Lisp in Summer Projects Submission

Submission Date	2013-10-21 00:31:51
Full Name	Eric Bergstrome
Country	Canada
Project Name	Chisa
Type of software	command-line/terminal app
General category	lisp compiler/interpreter
LISP dialect	other
GitHub URL	https://github.com/ebb/chisa
Did you start this project?	Yes, all the code is written by me
Project Description	I want to describe my project in this form.
Purpose	Chisa is a small collection of software tools for experimenting with compilation. In particular, Chisa is designed to help programmers gain insight into the fundamentals underlying some of the compilation techniques used to generate efficient code for languages similar to ML.
Function	Chisa's primary function is to help the programmer run programs written in a certain intermediate language called FI. Intermediate languages often lack any defined concrete syntax and often appear embedded deep in the core of large compiler projects. In Chisa, the intermediate language has a concrete syntax and is the most visible component of a small project.
Motivation	Chisa's design is primarily motivated by the desire to gain hands-on experience with an intermediate language similar to that used by the MLton compiler. In addition, Chisa is designed to accomodate a bootstrapping process with minimal dependencies.

Audience

Chisa's intended audience consists of fellow compiler enthusiasts. In particular, Chisa is intended for compiler enthusiasts who have yet to write an optimizing compiler and for those who are fond of functional languages. Chisa's users will be those programmers who write Fl programs, examine the generated C programs, and consider how transformations might be written to improve Fl programs.

Methodology

Chisa has a runtime written in C. This runtime provides data abstractions for numbers, strings, and tagged-tuples. All of these data abstractions are immutable and have indefinite extent. The runtime does not provide a garbage collector; all programs that use the runtime are expected to run to completion before allocating too much memory.

Chisa defines two languages designed to use the runtime. The first, called Hl, is a higher-order, call-by-value unityped functional language with pattern matching. The second, called Fl, is like Hl but is first-order, has relaxed scoping rules, and organizes functions into basic blocks that use explicit local and non-local control transfer.

Both Hl and Fl use fully-parenthesized prefix notation for their concrete syntax. Chisa provides a yacc parser for each language. These parsers construct abstract syntax trees using the runtime's data abstractions.

Chisa includes a component that can generate C code from the abstract syntax tree of a Fl program. This component is fairly straightforward because it performs little more than a syntactic rewriting from the full Fl language into a subset of C. The generated C code calls into the runtime to execute primitive operations and to create and manipulate data.

All of the program components described above are written in C. Chisa also includes some preliminary Hl and Fl programs that do not work but do begin to reveal the character of these languages.

The Hl and Fl programs are based on the idea of a first bootstrapping compilation pass that transforms a subset of Hl programs into equivalent Fl programs. This compilation pass assumes a Hl input program that is first-order and observes certain limitations regarding the nesting of expressions. Its purpose is to make control transfers explicit and to organize functions into basic blocks.

The structure of the compilation pass is a recursive traversal of HI abstract syntax trees that allocates labels on the way down toward the leaves and collects basic blocks on the way back up the recursion toward the root. The pass is presented three times: (1) a well-commented reference implementation in the full HI language, (2) an implementation in the subset of HI that is accepted by the pass itself, and (3) in the FI language so that it can be executed.

Conclusion

Chisa accomplishes the goal of making an intermediate language that is simple, accessible, concrete, and similar to at least one real intermediate language used in an advanced optimizing compiler, namely MLton. In doing so, Chisa may help some of us compiler enthusiasts gain the intuition we need to write great compilers.

Some of Chisa's limitations are intentional. For example, the inclusion of a garbage collector would only be a distraction.

However, there are a number of limitations that are worth revisiting in a future project. The most obvious is that Chisa does not include a compiler for HI. Such a compiler would help put FI in context and would allow many more experimental programs to be written. It would also serve to demonstrate the intricacies of a bootstrapping process.

Once a HI compiler is available and it becomes easy to write experimental HI programs, the runtime should be expanded to include more IO capabilities and more data structures.

In addition to a HI compiler, it would be useful to have a pretty printer for both HI and FI. But most importantly, future projects should include transformations over FI programs; transformations like inlining, contification, dead code elimination, and constant propagation.

Build Instructions

Simply type 'make' within the project directory. An executable named 'fic' will be created. That executable is the program that generates C programs from FI programs.

Test Instructions

There are no automated tests but the following command will run 'fic' on some sample code:

	cat bootpass1.fi bootmain1.fi | ./fic
Execution Instructions	Write some FI code and save it in a file, say test.fi. Then run 'fic' like so: ./fic
Describe any bugs or caveats	The build process constructs a program named 'bootpass1'. That program does not work. It is included only because its construction demonstrates the workflow. (See the Makefile for details). The project has only been tested on Linux. It doesn't use any Linux-specific features but it's likely that some tweaks are needed to build it in other environments. The code is mostly C code but that's really just an implementation detail related to the fact that the project goals involve low-level code generation in a POSIX environment. The project is really all about Lisp-family languages and their implementation.
Official	I have read rules and have abided by them. I am 18 years of age or older. I am not living in Brazil, Quebec, Saudi Arabia, Cuba, Iran, Myanmar (Burma), North Korea, Sudan, or Syria.