An Intersection Type System for Nominal Terms

Mauricio Ayala-Rincón
Maribel Fernández
Ana Cristina Rocha-Oliveira

Grupo de Teoria da Computação - UnB - Brasília - Brasil
Department of Informatics - KCL - London - UK

Work supported by CAPES/Brazil

29th January, 2016
Seminário Informal (, mas Formal!), Brasília/DF, Brasil
1 Motivation

2 Nominal Syntax

3 Intersection Types for Nominal Terms

4 Conclusion and Future Work
What is nominal good for?

- Deal with binders in an elegant way.
- Built in $\alpha$-equivalence.
- First-order substitutions.
- Decidable and efficient unification/matching.
- Frameworks based on nominal setting: $\alpha$-Prolog, Fresh ML, Calm...
Specifying binding operations:

- Explicit substitutions:

\[ M\{x \mapsto N\} \rightarrow M \quad (x \notin \text{fv}(M)) \]
Specifying binding operations:

- Explicit substitutions:

\[ M\{x \mapsto N\} \rightarrow M \quad (x \notin \text{fv}(M)) \]

- \(\pi\)-calculus:

\[ P \mid v a. Q \rightarrow v a.(P \mid Q) \quad (a \notin \text{fv}(P)) \]
Specifying binding operations:

- ** Explicit substitutions: 

\[ M\{x \mapsto N\} \rightarrow M \quad (x \notin \text{fv}(M)) \]

- ** π-calculus: 

\[ P \mid \nu a.Q \rightarrow \nu a.(P \mid Q) \quad (a \notin \text{fv}(P)) \]

- ** Logic: 

\[ P \text{ and } (\forall x.Q) \rightarrow \forall x.(P \text{ and } Q) \quad (x \notin \text{fv}(P)) \]
Motivation  Nominal Syntax  Intersection Types for Nominal Terms  Conclusion and Future Work

\[ t ::= a \mid \pi \cdot X \mid [a]s \mid f(t_1, \ldots, t_n) \]

\(\alpha\)-equivalence deduction rules:

\[
\begin{align*}
\Delta \vdash a \equiv_\alpha a & \quad (\equiv_\alpha a) \\
\Delta \vdash \pi \cdot X \equiv_\alpha \pi' \cdot X & \quad (\equiv_\alpha X) \\
\Delta \vdash s_1 \equiv_\alpha t_1 \ldots \Delta \vdash s_n \equiv_\alpha t_n & \\
\Delta \vdash f(s_1, \ldots, s_n) \equiv_\alpha f(t_1, \ldots, t_n) & \quad (\equiv_\alpha f) \\
\Delta \vdash s \equiv_\alpha t & \quad (\equiv_\alpha \text{absa}) \\
\Delta \vdash [a]s \equiv_\alpha [a]t & \\
\Delta \vdash a \# t & \quad (\equiv_\alpha \text{absb}) \\
\end{align*}
\]

where \(\Delta = \{a_1\#X_1, a_2\#X_2, \ldots, a_n\#X_n\}\).
Freshness deduction rules:

\[
\begin{align*}
\nabla \vdash a \# b & \quad \text{(\#ab)} \\
\nabla \vdash a \# () & \quad \text{(\#unit)} \\
\n\nabla \vdash a \# s_1 & \quad \nabla \vdash a \# s_2 & \quad \nabla \vdash a \# (s_1, s_2) & \quad \text{(\#pair)} \\
\n\nabla \vdash a \# [a] s & \quad \text{(\#absa)} \\
\nabla \vdash a \# [b] s & \quad \text{(\#absb)} \end{align*}
\]
Nominal Rewriting Systems

\[ \nabla \vdash l \rightarrow r \quad \text{Vars}(r, \nabla) \subseteq \text{Vars}(l) \]

(Beta): \[ \vdash (\lambda [a]X)X' \rightarrow X\{a \mapsto X'\} \]

(\sigma_{\text{app}}): \[ \vdash (X X')\{a \mapsto Y\} \rightarrow X\{a \mapsto Y\} X'\{a \mapsto Y\} \]

(\sigma_{\text{var}}): \[ \vdash a\{a \mapsto X\} \rightarrow X \]

(\sigma_{\epsilon}): a \# Y \vdash Y\{a \mapsto X\} \rightarrow Y

(\sigma_{\text{lam}}): b \# Y \vdash (\lambda [b]X)\{a \mapsto Y\} \rightarrow \lambda [b](X\{a \mapsto Y\})

Notation: \[ X\{a \mapsto Y\} = \text{subst}([a]X, Y). \]
Unexpected problems:

Confluence of orthogonal NRSs does not hold in general.
Unexpected problems:

Confluence of orthogonal NRSs does not hold in general.

