Vrije Universiteit Brussel

Proposal bachelor thesis

Title: Performant Scheme Interpreter in asm.js

Promotor: Coen De Roover
Advisors: Quentin Stievenart, Jens Nicolay

Includes preparation course: Yes

Context

It is possible to implement a simple Scheme interpreter in JavaScript in a few hundreds of lines of code. The performance however would not be that great.

JavaScript offers a low-level subset of the language called asm.js, for which most JavaScript engines offer excellent performance optimizations. The downside is that asm.js does not really resemble a high-level programming language anymore, so it requires a lot more effort to implement even a simple interpreter in it.

Proposal bachelor thesis

Your goal is to implement a small garbage-collecting Scheme interpreter in asm.js.

You start from an unoptimized Scheme interpreter implemented in regular JavaScript.

You then refactor this version to use asm.js. The refactoring happens in steps. Ideally, every step preserves or increases performance.

The lower-level version will have to perform manual memory management, so you need to implement allocation and garbage collection).

In terms of performance, you will compare against your own unoptimized, regular JavaScript implementation, and against an existing Scheme interpreter (SLIP) implemented in C, which can be compiled to asm.js using emscripten.

Preparatory course bachelor thesis

You will use the preparatory phase to get familiar with asm.js.

In order to make informed design choices for your Scheme interpreter, you will read relevant portions of the book “Programming Languages: Application and Interpretation”.

The Scheme and C implementations of both Pico and SLIP also serve as an inspiration for your interpreter. You should get acquainted with these implementations to estimate the work that needs to be done, and to have a roadmap for your own experiments.

The preparatory report contains an overview of different design and implementation choices that you made based on reading the “Programming Languages” book and by studying the real-life implementations.