Towards a better-behaved unification algorithm for Coq

Beta Ziliani (MPI-SWS) & Matthieu Sozeau (Inria πr^2)

UNIF Workshop
July 13th 2014
Vienna, Austria

Unification in Coq

Unification is a crucial tool:

- Type-checking/refinement.
  `Definition c : A := t.` $\Rightarrow$ unify the infered type T of t with A.
- Tactic applications.
  On goal $\Gamma \vdash A$, `apply (lemma : forall x .. xn, T)` unifies $T[?x_1 ... ?x_n]$ and A.

Currently two slightly different unification algorithms are used, one for each case.

Unification in Coq

- Not documented, not verified, but its results are directly seen by the users.
- Higher-order (pattern-unification + postponing + heuristics)
- Dependent on reductions: β, ι and δ
- Uses a backtracking first-order unification rule for fast success: $f\ u_1 \ldots u_n \approx f\ v_1 \ldots v_n \rightsquigarrow \overrightarrow{u_i \approx v_i}$
- Includes canonical structure resolution, an overloading mechanism relying on the precise behavior of the algorithm.
- "Untyped": does not rely on the types of terms at hands, which might even not be directly convertible.
- Unsuitable for automation.

Ideally

Document and verify a predictable, performant unification algorithm to be used both for type-checking and inside tactics.

The term language of Coq

$$
\begin{array}{rcll}
t, T & \widehat{=} & x \mid c \mid s & x \in \mathcal{V}, c \in \mathcal{C}, s \in \mathcal{S} \\
 & & \mid i \mid k & i \in \mathcal{I}, k \in \mathcal{K} \\
 & & \mid \forall x : T.T \mid \lambda x : T.t \mid t\ t & \\
 & & \mid \textsf{let}\ x = t : T\ \textsf{in}\ t \mid ?u[\sigma] & ?u \in \mathcal{M} \\
 & & \mid \textsf{case}_T\ t\ \textsf{of}\ k_1\ \overline{x} \Rightarrow t; \ldots ; k_n\ \overline{x} \Rightarrow t\ \textsf{end} & \\
 & & \mid \textsf{fix}_j\ \{x/n : T := t; \ldots ; x/n : T := t\} &
\end{array}
$$

$$
\begin{array}{rcl}
\Gamma, \Psi & \widehat{=} & \cdot \mid x : T, \Gamma \mid x := t : T, \Gamma \\
\Sigma & \widehat{=} & \cdot \mid ?u : T[\Psi], \Sigma \mid ?u := t : T[\Psi], \Sigma \\
E & \widehat{=} & \cdot \mid c : T, E \mid c := t : T, E \mid I, E
\end{array}
$$

Reduction

$$(\lambda x : T.t)\ t' \rightsquigarrow_\beta t\{t'/x\}$$

$$\text{let } x = t' : T \text{ in } t \rightsquigarrow_\zeta t\{t'/x\}$$

$$h \rightsquigarrow_\delta t \qquad \text{if } (h := t : T) \in E \text{ or } (h := t : T) \in \Gamma$$

$$?u[\sigma] \rightsquigarrow_\delta t\{\sigma/\Psi\} \qquad \text{if } (?u := t : T[\Psi]) \in \Sigma$$

$$\text{case}_T\ (k_j\ \overline{a}) \text{ of } \overline{k\ \overline{x} \Rightarrow t} \text{ end} \rightsquigarrow_\iota t_j\{\overline{a/x_j}\}$$

$$\text{fix}_j\ \{F\}\ \overline{a} \rightsquigarrow_\iota t_j\{\overline{\text{fix}_m\ \{F\}/x_m}\}\ \overline{a} \qquad F = \overline{x/n : T := t}$$

