Rule Formats for Nominal Operational Semantics

A very short and informal introduction

Luca Aceto
Gran Sasso Science Institute, L'Aquila, and
ICE-TCS, School of Computer Science, Reykjavik University

TLT — Types and Logic in Torino, 22 September 2017

Three academic leaders in Turin: The first 70 years

One joint paper (according to DBLP)

Mario Coppo, Mariangiola Dezani-Ciancaglini, Simona Ronchi Della Rocca: (Semi)-separability of finite sets of terms in Scott's D_∞-models of the lambda-calculus. ICALP 1978:142–164, but...

One joint paper (according to DBLP)

Mario Coppo, Mariangiola Dezani-Ciancaglini, Simona Ronchi Della Rocca: (Semi)-separability of finite sets of terms in Scott's D_∞-models of the lambda-calculus. ICALP 1978:142–164, but…

 $\overset{6}{\longleftrightarrow}$  $\overset{33}{\longleftrightarrow}$ 

Message of the talk

- Computer scientists have developed a theory of 'names' in order to understand what 'names' really matter in the behaviour of computer programs.
- Our goal: A general framework for defining operational semantics for 'nominal' calculi such as the λ- and π-calculi.

What's in a name? (Juliet's view)

In natural languages we use names to refer to persons, animals, places or things, amongst others.

Question: When does the meaning of what we say depend on the names we use?

What's in a name? (Juliet's view)

In natural languages we use names to refer to persons, animals, places or things, amongst others.

Question: When does the meaning of what we say depend on the names we use?

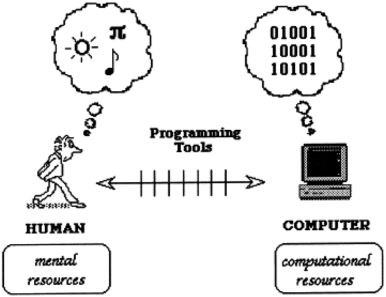
Juliet's view

"What's in a name? That which we call a rose
By any other name would smell as sweet."
(William Shakespeare, Romeo and Juliet (II, ii, 1–2))

Rest of the talk:

- Was Juliet right?
- What does this have to do with computer science?
- Some technicalities, alas.
- Three research leaders: The next 70 years.

The importance of languages in Computer Science



Fact of (Computer Science) Life

In Computer Science, we use programming languages to communicate with machines.

An important question: Programs use 'names' to describe the things they manipulate and their parts. On what names does the behaviour of a program depend?

On what names does the 'program' below depend?

 go home.

What happens if we swap ET with another name?

Names on which programs depend

On what names does the 'program' below depend?

 go home.

What happens if we swap ET with another name?

 go home.

Key insight (Gabbay and Pitts)

A program depends on a name if swapping that name for another changes its behaviour!

Names on which programs depend

On what names does the 'program' below depend?

 go home.

What happens if we swap ET with another name?

 go home.

Key insight (Gabbay and Pitts)

A program depends on a name if swapping that name for another changes its behaviour!

 is in the support of the program

 go home.

Example: Protocol for electronic voting

To any observer

 voted for 

should be the same as

 voted for 

 and  are not in the support of the relevant programs and can be swapped one for the other!

Back to Shakespeare

Was Juliet right?

Juliet's view (before swapping)

"What's in a name? That which

we call a 

By any other name would smell as sweet."

(Romeo and Juliet (II, ii, 1–2))

What do you think?

Back to Shakespeare

Was Juliet right?

Juliet's view (before swapping)

"What's in a name? That which

we call a 
By any other name would smell as sweet."
(Romeo and Juliet (II, ii, 1–2))

Juliet's view (after swapping rose with hákarl)

"What's in a name? That which

we call a 
By any other name would smell as sweet."
(Romeo and Juliet (II, ii, 1–2))

What do you think?

Nominal sets (semi-formally)

Countably many atoms $a, b, \ldots \in \mathbb{A}$.

Nominal sets (semi-formally)

Countably many atoms $a, b, \ldots \in \mathbb{A}$.

Finite permutations of atoms $\pi = (a_1\ b_1) \circ \ldots \circ (a_k\ b_k) \in \mathrm{Perm}\ \mathbb{A}$.

