A Formal Calculus for Informal Equality with Binding

Aad Mathijssen

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven

Joint work with Murdoch J. Gabbay

WoLLIC'2007
14th Workshop on Logic, Language, Information and Computation
Rio de Janeiro, Brazil

2nd July 2007

Motivation

The λ-calculus

The λ-calculus:

$$t \quad ::= \quad x \mid tt \mid \lambda x.t$$

Axioms:

$$\begin{array}{lll} (\alpha) & \lambda x.t = \lambda y.(t[x \mapsto y]) & \text{if } y \notin fv(t) \\ (\beta) & (\lambda x.t)u = t[x \mapsto u] & \\ (\eta) & \lambda x.(tx) = t & \text{if } x \notin fv(t) \end{array}$$

Free variables function fv:

$$fv(x) = \{x\} \qquad fv(tu) = fv(t) \cup fv(u) \qquad fv(\lambda x.t) = fv(t) \backslash \{x\}$$

Motivation

The λ-calculus

The λ-calculus:

$$t \quad ::= \quad x \mid tt \mid \lambda x.t$$

Axiom schemata:

$$\begin{array}{llll}
(\alpha) & \lambda x.t & = \lambda y.(t[x \mapsto y]) & \text{if } y \notin fv(t) \\
(\beta) & (\lambda x.t)u & = t[x \mapsto u] & \\
(\eta) & \lambda x.(tx) & = t & \text{if } x \notin fv(t)
\end{array}$$

Free variables function fv:

$$fv(x) = \{x\} \qquad fv(tu) = fv(t) \cup fv(u) \qquad fv(\lambda x.t) = fv(t) \setminus \{x\}$$

t and u are meta-variables ranging over terms.

Motivation

The λ-calculus

The λ-calculus with meta-variables:

$$t \quad ::= \quad x \mid tt \mid \lambda x.t \mid X$$

Axioms:

$$\begin{array}{llll}
(\alpha) & \lambda x.X & = \lambda y.(X[x \mapsto y]) & \text{if } y \notin fv(X) \\
(\beta) & (\lambda x.X)Y & = X[x \mapsto Y] & \\
(\eta) & \lambda x.(Xx) & = X & \text{if } x \notin fv(X)
\end{array}$$

Free variables function fv:

$$fv(x) = \{x\} \qquad fv(tu) = fv(t) \cup fv(u) \qquad fv(\lambda x.t) = fv(t) \backslash \{x\}$$

Motivation
The λ-calculus

The λ-calculus with meta-variables:

$$t \quad ::= \quad x \mid tt \mid \lambda x.t \mid X$$

Axioms:

$$\begin{array}{llll}
(\alpha) & \lambda x.X & = \lambda y.(X[x \mapsto y]) & \text{if } y \notin fv(X) \\
(\beta) & (\lambda x.X)Y & = X[x \mapsto Y] & \\
(\eta) & \lambda x.(Xx) & = X & \text{if } x \notin fv(X)
\end{array}$$

Free variables function fv:

$$fv(x) = \{x\} \qquad fv(tu) = fv(t) \cup fv(u) \qquad fv(\lambda x.t) = fv(t) \backslash \{x\}$$

Freshness occurs in the presence of meta-variables:
We only know if $x \notin fv(X)$ when X is instantiated.

Motivation

Other examples

In informal mathematical usage, we see equalities like:

- First-order logic: $(\forall x.\phi) \wedge \psi \quad = \forall x.(\phi \wedge \psi) \quad$ if $x \notin fv(\psi)$
- π-calculus: $(\nu x.P) \mid Q \quad = \nu x.(P \mid Q) \quad$ if $x \notin fv(Q)$
- μCRL/mCRL2: $\sum_x .p \quad = p \quad$ if $x \notin fv(p)$

And for any binder $\xi \in \{\lambda, \forall, \nu, \sum\}$:

- $(\xi x.t)[y \mapsto u] = \xi x.(t[y \mapsto u]) \quad$ if $x \notin fv(u)$
- α-equivalence: $\xi x.t \quad = \xi y.(t[x \mapsto y]) \quad$ if $y \notin fv(t)$

