Compilation Project

User Documentation

Master 1 INFO & MOSIG, UGA and Grenoble Inp
Team1
Ayan Hore, Lyubomyr Polyuha,Tuo Zhao, Kritika Mehta, Sitan Xiao,
17/01/2018

1 Introduction

The project implements a compiler for the language Mincaml and generates ARM code. The project is implemented in Java programming language.

2 Compilation and execution

2.1 Building

In order to build and run this project you need to have installed Java and Apache Ant on your machine.

2.2 Running

To run the program, execute run “make” from the main directory and after that ./scripts/mincaml followed by such command line options as (“-h”, “-v”). If the compiler is run with no command line arguments, the hit will be displayed.

For example: ./scripts/mincaml -v will print the current version number of our application.

If such error occurs: “bash: ./scripts/mincamlc: Permission denied” it is needed to change permission of mincamlc to “allow execution file as a program” or run this: “chmod -R 755 <path to folder TEAM1>”.

We faced the problem of executing the rest of command line options in terminal that require proved parser. The solution is to run compiler in any IDE that supports java. For that it is needed to edit “Run Configurations” and give as input a command line option.

Examples of pushing existing .ml file: “command line option” +

- /tests/inputs/if_else.ml
- /tests/inputs/netset_let.ml
- /tests/inputs/pr.ml
- /tests/inputs/test_add.ml
- /tests/inputs/test_print42_1.ml
- /tests/inputs/test_print42.ml

To see the results of compiler is possible just in console, not in the file.

2.3 Tests

To run the tests it is needed to run " make test". The test results will be shown.

3 Command-line options

Actions	Descriptions
-h	Displays usage instructions
-v	Displays the version
-asml <input file path>	Print the Asml
-t <input file path>	Type checking
-p <input file path>	Only parse the input (no further processing)
-o <output>	Optional. Specifies an output file;
-arm <input file path>	Generates ARM code and prints to console

4 Features

4.1 Supported features

Supported features of min-caml:

1. simple arithmetic expression
2. call to external functions
3. simple "first order" functions
4. if-then-else

4.2 Functionality implemented

1. K-normalization
2. Alpha Conversion
3. Reduction of nested let expressions
4. Closure Conversion
5. ASML Generation
6. Transformation to Data Structure
7. ARM generation

4.3 Limitation

- Typechecking is not implemented
- Writing into files using command line option -o is not correctly working
- Ability to see the result just in console

4.4 Future work

1. Type Checking
2. Beta Reduction
3. Constant Folding
4. Inlining
5. Closure Conversion
6. ARM Code generation for complex examples
7. New Testing scenarios