Nominal sets (semi-formally)

Countably many atoms $a, b, \ldots \in \mathbb{A}$.

Finite permutations of atoms $\pi = (a_1\ b_1) \circ \ldots \circ (a_k\ b_k) \in \mathrm{Perm}\ \mathbb{A}$.

$(\nu b)(\overline{a}b.0)$

Nominal sets (semi-formally)

Countably many atoms $a, b, \ldots \in \mathbb{A}$.

Finite permutations of atoms $\pi = (a_1\ b_1) \circ \ldots \circ (a_k\ b_k) \in \mathrm{Perm}\ \mathbb{A}$.

$(\nu b)(\overline{a}b.0)$

Nominal sets (semi-formally)

Countably many atoms $a, b, \ldots \in \mathbb{A}$.

Finite permutations of atoms $\pi = (a_1\ b_1) \circ \ldots \circ (a_k\ b_k) \in \mathrm{Perm}\ \mathbb{A}$.

$(b\ c) \cdot (\nu b)(\overline{a}b.0)$

Nominal sets (semi-formally)

Countably many atoms $a, b, \ldots \in \mathbb{A}$.

Finite permutations of atoms $\pi = (a_1\ b_1) \circ \ldots \circ (a_k\ b_k) \in \mathrm{Perm}\ \mathbb{A}$.

$(\nu c)(\overline{a}c.0)$

Nominal sets (semi-formally)

Countably many atoms $a, b, \ldots \in \mathbb{A}$.

Finite permutations of atoms $\pi = (a_1\ b_1) \circ \ldots \circ (a_k\ b_k) \in \mathrm{Perm}\ \mathbb{A}$.

$$(\nu c)(\overline{a}c.0) =_\alpha (\nu b)(\overline{a}b.0)$$

Nominal sets (semi-formally)

Countably many atoms $a, b, \ldots \in \mathbb{A}$.

Finite permutations of atoms $\pi = (a_1\ b_1) \circ \ldots \circ (a_k\ b_k) \in \mathrm{Perm}\ \mathbb{A}$.

$(\nu b)(\overline{a}b.0)$

Nominal sets (semi-formally)

Countably many atoms $a, b, \ldots \in \mathbb{A}$.

Finite permutations of atoms $\pi = (a_1\ b_1) \circ \ldots \circ (a_k\ b_k) \in \mathrm{Perm}\ \mathbb{A}$.

$(\nu b)(\overline{a}b.0)$

Nominal sets (semi-formally)

Countably many atoms $a, b, \ldots \in \mathbb{A}$.

Finite permutations of atoms $\pi = (a_1\ b_1) \circ \ldots \circ (a_k\ b_k) \in \mathrm{Perm}\ \mathbb{A}$.

$(a\ c) \cdot (\nu b)(\overline{a}b.0)$

Nominal sets (semi-formally)

Countably many atoms $a, b, \ldots \in \mathbb{A}$.

Finite permutations of atoms $\pi = (a_1\ b_1) \circ \ldots \circ (a_k\ b_k) \in \mathrm{Perm}\ \mathbb{A}$.

$(\nu b)(\overline{c}b.0)$

Nominal sets (semi-formally)

Countably many atoms $a, b, \ldots \in \mathbb{A}$.

Finite permutations of atoms $\pi = (a_1\ b_1) \circ \ldots \circ (a_k\ b_k) \in \mathrm{Perm}\ \mathbb{A}$.

$$(\nu b)(\overline{c}b.0) \neq_\alpha (\nu b)(\overline{a}b.0)$$

Nominal sets (semi-formally)

Countably many atoms $a, b, \ldots \in \mathbb{A}$.

Finite permutations of atoms $\pi = (a_1\ b_1) \circ \ldots \circ (a_k\ b_k) \in \mathrm{Perm}\ \mathbb{A}$.

$$(\nu b)(\overline{c}b.0) \neq_\alpha (\nu b)(\overline{a}b.0)$$

$$a \in \mathrm{supp}((\nu b)(\overline{a}b.0))$$

Nominal sets (semi-formally)

Countably many atoms $a, b, \ldots \in \mathbb{A}$.