Motivation
Other examples

In informal mathematical usage, we see equalities like:

- First-order logic: $(\forall x.\phi) \wedge \psi \quad = \forall x.(\phi \wedge \psi) \quad$ if $x \notin fv(\psi)$
- π-calculus: $(\nu x.P) \mid Q \quad = \nu x.(P \mid Q) \quad$ if $x \notin fv(Q)$
- μCRL/mCRL2: $\sum_x .p \quad = p \quad$ if $x \notin fv(p)$

And for any binder $\xi \in \{\lambda, \forall, \nu, \sum\}$:

- $(\xi x.t)[y \mapsto u] = \xi x.(t[y \mapsto u]) \quad$ if $x \notin fv(u)$
- α-equivalence: $\xi x.t \quad = \xi y.(t[x \mapsto y]) \quad$ if $y \notin fv(t)$

Here:

- $\phi, \psi, P, Q, p, t, u$ are meta-variables ranging over terms.

Motivation
Other examples

In informal mathematical usage, we see equalities like:

- First-order logic: $(\forall x.\phi) \wedge \psi \quad = \forall x.(\phi \wedge \psi) \quad$ if $x \notin fv(\psi)$
- π-calculus: $(\nu x.P) \mid Q \quad = \nu x.(P \mid Q) \quad$ if $x \notin fv(Q)$
- μCRL/mCRL2: $\sum_x .p \quad = p \quad$ if $x \notin fv(p)$

And for any binder $\xi \in \{\lambda, \forall, \nu, \sum\}$:

- $(\xi x.t)[y \mapsto u] = \xi x.(t[y \mapsto u]) \quad$ if $x \notin fv(u)$
- α-equivalence: $\xi x.t \quad = \xi y.(t[x \mapsto y]) \quad$ if $y \notin fv(t)$

Here:

- $\phi, \psi, P, Q, p, t, u$ are meta-variables ranging over terms.
- Freshness occurs in the presence of meta-variables.

Motivation

Formalisation

Question: Can we formalise binding and freshness in the presence of meta-variables?

Motivation

Formalisation

Question: Can we formalise binding and freshness in the presence of meta-variables?

Answer: Yes, using Nominal Terms (Urban, Pitts & Gabbay, 2003)

Motivation

Formalisation

Question:	Can we formalise binding and freshness in the presence of meta-variables?
Answer:	Yes, using Nominal Terms (Urban, Pitts & Gabbay, 2003)

Question:	Can we formalise equality with binding in the presence of meta-variables?

Motivation
Formalisation

Question: Can we formalise binding and freshness in the presence of meta-variables?

Answer: Yes, using Nominal Terms (Urban, Pitts & Gabbay, 2003)

Question: Can we formalise equality with binding in the presence of meta-variables?

Answer: Yes, using Nominal Algebra...

Overview

Overview:

- Nominal terms
- Nominal algebra:
  - Definitions
  - Examples
- α-conversion
- Derivability of equality
- A semantics in nominal sets
- Related work
- Conclusions and future work

Nominal Terms
Definition

Nominal terms are inductively defined by:

$$t \quad ::= \quad a \mid X \mid [a]t \mid \mathsf{f}(t_1, \ldots, t_n)$$

Here we fix:

- atoms $a, b, c, \ldots$ (for x, y)
- unknowns $X, Y, Z, \ldots$ (for t, u, ϕ, ψ, P, Q, p)
- term-formers $\mathsf{f}, \mathsf{g}, \mathsf{h}, \ldots$ (for λ, $\_\,\_$, $\forall$, $\wedge$, ν, $|$, $\sum$, $\_[\_ \mapsto \_]$)

We call $[a]t$ an abstraction (for the $x.\_$).

