MPRI - Course on Concurrency

Lecture 15

Expressiveness issues

Catuscia Palamidessi
INRIA Futurs and LIX
catuscia@lix.polytechnique.fr
www.lix.polytechnique.fr/~catuscia

Page of the course:
http://mpri.master.univ-paris7.fr/C-2-3.html

Plan of the lecture

- Discussion on the notion of expressiveness – encoding
- Encoding some of the features of the synchronous π-calculus into the asynchronous π-calculus
  - Output prefix
  - Blind choice
  - Input-guarded choice
- Separation results
  - Impossibility of encoding the π-calculus with mixed guarded choice into the asynchronous π-calculus
  - Impossibility of encoding the π-calculus with mixed guarded choice into ccs
- Bibliography
- Exercises

The π-calculus: syntax

Similar to CCS with value passing, but values are channel names, and recursion is replaced by replication (!)

$$\pi \;::=\; x(y) \mid \bar{x}y \mid \tau$$

action prefixes (input, output, silent)

x, y are channel names

$$
\begin{array}{rcll}
P & ::= & O & \text{inaction} \\
 & \mid & \pi.P & \text{prefix} \\
 & \mid & P \mid P & \text{parallel} \\
 & \mid & P \;+\; P & \text{sum} \\
 & \mid & (\nu x)P & \text{restriction, new name} \\
 & \mid & !\,P & \text{replication}
\end{array}
$$

The asynchronous π-calculus: syntax

It differs from the π-calculus for the absence of the output prefix (replaced by output action) and also for the absence of choice (+)

$\pi \; ::= \; x(y) \mid \tau$ — action prefixes (input, silent)

x, y are channel names

$$
\begin{array}{rcll}
P & ::= & O & \text{inaction} \\
 & | & \pi.P & \text{prefix} \\
 & | & \bar{x}y & \text{output action} \\
 & | & P \mid P & \text{parallel} \\
 & | & (\nu x)P & \text{restriction, new name} \\
 & | & !\,P & \text{replication}
\end{array}
$$

Expressive power of π_a wrt π

- Clearly the (synchronous) π–calculus is at least as expressive as the asynchronous π–calculus. In fact, the latter is practically a subset of the former.

  Indeed, the output action can be seen as the output-prefix process with continuation 0. This relation is a strong bisimulation:

  $$\bar{x}y \sim \bar{x}y.0$$

- What about the opposite direction?
- In general, in order to compare the expressive power of two languages, we look for the existence/non existence of an **encoding** with certain properties among these languages
- What is a good notion of encoding to be used as basis to measure the relative expressive power?

A "good" notion of encoding

A “good” notion of encoding

In general we would be happy with an encoding $[\![\cdot]\!] : \pi \longrightarrow \pi_a$ being:

A "good" notion of encoding

In general we would be happy with an encoding $[\![\cdot]\!] : \pi \to \pi_a$ being:

- Compositional wrt the operators $[\![P \; op \; Q]\!] = C_{op}[[\![P]\!], [\![Q]\!]]$

A "good" notion of encoding

In general we would be happy with an encoding $[\![\cdot]\!] : \pi \rightarrow \pi_a$ being:

- Compositional wrt the operators $[\![P \ op \ Q]\!] = C_{op}[[\![P]\!], [\![Q]\!]]$
- (Preferably) homomorphic wrt | (distribution-preserving) $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$

A "good" notion of encoding

In general we would be happy with an encoding $[\![\cdot]\!] : \pi \rightarrow \pi_a$ being:

- Compositional wrt the operators $[\![P \; op \; Q]\!] = C_{op}[[\![P]\!], [\![Q]\!]]$
- (Preferably) homomorphic wrt | (distribution-preserving) $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$
- Preserving some kind of semantics. Here there are several possibilities

A "good" notion of encoding

In general we would be happy with an encoding $[\![\cdot]\!] : \pi \rightarrow \pi_a$ being:

- Compositional wrt the operators $[\![P \; op \; Q]\!] = C_{op}[[\![P]\!], [\![Q]\!]]$
- (Preferably) homomorphic wrt | (distribution-preserving) $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$
- Preserving some kind of semantics. Here there are several possibilities
    - Preserving observables $Obs(P) = Obs([\![P]\!])$

A "good" notion of encoding

In general we would be happy with an encoding $[\![\cdot]\!] : \pi \longrightarrow \pi_a$ being:

- Compositional wrt the operators $[\![P \; op \; Q]\!] = C_{op}[[\![P]\!], [\![Q]\!]]$
- (Preferably) homomorphic wrt | (distribution-preserving) $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$
- Preserving some kind of semantics. Here there are several possibilities
  - Preserving observables $Obs(P) = Obs([\![P]\!])$
  - Preserving equivalence

$$[\![P]\!] \; equiv \; [\![Q]\!] \;\; \Rightarrow \;\; P \; equiv' \; Q \quad \text{(soundness)}$$

$$[\![P]\!] \; equiv \; [\![Q]\!] \;\; \Leftarrow \;\; P \; equiv' \; Q \quad \text{(completeness)}$$

$$[\![P]\!] \; equiv \; [\![Q]\!] \;\; \Leftrightarrow \;\; P \; equiv' \; Q \quad \text{(full abstraction, correctness)}$$

A "good" notion of encoding

In general we would be happy with an encoding $[\![\cdot]\!] : \pi \rightarrow \pi_a$ being:

- Compositional wrt the operators $[\![P \; op \; Q]\!] = C_{op}[[\![P]\!], [\![Q]\!]]$
- (Preferably) homomorphic wrt | (distribution-preserving) $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$
- Preserving some kind of semantics. Here there are several possibilities
  - Preserving observables $Obs(P) = Obs([\![P]\!])$
  - Preserving equivalence

$$[\![P]\!] \; equiv \; [\![Q]\!] \;\Rightarrow\; P \; equiv' \; Q \quad \text{(soundness)}$$

$$[\![P]\!] \; equiv \; [\![Q]\!] \;\Leftarrow\; P \; equiv' \; Q \quad \text{(completeness)}$$

$$[\![P]\!] \; equiv \; [\![Q]\!] \;\Leftrightarrow\; P \; equiv' \; Q \quad \text{(full abstraction, correctness)}$$

This is one of the most popular requirements for an encoding. However it is not clear how it relates to the notion of expressive power.

Encoding the output prefix

Encoding the output prefix

- The encoding of Boudol

Encoding the output prefix

- The encoding of Boudol

Boudol [1992] provided the following encoding of π (without choice) into $\pi_{a.}$: The idea is to force both partners to proceed only when it is sure that the communication can take place, by using a sort of rendez-vous protocol

Encoding the output prefix

- The encoding of Boudol

Boudol [1992] provided the following encoding of π (without choice) into $\pi_{a.}$: The idea is to force both partners to proceed only when it is sure that the communication can take place, by using a sort of rendez-vous protocol

- $[\![\bar{x}y.P]\!] = (\nu z)(\bar{x}z \mid (z(w).(\bar{w}y \mid [\![P]\!])))$
- $[\![x(y).Q]\!] = x(z).(\nu w)(\bar{z}w \mid w(y).[\![Q]\!])$

Encoding the output prefix

- ## The encoding of Boudol

Boudol [1992] provided the following encoding of π (without choice) into π_a. : The idea is to force both partners to proceed only when it is sure that the communication can take place, by using a sort of rendez-vous protocol

- $[\![\bar{x}y.P]\!] = (\nu z)(\bar{x}z \mid (z(w).(\bar{w}y \mid [\![P]\!])))$
- $[\![x(y).Q]\!] = x(z).(\nu w)(\bar{z}w \mid w(y).[\![Q]\!])$

$[\![\cdot]\!]$ is homomorphic for all the other operators. Namely:

- $[\![0]\!] = 0$
- $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$
- $[\![(\nu x)P]\!] = (\nu x)[\![P]\!]$
- $[\![!\, P]\!] =!\, [\![P]\!]$

Encoding the output prefix

- The encoding of Boudol

Boudol [1992] provided the following encoding of π (without choice) into $\pi_{a.}$: The idea is to force both partners to proceed only when it is sure that the communication can take place, by using a sort of rendez-vous protocol

- $[\![\bar{x}y.P]\!] = (\nu z)(\bar{x}z \mid (z(w).(\bar{w}y \mid [\![P]\!])))$
- $[\![x(y).Q]\!] = x(z).(\nu w)(\bar{z}w \mid w(y).[\![Q]\!])$

$[\![\cdot]\!]$ is homomorphic for all the other operators. Namely:

- $[\![0]\!] = 0$
- $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$
- $[\![(\nu x)P]\!] = (\nu x)[\![P]\!]$
- $[\![!\, P]\!] = !\, [\![P]\!]$

- Boudol proved this encoding sound wrt the Morris ordering

Encoding the output prefix

- ## The encoding of Boudol

Boudol [1992] provided the following encoding of π (without choice) into $\pi_{a.}$: The idea is to force both partners to proceed only when it is sure that the communication can take place, by using a sort of rendez-vous protocol

- $[\![\bar{x}y.P]\!] = (\nu z)(\bar{x}z \mid (z(w).(\bar{w}y \mid [\![P]\!])))$
- $[\![x(y).Q]\!] = x(z).(\nu w)(\bar{z}w \mid w(y).[\![Q]\!])$

$[\![\cdot]\!]$ is homomorphic for all the other operators. Namely:

- $[\![0]\!] = 0$
- $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$
- $[\![(\nu x)P]\!] = (\nu x)[\![P]\!]$
- $[\![!\, P]\!] = !\, [\![P]\!]$

- Boudol proved this encoding sound wrt the Morris ordering

- **Exercise.** Define an encoding which takes only two steps instead than three. (Such a kind of encoding was defined by Honda-Tokoro [1992].)

