Tunable Static Inference for Generic Universe Types

Werner Dietl\textsuperscript{1}, Michael D. Ernst\textsuperscript{1}, and Peter Müller\textsuperscript{2}

\textsuperscript{1} University of Washington
\texttt{\{wmdietl,mernst\}@cs.washington.edu},
\textsuperscript{2} ETH Zurich
\texttt{Peter.Mueller@inf.ethz.ch}

Abstract. Object ownership is useful for many applications, including program verification, thread synchronization, and memory management. However, the annotation overhead of ownership type systems hampers their widespread application. This paper addresses this issue by presenting a tunable static type inference for Generic Universe Types. In contrast to classical type systems, ownership types have no single most general typing. Our inference chooses among the legal typings via heuristics. Our inference is tunable: users can indicate a preference for certain typings by adjusting the heuristics or by supplying partial annotations for the program. We present how the constraints of Generic Universe Types can be encoded as a boolean satisfiability (SAT) problem and how a weighted Max-SAT solver finds a correct Universe typing that optimizes the weights. We implemented the static inference tool, applied our inference tool to four real-world applications, and inferred interesting ownership structures.

1 Introduction

Aliasing—multiple references to the same object—makes it hard to build complex object structures correctly and to guarantee invariants about their behavior. For example, mutation of an object through one reference can be observed through other references. This leads to problems in many areas of software engineering, including program verification, concurrent programming, and memory management.

Object ownership \cite{10} structures the heap hierarchically to control aliasing and access between objects. Ownership type systems express properties of the heap topology, for instance whether two instances of a list may share node objects. Such information is needed to show the correctness of a coarse-grained locking strategy, where the lock of the list protects the state of all its nodes \cite{6}. Ownership type systems also enforce encapsulation, for instance, by forcing all modifications of an object to be initiated by its owner. Such guarantees are useful to maintain invariants that relate the state of multiple objects \cite{33}. To obtain these benefits, ownership type systems require considerable annotation overhead, which is a significant burden for software engineers.
Helping software engineers to transition from un-annotated programs to code that uses an ownership type system is crucial to facilitate the adoption of ownership type systems. Standard techniques for static type inference [12] are not applicable. First, there is no need to check for the existence of a correct typing; a flat ownership structure gives a trivial typing. Second, there is no notion of a best or most general ownership typing. In realistic implementations, there are many possible typings and corresponding ownership structures, and the preferred one depends on the intent of the programmer. Ownership inference needs to support the developer in finding desirable structures by suggesting possible structures and allowing the programmer to guide the inference.

This paper presents static inference for Generic Universe Types [14,13], a lightweight ownership type system designed to enable program verification [32]. Our static inference builds a constraint system that is solved by a SAT solver. An important virtue of our approach is that the static inference is tunable; the SAT solver can be provided with weights that express the preference for certain solutions. These weights can be determined by general heuristics (for instance, to prefer deep ownership for fields and general typings for method signatures), by partial annotations, through a runtime analysis, or through interaction with the programmer.

The main contributions of this paper are:

**Static Inference:** an encoding of the Generic Universe Types rules into a constraint system that can be solved efficiently by a SAT solver to find possible annotations.

**Tunable Inference:** use of heuristics and programmer interaction to indicate which among many legal solutions is preferable; this approach is implemented by use of a weighted Max-SAT solver.

**Evaluation:** an implementation of our inference scheme on top of the OpenJDK compiler, and an illustration of its effectiveness on real programs.

This paper is organized as follows. Sec. 2 gives background on Generic Universe Types. Sec. 3 overviews the inference system using examples. Sec. 4 formalizes the static inference, consisting of the core programming language, the constraint generation rules, the weighting heuristics, and the encoding as a weighted SAT problem. Sec. 5 describes our implementation and our experience with it. Finally, Sec. 6 discusses related work, and Sec. 7 concludes.

## 2 Background on Generic Universe Types

Generic Universe Types (GUT) [14,13] is an ownership type system that allows programmers to describe and enforce hierarchical heap topologies and optionally enforces the owner-as-modifier encapsulation discipline based on the topology. GUT is integrated into the tool suite of the Java Modeling Language (JML) [23].