$
Rival	$\triangleq \overline{\text{wire}}(z).\overline{\text{wire}}\langle \text{vota ciccio} \rangle$



Posso utilizzare i canali ristretti per prevenire possibili intrusioni. Quindi proviamo a immaginare uno scenario dove, oltre allo speaker, nel sistema appare maliziosamente, un cosiddetto rivale che invece vuole promuovere il voto per ciccio. Quello che succede è che il rivale potrebbe raccogliere sul filo del microfono quello che arriva sostanzialmente dallo speaker e rimandare, non vota antonio come ha raccolto, ma vota ciccio come vuole lui, il che vuol dire che alla fine lui si intromette nella comunicazione e fa sì che ad alto volume venga detto vota ciccio invece che vota antonio, questo è un classico spiegato, con un esempio giocattolo, ovviamente un caso di attacco di tipo man in the middle quindi siccome wire è un canale che non è privato, il rivale può raccogliere, quindi bloccare sostanzialmente quello che arriva dal filo e rimandare quel che gli pare a lui. Come si fa dal punto di vista della modellazione a impedire questa intrusione che è sia passiva che attiva, nel senso che lui blocca la comunicazione da una parte e ne rimanda un'altra fake.

Si fa attraverso una campagna sicura in cui viene utilizzato questo meccanismo della restrizione ovvero si fa sì che ciò che passa dallo speaker al microfono e rispettivamente dal microfono all' altoparlante venga ristretto questo tipo di attacco man in the middle. Il canale ristretto mi rappresenta proprio qualcosa che non è accessibile a meno che uno non sia stato coinvolto esplicitamente nella comunicazione, quindi in questo caso questi due canali vengono condivisi dai partecipanti onesti di questo protocollo, che se volete sono lo speaker e il microfono e l' altoparlante.

Restriction against intrusion (cont.)

- The use of restricted channels can prevent intrusions*

Naive Campaigning	Vota Antonio
<i>Speaker</i>	$\triangleq \overline{\text{air}}\langle \text{vota antonio} \rangle$
<i>Microphone</i>	$\triangleq \text{air}(x).\overline{\text{wire}}\langle x \rangle$
<i>Loudspeaker</i>	$\triangleq \overline{\text{wire}(y).\text{highvolume}}\langle y \rangle$
<i>Ad</i>	$\triangleq \text{Speaker} \mid \text{Microphone} \mid \text{Loudspeaker}$
<i>Rival</i>	$\triangleq \overline{\text{wire}(z).\text{wire}}\langle \text{vota ciccio} \rangle$

<i>Ad</i> <i>Rival</i>	$\rightarrow \overline{\text{wire}}\langle \text{vota antonio} \rangle \mid \text{Loudspeaker} \mid \text{Rival}$
	$\rightarrow \overline{\text{Loudspeaker}} \mid \overline{\text{wire}}\langle \text{vota ciccio} \rangle$
	$\rightarrow \overline{\text{highvolume}}\langle \text{vota ciccio} \rangle$

Vota Antonio: secure campaigning
<i>SecureAd</i> $\triangleq (\nu \text{ air}, \text{wire})(\text{Speaker} \mid \text{Microphone} \mid \text{Loudspeaker})$