Analysis of the Go memory model

December 1st, 2015

candidate	Stian Valle
supervisors	Martin Steffen, Volker Stolz (UiO/HiB), Ka I Pun
group	PMA
type	60 ECTS
recommended background	program analysis, parsing, compilation
study program	computer science

Short description

The task is to formalize resp. analyze aspects of Go's memory model, in particular in connection with Go's communication and synchronizing constructs.

Background and motivation

The Go language is a relatively recent arrival to the programming language scene, and backed by Google for use in internal projects, upcoming Android support, and use by third parties like Docker.

Go is expressive, concise, clean, and efficient. Its concurrency mechanisms make it easy to write programs that get the most out of multicore and networked machines, while its novel type system enables flexible and modular program construction.

One important and tricky aspect of Go in connection with concurrecy is Go's *memory model*. As in basically all modern programming languages, Go supports a *weak* memory model. Such memory models, while crucial for efficient implementations and optimizing compilers reflecting advanced modern hardware, are notoriously hard to formalize, to analyze, and to program with.

Problem setting

The task of the master thesis is to provide means to *analyze* executions in Go's weak memory model. That task involves

- getting a firm and precise grip on the semantics of the memory model.
- find ways of analyzing, for instance statically, Go programs to detect situations where the program is not properly synchronized.
- implement a prototype analyzer based on the developed theory
- contribute to publications/conference submissions.

It may be possible to visit TU Darmstadt (Germany) during the thesis, within the *GoRETech*-project.

Keywords: program analysis, weak memory models, Go language, concurrency

References

[1] The Go programming language specification. `https://golang.org/ref/spec`, Aug. 2015.