# Causal Computational Complexity for Processes

Romain Demangeon – Nobuko Yoshida

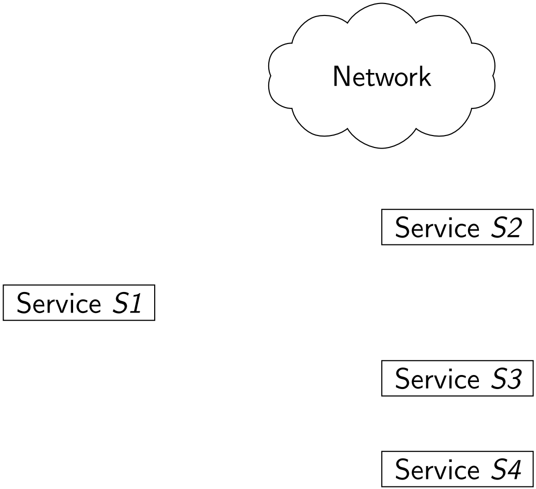
Sorbonne Université – Imperial College London

Enterrement Elica - 11/10/2018 - Sophie Germain

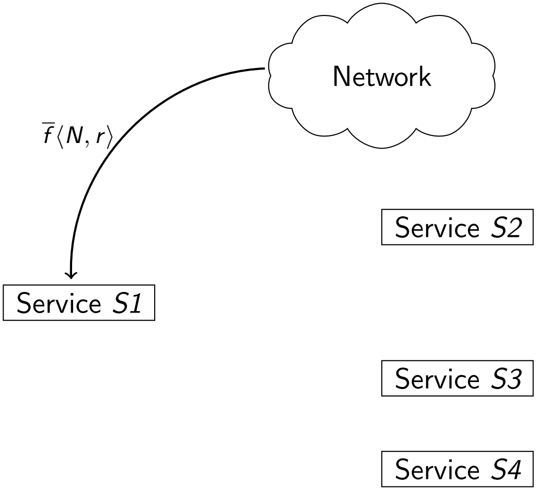
# Complexity for Processes

- Distributed systems:
  - Computation time vs. Communication time.
  - Interconnected recursive services
    - message generation can get out of hands
- Implicit complexity framework:
  - Formalism of process algebras (asynchronous $\pi$-calculus).
  - Static validation of reasonable systems.
- Contributions:
  1. Defining complexity for processes.
  2. Applying complexity analysis to the $\pi$-calculus.
  3. Refining the analysis.