Nominal Terms
Definition

Nominal terms are inductively defined by:

$$t \quad ::= \quad a \mid X \mid [a]t \mid \mathsf{f}(t_1, \ldots, t_n)$$

Here we fix:

- atoms $a, b, c, \ldots$ (for x, y)
- unknowns $X, Y, Z, \ldots$ (for t, u, ϕ, ψ, P, Q, p)
- term-formers $\mathsf{f}, \mathsf{g}, \mathsf{h}, \ldots$ (for λ, $\_\,\_$, $\forall$, $\wedge$, ν, $|$, $\sum$, $\_[\_ \mapsto \_]$)

We call $[a]t$ an abstraction (for the $x.\_$).

We can impose a sorting system on nominal terms …
but we don't do that here.

Nominal Terms

Examples

Representation of mathematical syntax in nominal terms:

mathematics	nominal terms	
	unsugared	sugared
$\lambda x.t$	$\lambda([a]X)$	$\lambda[a]X$
$\lambda x.(tx)$	$\lambda([a]\mathrm{app}(X, a))$	$\lambda[a](Xa)$
$(\forall x.\phi) \wedge \psi$	$\wedge(\forall([a]X), Y)$	$(\forall[a]X) \wedge Y$
$(\nu x.P) \mid Q$	$\mid(\nu([a]X), Y)$	$(\nu[a]X) \mid Y$
$\sum_x .p$	$\sum([a]X)$	$\sum[a]X$
$t[x \mapsto u]$	$\mathrm{sub}([a]X, Y)$	$X[a \mapsto Y]$

Nominal Terms

Freshness

Definition:

- Call $a\#X$ a primitive freshness (for '$x \notin fv(t)$').
- A freshness context Δ is a *finite set* of primitive freshnesses.

Nominal Terms
Freshness

Definition:

- Call $a\#X$ a primitive freshness (for '$x \notin fv(t)$').
- A freshness context Δ is a *finite set* of primitive freshnesses.

Generalise freshness on unknowns X to terms t:

- Call $a\#t$ a freshness, where t is a nominal term.
- Write $\Delta \vdash a\#t$ when $a\#t$ is derivable from Δ using

$$\frac{}{a\#b}(\#\mathbf{ab}) \qquad \frac{}{a\#[a]t}(\#[]\mathbf{a}) \qquad \frac{a\#t}{a\#[b]t}(\#[]\mathbf{b}) \qquad \frac{a\#t_1 \quad \cdots \quad a\#t_n}{a\#\mathsf{f}(t_1,\ldots,t_n)}(\#\mathbf{f})$$

Examples:

$$\vdash a\#b \qquad \vdash a\#\lambda[a]X \qquad a\#X \vdash a\#\lambda[b]X$$

$$\nvdash a\#a \qquad \nvdash a\#\lambda[b]X \qquad a\#X \nvdash a\#Y$$

Nominal Algebra
Definition

Nominal algebra is a theory of equality between nominal terms:

- $t = u$ is an equality where t and u are nominal terms.
- $\Delta \vdash t = u$ is an equality-in-context
  (for '$t' = u'$ if $x_1 \notin fv(v_1), \ldots, x_n \notin fv(v_n)$').

Nominal Algebra

Example equalities-in-context

Meta-level properties as equalities-in-context in nominal algebra:

- λ-calculus: $a\#X \vdash \lambda[a](Xa) = X$
- First-order logic: $a\#Y \vdash (\forall[a]X) \wedge Y = \forall[a](X \wedge Y)$
- π-calculus: $a\#Y \vdash (\nu[a]X) \mid Y = \nu[a](X \mid Y)$
- μCRL/mCRL2: $a\#X \vdash \sum[a]X = X$

And for any binder $\xi \in \{\lambda, \forall, \nu, \sum\}$:

- $a\#Y \vdash (\xi[a]X)[b \mapsto Y] = \xi[a](X[b \mapsto Y])$
- α-equivalence: $b\#X \vdash \xi[a]X = \xi[b](X[a \mapsto b])$

Nominal algebra

Theories

A theory in nominal algebra consists of:

- a set of term-formers
- a set of axioms: equalities-in-context $\Delta \vdash t = u$

