

Modelling Fusion calculus using History-Dependent automata

Gianluigi Ferrari Ugo Montanari **Emilio Tuosto**
{giangi,ugo,etuosto}@di.unipi.it



Dipartimento di Informatica,
Università di Pisa
Italy

Kidane Yemane

{kyemane,Bjorn.Victor}@it.uu.se

Björn Victor



Institutionen för informationsteknologi,
Uppsala Universitet,
Sweden,












CALCO05: Swansea, 3 – 6 September 2005



Profundis

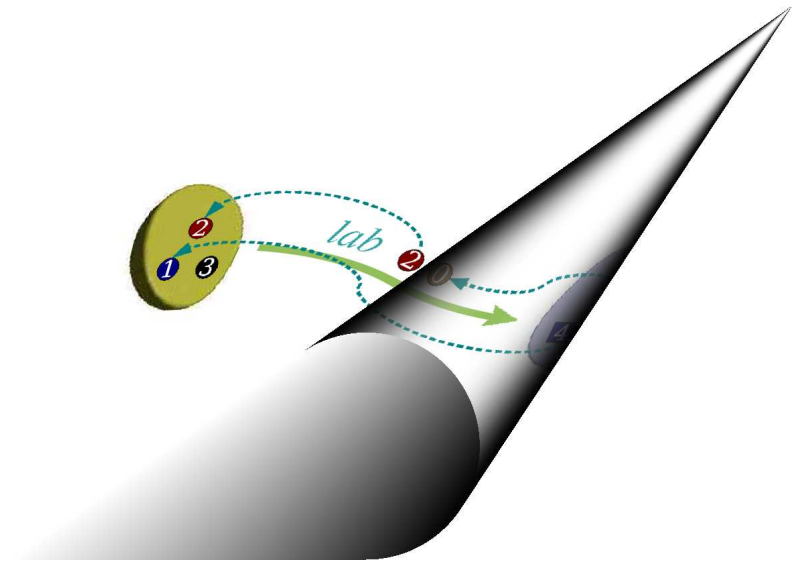


- 📍 Overview: verification of nominal calculi
- 📍 Definition of HD-automata
 - 📍 coalgebra over a category of named sets
 - 📍 description of the minimisation procedure for HD-automata
- 📍 A new (symbolic) semantics for Fusion calculus
- 📍 HD-automata for Fusion calculus
- 📍 Minimisation of HD-automata for Fusion calculus
- 📍 Some conclusions

-  Nominal calculi have been successfully applied to global computing for
 -  specifying
 -  verifying
-  Names as suitable abstractions for
 -  mobility
 -  localities
 -  distributed object
 -  security keys
 -  ...
-  Nominal calculi also provide a basic programming model for novel programming languages [CL99, BCF04]
-  and for workflow languages for Web Service coordination [BMM05, LZ05]

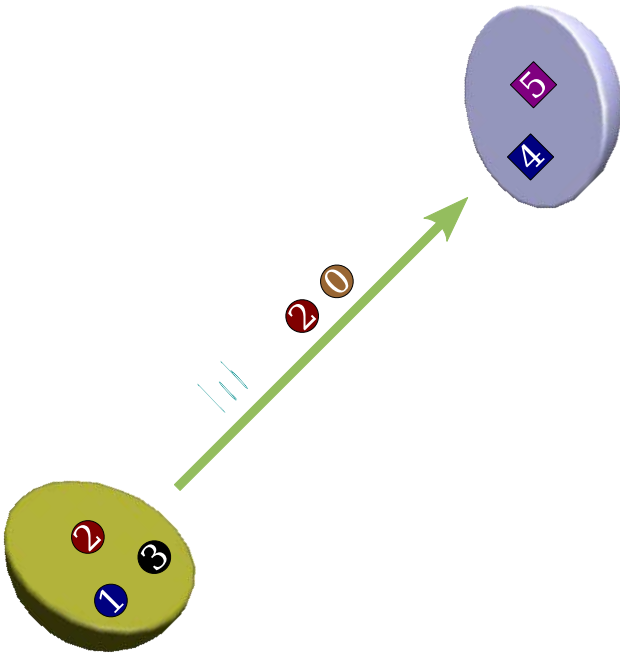
- Verification via semantic equivalence is intrinsically difficult for nominal calculi
 - Models tend to be infinite even for simplest cases
 - Explicit mechanisms are necessary to deal with names
- Symbolic semantics [HL95, BD96, Lin03]
 - Takes a syntax-based approach and generalises standard operational semantics by keeping track of equalities among names
 - Transitions are derived in the context of such constraints
 - The main advantage is that it yields a smaller transition system
- Syntax-free models explicitly deal with names regardless of the syntax
- HD-automata as an operational model of history-dependent calculi [Pis99, MP98]
- Verification techniques
 - Minimisation: partition refinement algorithm [FMP02]
 - Type-theoretic definition of HD-automata [FMT05]...
 - ...exploited in Mihda

History Dependent Automata

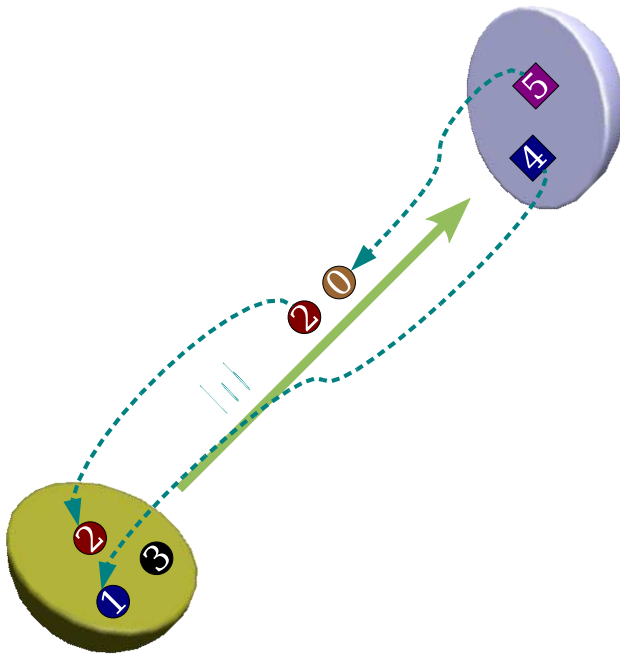


HD-automata...intuitively

- states and transitions have **local** names:
 - names **explicit** in the operational model
 - so that HD-automata model name creation/deallocation or extrusion
- Starting state has three names: 1, 2 and 3
- Arrival state has two names: 4 and 5
- The transition exposes 2 and a (fresh) name 0