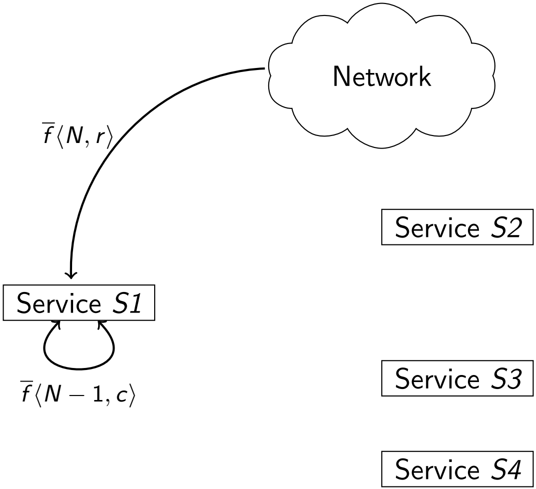
# Framework: Interacting Services



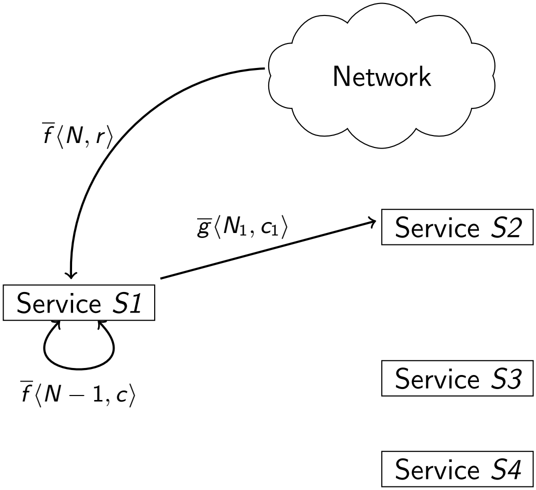
# Framework: Interacting Services



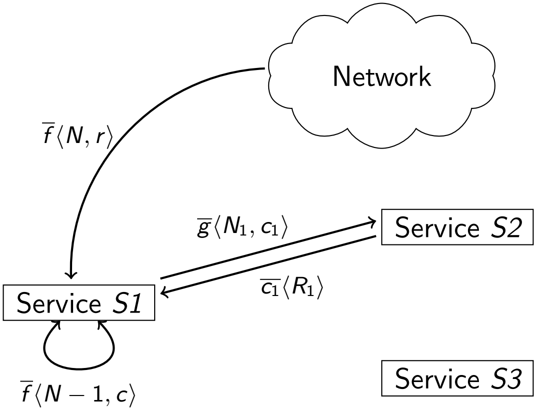
# Framework: Interacting Services



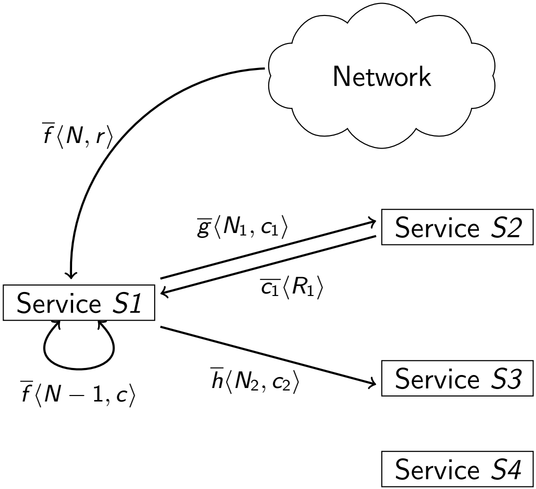
# Framework: Interacting Services



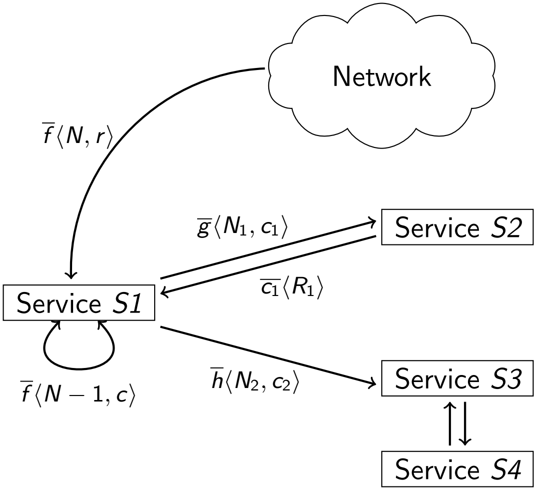
# Framework: Interacting Services



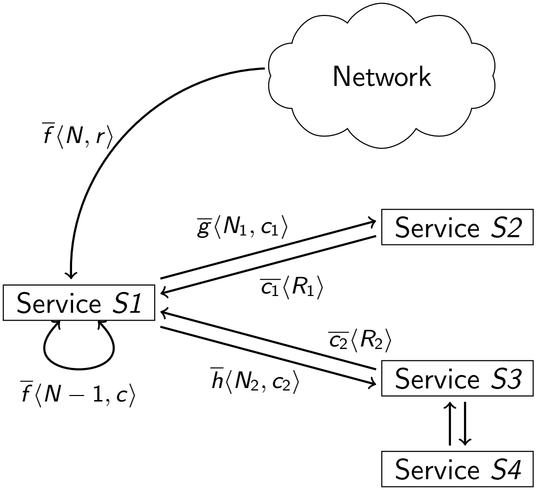
# Framework: Interacting Services



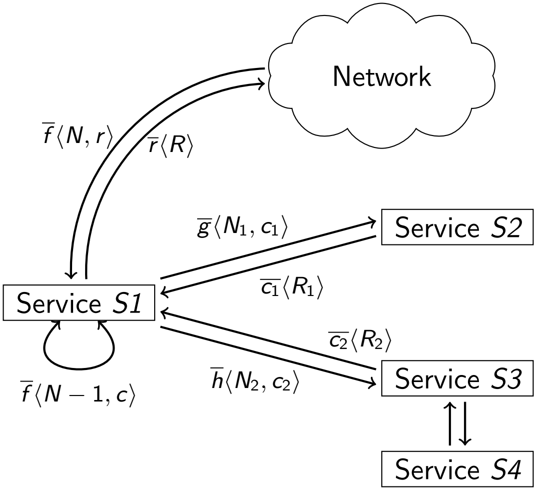
# Framework: Interacting Services



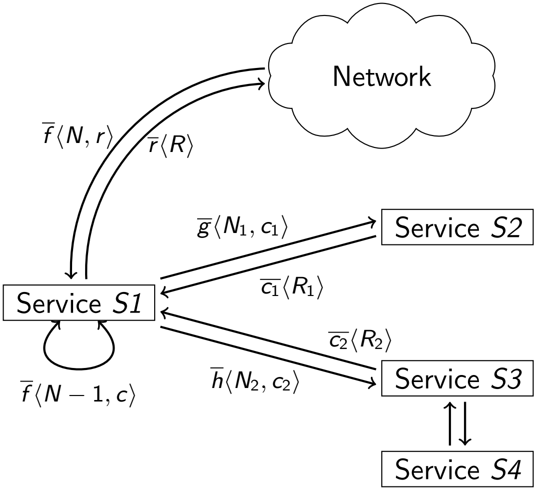
# Framework: Interacting Services



# Framework: Interacting Services

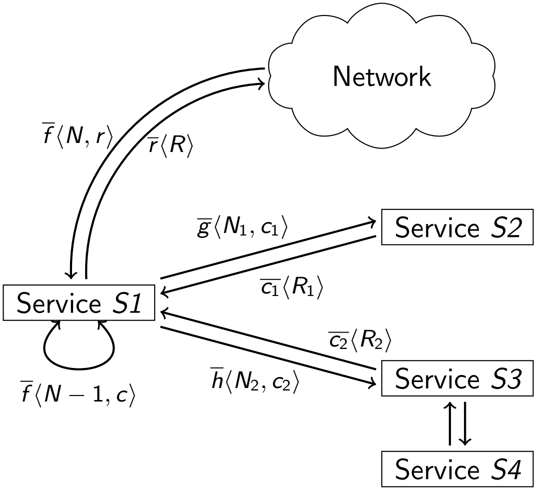


# Framework: Interacting Services




+ recursive calls for each service.

# Framework: Interacting Services




+ recursive calls for each service. Termination ? Complexity ?

# Asynchronous $\pi$-calculus: Syntax

## Expressions

- $a, b, c, \ldots$: channels,
- $1, 2, \ldots$: integers,
- $x, y, z, \ldots$: variables (channel variables and integer variables),
- $e+1, e-1$: successor and predecessor.

## Syntax

$$\begin{aligned} P \quad ::= \quad & \mathbf{0} \;\mid\; \overline{a}\langle\tilde{v}\rangle \;\mid\; a(\tilde{x}).P \;\mid\; (P \;\mid\; P) \;\mid\; ([x=0]P + [x \neq 0]P) \\ & \mid\; (\nu c)\; P \;\mid\; !a(\tilde{x}).P \end{aligned}$$

# Asynchronous $\pi$: Reduction Semantics

- (Com) $a(\tilde{x}).P \mid \overline{a}\langle\tilde{v}\rangle \rightarrow P[\tilde{v}/\tilde{x}]$
  - communication on channel $a$.
  - message $\tilde{v}$ is transmitted.
  - continuation $P$ is unlocked.
- (RCom) $!a(\tilde{x}).P \mid \overline{a}\langle\tilde{v}\rangle \rightarrow P[\tilde{v}/\tilde{x}] \mid !a(\tilde{x}).P$
  - replicated communication on channel $a$.
  - replicated process is persistent.
  - one copy of continuation $P$ is spawned and instantiated.
- Other (usual) semantics rules:
  - spectator: $P \rightarrow P'$ implies $(P \mid Q) \rightarrow (P' \mid Q)$
  - mobility: $(\nu c)\ (\overline{a}\langle c\rangle) \mid a(x).P \rightarrow (\nu c)\ (P[c/x])$
  - ...

We omit 0 after inputs: $a(x)$ for $a(x).0$.

We omit message when they are empty: $a.\overline{b}$ for $a().\overline{b}\langle\rangle$.