Unification

Judgment: $\Sigma; \Gamma \vdash t_1 \approx t_2 \triangleright \Sigma'$

Example rule:

$$\frac{\Sigma; \Gamma \vdash \bar{t} \approx \overline{t'} \triangleright \Sigma'}{\Sigma; \Gamma \vdash ?u[\xi]\ \bar{t} \approx ?u[\xi]\ \overline{t'} \triangleright \Sigma'}\ \textsc{Meta-Same-Same}$$

Type dependency and postponment

An example with the existential quantifier:

```
Check (exist : ∀ (A : Type) (P : A → Prop) (x : A) (p : P x),
                  { x : A | P x }).

Definition f : { x : nat | x ≤ x } :=
  @exist _ _ 0 (le_n 0).
```

Uses *constraint postponment* to delay the instanciation of $?P$.
Solves $?P\ 0 = 0 \leq 0 \wedge ?P = \lambda\ x, x \leq x$.

Type dependency and postponement II

This has unpredictable behavior:

- Non-local effect: hard to reason about, error locations hard to guess
- Source of exponential complexity in presence of backtracking.

We prefer a best-effort local type inference based on bidirectional type-checking, without postponement.

In this example, this would set $?P = \lambda\, x,\ x \leq x$ before typechecking the arguments.

First-Order unification

$$\frac{\Sigma_0; \Gamma \vdash u \approx u' \triangleright \Sigma_1}{\Sigma_0; \Gamma \vdash t\ u \approx t\ u' \triangleright \Sigma_1} \text{ App-FO}$$

- Applies even when t is an unfoldable constant (lose most general unifiers, but fast success in general).
- Requires backtracking when the argument unification fails $\Rightarrow$ the reduced terms might be unifiable (slow failures if repeateadly applied).

Solutions:

- Parameterize by a set of constants on which the rule applies without backtracking, e.g. abbreviations only (Pfenning and Schürmann, TYPES'98). Does not play well with reduction.
- Experiment with caching mechanisms to be less penalized by failures.

Canonical Structure Resolution

Overloading mechanism based on records.

```
Structure monoid :=
  { carrier : Set;
    unit : carrier;
    mult : carrier → carrier → carrier;
    mult_unit_left : ∀ x, mult unit x = x
  }.
Canonical Structure nat_monoid :=
  { carrier := nat;
      unit := 0;
      mult := plus;
      mult_unit_left := fun x ⇒ eq_refl }.
```

Structure resolution

```
Check (mult :
        ∀ m : monoid, carrier m → carrier m → carrier m).
Lemma on_nat_monoid (n : nat) : mult _ 0 n = n.
Proof.
  change (mult nat_monoid 0 n = n).
  apply mult_unit_left.
Qed.
```

The unification solved the problem carrier $?m$ $=$ nat by instantiating $?m$ with nat_monoid.

Canonical Structures

- Relies on unfolding behavior (step-by-step unfolding of constants)
- Hard to reason about if constraints can be postponed.

Results

A pencil and paper presentation (3 pages of rules plus pruning) and an implementation.

- No constraint postponement
- Clean definition of pruning and higher-order pattern unification.
- "Sound"
- No other heuristic than first-order unification.

Rules I/III

$$\frac{}{\Sigma;\Gamma \vdash \mathsf{Prop} \approx \mathsf{Prop} \triangleright \Sigma}$$

$$\frac{i = j}{\Sigma;\Gamma \vdash \mathsf{Type}(i) \approx \mathsf{Type}(j) \triangleright \Sigma}$$

$$\frac{i \leq j}{\Sigma;\Gamma \vdash \mathsf{Type}(i) \lesssim \mathsf{Type}(j) \triangleright \Sigma}$$

$$\frac{\Sigma;\Gamma \vdash A_1 \approx A_2 \triangleright \Sigma' \qquad \Sigma';\Gamma, x : A_1 \vdash t_1 \approx t_2 \triangleright \Sigma''}{\Sigma;\Gamma \vdash \lambda x : A_1.t_1 \approx \lambda x : A_2.t_2 \triangleright \Sigma''}$$

