Modeling Fresh Names in the π-calculus using Abstractions

Marino Miculan

miculan@dimi.uniud.it

University of Udine

(joint work with **Roberto Bruni, Furio Honsell, Marina Lenisa**)

Plan of the talk

- Motivation
- From π-calculus to $\pi\theta$-calculus
- Finitary (bi)coalgebraic semantics
- θ-automata
- Concluding remarks

Plan of the talk

- Motivation
  - dealing with fresh names
  - coalgebraic view of bisimilarity
  - finite state verification
- From π-calculus to $\pi\theta$-calculus
- Finitary (bi)coalgebraic semantics
- θ-automata
- Concluding remarks

Plan of the talk

- Motivation
  - dealing with fresh names
  - coalgebraic view of bisimilarity
  - finite state verification
- From π-calculus to $\pi\theta$-calculus
  - θ-binding
  - θ-bisimilarity
- Finitary (bi)coalgebraic semantics
- θ-automata
- Concluding remarks

Plan of the talk

- Motivation
  - dealing with fresh names
  - coalgebraic view of bisimilarity
  - finite state verification
- From π-calculus to $\pi\theta$-calculus
  - θ-binding
  - θ-bisimilarity
- Finitary (bi)coalgebraic semantics
  - coalgebras over permutation algebras
- θ-automata
- Concluding remarks

Plan of the talk

- Motivation
  - dealing with fresh names
  - coalgebraic view of bisimilarity
  - finite state verification
- From π-calculus to $\pi\theta$-calculus
  - θ-binding
  - θ-bisimilarity
- Finitary (bi)coalgebraic semantics
  - coalgebras over permutation algebras
- θ-automata
  - indexed bisimilarity
- Concluding remarks

Plan of the talk

- Motivation
  - dealing with fresh names
  - coalgebraic view of bisimilarity
  - finite state verification
- From π-calculus to $\pi\theta$-calculus
  - θ-binding
  - θ-bisimilarity
- Finitary (bi)coalgebraic semantics
  - coalgebras over permutation algebras
- θ-automata
  - indexed bisimilarity
- Concluding remarks
  - recap of main results
  - related work & future directions

Motivation

Fresh names

- Names are fundamental in process calculi
  - mobility aspects
  - dynamic reconfigurations of the network
  - private resources and ports
- Handling of fresh names is still a hot topic
  - CCS-like notions, techniques and tools hardly reusable
  - Needs of rigorous foundations and techniques
  - Montanari&Pistore, Gabbay, ...
- Overall aim
  - Clean syntax and operational/abstract semantics
  - Model-theoretically: Final semantics
  - Practical viewpoint: Efficient verification

Our proposal

- *$\pi\theta$-calculus*: a conservative extension of the π-calculus
- A third binding operator θ models (locally-bound) fresh names
- Transitions exploiting fresh names are based on *θ-abstracted processes and labels*
- Finitely branching semantics (w.r.t. fresh names)
- No need of side-conditions on free/bound names in bisimulation
- Amenable to (bi)coalgebraic semantics
- Effective procedure for checking bisimilarity via *θ-automata* (for finitary processes)

From π-calculus to $\pi\theta$-calculus

π-calculus

π-calculus is a process calculus for communicating systems where mobility is modeled through name passing

Two primitive entities:

names $(x, y \in \mathcal{N})$ and *processes* $(P, Q \in \mathcal{P})$

π-calculus

π-calculus is a process calculus for communicating systems where mobility is modeled through name passing

Two primitive entities:

names ($x, y \in \mathcal{N}$) and *processes* ($P, Q \in \mathcal{P}$)

$$P ::= 0 \mid \bar{x}y.P \mid x(y).P \mid \tau.P \mid (\nu^{x)}P \mid Z \mid \mathrm{rec}\, Z.P \mid P_1|P_2 \mid [x = y]P$$

(the bound process variable Z must be guarded in $\mathrm{rec}\, Z.P$)

Processes are taken up-to α-equivalence

π-calculus

π-calculus is a process calculus for communicating systems where mobility is modeled through name passing

Two primitive entities:

names ($x, y \in \mathcal{N}$) and *processes* ($P, Q \in \mathcal{P}$)

$P ::= 0 \mid \bar{x}y.P \mid x(y).P \mid \tau.P \mid (\nu^{x)}P \mid Z \mid \mathrm{rec}\, Z.P \mid P_1|P_2 \mid [x = y]P$

(the bound process variable Z must be guarded in $\mathrm{rec}\, Z.P$)

Processes are taken up-to α-equivalence

There is a plethora of LTS operational semantics: we focus on the original one for early operational semantics

π-calculus

π-calculus is a process calculus for communicating systems where mobility is modeled through name passing

Two primitive entities:

names ($x, y \in \mathcal{N}$) and *processes* ($P, Q \in \mathcal{P}$)

$$P ::= 0 \mid \bar{x}y.P \mid x(y).P \mid \tau.P \mid (\nu^{x)}P \mid Z \mid \mathrm{rec}\, Z.P \mid P_1|P_2 \mid [x = y]P$$

(the bound process variable Z must be guarded in $\mathrm{rec}\, Z.P$)

Processes are taken up-to α-equivalence

There is a plethora of LTS operational semantics: we focus on the original one for early operational semantics

Drawbacks: the LTS is infinitely branching and the abstract semantics requires constraints about bound names

Early Operational Semantics

$$\frac{-}{x(z).P \xrightarrow{xy} P\{y/z\}} \quad (\text{IN})$$

$$\frac{-}{\overline{x}y.P \xrightarrow{\overline{x}y} P} \quad (\text{OUT})$$

$$\frac{-}{\tau.P \xrightarrow{\tau} P} \quad (\text{TAU})$$

$$\frac{P \xrightarrow{\overline{x}y} P'}{(\nu y)P \xrightarrow{\overline{x}(y)} P'}\ y \neq x \quad (\text{OPEN})$$

$$\frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\overline{x}(y)} Q'}{P|Q \xrightarrow{\tau} (\nu y)(P'|Q')}\ y \notin fn(P) \quad (\text{CLOSE}_l)$$

$$\frac{P \xrightarrow{\mu} P'}{[x = x]P \xrightarrow{\mu} P'} \quad (\text{MATCH})$$

$$\frac{P \xrightarrow{\mu} P'}{(\nu y)P \xrightarrow{\mu} (\nu y)P'}\ y \notin n(\mu) \quad (\text{RES})$$

$$\frac{P\{\text{rec}\, Z.P/Z\} \xrightarrow{\mu} P'}{\text{rec}\, Z.P \xrightarrow{\mu} P'} \quad (\text{UNFOLD})$$

$$\frac{P \xrightarrow{\mu} P'}{P|Q \xrightarrow{\mu} P'|Q}\ bn(\mu) \cap fn(Q) = \emptyset \quad (\text{PAR}_l)$$

$$\frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\overline{x}y} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \quad (\text{COM}_l)$$

Early Operational Semantics

$$\frac{-}{x(z).P \xrightarrow{xy} P\{y/z\}} \quad (\text{IN})$$

$$\frac{-}{\overline{x}y.P \xrightarrow{\overline{x}y} P} \quad (\text{OUT})$$

$$\frac{-}{\tau.P \xrightarrow{\tau} P} \quad (\text{TAU})$$

$$\frac{P \xrightarrow{\overline{x}y} P'}{(\nu y)P \xrightarrow{\overline{x}(y)} P'}\ y \neq x \quad (\text{OPEN})$$

$$\frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\overline{x}(y)} Q'}{P|Q \xrightarrow{\tau} (\nu y)(P'|Q')}\ y \notin fn(P) \quad (\text{CLOSE}_l)$$

$$\frac{P \xrightarrow{\mu} P'}{[x = x]P \xrightarrow{\mu} P'} \quad (\text{MATCH})$$

$$\frac{P \xrightarrow{\mu} P'}{(\nu y)P \xrightarrow{\mu} (\nu y)P'}\ y \notin n(\mu) \quad (\text{RES})$$

$$\frac{P\{\text{rec}\, Z.P/Z\} \xrightarrow{\mu} P'}{\text{rec}\, Z.P \xrightarrow{\mu} P'} \quad (\text{UNFOLD})$$

$$\frac{P \xrightarrow{\mu} P'}{P|Q \xrightarrow{\mu} P'|Q}\ bn(\mu) \cap fn(Q) = \emptyset \quad (\text{PAR}_l)$$

$$\frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\overline{x}y} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \quad (\text{COM}_l)$$

Two examples about name extrusion

$$(\nu y).\overline{x}y.P \xrightarrow{\overline{x}(z)} P\{z/y\} \quad z \notin fn(P)$$

$$\frac{x(y).P \xrightarrow{xz} P\{z/y\} \quad (\nu z).\overline{x}z.Q \xrightarrow{\overline{x}(z)} Q}{x(y).P|(\nu z).\overline{x}(z).Q \xrightarrow{\tau} (\nu z)(P\{z/y\}|Q)} \quad z \notin fn(P)$$

Early Bisimilarity

A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is an *early bisimulation* iff, for all processes $P, Q \in \mathcal{P}$, if $P \mathcal{R} Q$ then

for each $P \xrightarrow{\mu} P'$ with $bn(\mu) \cap fn(P, Q) = \emptyset$ then
there exists Q' such that $Q \xrightarrow{\mu} Q'$ and $P' \mathcal{R} Q'$.

Early bisimilarity $\sim$ is the greatest early bisimulation.

Remark: Actions are not taken up-to α-equivalence

Early Bisimilarity

A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is an *early bisimulation* iff, for all processes $P, Q \in \mathcal{P}$, if $P \mathcal{R} Q$ then

for each $P \xrightarrow{\mu} P'$ with $bn(\mu) \cap fn(P, Q) = \emptyset$ then
there exists Q' such that $Q \xrightarrow{\mu} Q'$ and $P' \mathcal{R} Q'$.

Early bisimilarity $\sim$ is the greatest early bisimulation.

