A Reasonably Gradual Type Theory

Kenji Maillard[1], Meven Lennon-Bertrand[1], Nicolas Tabareau[1], and Éric Tanter[2]

[1] Gallinette Project-Team, Inria, Nantes, France
[2] PLEIAD Lab, Computer Science Department (DCC), University of Chile, Santiago, Chile

Abstract

Gradualizing the Calculus of Inductive Constructions (CIC) involves dealing with subtle tensions between normalization, graduality, and conservativity with respect to CIC [5]. We present GRIP, a novel approach to gradual dependent types combining an impure sort of types inhabited by type casts, unknown terms and errors, as well as a pure sort of strict propositions internalizing precision. Internal precision supports reasoning about graduality within GRIP itself, for instance to characterize gradual exception-handling terms, and supports gradual subset types. The metatheory of GRIP is supported by a model formalized in Coq, and we provide a prototype implementation in Agda.[1]

Extending gradual typing [13, 14] to dependent types is a challenging endeavor due to the intricacies of type checking and conversion in presence of imprecision at both the type and term levels. While early efforts looked at gradualizing specific aspects of a dependent type system (*e.g.,* subset types and refinements [4, 15], or the fragment without inductive types [2]), we recently studied gradual typing in the context of the Calculus of Inductive Constructions (CIC) [5], the theory at the core of many proof assistants such as Coq [16] and Agda [8, 9]. There, we introduced GCIC, a gradual source language, whose semantics is given by elaboration to a dependently-typed calculus, called CastCIC.

A Dependently-Typed Cast Calculus CastCIC is an extension of Martin-Löf type theory (MLTT) [6] with (non-indexed) inductive types, and with exceptions as introduced by [10]. For a given type A, there are two exceptional terms, namely $\mathtt{err}_A$ representing runtime type errors, and $?_A$ representing the *unknown term*, which can optimistically stand for any term of type A. Additionally, CastCIC features a cast operator $\langle B \Leftarrow A\rangle\, t$, which supports treating a term t of type A as a term of type B, without requiring any relation between A and B. Intuitively, the cast operator is the identity when A and B are convertible, and fails when A and B are incompatible, for instance when A is the type of natural number and B is a function type. This intuition is made explicit by the notion of *(im)precision*: when a type A is more precise than B, written $A \sqsubseteq B$, then casting from A to B does not fail, and doing the roundtrip back to A is the identity; the formal formulation of this property, coined *graduality* by [7], is that when $A \sqsubseteq B$, the cast operations induce an embedding-projection pair between A and B. Additionally, $?$ is the least precise type, and therefore casting from A to the unknown type $?$ and back is always the identity. We previously uncovered in [5] an inherent tension which states that three fundamentally desirable properties cannot be fully satisfied simultaneously: (1) strong normalization, a property of particular relevance in the context of proof assistants, (2) conservativity with respect to CIC, namely the ability to faithfully embed the static theory in the gradual theory,[2] and (3) graduality, which guarantees that typing and evaluation are monotone with respect to precision. The maximality of the unknown type is a key element of this tension. Indeed, if $? \to ? \sqsubseteq ?$, then by graduality it is possible to embed the untyped lambda calculus, and in particular the diverging term $\Omega := (\lambda\, x : ?.\ x\ x)\ (\lambda\, x : ?.\ x\ x)$.

[1] https://gitlab.inria.fr/kmaillar/grip-a-reasonably-gradual-type-theory
[2] Not to be confused with logical conservativity!

Relaxing ?'s Maximality In this work, we observe that an adequate stratification of precision enables a full account of graduality for an extension of CastCIC, called GRIP. The key idea is that $?_{\square_i}$ should be the least precise type among all types at level i and below, *except* for dependent function types at level i (which are however still less precise than $?_{\square_{i+1}}$). We can precisely characterize problematic terms as those that are not *self-precise* (*i.e.,* more precise than themselves), which for function types means non-monotone with respect to precision. The prototypical example of a non-monotone term is that of recursive large elimination, such as the type of n-ary functions over natural numbers (in Coq):

```
Fixpoint nArrow (n : ℕ) : □₀ := match n with 0 ⇒ ℕ | S m ⇒ ℕ → narrow m.
```

The term `nArrow n` is a type (*i.e.,* a term of type $\square_0$), and we have for example `nArrow` $0 \equiv \mathbb{N}$ and `nArrow` $2 \equiv \mathbb{N} \to \mathbb{N} \to \mathbb{N}$. `nArrow` is not monotone because, given $n \sqsubseteq ?_{\mathbb{N}}$, there is no fixed level i for which `nArrow` $n \sqsubseteq ?_{\square_i}$ for any n. Another more practical example is that of a dependently-typed `printf` function, whose actual arity depends on the input string. We prove that the dynamic gradual guarantee holds in GRIP for any self-precise context, and that casts between types related by precision induce embedding-projection pairs between self-precise terms. Therefore, this shift in perspective in the interpretation of the unknown type and the associated notion of precision yields a gradual theory that conservatively extends CIC, is normalizing, and satisfies graduality for a large and well-defined class of terms.

