

Higher-Order unification for free

Reusing the meta-language unification for the object language

Fissore Davide & Enrico Tassi

September 10, 2024

Supported by ANR-17-EURE-0004

UNIVERSITÉ
CÔTE D'AZUR



ÉCOLE UNIVERSITAIRE DE RECHERCHE
**SYSTÈMES NUMÉRIQUES
POUR L'HUMAIN**



Metaprogramming for type-class resolution

- Our goal:
 - ▶ Type-class solver for Coq in Elpi
 - ▶ The goal of a type-class solver is to back-chain lemmas taken from a database of 'type-class instances'.
- Our problem:
 - ▶ Elpi cannot unify correctly Coq's HO terms
 - ▶ But we want/need to use Elpi's unification algorithm
- Our contribution:
 - ▶ Reusing the meta-language unification for the object language

A type-class problem in Coq

Instance forall_dec: $\forall A\ P, \text{Finite } A \rightarrow$ (* r3 *)
 $(\forall x:A, \text{Decision } (P\ x)) \rightarrow \text{Decision } (\forall x:A, P\ x).$

Goal Decision $(\forall x: \text{fin } 7, \text{nfact } x\ 3).$ (* g *)

A type-class problem in Coq

Instance forall_dec: $\forall A\ P, \text{Finite } A \rightarrow$ (* r3 *)
 $(\forall x:A, \text{Decision } (P\ x)) \rightarrow \text{Decision } (\forall x:A, P\ x)$.

Goal Decision $(\forall x: \text{fin } 7, \text{nfact } x\ 3)$. (* g *)

- $\{A \mapsto \text{fin } 7; P \mapsto \lambda x. (\text{nfact } x\ 3)\}$

A type-class problem in Coq

Instance forall_dec: $\forall A\ P, \text{Finite } A \rightarrow$ (* r3 *)
 $(\forall x:A, \text{Decision } (P\ x)) \rightarrow \text{Decision } (\forall x:A, P\ x).$

Goal $\text{Decision } (\forall x: \text{fin } 7, \text{nfact } x\ 3).$ (* g *)

- $\{A \mapsto \text{fin } 7; P \mapsto \lambda x. (\text{nfact } x\ 3)\}$

- subgoals:

$\text{Finite } (\text{fin } 7)$ and $(\forall x:A, \text{Decision } ((\lambda x. (\text{nfact } x\ 3))\ x))$

Coq terms in elpi : HOAS

Coq	Elpi
f	$c\text{"f"}$
$f\ a$	$\text{app}[c\text{"f"}, c\text{"a"}]$
$\lambda(x : T). F\ x$	$\text{fun } T\ (x\ \backslash\ \text{app}[F, x])$
$\forall(x : T), F\ x$	$\text{all } T\ (x\ \backslash\ \text{app}[F, x])$
\dots	\dots

- variable bindings and substitutions are for free
- easy term inspection

The above type-class problem in elpi

Instance forall_dec: $\forall A\ P, \text{Finite } A \rightarrow$ (* r3 *)
 $(\forall x:A, \text{Decision } (P\ x)) \rightarrow \text{Decision } (\forall x:A, P\ x).$

Goal Decision $(\forall x: \text{fin } 7, \text{nfact } x\ 3).$ (* g *)

↓

The above type-class problem in elpi

```
Instance forall_dec:  $\forall A\ P, \text{Finite } A \rightarrow$  (* r3 *)  
  ( $\forall x:A, \text{Decision } (P\ x) \rightarrow \text{Decision } (\forall x:A, P\ x)$ ).
```

```
Goal Decision ( $\forall x: \text{fin } 7, \text{nfact } x\ 3$ ). (* g *)
```

↓

```
decision (all A (x\ app [P, x])) :- finite A, % r3  
  pi w\ decision (app [P, w]).
```

```
?- decision (all (app [c"fin", c"7"])  
  (x\ app [c"nfact", x, c"3"])). % g
```


Solving the goal in elpi

```
decision (all A (x\ app [P, x] )) :- finite A,           % r3
  pi w\ decision (app [P, w]).
```

```
?- decision (all (app [c"fin", c"7"] )                  % g
  (x\ app [c"nfact", x, c"3"] ) ).
```

What we propose

① Compilation:

- ▶ Recognize *problematic subterms* p_1, \dots, p_n
- ▶ Replace p_i with fresh unification variables X_i
- ▶ Link p_i with X_i

