Solution for Part 1A

Nominal Everywhere and in Isabelle/HOL

Christian Urban and

Benjamin Pierce, Dimitrios Vytiniotis
Stephanie Weirich and Steve
Zdancewic.

Overview

- It is based on the nominal approach by Pitts et al (I guess you have heard about it by now).
- In Munich we develop a nominal datatype package for Isabelle/HOL (the plan is that one can declare datatypes with binders and the theorem prover generates a reasoning infrastructure so that one can reason almost like on "paper").
- It is still work in **progress!**

Nominal Everywhere?

- Atoms are basic entities from which data-structures can be build up (names in terms, also binders)
- Freshness $a \# x$: "atom a does not appear free in x"
  ($\_ \# \_$ has polymorphic type)
- Support supp x: "supp x is the set of free atoms in x"
  (again has a polymorphic type...terms, lists, products and so on...functions)
- Permutations $\pi \bullet x$: roughly, a restricted & better behaved operation for renamings

HOL-Philosophy

- Your **whole** universe consists of 12 or so axioms and rules; the rest is syntactic sugar (that is no lists, no datatypes, no induction, no nothing...well just a tiny bit).
- In the existing datatype package:

$$\text{datatype } \alpha \text{ list} = \text{Nil} \mid \text{Cons } \text{“}\alpha \text{ list”}$$

the things behind the scenes of Isabelle/HOL make you believe that you made a constructor-based definition (they are extremely good at this). However, they do **everything** in terms of these 12 or so axioms and rules.

- That austerity of available features is really a strength (the amazing fact is that you can really organise your world in terms of those limited, but powerful, features).
- We can introduce a framework for binding where we can declare

$$\textbf{nominal\_datatype}\ \mathrm{trm} \;=\; \mathrm{Var}\ \text{"name"} \;\mid\; \mathrm{App}\ \text{"trm"}\ \text{"trm"} \;\mid\; \mathrm{Lam}\ \text{"}\langle\!\langle\text{name}\rangle\!\rangle\text{trm"}$$

for the lambda-calculus and actually define α-equivalence classes!! Now, the rest is just the task of providing a good reasoning infrastructure for reasoning about such α-equivalence classes. That is what the nom-data-pkg does.

Rough Walk Over the Code

```
atom_decl tyvrs vrs

nominal_datatype ty =
      Tvar "tyvrs"
    | Top
    | Arrow "ty" "ty"
    | Forall "«tyvrs»ty" "ty"

nominal_datatype trm =
      Var "vrs"
    | Lam "«vrs»trm" "ty"
    | Tabs "«tyvrs»trm" "ty"
    | App "trm" "trm"
    | Tapp "trm" "ty"
```

$T_1 \to T$

$\forall[X{<:}T_1].T_2$

Lam $[x{<:}T].t$

Tabs $[X{<:}T].t$

Rough Walk Over the Code

atom_decl tyvrs vrs

nominal_datatype ty =

- We really define α-equivalence classes:

  lemma alpha_illustration:
  shows "$\forall[X{<:}T].(\text{Tvar } X) = \forall[Y{<:}T].(\text{Tvar } Y)$"
  and "$\text{Lam}[x{<:}T].(\text{Var } x) = \text{Lam}[y{<:}T].(\text{Var } y)$"
  by (simp_all add: ty.inject trm.inject alpha fresh_atm)

| Tabs "«tyvrs»trm" "ty" — Tabs $[X{<:}T].t$
| App "trm" "trm"
| Tapp "trm" "ty"

Typing Contexts

In the POPLmark-paper:

"In $\Gamma, X <: T$, the X must not be in the domain of Γ, and the free variables of T must be in the domain of Γ."