HD-automata...intuitively



- states and transitions have local names:
 - names explicit in the operational model
 - so that HD-automata model name (de)allocation or extrusion
- State s has three names: 1, 2 and 3
- State d has two names: 4 and 5
- The transition exposes 2 and a (fresh) name 0
- $\sigma : 4 \mapsto 1$ and $\sigma : 5 \mapsto 0$, the new name
- 3 is “discharged”
- HD-automata + symmetries based on permutation algebras [MP00]
- compact representation of behaviour (states differing only for the renaming of local names are collapsed)

Named sets & named functions

Let \mathcal{N} be a set of names, and $\text{sym}(N) \triangleq \{\rho \in \text{Aut}(\mathcal{N}) \mid \forall x \notin N. \rho(x) = x\}$, if $N \subseteq \mathcal{N}$,

Definition A single-sorted algebra $\langle S, O \rangle$ is a permutation algebra iff

- O is a set of unary operators
- $\exists id \in O. \forall x : S. id(x) = x$
- $\forall \rho, \rho' \in O. \exists \hat{\rho} \in O. \forall x : S. \hat{\rho}(x) = \rho(\rho'(x))$

States of HD-automata are defined by means of **named sets**

Definition A **named set** (ns) is a pair $\langle Q, g \rangle$ s.t.

1. Q is a permutation algebra interpreting operators in $\text{Aut}(\mathcal{N})$
2. $g : Q \rightarrow \bigcup_{N \in \wp_{\text{fin}}(\mathcal{N})} \{\text{sym}(N)\}$ s.t.
 $\forall \rho \in g(q). q = \rho(q)$

Transitions among states are represented by means of **named functions**

Definition A **named function** $\langle h, \Sigma \rangle : D \rightarrow E$ is s.t. $h : Q_D \rightarrow Q_E$ and $\Sigma : Q_D \rightarrow \wp_{\text{fin}}(Q_E \times \mathcal{N}_*^{\mathcal{N}})$ and, for all $q \in Q_D$ and $(e, \sigma) \in \Sigma(q)$,

1. σ is injective, $\sigma(|e|) \subseteq |q| \cup \{\star\}$ and $\sigma(x) = x$, for any $x \in \mathcal{N} \setminus |e|$
2. $\sigma; g_D(q) \subseteq \Sigma_2(q) \wedge g_E(h(q)); \sigma = \Sigma_2(q)$

Named sets & named functions

Let \mathcal{N} be a set of names, and $\text{sym}(N) \triangleq \{\rho \in \text{Aut}(\mathcal{N}) \mid \forall x \notin N. \rho(x) = x\}$, if $N \subseteq \mathcal{N}$,



States of HD-automata are defined by means of **named sets**

Definition A **named set** (ns) is a pair $\langle Q, g \rangle$ s.t.

1. Q is a permutation algebra interpreting operators in $\text{Aut}(\mathcal{N})$
2. $g : Q \rightarrow \bigcup_{N \in \wp_{\text{fin}}(\mathcal{N})} \{\text{sym}(N)\}$ s.t.
 $\forall \rho \in g(q). q = \rho(q)$

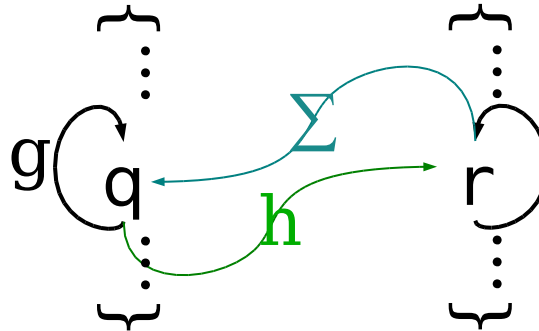
Transitions among states are represented by means of **named functions**

Definition A **named function** $\langle h, \Sigma \rangle : D \rightarrow E$ is s.t. $h : Q_D \rightarrow Q_E$ and $\Sigma : Q_D \rightarrow \wp_{\text{fin}}(Q_E \times \mathcal{N}_*^{\mathcal{N}})$ and, for all $q \in Q_D$ and $(e, \sigma) \in \Sigma(q)$,

1. σ is injective, $\sigma(|e|) \subseteq |q| \cup \{\star\}$ and $\sigma(x) = x$, for any $x \in \mathcal{N} \setminus |e|$
2. $\sigma; g_D(q) \subseteq \Sigma_2(q) \wedge g_E(h(q)); \sigma = \Sigma_2(q)$

Named sets & named functions

Let \mathcal{N} be a set of names, and $\text{sym}(N) \triangleq \{\rho \in \text{Aut}(\mathcal{N}) \mid \forall x \notin N. \rho(x) = x\}$, if $N \subseteq \mathcal{N}$,



States of HD-automata are defined by means of **named sets**

Definition A **named set** (ns) is a pair $\langle Q, g \rangle$ s.t.