A link is a suspended unification problem

② Runtime:

- ▶ Unify p_i and X_i only when some conditions hold
- ▶ Decompile remaining links

The idea

```
decision (all A (x\ P' x)) :-                               % r3
  link P' (fun A (x\ app[P, x])),
  finite A,
  pi w\ decision (P' w).

?- decision (all (app ["fin", "7"])                          % g
               (x\ app [c"nfact", x, c"3"])).
```

Some notations

- \mathbb{P} : the unification problems in the object language (ol)
- \mathbb{Q} : the unification problems in the meta-language (ml)
- \mathbb{L}, \mathbb{M} : the link store, the unification-variable map

- $\text{run}_o(\mathbb{P}, n) \mapsto \rho$: the run of n unif pb in the ol
- $\text{run}_m(\mathbb{P}, n) \mapsto \rho'$: the run of n unif pb in the ml
- $\text{step}_o(\mathbb{P}, i, \rho_{i-1}) \mapsto \rho_i$: the execution of the i^{th} unif pb in ol
- $\text{step}_m(\mathbb{Q}, i, \sigma_{i-1}, \mathbb{L}_{i-1}) \mapsto (\sigma_i, \mathbb{L}_i)$: the exec of the i^{th} unif pb in ml

Proven properties

Run Equivalence $\forall \mathbb{P}, \forall n$, if $\mathbb{P} \subseteq \mathcal{L}_\lambda$

$$\text{run}_o(\mathbb{P}, n) \mapsto \rho \wedge \text{run}_m(\mathbb{P}, n) \mapsto \rho' \Rightarrow \forall s \in \mathbb{P}, \rho s =_o \rho' s$$

Simulation fidelity $\forall \mathbb{P}$, in the context of run_o and run_m , $\forall i \in 1 \dots n$,

$$\text{step}_o(\mathbb{P}, i, \rho_{i-1}) \mapsto \rho_i \Leftrightarrow \text{step}_m(\mathbb{Q}, i, \sigma_{i-1}, \mathbb{L}_{i-1}) \mapsto (\sigma_i, \mathbb{L}_i)$$

Compilation round trip If $\langle s \rangle \mapsto (t, m, l)$ and $l \in \mathbb{L}$ and $m \in \mathbb{M}$ and $\sigma = \{A \mapsto t\}$ and $X \mapsto A \in \mathbb{M}$ then

$$\langle \sigma, \mathbb{M}, \mathbb{L} \rangle^{-1} \mapsto \rho \text{ and } \rho X =_o \rho s.$$

Problematic subterm recognition

Sketch of $\diamond\beta$ terms : the problem

- An example: given a bound variable x

$$\begin{aligned}\mathbb{P} &= \{ Y \cdot x \simeq_o f \cdot x \cdot a \} \\ \mathbb{Q} &= \{ \text{app}[A, x] \simeq_m \text{app}[c\text{"f"}, x, c\text{"a"}] \} \\ \mathbb{M} &= \{ Y \mapsto A \}\end{aligned}$$

- Unification fails...

Sketch of $\diamond\beta$ terms : the solution

- An example, let x be a bound variable:

$$\begin{aligned}\mathbb{P} &= \{ Y \cdot x \simeq_o f \cdot x \cdot a \} \\ \mathbb{Q} &= \{ A \ x \simeq_m \text{app}[c\text{"f"}, x, c\text{"a"}] \} \\ \mathbb{M} &= \{ Y \mapsto A \} \end{aligned}$$

- Unification of \mathbb{Q}_0 gives: $\{A \mapsto (w \setminus \text{app}[c\text{"f"}, w, c\text{"a"}])\}$
- Decompilation of A gives $\{Y \mapsto \lambda x. f \cdot x \cdot a\}$

Sketch of $\diamond\eta$ terms

- $\lambda x.s \in \diamond\eta$, if $\exists \rho, \rho(\lambda x.s)$ is an η -redex
- Detection of $\diamond\eta$ terms is not trivial:

$$\lambda x.f.(A\ x) \qquad \in \diamond\eta \quad \rho = \{ A \mapsto \lambda x.x \}$$

$$\lambda x.f.(A\ x).x \qquad \in \diamond\eta \quad \rho = \{ A \mapsto \lambda x.a \}$$