Example: \( R = \vdash f(X) \rightarrow f([a]X) \)

\[
\begin{array}{ccc}
  f(a) & & f([b]a) \\
  \langle R, \epsilon, I, [X \mapsto a] \rangle & \Rightarrow_{\alpha} & \langle R, \epsilon, (a b), [X \mapsto a] \rangle \\
  f([a]a) & & f([b]a) \\
\end{array}
\]
Unexpected problems:

Confluence of orthogonal NRSs does not hold in general.

Example: \( R = \vdash f(X) \rightarrow f([a]X) \)

\[
\begin{array}{c}
\langle R,\epsilon,I,[X\mapsto a]\rangle \\
\vdash f(a) \\
\quad \downarrow \\
\langle R,\epsilon,(a\ b),[X\mapsto a]\rangle \\
\end{array}
\begin{array}{c}
f([a]a) \\
\not\approx_{\alpha} \\
\quad \uparrow \\
f([b]a)
\end{array}
\]

Solution:

- Additional conditions (\( \alpha \)-stability); or
- Transform the notion of rewriting (closed rewriting).
Unexpected problems:

Confluence of orthogonal NRSs does not hold in general.

Example: \( R = \vdash f(X) \rightarrow f([a]X) \)

\[
\begin{array}{c}
\langle R, \epsilon, I, [X \mapsto a] \rangle \\
\downarrow \\
\langle R, \epsilon, (a \ b), [X \mapsto a] \rangle \\
\downarrow \\
f([a]a) \not\approx_\alpha f([b]a)
\end{array}
\]

Solution:

- Additional conditions (\( \alpha \)-stability); or
- Transform the notion of rewriting (closed rewriting).

What about types?
Importance of Types

- Add formalism to programming languages.
- Prevent errors.
- Existing nominal type systems: simple, polymorphic and dependent type systems.
Grammar of types

- A set of type variables $\mathbb{T}$
- A set of type constructors as $\mathbb{T}_C$ (bool, nat, real, list etc)
- A signature $\Sigma$ with function symbols and their corresponding type declarations $\sigma \hookrightarrow \tau$
- Types are given by

$$\tau ::= \beta \mid () \mid (\tau \times \tau) \mid C\tau \mid [\tau]\tau \mid \tau \cap \tau.$$
Grammar of types

- A set of type variables $\mathbb{T}$
- A set of type constructors as $\mathbb{T}_C$ ($\text{bool}$, $\text{nat}$, $\text{real}$, $\text{list}$ etc)
- A signature $\Sigma$ with function symbols and their corresponding type declarations $\sigma \mapsto \tau$
- Types are given by
  $$\tau ::= \beta \mid () \mid (\tau \times \tau) \mid C\tau \mid [\tau]\tau \mid \tau \cap \tau.$$  

Partial order between types

\begin{align*}
(A1) & \quad \sigma \leq \sigma \quad & (A2) & \quad \sigma \cap \tau \leq \sigma \\
(R1) & \quad \frac{\sigma \leq \tau, \sigma \leq \rho}{\sigma \leq \tau \cap \rho} \quad & (A3) & \quad \sigma \cap \tau \leq \tau \\
(R2) & \quad \frac{\sigma \leq \tau, \tau \leq \rho}{\sigma \leq \rho}
\end{align*}
## Type Inference Rules: Quasi-derivations

<table>
<thead>
<tr>
<th>Rule</th>
<th>Premises</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>$\Gamma \ni a : \sigma, \Delta \vdash a : \sigma$</td>
<td>$\Gamma, \Delta \vdash a : \sigma$</td>
</tr>
<tr>
<td>(abs)</td>
<td>$\Gamma \ni a : \sigma, \Delta \vdash t : \tau$</td>
<td>$\Gamma, \Delta \vdash [a] t : [\sigma] \tau$</td>
</tr>
<tr>
<td>($\cap_E$)</td>
<td>$\Gamma, \Delta \vdash t : \sigma_1 \cap \sigma_2$</td>
<td>$\Gamma, \Delta \vdash t : \sigma_i$</td>
</tr>
<tr>
<td>($\times_0$)</td>
<td>$\Gamma, \Delta \vdash \langle \rangle : ()$</td>
<td>$\Gamma, \Delta \vdash \langle \rangle : ()$</td>
</tr>
<tr>
<td>(var)</td>
<td>$\Gamma \ni X : \sigma, \Delta \vdash \pi \cdot X : \sigma$</td>
<td>$\Gamma, \Delta \vdash \pi \cdot X : \sigma$</td>
</tr>
<tr>
<td>($\cap_I$)</td>
<td>$\Gamma, \Delta \vdash t : \sigma_1$ $\Gamma, \Delta \vdash t : \sigma_2$</td>
<td>$\Gamma, \Delta \vdash t : \sigma_1 \cap \sigma_2$</td>
</tr>
<tr>
<td>($\times$)</td>
<td>$\Gamma, \Delta \vdash \langle t_1, t_2 \rangle : (\sigma_1 \times \sigma_2)$ $\Sigma_f = \tau \leftrightarrow \gamma$ $\sigma \leq \tau$</td>
<td>$\Gamma, \Delta \vdash \langle t_1, t_2 \rangle : (\sigma_1 \times \sigma_2)$ $\Sigma_f = \tau \leftrightarrow \gamma$ $\sigma \leq \tau$ $\Gamma, \Delta \vdash t : \sigma$</td>
</tr>
<tr>
<td>(f)</td>
<td>$\Gamma, \Delta \vdash f t : \gamma$</td>
<td>$\Gamma, \Delta \vdash f t : \gamma$</td>
</tr>
</tbody>
</table>
Type Inference Rules: Quasi-derivations

