Multiple Inheritance and Coercive Subtyping or how to teach an old dog (on steroids) new tricks

Claudio Sacerdoti Coen <sacerdot@cs.unibo.it>
Enrico Tassi <tassi@cs.unibo.it>

University of Bologna

02/05/2007

Outline

Outline

An old problem (1/2)

How to represent (a hierarchy of) mathematical structures

- structures must be first class objects (no modules)
- multiple inheritance (e.g. a Riesz space is a vector space that has a lattice structure s.t. . . .)
- controlled sharing (e.g. ring = group + monoid)
- inheritance must be "symmetric"
  (e.g. NO ring = group + monoid_on (carrier group))
- subtyping (e.g. a vectore space is a Riesz space)
- structure specialization (e.g. ∀ ring on natural numbers)
  - for multiple inheritance with controlled sharing
  - to prove theorems on specialized structures

An old problem (2/2)

Additional requirements

- notation should work properly
- lambda term $\in O(n)$ where n is the size of the formula (e.g. NO carrier of monoid of additive group of ring of ...)
- unification, type inference must work properly (e.g. $\forall x.0 + x \wedge \top = x$)

Theoretical solution 1

Betarte/Tasistro: dependently typed, extensible records

- $Monoid = \langle C : Type; \circ : C \rightarrow C \rightarrow C; e : C \rangle$
- $\langle C = nat; \circ = +; e = 0 \rangle : Monoid$
- dependently typed projections to extract fields value (e.g. $M.C$)
- inheritance by extensibility (e.g. $Group = \langle Monoid; opp : C \rightarrow C \rangle$)
- subtyping is structural (permutation and addition of fields) (e.g. an ordered semigroup is an ordered group)
- Pollack: "*structural subtyping depends on accident of structure, and does not support natural mathematical definitions*"

Theoretical solution 1

Betarte/Tasistro: dependently typed, extensible records

- complex type system (subtyping is hardcoded)
- no structure specialization
- manually built specialized structures are not subtypes
- conversion, typechecking: ok
- unification, refinement, type inference: ???

Theoretical solution 2

Pollack: Sigma types (for opaque fields) + Manifest types + Coercive subtyping

- inheritance by inclusion of sub-structures (as fields) (e.g. $Monoid = \langle S : Semigroup; e : S.C\rangle$)
- coercive subtyping replaces subtyping and extensibility (e.g. $(SM) : Semigroup$ whenever $M : Monoid$)
- manifest types for controlled sharing, symmetric inheritance and structure specialization (e.g. $\langle carrier = nat; \circ : carrier \rightarrow carrier \rightarrow carrier\rangle$)

Theoretical solution 2

Pollack: Sigma types (for opaque fields) + Manifest types + Coercive subtyping

- lambda term $\in O(d * n)$ where n is the size of the formula and d the height of the inheritance graph
- coded in type theory + induction/recursion
- conversion, typechecking: ok
- unification, refinement, type inference: ???

Practical solution

Geuvers, Pollack et alt.: non extensible dependently typed records + coercive subtyping + Pebble style sharing

- used for FTA (now CoRN); requires no changes to Coq
- inheritance by inclusion of sub-structures (as fields)
- coercive subtyping replaces subtyping and extensibility
- single inheritance is ok
- asymmetric multiple inheritance, controlled sharing structure (e.g.
  $Ring = \langle \quad G : group; mult : G.C \to G.C \to G.C;$
  $1 : G.C; H : is\_semigroup\ G.C\ mult\ e \rangle$)
- specialization by Pebble style sharing only (e.g. $monoid\_on : Type \to Type$)

Practical solution

Geuvers, Pollack et alt.: non extensible dependently typed records + coercive subtyping + Pebble style sharing

- multiple coercions not allowed
- satisfactory only for linear hierarchies
- lambda term $\in O(d * n)$ where n is the size of the formula and d the height of the inheritance graph
- it works!

Our proposal

non extensible dependently typed records + coercive subtyping + manifest fields via extensional equality

- no changes to the theory/implementation of Coq/Matita
- no need for induction/recursion
- less efficient/clean/computational than Pollack's proposal
- inheritance by inclusion of sub-structures (as fields)
- coercive subtyping replaces subtyping and extensibility

Our proposal

- ring = { G: group; M: monoid; with: G.carrier = M.carrier }
- problem: $\forall R : ring. 1_R + 0_R = 1_R$ not well typed
- hint:

$$\frac{\begin{array}{c}\Gamma \vdash P : \forall B : T. A =_T B \rightarrow Type \\ \Gamma \vdash H : A =_T B \\ \Gamma \vdash M : P\ A\ (refl\_eq_T\ A)\end{array}}{\Gamma \vdash (M :_H P\ B\ H) : P\ H\ B}$$

$$\Gamma \vdash (M :_{(refl\_eq_T\ A)} P\ A\ (refl\_eq_T\ A)) \rhd M$$

- thus:

$$\forall R.(1_R :_{R.with} R.G.carrier) + 0_R = (1_R :_{R.with} R.G.carrier)$$

well typed (but not practical)

Our proposal

- idea (assuming dependent records):

$$
\begin{array}{l}
ring'\ (R : ring) = \\
\quad \{\ G = R.G; \\
\qquad M = \ \ \{\ carrier = R.G.carrier; \\
\qquad\qquad\quad op = (R.M.op :_{R.with}\ carrier \to carrier \to carrier); \\
\qquad\qquad\quad e = (R.M.e :_{R.with}\ carrier); \\
\qquad\qquad\quad neutral : (R.M.neutral :_{R.with}\ \forall x : carrier.e * x = x)\} \\
\quad \}
\end{array}
$$

- Let $R' = ring'\ R$ for some ring R.
  $R.M.carrier$ is intensionally equal to $R.G.carrier$
- Re-define the projections/coercions $M : ring \to monoid$ as
  $M\ (R : ring) := M\ (ring'\ R)$

First Properties

- symmetric multiple inheritance and controlled sharing
- but now we have multiple coercion paths!
  - we need to improve coercions (second part of the talk!)
- size of lambda terms still unsatisfactory
  - we need to improve coercions (second part of the talk!)