Encoding the output prefix

Encoding the output prefix

- The encoding of Honda-Tokoro

Encoding the output prefix

- The encoding of Honda-Tokoro

Honda-Tokoro [1992] defined the following encoding of π (without choice) into π_{a}. in which the communication protocol takes two steps instead than three. The idea is to let the receiver take the initiative (instead than the sender)

Encoding the output prefix

- ## The encoding of Honda-Tokoro

  Honda-Tokoro [1992] defined the following encoding of π (without choice) into π_a, in which the communication protocol takes two steps instead than three. The idea is to let the receiver take the initiative (instead than the sender)

  - $[\![\bar{x}y.P]\!] = x(z).(\bar{z}y \mid [\![P]\!])$
  - $[\![x(y).Q]\!] = (\nu z)(\bar{x}z \mid z(y).[\![Q]\!])$

Encoding the output prefix

- ## The encoding of Honda-Tokoro

Honda-Tokoro [1992] defined the following encoding of π (without choice) into π_a. in which the communication protocol takes two steps instead than three. The idea is to let the receiver take the initiative (instead than the sender)

- $[\![\bar{x}y.P]\!] = x(z).(\bar{z}y \mid [\![P]\!])$
- $[\![x(y).Q]\!] = (\nu z)(\bar{x}z \mid z(y).[\![Q]\!])$

$[\![\cdot]\!]$ is homomorphic for all the other operators. Namely:

- $[\![0]\!] = 0$
- $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$
- $[\![(\nu x)P]\!] = (\nu x)[\![P]\!]$
- $[\![!\, P]\!] = !\, [\![P]\!]$

Encoding the output prefix

- The encoding of Honda-Tokoro

  Honda-Tokoro [1992] defined the following encoding of π (without choice) into π_a. in which the communication protocol takes two steps instead than three. The idea is to let the receiver take the initiative (instead than the sender)

  - $[\![\bar{x}y.P]\!] = x(z).(\bar{z}y \mid [\![P]\!])$
  - $[\![x(y).Q]\!] = (\nu z)(\bar{x}z \mid z(y).[\![Q]\!])$

  $[\![\cdot]\!]$ is homomorphic for all the other operators. Namely:

  - $[\![0]\!] = 0$
  - $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$
  - $[\![(\nu x)P]\!] = (\nu x)[\![P]\!]$
  - $[\![!\ P]\!] =!\ [\![P]\!]$

- Honda proved this encoding sound and "almost" complete wrt the a certain logical semantics

Encoding the output prefix

- ## The encoding of Honda-Tokoro

  Honda-Tokoro [1992] defined the following encoding of π (without choice) into $\pi_{a.}$ in which the communication protocol takes two steps instead than three. The idea is to let the receiver take the initiative (instead than the sender)

  - $[\![\bar{x}y.P]\!] = x(z).(\bar{z}y \mid [\![P]\!])$
  - $[\![x(y).Q]\!] = (\nu z)(\bar{x}z \mid z(y).[\![Q]\!])$

  $[\![\cdot]\!]$ is homomorphic for all the other operators. Namely:

  - $[\![0]\!] = 0$
  - $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$
  - $[\![(\nu x)P]\!] = (\nu x)[\![P]\!]$
  - $[\![!\,P]\!] =\, !\,[\![P]\!]$

- Honda proved this encoding sound and “almost” complete wrt the a certain logical semantics

- Honda-Tokoro defined also another encoding of π (without choice) into a polyadic version of $\pi_{a.}$ in which the communication protocol takes two steps and the sender takes the initiative. This encoding was shown in a previous lecture.

Properties of output encodings wrt testing

- Definition of testing semantics:
  - A process P **may** satisfy a test T (notation P may T) iff there exists a computation of [P | T] which reaches a state where the action ω (a special action of the test) is enabled.
  - A process P **must** satisfy a test T (notation P must T) iff every computation of [P | T] reaches a state where the action ω (a special action of the test) is enabled.
  - $P \sqsubseteq_{may} Q$ iff for every test T , if P may T then Q may T
  - $P \sqsubseteq_{must} Q$ iff for every test T , if P must T then Q must T
  - $P \simeq_x Q$ iff $P \sqsubseteq_{must} Q$ and $Q \sqsubseteq_{must} P$, X = may, must
- In contrast to weak bisimulation, testing semantics is sensitive wrt divergency
- We don't expect the encodings of output prefix to be correct wrt testing semantics (**why**?), but we would like the encoding to satisfy at least the following properties :

$$P\ may\ T \text{ iff } [\![P]\!]\ may\ [\![T]\!]$$

$$P\ must\ T \text{ iff } [\![P]\!]\ must\ [\![T]\!]$$

Properties of output encodings wrt testing

- The encodings of Boudol and Honda-Tokoro
  - Verify $P\ may\ T$ iff $[\![P]\!]\ may\ [\![T]\!]$
  - Do not verify $P\ must\ T$ iff $[\![P]\!]\ must\ [\![T]\!]$

    (they preserve may testing but not must testing)

- **Theorem** [Cacciagrano, Corradini and Palamidessi, 2004] Let [[ ]] be an encoding of π (without choice) into π_a such that:
  - [[ ]] is compositional wrt the prefixes
  - There exists a P such that [[P]] diverges

  then [[ ]] does not preserve must testing.

The problem however is only a problem of fairness:

- **Theorem** [Cacciagrano, Corradini and Palamidessi, 2004] The encodings of Boudol and Honda-Tokoro
- A) preserve must testing if we restrict to fair computations only
- B) preserve a version of must testing called "fair must testing"

Encoding internal choice in π_a

The blind choice (or internal choice) construct $P \oplus Q$ has the following semantics

$$\frac{}{P \oplus Q \xrightarrow{\tau} P} \qquad \frac{}{P \oplus Q \xrightarrow{\tau} Q}$$

In π this operator can be represented by the construct $\tau.P + \tau.Q$

Exercise: Let $\pi^{\oplus}$ be π where the + operator can only occur as a blind choice.

Give an encoding $[\![\cdot]\!] : \pi^{\oplus} \longrightarrow \pi_a$ such that $\forall P\ [\![P]\!] \sim P$

Encoding input-guarded choice in π_a

- Input-guarded choice is a construct of the form: $\sum_{i \in I} x_i(y_i).P_i$

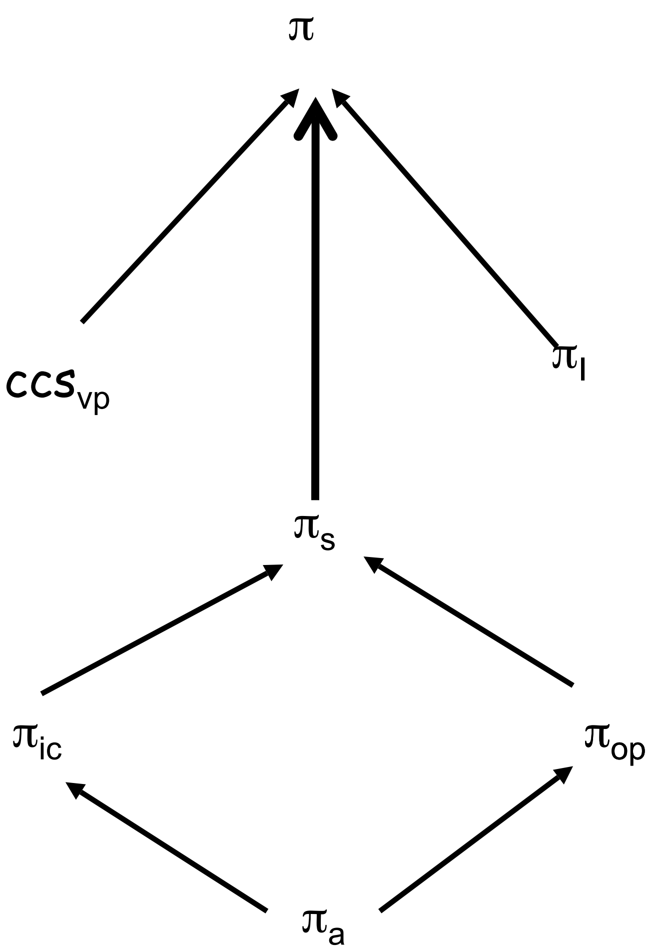
- Let π^i be π where + can only occur in an input-guarded choice. The following encoding of π^i into π_a was defined by Nestmann and Pierce [1996]

$$[\![\sum_{i \in I} x_i(y_i) P_i]\!] \quad = \quad (\nu l)(\bar{\ell}\, \mathit{true} \mid \prod_{i \in I} \mathit{Branch}_{\ell i})$$

$$\begin{array}{lcll} \mathit{Branch}_{\ell i} & = & x_i(z_i).\ell(w).(\mathit{if} & w \\ & & & \mathit{then}\ (\bar{\ell}\, \mathit{false} \mid [\![P_i]\!]) \\ & & & \mathit{else}\ (\bar{\ell}\, \mathit{false} \mid \bar{x}_i z_i)\ ) \end{array}$$