\[
\begin{align*}
\text{(a)} & \quad \frac{\Gamma \not\ni a : \sigma, \Delta \vdash a : \sigma}{\Gamma, \Delta \vdash a : \sigma} \\
\text{(abs)} & \quad \frac{\Gamma \ni a : \sigma, \Delta \vdash t : \tau}{\Gamma, \Delta \vdash [a] t : [\sigma] \tau} \\
\text{(\(\cap_E\))} & \quad \frac{\Gamma, \Delta \vdash t : \sigma_1 \cap \sigma_2}{\Gamma, \Delta \vdash t : \sigma_i} \\
\text{(\(\times\_0\))} & \quad \frac{\Gamma, \Delta \vdash \langle \rangle : ()}{\Gamma, \Delta \vdash \langle \rangle : ()} \\
\text{(var)} & \quad \frac{\Gamma \not\ni X : \sigma, \Delta \vdash \pi \cdot X : \sigma}{\Gamma \not\ni X : \sigma, \Delta \vdash \pi \cdot X : \sigma} \\
\text{(\(\cap_i\))} & \quad \frac{\Gamma, \Delta \vdash t : \sigma_1 \quad \Gamma, \Delta \vdash t : \sigma_2}{\Gamma, \Delta \vdash t : \sigma_1 \cap \sigma_2} \\
\text{(\(\times\))} & \quad \frac{\Gamma, \Delta \vdash t_1 : \sigma_1 \quad \Gamma, \Delta \vdash t_2 : \sigma_2}{\Gamma, \Delta \vdash \langle t_1, t_2 \rangle : (\sigma_1 \times \sigma_2)} \\
\text{(f)} & \quad \frac{\Sigma_f = \tau \leftrightarrow \gamma \quad \tau \leq \sigma \quad \Gamma, \Delta \vdash t : \sigma}{\Gamma, \Delta \vdash f t : \gamma}
\end{align*}
\]
Definition (Essential environment)

For a quasi-derivation of $\Gamma', \Delta \vdash t : \tau$, let $\Gamma \ntriangleright X : \sigma, \Delta \vdash \pi \cdot X : \sigma$ be a leaf of it such that $X \in \text{Vars}(t)$. Thus, the set

$$\pi^{-1} \Gamma \setminus \{ a : \tau' | \Delta \vdash a \not\# X \}$$

is an essential environment of the quasi-derivation with respect to $X$. 
Definition (Essential environment)

For a quasi-derivation of $\Gamma', \Delta \vdash t : \tau$, let $\Gamma \vDash X : \sigma, \Delta \vdash \pi \cdot X : \sigma$ be a leaf of it such that $X \in \text{Vars}(t)$. Thus, the set $\pi^{-1} \Gamma \setminus \{a : \tau' \mid \Delta \vdash a \# X\}$ is an *essential environment* of the quasi-derivation with respect to $X$.

Definition (Diamond property)

A quasi-derivation of $\Gamma', \Delta \vdash t : \tau$ has the *diamond property* if, for each $X \in \text{Vars}(t)$, the essential environments with respect to $X$ are equal.
Results

Lemma (Subtype property)

\( \text{If } \Gamma, \Delta \vdash t : \tau \text{ and } \tau \leq \tau', \text{ then } \Gamma, \Delta \vdash t : \tau'. \)
Results

Lemma (Subtype property)

If $\Gamma, \Delta \vdash t : \tau$ and $\tau \leq \tau'$, then $\Gamma, \Delta \vdash t : \tau'$.