Structure specialization?

- structure specialization through syntactic sugar?

$$\begin{aligned}\forall S : \mathit{semigroup}\ \text{with}\ M.\mathit{carrier} = \mathit{nat}.\\ P[S, S.\mathit{carrier}, S.\mathit{op}]\end{aligned}$$

  syntactic sugar for

$$\begin{array}{l}\forall S : \mathit{semigroup}.\forall \mathit{with} : S.\mathit{carrier} = \mathit{nat}.\\ \quad \text{let } S' := \langle\ \mathit{carrier} = \mathit{nat};\\ \qquad\qquad \mathit{op} = (\mathit{op} :_{\mathit{with}} \mathit{carrier} \to \mathit{carrier} \to \mathit{carrier})\rangle\\ \quad \text{in}\\ \qquad P[S, S'.\mathit{carrier}, S'.\mathit{op}]\end{array}$$

- inheritance: semigroup with M.carrier = nat is a semigroup

Major Problem

M := $\{$carrier = Z; op = *; e = 1$\}$
R := $\{$G = G; M = M; with: G.carrier = M.carrier$\}$
$0_M + 0_M \triangleright 0_M$ and $0_R + 0_R \triangleright 0_R$ but
$1_M * 1_M \triangleright 1_M$ and $1_R * 1_R \not\triangleright 1_R$

Outline

Mathematical structures

- Dependent records used to pack carriers, operations (and properties)
- Inheritance:
  - Subtyping between dependent records
  - Coercive subtyping
- Problems:
  - Chain of coercions:
    (Carrier_OF_Setoid (Setoid_OF_SemiGroup
      (SemiGroup_OF_Group (Group_OF_Ring . . .))))
  - Multiple coercion paths for multiple inheritance
    (an unification/type inference problem)

Composite coercions

- Every time a coercion is declared, the coercions graph is automatically completed with composite coercions
- Not so simple (requires unification/refinement):
  $k_1 : \forall S : Type.G\ S \rightarrow F\ (I\ S)$
  $k_2 : \forall S : Type.F\ (H\ S) \rightarrow U\ (K\ S)$
  If we can find u and v such that
  $I\ (u\ ?_1) \cong H\ ?_2$ and $K\ ?_2 \cong v\ ?_1$ then
  $k_{12} : \forall S : Type.G\ (u\ S) \rightarrow U\ (v\ S)$
  Implementation: apply k_1 (saturated) to k_2 (saturated), refine and then λ-abstract on remaining metavariables.
- Introduces multiple paths between nodes in the coercion graph
  - Unification problem: $k_{12}\ \_\ g = k_2\ \_\ (k_1\ \_\ g)$ but how to solve $k_{12}\ \_\ ?_1 \cong k_2\ \_\ ?_2$?
  - Untamed solution: unification up to conversion (too expensive)
  - Well behaved solution: see later

Multiple inheritance

- Multiple paths are not dangerous when they are intensionally equal
- E.g.: “a Riesz space is a vector space that is also a lattice”;
  “an algebra is a vector space with a multiplicative structure”;
  “an f-algebra is a Riesz space that is also an algebra”
- Intensionally equal multiple paths are necessary
- E.g.: $\forall f. f \leq 1 \rightarrow f * f \leq 1$
  $f * f$ has type carrier of vector space of algebra of f-algebra but is used with type carrier of vector space of Reisz space of f-algebra
- Serious unification problems:
  `carrier_OF_algebra` $?_1 \cong$
  `carrier_OF_Riesz_space` $?_2$

Pullbacks

$$\begin{tikzcd} & P \arrow[dl, dashed, "f'"'] \arrow[dr, dashed, "g'"] & \\ ?1 : A \arrow[dr, "f"'] & & ?2 : B \arrow[dl, "g"] \\ & M & \end{tikzcd}$$

Consider the unification problem
$f\ t \cong g\ t'$
where f is a coercion from A to M
g is a coercion from B to M
f', g' is the smallest pullback of A and B
f' is a coercion from P to A
g' is a coercion from P to B

- If $P \neq A$ and $t = ?_1$ then unify $?_1$ with $f'\ ?_3$ ($?_3$ of type P)
- If $P \neq B$ and $t' = ?_2$ then unify $?_2$ with $g'\ ?_3$ ($?_3$ of type P)
- Then solve the initial unification problem without using conversion
- Works also for composite coercions! (triangular pullback)

Pullbacks

- Same solution for dependent coercions (just saturate in advance)
- Hidden assumptions:
  - Invariant: only well-typed terms (with possibly different types) are unified
  - To unify $?_i$ with t first unify their types
- Under the previous assumption: no circular dependency between unification and refinement

Outline

Conclusions (1/2)

- We solve the problem of multiple intensionally equal coercions paths
- Solutions is fully satisfactory (so far)
- Size of proof terms dramatically reduced

Conclusions (2/2)

- We propose an improvement of Pollack, Geuvers et alt. to “capture” manifest types by extensional equality
- Symmetric multiple inheritance, controlled sharing, subtyping
- Structure specialization?
- Notation, unification, type inference work properly
- Conversion is (asymmetrically) not preserved by composition