- Nestmann and Pierce proved that his encoding is fully abstract wrt a notion of equivalence called coupled bisimulation, and it does not introduce divergences.

The π-calculus hierarchy



π_a : asynchronous π

π_{ic} : asynchronous π + input-guarded choice

π_{op} : asynchronous π + output prefix

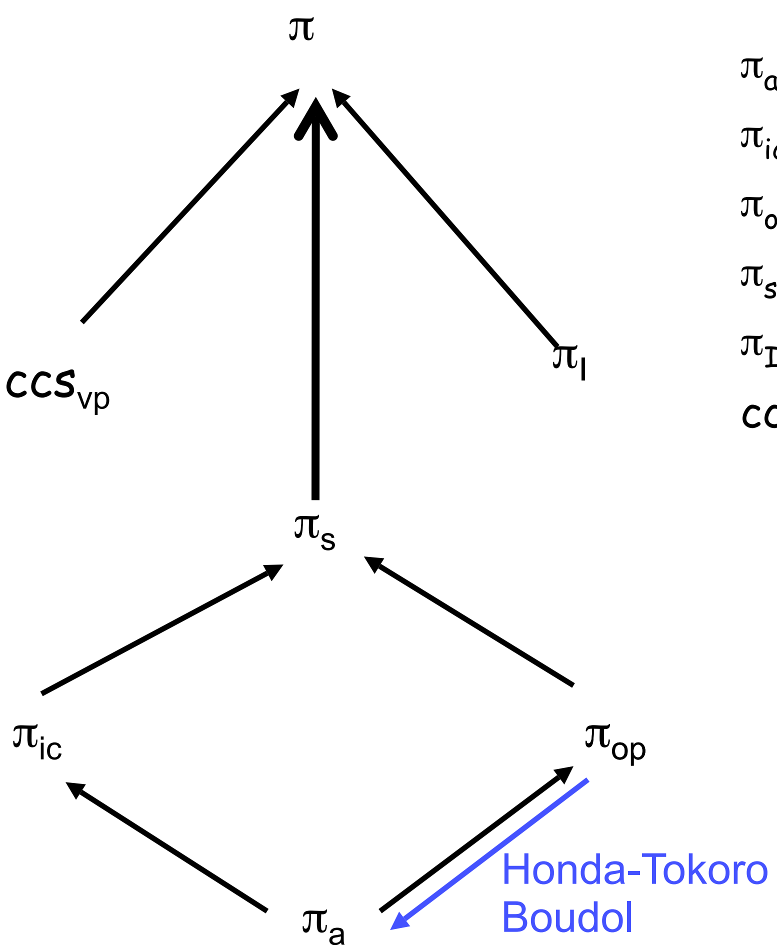
π_s : asynchronous π + separate choice

π_I : π with internal mobility (Sangiorgi)

ccs_{vp} : value-passing ccs

→ : Language inclusion

The π-calculus hierarchy



π_a : asynchronous π

π_{ic} : asynchronous π + input-guarded choice

π_{op} : asynchronous π + output prefix

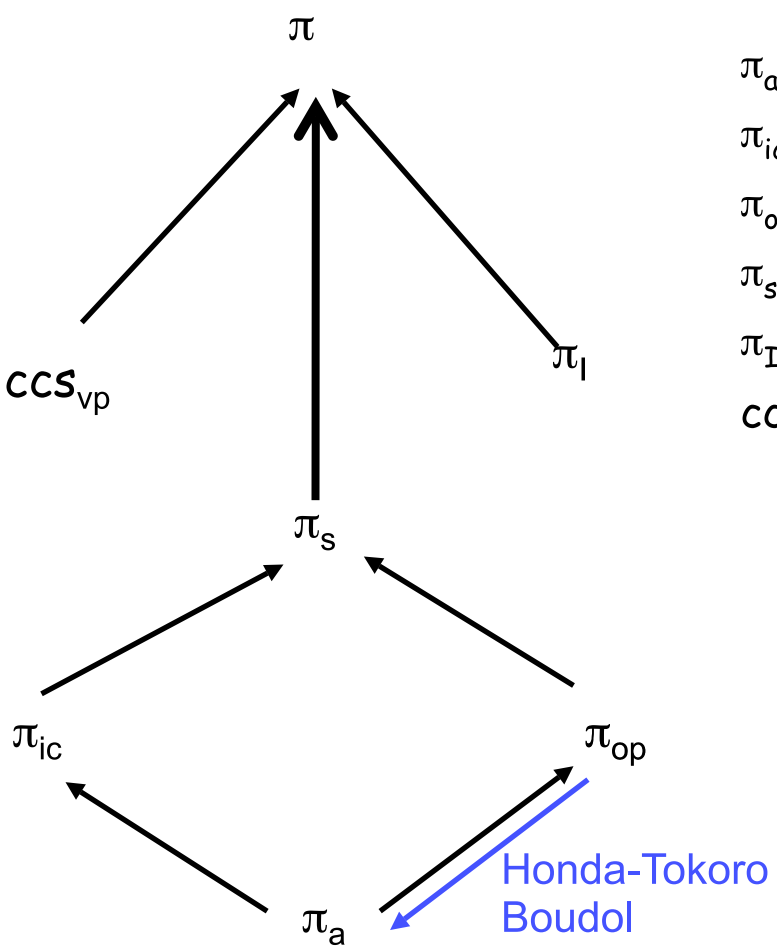
π_s : asynchronous π + separate choice

π_I : π with internal mobility (Sangiorgi)

ccs_{vp} : value-passing ccs

⟶ : Language inclusion

The π-calculus hierarchy



π_a : asynchronous π

π_{ic} : asynchronous π + input-guarded choice

π_{op} : asynchronous π + output prefix

π_s : asynchronous π + separate choice

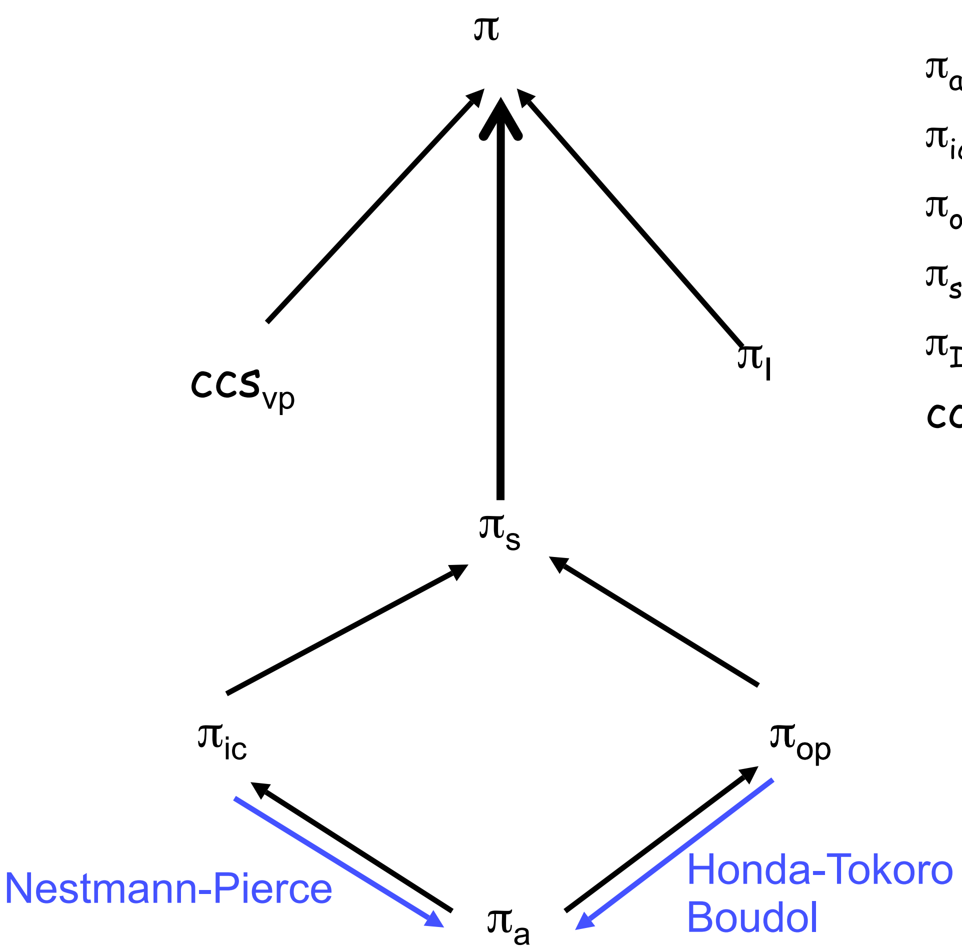
π_I : π with internal mobility (Sangiorgi)

