Proposal for a master’s thesis

Compiling Foreign Functions in Isabelle

Supervisors: Dr. Andreas Lochbihler
Professor: Prof. David Basin
Issue date:

Prerequisites

- Knowledge in functional programming (e.g., from the FMFP course)
- Knowledge of logic and program verification (at least at the level of the FMFP course)
- Background in semantics of programming languages is a plus

Introduction

Functional programs can be written and proven correct in the proof assistant Isabelle/HOL [6]. Isabelle’s compiler translates these programs to the functional languages Haskell, Scala, Standard ML, and OCaml. The translation guarantees that compiled code behaves according to the theorems proven in Isabelle. Haftmann et al. [2, 3] make this guarantee more precise. They have introduced an intermediate language, from which the translation to the target languages is straightforward, and several transformations that convert Isabelle/HOL programs into the intermediate language. For these transformations, they prove that whenever the intermediate program evaluates to a result (i.e., it terminates without an error), then the result could also have been obtained in Isabelle by rewriting with the defining equations of the functions. Consequently, if we have proven some property $\forall x. P (f x)$ of a function $f$ and we compile and evaluate $f$ on some argument $t$ to a result $r$, then $P$ also holds for $r$.

Going one step further, we can prove a theorem in Isabelle by compiling the statement and checking that it evaluates to $True$.

Unfortunately, the correctness proof makes two assumptions that are not met in practice. First, the translation from the intermediate to the target language must generate code for all types and functions involved. However, for reasons of efficiency, readability and interfacing with library code, it maps some types (e.g., booleans, integers, characters, and lists) and some functions to pre-defined types and functions of the target language (foreign types and functions). Thus, the proof breaks as the evaluation steps of the intermediate language no longer mimick one to one the evaluation in the target language.
Second, the transformations assume that some invariants on the types hold. For example, all lists in Isabelle are finite, but lists in Haskell can also be infinite. In applications, the generated code is executed together with drivers written in the target language. Thus, the generated functions may be called with arguments for which the invariants do not hold. Continuing the above example, we cannot deduce any more that $P$ holds for $r$ if the argument $t$ is not generated code, because the invariants need not hold, i.e., the universal quantifier in the proven property does not range over $t$.

Objectives

The goal of this thesis is to extend the intermediate language with foreign types and functions and analyse the impact on the correctness guarantees. Different kinds of mappings should be distinguished:

1. The types are isomorphic (e.g., booleans).
2. The foreign type is smaller or larger than the Isabelle type (e.g., ASCII vs. Unicode characters, lists in Haskell).
3. The foreign type or function refines the one in Isabelle (e.g., unsigned words have unspecified length in Isabelle, but fixed sizes in the target languages).
4. The foreign function or type cannot exist in Isabelle/HOL (e.g., impure functions, interaction with the environment).

The different kinds will lead to different requirements on the mapping and to variations of the correctness statement. For the existing mappings in the Isabelle/HOL library and elsewhere [1, 4, 5], it should be checked which correctness guarantees the mappings preserve.

Tasks

This project can be subdivided into the following tasks:

1. Study the existing correctness statements and their proofs for code generation from Isabelle/HOL.
2. Extend the intermediate language with foreign types and pure foreign functions.
3. Analyse how foreign types and functions affect the correctness statements and proofs. Identify conditions on the mappings to foreign types and functions for the statements to hold.
4. Design methods to validate whether a mapping meets the conditions.
5. Check for the mappings in the Isabelle/HOL library (types unit, bool, option, list, integer, char) whether they meet the conditions. Otherwise, find examples where correctness is violated.
6. Do the same for machine integers [4].

7. (optional) Extend the previous parts to the stateful functions from [1].

8. (optional) Extend the previous parts to the interactive functions from [5].

9. Write the final report and prepare the presentation.

**Deliverables**

The following deliverables are due at the end of the project:

**Final report** The final report should consist of an introduction; a theoretical background section; one or more sections describing the modelling, implementation and evaluation; and a conclusion. The report may be written in English or German. Two copies of the report must be delivered to the supervisor.

**Isabelle/HOL theories** Complete Isabelle/HOL development that runs with the latest release or a recent developer’s version.

**Presentation** At the end of the project, a presentation of 30 minutes must be given during an InfSec group seminar. It should give an overview and discuss the most important highlights of the work.

**References**