1. Q is a permutation algebra interpreting operators in $\text{Aut}(\mathcal{N})$
2. $g : Q \rightarrow \bigcup_{N \in \wp_{\text{fin}}(\mathcal{N})} \{\text{sym}(N)\}$ s.t.
 $\forall \rho \in g(q). q = \rho(q)$

Transitions among states are represented by means of **named functions**

Definition A **named function** $\langle h, \Sigma \rangle : D \rightarrow E$ is s.t. $h : Q_D \rightarrow Q_E$ and $\Sigma : Q_D \rightarrow \wp_{\text{fin}}(Q_E \times \mathcal{N}_*^{\mathcal{N}})$ and, for all $q \in Q_D$ and $(e, \sigma) \in \Sigma(q)$,

1. σ is injective, $\sigma(|e|) \subseteq |q| \cup \{\star\}$ and $\sigma(x) = x$, for any $x \in \mathcal{N} \setminus |e|$
2. $\sigma; g_D(q) \subseteq \Sigma_2(q) \wedge g_E(h(q)); \sigma = \Sigma_2(q)$

Category of named sets and HD-automata

- The category NS has named sets as objects and named functions as morphisms
 1. $\perp = \langle \emptyset, \emptyset \rangle$ is initial object, $I = \langle \{*\}, * \mapsto \emptyset \rangle$ is the terminal object and
 2. the covariant powerset functor on **Set** is $\wp_{\text{fin}}(D) = \langle \wp_{\text{fin}}(D_Q), g \rangle$, where, given $Q \subseteq Q_D$, $g(Q) = \{ \rho \mid \rho \text{ is a permutation over } \bigcup_{q \in Q} |q| \} \wedge Q\rho = Q$.

Definition Given a ns L , a HD-automaton over L is a coalgebra for

$$T_L(D) = \wp_{\text{fin}}(L \otimes D)$$

- where the pairing operation $D \otimes E = \langle Q_D \times Q_E \times \mathcal{N}_*, \mathcal{N}, g \rangle$ is s.t.
 $g : Q_D \times Q_E \rightarrow \bigcup_{N, M \in \wp_{\text{fin}}(\mathcal{N})} \{ \text{sym}(N) + \text{sym}(M) \}$ where

$$g(d, e) = \{ \rho_1 + \rho_2 \mid \rho_1 \in g_D(d) \wedge \rho_2 \in g_E(e) \}$$

(formally, $D \otimes E$ is not a ns but $g(d, e)$ is a symmetry on $|d| + |e|$)

Let $T : \mathbf{NS} \rightarrow \mathbf{NS}$ be the functor acting as T_L on $\text{obj}(\mathbf{NS})$ and mapping $\langle h_D, \Sigma_D \rangle \in \mathbf{NS}(E, F)$ to the nf

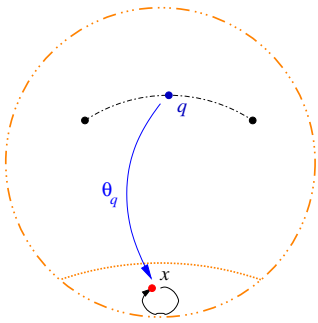
$$\begin{cases} h(B) &= \{ \langle l, h_D(q), \sigma \rangle \mid \langle l, q, \sigma \rangle \in B \} \\ \Sigma(B) &= \{ \langle l, h_D(q), \sigma'; \sigma \rangle \mid \langle l, q, \sigma \rangle \in B \wedge \langle l, q', \sigma' \rangle \in \Sigma_D(q) \} \end{cases}$$

where $B \in \wp_{\text{fin}}(L \otimes E)$

The minimisation algorithm on a T_L coalgebra $(D, K : D \rightarrow T_L(D))$ is

$$\begin{aligned} H_{(0)} &\triangleq \langle q \mapsto \perp, q \mapsto \emptyset \rangle, \quad \text{where } \text{dom}(H_{(0)}) = D \\ H_{(i+1)} &\triangleq K; N(T(H_{(i)})), \end{aligned}$$

where N is a normalisation functor, i.e. $N(D)$ is isomorphic to a subset of D

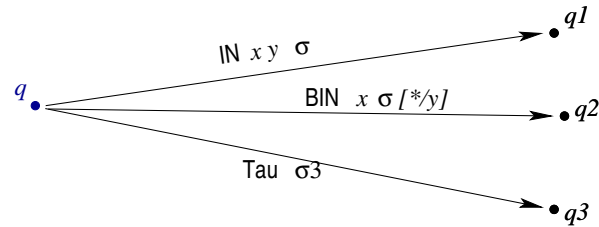
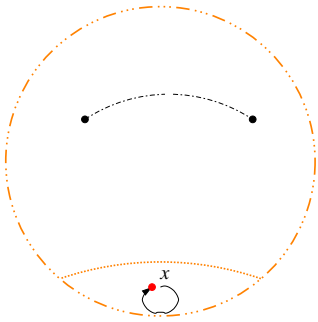


- At the $(i + 1)$ -th step, the outgoing transitions in K are “steeped” into $H_{(i)}$ through T and then normalised