Lemma (Object level equivariance)

If $\Gamma, \Delta \vdash t : \sigma$, then $\pi \Gamma, \Delta \vdash \pi \cdot t : \sigma$. 
## Results

### Lemma (Subtype property)

If $\Gamma, \Delta \vdash t : \tau$ and $\tau \leq \tau'$, then $\Gamma, \Delta \vdash t : \tau'$.

### Lemma (Object level equivariance)

If $\Gamma, \Delta \vdash t : \sigma$, then $\pi \Gamma, \Delta \vdash \pi \cdot t : \sigma$.

### Lemma ($\alpha$-equivalence preserves types)

If $\Gamma, \Delta \vdash t : \tau$ and $\Delta \vdash t \approx_{\alpha} s$, then $\Gamma, \Delta \vdash s : \tau$. 

---

**Nominal Intersection Types**

Informal (, mas Formal!), Brasília
Typeability

Definition (Typeability problem)

A given term in context $\Delta \vdash t$ is a typeability problem that asks if there exist a solution $\langle \Gamma, \tau \rangle$ such that $\Gamma, \Delta \vdash t : \tau$. 
Typeability

**Definition (Typeability problem)**

A given term in context $\Delta \vdash t$ is a typeability problem that asks if there exist a solution $\langle \Gamma, \tau \rangle$ such that $\Gamma, \Delta \vdash t : \tau$.

**Definition (Principal typings)**

A pair $\langle \Gamma, \tau \rangle$ is a principal typing of $\Delta \vdash t$ if it solves this typeability problem and, for any other solution $\langle \Gamma', \tau' \rangle$, $\Gamma' \leq \Gamma$ and $\tau \leq \tau'$ hold.
### Definition (Typeable closed rules)

A typeable closed rule $\Phi, \nabla \vdash l \rightarrow r : \tau$ satisfies:

- $\text{Vars}(\Phi, \nabla, r) \subseteq \text{Vars}(l)$;
- $\langle \Phi, (\tau \times \tau) \rangle$ is a principal typing of $\nabla \vdash \langle l, r \rangle$.
- $\nabla \vdash l \rightarrow r$ is a closed rule (it matches a freshened version).
- If $\Sigma_f = \tau \hookrightarrow \gamma$ and $\Gamma, \Delta \vdash f \, t : \gamma$ occurs in the derivation of types, then $\Gamma, \Delta \vdash t : \sigma$ and $\tau \leq \sigma$. 

**Lemma (Subject Reduction)**

Given a typeable closed rule $\Phi, \nabla \vdash l \rightarrow r : \tau$, if $\Gamma, \Delta \vdash s : \sigma$ and $\Delta \vdash s \mathcal{R} \rightarrow \mathcal{C} t$, then $\Gamma, \Delta \vdash t : \sigma$. 

---

Nominal Intersection Types

Informal (, mas Formal!), Brasíla
### Definition (Typeable closed rules)

A typeable closed rule \( \Phi, \nabla \vdash l \rightarrow r : \tau \) satisfies:

- \( \text{Vars}(\Phi, \nabla, r) \subseteq \text{Vars}(l) \);
- \( \langle \Phi, (\tau \times \tau) \rangle \) is a principal typing of \( \nabla \vdash \langle l, r \rangle \).
- \( \nabla \vdash l \rightarrow r \) is a closed rule (it matches a freshened version).
- If \( \Sigma_f = \tau \hookrightarrow \gamma \) and \( \Gamma, \Delta \vdash f \; t : \gamma \) occurs in the derivation of types, then \( \Gamma, \Delta \vdash t : \sigma \) and \( \tau \leq \sigma \).

### Lemma (Subject Reduction)

*Given a typeable closed rule \( \Phi, \nabla \vdash l \rightarrow r : \tau \), if \( \Gamma, \Delta \vdash s : \sigma \) and \( \Delta \vdash s \overset{R}{\Rightarrow_c} t \), then \( \Gamma, \Delta \vdash t : \sigma \).*
Conclusion and Future Work

- We have a preliminary intersection type system for nominal terms that preserves types for \( \alpha \)-equivalent terms.
- It is expected to develop an algorithm to return principal typings for terms in context.
- The conditions in which subject reduction (expansion) holds must be studied.
Elliot Fairweather, Maribel Fernández, and Murdoch James Gabbay.
Principal types for nominal theories.

Elliot Fairweather, Maribel Fernández, Nora Szasz, and Alvaro Tasistro.
Dependent types for nominal terms with atom substitutions.

M. Fernández and M. J. Gabbay.
Nominal rewriting (journal version).

Steffen van Bakel.
Rank 2 intersection type assignment in term rewriting systems.

Steffen van Bakel and Mariangiola Dezani-Ciancaglini.
Characterising strong normalisation for explicit substitutions.