ccs_{vp} : value-passing ccs

→ : Language inclusion

→ : Encoding

The π-calculus hierarchy



π_a : asynchronous π

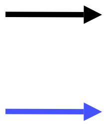
π_{ic} : asynchronous π + input-guarded choice

π_{op} : asynchronous π + output prefix

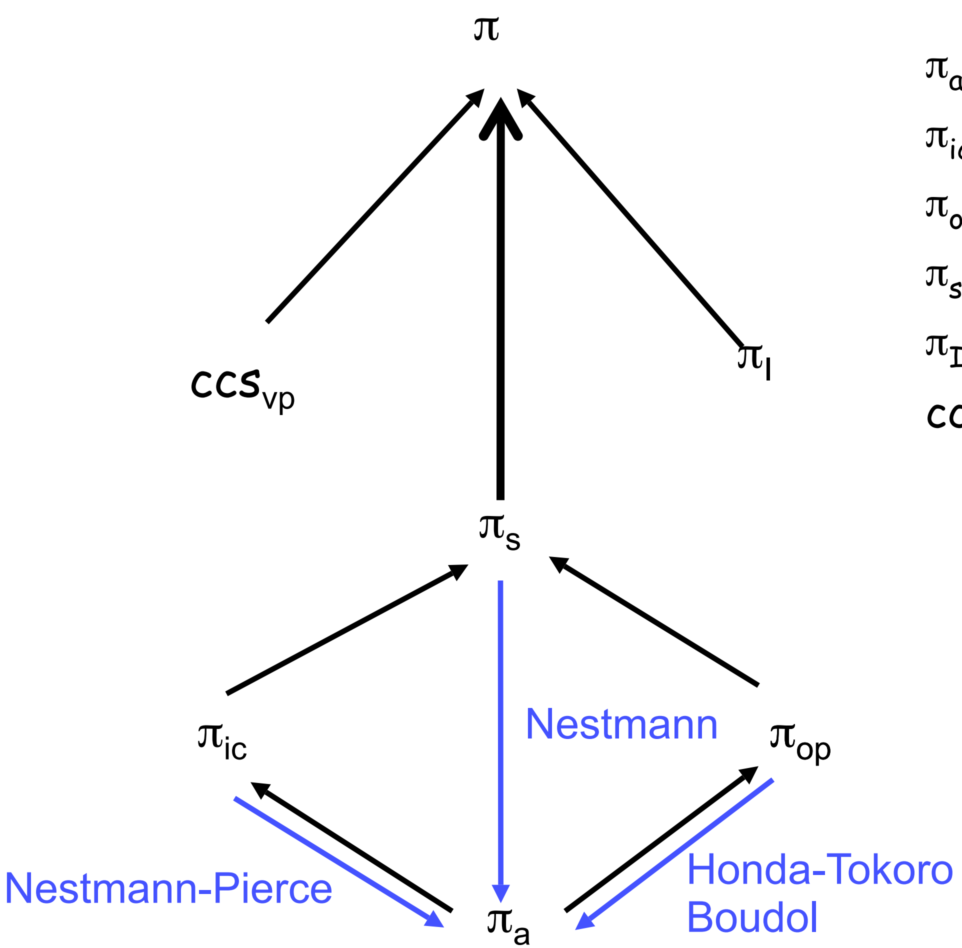
π_s : asynchronous π + separate choice

π_I : π with internal mobility (Sangiorgi)

ccs_{vp} : value-passing ccs



The π-calculus hierarchy



π_a : asynchronous π

π_{ic} : asynchronous π + input-guarded choice

π_{op} : asynchronous π + output prefix

π_s : asynchronous π + separate choice

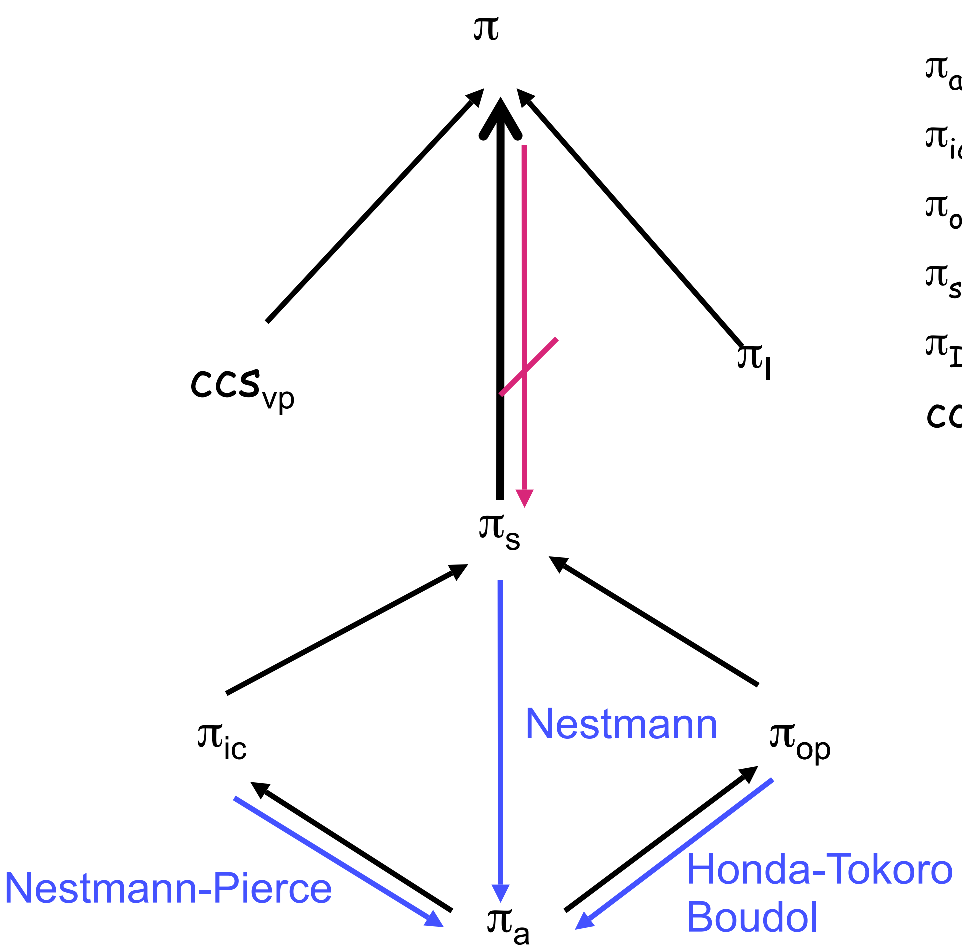
π_I : π with internal mobility (Sangiorgi)