Remark: Actions are not taken up-to α-equivalence

The side condition on free variables is **problematic to deal with in a coalgebraic framework!**

$\pi\theta$-calculus

$\pi\theta$-calculus is a conservative extension of π-calculus, where processes can be prefixed by a finite sequence of the new binding operator θ

$\pi\theta$-calculus

$\pi\theta$-calculus is a conservative extension of π-calculus, where processes can be prefixed by a finite sequence of the new binding operator θ

The sets of $\pi\theta$-processes and $\pi\theta$-actions are:

$$\mathcal{P}^\theta \triangleq \{(\theta x_1)\ldots(\theta x_n)P \mid P \in \mathcal{P},\ x_1,\ldots,x_n \in \mathcal{N},\ n \geq 0\}$$

$$\mathcal{L}^\theta \triangleq \{(\theta x_1)\ldots(\theta x_n)\mu \mid \mu \in \mathcal{L}_f \cup \mathcal{L}_b,\ x_1,\ldots,x_n \in \mathcal{N},\ n \geq 0\}$$

(where $\mathcal{L}_f$ is the set of *free actions* and $\mathcal{L}_b$ is the set of *bound actions*)

Terms are taken up-to α-renaming of bound variables.

$\pi\theta$-calculus

$\pi\theta$-calculus is a conservative extension of π-calculus, where processes can be prefixed by a finite sequence of the new binding operator θ

The sets of $\pi\theta$-processes and $\pi\theta$-actions are:

$$\mathcal{P}^\theta \triangleq \{(\theta x_1)\ldots(\theta x_n)P \mid P \in \mathcal{P},\ x_1,\ldots,x_n \in \mathcal{N},\ n \geq 0\}$$

$$\mathcal{L}^\theta \triangleq \{(\theta x_1)\ldots(\theta x_n)\mu \mid \mu \in \mathcal{L}_f \cup \mathcal{L}_b,\ x_1,\ldots,x_n \in \mathcal{N},\ n \geq 0\}$$

(where $\mathcal{L}_f$ is the set of *free actions* and $\mathcal{L}_b$ is the set of *bound actions*)

Terms are taken up-to α-renaming of bound variables.

$(\theta x)P$ represents a **process abstraction** obtained by instantiating P with a fresh name. The name which has to be fresh remains bound in P so that its freshness is guaranteed implicitly.

By abuse of notation, P, Q and μ will range also over $\mathcal{P}^\theta$ and $\mathcal{L}^\theta$, resp.

A few remarks on notation

$(\theta\vec{x})P$ and $(\theta\vec{x})\mu$ stand for the process $(\theta x_1)\ldots(\theta x_n)P$, and for the label $(\theta x_1)\ldots(\theta x_n)\mu$, respectively, where P and μ are θ-free.

A few remarks on notation

$(\theta\vec{x})P$ and $(\theta\vec{x})\mu$ stand for the process $(\theta x_1)\ldots(\theta x_n)P$, and for the label $(\theta x_1)\ldots(\theta x_n)\mu$, respectively, where P and μ are θ-free.

θ binds the occurrences of $x_1 \ldots x_n$ in $(\theta\vec{x})P$ and in $(\theta\vec{x})\mu$.

A few remarks on notation

$(\theta\vec{x})P$ and $(\theta\vec{x})\mu$ stand for the process $(\theta x_1)\ldots(\theta x_n)P$, and for the label $(\theta x_1)\ldots(\theta x_n)\mu$, respectively, where P and μ are θ-free.

θ binds the occurrences of $x_1 \ldots x_n$ in $(\theta\vec{x})P$ and in $(\theta\vec{x})\mu$.

Both processes and labels are taken up-to α-equivalence; hence $x_1, \ldots, x_n$ in $\vec{x}$ can always be assumed to be all distinct (e.g., $(\theta xx)P$ is the same as $(\theta xy)P\{y/x\}$).

A few remarks on notation

$(\theta\vec{x})P$ and $(\theta\vec{x})\mu$ stand for the process $(\theta x_1)\ldots(\theta x_n)P$, and for the label $(\theta x_1)\ldots(\theta x_n)\mu$, respectively, where P and μ are θ-free.

θ binds the occurrences of $x_1\ldots x_n$ in $(\theta\vec{x})P$ and in $(\theta\vec{x})\mu$.

Both processes and labels are taken up-to α-equivalence; hence $x_1,\ldots,x_n$ in $\vec{x}$ can always be assumed to be all distinct (e.g., $(\theta xx)P$ is the same as $(\theta xy)P\{y/x\}$).

(θx_i) in the process $(\theta x_1\ldots x_n)P$ is *vacuous* if $x_i \notin fn(P)$.

A few remarks on notation

$(\theta\vec{x})P$ and $(\theta\vec{x})\mu$ stand for the process $(\theta x_1)\ldots(\theta x_n)P$, and for the label $(\theta x_1)\ldots(\theta x_n)\mu$, respectively, where P and μ are θ-free.

θ binds the occurrences of $x_1\ldots x_n$ in $(\theta\vec{x})P$ and in $(\theta\vec{x})\mu$.

Both processes and labels are taken up-to α-equivalence; hence $x_1,\ldots,x_n$ in $\vec{x}$ can always be assumed to be all distinct (e.g., $(\theta xx)P$ is the same as $(\theta xy)P\{y/x\}$).

(θx_i) in the process $(\theta x_1\ldots x_n)P$ is *vacuous* if $x_i \notin fn(P)$.

For X a finite set of names, we denote by $\mathcal{P}^\theta_X$ and $\mathcal{L}^\theta_X$, the sets of $\pi\theta$-processes and $\pi\theta$-labels whose free names are in X, respectively. The set of *closed* $\pi\theta$-processes is $\mathcal{P}^\theta_\emptyset$.

A few remarks on notation

$(\theta\vec{x})P$ and $(\theta\vec{x})\mu$ stand for the process $(\theta x_1)\ldots(\theta x_n)P$, and for the label $(\theta x_1)\ldots(\theta x_n)\mu$, respectively, where P and μ are θ-free.

θ binds the occurrences of $x_1 \ldots x_n$ in $(\theta\vec{x})P$ and in $(\theta\vec{x})\mu$.

Both processes and labels are taken up-to α-equivalence; hence $x_1, \ldots, x_n$ in $\vec{x}$ can always be assumed to be all distinct (e.g., $(\theta xx)P$ is the same as $(\theta xy)P\{y/x\}$).

(θx_i) in the process $(\theta x_1 \ldots x_n)P$ is *vacuous* if $x_i \notin fn(P)$.

For X a finite set of names, we denote by $\mathcal{P}^\theta_X$ and $\mathcal{L}^\theta_X$, the sets of $\pi\theta$-processes and $\pi\theta$-labels whose free names are in X, respectively. The set of *closed* $\pi\theta$-processes is $\mathcal{P}^\theta_\emptyset$.

The operational semantics is defined by a family of relations $\twoheadrightarrow_X \subseteq \mathcal{P}^\theta_X \times \mathcal{L}^\theta_X \times \mathcal{P}^\theta_X$

Early Operational Semantics

$$\frac{-}{\tau.P \xrightarrow{\tau}\!\!\!\!\rightarrow_X P} \quad (\text{TAU})$$

$$\frac{-}{x(y).P \xrightarrow{xz}\!\!\!\!\rightarrow_X P\{z/y\}}\ z \in X \quad (\text{IN})$$

$$\frac{-}{x(y).P \xrightarrow{x(y)}\!\!\!\!\rightarrow_X (\theta y)P} \quad (\text{IN}^\theta)$$

$$\frac{-}{\overline{x}y.P \xrightarrow{\overline{x}y}\!\!\!\!\rightarrow_X P} \quad (\text{OUT})$$

$$\frac{P \xrightarrow{\mu}\!\!\!\!\rightarrow_X P'}{P|Q \xrightarrow{\mu}\!\!\!\!\rightarrow_X P'|Q}\ \mu \in \mathcal{L}_f \quad (\text{PAR}_l)$$

$$\frac{P \xrightarrow{\mu}\!\!\!\!\rightarrow_X (\theta x)P'}{P|Q \xrightarrow{\mu}\!\!\!\!\rightarrow_X (\theta x)P'|Q} \quad (\text{PAR}_l^\theta)$$

$$\frac{P \xrightarrow{xy}\!\!\!\!\rightarrow_X P' \quad Q \xrightarrow{\overline{x}y}\!\!\!\!\rightarrow_X Q'}{P|Q \xrightarrow{\tau}\!\!\!\!\rightarrow_X P'|Q'} \quad (\text{COM}_l)$$

$$\frac{P \xrightarrow{\overline{x}y}\!\!\!\!\rightarrow_{X \uplus \{y\}} P'}{(\nu y)P \xrightarrow{\overline{x}(y)}\!\!\!\!\rightarrow_X (\theta y)P'} \quad (\text{OPEN})$$

$$\frac{P\{\text{rec}\, Z.P/Z\} \xrightarrow{\mu}\!\!\!\!\rightarrow_X P'}{\text{rec}\, Z.P \xrightarrow{\mu}\!\!\!\!\rightarrow_X P'} \quad (\text{UNFOLD})$$

$$\frac{P \xrightarrow{\mu}\!\!\!\!\rightarrow_X P'}{[x = x]P \xrightarrow{\mu}\!\!\!\!\rightarrow_X P'} \quad (\text{MATCH})$$

$$\frac{P \xrightarrow{\mu}\!\!\!\!\rightarrow_{X \uplus \{y\}} P'}{(\nu y)P \xrightarrow{\mu}\!\!\!\!\rightarrow_X (\nu y)P'}\ y \notin fn(\mu),\ \mu \in \mathcal{L}_f \quad (\text{RES})$$

$$\frac{P \xrightarrow{\mu}\!\!\!\!\rightarrow_{X \uplus \{y\}} (\theta z)P'}{(\nu y)P \xrightarrow{\mu}\!\!\!\!\rightarrow_X (\theta z)(\nu y)P'}\ y \notin fn(\mu) \quad (\text{RES}^\theta)$$

$$\frac{P \xrightarrow{x(y)}\!\!\!\!\rightarrow_X (\theta y)P' \quad Q \xrightarrow{\bar{x}(y)}\!\!\!\!\rightarrow_X (\theta y)Q'}{P|Q \xrightarrow{\tau}\!\!\!\!\rightarrow_X (\nu y)P'|Q'} \quad (\text{CLOSE}_l)$$

$$\frac{P \xrightarrow{\mu}\!\!\!\!\rightarrow_{X \uplus \{x\}} Q}{(\theta x)P \xrightarrow{(\theta x)\mu}\!\!\!\!\rightarrow_X (\theta x)Q} \quad (\text{THETA})$$

A few remarks on the LTS

Rule IN receives a “previously known” name $z \in X$,
while rule IN$^{\theta}$ receives a fresh, θ-bounded name y:

Input transitions differing by the choice of the fresh name are collapsed (by α-rule) in a single transition, and the LTS becomes **finitely branching**.