- Lists of (tyvrs,ty)-pairs: not all of them are valid.

$$\frac{}{\vdash [\,]\; \mathit{ok}}$$

$$\frac{\vdash \Gamma\; \mathit{ok} \quad X \;\#\; (\mathsf{domain}\, \Gamma) \quad T\, \mathsf{closed\_in}\, \Gamma}{\vdash ((X, T) :: \Gamma)\; \mathit{ok}}$$

- $T\, \mathsf{closed\_in}\, \Gamma \stackrel{\mathsf{def}}{=} (\mathsf{supp}\, T) \subseteq (\mathsf{domain}\, \Gamma)$

Subtyping Relation

$$\frac{}{\Gamma \vdash S <: \mathsf{Top}}$$

$$\frac{}{\Gamma \vdash \mathsf{Tvar}\ X <: \mathsf{Tvar}\ X}$$

$$\frac{(X, U) \in \Gamma \quad \Gamma \vdash U <: T}{\Gamma \vdash \mathsf{Tvar}\ X <: T}$$

$$\frac{\Gamma \vdash S_1 <: T_1 \quad \Gamma \vdash T_2 <: T_2}{\Gamma \vdash S_1 \to S_2 <: T_1 \to T_2}$$

$$\frac{\Gamma \vdash T_1 <: S_1 \qquad (X, T_1) :: \Gamma \vdash S_2 <: T_2}{\Gamma \vdash \forall [X <: S_1] . S_2 <: \forall [X <: T_1] . T_2}$$

Subtyping Relation

$$\frac{\vdash \Gamma\ ok \quad S\ \text{closed\_in}\ \Gamma}{\Gamma \vdash S <: \text{Top}}$$

$$\frac{\vdash \Gamma\ ok \quad X \in (\text{domain}\ \Gamma)}{\Gamma \vdash \text{Tvar}\ X <: \text{Tvar}\ X}$$

$$\frac{(X, U) \in \Gamma \quad \Gamma \vdash U <: T}{\Gamma \vdash \text{Tvar}\ X <: T}$$

$$\frac{\Gamma \vdash S_1 <: T_1 \quad \Gamma \vdash T_2 <: T_2}{\Gamma \vdash S_1 \to S_2 <: T_1 \to T_2}$$

$$\frac{\Gamma \vdash T_1 <: S_1 \quad X \# \Gamma \quad (X, T_1) :: \Gamma \vdash S_2 <: T_2}{\Gamma \vdash \forall [X <: S_1].S_2 <: \forall [X <: T_1].T_2}$$

Reflexivity

lemma reflexivity:
assumes "$\vdash \Gamma\ ok$" and "T closed_in Γ"
shows "$\Gamma \vdash T <: T$"

"Proof: By induction on the structure of T."

It is not so easy:

nominal_datatype ty $=$
    Tvar "tyvrs"
  | Top
  | Arrow "ty" "ty"
  | Forall "«tyvrs»ty" "ty"

$$\frac{\begin{array}{l}\forall X.\ P\ (\mathsf{Tvar}\ X)\\ P\ \mathsf{Top}\\ \forall T_1\,T_2.\ P\ T_1 \wedge P\ T_2 \Longrightarrow P\ (T_1 \rightarrow T_2)\\ \forall \color{red}{X}\ T_1\,T_2.\ \ P\ T_1 \wedge P\ T_2 \Longrightarrow P\ (\forall[X{<:}T_1].T_2)\end{array}}{P\ T}$$

Typical informal proof:
By the variable convention we may that the binder is sufficiently fresh...

nominal_datatype ty =
    Tvar "tyvrs"
  | Top
  | Arrow "ty" "ty"
  | Forall "«tyvrs»ty" "ty"

Reflexivity

The nominal datatype package derives automatically for ty the following stronger induction principle:

$$
\frac{\begin{array}{l}
\forall X\ z.\ P\ z\ (\mathsf{Tvar}\ X) \\
\forall z.\ P\ z\ \mathsf{Top} \\
\forall T_1\ T_2\ z.\ \forall z.\ P\ z\ T_1 \wedge \forall z.\ P\ z\ T_2 \Longrightarrow P\ z\ (T_1 \rightarrow T_2) \\
\forall X\ T_1\ T_2\ z.\ X \# z \wedge \forall z.\ P\ z\ T_1 \wedge \forall z.\ P\ z\ T_2 \Longrightarrow \\
\qquad\qquad P\ z\ (\forall [X{<:}T_1].T_2)
\end{array}}{P\ z\ T}
$$

nominal_datatype ty =
 Tvar "tyvrs"
 | Top
 | Arrow "ty" "ty"
 | Forall "«tyvrs»ty" "ty"

Weakening

lemma weakening:
assumes "$\Gamma \vdash S{<:}T$" and "$\vdash \Delta\ ok$" and "Δ extends Γ"
shows "$\Delta \vdash S{<:}T$"

By induction over the definition of $\Gamma \vdash S{<:}T$. This requires also strengthening of the induction principle for $\_ \vdash \_{<:}\_$. This is the most painful part of the solution. :o(

It requires for example the property that the sub-typing relation is equivariant, that is:

$$\Gamma \vdash T{<:}S \text{ implies } (\pi \bullet \Gamma) \vdash (\pi \bullet T){<:}(\pi \bullet S)$$

This is not yet shown automatically.