Finite permutations of atoms $\pi = (a_1\ b_1) \circ \ldots \circ (a_k\ b_k) \in \mathrm{Perm}\ \mathbb{A}$.

$$(\nu b)(\overline{c}b.0) \neq_\alpha (\nu b)(\overline{a}b.0)$$

$$a \in \mathrm{supp}((\nu b)(\overline{a}b.0))$$

$b \# (\nu b)(\overline{a}b.0)$

Nominal sets (semi-formally)

Countably many atoms $a, b, \ldots \in \mathbb{A}$.

Finite permutations of atoms $\pi = (a_1\ b_1) \circ \ldots \circ (a_k\ b_k) \in \mathrm{Perm}\ \mathbb{A}$.

$$(\nu b)(\overline{c}b.0) \neq_\alpha (\nu b)(\overline{a}b.0)$$

$$a \in \mathrm{supp}((\nu b)(\overline{a}b.0))$$

$$b \# (\nu b)(\overline{a}b.0) \quad \text{iff} \quad b \notin \mathrm{supp}((\nu b)(\overline{a}b.0))$$

Nominal set

A *nominal set* S is a '$\mathrm{Perm}\ \mathbb{A}$-set' whose elements have finite support (set of names that matter to the element).

Examples of nominal sets

Some nominal sets

1. $\mathbb{A}$;
2. $\mathrm{Perm}\ \mathbb{A}$;
3. The set of raw lambda terms;
4. The set of lambda terms—that is, α-equivalence classes of raw lambda terms;
5. The set of finitely supported functions between nominal sets.

Examples of nominal sets

Some nominal sets

1. $\mathbb{A}$;
2. $\mathrm{Perm}\ \mathbb{A}$;
3. The set of raw lambda terms;
4. The set of lambda terms—that is, α-equivalence classes of raw lambda terms;
5. The set of finitely supported functions between nominal sets.

Equivariant function

A function f between nominal sets is equivariant if $\pi \cdot (f(s)) = f(\pi \cdot s)$ for every $\pi \in \mathrm{Perm}\ \mathbb{A}$. (Ditto for relations.)

SOS vs. nominal SOS theory

Classic SOS theory

1. Syntax: A (many-sorted) signature Σ.
2. Language: The algebra of Σ-terms.
3. Specification: Set of inference rules.
4. Target semantic object: Labelled transition system (LTS).

SOS vs. nominal SOS theory

Classic SOS theory

1. Syntax: A (many-sorted) signature Σ.
2. Language: The algebra of Σ-terms.
3. Specification: Set of inference rules.
4. Target semantic object: Labelled transition system (LTS).

Nominal SOS theory

1. Syntax: A many-sorted nominal signature Σ, including $[a]\_$. Function symbols produce terms of some base sort.

SOS vs. nominal SOS theory

Classic SOS theory

1. Syntax: A (many-sorted) signature Σ.
2. Language: The algebra of Σ-terms.
3. Specification: Set of inference rules.
4. Target semantic object: Labelled transition system (LTS).

Nominal SOS theory

1. Syntax: A many-sorted nominal signature Σ, including $[a]\_$. Function symbols produce terms of some base sort.
2. Language: The initial Σ-structure, whose elements capture α-equivalence classes of Σ-terms.

SOS vs. nominal SOS theory

Classic SOS theory

1. Syntax: A (many-sorted) signature Σ.
2. Language: The algebra of Σ-terms.
3. Specification: Set of inference rules.
4. Target semantic object: Labelled transition system (LTS).

Nominal SOS theory

1. Syntax: A many-sorted nominal signature Σ, including $[a]\_$. Function symbols produce terms of some base sort.
2. Language: The initial Σ-structure, whose elements capture α-equivalence classes of Σ-terms.
3. Specification: A nominal set of inference rules.

SOS vs. nominal SOS theory

Classic SOS theory

1. Syntax: A (many-sorted) signature Σ.
2. Language: The algebra of Σ-terms.
3. Specification: Set of inference rules.
4. Target semantic object: Labelled transition system (LTS).

Nominal SOS theory

1. Syntax: A many-sorted nominal signature Σ, including $[a]\_$. Function symbols produce terms of some base sort.
2. Language: The initial Σ-structure, whose elements capture α-equivalence classes of Σ-terms.
3. Specification: A nominal set of inference rules.
4. Target semantic object: Nominal transition system.

