Size-preserving dependent elimination

Hugo Herbelin

IRIF, CNRS, Université de Paris, Inria, France
hugo.herbelin@inria.fr

Elimination rules of positive connectives refine the context of a proof or computation with the different possible values of the object being eliminated, such as being on one side or the other of a disjunction or such as being a tuple of objects.

Sometimes, we may require that properties from the object being eliminated do not pervade in the context. A typical example is when the object being eliminated is up to a quotient, and we want the proof or computation to preserve the quotient.

Here, we consider another form of restriction intended to ensure that the elimination does not change the "size" of the proof or computation.

The motivation is multiple. First, it has been discovered in 2013 that the guard condition implemented in Coq was inconsistent with propositional extensionality and the analysis of the incompatibility revealed that transporting an expression along propositional extensionality had indeed the ability of changing the size of the expression, an observation which is reminiscent of the homotopic interpretation of type extensionality as an equivalence. A restriction of the conditions under which an elimination propagates the size was then implemented to prevent the inconsistency.

On the other side, while implementing a small-inversion-based variant [MS13] of Goguen-McBride-McKinna's and Cockx' compilation [GMM06, Coc17, Soz10] of dependent pattern-matching [Coq92], it was observed that the new criterion was too restrictive [Mar17]. Other limitations were also reported on the Coq bug tracker [Soz23].

We conjecture the existence of a compromise that exactly captures the restriction needed to preserve the compatibility with propositional extensionality while resolving the known limitations.

First, we briefly recall the main parts[1] of the guardedness check in the variant of the Calculus of Inductive Constructions implemented in Coq for fixpoints up to 2013 (i.e. up to Coq 8.4). We call *stack* any sequence $t_1 \cdot \ldots \cdot t_n$ of terms and say that a well-typed recursive expression $\mathit{fix}\, f\,(x : I) : T := c$ is *guarded in context* Γ when c is guarded for f on x in $\Gamma, f : \Pi(x : I).T, x : I$ relatively to an empty set of smaller variables and empty stack where c *is guarded for* f *on* x *in* Γ *relatively to a set* Ξ *of smaller variables and stack* π, shortly $\Gamma \mid f \mid x \mid \Xi \vdash c \mid \pi$ guarded, is defined by means of an auxiliary judgement $\Gamma \mid f \mid x \mid \Xi \vdash c$ smaller and the following excerpt of inference rules[2]:

$$\frac{y \in \Xi}{\Gamma \mid \Xi \vdash y \text{ smaller}} \qquad \frac{\Gamma \mid \Xi \vdash t \text{ smaller}}{\Gamma \mid \Xi \vdash t\, u \text{ smaller}} \qquad \frac{\Gamma \mid \Xi \vdash t \text{ smaller}}{\Gamma \mid f \mid x \mid \Xi \vdash f \mid t \text{ guarded}}$$

$$\frac{\Gamma, y : T \mid f \mid x \mid \Xi, y \vdash u \mid \pi \text{ guarded} \qquad \Gamma \mid f \mid x \mid \Xi \vdash t \text{ smaller}}{\Gamma \mid f \mid x \mid \Xi \vdash \lambda(y : T).\, u \mid t \cdot \pi \text{ guarded}} \qquad \frac{\Gamma, y : T \mid f \mid x \mid \Xi \vdash u \mid \pi \text{ guarded}}{\Gamma \mid f \mid x \mid \Xi \vdash \lambda(y : T).\, u \mid t \cdot \pi \text{ guarded}}$$

$$\frac{\Gamma, y : T \mid f \mid x \mid \Xi \vdash u \mid \text{ guarded}}{\Gamma \mid f \mid x \mid \Xi \vdash \lambda(y : T).\, u \mid \text{ guarded}} \qquad \frac{\Gamma \mid f \mid x \mid \Xi \vdash t \mid u \cdot \pi \text{ guarded}}{\Gamma \mid f \mid x \mid \Xi \vdash t\, u \mid \pi \text{ guarded}}$$

$$\frac{\begin{array}{l}\Gamma \mid f \mid x \mid \Xi \vdash c \mid \text{ guarded} \\ \Gamma, \overrightarrow{y : U}, y' : J\ \overrightarrow{y} \mid f \mid x \mid \Xi \vdash P \mid \text{ guarded} \\ \Gamma, \overrightarrow{z_k : V_k} \mid f \mid x \mid \Xi' \vdash u_k \mid \pi \text{ guarded}\end{array}}{\Gamma \mid f \mid x \mid \Xi \vdash \mathsf{match}\ c\ \mathsf{as}\ y\ \mathsf{in}\ J\ (\overrightarrow{y : U})\ \mathsf{return}\ P\ \mathsf{with}\ C_k(\overrightarrow{z_k : V_k})\ \Rightarrow u_k\ \mathsf{end} \mid \pi \text{ guarded}}$$

where Ξ' is $\Xi, |\overrightarrow{z_k}|_I$ if c is x or $\Gamma \mid \Xi \vdash c$ smaller and Ξ otherwise

where, in the Coq-like-syntax-based rule for dependent elimination, J is an inductive family and $|\overrightarrow{z_k}|_I$ selects the subset of $\overrightarrow{z_k}$ whose type ends with a recursive occurrence of I in the type of the corresponding constructor.