A few remarks on the LTS

Rule IN receives a "previously known" name $z \in X$,
while rule IN$^{\theta}$ receives a fresh, θ-bounded name y:

Input transitions differing by the choice of the fresh name are collapsed (by α-rule) in a single transition, and the LTS becomes **finitely branching**.

Example:

$(\nu y)\bar{x}y.P$ has just one move: $(\nu y)\bar{x}y.P \overset{(\theta y)\bar{x}y}{\twoheadrightarrow}_X (\theta y)P$

A few remarks on the LTS

Rule IN receives a "previously known" name $z \in X$,
while rule IN$^{\theta}$ receives a fresh, θ-bounded name y:

Input transitions differing by the choice of the fresh name are collapsed (by α-rule) in a single transition, and the LTS becomes **finitely branching**.

Example:

$(\nu y)\bar{x}y.P$ has just one move: $(\nu y)\bar{x}y.P \overset{(\theta y)\bar{x}y}{\twoheadrightarrow}_X (\theta y)P$

As in rule IN$^{\theta}$, also in rule OPEN, the allocation of a fresh name is delegated to the constructor θ.

A few remarks on the LTS

Rule IN receives a "previously known" name $z \in X$,
while rule IN$^\theta$ receives a fresh, θ-bounded name y:

Input transitions differing by the choice of the fresh name are collapsed (by α-rule) in a single transition, and the LTS becomes **finitely branching**.

Example:

$(\nu y)\bar{x}y.P$ has just one move: $(\nu y)\bar{x}y.P \overset{(\theta y)\bar{x}y}{\twoheadrightarrow}_X (\theta y)P$

As in rule IN$^\theta$, also in rule OPEN, the allocation of a fresh name is delegated to the constructor θ.

The rules PAR and RES are duplicated, to deal with θ-bound names in the target process.

A few remarks on the LTS

Rule IN receives a "previously known" name $z \in X$,
while rule IN$^\theta$ receives a fresh, θ-bounded name y:

Input transitions differing by the choice of the fresh name are collapsed (by α-rule) in a single transition, and the LTS becomes **finitely branching**.

Example:

$(\nu y)\bar{x}y.P$ has just one move: $(\nu y)\bar{x}y.P \overset{(\theta y)\bar{x}y}{\twoheadrightarrow}_X (\theta y)P$

As in rule IN$^\theta$, also in rule OPEN, the allocation of a fresh name is delegated to the constructor θ.

The rules PAR and RES are duplicated, to deal with θ-bound names in the target process.

Remark: All side conditions $z \in X$ (IN) and $y \notin fn(\mu)$ (RES and RES$^\theta$) are redundant, because they are always ensured by the type of $\twoheadrightarrow_X$.

Some properties of the LTS

If $(\theta\vec{x})P \overset{(\theta\vec{y})\mu}{\twoheadrightarrow}_X (\theta\vec{z})Q$, then $fn((\theta\vec{y})\mu, (\theta\vec{z})Q) \subseteq fn((\theta\vec{x})P)$.

Hence, if $(\theta\vec{x})P \in \mathcal{P}^{\theta}_{\emptyset}$, then X can be set to $\emptyset$.

Some properties of the LTS

If $(\theta\vec{x})P \xrightarrow{(\theta\vec{y})\mu}\!\!\!\!\!\!\rightarrow_X (\theta\vec{z})Q$, then $fn((\theta\vec{y})\mu, (\theta\vec{z})Q) \subseteq fn((\theta\vec{x})P)$.

Hence, if $(\theta\vec{x})P \in \mathcal{P}^{\theta}_{\emptyset}$, then X can be set to $\emptyset$.

Lemma

For all X finite, for all $(\theta\vec{x})P \in \mathcal{P}^{\theta}_{X}$:

i) $\forall (\theta\vec{x})Q, (\theta\vec{x})\mu$: $\quad (\theta\vec{x})P \xrightarrow{(\theta\vec{x})\mu}\!\!\!\!\!\!\rightarrow_X (\theta\vec{x})Q$ iff $P \xrightarrow{\mu}\!\!\!\!\!\!\rightarrow_{X\cup\{\vec{x}\}} Q$

ii) $\forall (\theta\vec{x}y)Q, (\theta\vec{x})\mu$: $\quad (\theta\vec{x})P \xrightarrow{(\theta\vec{x})\mu}\!\!\!\!\!\!\rightarrow_X (\theta\vec{x}y)Q$ iff $P \xrightarrow{\mu}\!\!\!\!\!\!\rightarrow_{X\cup\{\vec{x}\}} (\theta y)Q$

Some properties of the LTS

If $(\theta\vec{x})P \overset{(\theta\vec{y})\mu}{\twoheadrightarrow}_X (\theta\vec{z})Q$, then $fn((\theta\vec{y})\mu, (\theta\vec{z})Q) \subseteq fn((\theta\vec{x})P)$.

Hence, if $(\theta\vec{x})P \in \mathcal{P}^{\theta}_{\emptyset}$, then X can be set to $\emptyset$.

Lemma

For all X finite, for all $(\theta\vec{x})P \in \mathcal{P}^{\theta}_{X}$:

i) $\forall (\theta\vec{x})Q, (\theta\vec{x})\mu$: $\quad (\theta\vec{x})P \overset{(\theta\vec{x})\mu}{\twoheadrightarrow}_X (\theta\vec{x})Q$ iff $P \overset{\mu}{\twoheadrightarrow}_{X\cup\{\vec{x}\}} Q$

ii) $\forall (\theta\vec{x}y)Q, (\theta\vec{x})\mu$: $\quad (\theta\vec{x})P \overset{(\theta\vec{x})\mu}{\twoheadrightarrow}_X (\theta\vec{x}y)Q$ iff $P \overset{\mu}{\twoheadrightarrow}_{X\cup\{\vec{x}\}} (\theta y)Q$

Remark

Processes $(\nu x)P$ and $(\theta x)P$ behave differently, in general: Rules RES and OPEN do not allow for output actions whose subject is exactly x, while the process $(\theta x)P$ could make an output transition under θ.

Adequacy of the LTS

Proposition

For all X finite, for all $\vec{z}$ not in X, and $P \in \mathcal{P}_{X \cup \{\vec{z}\}}$, $x, y \in \mathcal{N}$:

- $P \xrightarrow{\tau} Q$ iff $(\theta\vec{z})P \overset{(\theta\vec{z})\tau}{\twoheadrightarrow}_X (\theta\vec{z})Q$;
- $P \xrightarrow{\bar{x}y} Q$ iff $(\theta\vec{z})P \overset{(\theta\vec{z})\bar{x}y}{\twoheadrightarrow}_X (\theta\vec{z})Q$;
- $P \xrightarrow{xy} Q$ iff $\Big($if $y \in X \cup \{\vec{z}\}$ then $(\theta\vec{z})P \overset{(\theta\vec{z})xy}{\twoheadrightarrow}_X (\theta\vec{z})Q$ else $(\theta\vec{z})P \overset{(\theta\vec{z})x(y)}{\twoheadrightarrow}_X (\theta\vec{z}y)Q\Big)$;
- $P \xrightarrow{\bar{x}(y)} Q$ iff $(\theta\vec{z})P \overset{(\theta\vec{z})\bar{x}(y)}{\twoheadrightarrow}_X (\theta\vec{z}y)Q$.

The case of θ-free processes

In particular, when $\vec{z}$ is empty:

Corollary

For all X finite, for all $P \in \mathcal{P}_X$, for all $x, y \in \mathcal{N}$:

- $P \xrightarrow{\tau} Q$ iff $P \xrightarrow{\tau}\!\!\!\!\rightarrow_X Q$;
- $P \xrightarrow{\bar{x}y} Q$ iff $P \xrightarrow{\bar{x}y}\!\!\!\!\rightarrow_X Q$;
- $P \xrightarrow{xy} Q$ iff $\Big($if $y \in X$ then $P \xrightarrow{xy}\!\!\!\!\rightarrow_X Q$ else $P \xrightarrow{x(y)}\!\!\!\!\rightarrow_X (\theta y)Q\Big)$;
- $P \xrightarrow{\bar{x}(y)} Q$ iff $P \xrightarrow{\bar{x}(y)}\!\!\!\!\rightarrow_X (\theta y)Q$.

From $\xrightarrow{\mu}\!\!\!\!\rightarrow_X$ to $\xrightarrow{\mu}\!\!\!\!\rightarrow$

The relations $\longrightarrow\!\!\!\!\rightarrow_X$ can be seen as a family of coherent approximations of the usual early operational semantics:

For μ action of the $\pi\theta$-calculus, we define $\xrightarrow{\mu}\!\!\!\!\rightarrow \triangleq \bigcup_X \xrightarrow{\mu}\!\!\!\!\rightarrow_X$

Note that

- each $\longrightarrow\!\!\!\!\rightarrow_X$ is *truly* finitely branching, because the names which can be chosen in the rule IN must belong to X, which is finite.
- if we drop the X parameter and consider all the transition systems simultaneously, the operational semantics would then be finitely branching *only* w.r.t. fresh names.