What is a nominal transition system?

Semantics of a nominal SOS

Nominal transition system [Parrow et al., CONCUR'15]

Quadruple $(S, Act, \mathrm{bn}, \rightarrow)$ where

(i) S and Act are nominal sets of states and actions respectively,

(ii) $\rightarrow \subseteq S \times (Act \times S)$ is an equivariant binary transition relation from states to residuals,

Nominal transition system [Parrow et al., CONCUR'15]

Quadruple $(S, Act, \mathrm{bn}, \rightarrow)$ where

(i) S and Act are nominal sets of states and actions respectively,

(ii) $\rightarrow \subseteq S \times (Act \times S)$ is an equivariant binary transition relation from states to residuals,

(iii) $\mathrm{bn} : Act \rightarrow \mathcal{P}_\omega(\mathbb{A})$ is an equivariant function from actions to finite sets of binding names, and

(iv) $\rightarrow$ satisfies alpha-conversion of residuals:
If $p \xrightarrow{\ell} p'$, $b \in \mathrm{bn}(\ell)$ and c fresh in (ℓ, p') then $p \xrightarrow{(b\,c)\cdot\ell} (b\,c) \cdot p'$.

Nominal transition system [Parrow et al., CONCUR'15]

Quadruple $(S, Act, \mathrm{bn}, \rightarrow)$ where

(i) S and Act are nominal sets of states and actions respectively,

(ii) $\rightarrow \subseteq S \times (Act \times S)$ is an equivariant binary transition relation from states to residuals,

(iii) $\mathrm{bn} : Act \rightarrow \mathcal{P}_\omega(\mathbb{A})$ is an equivariant function from actions to finite sets of binding names, and

(iv) $\rightarrow$ satisfies alpha-conversion of residuals:
If $p \xrightarrow{\ell} p'$, $b \in \mathrm{bn}(\ell)$ and c fresh in (ℓ, p') then $p \xrightarrow{(b\,c)\cdot\ell} (b\,c) \cdot p'$.

$$(\nu b)(\overline{a}b.p) \xrightarrow{\overline{a}(\nu b)} p$$

Nominal transition system [Parrow et al., CONCUR'15]

Quadruple $(S, Act, \mathrm{bn}, \rightarrow)$ where

(i) S and Act are nominal sets of states and actions respectively,

(ii) $\rightarrow \subseteq S \times (Act \times S)$ is an equivariant binary transition relation from states to residuals,

(iii) $\mathrm{bn} : Act \rightarrow \mathcal{P}_\omega(\mathbb{A})$ is an equivariant function from actions to finite sets of binding names, and

(iv) $\rightarrow$ satisfies alpha-conversion of residuals:
If $p \xrightarrow{\ell} p'$, $b \in \mathrm{bn}(\ell)$ and c fresh in (ℓ, p') then $p \xrightarrow{(b\,c)\cdot\ell} (b\,c) \cdot p'$.

$$(\nu b)(\overline{a}b.p) \xrightarrow{\overline{a}(\nu b)} p$$

Nominal transition system [Parrow et al., CONCUR'15]

Quadruple $(S, \mathit{Act}, \mathrm{bn}, \rightarrow)$ where

(i) S and Act are nominal sets of states and actions respectively,

(ii) $\rightarrow \subseteq S \times (\mathit{Act} \times S)$ is an equivariant binary transition relation from states to residuals,

(iii) $\mathrm{bn} : \mathit{Act} \rightarrow \mathcal{P}_\omega(\mathbb{A})$ is an equivariant function from actions to finite sets of binding names, and

(iv) $\rightarrow$ satisfies alpha-conversion of residuals:
If $p \xrightarrow{\ell} p'$, $b \in \mathrm{bn}(\ell)$ and c fresh in (ℓ, p') then $p \xrightarrow{(b\,c)\cdot\ell} (b\,c)\cdot p'$.

$$(\nu b)(\overline{a}b.p) \xrightarrow{\overline{a}(\nu c)} (b\,c)\cdot p$$

Syntactic ingredients in a nominal SOS specification

Raw terms and their nominal sorts (used in SOS rules)