The minimisation algorithm on a T_L coalgebra $(D, K : D \rightarrow T_L(D))$ is

$$\begin{aligned} H_{(0)} &\triangleq \langle q \mapsto \perp, q \mapsto \emptyset \rangle, \quad \text{where } \text{dom}(H_{(0)}) = D \\ H_{(i+1)} &\triangleq K; N(T(H_{(i)})), \end{aligned}$$

where N is a normalisation functor, i.e. $N(D)$ is isomorphic to a subset of D



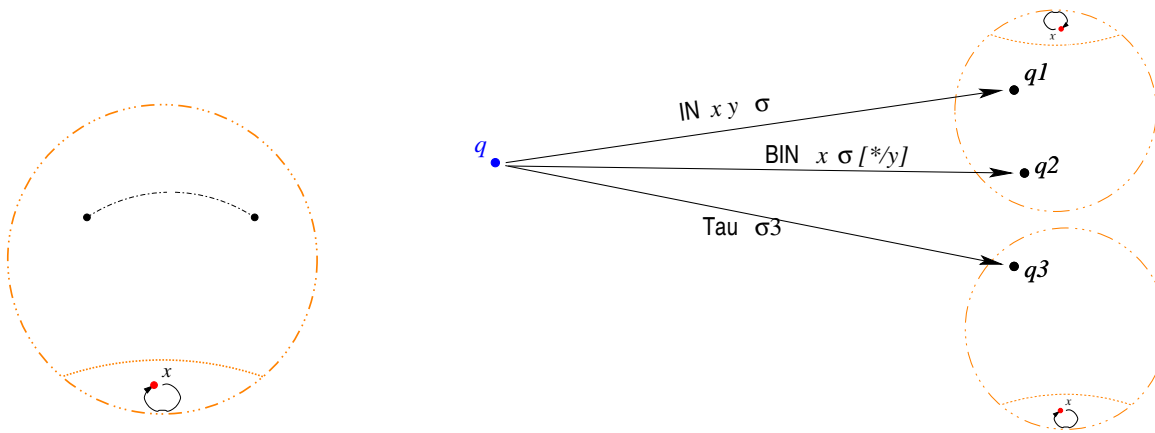
- At the $(i + 1)$ -th step, the outgoing transitions in K are “steeped” into $H_{(i)}$ through T and then normalised

The minimisation algorithm on a T_L coalgebra $(D, K : D \rightarrow T_L(D))$ is

$$\begin{aligned} H_{(0)} &\triangleq \langle q \mapsto \perp, q \mapsto \emptyset \rangle, \quad \text{where } \text{dom}(H_{(0)}) = D \\ H_{(i+1)} &\triangleq K; N(T(H_{(i)})), \end{aligned}$$

where N is a normalisation functor, i.e. $N(D)$ is isomorphic to a subset of D

Minimising HD-automata



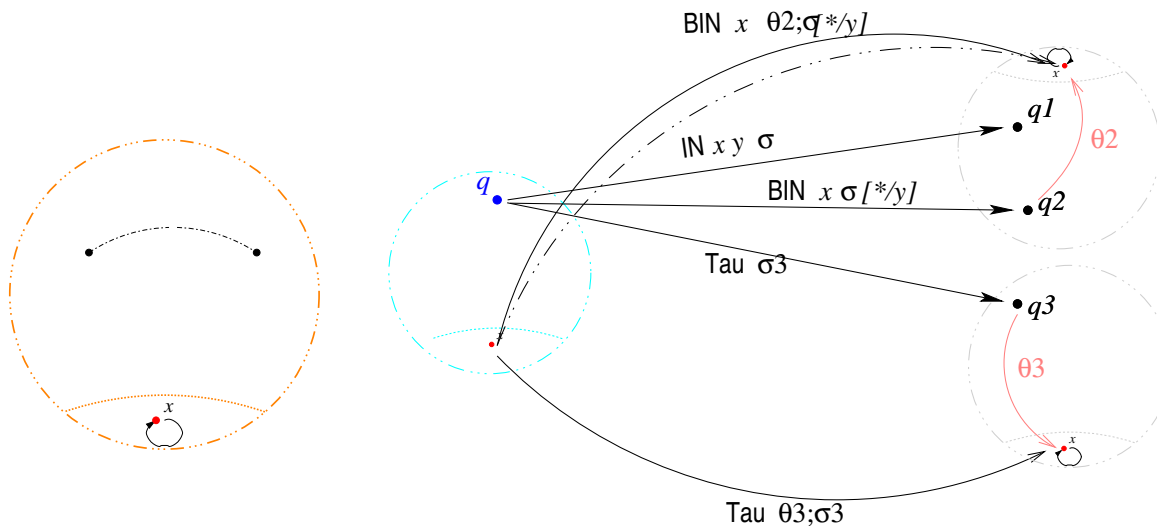
At the $(i + 1)$ -th step, the outgoing transitions in K are “steeped” into $H_{(i)}$ through T and then normalised

The minimisation algorithm on a T_L coalgebra $(D, K : D \rightarrow T_L(D))$ is

$$\begin{aligned} H_{(0)} &\triangleq \langle q \mapsto \perp, q \mapsto \emptyset \rangle, \quad \text{where } \text{dom}(H_{(0)}) = D \\ H_{(i+1)} &\triangleq K; N(T(H_{(i)})), \end{aligned}$$

where N is a normalisation functor, i.e. $N(D)$ is isomorphic to a subset of D

Minimising HD-automata



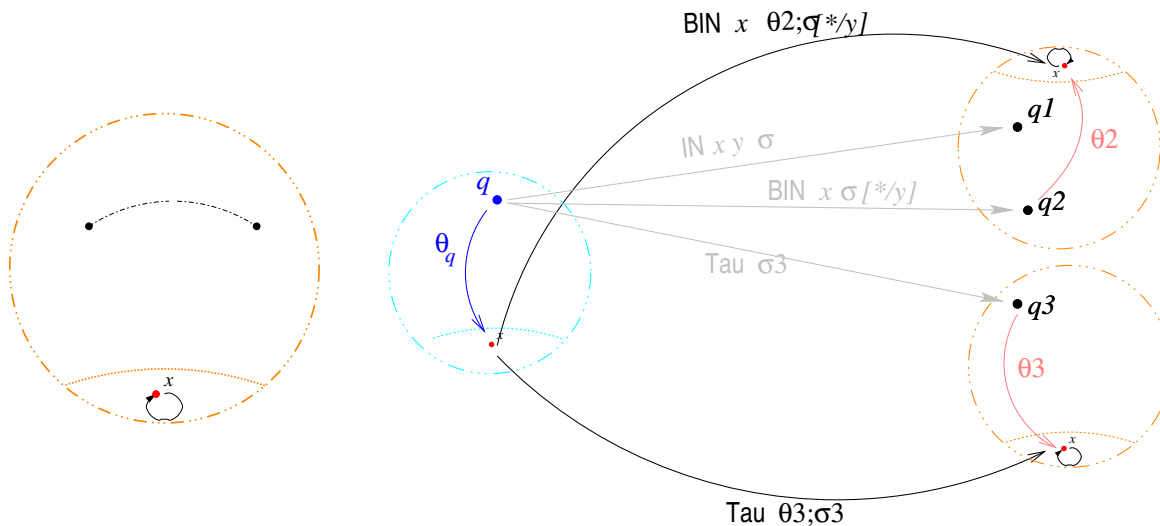
- At the $(i + 1)$ -th step, the outgoing transitions in K are “steeped” into $H_{(i)}$ through T and then normalised