Early θ-bisimilarity

Let X be a finite set of names. A symmetric relation $\mathcal{R}_X \subseteq \mathcal{P}_X^\theta \times \mathcal{P}_X^\theta$ is an *early θ-bisimulation at stage X* iff, for all $P, Q \in \mathcal{P}_X^\theta$ processes, $P \; \mathcal{R}_X \; Q$ implies:

> if $P \xrightarrow{\mu}\!\!\!\!\rightarrow_X P'$, then there exists Q' such that $Q \xrightarrow{\mu}\!\!\!\!\rightarrow_X Q'$ and $P' \; \mathcal{R}_X \; Q'$.

The *early θ-bisimilarity at stage X*, $\approx_X$, is the greatest early θ-bisimulation at stage X.

The *early θ-bisimilarity* $\approx$ is defined as $\approx \triangleq \bigcup_X \approx_X$.

Remark: No side conditions in the definition

A trade-off

We could disregard X completely. Anyway, for θ-closed processes, any reference to names disappears altogether.

The price to pay is that each time a fresh name is needed, an extra (possibly vacuous) θ is generated, and the set of processes reached during the evolution of a finitary process is finite, only *up-to vacuous θ's*.

Next we show examples of

- evolutions with vacuous θ's that one would like to eliminate
- situation where elimination of vacuous θ's is not safe, (even if we restrict to processes whose occurrences of names are all *active*, in the sense that they can be exposed in some action)

On vacuous θ's (1)

Example

Let $X = \{x\}$ and $P = \mathrm{rec}\, Z.(\nu y).\bar{x}y.Z \in \mathcal{P}_X$

In the π-calculus,

$$P \xrightarrow{\bar{x}(y)} P \xrightarrow{\bar{x}(y)} \ldots$$

In the $\pi\theta$-calculus,

$$P \overset{\bar{x}(y_0)}{\twoheadrightarrow}_X (\theta y_0)P \overset{(\theta y_0)\bar{x}(y_1)}{\twoheadrightarrow}_X (\theta y_0 y_1)P \overset{(\theta y_0 y_1)\bar{x}(y_2)}{\twoheadrightarrow}_X (\theta y_0 y_1 y_2)P \ldots$$

The states reached by P after a finite number of transition steps differ by a finite number of vacuous θ's.

On vacuous θ's (2)

Example

Let $P = (\nu x)\bar{w}x.(\nu y)\bar{x}y.\bar{u}x.0$ and $Q = (\nu x)\bar{w}x.(\nu y)\bar{x}y.\bar{u}y.0$.

$P \not\sim Q$, because the last action of P consists in communicating the first extruded name, while Q communicates the second extruded name.

If we allow to eliminate vacuous θ's from $\pi\theta$-processes, then P and Q are erroneously equated:

- $P \twoheadrightarrow^* (\theta xy)\bar{u}x.0$ (in two steps), where θy is vacuous
- $Q \twoheadrightarrow^* (\theta xy)\bar{u}y.0$ (in two steps), where θx is vacuous
- but the pair of processes $(\theta x)\bar{u}x$ and $(\theta y)\bar{u}y$, are trivially α-equivalent and thus bisimilar!

Relating abstract semantics

As a main correspondence result, early θ-bisimilarity is a conservative extension of usual bisimilarity:

Theorem: Let $P, Q \in \mathcal{P}$.
Then, $P \sim Q$ iff $P \approx Q$.

The proof is by coinduction and it exploits the following lemma.

Lemma: Let $(\theta\vec{x})P, (\theta\vec{x})Q \in \mathcal{P}^{\theta}_{X}$.
Then, $(\theta\vec{x})P \approx_X (\theta\vec{x})Q$ iff $P \approx_{X \cup \{\vec{x}\}} Q$.

Back to θ-automata

Further motivations

We can further capitalize on $\pi\theta$-calculus w.r.t. two issues:

- There might be names which are not active like in

$$(\nu x.\overline{x}y.0) \mid \overline{w}z.0 \xrightarrow{\overline{w}z} (\nu x.\overline{x}y.0) \mid 0$$

  Can we get rid of them in the semantics?

- Can we abstract away from some vacuous θ's?

Further motivations

We can further capitalize on $\pi\theta$-calculus w.r.t. two issues:

- There might be names which are not active like in

$$(\nu x.\overline{x}y.0) \mid \overline{w}z.0 \xrightarrow{\overline{w}z} (\nu x.\overline{x}y.0) \mid 0$$

  Can we get rid of them in the semantics?

  YES: Final semantics!

- Can we abstract away from some vacuous θ's?

Further motivations

We can further capitalize on $\pi\theta$-calculus w.r.t. two issues:

- There might be names which are not active like in

$$(\nu x.\overline{x}y.0) \mid \overline{w}z.0 \xrightarrow{\overline{w}z} (\nu x.\overline{x}y.0) \mid 0$$

  Can we get rid of them in the semantics?

  YES: Final semantics!

- Can we abstract away from some vacuous θ's?

  YES: θ-automata!

Further motivations

We can further capitalize on $\pi\theta$-calculus w.r.t. two issues:

- There might be names which are not active like in

$$(\nu x.\overline{x}y.0) \mid \overline{w}z.0 \xrightarrow{\overline{w}z} (\nu x.\overline{x}y.0) \mid 0$$

  Can we get rid of them in the semantics?
  YES: Final semantics!

- Can we abstract away from some vacuous θ's?
  YES: θ-automata!

This is what the rest of the story is about...

Further motivations

We can further capitalize on $\pi\theta$-calculus w.r.t. two issues:

- There might be names which are not active like in

$$(\nu x.\overline{x}y.0) \mid \overline{w}z.0 \xrightarrow{\overline{w}z} (\nu x.\overline{x}y.0) \mid 0$$

  Can we get rid of them in the semantics?
  YES: Final semantics!

- Can we abstract away from some vacuous θ's?
  YES: θ-automata!

This is what the rest of the story is about...
... and it explains the reason why the story is told at CMCS

Finitary (bi)coalgebraic semantics

Final semantics

The gist of Final Semantics is to express the interpretation function as a co-algebra morphism from the dynamical system, given by the transition relation, to the set of behaviours. Behavioural equivalence is induced by this unique morphism.

Final semantics

The gist of Final Semantics is to express the interpretation function as a co-algebra morphism from the dynamical system, given by the transition relation, to the set of behaviours. Behavioural equivalence is induced by this unique morphism.

In this view dynamical systems come equipped with algebraic operations, arising from the syntax, but there can be more. One would like to have that behavioural equivalence be a congruence w.r.t. to such operations.

Final semantics

The gist of Final Semantics is to express the interpretation function as a co-algebra morphism from the dynamical system, given by the transition relation, to the set of behaviours. Behavioural equivalence is induced by this unique morphism.

In this view dynamical systems come equipped with algebraic operations, arising from the syntax, but there can be more. One would like to have that behavioural equivalence be a congruence w.r.t. to such operations.

It is here that structured co-algebras and bialgebras, i.e. objects which have both an algebra and a coalgebra structure, come into play.

Bialgebras

Algebras offer a uniform account of Syntax.

- A signature yields a functor $\Sigma X = \oplus_{f \in \Sigma_k} X^k$.
- Terms over a set of variables X are the initial $Id + \Sigma$-algebra, i.e. the initial such that: $TX = X + \Sigma(TX)$.

Bialgebras

Algebras offer a uniform account of Syntax.

- A signature yields a functor $\Sigma X = \oplus_{f \in \Sigma_k} X^k$.
- Terms over a set of variables X are the initial $Id + \Sigma$-algebra, i.e. the initial such that: $TX = X + \Sigma(TX)$.

Coalgebras offer a uniform account of Dynamical systems.

- The pattern of the transition system yields a functor $BX = \mathcal{P}(A \times X)$.
- Behaviours of objects labeled by a set of names X are the final $Id \times B$-coalgebra, i.e. the final such that $MX = X \times B(MX)$.

Formally

$G, F : \mathcal{C} \to \mathcal{C}$

- G-algebra: $(X, \beta_X), \ \beta_X : GX \to X$
- F-coalgebra: $(X, \alpha_X), \ \alpha_X : X \to FX$
- $\langle G, F \rangle$-bialgebra: (X, β_X, α_X)

Formally

$G, F : \mathcal{C} \to \mathcal{C}$

- G-algebra: (X, β_X), $\beta_X : GX \to X$
- F-coalgebra: (X, α_X), $\alpha_X : X \to FX$
- $\langle G, F \rangle$-bialgebra: (X, β_X, α_X)

$$\begin{tikzcd}
GX \arrow[r, "\beta_X"] \arrow[d, "Gf"'] \arrow[dr, phantom, "(1)"] & X \arrow[r, "\alpha_X"] \arrow[d, "f"] \arrow[dr, phantom, "(2)"] & FX \arrow[d, "Ff"] \\
GY \arrow[r, "\beta_Y"'] & Y \arrow[r, "\alpha_Y"'] & FY
\end{tikzcd}$$

Formally

$G, F : \mathcal{C} \to \mathcal{C}$

- G-algebra: (X, β_X), $\beta_X : GX \to X$
- F-coalgebra: (X, α_X), $\alpha_X : X \to FX$
- $\langle G, F \rangle$-bialgebra: (X, β_X, α_X)

$$\begin{tikzcd}
GX \arrow[r, "\beta_X"] \arrow[d, "Gf"'] \arrow[dr, phantom, "(1)"] & X \arrow[r, "\alpha_X"] \arrow[d, "f"] \arrow[dr, phantom, "(2)"] & FX \arrow[d, "Ff"] \\
GY \arrow[r, "\beta_Y"'] & Y \arrow[r, "\alpha_Y"'] & FY
\end{tikzcd}$$

$f : X \to Y$ is

- a G-algebra homomorphism from (X, β_X) to (Y, β_Y) if it makes diagram (1) commute.
- an F-coalgebra homomorphism from (X, α_X) to (Y, α_Y) if it makes diagram (2) commute.
- a $\langle G, F \rangle$-bialgebra homomorphism from (X, β_X, α_X) to (Y, β_Y, α_Y) if it makes both diagrams (1) and (2) commute.