Nominal Algebra

LAM: the λ-calculus

A theory LAM for the λ-calculus with meta-variables:

- term-formers λ, app and sub
  (recall that $t[a \mapsto u]$ is just sugar for $\mathrm{sub}([a]t, u)$)
- axioms:

$$
\begin{array}{llcll}
(\alpha) & b\#X & \vdash & \lambda[a]X & = & \lambda[b](X[a \mapsto b]) \\
(\beta) & & \vdash & (\lambda[a]Y)X & = & Y[a \mapsto X] \\
(\eta) & a\#X & \vdash & \lambda[a](Xa) & = & X
\end{array}
$$

Nominal Algebra
FOL: first-order logic

A theory FOL for first-order logic with meta-variables, also called one-and-a-halfth-order logic:

- term-formers:
  - $\perp$, $\supset$, $\forall$, $\approx$ and sub for the basic operators ($\top$, $\neg$, $\wedge$, $\vee$, $\Leftrightarrow$, $\exists$ are sugar)
  - $p_1, \ldots, p_m$ and $f_1, \ldots, f_n$ for object-level predicates and terms
- axioms: ...

Nominal Algebra

Axioms of FOL

Axioms of one-and-a-halfth-order logic:

$$
\begin{array}{rl}
\textbf{(MP)} & \vdash \top \supset P \;=\; P \\
\textbf{(M)} & \vdash ((((P \supset Q) \supset (\neg R \supset \neg S)) \supset R) \supset T) \\
 & \qquad \supset ((T \supset P) \supset (S \supset P)) \quad = \top \\
\textbf{(Q1)} & \vdash \forall[a]P \supset P[a \mapsto T] \;=\; \top \\
\textbf{(Q2)} & \vdash \forall[a](P \wedge Q) \;=\; \forall[a]P \wedge \forall[a]Q \\
\textbf{(Q3)} & a\#P \;\vdash\; \forall[a](P \supset Q) \;=\; P \supset \forall[a]Q \\
\textbf{(E1)} & \vdash T \approx T \;=\; \top \\
\textbf{(E2)} & \vdash U \approx T \wedge P[a \mapsto T] \supset P[a \mapsto U] \;=\; \top
\end{array}
$$

Nominal Algebra

SUB: a theory of capture-avoiding substitution

A theory SUB for capture-avoiding substitution with meta-variables:

$$
\begin{array}{rrcl}
(\mathbf{var}{\mapsto}) & \vdash a[a \mapsto T] & = & T \\
(\#{\mapsto}) & a\#X \vdash X[a \mapsto T] & = & X \\
(\mathbf{f}{\mapsto}) & \vdash \mathsf{f}(X_1, \ldots, X_n)[a \mapsto T] & = & \mathsf{f}(X_1[a \mapsto T], \ldots, X_n[a \mapsto T]) \\
(\mathbf{abs}{\mapsto}) & b\#T \vdash ([b]X)[a \mapsto T] & = & [b](X[a \mapsto T])
\end{array}
$$

α-conversion
Problem

Formalising binding implies formalising α-conversion.

Idea: use theory SUB:

$$b\#X \vdash [a]X = [b](X[a \mapsto b])$$

α-conversion
Problem

Formalising binding implies formalising α-conversion.

Idea: use theory SUB:

$$b\#X \vdash [a]X = [b](X[a \mapsto b])$$

This destroys the proof theory:

- When proving properties by induction on the size of terms, you often want to freshen up a term using α-conversion.
- Freshening using the above α-conversion increases term size.

α-conversion
Problem

Formalising binding implies formalising α-conversion.

Idea: use theory SUB:

$$b\#X \vdash [a]X = [b](X[a \mapsto b])$$

This destroys the proof theory:

- When proving properties by induction on the size of terms, you often want to freshen up a term using α-conversion.
- Freshening using the above α-conversion increases term size.

Not all systems need substitution of terms for atoms, e.g. the π-calculus.

