

# 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 HO term of Coq
  - ▶ But we want to make unification in Elpi
- Our contribution:
  - ▶ Reusing the meta-language unification for the object language

## A type-class problem in Coq

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

---

```
Goal Decision ( $\forall x$ : fin 7, 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
$f$	$c\text{"f"}$
$f \cdot a$	$\text{app}[c\text{"f"}, c\text{"a"}]$
$\lambda(x : T). F \cdot x$	$\text{fun } T \ (x \backslash \text{app}[F, x])$
$\forall(x : T), F \cdot x$	$\text{all } T \ (x \backslash \text{app}[F, x])$
$\dots$	$\dots$

- 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"])  
  (x\ app [c"nfact", x, c"3"] ))).           % g
```

# 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' (app[P, x]),
  finite A,
  pi w\ decision (P' x).

?- 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 map store
- 

- $\text{run}_o(\mathbb{P}, n) \mapsto \rho$ : 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  $i^{\text{th}}$  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  $i^{\text{th}}$  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  $\text{run}_o$  and  $\text{run}_m$ ,  $\forall i \in 1 \dots n$ ,

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

**Compilation round trip** If the compilation of  $s$  gives a term  $t$  and the stores  $\mathbb{L}$  and  $\mathbb{M}$  then

$$\langle \cdot, \mathbb{M}, \mathbb{L} \rangle^{-1} \mapsto \rho \wedge \rho t =_o s$$

## Problematic subterm recognition

## Sketch of $\diamond\beta$ links

- An example, let  $x$  be a bound variable:

$$\begin{aligned}\mathbb{P} &= \{ X \cdot x \simeq_o f \cdot x \cdot a \} \\ \mathbb{Q} &= \{ \textcolor{blue}{A} \ x \simeq_m \text{app}[\textcolor{green}{c}\textcolor{red}{f}, x, \textcolor{green}{c}\textcolor{red}{a}] \} \\ \mathbb{M} &= \{ X \mapsto \textcolor{blue}{A} \}\end{aligned}$$

- Unification of  $\mathbb{Q}_0$  gives:  $\{A \mapsto (\textcolor{blue}{w} \backslash \text{app}[\textcolor{green}{c}\textcolor{red}{f}, w, \textcolor{green}{c}\textcolor{red}{a}])\}$
- Decompilation of  $A$  gives  $\{X \mapsto \lambda y. f \cdot y \cdot a\}$

## Sketch of $\diamond\eta$ links

- $\lambda x.s \in \diamond\eta$ , if  $\exists \rho, \rho(\lambda x.s)$  is an  $\eta$ -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 resumption

- An example:

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

- After unification of  $A$  with  $c\text{"f"}$ , the lhs of the link becomes rigid and  $\text{fun } (x \backslash \text{app}[c\text{"f"}, B \ x])$  is unified with  $\text{fun } (x \backslash \text{app}[c\text{"f"}, x])$
- That is  $\{B \mapsto x \backslash x\}$
- Decompile will assign  $\lambda x. x$  to  $X$

## Sketch of $\diamond \mathcal{L}_\lambda$ links

- An example:

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

- After unification of  $A$  with  $\text{fun } (x \backslash c"a")$ ,  
the rhs of the  $\mathcal{L}_\lambda$ -link becomes  $c"a"{}^1$ ,  
the link is triggered and  $B$  is unified to  $c"a"$
- Decompile will assign  $\lambda x.a$  to  $X$

---

<sup>1</sup>after a  $\beta$ -reduction step

## Going further: the Constraint Handling Rules

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

This can easily introduce new unification behaviors

- We can for example mimic the unification of the ol
  - 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