Bialgebraic approaches

Structured (Co)-algebras [Corradini,Heckel,Montanari]:
define coalgebras over algebras, or viceversa.

E.g. show that the behavioural functor can be *lifted* to the category of algebras for the given signature and show that α_X is an algebra morphism w.r.t. to that lifting. The final structured coalgebra is the lifting of the final coalgebra.

Bialgebraic approaches

Structured (Co)-algebras [Corradini,Heckel,Montanari]:
define coalgebras over algebras, or viceversa.

E.g. show that the behavioural functor can be *lifted* to the category of algebras for the given signature and show that α_X is an algebra morphism w.r.t. to that lifting. The final structured coalgebra is the lifting of the final coalgebra.

λ-bialgebras [Plotkin,Turi]:
define a category of special bialgebras, i.e. λ-bialgebras, and show that the given coalgebra is, in fact a λ-bialgebra.

A λ-bialgebra, for $\lambda : GF \dot{\to} FG$, is a $\langle G, F\rangle$-bialgebra (X, β_X, α_X) s.t.:

$$\begin{tikzcd}
 & GFX \arrow[r, "\lambda_X"] & FGX \arrow[dr, "F(\beta_X)"] & \\
GX \arrow[ur, "G(\alpha_X)"] \arrow[r, "\beta_X"'] & X \arrow[r, "\alpha_X"'] & & FX
\end{tikzcd}$$

Permutation algebras

We work in a category of stuctured coalgebras. More precisely, we consider coalgebras over the category Alg_π of *permutation algebras*

Permutation algebras

We work in a category of stuctured coalgebras. More precisely, we consider coalgebras over the category Alg_π of *permutation algebras*

Notation:

Let us consider permutations π over natural numbers $\{1, 2, 3 \ldots\}$.
The *kernel* of π is the set $\ker(\pi) \triangleq \{i \mid \pi(i) \neq i\}$.
$perm^{fk} \triangleq \{\pi \mid \ker(\pi)$ finite$\}$ is the countable group of *finite-kernel permutations*.

Permutation algebras

We work in a category of stuctured coalgebras. More precisely, we consider coalgebras over the category Alg_π of *permutation algebras*

Notation:

Let us consider permutations π over natural numbers $\{1, 2, 3 \ldots\}$.
The *kernel* of π is the set $\ker(\pi) \triangleq \{i \mid \pi(i) \neq i\}$.
$perm^{fk} \triangleq \{\pi \mid \ker(\pi)$ finite$\}$ is the countable group of *finite-kernel permutations*.

Signature:

Σ_π consists of finite-kernel permutations, together with the axioms schemata $id(x) = x$ and $\pi_1(\pi_2(x)) = (\pi_1\pi_2)(x)$.

Permutation algebras

We work in a category of stuctured coalgebras. More precisely, we consider coalgebras over the category Alg_π of *permutation algebras*

Notation:

Let us consider permutations π over natural numbers $\{1, 2, 3 \ldots\}$.
The *kernel* of π is the set $\ker(\pi) \triangleq \{i \mid \pi(i) \neq i\}$.
$perm^{fk} \triangleq \{\pi \mid \ker(\pi)$ finite$\}$ is the countable group of *finite-kernel permutations*.

Signature:

Σ_π consists of finite-kernel permutations, together with the axioms schemata $id(x) = x$ and $\pi_1(\pi_2(x)) = (\pi_1\pi_2)(x)$.

A *permutation algebra* $\mathcal{A} = (A, \{\widehat{\pi}_A\})$ is an algebra for Σ_π.
A *permutation morphism* $\sigma : \mathcal{A} \to \mathcal{B}$ is an algebra homomorphism.
(i.e., a function $\sigma : A \to B$ such that $\sigma(\widehat{\pi}_A(x)) = \widehat{\pi}_B(\sigma(x))$)
Alg_π is the category of permutation algebras and their morphisms.

The algebraic structure of $\pi\theta$

The set $\mathcal{P}_{\emptyset}^{\theta}$ of closed $\pi\theta$-calculus processes can be endowed with a structure of a permutation algebra.

The algebraic structure of $\pi\theta$

The set $\mathcal{P}_{\emptyset}^{\theta}$ of closed $\pi\theta$-calculus processes can be endowed with a structure of a permutation algebra.

The *name reindexing* operator $\rho_\pi : \mathcal{P}_{\emptyset}^{\theta} \to \mathcal{P}_{\emptyset}^{\theta}$ is defined as $\rho_\pi((\theta\vec{x})P) \triangleq (\theta x_1 \ldots x_{\max_n \pi})P[\pi]$, where:

- $\max_n \pi \triangleq \max\{\pi(i) \mid i = 1, \ldots, n\}$ (thus $\max_n \pi \geq n$).
- $P[\pi]$ denotes the process obtained by the application of π to the free names occurring in P.

The algebraic structure of $\pi\theta$

The set $\mathcal{P}_{\emptyset}^{\theta}$ of closed $\pi\theta$-calculus processes can be endowed with a structure of a permutation algebra.

The *name reindexing* operator $\rho_\pi : \mathcal{P}_{\emptyset}^{\theta} \to \mathcal{P}_{\emptyset}^{\theta}$ is defined as $\rho_\pi((\theta\vec{x})P) \triangleq (\theta x_1 \ldots x_{\max_n \pi})P[\pi]$, where:

- $\max_n \pi \triangleq \max\{\pi(i) \mid i = 1, \ldots, n\}$ (thus $\max_n \pi \geq n$).
- $P[\pi]$ denotes the process obtained by the application of π to the free names occurring in P.

We let ρ be the countable set of unary operators $\rho \triangleq \{\rho_\pi \mid \pi \in perm^{fk}\}$.

The algebraic structure of $\pi\theta$

The set $\mathcal{P}_{\emptyset}^{\theta}$ of closed $\pi\theta$-calculus processes can be endowed with a structure of a permutation algebra.

The *name reindexing* operator $\rho_\pi : \mathcal{P}_{\emptyset}^{\theta} \to \mathcal{P}_{\emptyset}^{\theta}$ is defined as $\rho_\pi((\theta\vec{x})P) \triangleq (\theta x_1 \ldots x_{\max_n \pi})P[\pi]$, where:

- $\max_n \pi \triangleq \max\{\pi(i) \mid i = 1, \ldots, n\}$ (thus $\max_n \pi \geq n$).
- $P[\pi]$ denotes the process obtained by the application of π to the free names occurring in P.

We let ρ be the countable set of unary operators $\rho \triangleq \{\rho_\pi \mid \pi \in perm^{fk}\}$.

Proposition:
The pair $(\mathcal{P}_{\emptyset}^{\theta}, \rho)$ is a permutation algebra.

The semantic functor

The functor $T : Alg_\pi \to Alg_\pi$ is the canonical extension to arrows of:

$$T(A, \rho^A) \triangleq \left(\wp_{\mathrm{f}}(\mathcal{L}^\theta \times A), \rho^{\wp_{\mathrm{f}}(\mathcal{L}^\theta \times A)} = \{\rho_\pi^{\wp_{\mathrm{f}}(\mathcal{L}^\theta \times A)} \mid \pi \in perm^{fk}\}\right)$$

where, for any $u \in \wp_{\mathrm{f}}(\mathcal{L}^\theta \times A)$, $\pi \in perm^{fk}$,

$$\rho_\pi^{\wp_{\mathrm{f}}(\mathcal{L}^\theta \times A)}(u) \;=\; \{((\theta x_1 \ldots x_{\max_n \pi})\mu[\pi], \rho_\pi a) \mid ((\theta x_1 \ldots x_n)\mu, a) \in u\}$$

The semantic functor

The functor $T : Alg_\pi \to Alg_\pi$ is the canonical extension to arrows of:

$$T(A, \rho^A) \triangleq \left(\wp_{\mathrm{f}}(\mathcal{L}^\theta \times A), \rho^{\wp_{\mathrm{f}}(\mathcal{L}^\theta \times A)} = \{\rho_\pi^{\wp_{\mathrm{f}}(\mathcal{L}^\theta \times A)} \mid \pi \in perm^{fk}\}\right)$$

where, for any $u \in \wp_{\mathrm{f}}(\mathcal{L}^\theta \times A)$, $\pi \in perm^{fk}$,

$$\rho_\pi^{\wp_{\mathrm{f}}(\mathcal{L}^\theta \times A)}(u) \;=\; \{((\theta x_1 \ldots x_{\max_n \pi})\mu[\pi], \rho_\pi a) \mid ((\theta x_1 \ldots x_n)\mu, a) \in u\}$$

T is the lifting of the corresponding polynomial functor $T_s : \mathcal{S}et \to \mathcal{S}et$,

hence we lift the adjunction $\mathcal{S}et \underset{V}{\overset{F}{\underset{\longleftarrow}{\overset{\longrightarrow}{\perp}}}} Alg_\pi$ to $T_s\text{-Coalg} \underset{V_T}{\overset{F_T}{\underset{\longleftarrow}{\overset{\longrightarrow}{\perp}}}} T\text{-Coalg}$.

