FAILURE OF NORMALIZATION IN IMPREDICATIVE TYPE THEORY WITH PROOF-IRRELEVANT PROPOSITIONAL EQUALITY

ANDREAS ABEL AND THIERRY COQUAND

Department for Computer Science and Engineering, Chalmers and Gothenburg University
e-mail address: andreas.abel@cse.gu.se

Department for Computer Science and Engineering, Chalmers and Gothenburg University
e-mail address: thierry.coquand@cse.gu.se

Abstract. Normalization fails in type theory with an impredicative universe of propositions and a proof-irrelevant propositional equality. The counterexample to normalization is adapted from Girard's counterexample against normalization of System F equipped with a decider for type equality. It refutes Werner's normalization conjecture [LMCS 2008].

Introduction

Type theories with an impredicative universe Prop of propositions, such as the Calculus of Constructions (Coquand and Huet, 1988), lose the normalization property in the presence of a *proof-irrelevant* propositional equality $\_{=}\_ : \Pi\, A : \mathsf{Type}.\ A \to A \to \mathsf{Prop}$ with the standard elimination principle. The loss of normalization is facilitated already by a coercion function with a reduction rule

$$\begin{array}{l} \mathsf{cast} : \Pi\, A\, B : \mathsf{Prop}.\ A =_{\mathsf{Prop}} B \to A \to B \\ \mathsf{cast}\ A\ A\ e\ x \rhd x \end{array}$$

that does not inspect the equality proof $e : A =_{\mathsf{Prop}} A$ but only checks whether the endpoints are (definitionally) equal.

The failure of normalization refutes a conjecture by Werner (2008, Conjecture 3.14). Consistency and canonicity is not at stake; thus, the situation is comparable to type theory with equality reflection (Martin-Löf, 1984a,b), aka *Extensional Type Theory*. At the moment, it is unclear whether the use of impredicativity is essential to break normalization; predicative type theory might be able to host a proof-irrelevant propositional equality (Abel, 2009) while retaining normalization.

Key words and phrases: impredicativity, normalization, proof-irrelevance, propositional equality.

Preprint submitted to
Logical Methods in Computer Science

Counterexample to Normalization

We employ the usual impredicative definition of absurdity $\bot$ and negation $\neg A$ and a derived definition of truth $\top$:

$$\begin{array}{lcl} \bot & : & \mathsf{Prop} \\ \bot & = & \Pi\, A : \mathsf{Prop}.\ A \end{array} \qquad \begin{array}{lcl} \neg & : & \mathsf{Prop} \to \mathsf{Prop} \\ \neg A & = & A \to \bot \end{array} \qquad \begin{array}{lcl} \top & : & \mathsf{Prop} \\ \top & = & \neg \bot \end{array}$$

The presence of cast allows us to define self-application under the assumption that all propositions all equal. The self-application term ω refutes this assumption.

$$\begin{array}{lcl} \delta & : & \top \\ \delta\ z & = & z\ \top\ z \end{array} \qquad \begin{array}{lcl} \omega & : & \neg \Pi\, A\, B : \mathsf{Prop}.\ A =_{\mathsf{Prop}} B \\ \omega\ h\ A & = & \mathsf{cast}\ \top\ A\ (h\ \top\ A)\ \delta \end{array}$$

Impredicativity is exploited in δ when applying $z : \bot$ to type $\top = \bot \to \bot$ so that it can be applied to z again. The type of δ is $\top$ which we cast to A thanks to the assumption h that all propositions are equal.

We build a non-normalizing term Ω by applying ω to itself through δ, reminiscent of the shortest diverging term in untyped λ-calculus.

$$\begin{array}{lcl} \Omega & : & \neg \Pi\, A\, B : \mathsf{Prop}.\ A =_{\mathsf{Prop}} B \\ \Omega\ h & = & \delta\ (\omega\ h) \end{array}$$

Thanks to the reduction rule of cast, term $\Omega\ h$ reduces to itself:

$$\begin{array}{lclcl} \Omega\ h & = & \delta\ (\omega\ h) & \rhd & \omega\ h\ \top\ (\omega\ h) \\ & = & \mathsf{cast}\ \top\ \top\ (h\ \top\ \top)\ \delta\ (\omega\ h) & \rhd & \delta\ (\omega\ h) \\ & = & \Omega\ h \end{array}$$

Thus, normalization is lost in the presence of a hypothesis (free variable) h. As a consequence, normalization that proceeds under λ-abstraction can diverge. This means that equality of open terms cannot be decided just by normalization.

