ABSTRACT

Functional Programming in Computer Science

We explore functional programming through a 16-week internship at Los Alamos National Laboratory. Functional programming is a branch of computer science that has exploded in popularity over the past decade due to its high-level syntax, ease of parallelization, and abundant applications. First, we summarize functional programming by listing the advantages of functional programming languages over the usual imperative languages, and we introduce the concept of parsing. Second, we discuss the importance of lambda calculus in the theory of functional programming. Lambda calculus was invented by Alonzo Church in the 1930s to formalize the concept of effective computability, and every functional language is essentially some implementation of lambda calculus. Finally, we display the lasting products of the internship: additions to a compiler and runtime system for the pure functional language STG, including both a set of tests that indicate the validity of updates to the compiler and a compiler pass that checks for illegal instances of duplicate names.

Functional Programming in Computer Science

Loren James Anderson
Los Alamos National Laboratory

December 18, 2015

About

- Home Institution: North Dakota State University

About

- Home Institution: North Dakota State University
- CCS-7

About

- Home Institution: North Dakota State University
- CCS-7
- Advisor: Dr. Kei Davis

About

- Home Institution: North Dakota State University
- CCS-7
- Advisor: Dr. Kei Davis
- Topic: Functional Programming in Computer Science

Functional Programming

Functional Programming

Advantages of Functional Programming

- parallelization
- function composition
- high-order functions

Functional Programming

Advantages of Functional Programming

- parallelization
- function composition
- high-order functions

Task

Read the textbook *Programming in Haskell* and complete all exercises [Hutton, 2007].

Parsing

Parsing

Example: $2 + 3 \cdot 4$

$$\begin{tikzcd}
& + \arrow[dl] \arrow[dr] & & \\
2 & & \cdot \arrow[dl] \arrow[dr] & \\
& 3 & & 4
\end{tikzcd}$$

Parsing

Example: $2 + 3 \cdot 4$

$$\begin{tikzcd}
 & + \arrow[dl] \arrow[dr] & & \\
2 & & \cdot \arrow[dl] \arrow[dr] & \\
 & 3 & & 4
\end{tikzcd}$$

Task

Read and code all segments in the paper *Higher-order functions for parsing* [Hutton, 1992].

Lambda Calculus

Lambda Calculus

Church’s Thesis

The effectively computable functions on the positive integers are precisely those functions definable in the pure lambda calculus (and equivalently computable by Turing machines).

Lambda Calculus

Church’s Thesis

The effectively computable functions on the positive integers are precisely those functions definable in the pure lambda calculus (and equivalently computable by Turing machines).

Task

Create a lambda calculus parser and evaluator and implement:

- basic mathematics functions
- list operations
- logical operators
- fixed point combinator (recursion)

Compiler

Compiler

Task

Create STG tests for the compiler using exercises from the textbook *Algorithms* [Sedgewick-Wayne, 2011].

Compiler

Task

Create STG tests for the compiler using exercises from the textbook *Algorithms* [Sedgewick-Wayne, 2011].

Task

Make a compiler pass checking for illegal instances of duplicate names within any file and between a file and the prelude. Construct error messages displaying the exact location of the duplication.

Compiler

Task

Create STG tests for the compiler using exercises from the textbook *Algorithms* [Sedgewick-Wayne, 2011].

Task

Make a compiler pass checking for illegal instances of duplicate names within any file and between a file and the prelude. Construct error messages displaying the exact location of the duplication.

Example Error Message

```
add = FUN(x x → plusInt x x);
```

```
variable x duplicated 2 times in location:  toplevel.add.
```

Acknowledgments

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internship (SULI) program.

- Dr. Kei Davis
- Dean Prichard
- Los Alamos National Laboratory

References

Hutton, Graham (1992)
Higher-order functions for parsing
J. Funct. Program. 2.3 (1992): 323-343.

Hutton, Graham (2007)
Programming in Haskell.
Cambridge University Press, 2007.

R. Sedgewick, K. Wayne (2011)
Algorithms
Addison-Wesley, 2011.