$$\frac{\Sigma;\Gamma \vdash A_1 \approx A_2 \triangleright \Sigma' \qquad \Sigma';\Gamma, x : A_1 \vdash B_1 \approx B_2 \triangleright \Sigma''}{\Sigma;\Gamma \vdash \forall x : A_1.B_1 \approx \forall x : A_2.B_2 \triangleright \Sigma''}$$

$$\frac{\Sigma;\Gamma \vdash t_2 \approx t_2' \triangleright \Sigma' \qquad \Sigma';\Gamma, x := t_2 \vdash t_1 \approx t_1' \triangleright \Sigma''}{\Sigma;\Gamma \vdash \mathsf{let}\ x = t_2 : T\ \mathsf{in}\ t_1 \approx \mathsf{let}\ x = t_2' : T'\ \mathsf{in}\ t_1' \triangleright \Sigma''}$$

$$\frac{\Sigma;\Gamma \vdash t_1\{t_2/x\} \approx t_1'\{t_2'/x\} \triangleright \Sigma'}{\Sigma;\Gamma \vdash \mathsf{let}\ x = t_2 : T\ \mathsf{in}\ t_1 \approx \mathsf{let}\ x = t_2' : T'\ \mathsf{in}\ t_1' \triangleright \Sigma'}$$

$$\frac{h \in \mathscr{V} \cup \mathscr{C} \cup \mathscr{I} \cup \mathscr{K}}{\Sigma;\Gamma \vdash h \approx h \triangleright \Sigma}$$

$$\frac{\begin{array}{c}\Sigma_0;\Gamma \vdash T \approx T' \triangleright \Sigma_1 \\ \Sigma_1;\Gamma \vdash t \approx t' \triangleright \Sigma_2 \qquad \Sigma_2;\Gamma \vdash \overline{b} \approx \overline{b'} \triangleright \Sigma_3\end{array}}{\Sigma_0;\Gamma \vdash \mathsf{case}_T\ t\ \mathsf{of}\ \overline{b}\ \mathsf{end} \approx \mathsf{case}_{T'}\ t'\ \mathsf{of}\ \overline{b'}\ \mathsf{end} \triangleright \Sigma_3}$$

$$\frac{\Sigma_0;\Gamma \vdash \overline{T} \approx \overline{T'} \triangleright \Sigma_1 \qquad \Sigma_1;\Gamma \vdash \overline{t} \approx \overline{t'} \triangleright \Sigma_2}{\Sigma_0;\Gamma \vdash \mathsf{fix}_j\ \{\overline{x/n : T := t}\} \approx \mathsf{fix}_j\ \{\overline{x'/n' : T' := t'}\} \triangleright \Sigma_2}$$

$$\frac{?u := t : A[\Gamma] \in \Sigma \qquad \Sigma;\Gamma \vdash t' \approx t\{\sigma/\hat{\Gamma}\}\ \overline{t_n} \triangleright \Sigma'}{\Sigma;\Gamma \vdash t' \approx ?u[\sigma]\ \overline{t_n} \triangleright \Sigma'}$$

$$\frac{\Sigma;\Gamma \vdash t' \approx t\{t_1/x\}\ t_2\ \ldots\ t_n \triangleright \Sigma'}{\Sigma;\Gamma \vdash t' \approx (\lambda x : A.t)\ t_1\ \ldots\ t_n \triangleright \Sigma'}$$

$$\frac{\Sigma;\Gamma \vdash t' \approx t_1\{t_2/x\}\ \overline{t_n} \triangleright \Sigma'}{\Sigma;\Gamma \vdash t' \approx (\mathsf{let}\ x = t_2 : T\ \mathsf{in}\ t_1)\ \overline{t_n} \triangleright \Sigma'}$$

$$\frac{(x := t : A) \in \Gamma \qquad \Sigma;\Gamma \vdash t' \approx t\ \overline{t_n} \triangleright \Sigma'}{\Sigma;\Gamma \vdash t' \approx x\ \overline{t_n} \triangleright \Sigma'}$$

