Size-preserving dependent elimination

Hugo Herbelin

TYPES 2024

10 June 2024

The guard checker of Coq

The Calculus of Inductive Constructions currently implemented in Coq relies on case analysis + guarded recursion. E.g. dependent system T recursor is not primitive but defined by:

```
Section NatRec.

Context (P : nat -> Type) (f : P 0) (f0 : forall n : nat, P n -> P (S n)).
Definition nat_rect :=
  fix F (n : nat) : P n :=
  match n return (P n) with
  | 0 => f
  | S n0 => f0 n0 (F n0)
  end.

End NatRec.
```

Then, a *guard criterion* ensures well-foundedness.

See Bruno Barras' slides "The syntactic guard condition of Coq" (https://coq.inria.fr/files/adt-2fev10-barras.pdf), 2010.

Guard: the basic idea

$$\frac{\begin{array}{l}\Gamma, x : I \ \mathsf{ok} \\ \Gamma, (f : \Pi x : I.\, U), x : I \vdash u : U \\ \Gamma \,|\, f \,|\, x \,|\, \vdash u \,|\ \ \mathsf{guarded}\end{array}}{\Gamma \vdash (\mathsf{fix}\, f(x : I) : U := u) : \Pi x : I.\, U}$$

based on a special kind of judgement

$$\Gamma \,|\, f \,|\, x \,|\, \Xi \vdash u \,|\, \pi \ \mathsf{guarded}$$

where:

- Ξ is a list of variables of Γ known to be of strictly smaller size than the main argument x of f

- π is a stack of arguments applied to u, of the form $t_1 \cdot \ldots \cdot t_n$

First key case

The first key case is when the recursive function is applied:

$$\frac{\Gamma \,|\, \Xi \vdash t \ \textsf{smaller}}{\Gamma \,|\, f \,|\, x \,|\, \Xi \vdash f \,|\, t \cdot \pi \ \textsf{guarded}}$$

It relies on the auxiliary judgement

$$\Gamma \,|\, \Xi \vdash u \ \textsf{smaller}$$

whose main inference rule is

$$\frac{y \in \Xi}{\Gamma \,|\, \Xi \vdash y \ \textsf{smaller}}$$

Second key case

The second key case is when we traverse a case analysis:

$$
\frac{\begin{array}{ll}
\Gamma \,|\, f \,|\, x \,|\, \Xi \vdash c \,|\, \textsf{guarded} & \\
\Gamma, \overrightarrow{y : U}, y' : J\ \overrightarrow{y} \,|\, f \,|\, x \,|\, \Xi \vdash P \,|\, \textsf{guarded} & \text{where } \Xi' \text{ is } \Xi, |\overrightarrow{z_k}|_I \text{ if } c \text{ is } x \\
\Gamma, \overrightarrow{z_k : V_k} \,|\, f \,|\, x \,|\, \Xi' \vdash u_k \,|\, \pi\ \textsf{guarded} & \text{or } \Gamma \,|\, \Xi \vdash c\ \textsf{smaller} \text{ and } \Xi \text{ otherwise}
\end{array}}{\Gamma \,|\, f \,|\, x \,|\, \Xi \vdash \textsf{match}\ c\ \textsf{as}\ y\ \textsf{in}\ J\ (\overrightarrow{y : U})\ \textsf{return}\ P\ \textsf{with}\ C_k(\overrightarrow{z_k : V_k}) \Rightarrow u_k\ \textsf{end} \,|\, \pi\ \textsf{guarded}}
$$

It can be a case analysis:

- either on a smaller term c, in which case, its subterms are recursively declared smaller

- or an arbitrary term

In both cases, the stack of argument traverses the case analysis.

A quick history of the evolution of the guard checker

- Initial implementation in Coq V5.10.2 (1994)
- Propagation of smallerness through inner fixpoints applied to a smaller argument (in the 90's)
- Support for nested fixpoints (in the 90's)
- Smallerness traverses β-redexes blocked by a case analysis ("pseudo-commutative cuts") (2010)
- Restore compatibility with propositional extensionality (2014)
- Guard criterion ensures strong normalisation (2022, PR #15434)
- Extrusion of uniform parameters of fixpoints in nested fixpoints (2024, PR #17986)

Incompatibility with propositional extensionality

Propagation of smallerness across pseudo-commutative cuts refuted propositional extensionality:

```
Axiom prop_ext: forall {P Q}, (P <-> Q) -> P=Q.

Inductive True2 : Prop := C2 : (False -> True2) -> True2.

Theorem T2T: True2 = True.
Proof. exact (prop_ext (conj (fun _ => I) (fun _ => C2 (False_rect True2)))). Qed.

Theorem T2F_FT2F : (True2 -> False) = ((False -> True2) -> False).
Proof. rewrite T2T; apply prop_ext; split; auto. Qed.

Fixpoint loop (x : True2) : False :=
 match x with
 | C2 f => (match T2F_FT2F in _=T return T with eq_refl => fun f => loop f end) f
 end.
```

The remedy: deactivate the contribution of the indices of the term being matched to the preservation of the size.

