ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Adding Algebraic Data Types to a Verification Language

Practical Work - Description

Alessandro Maissen

Tuesday 26th October, 2021

Advisors: Prof. Dr. Peter Müller, Dr. Malte Schwerhoff
Department of Computer Science, ETH Zürich

Introduction

Viper [2, 3] is an intermediate language and toolchain for program verification, based on separation logic, an extension of Hoare logic that is well-suited for verifying concurrent programs with mutable state. Program verification often requires to prove properties over algebraic data types (ADTs). Unfortunately Viper currently does not directly support ADTs, but they can be encoded [1]. However this results in a lot of boilerplate encoding. Therefore it would be more convenient to have a direct support of ADTs in Viper, which this project aims to provide, via a suitable Viper plugin.

Algebraic Data Types

Algebraic data types (ADTs), popularised by functional programming languages, are a language feature commonly used to implement composite types, e.g. enums, and recursive data types, e.g. trees, as illustrated by the following pseudocode snippet of a binary tree with integer values in their leaves:

```
data Tree = Leaf(val: Int) | Node(left: Tree, right: Tree)
```

The arguably most well-known property of ADTs is enabling pattern matching, but they also have other, less obvious properties: e.g. given two leaves `n := Leaf(x)` and `m := Leaf(y)`, the equality `n == m` implies the equality `x == y`, and vice versa. Another less obvious property is that, if `t` is of type `Tree`, then `t` must either be a `Leaf` or a `Node`.

Approach and Focus

A Viper source program is parsed into an extended Viper AST, which is desugared into a Viper core AST before it is passed to a verifier. A Viper plugin [4] can inject additional source syntax and corresponding AST extension nodes, but if done, the plugin must also provide desugarings into Viper's core AST. This project can focus on the first steps — source syntax design, AST extention nodes, plugin architecture — because for the final desugaring step, we can most likely reuse code from the Gobra verifier [6], which already contains substantial code for encoding ADTs using core Viper features.

Goals

The main goal of this project is to support ADTs in Viper as a build-in type via a Viper plugin by:

- Designing a suitable source syntax, and implementing the necessary parsing, typechecking etc. steps

- Designing and implementing an internal representation of ADTs (extened Viper AST)
- Designing and implementing a desugaring into core Viper (by using code from Gobra, if possible)
- Designing and implementing a way of automatically deriving useful functions (e.g. contains, map, fold) from ADT definitions
- Evaluating the results via suitable tests

A possible project extension could be to implement a direct translation from extended Viper to SMT code, and to compare the performance of Viper desugaring vs. direct SMT encoding. This extension would only target Viper's symbolic-execution-based verifier Silicon [5].

Bibliography

[1] Programming Methodology Group. Encoding ADTs. http://viper.ethz.ch/examples/encoding-adts.html. Online; accessed 19 October 2021.

[2] Programming Methodology Group. Viper. http://viper.ethz.ch. Online; accessed 19 October 2021.

[3] P. Müller, M. Schwerhoff, and A. J. Summers. Viper: A verification infrastructure for permission-based reasoning. In B. Jobstmann and K. R. M. Leino, editors, *Verification, Model Checking, and Abstract Interpretation (VMCAI)*, volume 9583 of *LNCS*, pages 41–62. Springer-Verlag, 2016.

[4] Benjamin Schmid. *Abstract Read Permission Support for an Automatic Python Verifier*. Bachelor's thesis, 2017.

[5] M. Schwerhoff. *Advancing Automated, Permission-Based Program Verification Using Symbolic Execution*. PhD thesis, ETH Zurich, 2016.

[6] F. A. Wolf, L. Arquint, M. Clochard, W. Oortwijn, J. C. Pereira, and P. Müller. Gobra: Modular specification and verification of go programs. In Alexandra Silva and K. Rustan M. Leino, editors, *Computer Aided Verification (CAV)*, volume 12759 of *LNCS*, pages 367–379. Springer International Publishing, 2021.