The counterexample can be implemented in Werner's type theory with proof-irrelevance (2008), refuting the normalization conjecture (3.14). We implement cast as instance of Werner's more general equality elimination rule:

$$\begin{array}{lcl} \mathsf{Eq\_rec} & : & \Pi\, A : \mathsf{Type}.\ \Pi\, P : A \to \mathsf{Type}.\ \Pi\, a\, b : A.\ P\ a \to a =_A b \to P\ b \\ \mathsf{Eq\_rec}\ A\ P\ a\ b\ x\ e & \rhd & x \qquad \text{if } a = b \\[1ex] \mathsf{cast}\ A\ B\ e\ x & = & \mathsf{Eq\_rec}\ \mathsf{Prop}\ (\lambda\, a{:}A.\ \mathsf{Prop})\ A\ B\ x\ e \end{array}$$

The term Ω also serves as counterexample to normalization in the theorem prover Lean (de Moura et al., 2015), version 3.4.2 (Microsoft Research, 2019).

```
def False := ∀ A : Prop, A
def Not   := λ A, A → False
def True  := Not False

def delta :  True                        := λ z, z True z
def omega :  Not (∀ A B : Prop, A = B) := λ h A, cast (h True A) delta
def Omega :  Not (∀ A B : Prop, A = B) := λ h, delta (omega h)
```

Infinite reduction can now be triggered with the command `#reduce Omega`, which diverges.

A Counterexample Using Propositional Extensionality

The counterexample of the last section used the absurd assumption that *all* proposition are equal. The following counterexample utilizes just the axiom of propositional extensionality, $\mathsf{propext}$, which is a default axiom of Lean. In fact, the weaker statement $\mathsf{tautext}$, which states the equality of *true* propositions, is sufficient.

$$\begin{array}{lll} \mathsf{propext} & : & \Pi\, A\, B : \mathsf{Prop}.\ (A \leftrightarrow B) \to A =_{\mathsf{Prop}} B \\ \mathsf{tautext} & : & \Pi\, A\, B : \mathsf{Prop}.\ A \to B \to A =_{\mathsf{Prop}} B \end{array}$$

The counterexample uses the standard impredicative definition of truth,

$$\top \quad = \quad \Pi\, A : \mathsf{Prop}.\ A \to A$$

and a cast from $\top \to \top$ to A, which are both tautologies under the assumption $a : A$.

$$\begin{array}{lllllll} \mathsf{id},\ \delta & : & \top \to \top & \qquad & \omega,\ \Omega & : & \top \\ \mathsf{id}\ x & = & x & & \omega\ A\ a & = & \mathsf{cast}\ (\top \to \top)\ A\ (\mathsf{tautext}\ (\top \to \top)\ A\ \mathsf{id}\ a)\ \delta \\ \delta\ z & = & z\ (\top \to \top)\ \mathsf{id}\ z & & \Omega & = & \delta\ \omega \end{array}$$

These definitions can be directly replayed in Lean 3.4.2 with the standard prelude, yielding a non-normalizing term `Omega`.

```
def tautext {A B : Prop} (a : A) (b : B)
         : A = B        := propext (iff.intro (λ _, b) (λ _, a))
def True  : Prop        := ∀ A : Prop, A → A

def delta : True → True := λ z : True, z (True → True) id z
def omega : True        := λ A a, cast (tautext id a) delta
def Omega : True        := delta omega
```

Note that term `Omega` is closed with respect the standard axioms of Lean, and does not even have a weak head normal form.

Related Work and Conclusions

The cast operator is inspired by Girard's operator $J : \Pi\, A\, B : \mathsf{Prop}.\ A \to B$ with reduction rule $J\ A\ A\ M \rhd M$ that destroys the normalization property of System F (Girard, 1971; Harper and Mitchell, 1999). In contrast to J, our cast also requires a proof of equality of A and B, but this proof is not inspected and thus does not block reduction if it is non-canonical. Thus, the simple lie that all propositions are equal is sufficient to trigger divergence.

Historically, the Automath system AUT-4 is maybe the first type-theoretic proof assistant to feature proof-irrelevant propositions (de Bruijn, 1994). The terminology used by de Bruijn is *fourth degree identification*, where proofs are expressions considered to have degree 4, propositions and values degree 3, types and Prop degree 2, and the universe Type of types degree 1.

