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.

Guard: the basic idea

\[ \Gamma, x : I \text{ ok} \]
\[ \Gamma, (f : \Pi x : I. U), x : I \vdash u : U \]
\[ \Gamma \parallel f \parallel x \parallel \vdash u \parallel \text{ guarded} \]
\[ \Gamma \vdash (\text{fix } f(x : I) : U := u) : \Pi x : I. U \]

based on a special kind of judgement

\[ \Gamma \parallel f \parallel x \parallel \Xi \vdash u \parallel \pi \text{ guarded} \]

where:
- \( \Xi \) is a list of variables of \( \Gamma \) known to be of strictly smaller size than the main argument \( x \) of \( f \)
- \( \pi \) 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:

$$\Gamma \vdash t \text{ smaller}$$

$$\Gamma \vdash f \mid x \mid \Xi \vdash f \mid t \cdot \pi \text{ guarded}$$

It relies on the auxiliary judgement

$$\Gamma \vdash u \text{ smaller}$$

whose main inference rule is

$$y \in \Xi$$

$$\Gamma \vdash y \text{ smaller}$$
Second key case

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

\[ \Gamma | f | x | \Xi \vdash c | \text{guarded} \]
\[ \Gamma, y : \overrightarrow{U}, y' : J \overrightarrow{y} | f | x | \Xi \vdash P | \text{guarded} \]
\[ \text{where } \Xi' \text{ is } \Xi, |z_k^y| I \text{ if } c \text{ is } x \]
\[ \Gamma, z_k : V_k | f | x | \Xi' \vdash u_k | \pi \text{ guarded} \]
\[ \text{or } \Gamma | \Xi \vdash c \text{ smaller and } \Xi \text{ otherwise} \]

\[ \Gamma | f | x | \Xi \vdash \text{match } c \text{ as } y \text{ in } J (\overrightarrow{y : U}) \text{ return } P \text{ with } C_k(z_k : V_k) \Rightarrow u_k \text{ end} | \pi \text{ guarded} \]

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 $\beta$-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 smallness across pseudo-commutative cuts refutes propositional extensionality:

Axiom prop_ext: \( \forall \{P \ Q\}, (P \leftrightarrow Q) \rightarrow P=Q \).

Inductive True2 : Prop := C2 : (False \rightarrow True2) \rightarrow True2.

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

Theorem T2F_FT2F : (True2 \rightarrow False) = ((False \rightarrow True2) \rightarrow 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 $\Xi$ can now be restricted + new rule to traverse case analysis that deactivates some recursive occurrences:

$\Gamma \vdash f \mid x \mid \Xi \vdash c \mid \text{guarded}$

$\Gamma, y : U, y' : J \vdash f \mid x \mid \Xi \vdash P \mid \text{guarded}$

where $\Xi'$ is $\Xi, |z_k : V_k|_W$ if $c$ is $x$

$\Gamma \vdash f \mid x \mid \Xi' \vdash [u_k]_{P[y := \bot, y' := \bot]} \mid \pi \text{ guarded}$

or $\Gamma \vdash \Xi \vdash c \text{ $W$-smaller}, \text{ and } \Xi \text{ otherwise}$

\[
\Gamma \vdash f \mid x \mid \Xi \vdash \text{match } c \text{ as } y \text{ in } J (y : U) \text{ return } P \text{ with } C_k(z_k : V_k) \Rightarrow u_k \text{ end } \mid \pi \text{ guarded}
\]

where

- $[u]_P$ propagates the domains of dependent products in $P$ to the corresponding domain of $\lambda$'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 $\lambda$'s

- $|z_k : V_k|_W$ 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 propagating 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:

\[
\begin{align*}
\Gamma \mid f \mid x \mid \Xi \vdash c \mid \text{guarded} \\
\Gamma, y : U, y' : J \vdash f \mid x \mid \Xi \vdash P \mid \text{guarded} & \quad \text{where } \Xi' \text{ is } \Xi, \mid z_k : V_k \mid W \text{ if } c \text{ is } x \\
\Gamma \mid f \mid x \mid \Xi' \vdash [u_k]_{P[y := \downarrow v_k, y' := C_k(z_k)]} \mid \pi \mid \text{guarded} & \quad \text{or } \Gamma \mid \Xi \vdash c \ W\text{-smaller, and } \Xi \text{ otherwise} \\
\Gamma \mid f \mid x \mid \Xi \vdash \text{match } c \text{ as } y \text{ in } J \left( y : U \right) \text{ return } P \text{ with } C_k(z_k : V_k) \Rightarrow u_k \text{ end } \mid \pi \mid \text{guarded}
\end{align*}
\]

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):

\[
\frac{t \equiv u}{(\text{match } e : t = u \text{ as } y \text{ in } _ = z \text{ return } P \text{ with refl } \Rightarrow v \text{ end}) \rightarrow v}
\]

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 impredicative universe, so that the following restriction would preserve normalisation:

\[
\frac{t \equiv u \quad P[z := \downarrow u][y := \downarrow e] \text{ is } \bot\text{-free}}{(\text{match } e : t = u \text{ as } y \text{ in } _ = z \text{ return } P \text{ with refl } \Rightarrow v \text{ end}) \rightarrow v}
\]

where \( \downarrow \) masks subterms in an impredicative universe.