[1] Intentionally simplified to focus on the point under concern.

[2] The role of the stack is to propagate smallerness from outside of a case analysis to the branches of the case analysis.

From Coq 8.5, the guard was modified to prevent the inconsistency with propositional extensionality [Col14]. A way to present the change is by introducing an "undefined" term $\bot$ serving to mask the parts of the recursive structure of an inductive type that should not be considered as smaller. Then, if T and U possibly contain $\bot$ at some places, we write $T \cap U$ for the type obtained by setting $\bot$ at every occurrence where T and U differ (or are both already $\bot$). We also write $T \leq U$ is T has $\bot$ at all occurrences where U has already $\bot$. Even though the purpose of the inference rules is not to check typing (which is assumed to be done beforehand), the smaller variables now carry a type, leading to a judgement of the form $\Gamma \mid f \mid x \mid x_1 : T_1, ..., x_n : T_n \vdash c \mid \pi$ guarded, with π a sequence of typed terms.

The inference rules are then changed as follows:

$$\frac{y : U \in \Xi}{\Gamma \mid f \mid \Xi \vdash y \ U\text{-smaller}} \qquad \frac{\Gamma \mid \Xi \vdash t \ U\text{-smaller}}{\Gamma \mid \Xi \vdash t\,u \ U\text{-smaller}} \qquad \frac{\Gamma \mid \Xi \vdash t \ U\text{-smaller} \qquad I \leq U}{\Gamma \mid f \mid x \mid \Xi \vdash f \mid t \text{ guarded}}$$

$$\frac{\Gamma, y : T \mid f \mid x \mid \Xi, y : T \cap U \vdash u \mid \pi \text{ guarded} \qquad \Gamma \mid f \mid x \mid \Xi \vdash t \ U\text{-smaller}}{\Gamma \mid f \mid x \mid \Xi \vdash \lambda(y : T).\, u \mid t \cdot \pi \text{ guarded}}$$

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

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.

We now come to our refinement of the 2013 criterion [Her21]. The idea is not to restrict all dependencies in the indices of the family in the elimination predicate, but only the dependencies in indices that are types, or type families, that is, for t a term, we define its *size-preserving mask* $\downarrow t$ compositionally except that all occurrences of an inductive type or inductive family J in t are replaced by $\bot$, since, after all, only inductive types are able to contribute to smallerness, so it is enough to mask inductive types. Then, the rule for dependent case analysis is refined into (where v_k are the indices in the conclusion of the type of C_k):

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

Our main conjecture is then:

Conjecture: Propositional extensionality is not refutable in the Calculus of Inductive Constructions equipped with the modified rule.

Additionally, we conjecture that the same kind of idea would support convertibility-based elimination of equality proofs $t = u$ in strict propositions without leading to a failure of normalisation in the presence of an impredicative sort [AC20]: instead of masking inductive types in the indices, our guess is that the following reduction rule, where $\Downarrow$ masks subterms of type an impredicative sort, preserves normalisation[3]:

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

The motivation for our guess is that the example in [AC20] crucially relies on eliminating a proof of extensional equality, which the modified reduction rule prevents.

[3] In Coq, this rule, without our restriction, is activated by setting the option Definitional UIP.

References

[AC20] Andreas Abel and Thierry Coquand. Failure of normalization in impredicative type theory with proof-irrelevant propositional equality. *Log. Methods Comput. Sci.*, 16(2), 2020.

[Coc17] Jesper Cockx. *Dependent Pattern Matching and Proof-Relevant Unification*. PhD thesis, KU Leuven, 2017.

[Col14] Collective. Coq modified guard checker. https://github.com/coq/coq/blob/master/kernel/inductive.ml, starting with commit 35e47b6be8d9e, 2014.

[Coq92] Thierry Coquand. Pattern Matching with Dependent Types. In B. Nordström, K. Petersson, and G. Plotkin, editors, *Proceedings of the 1992 Workshop on Types for Proofs and Programs*, 1992.

[GMM06] Healfdene Goguen, Conor McBride, and James McKinna. Eliminating dependent pattern matching. In Kokichi Futatsugi, Jean-Pierre Jouannaud, and José Meseguer, editors, *Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday*, volume 4060 of *Lecture Notes in Computer Science*, pages 521–540. Springer, 2006.

[Her21] Hugo Herbelin. Optimizing the propagation of the guard condition under a match on a type with indices by discarding only the type information from the indices. https://github.com/coq/coq/pull/14359, 2021.

[Mar17] Thierry Martinez. Fixpoint guard when generalization occurs on strict subterms. https://github.com/coq/coq/issues/5530, 2017.

[MS13] Jean-François Monin and Xiaomu Shi. Handcrafted inversions made operational on operational semantics. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings*, volume 7998 of *Lecture Notes in Computer Science*, pages 338–353. Springer, 2013.

[Soz10] Matthieu Sozeau. Equations: A dependent pattern-matching compiler. In *ITP*, volume 6172 of *Lecture Notes in Computer Science*, pages 419–434. Springer, 2010.

[Soz23] Matthieu Sozeau. Guard checking regression. https://github.com/coq/coq/issues/17062, 2023.