The minimisation algorithm on a T_L coalgebra $(D, K : D \rightarrow T_L(D))$ is

$$\begin{aligned} H_{(0)} &\triangleq \langle q \mapsto \perp, q \mapsto \emptyset \rangle, \quad \text{where } \text{dom}(H_{(0)}) = D \\ H_{(i+1)} &\triangleq K; N(T(H_{(i)})), \end{aligned}$$

where N is a normalisation functor, i.e. $N(D)$ is isomorphic to a subset of D

Minimising HD-automata



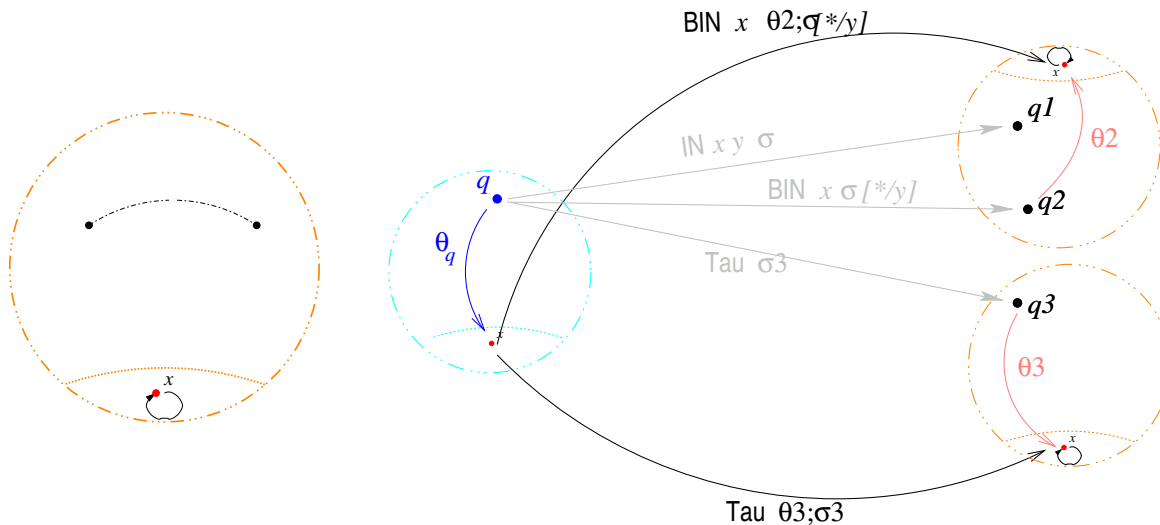
- At the $(i + 1)$ -th step, the outgoing transitions in K are “steeped” into $H_{(i)}$ through T and then normalised

The minimisation algorithm on a T_L coalgebra $(D, K : D \rightarrow T_L(D))$ is

$$\begin{aligned} H_{(0)} &\triangleq \langle q \mapsto \perp, q \mapsto \emptyset \rangle, \quad \text{where } \text{dom}(H_{(0)}) = D \\ H_{(i+1)} &\triangleq K; N(T(H_{(i)})), \end{aligned}$$

where N is a normalisation functor, i.e. $N(D)$ is isomorphic to a subset of D

Minimising HD-automata



- At the $(i + 1)$ -th step, the outgoing transitions in K are “steeped” into $H_{(i)}$ through T and then normalised

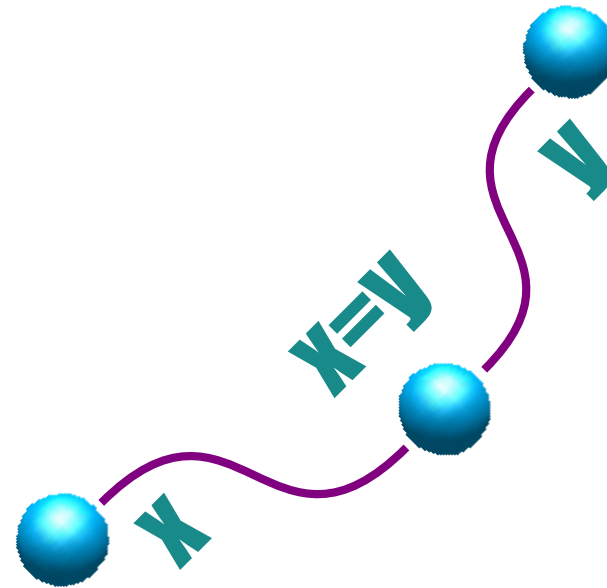
The minimisation algorithm on a T_L coalgebra $(D, K : D \rightarrow T_L(D))$ is

$$\begin{aligned} H_{(0)} &\triangleq \langle q \mapsto \perp, q \mapsto \emptyset \rangle, \quad \text{where } \text{dom}(H_{(0)}) = D \\ H_{(i+1)} &\triangleq K; N(T(H_{(i)})), \end{aligned}$$

where N is a normalisation functor, i.e. $N(D)$ is isomorphic to a subset of D

Theorem The iterative partition refinement algorithm is convergent on finite HD-automata whenever the normalisation functor is monotone on nfs.