# Asynchronous $\pi$: Examples

- Non-Determinism: $a(x).a(y).(d \mid \overline{d}) \mid \overline{a}\langle b\rangle \mid \overline{a}\langle c\rangle$
  - either $\rightarrow a(y).(d \mid \overline{d}) \mid \overline{a}\langle c\rangle \rightarrow (d \mid \overline{d}) \rightarrow 0$
  - or $\rightarrow a(y).(d \mid \overline{d}) \mid \overline{a}\langle b\rangle \rightarrow (d \mid \overline{d}) \rightarrow 0$
- Non-Confluence: $a(x).a(y).(x \mid \overline{b}) \mid \overline{a}\langle b\rangle \mid \overline{a}\langle c\rangle$
  - either $\rightarrow a(y).(b \mid \overline{b}) \mid \overline{a}\langle c\rangle \rightarrow (b \mid \overline{b}) \rightarrow 0$
  - or $\rightarrow a(y).(c \mid \overline{b}) \mid \overline{a}\langle b\rangle \rightarrow (c \mid \overline{b}) \not\rightarrow$
- Divergence: $!a(x).\overline{a}\langle v\rangle \mid \overline{a}\langle v\rangle$
  - $\rightarrow !a(x).\overline{a}\langle v\rangle \mid \overline{a}\langle v\rangle$.
- Computation:

$$\overline{add}\langle 3, 2, d\rangle \mid \overline{add}\langle 100, 0, b\rangle$$
$$\mid !add(x, y, r).[x = 0]\overline{r}\langle y\rangle$$
$$+[x \neq 0](\nu c)\ (\overline{add}\langle x-1, y, c\rangle \mid c(z).\overline{r}\langle z+1\rangle)$$

# Asynchronous $\pi$: Simple types

- Principle: a channel type describes the way it is used.
- Syntax: $T ::= \mathsf{nat} \mid \#(\tilde{T})$
- Environment: $\Gamma = \{\tilde{v} : \tilde{T}\}$ (associate names with types)

## Examples

- $p(x,y).(\overline{x}\langle y\rangle \mid \overline{y}\langle 3\rangle)$ typable with $y : \#(\mathsf{nat})$, $x : \#(\#(\mathsf{nat}))$ et $p : \#(\#(\#(\mathsf{nat})), \#(\mathsf{nat}))$
- $p(x,y).(\overline{x}\langle y\rangle \mid \overline{x}\langle 3\rangle)$ not typable (mismatch $x : \#(\mathsf{nat})$ and $x : \#(\#(\mathsf{nat}))$
- $\overline{a}\langle a\rangle$ not typable (recursive type)

$$\frac{\Gamma \vdash P \qquad \Gamma(a) = \#(\tilde{T}) \qquad \Gamma(\tilde{v}) = \tilde{T}}{\Gamma \vdash \overline{a}\langle \tilde{v}\rangle.P} \qquad\qquad \frac{\Gamma \vdash P_1 \qquad \Gamma \vdash P_2}{\Gamma \vdash P_1 \mid P_2}$$

# Ruling Out Divergence in $\pi$

## Motivation

Services are always available, but requests must terminate.

- $D_1 = !a.\overline{a} \mid \overline{a}$
- $D_2 = !a.\overline{b} \mid !b.\overline{a} \mid \overline{a}$
- $D_3 = c(x).!a.\overline{x} \mid \overline{a} \mid \overline{c}\langle b \rangle \mid \overline{c}\langle a \rangle$

# Ruling Out Divergence in $\pi$

## Motivation

Services are always available, but requests must terminate.

- $D_1 = !a.\overline{a} \mid \overline{a} \rightarrow D_1$
- $D_2 = !a.\overline{b} \mid !b.\overline{a} \mid \overline{a}$
- $D_3 = c(x).!a.\overline{x} \mid \overline{a} \mid \overline{c}\langle b\rangle \mid \overline{c}\langle a\rangle$

# Ruling Out Divergence in $\pi$

## Motivation

Services are always available, but requests must terminate.

- $D_1 = !a.\overline{a} \mid \overline{a} \rightarrow D_1$
- $D_2 = !a.\overline{b} \mid !b.\overline{a} \mid \overline{a} \rightarrow\rightarrow D_2$
- $D_3 = c(x).!a.\overline{x} \mid \overline{a} \mid \overline{c}\langle b\rangle \mid \overline{c}\langle a\rangle$

# Ruling Out Divergence in $\pi$

## Motivation

Services are always available, but requests must terminate.

- $D_1 = !a.\overline{a} \mid \overline{a} \rightarrow D_1$
- $D_2 = !a.\overline{b} \mid !b.\overline{a} \mid \overline{a} \rightarrow\rightarrow D_2$
- $D_3 = c(x).!a.\overline{x} \mid \overline{a} \mid \overline{c}\langle b \rangle \mid \overline{c}\langle a \rangle \rightarrow D_1 \mid \overline{c}\langle b \rangle$

# Ruling Out Divergence in $\pi$

## Motivation

Services are always available, but requests must terminate.

- $D_1 = !a.\overline{a} \mid \overline{a} \rightarrow D_1$
- $D_2 = !a.\overline{b} \mid !b.\overline{a} \mid \overline{a} \rightarrow\rightarrow D_2$
- $D_3 = c(x).!a.\overline{x} \mid \overline{a} \mid \overline{c}\langle b\rangle \mid \overline{c}\langle a\rangle \rightarrow D_1 \mid \overline{c}\langle b\rangle$

- Usual termination analyses: find a strict decreasing.
- Examples can use $\overline{a}$ to produce $\overline{a}$: loop.
- Simple types:
  - for $\lambda$, simple typing guarantees termination,
    - Encoding of $\lambda$ into $\pi$.
    - [*Strong Normalisation in the $\pi$-calculus*, Berger, Honda, Yoshida 04]
  - in $\pi$, simple typing does not rule out divergence.
    - examples above are typable.

# A first type system

[*Ensuring Termination by Typability*, Deng, Sangiorgi, 06]

$$(\text{Nil})\frac{}{\Gamma \vdash \mathbf{0} : 0} \qquad (\text{Par})\frac{\Gamma \vdash P_1 : n_1 \qquad \Gamma \vdash P_2 : n_2}{\Gamma \vdash P_1 \mid P_2 : \max(n_1, n_2)}$$

$$(\text{Res})\frac{\Gamma \vdash P : n \qquad \Gamma(a) = \#^k(\tilde{T})}{\Gamma \vdash (\nu a)\ P : n} \qquad (\text{Out})\frac{\Gamma(a) = \#^k(\tilde{T}) \qquad \Gamma(\tilde{v}) = \tilde{T}}{\Gamma \vdash \overline{a}\langle v\rangle : k}$$

$$(\text{In})\frac{\Gamma \vdash P : n \qquad \Gamma(a) = \#^k(\tilde{T}) \qquad \Gamma(\tilde{x}) = \tilde{T}}{\Gamma \vdash a(\tilde{x}).P : n}$$

$$(\text{Rep})\frac{\Gamma \vdash P : n \qquad \Gamma(a) = \#^k(\tilde{T}) \qquad \Gamma(\tilde{x}) = \tilde{T} \qquad k > n}{\Gamma \vdash !a(\tilde{x}).P : 0}$$

- Outputs inside the continuation $P$ of replication $!a.P$ have strictly smaller levels than $a$.
  - $D_1 = !a^n.\overline{a}^n \mid \overline{a}^n$
  - $D_2 = !a^n.\overline{b}^k \mid !b^k.\overline{a}^n \mid \overline{a}^n$ .
  - $D_3 = c^t(x).!a^n.\overline{x}^k \mid \overline{a}^n \mid \overline{c}^t\langle a\rangle \mid \overline{c}^t\langle b\rangle$.