ccs_{vp} : value-passing ccs

→ : Language inclusion

→ : Encoding

The π-calculus hierarchy



π_a : asynchronous π

π_{ic} : asynchronous π + input-guarded choice

π_{op} : asynchronous π + output prefix

π_s : asynchronous π + separate choice

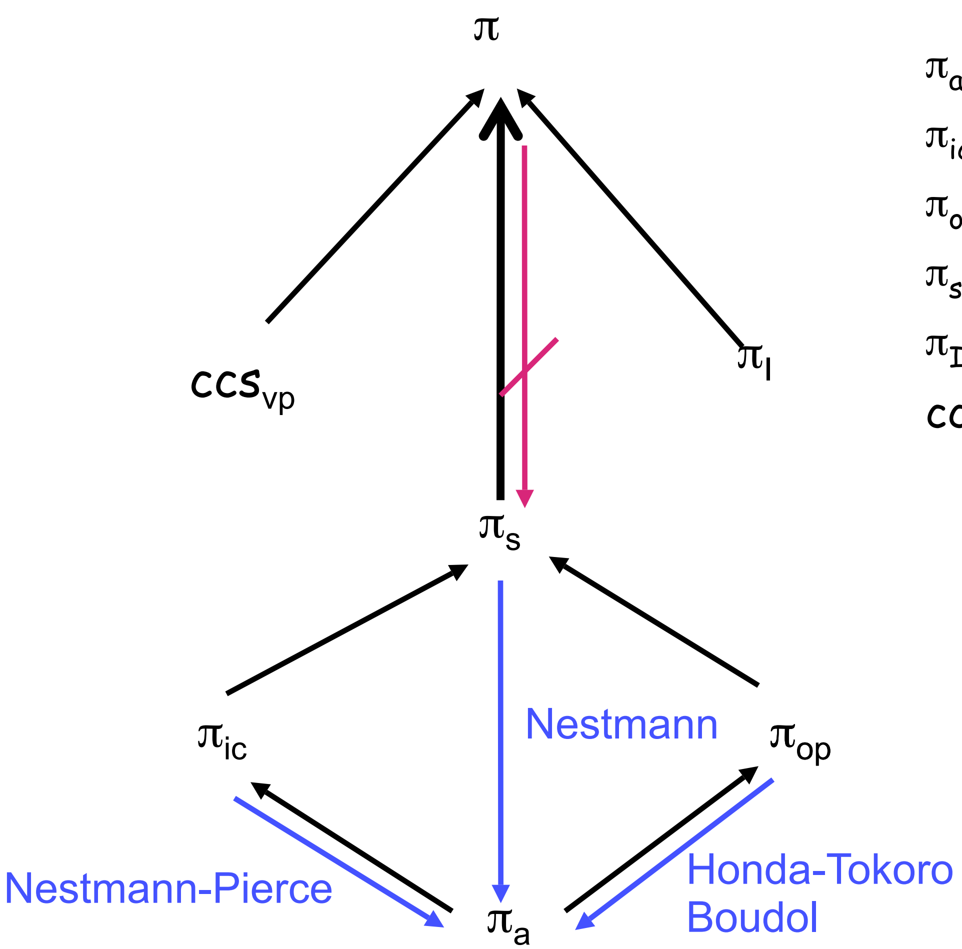
π_I : π with internal mobility (Sangiorgi)

ccs_{vp} : value-passing ccs

→ : Language inclusion

→ : Encoding

The π-calculus hierarchy



π_a : asynchronous π

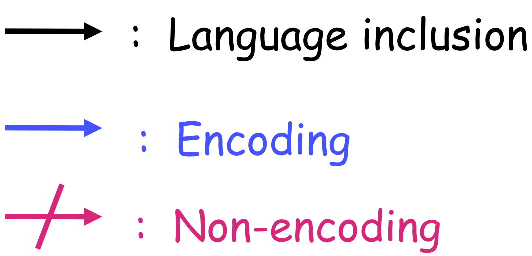
π_{ic} : asynchronous π + input-guarded choice

π_{op} : asynchronous π + output prefix

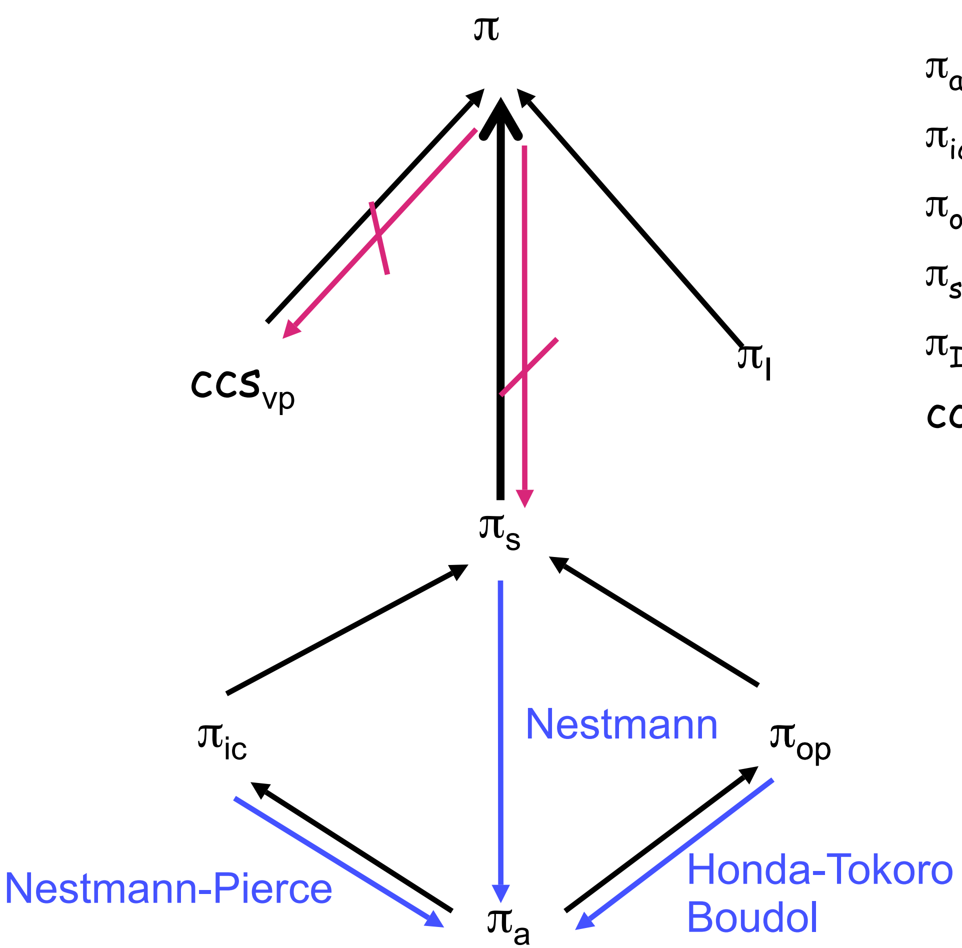
π_s : asynchronous π + separate choice

π_I : π with internal mobility (Sangiorgi)

ccs_{vp} : value-passing ccs



The π-calculus hierarchy



π_a : asynchronous π

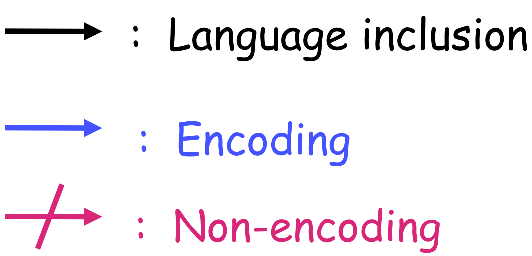
π_{ic} : asynchronous π + input-guarded choice

π_{op} : asynchronous π + output prefix

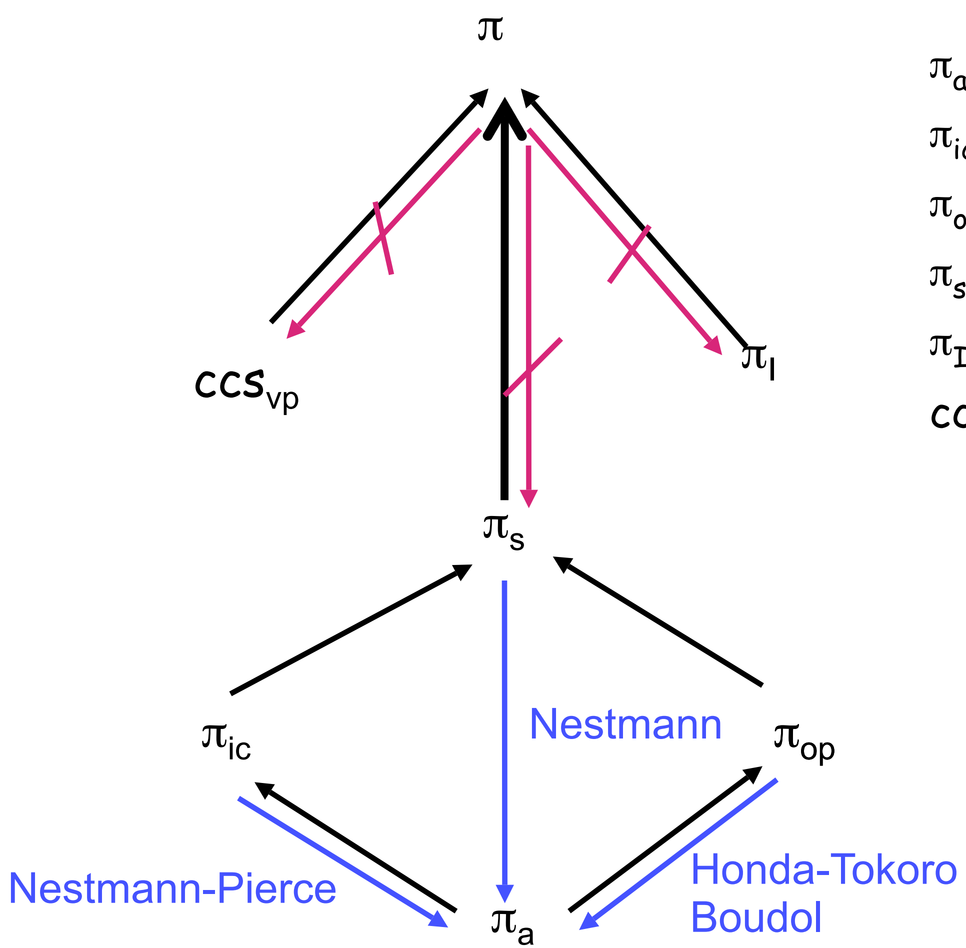
π_s : asynchronous π + separate choice

π_I : π with internal mobility (Sangiorgi)