Lean's type theory (Carneiro, 2019) features an impredicative universe of proof-irrelevant propositions which hosts both propositional equality and the accessibility predicate (Aczel, 1977, 1.2). As both may be eliminated into computational universes, decidability of definitional equality is lost, as demonstrated by Carneiro (2019) for the case of accessibility. As a consequence, typing is not decidable.

The type-theoretic proof assistants Agda and Coq have recently (Gilbert et al., 2019) been equipped with a proof-irrelevant universe of propositions ("strict Prop"). In this universe, propositional equality can be defined, but cannot be eliminated into types that are not strict propositions themselves. Under this restriction, Gilbert (2019, 4.3) formally proved normalization and decidability of type checking for the predicative case.

Several open problems remain:

(1) Does the theory with impredicative strict Prop have normalization and decidability of type checking as well?
(2) Does the addition of Werner's rule, while destroying proof normalization, preserve decidability of conversion and type checking? (Since proofs are irrelevant for equality, they need not be normalized during type checking.)
(3) Does Werner's rule preserve normalization in the predicative case? (Our counterexamples make use of impredicativity.)

Acknowledgments.

The authors acknowledge support by the Swedish Research Council (Vetenskapsrådet) under grants 2014-04864 *Termination Certificates for Dependently-Typed Programs and Proofs via Refinement Types* and 2017-04064 *Syntax and Semantics of Univalent Type Theory*. Our research group is part of the EU Cost Action CA15123 The European research network on types for programming and verification (EUTypes). We thank Mario Carneiro for contributing the original Lean implementation of the first counterexample.

References

A. Abel. Extensional normalization in the logical framework with proof irrelevant equality. In O. Danvy, editor, *Workshop on Normalization by Evaluation, affiliated to LiCS 2009, Los Angeles, 15 August 2009*, 2009. URL `http://www.cse.chalmers.se/~abela/nbe09.pdf`.

P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, pages 739–782. Elsevier, 1977. URL `https://doi.org/10.1016/S0049-237X(08)71120-0`.

M. Carneiro. The type theory of Lean. Master's thesis, Department of Philosophy, Carnegie Mellon University, 2019. URL `https://github.com/digama0/lean-type-theory`.

T. Coquand and G. P. Huet. The calculus of constructions. *Information and Computation*, 76(2/3):95–120, 1988. URL `https://doi.org/10.1016/0890-5401(88)90005-3`.

N. de Bruijn. Some extensions of Automath: The AUT-4 family. In R. Nederpelt, J. Geuvers, and R. de Vrijer, editors, *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics*, pages 283–288. Elsevier, 1994. URL `https://doi.org/10.1016/S0049-237X(08)70209-X`.

L. M. de Moura, S. Kong, J. Avigad, F. van Doorn, and J. von Raumer. The Lean theorem prover (system description). In A. P. Felty and A. Middeldorp, editors, *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 378–388. Springer, 2015. URL `https://doi.org/10.1007/978-3-319-21401-6_26`.

G. Gilbert. *A type theory with definitional proof-irrelevance*. PhD thesis, École Nationale Supérieure Mines-Télécom Atlantique, 2019.

G. Gilbert, J. Cockx, M. Sozeau, and N. Tabareau. Definitional proof-irrelevance without K. *Proceedings of the ACM on Programming Languages*, 3(POPL):3:1–3:28, 2019. URL `https://doi.org/10.1145/3290316`.

J.-Y. Girard. Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In J. E. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium (Univ. Oslo, 1970)*, volume 63 of *Studies in Logic and the Foundations of Mathematics*, pages 63–92. Elsevier, 1971. URL `https://doi.org/10.1016/S0049-237X(08)70843-7`.

R. Harper and J. C. Mitchell. Parametricity and variants of Girard's J operator. *Inf. Process. Lett.*, 70(1):1–5, 1999. URL `https://doi.org/10.1016/S0020-0190(99)00036-8`.

P. Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, 1984a.

P. Martin-Löf. Constructive mathematics and computer programming. volume 312, pages 501–518. The Royal Society, 1984b. URL `https://doi.org/10.1098/rsta.1984.0073`.

Microsoft Research. Lean theorem prover, 2019. URL `https://leanprover.github.io`. Version 3.4.2.

B. Werner. On the strength of proof-irrelevant type theories. *Logical Methods in Computer Science*, 4(3), 2008. URL `doi:10.2168/LMCS-4(3:13)2008`.