# A first type system

[*Ensuring Termination by Typability*, Deng, Sangiorgi, 06]

$$\text{(Nil)}\,\frac{}{\Gamma \vdash \mathbf{0} : 0} \qquad \text{(Par)}\,\frac{\Gamma \vdash P_1 : n_1 \qquad \Gamma \vdash P_2 : n_2}{\Gamma \vdash P_1 \mid P_2 : \max(n_1, n_2)}$$

$$\text{(Res)}\,\frac{\Gamma \vdash P : n \qquad \Gamma(a) = \#^k(\tilde{T})}{\Gamma \vdash (\nu a)\ P : n} \qquad \text{(Out)}\,\frac{\Gamma(a) = \#^k(\tilde{T}) \qquad \Gamma(\tilde{v}) = \tilde{T}}{\Gamma \vdash \overline{a}\langle v \rangle : k}$$

$$\text{(In)}\,\frac{\Gamma \vdash P : n \qquad \Gamma(a) = \#^k(\tilde{T}) \qquad \Gamma(\tilde{x}) = \tilde{T}}{\Gamma \vdash a(\tilde{x}).P : n}$$

$$\text{(Rep)}\,\frac{\Gamma \vdash P : n \qquad \Gamma(a) = \#^k(\tilde{T}) \qquad \Gamma(\tilde{x}) = \tilde{T} \qquad k > n}{\Gamma \vdash !a(\tilde{x}).P : 0}$$

- Outputs inside the continuation $P$ of replication $!a.P$ have strictly smaller levels than $a$.
  - $D_1 = !a^n.\overline{a}^n \mid \overline{a}^n$ not typable: $n > n$
  - $D_2 = !a^n.\overline{b}^k \mid !b^k.\overline{a}^n \mid \overline{a}^n$ .
  - $D_3 = c^t(x).!a^n.\overline{x}^k \mid \overline{a}^n \mid \overline{c}^t\langle a \rangle \mid \overline{c}^t\langle b \rangle$.

# A first type system

[*Ensuring Termination by Typability*, Deng, Sangiorgi, 06]

$$\text{(Nil)}\,\frac{}{\Gamma \vdash \mathbf{0} : 0} \qquad \text{(Par)}\,\frac{\Gamma \vdash P_1 : n_1 \qquad \Gamma \vdash P_2 : n_2}{\Gamma \vdash P_1 \mid P_2 : \max(n_1, n_2)}$$

$$\text{(Res)}\,\frac{\Gamma \vdash P : n \qquad \Gamma(a) = \#^k(\tilde{T})}{\Gamma \vdash (\nu a)\ P : n} \qquad \text{(Out)}\,\frac{\Gamma(a) = \#^k(\tilde{T}) \qquad \Gamma(\tilde{v}) = \tilde{T}}{\Gamma \vdash \overline{a}\langle v\rangle : k}$$

$$\text{(In)}\,\frac{\Gamma \vdash P : n \qquad \Gamma(a) = \#^k(\tilde{T}) \qquad \Gamma(\tilde{x}) = \tilde{T}}{\Gamma \vdash a(\tilde{x}).P : n}$$

$$\text{(Rep)}\,\frac{\Gamma \vdash P : n \qquad \Gamma(a) = \#^k(\tilde{T}) \qquad \Gamma(\tilde{x}) = \tilde{T} \qquad k > n}{\Gamma \vdash !a(\tilde{x}).P : 0}$$

- Outputs inside the continuation $P$ of replication $!a.P$ have strictly smaller levels than $a$.
  - $D_1 = !a^n.\overline{a}^n \mid \overline{a}^n$ not typable: $n > n$
  - $D_2 = !a^n.\overline{b}^k \mid !b^k.\overline{a}^n \mid \overline{a}^n$ not typable: $n > k > n$.
  - $D_3 = c^t(x).!a^n.\overline{x}^k \mid \overline{a}^n \mid \overline{c}^t\langle a\rangle \mid \overline{c}^t\langle b\rangle$.

# A first type system

[*Ensuring Termination by Typability*, Deng, Sangiorgi, 06]

$$(\text{Nil})\frac{}{\Gamma \vdash \mathbf{0} : 0} \qquad (\text{Par})\frac{\Gamma \vdash P_1 : n_1 \qquad \Gamma \vdash P_2 : n_2}{\Gamma \vdash P_1 \mid P_2 : \max(n_1, n_2)}$$

$$(\text{Res})\frac{\Gamma \vdash P : n \qquad \Gamma(a) = \#^k(\tilde{T})}{\Gamma \vdash (\nu a)\ P : n} \qquad (\text{Out})\frac{\Gamma(a) = \#^k(\tilde{T}) \qquad \Gamma(\tilde{v}) = \tilde{T}}{\Gamma \vdash \overline{a}\langle v\rangle : k}$$

$$(\text{In})\frac{\Gamma \vdash P : n \qquad \Gamma(a) = \#^k(\tilde{T}) \qquad \Gamma(\tilde{x}) = \tilde{T}}{\Gamma \vdash a(\tilde{x}).P : n}$$

$$(\text{Rep})\frac{\Gamma \vdash P : n \qquad \Gamma(a) = \#^k(\tilde{T}) \qquad \Gamma(\tilde{x}) = \tilde{T} \qquad k > n}{\Gamma \vdash !a(\tilde{x}).P : 0}$$

- Outputs inside the continuation $P$ of replication $!a.P$ have strictly smaller levels than $a$.
  - $D_1 = !a^n.\overline{a}^n \mid \overline{a}^n$ not typable: $n > n$
  - $D_2 = !a^n.\overline{b}^k \mid !b^k.\overline{a}^n \mid \overline{a}^n$ not typable: $n > k > n$.
  - $D_3 = c^t(x).!a^n.\overline{x}^k \mid \overline{a}^n \mid \overline{c}^t\langle a\rangle \mid \overline{c}^t\langle b\rangle$.
    $c : \#^t(\#^n(T))$ so $n = k$ and $n > k$.