ccs_{vp} : value-passing ccs



The π-calculus hierarchy



π_a : asynchronous π

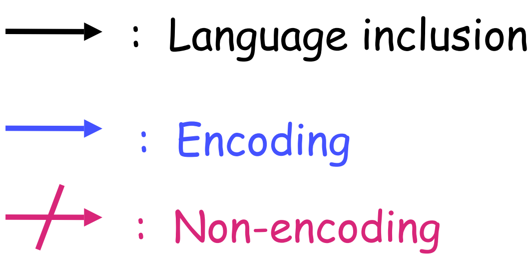
π_{ic} : asynchronous π + input-guarded choice

π_{op} : asynchronous π + output prefix

π_s : asynchronous π + separate choice

π_I : π with internal mobility (Sangiorgi)

ccs_{vp} : value-passing ccs



The separation between π and π_s

This separation result is based on the fact that it is not possible to solve the symmetric leader election problem in π_s, while it is possible in π

- Some definitions:
  - **Leader Election Problem (LEP):** All the nodes of a distributed system must agree on who is the leader. This means that in every possible computation, all the nodes must eventually output the name of the leader on a special channel out
    - No deadlock
    - No livelock
    - No conflict (only one leader must be elected, every process outputs its name and only its name)
  - **Symmetric LEP:** the LEP on a symmetric network
    - Hypergraphs and hypergraph associated to a network
    - Hypergraph automorphism
    - Orbits, well-balanced automorphism
    - Examples
    - Symmetry

The separation between π and π_s

- **Theorem:** If a network with at last two nodes has an automorphism $\sigma \neq id$ with only one orbit, then it is not possible to write in π_s a symmetric solution to the LEP

- **Corollary:** The same holds if the authomorphism is well-balanced

- **Proof** (sketch). We prove that in π_s every system trying to solve the electoral problem has at least one diverging computation

  1. If the system is symmetric, then the first action cannot be $\overline{out}\, k$

  2. As soon as a process perform an action, let all the other processes in the same orbit perform the same action as well. At the end of the round in the orbit, the system is again symmetric.

Note that the system can change communication structure dynamically

The separation between π and π_s

- Crucial point: if the action performed by P_i is a communication with P_j in the same orbit, we need to ensure that P_j can do the same action afterwards.

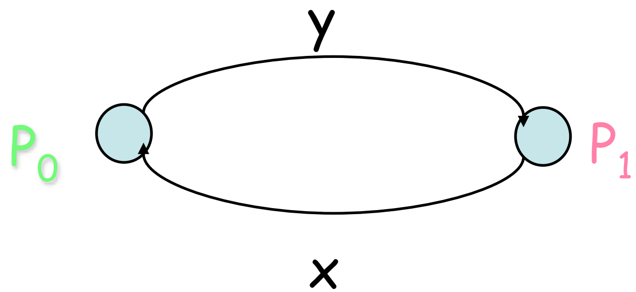
- This property holds in fact, due to the following:

- **Lemma**: Diamond lemma for π_s

- Note that in π (in π with mixed choice) the diamond lemma does not hold

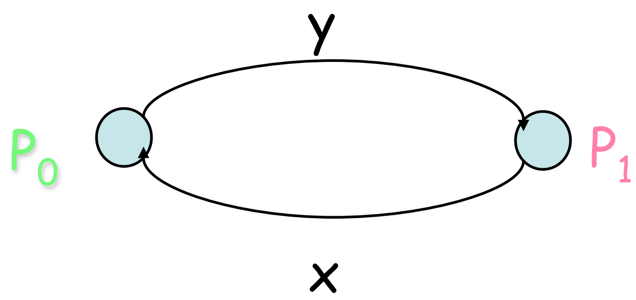
The separation between π and π_s

- **Remark:** In π (in π with mixed choice) we can easily write a symmetric solution for the LEP in a network of two nodes:



The separation between π and π_s

- **Remark:** In π (in π with mixed choice) we can easily write a symmetric solution for the LEP in a network of two nodes:



$$P_0 = x.\overline{out}\,0 + \bar{y}.\overline{out}\,1$$

$$P_1 = y.\overline{out}\,1 + \bar{x}.\overline{out}\,0$$

The separation between π and π_s

- **Corollary:** there does not exists an encoding of π (π with mixed choice) in π_s which is homomorphic wrt | and renaming, and preserves the observables on every computation.

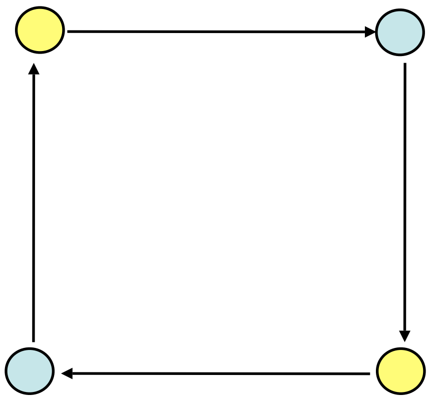
- **Proof** (scketch): *An encoding homomorphic wrt | and renaming transforms a symmetric solutions to the LEP in the source language into a symmetric solution to the LEP in the target language*

The separation between π and π_I, ccs_{vp}

- **Theorem:** If a network with at least two nodes has a well-balanced automorphism $\sigma \neq id$ such that
  - $\forall i$ and $\forall$ node P, if $\sigma^i \neq id$ then there is no arc between P and $\sigma^i(P)$,

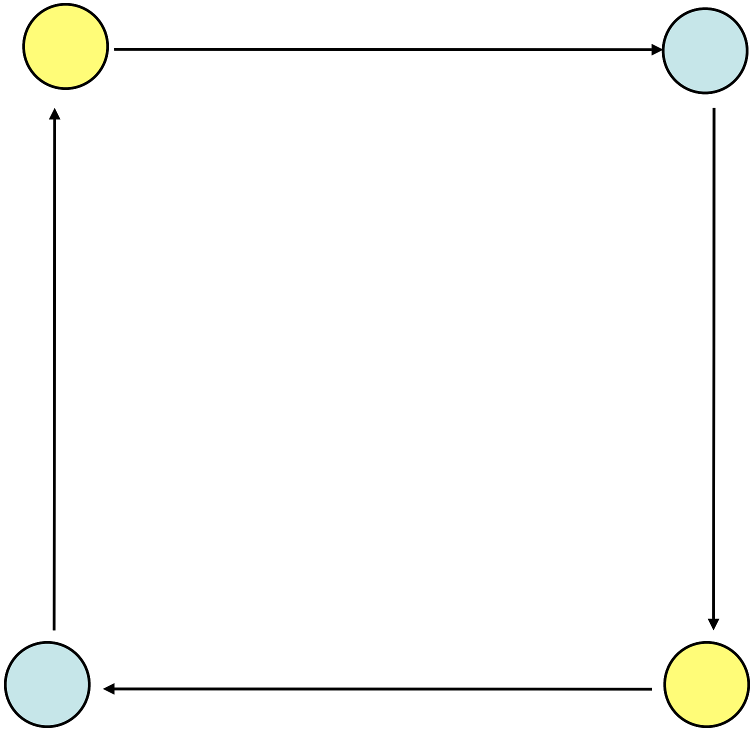
  then in π_I and ccs_{vp} there is no symmetric solution to the LEP.