- $\lambda x.\lambda y.f.(A\ x).(B\ y\ x) \in \diamond\eta \quad \rho = \{ A \mapsto \lambda x.x ; B \mapsto \lambda y.\lambda x.y \}$

$$\lambda x.f\ x.(A\ x) \qquad \notin \diamond\eta$$

Sketch of $\diamond\eta$ link : the problem

- An example:

$$\begin{aligned}\mathbb{P} &= \{ f \simeq_o \lambda x.(f.(Y.x)) \} \\ \mathbb{Q} &= \{ c\text{"f"} \simeq_m \text{fun } (x \backslash \text{app}[c\text{"f"}, B\ x]) \} \\ \mathbb{M} &= \{ Y \mapsto B \}\end{aligned}$$

- We have recognized the $\diamond\beta$ subterm $Y.x$
- But the unification problem in \mathbb{Q} raises a failure...

Sketch of $\diamond\eta$ link: the solution

- An example:

$$\begin{aligned}\mathbb{P} &= \{ f \simeq_o \lambda x. (f.(Y.x)) \} \\ \mathbb{Q} &= \{ c\textcolor{red}{f} \simeq_m A \} \\ \mathbb{M} &= \{ Y \mapsto B \} \\ \mathbb{L} &= \{ \vdash A =_\eta \text{fun } (x \backslash \text{app}[c\textcolor{red}{f}, B\ x]) \} \end{aligned}$$

- After unification of $c\textcolor{red}{f}$ with A ,
its η -expansion is unified with $\text{fun } (x \backslash \text{app}[c\textcolor{red}{f}, B\ x])$
Hence B is assigned to $x \backslash x$
- Decompilation will assign $\lambda x.x$ to Y

Sketch of $\diamond\mathcal{L}_\lambda$ links: the problem

- An example:

$$\begin{aligned}\mathbb{P} &= \{ Y \simeq_o \lambda x. a & (Y \cdot a) \simeq_o a \} \\ \mathbb{Q} &= \{ A \simeq_m \text{fun } (x \setminus c"a") & \text{app}[A, c"a"] \simeq_m c"a" \} \\ \mathbb{M} &= \{ Y \mapsto A \}\end{aligned}$$

- Note that $Y \cdot a$ is not a $\diamond\beta$: a is not a bound variable
- We can solve \mathbb{Q}_0 , and assign $\text{fun } (x \setminus c"a")$ to A
- However, we fail to solve $\mathbb{Q}_1 \dots$

Sketch of $\diamond\mathcal{L}_\lambda$ links: the solution

- An example:

$$\begin{aligned}\mathbb{P} &= \{ Y \simeq_o \lambda x.a & (Y.a) \simeq_o a \} \\ \mathbb{Q} &= \{ A \simeq_m \text{fun } (x \setminus c"a") & B \simeq_m c"a" \} \\ \mathbb{M} &= \{ Y \mapsto A \} \\ \mathbb{L} &= \{ \vdash B =_{\mathcal{L}_\lambda} A (c"a") \}\end{aligned}$$

- After unification of A with $\text{fun } (x \setminus c"a")$,
the rhs of the \mathcal{L}_λ -link becomes $c"a"$, after a β -reduction step,
the link is triggered and B is unified to $c"a"$
- Decompilation will assign $\lambda x.a$ to Y

Going further: the Constraint Handling Rules

- Elpi has CHR for goal suspension and resumption
- This fits well our notion of link: a suspended unification problem

```
pred link-eta i:term, i:term.  
link-eta A (fun _ _ B as T) :- not (var A), not (var B), !,  
    unify-left-right A T.  
link-eta A B :- progress-eta-right B B', !, A = B'.  
link-eta A B :- progress-eta-left A A', !, A' = B.  
link-eta A B :- scope-check A B, get-vars B Vars,  
    declare_constraint (link-eta A B) [A|Vars].
```

This can easily introduce new unification behaviors

- Add heuristic for HO unification outside the pattern fragment

```
% By def, R is not in the pattern fragment  
link-llam L R :- not (var L), unif-heuristic L R.
```

Conclusion

- Takes advantage of the unification capabilities of the meta language at the price of handling problematic sub-terms on the side.
- It is worth mentioning that we replace terms with variables only when it is strictly needed, leaving the rest of the term structure intact and hence **indexable**.
- Our approach is flexible enough to accommodate different strategies and **heuristics** to handle terms outside the pattern fragment

Thanks!

Thanks!

Questions ?