The semantic functor

The functor $T : Alg_\pi \to Alg_\pi$ is the canonical extension to arrows of:

$$T(A, \rho^A) \triangleq \left(\wp_{\mathrm{f}}(\mathcal{L}^\theta \times A), \rho^{\wp_{\mathrm{f}}(\mathcal{L}^\theta \times A)} = \{\rho_\pi^{\wp_{\mathrm{f}}(\mathcal{L}^\theta \times A)} \mid \pi \in perm^{fk}\}\right)$$

where, for any $u \in \wp_{\mathrm{f}}(\mathcal{L}^\theta \times A)$, $\pi \in perm^{fk}$,

$$\rho_\pi^{\wp_{\mathrm{f}}(\mathcal{L}^\theta \times A)}(u) \;=\; \{((\theta x_1 \ldots x_{\max_n \pi})\mu[\pi], \rho_\pi a) \mid ((\theta x_1 \ldots x_n)\mu, a) \in u\}$$

T is the lifting of the corresponding polynomial functor $T_s : \mathcal{S}et \to \mathcal{S}et$,

hence we lift the adjunction $\begin{tikzcd} \mathcal{S}et \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & Alg_\pi \arrow[l, bend left, "V"] \end{tikzcd}$ to $\begin{tikzcd} T_s\text{-Coalg} \arrow[r, bend left, "F_T"] \arrow[r, phantom, "\perp"] & T\text{-Coalg} \arrow[l, bend left, "V_T"] \end{tikzcd}$.

Proposition

The functor $T : Alg_\pi \to Alg_\pi$ has a final coalgebra $\Omega = (\Omega, \rho^\Omega, \alpha_\Omega)$.
Moreover, $V_T(\Omega)$ is the final T_s-coalgebra.

T-bisimulation

The equivalence on a T-coalgebra (A, α) induced by the unique arrow $\mathcal{M} : (A, \alpha) \to (\Omega, \alpha_\Omega)$ is the counterpart (in the final semantics approach) of the ordinary notion of bisimulation.

T-bisimulation

The equivalence on a T-coalgebra (A, α) induced by the unique arrow $\mathcal{M} : (A, \alpha) \to (\Omega, \alpha_\Omega)$ is the counterpart (in the final semantics approach) of the ordinary notion of bisimulation.

A relation $\mathcal{R} \subseteq A \times A$ is a *T-bisimulation* if $\exists\, \gamma : \mathcal{R} \to T(\mathcal{R})$ s.t. $\pi_1, \pi_2 : \mathcal{R} \to A$ can be lifted to T-coalgebra morphisms $(\mathcal{R}, \gamma) \to (A, \alpha)$

T-bisimulation

The equivalence on a T-coalgebra (A, α) induced by the unique arrow $\mathcal{M} : (A, \alpha) \to (\Omega, \alpha_\Omega)$ is the counterpart (in the final semantics approach) of the ordinary notion of bisimulation.

A relation $\mathcal{R} \subseteq A \times A$ is a *T-bisimulation* if $\exists\, \gamma : \mathcal{R} \to T(\mathcal{R})$ s.t. $\pi_1, \pi_2 : \mathcal{R} \to A$ can be lifted to T-coalgebra morphisms $(\mathcal{R}, \gamma) \to (A, \alpha)$

Proposition

Let $\alpha : (\mathcal{P}^\theta_\emptyset, \rho) \to T(\mathcal{P}^\theta_\emptyset, \rho)$ be defined as $\alpha(P) \triangleq \{(\mu, Q) \mid P \xrightarrow{\mu}\!\!\!\!\rightarrow_\emptyset Q\}$.
Then, $\mathcal{C}_{\pi\theta} \triangleq (\mathcal{P}^\theta_\emptyset, \rho, \alpha)$ is a T-coalgebra.

T-bisimulation

The equivalence on a T-coalgebra (A, α) induced by the unique arrow $\mathcal{M} : (A, \alpha) \to (\Omega, \alpha_\Omega)$ is the counterpart (in the final semantics approach) of the ordinary notion of bisimulation.

A relation $\mathcal{R} \subseteq A \times A$ is a *T-bisimulation* if $\exists\, \gamma : \mathcal{R} \to T(\mathcal{R})$ s.t. $\pi_1, \pi_2 : \mathcal{R} \to A$ can be lifted to T-coalgebra morphisms $(\mathcal{R}, \gamma) \to (A, \alpha)$

Proposition

Let $\alpha : (\mathcal{P}^\theta_\emptyset, \rho) \to T(\mathcal{P}^\theta_\emptyset, \rho)$ be defined as $\alpha(P) \triangleq \{(\mu, Q) \mid P \xrightarrow{\mu}\!\!\!\!\rightarrow_\emptyset Q\}$. Then, $\mathcal{C}_{\pi\theta} \triangleq (\mathcal{P}^\theta_\emptyset, \rho, \alpha)$ is a T-coalgebra.

Theorem

Let $(\theta\vec{x})P, (\theta\vec{x})Q \in \mathcal{P}^\theta_\emptyset$. Then $P \sim Q$ iff $\exists\mathcal{R}$, a T-bisimulation on the coalgebra $\mathcal{C}_{\pi\theta}$, such that $(\theta\vec{x})P \;\mathcal{R}\; (\theta\vec{x})Q$.

T-bisimulation

The equivalence on a T-coalgebra (A, α) induced by the unique arrow $\mathcal{M} : (A, \alpha) \to (\Omega, \alpha_\Omega)$ is the counterpart (in the final semantics approach) of the ordinary notion of bisimulation.

A relation $\mathcal{R} \subseteq A \times A$ is a *T-bisimulation* if $\exists\, \gamma : \mathcal{R} \to T(\mathcal{R})$ s.t. $\pi_1, \pi_2 : \mathcal{R} \to A$ can be lifted to T-coalgebra morphisms $(\mathcal{R}, \gamma) \to (A, \alpha)$

Proposition

Let $\alpha : (\mathcal{P}^\theta_\emptyset, \rho) \to T(\mathcal{P}^\theta_\emptyset, \rho)$ be defined as $\alpha(P) \triangleq \{(\mu, Q) \mid P \xrightarrow{\mu}\!\!\!\!\rightarrow_\emptyset Q\}$. Then, $\mathcal{C}_{\pi\theta} \triangleq (\mathcal{P}^\theta_\emptyset, \rho, \alpha)$ is a T-coalgebra.

Theorem

Let $(\theta\vec{x})P, (\theta\vec{x})Q \in \mathcal{P}^\theta_\emptyset$. Then $P \sim Q$ iff $\exists\mathcal{R}$, a T-bisimulation on the coalgebra $\mathcal{C}_{\pi\theta}$, such that $(\theta\vec{x})P \;\mathcal{R}\; (\theta\vec{x})Q$.

Proposition

The equivalence induced by the unique morphism $\mathcal{M} : \mathcal{C}_{\pi\theta} \longrightarrow \Omega$ coincides with the union of all T-bisimulations on the T-coalgebra $\mathcal{C}_{\pi\theta}$.

Some considerations

All the constructions work because we have done away with the conditions in the definition of bisimulation.

Can we consider other algebraic structures beside ρ's?

Some considerations

All the constructions work because we have done away with the conditions in the definition of bisimulation.

Can we consider other algebraic structures beside ρ's?

The key point is that α has to respect the algebraic structures.
This holds for ν and $|$ but it does not hold in the case of input prefix!

Some considerations

All the constructions work because we have done away with the conditions in the definition of bisimulation.

Can we consider other algebraic structures beside ρ's?

The key point is that α has to respect the algebraic structures.
This holds for ν and $|$ but it does not hold in the case of input prefix!

Counterexample: let $\iota_{zx}(.)$ defined by:

$$\iota_{zx}((\theta\vec{y})P) \triangleq (\theta\vec{y}z)z(x).P$$

Then, for $P = (\theta yw)\bar{y}w$ we have $\alpha(\iota_{zx}(P)) \neq T(\iota_{zx})(\alpha(P))$, whatever is the effect of ι_{zx} on labels.

Some considerations

[Montanari&Pistore] shows that the support of the interpretation of a π-calculus process in the final model amounts exactly to the active names of the process.

We mention a similar result for $\pi\theta$-calculus:

Some considerations

[Montanari&Pistore] shows that the support of the interpretation of a π-calculus process in the final model amounts exactly to the active names of the process.

We mention a similar result for $\pi\theta$-calculus:

Proposition

For $(\theta\vec{x})P \in \mathcal{P}^{\theta}_{\emptyset}$ and $n = |\vec{x}|$, the family

$$\{\rho^{\Omega}_{\pi}(\mathcal{M}((\theta\vec{x})P)) \mid \pi \in perm^{fk}, \pi(\{1\ldots n\}) \subseteq \{1\ldots n\}\}$$

has at most $n!/k!$ distinct elements

(where k is the number of active names of P)

Some considerations

[Montanari&Pistore] shows that the support of the interpretation of a π-calculus process in the final model amounts exactly to the active names of the process.

We mention a similar result for $\pi\theta$-calculus:

Proposition

For $(\theta\vec{x})P \in \mathcal{P}^{\theta}_{\emptyset}$ and $n = |\vec{x}|$, the family

$$\{\rho^{\Omega}_{\pi}(\mathcal{M}((\theta\vec{x})P)) \mid \pi \in perm^{fk}, \pi(\{1\ldots n\}) \subseteq \{1\ldots n\}\}$$

has at most $n!/k!$ distinct elements

(where k is the number of active names of P)

Explanation

The action of ρ^{Ω}_{π} commutes with the final semantics.

Swapping non-active names in the process does not change its bisimilarity class.

Therefore, the number of distinct elements is bounded by the number of different permutations of n objects, k of which are equal.

θ-automata

Why θ-automata

The last part of the talk concerns the quest for an effective procedure for testing bisimilarity.

Our approach is inspired by *HD-automata* of [Montanari&Pistore]

Here, the crucial problem is to get rid of vacuous θ's.

Why θ-automata

The last part of the talk concerns the quest for an effective procedure for testing bisimilarity.

Our approach is inspired by *HD-automata* of [Montanari&Pistore]

Here, the crucial problem is to get rid of vacuous θ's.

θ-automaton are designed for representing in a *finite* way the evolution of *finitary $\pi\theta$-processes*.

These are processes whose descendants have a bounded number of possible parallel actions.