- **Example**: a network which satisfies the above condition



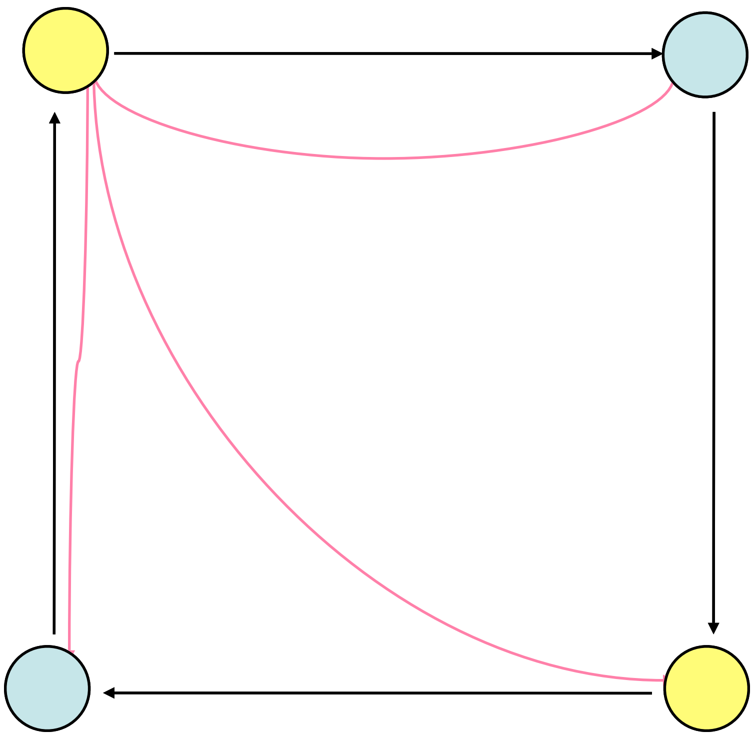
The separation between π and π_I, ccs_{vp}

- *A solution to the leader election problem for the same network in π*



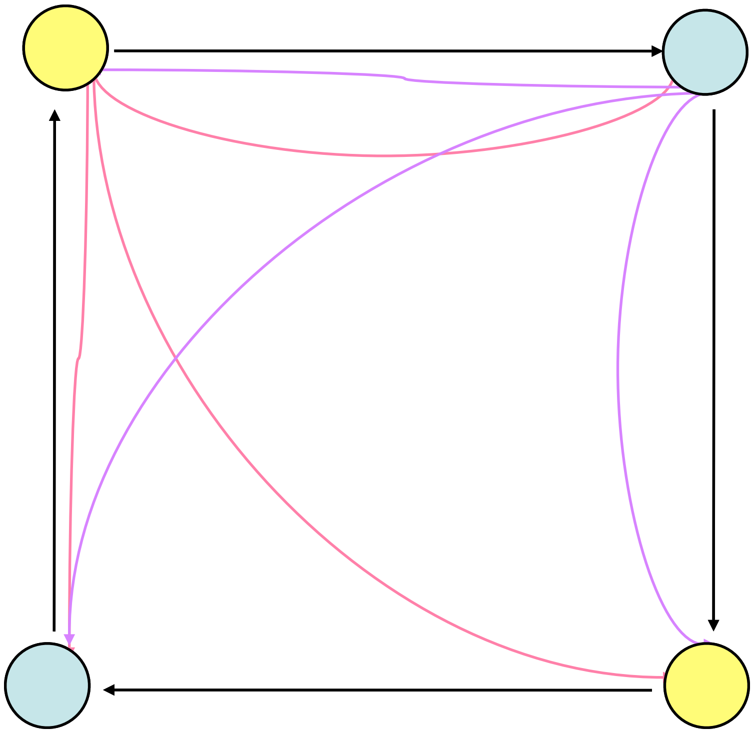
The separation between π and π_I, ccs_{vp}

- *A solution to the leader election problem for the same network in π*



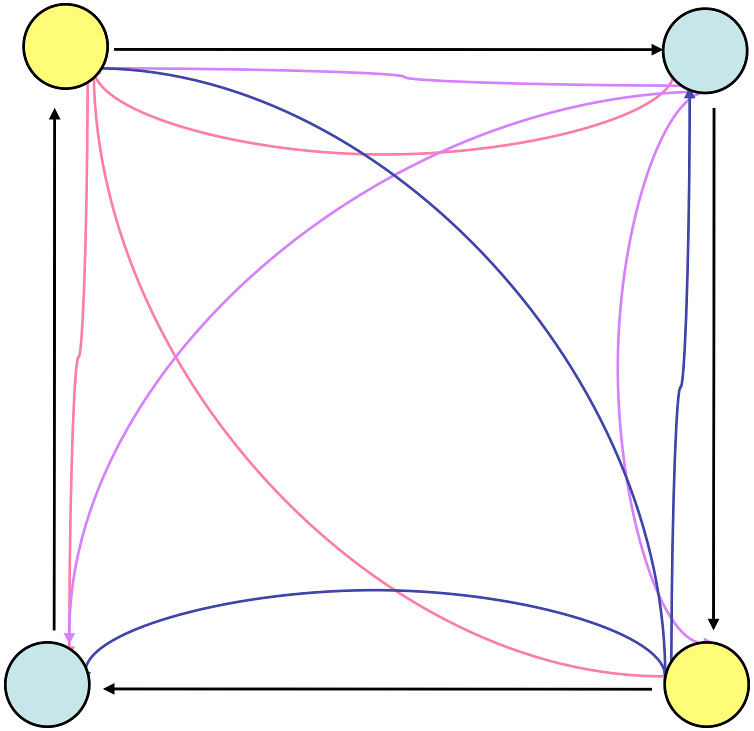
The separation between π and π_I, ccs_{vp}

- A solution to the leader election problem for the same network in π



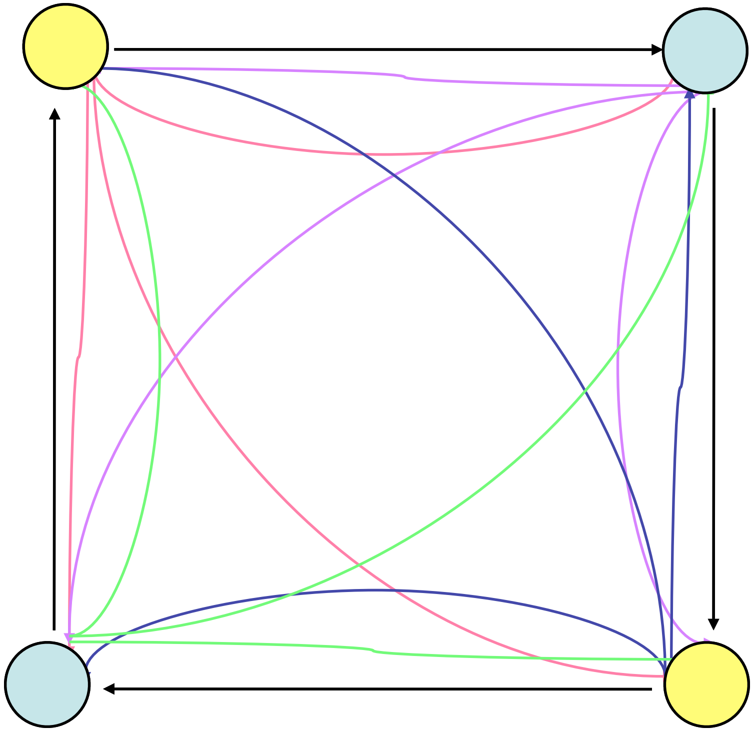
The separation between π and π_I, ccs_{vp}

- A solution to the leader election problem for the same network in π



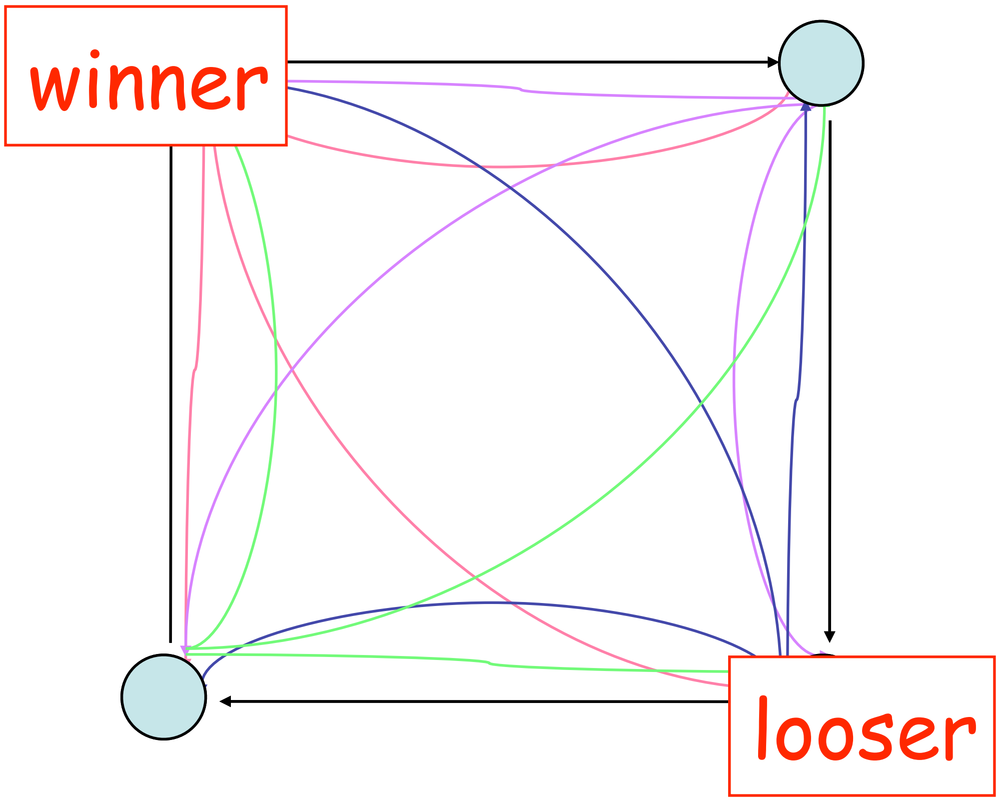
The separation between π and π_I, ccs_{vp}