# Weight Decreasing

## Reduction

- $T_1 = !a.(\overline{b} \mid \overline{b} \mid \overline{c}) \mid !b.(\overline{c} \mid \overline{c})$
- $T_1 \mid \overline{a} \mid \overline{b} \rightarrow T_1 \mid \overline{a} \mid \overline{c} \mid \overline{c} \rightarrow T_1 \mid \overline{b} \mid \overline{b} \mid \overline{c} \mid \overline{c} \mid \overline{c} \rightarrow\rightarrow\not\rightarrow$

- Soundness: every typed process is terminating.
- Completeness:
  - Is every terminating process typable ?
    - No (decidable type system).
  - Is every terminating process bisimilar to a typable process ?
    - Yes (reduction is finitely branching).
    - Not interesting (reduction bisimulation).

# Weight Decreasing

## Reduction

- $T_1 = !a^3.(\overline{b}^2 \mid \overline{b}^2 \mid \overline{c}^1) \mid !b^2.(\overline{c}^1 \mid \overline{c}^1)$
- $T_1 \mid \overline{a} \mid \overline{b} \rightarrow_2 T_1 \mid \overline{a} \mid \overline{c} \mid \overline{c} \rightarrow_3 T_1 \mid \overline{b} \mid \overline{b} \mid \overline{c} \mid \overline{c} \mid \overline{c} \rightarrow_2 \rightarrow_2 \not\rightarrow$
- $\{3, 2\} \rightarrow_2 \{3, 1, 1\} \rightarrow_3 \{2, 2, 1, 1, 1\} \rightarrow_2 \{2, 1, 1, 1, 1, 1\} \rightarrow_2 \{1, 1, 1, 1, 1, 1, 1\}$

- Soundness: every typed process is terminating.
- Completeness:
  - Is every terminating process typable ?
    - No (decidable type system).
  - Is every terminating process bisimilar to a typable process ?
    - Yes (reduction is finitely branching).
    - Not interesting (reduction bisimulation).

# Further Type Systems

- Motivation: expressiveness of type system.
- "System 2" in [DengSangiorgi06] ensures termination of a value-passing $\pi$ by allowing recursive calls and ensuring decreasing of arguments.
  - $!add(x, y, r).([x = 0]\overline{r}\langle y\rangle + [n \neq 0](\nu c)\ (\overline{add}\langle x - 1, y, c\rangle \ \mid\ c(x).\overline{r}\langle x + 1\rangle))$
  - $x$ cannot be negative.
  - *add* is "calling itself" on strictly smaller arguments.
- "System 3" compares multiset of input levels against multisets of output levels.
  - $!a.a.(\overline{a})$: innocuous as $\{\mathbf{lv}(a), \mathbf{lv}(a)\}$ produces $\{\mathbf{lv}(a)\}$.
  - $!a.b.(\overline{a} \ \mid\ \overline{c})$: ok if $\mathbf{lv}(b) > \mathbf{lv}(c)$.
- "System 4" is an involved system using ordering between parameters to allow list typing.
  - $!p(a, b).a(m).(\overline{b}\langle m\rangle \ \mid\ \overline{p}\langle a, b\rangle) \ \mid\ \overline{p}\langle a_1, a_2\rangle \ \mid\ \overline{p}\langle a_2, a_3\rangle \ \mid\ \overline{a_1}\langle 3\rangle$
  - $p : \#_{1>2}(\#(\text{nat}), \#(\text{nat}))$

# Inference

[*On the Complexity of Termination Inference for Processes*, D., Hirschkoff, Kobayashi, Sangiorgi, 07]

- Inference for all sytems is decidable.
- Inference for System 1 is polynomial:
  - Identify names which must have the same type (and same level).
    - in $a(x).a(y).P$, names $x$ and $y$ must have same type.
  - Search process for level constraints: " $\mathbf{lv}(a) < \mathbf{lv}(b)$".
  - Perform a topological sort on constraints.
- Inference for System 3 is NP-complete:
  - Reduction of 3-Sat to level inference problem.

- Follow-up of "System 2": What about complexity ?
  - Can we ensure complexity bounds on the number of reductions through types ?

# Polynomial Complexity

$$\mathbf{A} \;=\; !add(x,y,r).[x=0]\ \overline{r}\langle y\rangle + [x\neq 0]\ (\nu c)\ (\overline{add}\langle x-1,y,c\rangle \ \mid\ c(z).\overline{r}\langle z+1\rangle)$$

$$\mathbf{P} \;=\; \mathbf{A} \mid !mult(x,y,r).[x=0]\ \overline{r}\langle 0\rangle \\ +[x\neq 0]\ (\nu d_1,d_2)\ (\overline{mult}\langle x-1,y,d_1\rangle \ \mid\ d_1(res).\overline{add}\langle y,res,d_2\rangle \ \mid\ d_2(z).\overline{r}\langle z\rangle)$$

$$\mathbf{F} \;=\; \mathbf{P} \mid !fact(x,r).[x=0]\ \overline{r}\langle 1\rangle \\ +[x\neq 0]\ (\nu d_1,d_2)\ (\overline{fact}\langle x-1,d_1\rangle \ \mid\ d_1(res).\overline{mult}\langle x,res,d_2\rangle \ \mid\ d_2(z).\overline{r}\langle z\rangle)$$

- Service **A** → recursive addition → polynomial.
- Service **P** → recursive multiplication → polynomial.
- Service **F** → recursive factorial function → not polynomial.

- Formal complexity:
  - in literature → reduction semantics
    - $(\mathbf{A} \mid \overline{a}\langle N,M,c\rangle) \to \Theta(N)$ reductions.
  - in our work → service complexity:
    - input $a(N,M,c)$ causes $\Theta(N)$ transitions.

# Implicit Complexity for $\pi$-calculus

## [BellantoniCook94]

*A new recursion-theoretic characterisation of polytime functions*

- Predicativity of recursion $\leftrightarrow$ Polynomial recursive functions:
  - result of recursive calls cannot be used in recursion position.
- Syntactic characterisation $\rightarrow$ safe and unsafe:
  - recursion can be done on unsafe parameters only.
  - recursive calls only appear as safe argument.