Finitary processes

The *degree of parallelism* $\deg(P)$ of a $\pi\theta$-process P is defined as follows (for μ a generic action prefix):

$$\deg(0) = 0 \qquad \deg(Z) = 0$$

$$\deg(\nu^x)P) = \deg(P) \qquad \deg((\theta x)P) = \deg(P)$$

$$\deg([x = y]P) = \deg(P) \qquad \deg(\ \mathrm{rec}\, Z.P) = \deg(P)$$

$$\deg(\mu.P) = 1 \qquad \deg(P \mid Q) = \deg(P) + \deg(Q)$$

Finitary processes

The *degree of parallelism* $\deg(P)$ of a $\pi\theta$-process P is defined as follows (for μ a generic action prefix):

$$\deg(0) = 0 \qquad \deg(Z) = 0$$

$$\deg(\nu x)P) = \deg(P) \qquad \deg((\theta x)P) = \deg(P)$$

$$\deg([x = y]P) = \deg(P) \qquad \deg(\text{rec}\, Z.P) = \deg(P)$$

$$\deg(\mu.P) = 1 \qquad \deg(P \mid Q) = \deg(P) + \deg(Q)$$

The set of *descendants* of $P \in \mathcal{P}_X^\theta$ is $des(P) \triangleq \{Q \mid P \twoheadrightarrow_X^* Q\}$.

Finitary processes

The *degree of parallelism* $\deg(P)$ of a $\pi\theta$-process P is defined as follows (for μ a generic action prefix):

$$\deg(0) = 0 \qquad \deg(Z) = 0$$
$$\deg(\nu^x)P) = \deg(P) \qquad \deg((\theta x)P) = \deg(P)$$
$$\deg([x = y]P) = \deg(P) \qquad \deg(\text{ rec } Z.P) = \deg(P)$$
$$\deg(\mu.P) = 1 \qquad \deg(P \mid Q) = \deg(P) + \deg(Q)$$

The set of *descendants* of $P \in \mathcal{P}_X^\theta$ is $des(P) \triangleq \{Q \mid P \twoheadrightarrow_X^* Q\}$.

Let $\overline{\deg}(P) \triangleq \sup\{\deg(Q) \mid Q \in de^s(P)\}$.
A process $P \in \mathcal{P}_X^\theta$ is *finitary* if $\overline{\deg}(P) < \infty$.

Structural congruence

The *structural congruence* $\equiv$ on π-calculus processes is the smallest congruence that satisfies the following:

(alpha) $P \equiv Q$ if P and Q are α-equivalent

(par) $P \mid 0 \equiv P \quad P \mid Q \equiv Q \mid P \quad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$

(res) $(\nu x)0 \equiv 0 \quad (\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$

$(\nu x)(P \mid Q) \equiv P \mid (\nu x)Q$

if $x \notin fn(P)$

$(\nu x)\mu.P \equiv \mu.(\nu x)P$ if $x \notin fn(\mu)$

(match) $[x = x]P \equiv P \quad [x = y]0 \equiv 0$

(unfold) $recZ.P \equiv P\{\text{rec}\, Z.P/Z.$

Structural congruence

The *structural congruence* $\equiv$ on π-calculus processes is the smallest congruence that satisfies the following:

(alpha) $P \equiv Q$ if P and Q are α-equivalent

(par) $P \mid 0 \equiv P \quad P \mid Q \equiv Q \mid P \quad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$

(res) $(\nu x)0 \equiv 0 \quad (\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$

$(\nu x)(P \mid Q) \equiv P \mid (\nu x)Q$

if $x \notin fn(P)$

$(\nu x)\mu.P \equiv \mu.(\nu x)P$ if $x \notin fn(\mu)$

(match) $[x = x]P \equiv P \quad [x = y]0 \equiv 0$

(unfold) $recZ.P \equiv P\{\text{rec}\, Z.P/Z.$

$(\theta\vec{x})P \equiv (\theta\vec{x})Q$ iff $P \equiv Q$.

Note that $P \equiv Q$ does not imply $fn(P) = fn(Q)$ (by rule **(match)**).

Structural congruence

The *structural congruence* $\equiv$ on π-calculus processes is the smallest congruence that satisfies the following:

(alpha) $P \equiv Q$ if P and Q are α-equivalent

(par) $P \mid 0 \equiv P \quad P \mid Q \equiv Q \mid P \quad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$

(res) $(\nu x)0 \equiv 0 \quad (\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$

$(\nu x)(P \mid Q) \equiv P \mid (\nu x)Q$

if $x \notin fn(P)$

$(\nu x)\mu.P \equiv \mu.(\nu x)P$ if $x \notin fn(\mu)$

(match) $[x = x]P \equiv P \quad [x = y]0 \equiv 0$

(unfold) $recZ.P \equiv P\{\text{rec}\, Z.P/Z.$

$(\theta\vec{x})P \equiv (\theta\vec{x})Q$ iff $P \equiv Q$.

Note that $P \equiv Q$ does not imply $fn(P) = fn(Q)$ (by rule **(match)**).

For each class $\mathcal{S} \subseteq \mathcal{P}^\theta$ of congruent processes, let us fix a representative process P such that $|fn(P)| = min\{|fn(Q)| \mid Q \in \mathcal{S}\}$.

Preliminary definitions (1)

The states of a θ-automaton represent sets of $\pi\theta$-processes, differing by vacuous θ's.

Without loss of generality (see Lemma on slide 22), we introduce θ-automata only for *closed* $\pi\theta$-processes.

Preliminary definitions (1)

The states of a θ-automaton represent sets of $\pi\theta$-processes, differing by vacuous θ's.
Without loss of generality (see Lemma on slide 22), we introduce θ-automata only for *closed* $\pi\theta$-processes.

$(\theta x_1 \ldots x_n)P \in \mathcal{P}_\emptyset^\theta$ is *canonical* if it is the representative of a $\equiv$-class and if no x_i is vacuous.

$\mathrm{can}(\mathcal{P}_\emptyset^\theta)$ denotes the set of canonical $\pi\theta$-processes.

Preliminary definitions (1)

The states of a θ-automaton represent sets of $\pi\theta$-processes, differing by vacuous θ's.
Without loss of generality (see Lemma on slide 22), we introduce θ-automata only for *closed* $\pi\theta$-processes.

$(\theta x_1 \ldots x_n)P \in \mathcal{P}_\emptyset^\theta$ is *canonical* if it is the representative of a $\equiv$-class and if no x_i is vacuous.

$\mathrm{can}(\mathcal{P}_\emptyset^\theta)$ denotes the set of canonical $\pi\theta$-processes.

The *orbit* of a canonical process $(\theta\vec{x})P$, is the set

$$\mathrm{orbit}((\theta\vec{x})P) \triangleq \{(\theta\vec{z}_0 x_1 \ldots \vec{z}_{n-1} x_n \vec{z}_n)P' \in \mathcal{P}_\emptyset^\theta \mid P' \equiv P \wedge |\vec{z}_0|, \ldots, |\vec{z}_n| \geq 0 \wedge \forall i, j. x_j \notin \vec{z}_i \}.$$

Preliminary definitions (2)

We denote by $\mathbb{M}(n, m)$ the set of partial strictly monotone functions from $\{1, \ldots, n\}$ to $\{1, \ldots, m\}$ and by $\mathbb{M}$ the set $\bigcup_{n,m \in \omega} \mathbb{M}(n, m)$.

Preliminary definitions (2)

We denote by $\mathbb{M}(n,m)$ the set of partial strictly monotone functions from $\{1,\ldots,n\}$ to $\{1,\ldots,m\}$ and by $\mathbb{M}$ the set $\bigcup_{n,m\in\omega}\mathbb{M}(n,m)$.

The *normalization function* $\|\cdot\| : \mathcal{P}^{\theta}_{\emptyset} \to \mathrm{can}(\mathcal{P}^{\theta}_{\emptyset})$ is defined by $\|(\theta x_1 \ldots x_n)P\| \triangleq (\theta y_1 \ldots y_m)P'$, where

- P' is the representative of the equivalence class of P,
- $\{y_1,\ldots,y_m\} = fn(P')$,
- $x_1 \ldots x_n = \vec{z}_0 y_1 \vec{z}_1 \ldots y_m \vec{z}_m$ (where $|\vec{z}_i| \geq 0$).

Preliminary definitions (2)

We denote by $\mathbb{M}(n,m)$ the set of partial strictly monotone functions from $\{1,\ldots,n\}$ to $\{1,\ldots,m\}$ and by $\mathbb{M}$ the set $\bigcup_{n,m\in\omega}\mathbb{M}(n,m)$.

The *normalization function* $\|\cdot\| : \mathcal{P}^{\theta}_{\emptyset} \to \mathrm{can}(\mathcal{P}^{\theta}_{\emptyset})$ is defined by $\|(\theta x_1 \ldots x_n)P\| \triangleq (\theta y_1 \ldots y_m)P'$, where

- P' is the representative of the equivalence class of P,
- $\{y_1,\ldots,y_m\} = fn(P')$,
- $x_1 \ldots x_n = \vec{z}_0 y_1 \vec{z}_1 \ldots y_m \vec{z}_m$ (where $|\vec{z}_i| \geq 0$).

The *reindexing* of $(\theta x_1 \ldots x_n)P$ is the partial strictly monotone function $\xi((\theta x_1 \ldots x_n)P) \in \mathbb{M}$ defined by $\xi((\theta x_1 \ldots x_n)P)(i) = j \iff x_i = y_j$.

θ-automata, formally

The *θ-automaton* $\mathcal{A}_P$ induced by $P \in \mathcal{P}_\emptyset^\theta$ is the triple $(\mathcal{S}, \|P\|, \mapsto)$, where:

- $\mathcal{S}$ is the set of *states*. Each state is the orbit of the canonical process corresponding to a descendant of P, and it is denoted by the canonical representative itself.
- $\|P\|$ is the *initial state*.
- $\mapsto \subseteq \mathcal{S} \times \mathcal{L}^\theta \times \mathcal{S}$ is the *transition relation* defined by:

$$P_1 \overset{\mu}{\mapsto} P_2 \text{ iff there exists } P_2' \text{ such that } P_1 \xrightarrow{\mu}\!\!\!\!\rightarrow_\emptyset P_2' \text{ and } \|P_2'\| = P_2 \ .$$

Finiteness result

Theorem

Let $(\theta\vec{x})P \in \mathcal{P}_{\emptyset}^{\theta}$ be a finitary process. Then $\mathcal{A}_{(\theta\vec{x})P}$ is finite.