$$\frac{\begin{array}{c}t' \text{ is fix or case} \\ \Sigma;\Gamma \vdash t' \downarrow^{\mathrm{w}}_{\beta\iota\theta} t'' \qquad t' \neq t'' \qquad \Sigma;\Gamma \vdash t \approx t'' \triangleright \Sigma'\end{array}}{\Sigma;\Gamma \vdash t \approx t' \triangleright \Sigma'}$$

$$\frac{\begin{array}{c}(c := t : A) \in E \\ \text{not } \Sigma;\Gamma \vdash \mathsf{is\_stuck}\ (c\ \overline{t_n}) \qquad \Sigma;\Gamma \vdash t' \approx t\ \overline{t_n} \triangleright \Sigma'\end{array}}{\Sigma;\Gamma \vdash t' \approx c\ \overline{t_n} \triangleright \Sigma'}$$

$$\frac{\begin{array}{c}(c := t : A) \in E \\ \Sigma;\Gamma \vdash \mathsf{is\_stuck}\ t' \qquad \Sigma;\Gamma \vdash t\ \overline{t_n} \approx t' \triangleright \Sigma'\end{array}}{\Sigma;\Gamma \vdash c\ \overline{t_n} \approx t' \triangleright \Sigma'}$$

$$\frac{(c := t : A) \in E \qquad \Sigma;\Gamma \vdash t' \approx t\ \overline{t_n} \triangleright \Sigma'}{\Sigma;\Gamma \vdash t' \approx c\ \overline{t_n} \triangleright \Sigma'}$$

$$\frac{\begin{array}{c}\boxed{\Sigma_0;\Gamma \vdash t : T} \\ \Sigma_0, ?v : \mathsf{Type}(i)[\Gamma, y : A];\Gamma \vdash T \approx \forall y : A.?v[\mathsf{id}_{\Gamma,y}] \triangleright \Sigma_1 \\ \Sigma_1;\Gamma, x : A \vdash (t\ x) \approx t_1 \triangleright \Sigma_2\end{array}}{\Sigma_0;\Gamma \vdash t \approx \lambda x : A.t_1 \triangleright \Sigma_2}$$

Rules II/III

$$\frac{\Sigma;\Gamma \vdash \bar{t} \approx \overline{t'} \triangleright \Sigma'}{\Sigma;\Gamma \vdash ?u[\sigma]\ \bar{t} \approx ?u[\sigma]\ \overline{t'} \triangleright \Sigma'}$$

$$\frac{\begin{array}{c} ?u : T[\Psi_1] \in \Sigma \qquad \Psi_1 \vdash \sigma \cap \sigma' \triangleright \Psi_2 \qquad \Sigma \vdash \Psi_2 \\ \mathsf{FV}(T) \subseteq \Psi_2 \qquad \Sigma \cup \{?v : T[\Psi_2], ?u := ?v[\mathsf{id}_{\Psi_2}]\};\Gamma \vdash \bar{t} \approx \overline{t'} \triangleright \Sigma' \end{array}}{\Sigma;\Gamma \vdash ?u[\sigma]\ \bar{t} \approx ?u[\sigma']\ \overline{t'} \triangleright \Sigma'}$$

$$\frac{\begin{array}{c} ?u : T[\Psi] \in \Sigma_0 \\ \xi_1, \xi_2, \overline{t''} = \max_{|\xi_2|}(\xi_1, \xi_2, \overline{t''} \mid t' = h\ \overline{t''}\ \xi_2 \wedge \xi' = \xi_1\ \xi_2 \wedge \nexists x \in \xi_2.\ x \in h\ \overline{t''}\ \xi_1) \\ \Sigma_0 \vdash \mathsf{prune}(\xi, \xi_2; t) \triangleright \Sigma_1 \\ t' = \lambda x : A\{\xi, \xi_1/\hat{\Psi}, \bar{x}\}^{-1}.t\{\xi, \xi_1/\hat{\Psi}, \bar{x}\}^{-1} \\ \Sigma_1;\Psi \vdash t' : T' \qquad \Sigma_1;\Psi \vdash T' \lesssim T \triangleright \Sigma_2 \qquad ?u \notin t' \end{array}}{\Sigma_0;\Gamma \vdash t \approx ?u[\xi]\ \xi' \triangleright \Sigma_2 \cup \{?u := t'\}}$$