$t_\sigma ::= x_\sigma \mid a_\alpha \mid (\pi \bullet t_\sigma)_\sigma \mid ([a_\alpha]t_\sigma)_{[\alpha]\sigma} \mid (t_{\sigma_1}, \ldots, t_{\sigma_k})_{\sigma_1 \times \ldots \times \sigma_k} \mid$
$(f(t_\sigma))_\delta \quad (f \in \Sigma,\ \delta \text{ a base sort})$

Syntactic ingredients in a nominal SOS specification

Raw terms and their nominal sorts (used in SOS rules)

$t_\sigma ::= x_\sigma \mid a_\alpha \mid (\pi \bullet t_\sigma)_\sigma \mid ([a_\alpha]t_\sigma)_{[\alpha]\sigma} \mid (t_{\sigma_1}, \ldots, t_{\sigma_k})_{\sigma_1 \times \ldots \times \sigma_k} \mid$
$(f(t_\sigma))_\delta \quad (f \in \Sigma,\ \delta$ a base sort$)$

Nominal terms

- The nominal term $NT[\![p]\!]$ is the interpretation of ground term p in the initial 'Σ-structure'.
- $NT[\![p]\!] \approx$ representation of the α-equivalence class of p.
- Nominal terms are states of the nominal transition system defined by a nominal SOS specification.

Interpretations of ground terms in NT coincide with the nominal algebraic datatypes of Pitts.

Example: The π-calculus (in nominal form)

Base sorts $\Delta = \{\mathsf{pr}, \mathsf{ac}\}$, atom sorts $A = \{\mathsf{ch}\}$ and

$$\begin{aligned}\Sigma = \{\ & null : \mathbf{1} \longrightarrow \mathsf{pr}, \\ & tau : \mathsf{pr} \longrightarrow \mathsf{pr}, \\ & in : (\mathsf{ch} \times [\mathsf{ch}]\mathsf{pr}) \longrightarrow \mathsf{pr}, \\ & out : (\mathsf{ch} \times \mathsf{ch} \times \mathsf{pr}) \longrightarrow \mathsf{pr}, \\ & par : (\mathsf{pr} \times \mathsf{pr}) \longrightarrow \mathsf{pr}, \\ & sum : (\mathsf{pr} \times \mathsf{pr}) \longrightarrow \mathsf{pr}, \\ & rep : \mathsf{pr} \longrightarrow \mathsf{pr}, \\ & new : [\mathsf{ch}]\mathsf{pr} \longrightarrow \mathsf{pr}, \\ & tauA : \mathbf{1} \longrightarrow \mathsf{ac}, \\ & inA : (\mathsf{ch} \times \mathsf{ch}) \longrightarrow \mathsf{ac}, \\ & outA : (\mathsf{ch} \times \mathsf{ch}) \longrightarrow \mathsf{ac}, \\ & boutA : (\mathsf{ch} \times \mathsf{ch}) \longrightarrow \mathsf{ac} \quad \}.\end{aligned}$$

$$NT[\![new([\mathsf{b}](out(\mathsf{a}, \mathsf{b}, null)))]\!] \longrightarrow NT[\![(boutA(\mathsf{a}, \mathsf{b}), null)]\!]$$

stands for $\qquad (\nu b)(\overline{a}b.\mathbf{0}) \stackrel{\overline{a}(\nu b)}{\longrightarrow} \mathbf{0}$

Nominal rules

$$\frac{\{u_i \rightarrow u_i' \mid i \in I\} \quad \{a_j \not\#\, v_j \mid j \in J\}}{t \rightarrow t'} \quad \text{Ru}$$

where $\{u_i \rightarrow u_i' \mid i \in I\}$ is finitely supported and J is finite.

Nominal rules

$$\frac{\{u_i \longrightarrow u_i' \mid i \in I\} \quad \{a_j \not\approx v_j \mid j \in J\}}{t \longrightarrow t'} \quad \text{Ru}$$

where $\{u_i \longrightarrow u_i' \mid i \in I\}$ is finitely supported and J is finite.

Key ideas

Let φ be a ground substitution mapping variables to raw terms. In proofs of transitions,

- $t \longrightarrow t'$ will be instantiated to $NT[\![\varphi(t)]\!] \longrightarrow NT[\![\varphi(t')]\!]$ and
- $a \not\approx t$ is satisfied by φ if $a\#NT[\![\varphi(t)]\!]$.

