Programming Languages Literature Review

Emily Herbert

November 19, 2018

Contents

1 Introduction

2 Functional Programming

2.1 Data Structures . . . .

3 Object Orientated Programming

3.1 Java . . . .

4 Machine Learning

4.1 Type Safety . . . .

5 Simulation

5.1 Scala . . . .

Chapter 1

Introduction

Accumulation of summaries and notes from various PL papers. Entry format is as follows.

Title of Paper [citation].
Description of contents.

- Side note.
- Clarifying questions?

Chapter 2

Functional Programming

2.1 Data Structures

An Introduction to the Theory of Lists [1].

Formal definitions for lists and list operations. Discusses common list operations (map, filter, reduce, etc.) and offers best use cases. Provides examples of multiple interacting operations and operation equivalences.

- What are infinite lists?

Functional Pearl: The Zipper [3].

Describes a data structure that is akin to a zipper, for use in situations in which trees need to be modified non-destructively. Handles can be on particular elements of the tree, where locations hold the downward current subtree and the upward path. Functions are given for navigation left, right, up, and down, retrieving the nth element, changing an element in place, inserting elements left, right, and up, and deleting elements. A memoization approach is suggested, where ”scars” hold tree structure for frequently visited elements.

- Scala implementation at https://github.com/stanch/zipper.
- Binary trees are shown, but binary tree example doesn't allow for any data stored in the tree?

Chapter 3

Object Orientated Programming

3.1 Java

The Expression Problem [5].

Proposes a solution to the expression problem - ”goal is to define a datatype by cases, where one can add new cases to the datatype and new functions over the datatype, without recompiling existing code and while retaining static type safety.” Solution in GJ creates a series of types that are dependent on one another, and relies on Interfaces for expression evaluation. Type variables can be indexed by any inner class defined in the variable’s bound.

- I’m not sure I know what ”indexing a type variable” is ?

Chapter 4

Machine Learning

4.1 Type Safety

Typesafe Abstractions for Tensor Operations (Short Paper) [2].

Typesafe tensors for tensor operations and machine learning. Tensors created with two types, the type (primitives) of the elements, and the phantom types used to label dimensions. This allows operations to be checked at compile time to ensure that all tensor operations are valid. Typesafe tensors are differentiable and can be type-checked. Computation graphs are also typed, and inputs/ outputs are also type-checked. Examples given for tensor operations - matrix multiplication, tensor contraction. Examples given for NN layers - FC, conv, recurrent (recursive).

Chapter 5

Simulation

5.1 Scala

Using Domain Specific Languages for Modeling and Simulation [4].

Gives an overview of the Scalation simulation DSL, implemented in Scala. Discusses Scalation motivations and realizations, and how they make it a good fit for Scala. Mentions use of Actors for parallel simulation, but does not discuss further. System built on event graphs, "where the nodes represent types of events and the edges represent causal links between the events."

- https://github.com/scalation/scalation
- Scala Parser Combinator Library.
- What is the difference between a DSL and a library/ package?
- Hyrid Functional Petri Nets.
- Fortress.

Bibliography

[1] Richard S Bird. An introduction to the theory of lists. In *Logic of programming and calculi of discrete design*, pages 5–42. Springer, 1987.

[2] Tongfei Chen. Typesafe abstractions for tensor operations (short paper). In *Proceedings of the 8th ACM SIGPLAN International Symposium on Scala*, SCALA 2017, pages 45–50, 2017.

[3] Gérard Huet. The zipper. *Journal of functional programming*, 7(5):549–554, 1997.

[4] John A Miller, Jun Han, and Maria Hybinette. Using domain specific language for modeling and simulation: Scalation as a case study. In *Simulation Conference (WSC), Proceedings of the 2010 Winter*, pages 741–752. IEEE, 2010.

[5] Philip Wadler. The expression problem. *Java-genericity mailing list*, 1998.