$$\frac{\begin{array}{c} ?u : T[\Psi] \in \Sigma_0 \\ \Sigma_0;\Gamma \vdash t_1 \approx t_2 \triangleright \Sigma_1 \qquad \Sigma_1;\Gamma \vdash t'\ \overline{t'_m} \approx ?u[\sigma]\ \overline{t_n} \triangleright \Sigma_2 \end{array}}{\Sigma_0;\Gamma \vdash t'\ \overline{t'_m}\ t_1 \approx ?u[\sigma]\ \overline{t_n}\ t_2 \triangleright \Sigma_2}$$

$$\frac{?u : T[\Psi] \in \Sigma_0 \qquad t \rightsquigarrow_{\delta} \cup \downarrow^{\mathrm{w}}_{\beta\iota\theta} t' \qquad \Sigma_0;\Gamma \vdash t' \approx ?u[\sigma]\ \overline{t_n} \triangleright \Sigma_1}{\Sigma_0;\Gamma \vdash t \approx ?u[\sigma]\ \overline{t_n} \triangleright \Sigma_1}$$

Rules III/III

$$\frac{\begin{array}{c}(p_j, h, \iota) \in \Delta_{\text{db}} \\ \iota := \lambda \overline{x : B}.k\ \overline{a'}\ \overline{v} \qquad v_j = h\ \overline{u'} \qquad \Sigma_1 = \Sigma_0, \overline{?y : B} \\ \Sigma_1; \Gamma \vdash \overline{a} \approx \overline{a'\{\overline{?y}/\overline{x}\}} \triangleright \Sigma_2 \qquad \Sigma_2; \Gamma \vdash \overline{u} \approx \overline{u'\{\overline{?y}/\overline{x}\}} \triangleright \Sigma_3 \\ \Sigma_3; \Gamma \vdash c \approx \iota\ \overline{?y} \triangleright \Sigma_4 \qquad \Sigma_4; \Gamma \vdash \overline{t} \approx \overline{t'} \triangleright \Sigma_5\end{array}}{\Sigma_0; \Gamma \vdash p_j\ \overline{a}\ c\ \overline{t} \approx h\ \overline{u}\ \overline{t'} \triangleright \Sigma_5}$$

Dealing with first-order unification

Problem: $?t +\!\!+ ?u \approx [1;2] +\!\!+ [3;4]$.

- In general, many solutions, no m.g.u.
- Here we allow fist-order unification, so $?t := [1;2]$, $?u := [3;4]$ is found.
- A most natural unifier notion?

Implementation

- "Easy" debugging
- Not fast (yet).
- No control on reduction/expansion (yet).

Tested on SSREFLECT library (one of the largest in Coq, supporting the 4 color theorem and Feit-Thompson):

- 40/62 files compile (time blowup after that)
- 60kLoC, 10M unification problems.
- $<$ 600 problems solvable by the legacy algorithm only.
- Bidirectional typechecking would reduce this.

Related work

- Higher-Order Dynamic Pattern Unification (Abel & Pientka): uses postponment.
- Same for J. Reed, U. Norrel's unifier for Agda and C.S.Coen's unifier for CIC.

Conclusion and Future Work

- A simpler, documented and predictable unification algorithm for Coq.
- A prototype implementation.

What's left:

- Parameterization.
- Proofs (on paper and in Coq).
- Debug & integrate the algorithm.

The End

That's all folks!