A Guraded Syntactic Model of Gradual Dependent Types

Translation to Support Implementation and Metatheory

ANONYMOUS AUTHOR(S)

TODO write abstract

1 INTRODUCTION

Gradual dependent give principled static and dynamic semantics to programs where part of a type or term is missing.

Gradual dependent types have not yet been meaningfully implemented. Constructing a compiler for a dependently typed language is a massive engineering effort, and involves writing a type checker, a convertability check for terms, and unification engine for inference, in addition to the code generation and optimization. Writing a compiler for a gradual dependently typed language is

On the theoretical side, the story of gradual dependent types thus far has been one of compromise. The most comprehensive development of gradual dependent types is GCIC [Lennon-Bertrand et al. 2022], which extends the Calculus of Inductive Constructions (CIC) with gradual types. GCIC comes in three, each of which satisfies two of the following properties: (1) decidable type checking, (2) the gradual guarantees, and (3) conservatively extending CIC. JE ►*TODO: mention equality if either paper gets accepted*◄ The authors show that, to a degree, the compromise of GCIC is necessary: they establish a "fire-triangle" theorem that shows no dependently typed language can satisfy strong normalization, conservative extension of CIC, and the Embedding-Projection Pairs (EP-pairs) property, a strengthening of the gradual guarantees.

In this paper, we take the view that the fire triangle properties are means to an end, rather than ends unto themselves. Conservatively extending CIC is a useful trait on its own, since it ensures that no ill-typed static programs are allowed in a gradual language. Strong normalization, conversely, is mainly useful to prove that type-checking is decidable, as well as to establish logical consistency. Likewise, EP-pairs are mainly useful for showing the gradual guarantees, and showing that gradual programs to not unnecessarily produce ? as their result.

We present a gradual language GrInd, that has both an impelmentation strategy and a rich metatheory. GrInd circumvents the fire-triangle, sacrificing strong normalization, but keeping the properties that we actually want. Specifically, GrInd features decidable type checking, (weak) consistency and canonicity, the gradual guarantees, and conservative extension of (predicative) CIC. Moreover, we show that GrInd does not violate static reasoning principles: propositionally-equal CIC terms embedding in GrInd are observationally-equivalent, and while we lack full EP-pairs, we show that casts only change the behaviour of terms relative to dynamic type errors: applying casts may produce a term that raises an error in more or less situations, but when a concrete result is produced, it is the same result. JE ►*TODO explain EP pairs*◄

The key to proving GrInd's desirable properties is the same as the key making it straightforward to implement: we provide a translation from GrInd into static dependent types. For implementation, this translation means that existing dependently-typed normalization techniques, unification algorithms, and code generators can be used "off-the-shelf" to compile GrInd programs. For metatheory, the translation serves as a syntactic model in the style of Boulier et al. [2017], piggybacking off of the denotational semantics of a static type theory to give denotational semantics to GrInd. Moreover, by translating to a consistent type theory, we can prove theorems about GrInd in the type theory itself. Our translation and the theorems about it have been mechanized in Guarded Cubical Agda [?].

2 TWO PROBLEMS, ONE SOLUTION

2.1 Translation to Support Implementation

2.1.1 Don't Reinvent The Wheel.

2.1.2 An Implementation Strategy.

2.1.3 The Idris Model of Non-Termination.

2.1.4 Translating Approximate Normalization.

2.2 Metatheory

2.2.1 Extinguishing the Fire Triangle.

2.2.2 Static Reasoning in Gradual Code.

2.3 Modelling Gradual Dependent Types

2.3.1 Modelling Approximate Normalization.

2.3.2 Guarded Type Theory.

2.3.3 Relating Approximate and Exact Normalization.

REFERENCES

Simon Boulier, Pierre-Marie Pédrot, and Nicolas Tabareau. 2017. The next 700 Syntactical Models of Type Theory. In *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs* (Paris, France) *(CPP 2017)*. Association for Computing Machinery, New York, NY, USA, 182–194. https://doi.org/10.1145/3018610.3018620

Meven Lennon-Bertrand, Kenji Maillard, Nicolas Tabareau, and Éric Tanter. 2022. Gradualizing the Calculus of Inductive Constructions. *ACM Trans. Program. Lang. Syst.* 44, 2, Article 7 (apr 2022), 82 pages. https://doi.org/10.1145/3495528