Programming Languages and Systems Literature Review

Emily Herbert

September 4, 2019

Contents

Chapter 1

Introduction

Accumulation of summaries and notes from various PL papers. This is endeavor is mostly for my own personal use, but I figured eh, open science is a good thing. So, a lot of what's written here may just be half-formed ideas, or notes that really only make sense to me.

Entry format is (sometimes) as follows.

Title of Paper [citation].

Description of contents.

- Side note.
- Clarifying question?

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 [4].

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 [9].
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

Multi-agent System Simulation in Scala: An Evaluation of Actors for Parallel Simulation [8].
Proposes multi-agent simulation (MAS) in Scala that utilizes the Actor framework. Satisfies motivations for removing possibility of race conditions. Provides benchmarks comparing Actor framework to threads - Actor framework displays slower results, but the prospect of safer simulations justifies the slow-down.

- This design is outdated. What would a modern design look like?
- Would there be a motivation for simulation for RL agents?
- Scala's delimited continuations library.

Using Domain Specific Languages for Modeling and Simulation [6].
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.

Chapter 6

Software

6.1 Spreadsheets

Calculation View: multiple-representation editing in spreadsheets [7].
Presents "Calculation View" (CV), a novel alternate view in Microsoft Excel. Allows users to edit spreadsheets in a more high-level way. Introduces ranges, named ranges, pseudocells, and a block detection algorithm. Details further work to expand CV view, including text interface and expanded features.

- Can CV store intermediate values that the user might not need represented on the sheet?
- Can CV be used on its own? Is there a use case for CV being used on its own?

Elastic Sheet-Defined Functions: Generalising Spreadsheet Functions to Variable-Size Input Arrays [5].
Defines formal syntax and semantics for generalising spreadsheet functions to variable-size input arrays (ranges). Outlines algorithm for identifying generalizations and interpreting most likely generalizations given multiple options.

Chapter 7

Systems History

7.1 Template to Copy

Summary

Context

Discussion Points

Interest Factor

Significance

Personal Assessment

7.2 The Education of a Computer [3]

Summary This paper goes through iterations of "educating" a computer, to where the mathematician using a computer gets the boot, the computer becomes the mathematician, and the programmer becomes an integral part of the computer. A subroutine is a function-like that performs some computation. It has an entry line, exit line, result line, argument lines, and routine lines. Different procedures interact with different lines of the subroutines. The computer is given a bunch of smaller mathematical subroutines that the programmer can use to help construct their programs. It is hypothesised that more "subroutines" could be developed and combined. It is unclear to me what the correlation is between subroutines and modern day functions.

Context I think this work is difficult for modern day programmers to understand/ fathom (myself included). It is difficult to think of what may lie between assembly-style jumps and Haskell, for example, which is the space that this paper explores.

Discussion Points (1) Page 245 discusses turning using programs that contain subroutines as a subroutines itself, which I would consider a profound observation. Yet the "conclusion"

focuses just on the arithmetic type mathematical advanecs. Hmm. (2) The proposed UNIVAC is claimed to "not forget" and "not make mistakes." We all know that that is not the case.

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] Grace Murray Hopper. The education of a computer. In *Proceedings of the 1952 ACM national meeting (Pittsburgh)*, pages 243–249. ACM, 1952.

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

[5] Matt McCutchen, Judith Borghouts, Andy Gordon, Simon Peyton Jones, and Advait Sarkar. Elastic sheet-defined functions: Generalising spreadsheet functions to variable-size input arrays. November 2018.

[6] 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.

[7] Advait Sarkar, Andrew D Gordon, Simon Peyton Jones, and Neil Toronto. Calculation view: multiple-representation editing in spreadsheets. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 85–93. IEEE, 2018.

[8] Aaron B Todd, Amara K Keller, Mark C Lewis, and Martin G Kelly. Multi-agent system simulation in scala: An evaluation of actors for parallel simulation. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2011.

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