Fusion calculus



$$\alpha ::= u\tilde{x} \mid \bar{u}\tilde{x} \mid \varphi$$

$$\varphi ::= \{\tilde{x} = \tilde{y}\}$$

$$P ::= \mathbf{0} \mid \alpha.Q \mid Q + R \mid Q \mid R \\ \mid (x)Q \mid [x = y]Q \mid A\langle\tilde{x}\rangle$$

Input and Output are
symmetric

$$\alpha ::= u\tilde{x} \mid \bar{u}\tilde{x} \mid \varphi$$

$$\varphi ::= \{\tilde{x} = \tilde{y}\}$$

$$P ::= \mathbf{0} \mid \alpha.Q \mid Q + R \mid Q \mid R \\ \mid (x)Q \mid [x = y]Q \mid A\langle\tilde{x}\rangle$$

total equivalence relations over \mathcal{N}
(i.e. $\text{dom}(\varphi) = \mathcal{N}$) with only finitely many
non-singular equivalence classes

$$\alpha ::= u\tilde{x} \mid \bar{u}\tilde{x} \mid \varphi$$

$$\varphi ::= \{\tilde{x} = \tilde{y}\}$$

$$P ::= \mathbf{0} \mid \alpha.Q \mid Q + R \mid Q \mid R \\ \mid (x)Q \mid [x = y]Q \mid A\langle\tilde{x}\rangle$$

$$\alpha ::= u\tilde{x} \mid \bar{u}\tilde{x} \mid \varphi$$

$$\varphi ::= \{\tilde{x} = \tilde{y}\}$$

$$P ::= \mathbf{0} \mid \alpha.Q \mid Q + R \mid Q \mid R \\ \mid (x)Q \mid [x = y]Q \mid A\langle\tilde{x}\rangle$$

Fusion actions as prefixes:
represent an obligation to
make lhs and rhs equal
everywhere

Canonical symbolic semantics (1)

- Symbolic semantics efficiently characterise bisimulation equivalences for value-passing calculi.
- Canonical symbolic semantics as for the π -calculus [San96, PS01]

Symbolic transition $P \xRightarrow{M, \gamma} Q$

M is the enabling condition of the action γ

Canonical symbolic semantics (1)

- Symbolic semantics efficiently characterise bisimulation equivalences for value-passing calculi.
- Canonical symbolic semantics as for the π -calculus [San96, PS01]

Symbolic transition

$$P \xrightarrow{M, \gamma} Q$$

The substitutive effect of M is an idempotent substitution σ_M s.t.

$$\sigma_M(x) = \sigma_M(y) \iff M \Rightarrow x = y$$

M is the enabling condition of the action γ

Canonical symbolic semantics (1)

- Symbolic semantics efficiently characterise bisimulation equivalences for value-passing calculi.
- Canonical symbolic semantics as for the π -calculus [San96, PS01]

Symbolic transition

$$P \xrightarrow{M, \gamma} Q$$

We apply substitutions σ_M and σ_γ to the continuations of the transition. Thus simplifies the definition of bisimulation and more in line with the minimisation algorithms of HD-automata.

M is the enabling condition of the action γ

pref
$$\frac{}{\alpha . P \xrightarrow{\emptyset, \alpha} P \sigma_\alpha}$$

sum
$$\frac{P \xrightarrow{M, \gamma} P'}{P + Q \xrightarrow{M, \gamma} P'}$$

par
$$\frac{P \xrightarrow{M, \gamma} P'}{P \mid Q \xrightarrow{M, \gamma} P' \mid Q \sigma_M \sigma_\gamma}$$

scope
$$\frac{P \xrightarrow{M, \varphi} P', \ z \varphi x, \ z \neq x, \ z \notin n(M)}{(z)P \xrightarrow{M, \varphi \setminus z} P' \{x/z\}}$$

match
$$\frac{P \xrightarrow{M, \gamma} P' \quad M' = M \star [x = y]}{[x = y]P \xrightarrow{M', \gamma \sigma_{[x=y]}} P' \sigma_{M'}}$$

pass
$$\frac{P \xrightarrow{M, \gamma} P', \ z \notin n(M, \gamma)}{(z)P \xrightarrow{M, \gamma} (z)P'}$$

open
$$\frac{P \xrightarrow{M, (\tilde{y})a \tilde{x}} P', \ z \in \tilde{x} - \tilde{y}, \ a \notin \{z, \bar{z}\}, \ z \notin n(M)}{(z)P \xrightarrow{M, (z\tilde{y})a \tilde{x}} P'}$$

com
$$\frac{P \xrightarrow{M, u\tilde{x}} P', \ Q \xrightarrow{N, \bar{v}\tilde{y}} Q', \ |\tilde{x}| = |\tilde{y}|, \ L = M \star N \star [u = v], \ \varphi = \{\tilde{x} = \tilde{y}\} \sigma_L}{P \mid Q \xrightarrow{L, \varphi} (P' \mid Q') \sigma_L \sigma_\varphi}$$

Symbolic hyperbisimulation and symbolic open bisimulation [San96, PS96] have similar definitions, but the former avoids distinctions.