Internalizing Precision, Reasonably While we could study graduality for GRIP externally, we observe that we can exploit the expressiveness of the type-theoretic setting to internalize precision and its associated reasoning. In particular this makes it possible to state and prove, within the theory itself, results about (self-)precision and graduality for specific terms. Internalizing precision requires solving an important obstacle: when adding exceptions to MLTT [10], the theory becomes inconsistent as a logic, because it is possible to inhabit any type A by raising an exception $\texttt{err}_A$. In the gradual setting, there is also the alternative of using the unknown term $?_A$ to inhabit any type A, so we need to avoid these degenerate proofs and provide a logically consistent theory. Moreover, useful reasoning on a notion of precision as error approximation [7] requires support from the gradual type theory in the shape of extensionality principles, an established challenge inside intensional type theories such as MLTT or CIC.

We address both issues by combining recent advances in type theory: the reasonably exceptional type theory RETT [11], that supports consistent reasoning about exceptional terms, and the observational type theory $\mathsf{TT}^{\mathsf{obs}}$ [12] that provides a setoidal equality in a specific universe $\mathbb{P}$ of definitionally proof-irrelevant propositions. A major insight of this work is to realize that we can actually merge the logical universe of RETT used to reason about exceptional terms with the universe $\mathbb{P}$ of proof-irrelevant propositions in order to define an internal notion of precision that is extensional and whose proofs cannot be trivialized with exceptional terms. We support this claim by formalizing a model in Coq using partial preorders (Footnote 1).

Applications of Internal Precision In addition to supporting reasoning about the graduality of terms in a theory that is not globally gradual, internal precision makes it possible to support gradual subset types, in which a type can be refined by a proposition expressed using precision. Moreover, in the literature, exception handling is never considered when proving graduality because this mechanism inherently allows terms that do not behave monotonically with respect to precision. Internal precision enables us to support exception handling in the impure layer of the type theory, and to consistently reason about the graduality (or not) of exception-handling terms. We illustrate these examples in a proof-of-concept implementation in Agda using rewrite rules [1] to evaluate the design of GRIP (Footnote 1).

References

[1] Jesper Cockx, Nicolas Tabareau, and Théo Winterhalter. The Taming of the Rew: A Type Theory with Computational Assumptions. *Proceedings of the ACM on Programming Languages*, 2020.

[2] Joseph Eremondi, Éric Tanter, and Ronald Garcia. Approximate normalization for gradual dependent types. In ICFP 2019 [3], pages 88:1–88:30.

[3] *Proceedings of the 24th ACM SIGPLAN Conference on Functional Programming (ICFP 2019)*, volume 3. ACM Press, August 2019.

[4] Nico Lehmann and Éric Tanter. Gradual refinement types. In *Proceedings of the 44th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2017)*, pages 775–788, Paris, France, January 2017. ACM Press.

[5] Meven Lennon-Bertrand, Kenji Maillard, Nicolas Tabareau, and Éric Tanter. Gradualizing the calculus of inductive constructions. *ACM Transactions on Programming Languages and Systems*, 44(2), June 2022.

[6] Per Martin-Löf. An intuitionistic theory of types, 1971. Unpublished manuscript.

[7] Max S. New and Amal Ahmed. Graduality from embedding-projection pairs. In *Proceedings of the 23rd ACM SIGPLAN Conference on Functional Programming (ICFP 2018)*, volume 2, pages 73:1–73:30. ACM Press, September 2018.

[8] Ulf Norell. *Towards a practical programming language based on dependent type theory*. PhD thesis, Department of Computer Science and Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Sweden, September 2007.

[9] Ulf Norell. Dependently typed programming in Agda. In *Advanced Functional Programming (AFP 2008)*, volume 5832 of *Lecture Notes in Computer Science*, pages 230–266. Springer-Verlag, 2009.

[10] Pierre-Marie Pédrot and Nicolas Tabareau. Failure is not an option - an exceptional type theory. In Amal Ahmed, editor, *Proceedings of the 27th European Symposium on Programming Languages and Systems (ESOP 2018)*, volume 10801 of *Lecture Notes in Computer Science*, pages 245–271, Thessaloniki, Greece, April 2018. Springer-Verlag.

[11] Pierre-Marie Pédrot, Nicolas Tabareau, Hans Fehrmann, and Éric Tanter. A reasonably exceptional type theory. In ICFP 2019 [3], pages 108:1–108:29.

[12] Loïc Pujet and Nicolas Tabareau. Observational Equality: Now For Good. *Proceedings of the ACM on Programming Languages*, 6(POPL), January 2022.

[13] Jeremy Siek and Walid Taha. Gradual typing for functional languages. In *Proceedings of the Scheme and Functional Programming Workshop*, pages 81–92, September 2006.

[14] Jeremy G. Siek, Michael M. Vitousek, Matteo Cimini, and John Tang Boyland. Refined criteria for gradual typing. In *1st Summit on Advances in Programming Languages (SNAPL 2015)*, volume 32 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 274–293, Asilomar, California, USA, May 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[15] Éric Tanter and Nicolas Tabareau. Gradual certified programming in Coq. In *Proceedings of the 11th ACM Dynamic Languages Symposium (DLS 2015)*, pages 26–40, Pittsburgh, PA, USA, October 2015. ACM Press.

[16] The Coq Development Team. *The Coq proof assistant reference manual.* 2020. Version 8.12.