α-conversion
Solution

Solution: use permutations of atoms:

$$b\#X \vdash [a]X = [b]((a\ b) \cdot X)$$

α-conversion
Solution

Solution: use permutations of atoms:

$$b\#X \vdash [a]X = [b]((a\ b) \cdot X)$$

Redefine nominal terms:

$$t \quad ::= \quad a \mid \pi \cdot X \mid \mathsf{f}(t_1, \ldots, t_n) \mid [a]t$$

Here:

- we call $\pi \cdot X$ a moderated unknown
- write X when π is the trivial permutation **Id**
- instantiation of X to t in $\pi \cdot X$ gives us $\pi \cdot t$:

$$\pi \cdot a \equiv \pi(a) \qquad \pi \cdot (\pi' \cdot X) \equiv (\pi \circ \pi') \cdot X \qquad \pi \cdot [a]t \equiv [\pi(a)](\pi \cdot t)$$

$$\pi \cdot \mathsf{f}(t_1, \ldots, t_n) \equiv \mathsf{f}(\pi \cdot t_1, \ldots, \pi \cdot t_n)$$

α-conversion
Consequence

Add freshness derivation rule:

$$\frac{\pi^{-1}(a)\#X}{a\#\pi\cdot X}\ (\#\mathbf{X}) \quad (\pi \neq \mathbf{Id})$$

Redefine theory SUB for capture-avoiding substitution:

$$\begin{array}{rrcl}
(\mathbf{var}\mapsto) & \vdash a[a\mapsto T] & = & T \\
(\#\mapsto) & a\#X \vdash X[a\mapsto T] & = & X \\
(\mathbf{f}\mapsto) & \vdash \mathsf{f}(X_1,\ldots,X_n)[a\mapsto T] & = & \mathsf{f}(X_1[a\mapsto T],\ldots,X_n[a\mapsto T]) \\
(\mathbf{abs}\mapsto) & b\#T \vdash ([b]X)[a\mapsto T] & = & [b](X[a\mapsto T]) \\
(\mathbf{ren}\mapsto) & b\#X \vdash X[a\mapsto b] & = & (b\ a)\cdot X
\end{array}$$

Derivability of equalities

Definition

Write $\Delta \vdash_{\mathsf{T}} t = u$ when $t = u$ is derivable from the rules below, s.t.

- each axiom used in $(\mathbf{ax}_{\nabla \,\vdash\, t' = u'})$ is from theory T only
- only assumptions from Δ are used in freshness derivations

$$\frac{}{t = t}\,(\mathbf{refl}) \qquad \frac{t = u}{u = t}\,(\mathbf{symm}) \qquad \frac{t = u \quad u = v}{t = v}\,(\mathbf{tran}) \qquad \frac{a\#t \quad b\#t}{(a\ b)\cdot t = t}\,(\mathbf{perm})$$

$$\frac{t = u}{[a]t = [a]u}\,(\mathbf{cong}[]) \qquad \frac{t = u}{\mathsf{f}(t_1,\ldots,t,\ldots,t_n) = \mathsf{f}(t_1,\ldots,u,\ldots,t_n)}\,(\mathbf{cong}\mathsf{f})$$

$$\frac{\pi\cdot\nabla\sigma}{\pi\cdot t\sigma = \pi\cdot u\sigma}\,(\mathbf{ax}_{\nabla \,\vdash\, t = u}) \qquad \frac{\begin{matrix}[a\#X_1,\ldots,a\#X_n] \quad \Delta \\ \vdots \\ t = u\end{matrix}}{t = u}\,(\mathbf{fr}) \quad (a \notin t, u, \Delta)$$

Derivability of equalities

Instantiation of (β) in LAM

$$(\beta) \quad \vdash (\lambda[a]Y)X = Y[a \mapsto X]$$

Instantiation of the (β) axiom:

