Translating Specifications from Nominal Logic to CIC with the Theory of Contexts

Marino Miculan Ivan Scagnetto Furio Honsell

Department of Mathematics and Computer Science
University of Udine

MERλIN 2005, Tallinn, September 30, 2005

Metalogics for binders

- Many logics for reasoning about object systems with *binders*: Nominal Logics, CIC/ToC, Fresh Logic, $FO\lambda^{\nabla}$, ...
- Intended to be *metalogical specification systems*:
  - a formalism (*metalanguage*) $\mathcal{L}$ equipped with an *encoding methodology*
  - a given object system $\mathcal{S}$ (e.g., λ-calculus, π-calculus) can be encoded, yielding a logic $\mathcal{L}(\mathcal{S})$, where tools and techniques are provided for reasoning about it.
- These logics differ in many aspects, e.g.:
  - kind of logic (first-order, higher-order, type theory,...)
  - how binders are represented (FO, SO, HO, eq. classes...)
  - "intended behaviour" of bound symbols (names, variables...)

$\Rightarrow$ One object system $\mathcal{S}$, many different formalization and logics $\mathcal{L}_1(\mathcal{S}), \mathcal{L}_2(\mathcal{S}), \ldots$

How to compare different metalogics?

In this work we consider *logical expressivity:*

Question

for any given object system $\mathcal{S}$, can all properties derivable in $\mathcal{L}_1(\mathcal{S})$ be derived also in $\mathcal{L}_2(\mathcal{S})$?

Strategy

Define a *translation* of the terms and formulas of $\mathcal{L}_1(\mathcal{S})$ into $\mathcal{L}_1(\mathcal{S})$, and check that the translation preserves derivability.

In this work

We define a translation from *(Intuitionistic) Nominal Logic* (NL) to *Calculus of Inductive Constructions with the Theory of Contexts* (CIC/ToC).

Why?

Motivations:

- compare the logical expressivity
- enlighten similarities and differences
- streamlining encoding methodologies in CIC/ToC
- reusing existing implementations of CIC/ToC (i.e., Coq), for NL (albeit not as efficient as specially-designed implementations)

But notice: no reductionism intended! Many other theoretical and pragmatical issues should be considered, including:

- proof theory, proof search, decidability, model theory...
- closeness to informal reasoning (cf. POPLMark challenge)

For the impatient: the results

The translation from NL specifications into CIC/ToC works, i.e.:

there is a systematic way for transforming terms, formulas and sequents of NL into terms and propositions of CIC/ToC, which does preserve derivability of properties.

(Not surprisingly,) the translation is *not* conservative: there are valid sequents, provable in CIC/ToC but not in NL.

End of the talk.

Still there? Ok: for the curious, in the rest of the talk we will enter a bit in the details. . .

NL vis-a-vis CIC/ToC

Let us compare some issues of the two frameworks:

	NL	CIC/ToC
logic	first order	higher order
abstractions	equiv. classes	true functions
binding operators	first order	second order
bound symbols	a free in $\langle a \rangle t$	x not free in $\lambda x.t$
new quantifier	$\text{И}x.A$	—
Axiom of Unique Choice	consistent	inconsistent
	$\Rightarrow$ powerful functional language	$\Rightarrow$ weak functional language

The translation is going to be tricky, because of all these differences.

Nominal signatures

Definition (Nominal signatures)

A *nominal signature* is $\mathcal{S} = (N, D, C, P)$ where

- $N = \{\nu_1, \ldots, \nu_n\}$ are the *name types symbols*;
- $D = \{\delta_1, \ldots, \delta_m\}$ are the *data types symbols*;
  The *sorts* σ and *arities* α are defined as:

$$\sigma ::= () \mid \nu, \sigma \mid \langle \nu_1 \ldots \nu_k \rangle \delta, \sigma \quad (k \geq 0)$$
$$\alpha ::= \sigma \to \delta$$

- $C = \{c_1{:}\alpha_1, \ldots, c_j{:}\alpha_j\}$ are the *data constructors*.
- $P = \{p_1{:}\sigma_1, \ldots, p_k{:}\sigma_k\}$ are *(atomic) predicate symbols*.

Essentially, in sorts only name types may appear in negative positions, denoting that binders act on names.

Nominal signatures (cont.)