Restoring compatibility with propositional extensionality

The smaller variables in Ξ can now be restricted + new rule to traverse case analysis that deactivates some recursive occurrences:

$$\frac{\begin{array}{ll} \Gamma \,|\, f \,|\, x \,|\, \Xi \vdash c \,|\, \textsf{guarded} & \\ \Gamma, \overrightarrow{y : U}, y' : J\,\overrightarrow{y} \,|\, f \,|\, x \,|\, \Xi \vdash P \,|\, \textsf{guarded} & \text{where } \Xi' \text{ is } \Xi, |\overrightarrow{z_k : V_k}|_{W} \text{ if } c \text{ is } x \\ \Gamma \,|\, f \,|\, x \,|\, \Xi' \vdash [u_k]_{P[\overrightarrow{y := \bot}, y' := \bot]} \,|\, \pi \ \textsf{guarded} & \text{or } \Gamma \,|\, \Xi \vdash c\ W\textsf{-smaller}, \text{ and } \Xi \text{ otherwise} \end{array}}{\Gamma \,|\, f \,|\, x \,|\, \Xi \vdash \textsf{match}\ c\ \textsf{as}\ y\ \textsf{in}\ J\ (\overrightarrow{y : U})\ \textsf{return}\ P\ \textsf{with}\ C_k(\overrightarrow{z_k : V_k}) \Rightarrow u_k\ \textsf{end} \,|\, \pi\ \textsf{guarded}}$$

where

- $[u]_P$ propagates the domains of dependent products in P to the corresponding domain of λ's in u, if any, with the result of discarding, according to the elimination predicate, all possible contributions of the indices to guardedness in the domain of λ's

- $|\overrightarrow{z_k : V_k}|_W$ restricts the recursive occurrences of I in the V_k's according to the restriction made in W

Note: a similar treatment has to be done when smallerness traverses case analysis (not shown in the talk).

Limitations of the 2014 remedy

- Deactivate all indices while only indices contributing to propogating type constraints are really problematic (the "univalence"-as-coercions intuition).

- Breaks the generality of Monin-Boutillier's compilation of pattern-matching by small inversion.

New proposed remedy:

$$\frac{\begin{array}{ll} \Gamma \,|\, f \,|\, x \,|\, \Xi \vdash c \,|\, \textsf{guarded} & \\ \Gamma, \overrightarrow{y : U}, y' : J\, \overrightarrow{y} \,|\, f \,|\, x \,|\, \Xi \vdash P \,|\, \textsf{guarded} & \text{where } \Xi' \text{ is } \Xi, |\overrightarrow{z_k : V_k}|_W \text{ if } c \text{ is } x \\ \Gamma \,|\, f \,|\, x \,|\, \Xi' \vdash [u_k]_{P[\overrightarrow{y := \downarrow v_k}, y' := C_k(\overrightarrow{z_k})]} \,|\, \pi \; \textsf{guarded} & \text{or } \Gamma \,|\, \Xi \vdash c\; W\textsf{-smaller}, \text{ and } \Xi \text{ otherwise} \end{array}}{\Gamma \,|\, f \,|\, x \,|\, \Xi \vdash \textsf{match}\; c \;\textsf{as}\; y \;\textsf{in}\; J\, (\overrightarrow{y : U}) \;\textsf{return}\; P \;\textsf{with}\; C_k(\overrightarrow{z_k : V_k}) \Rightarrow u_k \;\textsf{end} \,|\, \pi \; \textsf{guarded}}$$

where $\downarrow v$ masks only type arguments.

Conjectured to still be compatible with propositional extensionality.

Another application

Coq supports an optional "definitional uip" reduction rule in the universe of impredicative strict propositions (`SProp`):

$$\frac{t \equiv u}{(\textsf{match } e : t = u \textsf{ as } y \textsf{ in } \_ = z \textsf{ return } P \textsf{ with refl } \Rightarrow v \textsf{ end}) \to v}$$

which is known to break normalisation, by Abel-Coquand 2020.

We conjecture that the only source of non-normalisation is when the predicate P rewrites subterms of type an impredicate universe, so that the following restriction would preserve normalisation:

$$\frac{t \equiv u \qquad P[z := \Downarrow u][y := \Downarrow e] \textsf{ is } \bot\textsf{-free}}{(\textsf{match } e : t = u \textsf{ as } y \textsf{ in } \_ = z \textsf{ return } P \textsf{ with refl } \Rightarrow v \textsf{ end}) \to v}$$

where $\Downarrow$ masks subterms in an impredicative universe.