σ	π	Result
$[]$	**Id**	$\vdash (\lambda[a]Y)X = Y[a \mapsto X]$
$[b/Y, c/X]$	**Id**	$\vdash (\lambda[a]b)c = b[a \mapsto c]$
$[a/Y, c/X]$	**Id**	$\vdash (\lambda[a]a)c = a[a \mapsto c]$
$[a/Y, c/X]$	$(a\ b)$	$\vdash (\lambda[b]b)c = b[b \mapsto c]$
$[(\lambda[b]Z)Y/Y]$	**Id**	$\vdash (\lambda[a](\lambda[b]Z)Y)X = ((\lambda[b]Z)Y)[a \mapsto X]$

Derivability of equalities

Instantiation of (η) in LAM

$$(\eta) \quad a\#X \vdash \lambda[a](Xa) = X$$

Instantiation of the (η) axiom:

σ	π	Resulting equality-in-context
$[a/X]$	**Id**	none, since $\nvdash a\#a$
$[b/X]$	**Id**	$\vdash \lambda[a](ba) = b$
$[YZ/X]$	**Id**	$a\#Y, a\#Z \vdash \lambda[a]((YZ)a) = YZ$
$[\lambda[a]Y/X]$	**Id**	$\vdash \lambda[a]((\lambda[a]Y)a) = \lambda[a]Y$
$[\lambda[b]Y/X]$	**Id**	$a\#Y \vdash \lambda[a]((\lambda[b]Y)a) = \lambda[b]Y$

Derivability of equalities

Example derivation: representing a calculation

$$\lambda x.(((\lambda x.y)x)x) =_\beta \lambda x.(yx) =_\eta y$$

Formal derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(\lambda[a]b)a = b[a \mapsto a]}\,(\mathbf{ax}_\beta) \qquad \dfrac{\dfrac{}{a\#b}\,(\#\mathbf{ab})}{b[a \mapsto a] = b}\,(\mathbf{ax}_{\#\mapsto})}{(\lambda[a]b)a = b}\,(\mathbf{tran})}{((\lambda[a]b)a)a = ba}\,(\mathbf{congf})}{[a](((\lambda[a]b)a)a) = [a](ba)}\,(\mathbf{cong[]})}{\lambda[a](((\lambda[a]b)a)a) = \lambda[a](ba)}\,(\mathbf{congf})$$

$$\dfrac{\dfrac{}{a\#b}\,(\#\mathbf{ab})}{\lambda[a](ba) = b}\,(\mathbf{ax}_\eta)$$

$$\dfrac{\lambda[a](((\lambda[a]b)a)a) = \lambda[a](ba) \qquad \lambda[a](ba) = b}{\lambda[a](((\lambda[a]b)a)a) = b}\,(\mathbf{tran})$$

Derivability of equalities

Example derivation: the substitution lemma

$$a\#U \vdash_{\mathsf{SUB}} X[a \mapsto T][b \mapsto U] = X[b \mapsto U][a \mapsto T[b \mapsto U]]$$

Writing $\mathfrak{s}$ for $[b \mapsto U]$ and using the unsugared syntax for the other substitutions:

$$\dfrac{\dfrac{}{\mathsf{sub}([a]X, T)\mathfrak{s} = \mathsf{sub}(([a]X)\mathfrak{s}, T\mathfrak{s})}\,(\mathbf{ax}_{\mathsf{f}\mapsto}) \quad \dfrac{\dfrac{a\#U}{([a]X)\mathfrak{s} = [a](X\mathfrak{s})}\,(\mathbf{ax}_{\mathsf{abs}\mapsto})}{\mathsf{sub}(([a]X)\mathfrak{s}, T\mathfrak{s}) = \mathsf{sub}([a](X\mathfrak{s}), T\mathfrak{s})}\,(\mathbf{congf})}{\mathsf{sub}([a]X, T)\mathfrak{s} = \mathsf{sub}([a](X\mathfrak{s}), T\mathfrak{s})}\,(\mathbf{tran})$$