Example: untyped λ-calculus

$$
\begin{array}{ll}
\mathcal{S}_\lambda = (\{\nu\}, & \text{one sort of variables} \\
\quad \{\Lambda\}, & \text{one sort of terms\dots} \\
\quad \{var{:}\nu \to \Lambda, & \text{\dots with three constructors} \\
\quad\ \ \lambda{:}\langle\nu\rangle\Lambda \to \Lambda, & \\
\quad\ \ app{:}(\Lambda, \Lambda) \to \Lambda\}, & \\
\quad \{\longrightarrow: (\Lambda, \Lambda)\}) & \text{and a binary predicate}
\end{array}
$$

Formal terms are generated by usual typing rules. In particular

$$
\frac{\Gamma, \vec{n}_1 : \vec{\nu}_1 \vdash t_1 : \delta_1 \quad \dots \quad \Gamma, \vec{n}_k : \vec{\nu}_k \vdash t_k : \delta_k}{\Gamma \vdash c((\vec{n}_1)t_1, \dots, (\vec{n}_k)t_k) : \delta} \; Constr_c
$$

where $c{:}(\langle\vec{\nu}_1\rangle\delta_1, \dots, \langle\vec{\nu}_k\rangle\delta_k) \to \delta \in C$.
E.g.: $\lambda((x)app(var(x), var(x)))$ is the formal notation for $\lambda x.(x\ x)$.

Nominal Logic of a Nominal Signature: types and terms

Given a signature $\mathcal{S} = (N, D, C, P)$, we can define a *nominal logic for* $\mathcal{S}$ NINL($\mathcal{S}$) (J.Cheney's style).

Terms: a simply-typed λ-calculus with constants and types from $\mathcal{S}$

- types: for $\delta \in D$ and $\nu \in N$: $\tau ::= \delta \mid \nu \mid \tau \rightarrow \tau' \mid \langle\nu\rangle\tau$
  Arities of $\mathcal{S}$ are represented by types in currified form.
- terms: for $c \in C$:

$$t, u ::= x \mid \mathrm{a} \mid \lambda x{:}\tau.t \mid t\ \ u \mid c \mid swap_{\nu\tau} \mid abs_{\nu\tau}$$

  $(swap\ a\ b\ v)$ (shortened $(a\ b) \cdot v$) represents the term obtained by swapping all occurences of a and b in t;
  $(abs\ a\ u)$ (shortened $\langle a\rangle u$), represents the term obtained by "abstracting" a in t.

Nominal Logic of a Nominal Signature: formulas

Formulas: first order logic, with atomic propositions from P.

$$\begin{aligned}\phi, \psi ::= &\top \mid \bot \mid p(\vec{t}) \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \supset \psi \\ &\mid t \approx u \mid a\#t \mid \forall x{:}\tau.\phi \mid \exists x{:}\tau.\phi \mid \text{И}a{:}\nu.\phi\end{aligned}$$

Well-formedness of $\text{И}a.\phi$ is subject to some *freshness condition* about the bound variable:

$$\frac{\Sigma\#a{:}\nu \vdash \phi \ \mathit{form}}{\Sigma \vdash \text{И}a{:}\nu.\phi \ \mathit{form}}$$

To this end, the *(typing) contexts* may contain variables (of names) subject to freshness informations:

$$\Sigma ::= \langle\rangle \mid \Sigma, x{:}\tau \mid \Sigma\#a{:}\nu$$

$\Sigma\#a{:}\nu$ means "a is a variable to be instantiated with names different from those used in Σ".

Nominal Logic of a Nominal Signature: axioms

(S_1) $(a\ a) \cdot x \approx x$
(S_2) $(a\ b) \cdot (a\ b) \cdot x \approx x$
(S_3) $(a\ b) \cdot a \approx b$
(E_1) $(a\ b) \cdot c \approx c$
(E_2) $(a\ b) \cdot (t\ u) \approx ((a\ b) \cdot t)((a\ b) \cdot u)$
(E_3) $p(\vec{x}) \supset p((a\ b) \cdot \vec{x})$
(E_4) $(a\ b) \cdot \lambda x{:}\tau.t \approx \lambda x{:}\tau.(a\ b) \cdot t[((a\ b) \cdot x)/x]$
(F_1) $a\#x \wedge b\#x \supset (a\ b) \cdot x \approx x$
(F_2) $a\#b \quad (a{:}\nu, b{:}\nu', \nu \neq \nu')$
(F_3) $a\#a \supset \bot$
(F_4) $a\#b \vee a \approx b$
(A_1) $a\#y \wedge x \approx (a\ b) \cdot y \supset \langle a\rangle x \approx \langle b\rangle y$
(A_2) $\langle a\rangle x \approx \langle b\rangle y \supset (a \approx b \wedge x \approx y) \vee (a\#y \wedge x \approx (a\ b) \cdot y)$
(A_3) $\forall y : \langle \nu\rangle\tau \exists a : \nu \exists x : \tau . y \approx \langle a\rangle x$