Transitivity/Narrowing

lemma trans_narrow:
shows "$\Gamma \vdash S <: Q \Longrightarrow \Gamma \vdash Q <: T \Longrightarrow \Gamma \vdash S <: T$"
and "$\Delta @ [(X, Q)] @ \Gamma \vdash M <: N \Longrightarrow \Gamma \vdash P <: Q$
$\Delta @ [(X, P)] @ \Gamma \vdash M <: N$"

"The two parts are proved simultaneously, by induction on the size of Q. The argument for part (2) assumes that part (1) has been established already for the Q in question; part (1) uses part (2) only for strictly smaller Q."

Transitivity/Narrowing

lem
sho
and

“Th
on t
tha
ques
Q.”

The induction principle we use:

$$\frac{\forall S\, T.\ (\mathsf{size}\, T < \mathsf{size}\, S \longrightarrow P\ T) \longrightarrow P\ S}{\forall Q.\ P\ Q}$$

Health warning: Currently the nominal datatype package does not allow a direct, simple definition for functions (functions over α-equivalence classes). We just assume (axiom) here that the size-function exists.

Structure of the Proof

lemma trans_narrow:
shows "$\Gamma \vdash S{<:}Q \Longrightarrow \Gamma \vdash Q{<:}T \Longrightarrow \Gamma \vdash S{<:}T$"
and "$\Delta @ [(X, Q)] @ \Gamma \vdash M{<:}N \Longrightarrow \Gamma \vdash P{<:}Q$
$\Delta @ [(X, P)] @ \Gamma \vdash M{<:}N$"
proof (induct Q fixing: $\Gamma\ S\ T\ \Delta\ X\ P\ M\ N$ rule: measure_induct_rule)
 case (less Q)
 have ih_trans:
  "$\bigwedge Q' \Gamma S T.$
   $[\text{size}\, Q' < \text{size}\, Q; \Gamma \vdash S{<:}Q'; \Gamma \vdash Q'{<:}T] \Longrightarrow \Gamma \vdash S{<:}T$"
 have ih_narrow:
  "$\bigwedge Q' \Delta \Gamma X M N P.$
   $[\text{size}\, Q' < \text{size}\, Q; \Delta @ [(X, Q')] @ \Gamma \vdash M{<:}N; \Gamma \vdash Q'{<:}P]$
   $\Longrightarrow \Delta @ [(X, P)] @ \Gamma \vdash M{<:}N$"
 have trans_case: "$\bigwedge \Gamma S T. [\Gamma \vdash S{<:}Q; \Gamma \vdash Q{<:}T] \Longrightarrow \Gamma \vdash S{<:}T$"
  proof. . .qed
 { case 1... (* goal transitivity *)
  case 2... (* goal narrowing *)
 }
qed

Forall-Case in Trans.