Derivability of equalities

Example derivation: introducing a fresh atom

$$\vdash_{\mathsf{SUB}} X[a \mapsto a] = X$$

Formal derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a\#[a]X}\,(\#[]\mathbf{a}) \qquad \dfrac{[b\#X]^1}{b\#[a]X}\,(\#[]\mathbf{b})}{[b](b\ a)\cdot X = [a]X}\,(\mathbf{perm})}{[a]X = [b](b\ a)\cdot X}\,(\mathbf{symm})}{X[a \mapsto a] = ((b\ a)\cdot X)[b \mapsto a]}\,(\mathbf{congf}) \qquad \dfrac{\dfrac{[b\#X]^1}{a\#(b\ a)\cdot X}\,(\#\mathbf{X})}{((b\ a)\cdot X)[b \mapsto a] = X}\,(\mathbf{ax}_{\mathbf{ren}\mapsto})}{\dfrac{X[a \mapsto a] = X}{X[a \mapsto a] = X}\,(\mathbf{fr})^1}\,(\mathbf{tran})$$

Derivability of equalities

Results for specific theories

Results on the CORE theory with no axioms:

- Syntactic criteria for deciding equality between terms
- Equivalent to α-equality in Nominal Unification and Rewriting

Results on theory SUB (other work):

- It is decidable whether $\Delta \vdash_{\mathsf{SUB}} t = u$
- Omega-complete: sound and complete w.r.t. the term model

Results on theory FOL (other work):

- has an equivalent sequent calculus:
  - representing schemas of derivations in first-order logic
  - satisfies cut-elimination
- equivalent to first-order logic for terms without unknowns

A semantics in nominal sets

Nominal sets (Gabbay & Pitts, 1999):

- A set-based model for names and binding
- Atoms are built-in
- Support for binding and freshness
- Inspired the development of nominal terms

Nominal algebra theories have a semantics in nominal sets:

- Derivability of equality is sound and complete
- Derivability of freshness is sound but incomplete
- Semantic freshness can be expressed using equalities

Related work
Nominal Equational Logic

Closely related to Nominal Algebra:

- Nominal Equational Logic (NEL) by Pitts and Clouston

Derivability of freshness is semantic and not syntactic:

- In NEL freshness derivability is complete
- Potentially undecidable
- Expressing syntactic freshness is impossible:

  $x \notin fv(t)$ does not correspond to $\vdash a \# t'$

Related work
Non-nominal approaches

Other related work:

- Higher-Order Algebra (HOA)
- Cylindric Algebra and Lambda-Abstraction Algebra (CA/LAA)

These do not mirror informal equality like nominal algebra does:

- Binding and freshness are encoded:
  - by higher-order functions in HOA
  - by replacing t by $c_i t$ to ensure $x_i \notin fv(t)$ in CA/LAA
- Reasoning about binding becomes different.
- Capturing substitution cannot be defined HOA.
  Default notion of (meta-level) substitution in nominal algebra.

Conclusions

Nominal algebra:

- is a theory of algebraic equality on nominal terms
- allows us to reason about systems with binding
- closely mirrors informal mathematical usage:
  - existing axiom schemata can be expressed directly
  - equational proofs carry over directly
  - natural notion of instantiation of meta-variables:

    informal notation: instantiating t to x in $\lambda x.t$ yields $\lambda x.x$
    nominal terms: instantiating X to a in $\lambda[a]X$ yields $\lambda[a]a$
  - α-equivalence in the presence of meta-variables
  - introduce fresh atoms inside a derivation

Future work

Future work on nominal algebra:

- further develop theory on:
  - the λ-calculus
  - choice quantification in μCRL/mCRL2
  - π-calculus and its variants
  - reversibility
- formalise meta-level reasoning, meta-meta-level reasoning,...
  a hierarchy of variables.
- develop a theorem prover

Further reading

- Murdoch J. Gabbay, Aad Mathijssen:
  Capture-Avoiding Substitution as a Nominal Algebra.
  ICTAC'06.
- Murdoch J. Gabbay, Aad Mathijssen:
  One-and-a-halfth-order Logic.
  PPDP'06.

Papers and slides of talks can be found on my web page:
http://www.win.tue.nl/~amathijs