Definition A binary symmetric process relation \mathcal{S} is a symbolic hyperbisimulation if $(P, Q) \in \mathcal{S}$ implies:

\mathcal{S} implies:

If $P \xrightarrow{M, \gamma} P'$ with $\text{bn}(\gamma) \cap \text{fn}(Q) = \emptyset$ then $Q \xrightarrow{N, \gamma'} Q'$ such that

• $M \Rightarrow N$,

• $\gamma = \gamma' \sigma_M$, (note $\gamma = \gamma \sigma_M$)

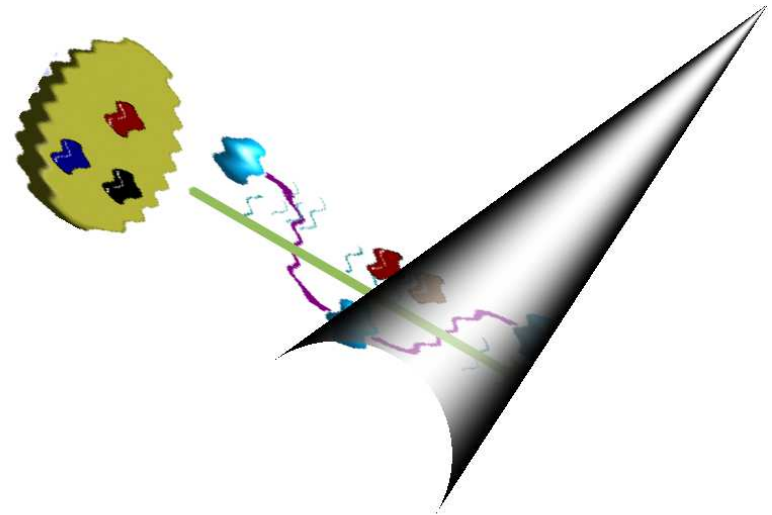
• and $(P', Q' \sigma_M) \in \mathcal{S}$ (note $P' = P' \sigma_M$).

P is symbolically hyperequivalent to Q , written $P \simeq Q$, if $(P, Q) \in \mathcal{S}$ for some symbolic hyperbisimulation \mathcal{S} .

Since the symbolic semantics applies the substitutive effects, we can leave most of that out of the bisimulation definition. It is still necessary to apply substitution corresponding to the stronger condition, σ_M , to the label and continuation of the transition of Q . (Note that $Q' \sigma_M = Q' \sigma_M \sigma_\gamma$.)

Theorem $P \sim Q$ iff $P \simeq Q$

From Fusion calculus to HD-automata



- Labels consists of
 - enabling conditions
 - actions
- Both of them can be represented as nss
- Let $Lab = \{\text{tau}, \text{in}, \text{out}, \text{fuse}\}$ and M denote the ns $\langle \{\bullet\}, g \rangle$ where $g = \{id_2, \text{exch}_2\}$

Definition The labels for Fusion calculus are the nss $M \otimes \langle Lab, g \rangle$ where

$M = M \otimes \dots \otimes M$ M is the enabling part

$$\|\text{tau}\| = 0$$

$$\|\text{fuse}\| = 2 \quad \wedge \quad g(\text{fuse}) = \{id_2, \text{exch}_2\}$$

$$\|\text{in}\|, \|\text{out}\| \leq 2 \quad \wedge \quad g(\text{in}), g(\text{out}) \text{ have only the identity permutation.}$$

Let's define, for a Fusion calculus agent P ,

$$Q_{D[P]} \triangleq \{P\} \cup \bigcup_{P \xrightarrow{M, \gamma} P'} \{P'\} \cup Q_{D[P']}$$

namely $Q_{D[P]}$ are the processes reachable from P .

Let's define, for a Fusion calculus agent P ,

$$Q_{D[P]} \triangleq \{P\} \cup \bigcup_{P \xrightarrow{M, \gamma} P'} \{P'\} \cup Q_{D[P']}$$

namely $Q_{D[P]}$ are the processes reachable from P .

It is trivial to equip $Q_{D[P]}$ with a named set structure. Indeed, for any $q \in Q_{D[P]}$, the group component $g_{D[P]}(q)$ is the identity on $\text{fn}(q)$.

$D[P]$ denotes such ns

Let's define, for a Fusion calculus agent P ,

$$Q_{D[P]} \triangleq \{P\} \cup \bigcup_{P \xrightarrow{M, \gamma} P'} \{P'\} \cup Q_{D[P']}$$

namely $Q_{D[P]}$ are the processes reachable from P .

It is trivial to equip $Q_{D[P]}$ with a named set structure. Indeed, for any $q \in Q_{D[P]}$, the group component $g_{D[P]}(q)$ is the identity on $\text{fn}(q)$.

$D[P]$ denotes such ns

The HD-automaton associated to P is the T_L -coalgebra $K[P]$ s.t.

$$h_{K[P]}(q) = \{\langle l, q', \sigma \rangle \mid q \xrightarrow{M, \gamma} q' \wedge l \text{ corresponds to } M, \gamma\}$$

Given $\langle l, q', \sigma \rangle \in h_{K[P]}(q)$

- σ maps $\text{fn}(q')$ on $\text{fn}(q)$ and new names to \star
- and similarly for the names of l

Minimising HD-automata for Fusion calculus

Let us now define the normalisation functor for Fusion calculus.

Definition Let $\langle l, q, \sigma \rangle$ and $\langle l', q, \sigma' \rangle$ be two hdt of K . Assuming that the matching ns of l (resp. l') is M (resp. M'), l is redundant wrt l' iff

- l and l' have the same action part and
- $\llbracket M \rrbracket_\sigma$ logically implies $\llbracket M' \rrbracket_{\sigma'}$ but not vice versa.

Definition Let $q \in \text{dom}(K)$ be a state. A hdt $\langle l, q', \sigma \rangle$ is redundant for q if there is $\langle l', q', \sigma' \rangle$ in $\Sigma_K(q)$ such that l is redundant wrt l' and, for a substitution σ'' accomplishing with the interpretation of the enabling part of l , $\sigma'; \sigma'' = \sigma$.