Nominal Logic of a Nominal Signature: rules (in ND-style)

$$\frac{}{\Sigma : \Gamma \Rightarrow \phi}\ Ax \quad \phi \text{ instance of some axiom}$$

$$\frac{\Sigma\#a{:}\nu : \Gamma \Rightarrow \phi}{\Sigma : \Gamma \Rightarrow \phi}\ \mathit{Fresh}$$

$$\frac{\Sigma\#a{:}\nu : \Gamma \Rightarrow \phi}{\Sigma : \Gamma \Rightarrow \text{И}a.\phi}\ \text{И}\mathcal{I}$$

$$\frac{\Sigma : \Gamma \Rightarrow \text{И}a.\phi \quad \Sigma\#a{:}\nu : \Gamma, \phi \Rightarrow \psi}{\Sigma : \Gamma \Rightarrow \psi}\ \text{И}\mathcal{E}$$

$$\frac{\phi \in \Sigma^{\#}}{\Sigma : \Gamma \Rightarrow \phi}\ \Sigma\#$$

where $\Sigma^{\#}$ denotes the set of *freshness formulas in* Σ, i.e., the formulas $a\#t$ "derivable" in Σ.

Nominal Signatures in CIC/ToC

A nominal signature $\mathcal{S}$ can be encoded in CIC in 4 easy steps:

1. encoding of the syntax of terms, using weak higher-order abstract syntax;
2. syntax-driven definition of the "non-occurrence predicates"
3. atomic predicates are defined as `(Co)Inductive` propositions ("shallow embedding")
4. addition of the axioms of the Theory of Contexts for the given signature (using the `notin` predicates previously defined).

The resulting system is denoted as CIC/ToC($\mathcal{S}$).

Nominal Signatures in CIC/ToC (cont.)

For instance, the λ-calculus:

```
Parameter Var: Set.
Inductive Term: Set :=
    var: Var -> Term
  | lam: (Var -> Term) -> Term
  | app: Term -> Term -> Term.
```

```
Inductive notin_Term (x:Var): Term -> Prop :=
 notin_var: forall y:Var, x<>y -> (notin_Term x (var y))
|notin_lam: forall t: Var -> Term,
            (forall y:Var, x<>y -> (notin_Term x (t y)))
              -> (notin_Term x (lam t))
[...]
```

Formal meaning: `(notin_Term x A)` holds iff $x \notin FV(A)$.

The Theory of Contexts (ToC)

The Theory of Contexts is a set of axioms formalizing some simple properties about variables (ranging over names) and term contexts (i.e., terms with *holes*):

```
(* existence of fresh names *)
Axiom fresh_i: forall t:tau, exists a:Name_i, (notin a t).
(* decidability of equality of names *)
Axiom Name_i_dec_i: forall a b:Name_i, a=b \/ a<>b.
(* restricted beta-expansion *)
Axiom tau_exp: forall t:tau, forall x:Name,
          exists t':Name->tau, (notin x t') /\ t=(t' x).
(* restricted extensionality *)
Axiom tau_ext: forall f g:Name->tau,forall x:Name,
       (notin x f) -> (notin x g) ->
       (f x)=(g x) -> f=g.
```

Translating NINL($\mathcal{S}$) into CIC/ToC($\mathcal{S}$)

The translation is defined by giving a series of maps.

Types:

$$
\begin{aligned}
[\![\delta_i]\!] &= \texttt{delta\_}i \quad (\delta \in D) \\
[\![\nu_i]\!] &= \texttt{Name\_}i \quad (\nu_i \in N) \\
[\![\tau \to \tau']\!] &= [\![\tau]\!]\ \texttt{->}\ [\![\tau']\!] \\
[\![\langle\nu\rangle\tau]\!] &= [\![\nu]\!]\ \texttt{->}\ [\![\tau]\!]
\end{aligned}
$$

Signatures are also easy, but notice that

$$
\begin{aligned}
[\![\Sigma \# a : \nu]\!] = [\![\Sigma]\!] \\
&\texttt{Variable}\,\texttt{a} : [\![\nu]\!] \\
&\texttt{Hypothesis fresh\_a:(notin a x1)/\textbackslash...(notin a xn).}
\end{aligned}
$$

(where $dom(\Sigma) = \{x_1, \ldots, x_n\}$)

Translation of terms

Tricky, due to the fact that NINL has a first-order approach, while CIC/ToC is second-order.
Consider the case $\langle a \rangle t$ in some NINL($\mathcal{S}$).

- Here, a is free (actually can be *any* term (of the right name type))
- But $\langle a \rangle t$ should be mapped to some *functional* term `u:Name->tau` in CIC/ToC($\mathcal{S}$), where
  - $a \notin FV(u)$ and
  - such that `(u` $[\![a]\!]$`)` corresponds to t.
- How to define such `u`?