Finiteness result

Theorem

Let $(\theta\vec{x})P \in \mathcal{P}^{\theta}_{\emptyset}$ be a finitary process. Then $\mathcal{A}_{(\theta\vec{x})P}$ is finite.

The fundamental Theorem above is based on the lemma below.

Lemma

If $P \in \mathcal{P}^{\theta}$, then the set $sub(P)$ of its subprocesses is finite.
Moreover, for all $Q \in sub(P)$: $sub(Q) \subseteq sub(P)$.

Definition

The *set of subprocesses* of P is $sub(P) \triangleq \{P\} \cup sub'(P)$, where:

$$sub'(0) = sub'(Z) \triangleq \emptyset \qquad sub'(P|Q) \triangleq sub(P) \cup sub(Q)$$

$$sub'(\pi.P) = sub'([x = y]P) = sub'((\nu x)P) = sub'((\theta x)P) \triangleq sub(P)$$

$$sub'(\text{rec}\, Z.P) \triangleq \{Q\{\text{rec}\, Z.P/Z\} \mid Q \in sub(P)\}$$

θ-automata at work

The natural application for θ-automata is in checking bisimilarity of $\pi\theta$-processes.

θ-automata at work

The natural application for θ-automata is in checking bisimilarity of $\pi\theta$-processes.

To this aim, we must keep track, at each step, of the correspondence between θ-bound names in the canonical processes, which is where the reindexing function $\xi((\theta x_1 \ldots x_n)P) \in \mathbb{M}$ comes into play.

θ-automata at work

The natural application for θ-automata is in checking bisimilarity of $\pi\theta$-processes.

To this aim, we must keep track, at each step, of the correspondence between θ-bound names in the canonical processes, which is where the reindexing function $\xi((\theta x_1 \ldots x_n)P) \in \mathbb{M}$ comes into play.

More precisely, we define a notion of $\mathbb{M}$-indexed bisimilarity on θ-automata and show that

Theorem

Let $P, Q \in \mathcal{P}_{\{x_1,\ldots,x_n\}}$, and let $f = \xi((\theta\vec{x})Q) \circ (\xi((\theta\vec{x})P))^{-1}$
Then, $P \sim Q$ iff $\mathcal{A}_{(\theta\vec{x})P}$ and $\mathcal{A}_{(\theta\vec{x})Q}$ are f-bisimilar.

Indexed bisimulation

Let $\mathcal{A} = (\mathcal{S}, P_0, \mapsto)$, $\mathcal{A}' = (\mathcal{S}', Q_0, \mapsto')$ be θ-automata.

An *indexed bisimulation* $\mathcal{R} \subseteq \mathcal{S} \times \mathbb{M} \times \mathcal{S}'$ is a relation such that,

for $(\theta\vec{x})P \in \mathcal{S}$, $(\theta\vec{y})Q \in \mathcal{S}'$, $f \in \mathbb{M}(|\vec{x}|, |\vec{y}|)$, if $((\theta\vec{x})P, f, (\theta\vec{y})Q) \in \mathcal{R}$ then:

let $\mathrm{dom}(f) = \{i_1, \ldots, i_k\}$, $\vec{x} = \vec{u}_0 x_{i_1} \vec{u}_1 \ldots x_{i_k} \vec{u}_k$, $\vec{y} = \vec{v}_0 y_{f(i_1)} \vec{v}_1 \ldots y_{f(i_k)} \vec{v}_k$,
$\vec{x} \cap \vec{y} = \emptyset$, $\vec{z} = \vec{u}_0 \vec{v}_0 x_{i_1} \vec{u}_1 \vec{v}_1 \ldots x_{i_k} \vec{u}_k \vec{v}_k$,

- if $(\theta\vec{z})P \overset{(\theta\vec{z})\mu}{\twoheadrightarrow}_{\emptyset} (\theta\vec{z})P'$, then there exists $(\theta\vec{z})Q'$ such that
  - $(\theta\vec{z})Q\{x_{i_1}/y_{f(i_1)}, \ldots, x_{i_k}/y_{f(i_k)}\} \overset{(\theta\vec{z})\mu}{\twoheadrightarrow}_{\emptyset} (\theta\vec{z})Q'$
  - $(\|(\theta\vec{z})P'\|, f', \|(\theta\vec{z})Q')\| \in \mathcal{R}$, where $f' = \xi((\theta\vec{z})Q') \circ (\xi((\theta\vec{z})P'))^{-1}$.
- if $(\theta\vec{z})P \overset{(\theta\vec{z})\mu}{\twoheadrightarrow}_{\emptyset} (\theta\vec{z}z')P'$, then there exists $(\theta\vec{z}z')Q'$ such that
  - $(\theta\vec{z})Q\{x_{i_1}/y_{f(i_1)}, \ldots, x_{i_k}/y_{f(i_k)}\} \overset{(\theta\vec{z})\mu}{\twoheadrightarrow}_{\emptyset} (\theta\vec{z}z')Q'$
  - $(\|(\theta\vec{z}z')P'\|, f', \|(\theta\vec{z}z')Q')\| \in \mathcal{R})$, where $f' = f_1 \cup f_2$,
    $f_1 = \xi((\theta\vec{z}z')Q') \circ (\xi((\theta\vec{z}z')P'))^{-1}$ and
    $$f_2 = \begin{cases} \{(\max(\mathrm{dom}(f_1)) + 1, \max(\mathrm{cod}(f_1)) + 1)\} & \text{if } z' \in fn(P') \cap fn(Q') \\ \emptyset & \text{otherwise} \end{cases}$$
- $((\theta\vec{y})Q, f^{-1}, (\theta\vec{x})P) \in \mathcal{R}$.

Indexed bisimilarity

The *indexed bisimilarity*, $\simeq$, is the greatest indexed bisimulation.

We say that the automata $\mathcal{A}$ and $\mathcal{A}'$ are *f-bisimilar* if $(P_0, f, Q_0) \in \simeq$, for some $f \in \mathbb{M}(|\vec{x}|, |\vec{y}|)$.

Indexed bisimilarity

The *indexed bisimilarity*, $\simeq$, is the greatest indexed bisimulation.

We say that the automata $\mathcal{A}$ and $\mathcal{A}'$ are *f-bisimilar* if $(P_0, f, Q_0) \in \simeq$, for some $f \in \mathbb{M}(|\vec{x}|, |\vec{y}|)$.

Now we have all ingredients to show that if the sets of states $\mathcal{S}$ and $\mathcal{S}'$ of the θ-automata are finite, then indexed bisimulations are finite objects, making verification effective

In particular, the next result gives an upper bound k of the domain of the reindexing functions between all descendents of P and Q.

Since $\mathbb{M}(k, k)$ is finite, there are only finitely many candidate relations to be indexed bisimulations.

Hence, we have an algorithm for deciding bisimilarity of finitary processes.

Finite verification is possible

Theorem

Let $\mathcal{A}_{(\theta\vec{x})P} = (\mathcal{S}, \|(\theta\vec{x})P\|, \mapsto)$ and $\mathcal{A}_{(\theta\vec{x})Q} = (\mathcal{S}', \|(\theta\vec{x})Q\|, \mapsto')$, for $P, Q \in \mathcal{P}_{\{x_1,\ldots,x_n\}}$ finitary. Let $k = \max(\overline{\deg}(P) \times |n(P)|, \overline{\deg}(Q) \times |n(Q)|)$.

Then $P \sim Q$ iff $\exists\, \mathcal{R} \subseteq \mathcal{S} \times \mathbb{M}(k,k) \times \mathcal{S}'$, an indexed bisimulation, such that $(\|(\theta\vec{x})P\|, f, \|(\theta\vec{x})Q\|) \in \mathcal{R}$ (where $f = \xi((\theta\vec{x})Q) \circ (\xi((\theta\vec{x})P))^{-1}$)

Concluding remarks

Recap

- We have introduced $\pi\theta$-calculus, a conservative extension of π-calculus
- The operational semantics is finitely branching,
- The operational and abstract semantics of $\pi\theta$-calculus do not involve any side conditions on names,
- We have given a (bi)coalgebraic presentation of bisimilarity, in the spirit of final semantics
- We have introduced θ-automata which allow us to get finite representations for finitary processes by dealing with vacuous θ's and structural congruence
- We have also shown that, by exploiting θ-automata, finite verification of bisimilarity is possible for finitary processes

Related and future work

Closely related work

- HD-automata [Montanari&Pistore]
- Fresh graphs [Gabbay]

Related and future work

Closely related work

- HD-automata [Montanari&Pistore]
- Fresh graphs [Gabbay]

Future directions

- To address late and open bisimilarity
- HOAS presentation of $\pi\theta$-calculus
- Push further the study of θ-automata, by
  - introducing a general notion of θ-automaton, independent from the π-calculus
  - prove standard results on automata such as minimalization
  - define corresponding notion of transition system, generalizing the transition system of the $\pi\theta$-calculus.

Acknowledgments

Research supported by

- MIUR Project COFIN 2001013518 CoMeta
- EU Project IST-2001-32747 Agile
- EU Project IST-2001-33477 DART
- Italian CNR fellowship for research on Information Sciences and Technologies

Acknowledgments

Research supported by

- MIUR Project COFIN 2001013518 CoMeta
- EU Project IST-2001-32747 Agile
- EU Project IST-2001-33477 DART
- Italian CNR fellowship for research on Information Sciences and Technologies

(- : THANKS FOR THE ATTENTION : -)