Partially Static Types and Staged Partial Evaluation in Two-Level Type Theory

ANDRÁS KOVÁCS, Eötvös Loránd University, Hungary

Two-level type theory (2LTT) is a system that can be used for two-stage compilation, which allows dependent types both in metaprograms and in generated code output. Partially static data types are used in staged compilation in situations where only some parts of structures are known at compile time. We investigate partially static types and partial evaluation in the context of 2LTT. A key question is whether we can mechanically obtain partially static types from non-staged data types, and likewise lift non-staged operations to operations on partially static data. We show that all types and constructions in a Martin-Löf type theory (MLTT) can be given a partially static interpretation. Moreover, this interpretation yields a staged partial evaluator for MLTT, defined within 2LTT, which can be used to compile embedded MLTT syntax to normalized object code. We show that the interpretation can be used to partially mechanize the construction of staged normalizers for certain algebras. Finally, we compare and investigate free extensions of algebras ("frex") in 2LTT as an alternative generic approach to partially static types.

CCS Concepts: • **Theory of computation** → **Type theory**; • **Software and its engineering** → **Source code generation**.

Additional Key Words and Phrases: type theory, two-level type theory, staged compilation, partial evaluation

ACM Reference Format:
András Kovács. 2023. Partially Static Types and Staged Partial Evaluation in Two-Level Type Theory. 1, 1 (July 2023), 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In most practical programming languages it is possible to write code-generating code by simply manipulating strings or syntax trees, but this tends to be tedious, unsafe and non-composable. The purpose of *staged compilation* is to support metaprogramming with better ergonomics and more safety guarantees. In two-stage systems, user-written metaprograms are executed at compile time to produce code that is processed in further compilation.

Two-level type theory (2LTT) is a framework which supports two-stage compilation with strong safety and correctness properties. It is also compatible with a wide range of object-level and meta-level language features; in particular it allows dependent types at both levels. 2LTT was originally used in a purely mathematical context in synthetic homotopy theory [Annenkov et al. 2019], but it can be also applied in staged compilation [Kovács 2022].

A common application of staging is to use embedded languages without interpretative overhead, using *staged interpretation*, or going further, to optimize embedded programs or partially evaluate them at compile time. *Partially static data* is commonly used in staged interpreters and partial evaluators [**?**]. Such data contains a mixture of expressions of the object language and actual

Author's address: András Kovács, kovacsandras@inf.elte.hu, Eötvös Loránd University, Hungary, Budapest.

XXXX-XXXX/2023/7-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

structured data that is computed at compile time. A simple example is a compile-time list whose elements are object-level expressions.

In this paper, we investigate partially static types and partial evaluation in the context of 2LTT. In this context, we have a full-powered mathematical language at our disposal on the meta level. Our object language is a bit simpler but still highly expressive. Hence, we can internalize much of the formal reasoning about staging in 2LTT itself, and internalize constructions which were not feasible in systems with weaker type systems.

1.1 Overview & Contributions

- In **Section 2** we describe the specific variant of 2LTT used in this paper and give a short overview of its staging features.
- In **Section 3** we consider deep embeddings of a Martin-Löf type theory (MLTT) into 2LTT, and describe its staged interpretation and partial evaluation. The former maps embedded MLTT syntax to object-level code. The latter is similar, but it can also perform $\beta\eta$-normalization and could serve as a basis for more sophisticated compile-time optimization. We define a model of MLTT where each type is interpreted as an object-level type ("dynamic" type) together with a partially static type, and there is an embedding of the former into the latter. The interpretation of MLTT syntax in this model corresponds to staged partial evaluation.
  - From another point of view, this model explains how to extend all MLTT types with neutral values representing object-level expressions, obtaining partially static types.
  - Since MLTT terms are also modeled, any function in MLTT can be interpreted as a function operating on partially static types in 2LTT.
- In **Section 4** we describe an application of the partially static interpretation in normalization for algebraic structures. In some cases it is possible to replace an algebra with an equivalent algebra, but in which some or all equations hold definitionally. Then, we can take the partially static interpretation of such strict algebras. This yields a staged normalizer with respect to the definitional equations.
- In **Section 5** we look at free extensions of algebras, called *frex* in prior literature [Yallop et al. 2018]. A key advantage here is that algebras can be extended with decidable or even finite sets of variables, which makes it possible to compute more normal forms. For example, normalization for free groups require decidable equality of variables. Some subtleties arise here from the 2LTT staging setup and the fact that we internally verify algebraic laws. Free extensions always exist, but only those can be actually interpreted in object-level algebras which have a "fully normalized" presentation that does not use quotients.

2 OVERVIEW OF TWO-LEVEL TYPE THEORY

3 STAGED INTERPRETATION AND NORMALIZATION FOR MARTIN-LÖF TYPE THEORY

4 PARTIALLY STATIC TYPES IN NORMALIZATION OF ALGEBRAS

5 FREE EXTENSIONS OF ALGEBRAS IN TWO-LEVEL TYPE THEORY

6 RELATED WORK & CONCLUSIONS

REFERENCES

Danil Annenkov, Paolo Capriotti, Nicolai Kraus, and Christian Sattler. 2019. Two-Level Type Theory and Applications. *ArXiv e-prints* (may 2019). http://arxiv.org/abs/1705.03307

András Kovács. 2022. Staged compilation with two-level type theory. *Proc. ACM Program. Lang.* 6, ICFP (2022), 540–569. https://doi.org/10.1145/3547641

Jeremy Yallop, Tamara von Glehn, and Ohad Kammar. 2018. Partially-static data as free extension of algebras. *Proc. ACM Program. Lang.* 2, ICFP (2018), 100:1–100:30. https://doi.org/10.1145/3236795