- Applied to a type system for $\pi$ (similar to [DengSangiorgi06]):
  - control of replicated inputs,
  - names are compared with levels,
  - decreasing in levels ensures termination.

# Polytime Functions Characterisation

[*A new recursion-theoretic characterization of polytime functions*, Bellantoni, Cook, 94]

- Background: recursion theory
  - **type** nat = Z | S **of** nat
- Counting function calls w.r.t the parameters.
- ```
  let rec double = function
        Z -> Z
      | S x -> S (S (double x))
  ```
  → Linear complexity.
- ```
  let rec add x y = match x with
        Z -> y
      | S z -> S (add z y)
  ```
  → Linear complexity.

# Polytime Functions Characterisation (II)

- ```
  let rec exp_v1 = function
      Z -> S Z
    | S x -> add (exp_v1 x) (exp_v1 x)
  ```

  → Exponential complexity.

- ```
  let rec exp_v2 = function
      Z -> S Z
    | S x -> double (exp_v2 x)
  ```

  → Exponential complexity.

- Predicativity: no recursion performed on recursive calls.
- Two kinds of parameters:
  - unsafe: will be used to perform a recursion.
  - safe: will not be used in a recursion.
  - Constraint: the recursive call do not appear in unsafe position.

# ICC: Polytime Functions Characterisation (III)

- ```
  let rec add x y = match x with
       Z -> y
     | S z -> S (add z y)

  let rec mult x y = match x with
       Z -> Z
     | S z -> (add y (mult z y) )

  let rec fact x = match x with
       Z -> S Z
     | S z -> (mult x (fact z) )
  ```
- recursive call fact z appears in unsafe position.
  - fact is rejected.
- Soundness: abides to the rule ⇒ executes in polynomial time.
- Completeness: computable by a polynomial program ⇒ computable by a verified program.
  ```

# Asynchronous $\pi$-calculus: Causal Semantics

[*Causality for mobile processes*, Degano, Priami, 95]

- No commutativity, associativity, neutral for $\mid$ in $\equiv$.
- Processes seen as binary trees of parallel compositions.
- Transitions decorated with paths (0 left, 1 right).

$$\begin{array}{rl} & b.(\overline{a}\langle v\rangle \mid a(y).(\overline{d_1} \mid \overline{d_2})) \\ \xrightarrow{b} & \overline{a}\langle v\rangle \mid a(y).(\overline{d_1} \mid \overline{d_2}) \\ \xrightarrow{\langle 0.\overline{a}\langle v\rangle, 1.a(y)\rangle} & 0 \mid (\overline{d_1} \mid \overline{d_2}) \\ \xrightarrow{11.\overline{d_2}} & 0 \mid (\overline{d_1} \mid 0) \end{array}$$

# Asynchronous $\pi$-calculus: Causal Semantics (II)

- Causality relation between transitions (labels).
- Modified towards complexity analysis.
- Dependency relation:

$$
\begin{array}{rcll}
a(\tilde{v}) & \subseteq & l & \\
!a(\tilde{v}) & \subseteq & 1.l & \\
i.l & \subseteq & i.l' & \text{if } l \subseteq l' \text{ for } i \in \{0,1\} \\
\langle l_0, l_1 \rangle & \subseteq & \langle l'_0, l'_1 \rangle & \text{if } l_i \subseteq l'_j \text{ for some } i, j \text{ and } !a(\tilde{v}) \in \langle l_0, l_1 \rangle \\
\langle l_0, l_1 \rangle & \subseteq & l' & \text{if } l_i \subseteq l' \text{ for some } i \text{ and } !a(\tilde{v}) \in \langle l_0, l_1 \rangle \\
l & \subseteq & \langle l'_0, l'_1 \rangle & \text{if } l \subseteq l'_j \text{ for some } j
\end{array}
$$

(closed by reflexivity and transitivity)

# Causal Complexity

## Definition: Complexity

- In a (possibly) infinite computation $S : P \xrightarrow{l_1} P_1 \xrightarrow{l_2} P_2 \ldots,$

  dependent set $\mathbf{D}(l_m)_S = \{l_k \in S \mid l_m \subseteq l_k\}$.
- $P$ is bound by function $\mathcal{F} : \mathbb{N} \to \mathbb{N}$

  $$\forall S,\ l_m = \theta.!a(\tilde{v}) \Rightarrow |\mathbf{D}(l_m)_S| \leq \mathcal{F}(|\tilde{v}|).$$

$$P \xrightarrow{\theta_1.!a(10,c)} \xrightarrow{\tau} \xrightarrow{\tau} \xrightarrow{\theta_2.!a(10000,d)} \xrightarrow{\tau} \xrightarrow{\tau} \xrightarrow{\tau} \xrightarrow{\theta_3.\overline{d}\langle 28\rangle} \xrightarrow{\tau} \xrightarrow{\tau} \xrightarrow{\theta_4.b(10)} \xrightarrow{\tau} \ldots$$

# [BellantoniCook94] applied to $\pi$-calculus

$$\begin{aligned}
\mathbf{A} \quad &= \quad !add(x, y, r).[x = 0]\ \overline{r}\langle y\rangle + [x \neq 0]\ (\nu c)\ (\overline{add}\langle x-1, y, c\rangle \mid c(z).\overline{r}\langle z+1\rangle) \\