This is in agreement with standard conventions in nominal calculi.

Nominal rules

$$\frac{\{u_i \rightarrow u_i' \mid i \in I\} \quad \{a_j \not\# v_j \mid j \in J\}}{t \rightarrow t'} \quad \text{Ru}$$

where $\{u_i \rightarrow u_i' \mid i \in I\}$ is finitely supported and J is finite.

Key ideas

Let φ be a ground substitution mapping variables to raw terms. In proofs of transitions,

- $t \rightarrow t'$ will be instantiated to $NT[\![\varphi(t)]\!] \rightarrow NT[\![\varphi(t')]\!]$ and
- $a \not\# t$ is satisfied by φ if $a \# NT[\![\varphi(t)]\!]$.

This is in agreement with standard conventions in nominal calculi.

Can we ensure syntactically that a set of rules induces a nominal transition system?

Let $\mathcal{R}$ be a specification system that induces a transition system $\mathcal{T}$. Is $\mathcal{T}$ equivariant?

Let $\mathcal{R}$ be a specification system that induces a transition system $\mathcal{T}$. Is $\mathcal{T}$ equivariant?

Equivariant format

$\mathcal{R}$ is in equivariant format iff the rule $(a\,b) \cdot \mathrm{R{\scriptsize U}}$ is in $\mathcal{R}$, for every rule $\mathrm{R{\scriptsize U}}$ in $\mathcal{R}$ and for each $a, b \in \mathbb{A}$.

Let bn be an equivariant binding-names function.
Does $\mathcal{T}$ satisfy alpha-conversion of residuals with respect to bn?

Ensuring equivariance

Let $\mathcal{R}$ be a specification system that induces a transition system $\mathcal{T}$. Is $\mathcal{T}$ equivariant?

Equivariant format

$\mathcal{R}$ is in equivariant format iff the rule $(a\,b) \cdot \mathrm{R{\scriptsize U}}$ is in $\mathcal{R}$, for every rule $\mathrm{R{\scriptsize U}}$ in $\mathcal{R}$ and for each $a, b \in \mathbb{A}$.

Theorem

If $\mathcal{R}$ is in equivariant format then $\mathcal{T}$ is equivariant.

Let bn be an equivariant binding-names function.

Does $\mathcal{T}$ satisfy alpha-conversion of residuals with respect to bn?

We have a grotty rule format for that. See CONCUR 2017 paper.

Let $\mathcal{R}$ be a specification system that induces a transition system $\mathcal{T}$. Is $\mathcal{T}$ equivariant?

Equivariant format

$\mathcal{R}$ is in equivariant format iff the rule $(a\,b) \cdot \mathrm{R}\textsc{u}$ is in $\mathcal{R}$, for every rule $\mathrm{R}\textsc{u}$ in $\mathcal{R}$ and for each $a, b \in \mathbb{A}$.

Theorem

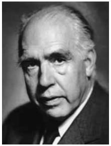
If $\mathcal{R}$ is in equivariant format then $\mathcal{T}$ is equivariant.

Let bn be an equivariant binding-names function.
Does $\mathcal{T}$ satisfy alpha-conversion of residuals with respect to bn?
We have a grotty rule format for that. See CONCUR 2017 paper.

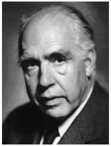
Summary

- Framework for SOS of languages with binding operation:
  - Raw terms (not up to alpha-equivalence) for specifications.
  - Nominal terms (up to alpha-equivalence) for proof trees.
- Rule format for equivariance (pleasing).
- ACR format, which ensures that a specification system together with a function bn induces a nominal TS (not given and not so pleasing yet, alas).

"Det er svært at spå — især om fremtiden."



"Det er svært at spå — især om fremtiden."



I trust that CS in Turin will be invariant under name permutations!

The message (reprise)

Name independence in computer science can be expressed in terms of invariance under swapping of names both syntactically and semantically. There is much left to do.

Want to know more? Ask the team of researchers at RU and IMDEA working on Nominal Structural Operational Semantics.

Thank you!

Umberto Eco (1932–2016)