Gloca-lang: reason globally, optimize locally!

Contacts:

- Thomas Colcombet (thomas.colcombet@irif.fr)
- Giovanni Bernardi (gio@irif.fr)
- Pierre-Evariste Dagand (dagand@irif.fr)

This project is specifically meant for Mathematicians with an inclination towards programming, or for Programmers with an eye for mathematics, or both.

Imagine writing a program in your favorite programming language (which is OCaml, undoubtedly). If, at some point, performance becomes critical, you are likely to be caught in the tension between writing high-level but somewhat inefficient code or low-level but somewhat tedious code.

This tension is traditionally arbitrated by programming language designers: they took charge of abstracting away what *they* deemed non critical (eg., handling memory management through garbage collection), which enables them to implement a sufficiently smart compiler[1] for the problem at hand. These compilers rely on an extensive set of static analyses[2] to infer the programmers' intents and produce as efficient code as possible.

We do not believe in sufficiently smart compilers. We believe in (moderately smart) programmers. We call for putting the smarts that are currently locked in the compiler backends back into the hands of programmers!

To toy with this idea, we are developing a novel programming language, called Gloca[3]. This research prototype aims at, on the one hand, studying the formal semantics necessary to model programming paradigm and, one the other hand, illustrating the possibilities this paradigm opens up.

Our prototype processes a tiny imperative language and produces a relational model. Further semantic inferences are described in a Datalog[4] model, processed by Souffle[5].

[1] https://wiki.c2.com/?SufficientlySmartCompiler
[2] https://clang.llvm.org/docs/DataFlowAnalysisIntro.html
[3] https://en.wikipedia.org/wiki/Glocalization
[4] https://en.wikipedia.org/wiki/Datalog
[5] https://souffle-lang.github.io/

The objective of this project is threefold:

- complete the Datalog model to handle static code specialization
- extend the model with several, folklore dataflow analysis
- illustrate the framework with original examples and contrast with the traditional approach

References:

- Gloca-lang[6]
- Souffle[7]
- Static program analysis[8]

[6] https://github.com/pedagand/gloca-lang
[7] https://souffle-lang.github.io/
[8] https://cs.au.dk/~amoeller/spa/