\mathbf{P} \quad &= \quad A \mid !mult(x, y, r).[x = 0]\ \overline{r}\langle 0\rangle \\
&\qquad +[x \neq 0]\ (\nu d_1, d_2)\ (\overline{mult}\langle x-1, y, d_1\rangle \mid d_1(res).\overline{add}\langle y, res, d_2\rangle \mid d_2(z).\overline{r}\langle z\rangle) \\
\mathbf{F} \quad &= \quad P \mid !fact(x, r).[x = 0]\ \overline{r}\langle 1\rangle \\
&\qquad +[x \neq 0]\ (\nu d_1, d_2)\ (\overline{fact}\langle x-1, d_1\rangle \mid d_1(res).\overline{mult}\langle x, res, d_2\rangle \mid d_2(z).\overline{r}\langle z\rangle)
\end{aligned}$$

- Recursion parameters $\rightarrow$ $\mathsf{nat}_\star$ type.
- Results of recursive calls $\rightarrow$ nat type.
- Type System:
  - levels to enforce termination,
  - integer kinds to enforce predicativity of recursion.
- Validates $A$, $P$, but not $F$ (type mismatch).

# Type system

- Types: $T ::= \mathsf{nat} \mid \mathsf{nat}_\star \mid \#(\tilde{T}) \mid \#(\tilde{T})^N$
- Rules:

$$\frac{}{\Gamma \vdash_N \mathbf{0}} \qquad \frac{\Gamma \vdash_N P \quad \Gamma \vdash u : \#(\tilde{T}) \quad \Gamma \vdash \tilde{x} : \tilde{T}}{\Gamma \vdash_N u(\tilde{x}).P}$$

$$\frac{\Gamma \vdash u : \#(\tilde{T}) \quad \Gamma \vdash \tilde{e} : \tilde{T}}{\Gamma \vdash_N \overline{u}\langle \tilde{e} \rangle} \qquad \frac{\Gamma \vdash_N P_i \quad (i = 1, 2)}{\Gamma \vdash_N P_1 \mid P_2} \qquad \frac{\Gamma \vdash_N P}{\Gamma \vdash_N (\nu c)\ P}$$

$$\frac{\Gamma \vdash_N P_i \quad (i = 1, 2) \quad \Gamma \vdash e : \circ\mathsf{nat}}{\Gamma \vdash_N [e = 0]P_1 + [e \neq 0]P_2} \qquad \frac{\Gamma \vdash u : \#(\tilde{T})^M \quad \Gamma \vdash \tilde{e} : \tilde{T} \quad M \leq N}{\Gamma \vdash_N \overline{u}\langle \tilde{e} \rangle}$$

$$\frac{\Gamma \vdash u : \#(\tilde{T})^M \quad \Gamma \vdash \tilde{e} : [\tilde{T}]_\star \quad M < N}{\Gamma \vdash_N \overline{u}\langle \tilde{e} \rangle}$$

$$\frac{\begin{array}{ll} \Gamma \vdash_N P & \\ \Gamma \vdash u : \#(\tilde{T})^N & \begin{array}{l} (1)\ \mathsf{out}(\Gamma \vdash_N P) = \emptyset \text{ or;} \\ (2)\ \mathsf{out}(\Gamma \vdash_N P) = \{\overline{b}\langle \tilde{e} \rangle\}\ \Gamma(b) = \Gamma(u),\ (\tilde{e} \triangleleft \tilde{T}) < (\tilde{y} \triangleleft \tilde{T}) \end{array} \\ \Gamma \vdash \tilde{y} : \tilde{T} & \end{array}}{\Gamma \vdash_\infty !u(\tilde{y}).P}$$

# Counter-Example

$$\mathbf{A} \quad = \quad !add(x,y,r).[x=0]\ \overline{r}\langle y\rangle + [x \neq 0]\ (\nu c)\ (\overline{add}\langle x-1, y, c\rangle \ \mid\ c(z).\overline{r}\langle z+1\rangle)$$

$$\begin{aligned}\mathbf{P} \quad = \quad & \mathbf{A} \mid\ !mult(x,y,r).[x=0]\ \overline{r}\langle 0\rangle \\ & +[x \neq 0]\ (\nu d_1, d_2)\ (\overline{mult}\langle x-1, y, d_1\rangle \ \mid\ d_1(res).\overline{add}\langle y, res, d_2\rangle \ \mid\ d_2(z).\overline{r}\langle z\rangle)\end{aligned}$$

$$\begin{aligned}\mathbf{P}' \quad = \quad & \mathbf{A} \mid\ !mult(x,y,r).[x=0]\ \overline{r}\langle 0\rangle \\ & +[x \neq 0]\ (\nu d_2)\ (\overline{mult}\langle x-1, y, d_1\rangle \ \mid\ d_1(res).\overline{add}\langle y, res, d_2\rangle \ \mid\ d_2(z).\overline{r}\langle z\rangle)\end{aligned}$$

- $\mathbf{P}'$ typable.
- Channel $d_1$ is free in $\mathbf{P}'$.
  - $\Rightarrow$ anything can be received, used in further computation.
- Breaks polynomiality
  - arbitrary large values from the environments
- Translating [BellantoniCook94] is not enough.

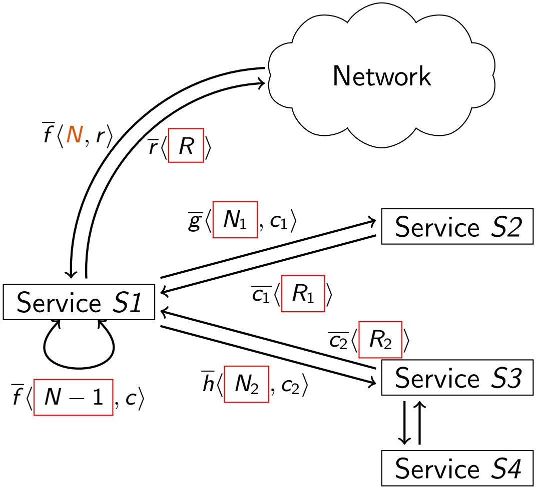
# Counter-Example (II)

External interaction is not needed to bypass type system:

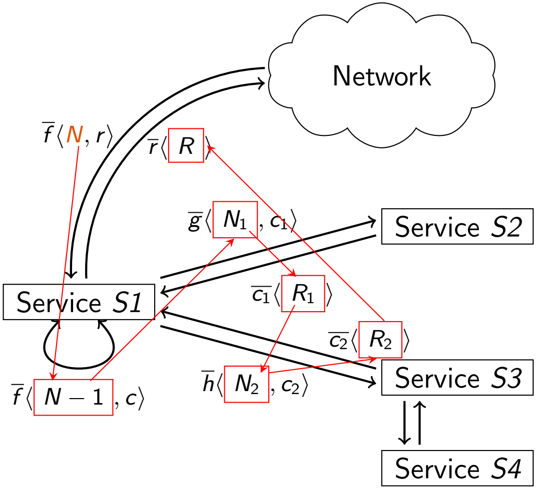
$$\begin{aligned}
\mathit{incr} &= d(z).\overline{d}\langle z+1\rangle \\
CE &= !\mathit{add}(x,y,r).[x=0]\overline{r}\langle y\rangle + [x\neq 0](\nu c)\ (\overline{\mathit{add}}\langle x-1,y,c\rangle \\
&\qquad \mid\ c(z).\overline{r}\langle z+1\rangle\ \mid\ \mathit{incr}) \\
&\mid !\mathit{mult}(x,y,r).[x=0]\overline{r}\langle 0\rangle + [x\neq 0](\nu c1,c2)\ (\overline{\mathit{mult}}\langle x-1,y,c1\rangle \\
&\qquad \mid\ c1(z1).\overline{\mathit{add}}\langle y,z1,c2\rangle\ \mid\ c2(z2).\overline{r}\langle z2\rangle\ \mid\ \mathit{incr}) \\
