Static analyses for Vellvm
Laure Gonnord, Université Lyon1/CASH/ LIP
Yannick Zakowski Inria CASH/LIP
2020

Keywords
Abstract interpretation, compilation, abstract domain, certified compilation, LLVM, Vellvm

General Context
Optimizing compilers have been argued to be among the most complex humanly-built objects. In all account, they have been consistently observed to exhibit difficult to diagnose bugs [7], despite being thoroughly tested. CompCert [3] in contrast has paved the road to fully verified optimizing compilers : the compiler is coded and proved correct in a proof assistant, Coq for instance.

Vellvm [9] has been initiated in 2013 as a project similar in fashion to CompCert, but aiming to formalize parts of the LLVM [2] infrastructure. Central to this project is the definition of a formal semantics of (the sequential fragment of) LLVM IR. More recently, Zakowski et. al [8] have completely rebuilt the semantics of Vellvm based on modern semantics tools, the Interaction Trees [6] more specifically. Rather than relying on a traditional small-step operational semantics, a denotational semantics is built in a modular fashion.

The resulting semantics admits appealing compositional reasoning principles, as well as a very expressive relational program logics allowing for the construction of arbitrary refinements, allowing for the proof of semantic preservation of transformations.

While this new semantics has been so far exercised through the ongoing proof of a front-end for the Helix project and for elementary optimizations over IR, no static analyses has been developed yet to infer invariants of llvm code.

We thus propose to explore the possibilities opened by the new semantics in terms of abstract-interpretation based static analyses. Depending on the intern, the project may involve major Coq development or more traditional “on paper” proofs. However we aim to have a final working proof of concept.

Location and supervision
The internship will take place in LIP, Lyon. It will be co-advised by Yannick Zakowski and Laure Gonnord, in the CASH team.

The two supervisors have experience in developing SSA-based static analyses (Laure Gonnord) and developing formal proofs of software in Coq (Yannick Zakowski).

Internship content
Through this internship, we propose three possible exploratory projects :
— to develop unverified static analyses for LLVM programs aiming at scale and compositionality, in the spirit of sparse abstract interpretation for numerical or memory properties [5, 4]. In this idea, the connection to vellvm would be to use its formal semantics as an auxiliary tool to rationally design the analysis. Additionally, we encourage the analysis to be coded in OCaml to allow for plugin it with Vellvm, even if unverified.

1. https://github.com/vellvm/vellvm
2. https://github.com/vzaliva/helix/
to develop validated static analyses for Vellvm. The analysis would be coded in OCaml, and unverified itself. However, it would produce witnesses suitable to be checked. A checker, coded in Coq, takes such a witness as input and decides if the witness is valid or not. The checker is itself proved: if it validates the witness, the invariant is guaranteed to hold semantically. With this option, the first objective of the internship would be the design of the analysis, the notion of witness produced and the checker. The formal verification of the checker would be considered as a secondary objective.

— to develop verified static analyses for Vellvm. The analysis would therefore be itself coded in Coq, and proved correct in the style of Verasco [1].

The internship will consist of the following steps:
— Understand the software architecture and the main semantic ideas of Vellvm.
— Study sparse dataflow analysis and variants to understand and propose variants suited to the Vellvm semantics.
— Chose the direction according to your taste and the previous steps.

Références