Translation of terms

- "Solution:" assume that the correct u is an *auxiliary contextual variable* provided by a *quantification* outside the atomic proposition containing $\langle a \rangle t$.
- An atomic proposition $p(\langle a \rangle t)$ will be mapped to
  `forall u:Name->tau,(u a)=t ->(notin a u) -> (p u)`
  The local assumptions are essential.
- The translation of swapping is similar: $p((a\ b) \cdot t)$ is mapped to
  `forall u:Name->Name->tau,(u a b)=t -> (notin a u) -> (notin b u) -> (p (u b a))`
- (Eventually, during the proofs, existence of such u's can be proved using the axiom of β-expansion.)

(This is the "relational feel" of CIC/ToC!).

Translation of formulas

Mostly easy. Interesting cases:

$$[\![\text{И} a{:}\nu.\phi]\!]_\Sigma = \texttt{forall a:Name,}$$
$$\texttt{(notin a x1)->...->(notin a xn)->} [\![\phi]\!]_{\Sigma,a:\nu}$$
$$\text{where } dom(\Sigma) = \{x_1, \ldots, x_n\}$$
$$[\![a\#t]\!]^\Phi_\Sigma = \texttt{(notin } [\![a]\!]^\Phi_\Sigma\ [\![t]\!]^\Phi_\Sigma\texttt{)}$$
$$[\![t_1 \approx t_2]\!]^\Phi_\Sigma = [\![t_1]\!]^\Phi_\Sigma \;\texttt{=}\; [\![t_2]\!]^\Phi_\Sigma$$

For atomic proposition $p(t_1, \ldots, t_n)$, the translation must allocate enough auxiliary contextual variables to make the translation of t_i's possible.

The translation preserves derivability

Definition

A sequent $\Sigma : \Gamma \Rightarrow \phi$ of NINL($\mathcal{S}$) is *derivable in CIC/ToC* if there is a term $\mathtt{d}$ of CIC/ToC($\mathcal{S}$) such that $[\![\Sigma]\!] \vdash_{\mathsf{ToC}(\mathcal{S})} \mathtt{d} : [\![\bigwedge \Gamma \supset \phi]\!]_{\Sigma}$.

Theorem

For all Γ, ϕ in NINL(S), if a sequent $\Sigma : \Gamma \Rightarrow \phi$ is derivable in NINL then it is derivable in CIC/ToC.

Proved by showing that the translation of all rules and axioms of NL are either derivable or admissible in CIC/ToC(S).

Examples

Axiom (S_2) : $\quad (a\ b) \cdot (a\ b) \cdot x \approx x$
translates into

```
Lemma S2: forall x: tau, forall a b: Name,
          forall y1: Name -> Name -> tau,
          (notin_tau_ho2 a y1) ->
          (notin_tau_ho2 b y1) ->
          forall y2: Name -> Name -> tau,
          (notin_tau_ho2 a y2) ->
          (notin_tau_ho2 b y2) ->
          (y2 a b)=x -> (y1 a b)=(y2 b a) ->
          (y1 b a)=x.
```

What about completeness?

Question:

if a sequent $\Sigma : \Gamma \Rightarrow \phi$ of NINL($\mathcal{S}$) is derivable in CIC/ToC($\mathcal{S}$), is it derivable in NINL($\mathcal{S}$) as well?

Answer

No, trivially. CIC/ToC is a higher-order logic, and we can prove, e.g., Peano axioms for the signature of natural numbers.

Let $\mathcal{S} = (\emptyset, \{nat\}, \{0 : nat, S : nat \rightarrow nat\}, \emptyset)$,
and $\phi \triangleq (0 \approx S(0)) \supset \bot$.
Then $\ : \ \Rightarrow \phi$ is not derivable in NINL, but it is derivable in CIC/ToC.

Completeness of the translation?

Completeness is hard to achieve. Two strategies:

1. Try to weaken CIC/ToC, e.g., by renouncing to HO features. Too bad, Soundness fails because the proofs of lemmas rely heavily on induction.
2. Try to strengthen NINL, to match the power used in CIC/ToC. Second order with induction? It may be sufficient, but then, will the good features of NL (cut elim, decidibility, etc?) still hold?
3. Third possibility: who cares? They're so different beasts...

Final remarks

- We have given a sound translation from NINL specifications to CIC/ToC.
- . . . but in CIC/ToC we can prove strictly more than in NINL.

Moral of the story:

- if you look for a "package" for reasoning about binders in your favorite HO logical framework (like Coq), CIC/ToC is a reasonable possibility: simple, compact, deeply tied with induction.
- if you prefer working in FO logic, without induction, and maybe looking for good proof theoretical properties: better if you go for NL (or $FO\lambda^{\nabla}$, but that's another story).