- A solution to the leader election problem for the same network in π



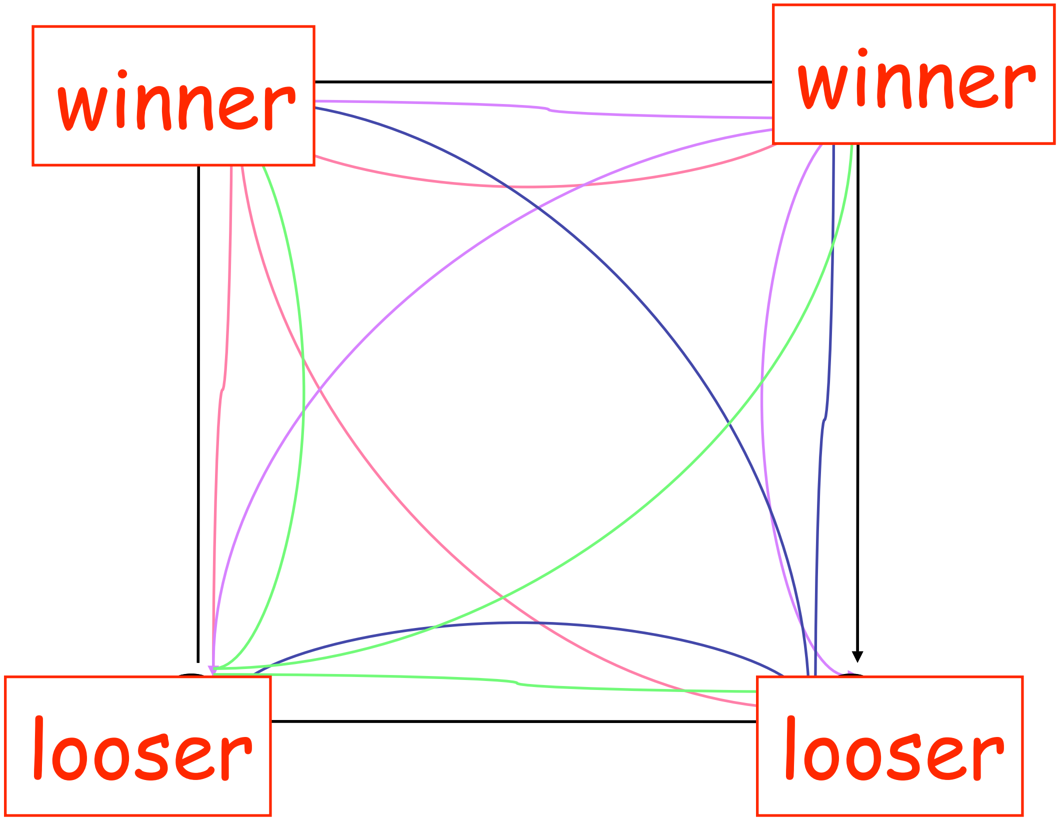
The separation between π and π_I, ccs_{vp}

- *A solution to the leader election problem for the same network in π*



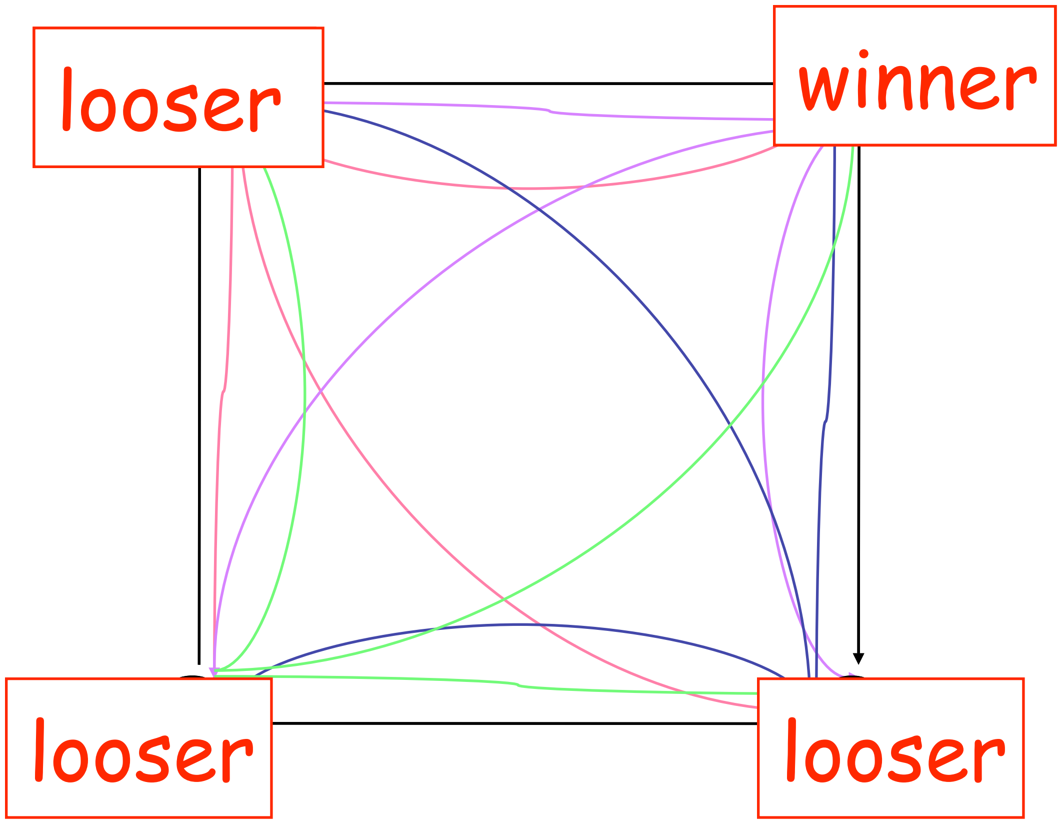
The separation between π and π_I, ccs_{vp}

- *A solution to the leader election problem for the same network in π*



The separation between π and π_I, ccs_{vp}

- *A solution to the leader election problem for the same network in π*



The separation between π and π_I, ccs_{vp}

- **Corollary:** there does not exists an encoding of π (π with mixed choice) in π_s which is homomorphic wrt | and renaming, does not increase the connectivity, and preserves the observables on every computation.

Bibliography

- Encodings of the output prefix and of the blind choice

Kohei Honda and Mario Tokoro, An Object Calculus for Asynchronous Communication. Proc. of ECOOP'91, LNCS 512, pp.133-147, Springer-Verlag, 1991 (Lecture 2). http://www.lix.polytechnique.fr/~catuscia/teaching/papers_and_books/HT91.ps

- Encodings of the input guarded choice

Uwe Nestmann and Benjamin Pierce, Decoding Choice Encodings. Journal of Information & Computation 163(1): 1-59, 2000.

http://www.lix.polytechnique.fr/~catuscia/teaching/papers_and_books/BRICS-RS-9942.ps

- impossibility results shown in these slides

Catuscia Palamidessi. Comparing the Expressive Power of the Synchronous and the Asynchronous π-calculus. Mathematical Structures in Computer Science, 13(5): 685-719, 2003.

http://www.lix.polytechnique.fr/~catuscia/papers/pi_calc/mscs.pdf

Exercises

- Prove the first Theorem at Page 10
- Formulate a notion of fair testing semantics and prove the Theorem at Page 10
- Consider the result of Nestmann and Pierce, at Page 12. Would that still holdif we replace coupled bisimulation by weak bisimulation? Motivate your answer
- Give a ring with three symmetric processes, write a program for them in the π-calculus solving the leader election problem.
- Given a ring of n symmetric processes, program them in π-calculus so to complete the graph