&\mid !\mathit{fact}(x).[x=0]\overline{r}\langle 1\rangle + [x\neq 0](\nu c)\ (\overline{\mathit{fact}}\langle x-1\rangle \\
&\qquad \mid\ d(z).(\overline{\mathit{mult}}\langle z,x-1,c\rangle\ \mid\ \overline{d}\langle z+1\rangle\ \mid\ \overline{d}\langle 1\rangle))
\end{aligned}$$

- Name $d$ carry information between different calls.
- $z$ contains the "result" of the recursive call to *fact*
  - the process is exponential.
- The type system fails to detect that the content of $d$ should not be trusted.
- → additional constraints are required to guarantee that outside values do not interfer inside computation.
- → hints toward a result for open systems.

# Information Flow

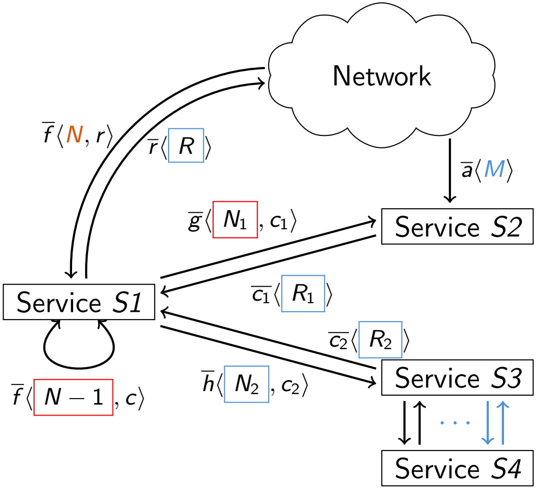


# Information Flow



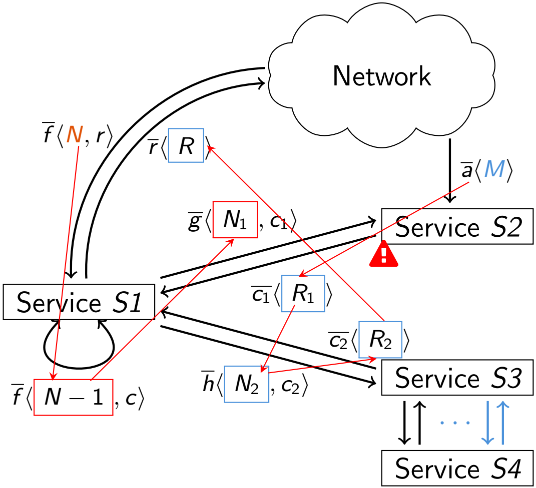

$A \rightarrow B$: value $B$ originates from $A$

# Information Flow




$A \rightarrow B$: value $B$ originates from $A$

# Information Flow




$A \rightarrow B$: value $B$ originates from $A$

# Control on Origin

Solution:

- Origin of integers appearing in crucial prefixes:
  - either free or received on controlled channels.
- Controlled channels are created locally.
  - $\mathbf{P}'$ is rejected because $d_1$ is free.
- Channels passed in calls are controlled.

$Q$ subprocess of $P$. $e$ an occurring in $Q$.
$\mathrm{ok}(e, Q \in P)$, whenever either:

1. $e$ does not contain any variable;
2. $e$ contains variable $x_i$, bound in $!a(\tilde{x}).R \in Q$; or
3. $e$ contains variable $y_i$, bound in $b(\tilde{y}).R \in Q$ with:

   3.1 $b$ is bound by restriction $(\nu b)\ R' \in Q$;
   3.2 $\mathrm{sub}(b, R') = \{b(\tilde{y})\}$; and
   3.3 $\mathrm{obj}(b, R') \subseteq \{\overline{d}\langle \ldots, b, \ldots \rangle\}$

# Results

## Soundness

$\Gamma \vdash_N P$ and $P$ is controlled $\Rightarrow$ $P$ is bound by a polynomial.

## Completeness

$f \in$ POLYTIME $\Rightarrow \exists$ controlled $P$ which computes $f$ and $\Gamma \vdash_N P$.

## Inference

"For a given $P$, is there $\Gamma, N$ s.t. $\Gamma \vdash_N P$ and $P$ is controlled ?" is decidable in polynomial time.

# Inference

- Several analyses stacked on top of each other.
  1. Assign simple types.
     - unification.
     - difficulty: propagation.
  2. Decide linear/replicated channel types.
     - pass on the process and propagation.
  3. Decide levels.
     - generate constraints.
     - collapse strongly connected component.
     - topological sort on the resulting graph.
  4. Decide integer types.
     - integers used in recursion are unsafe.
     - integers received are safe, by default.
  5. Check typing rules.
     - matching of types and control of recursive calls.
  6. Check information flow constraints.
     - origin of integers and name creation.

# Inference: Propagation

- Consider $a(x).\overline{x}\langle 3\rangle \mid \overline{a}\langle c\rangle \mid c(z).\overline{d}\langle 2\rangle$
- Simple typing gives $\{a : \#(\#(\mathrm{nat})),\ x : \#(\mathrm{nat}),\ c : \#(\mathrm{nat}),\ d : \#(\mathrm{nat}),\ z : \mathrm{nat}\}$
- $x$, $c$ and $d$ have same type.
- Suppose we want to add some information on $x$'s type:
  - $x : \#(\mathrm{nat}) \mapsto x : \#(\mathrm{nat})^2$
  - propagate the change to names using $x$ as an object:
  - $a : \#(\#(\mathrm{nat})) \mapsto a : \#(\#(\mathrm{nat})^2)$
  - propagate the change to names appearing as objects of $a$:
  - $c : \#(\mathrm{nat}) \mapsto c : \#(\mathrm{nat})^2$
- we get $\{a : \#(\#(\mathrm{nat})^2),\ x : \#(\mathrm{nat})^2,\ c : \#(\mathrm{nat})^2,\ d : \#(\mathrm{nat}),\ z : \mathrm{nat}\}$
- $x$, $c$ have same type, but not the same as $d$'s.
- $x$ and $c$ "must have the same type".
  - type environment is not enough, vision of process is still needed to update types,
  - implemented as a directed graph between identifiers.

# Conclusion

- Contributions:
  - Causal complexity framework for $\pi$-calculus.
  - Translation of [BellantoniCook94] in $\pi$-calculus.
  - Information flow control.
- Future Works:
  - Inference implementation.
  - Expressiveness improvements.
  - Other complexity classes.
  - Semantic causality.