have trans_case: "$\bigwedge \Gamma S T.[\Gamma \vdash S{<:}Q; \Gamma \vdash Q{<:}T] \Longrightarrow \Gamma \vdash S{<:}T$"
proof (nominal_induct Γ S $Q \equiv Q$ rule: subtype_induct)
case (Forall Γ X S_1 S_2 Q_1 Q_2)
  hence lh_drv_prms: "$\Gamma \vdash Q_1{<:}S_1$" and "$(X, Q_1) :: \Gamma \vdash S_2{<:}Q_2$"
  and rh_drv: "$\Gamma \vdash \forall[X{<:}Q_1].Q_2{<:}T$"
  and fresh_cond: "$X \ \# \ \Gamma$" "$X \ \# \ Q_1$" by simp_all
  from "$\forall[X{<:}Q_1].Q_2 = Q$"
    have Q_{12}_less: "size Q_1 < size Q" "size Q_2 < size Q" by simp
  from rh_drv have "$T =$ Top $\vee$
  $\exists T_1 T_2. T = \forall[X{<:}T_1].T_2 \wedge \Gamma \vdash T_1{<:}Q_1 \wedge (X, T_1) :: \Gamma \vdash Q_2{<:}T_2$"
    using fresh_cond by (simp add: S_ForallE)
  moreover have "S_1 closed_in Γ" "S_2 closed_in $((X, Q_1) :: \Gamma)$"
    using lh_drv_prms by (simp add: subtype_implies_closed)
  hence "$(\forall[X{<:}S_1].S_2)$ closed_in Γ" by (force simp add: closed_in_def)
  moreover have "$\vdash \ \Gamma ok$" using rh_drv by (simp add: subtype_implies_ok)
  moreover
  { assume
  "$\exists T_1 T_2. T = \forall[X{<:}T_1].T_2 \wedge \Gamma \vdash T_1{<:}Q_1 \wedge (X, T_1) :: \Gamma \vdash Q_2{<:}T_2$"

Forall-Case in Trans. (II)

{ assume
"$\exists T_1 T_2.\, T = \forall[X{<:}T_1].T_2 \wedge \Gamma \vdash T_1{<:}Q_1 \wedge (X,T_1)::\Gamma \vdash Q_2{<:}T_2$"
  then obtain T_1 T_2
  where T_inst: "$T = \forall[X{<:}T_1].T_2$"
  and rh_drv_prms: "$\Gamma \vdash T_1{<:}Q_1$" "$(X,T_1)::\Gamma \vdash Q_2{<:}T_2$" by force
  from ih_trans[of "Q_1"] have "$\Gamma \vdash T_1{<:}S_1$"
    using lh_drv_prms rh_drv_prms Q_{12}_less by blast
  moreover from ih_narrow[of "Q_1"] have "$(X,T_1)::\Gamma \vdash S_2{<:}Q_2$"
    using lh_drv_prms rh_drv_prms Q_{12}_less by simp
  moreover from ih_trans[of "Q_2"] have "$(X,T_1)::\Gamma \vdash S_2{<:}T_2$"
    using rh_drv_prms Q_{12}_less by simp
  ultimately have "$\Gamma \vdash \forall[X{<:}S_1].S_2{<:}\forall[X{<:}T_2].T_2$"
    using fresh_cond by simp
  hence "$\Gamma \vdash \forall[X{<:}S_1].S_2{<:}T$ using T_inst by simp
}
ultimately show "$\Gamma \vdash \forall[X{<:}S_1].S_2{<:}T$ by blast
qed

Nominal Datatype Pkg

My favourite theorem-prover is X ($X \neq$ Isabelle/HOL). Why can't I have a nominal datatype package in X? Christian's prediction: there won't be any for the foreseeable future:

- In Isabelle we can generate the **type** of α-equivalence classes. We can write

$$\forall\Gamma\ (T :: \mathsf{ty})\ (S :: \mathsf{ty}).\ \ldots\Gamma \vdash S{<:}T \ldots$$

- The permutation operation needs to be defined for "everything": nominal datatypes, lists, sets, products, functions, etc. In Isabelle you can overload definitions according to their type; that is not a show-stopper, but without overloading you just do not want to do nominal stuff.

Axiomatic Type-Classes

My favourite feature of Isabelle is axiomatic type classes. They "marry" the type-system with reasoning.

- One knows that **tys** and **trms** have finite support. The infrastructure is able to choose fresh atoms for finitely supported things.
- One can introduce an axiomatic type-class of finitely supported things. One can then abstractly show that every list containing finitely supported things, is finitely supported (likewise for products, finite sets etc).
- So, when one writes

$$\forall(\Gamma :: (\mathsf{tyvrs} \times \mathsf{ty})\ \mathsf{list}). \ldots$$

Isabelle automatically knows that typing-contexts are finitely supported.

Conclusion

- The other parts of POPLmark, in order to look nice, still need some infrastructure that is not yet implemented.
- I found Isabelle's ISAR language to be very useful (if needed one can mix the readable ISAR-proofs with the old tactic-based proofs).
- Work is progressing...however if you want to prove anything about the lambda-calculus then everything is already in place.
- There is a nominal-isabelle mailing list.