The intuition is that $t = \langle l, q', \sigma \rangle$ is dominated by another transition $t' = \langle l', q', \sigma' \rangle$

- reaching the same target state as t and with the same label but having
- enabling conditions weaker than those of t and,
- under the conditions of t , the names associated to the label of t' are the same as those of t .

Definition The normalisation functor for Fusion calculus $N : \mathbf{NS} \rightarrow \mathbf{NS}$ is as follows:

$$N(D) = \begin{cases} \langle h, \Sigma \rangle & D = \langle h_D, \Sigma_D \rangle \in \mathbf{NS}(\wp_{\text{fin}}(L \otimes E), \wp_{\text{fin}}(L \otimes F)) \text{ for } E, F \in \text{obj}(\mathbf{NS}) \\ D & \text{otherwise.} \end{cases}$$

where, for $B \in \wp_{\text{fin}}(L \otimes E)$,

$$\begin{aligned} h(B) &= \{ \langle l, q, \sigma \rangle \mid \langle l, q, \sigma \rangle \text{ not redundant in } \Sigma(B) \} \\ \Sigma(B) &= h(B) \end{aligned}$$

N filters those transitions out of a given state q that are redundant because of the presence of another transition having weaker conditions on names.

The functor N is monotonic on nfs.

Theorem The minimisation algorithm converges on finite HD-automata for Fusion calculus.

N_H mimics redundancy conditions of symbolic hyperbisimulation. Indeed,

Theorem Two finitary Fusion calculus processes are hyperbisimilar iff they have the same minimal realisation.

Conclusions and future directions

- New presentation of the minimisation algorithm
- HD-automata machinery for Fusion calculus: providing a new symbolic semantics of Fusion calculus
- In the future, we wish to extend the approach to open semantics of π -calculus.
- It would be interesting to study relationships among presheaf models of open semantics of π -calculus ([\[GVY04\]](#)) and other approaches e.g., [\[FS04, GMM03\]](#)

References

- [BCF04] Nick Benton, Luca Cardelli, and Cédric Fournet. Modern Concurrency Abstractions for C#. *ACM Transactions on Programming Languages and Systems*, 26(5):269–304, September 2004.
- [BD96] Michele Boreale and Rocco De Nicola. A Symbolic Semantics for the π -calculus. *Information and Computation*, 126(1):34–52, April 1996.
- [BMM05] Roberto Bruni, Hernán Melgratti, and Ugo Montanari. Theoretical Foundations for Compensations in Flow Composition Languages. In *Annual Symposium on Principles of Programming Languages POPL*, 2005. To appear.
- [CL99] Silvain Conchon and Fabrice Le Fessant. Jocaml: Mobile Agents for Objective-Caml. In *International Symposium on Agent Systems and Applications*, pages 22–29, Palm Springs, California, October 1999.
- [FMP02] Gianluigi Ferrari, Ugo Montanari, and Marco Pistore. Minimizing Transition Systems for Name Passing Calculi: A Co-algebraic Formulation. In Mogens Nielsen and Uffe Engberg, editors, *FOSSACS 2002*, volume 2303 of *Lecture Notes in Computer Science*, pages 129–143. Springer-Verlag, 2002.
- [FMT05] Gianluigi Ferrari, Ugo Montanari, and Emilio Tuosto. Coalgebraic Minimisation of HD-automata for the π -Calculus in a Polymorphic λ -Calculus. *Theoretical Computer Science*, 331:325–365, 2005.
- [FS04] Marcelo Fiore and Sam Staton. Comparing Operational Models of Name-Passing Process Calculi. In Jiří Adamek, editor, *Proc. CMCS’04, ENTCS*. Elsevier, 2004.
- [GMM03] Fabio Gadducci, Marino Miculan, and Ugo Montanari. About permutation algebras and sheaves (and named sets, too!). Technical Report UDMI/26/2003/RR, Department of Mathematics and Computer Science, University of Udine, 2003.
- [GVY04] Neil Ghani, Bjorn Victor, and Kidane Yemane. Relationally Staged Computation in the π -calculus. In *Proceedings of CMCS 2004*, number 106, 11 in ENTCS, pages 105–120, 2004.
- [HL95] Matthew Hennessy and Huimin Lin. Symbolic Bisimulations. *Theoretical Computer Science*, 138(2):353–389, February 1995.
- [Lin03] Huimin Lin. Complete Inference Systems for Weak Bisimulation Equivalences in the π -Calculus. *Information and Computation*, 180(1):1–29, January 2003.
- [LZ05] Cosimo Laneve and Gianluigi Zavattaro. Foundations of Web Transactions. In *Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science, 2005. To appear.
- [MP98] Ugo Montanari and Marco Pistore. History Dependent Automata. Technical report, Computer Science Department, Università di Pisa, 1998. TR-11-98.
- [MP00] Ugo Montanari and Marco Pistore. π -Calculus, Structured Coalgebras, and Minimal HD-Automata. In Mogens Nielsen and Branislav Roman, editors, *Mathematical Foundations of Computer Science*, volume 1983 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000. An extended version will be published on Theoretical Computer Science.
- [Pis99] Marco Pistore. *History Dependent Automata*. PhD thesis, Computer Science Department, Università di Pisa, 1999.

- [PS96] Marco Pistore and Davide Sangiorgi. A Partition Refinement Algorithm for the π -Calculus. In Rajeev Alur, editor, *Proceedings of CAV '96*, volume 1102 of *Lecture Notes in Computer Science*, pages 38–49, 1996.
- [PS01] Marco Pistore and Davide Sangiorgi. A Partition Refinement Algorithm for the π -Calculus. *Information and Computation*, 164(2):467–509, 2001.
- [San96] Davide Sangiorgi. A Theory of Bisimulation for the π -Calculus. *Acta Informatica*, 33(1):69–97, 1996.