COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

COMS W4995 002: Parallel Functional Programming Fall 2021

Project Proposal

Professor Stephen A. Edwards

PowerList

Yash Datta

UNI yd2590

November 22, 2021

Contents

1 Introduction

J. Misra [1], has introduced **powerlist**, a new recursive data structure that permits concise and elegant description of many data parallel algorithms like prefix-sum, Batcher's sorting schemes, FFT etc. Powerlist is proposed as a recursive data structure that is more suitable for describing parallel algorithms. Similar to a list structure, the base case of powerlist is a list of one element. Longer power lists are constructed from the elements of 2 powerlists, p and q of the same length using 2 operators described below:

- $p \mid q$ is the powerlist formed by concatenating p and q.
- $p \bowtie q$ is the powerlist formed by successively taking alternate items from p and q, starting with p.

Further, both p and q are restricted to contain similar elements. In the following examples, elements of powerlist are enclosed within angular brackets.

- $\langle 0 \rangle \mid \langle 1 \rangle = \langle 0\ 1 \rangle$
- $\langle 0 \rangle \bowtie \langle 1 \rangle = \langle 0\ 1 \rangle$
- $\langle 0\ 1 \rangle \mid \langle 2\ 3 \rangle = \langle 0\ 1\ 2\ 3 \rangle$
- $\langle 0\ 1 \rangle \bowtie \langle 2\ 3 \rangle = \langle 0\ 2\ 1\ 3 \rangle$

There are many applications of this structure for parallel algorithms, some of which are:

- Prefix-sum (scan)
- Batcher sort
- Rank sort
- Fast Fourier Transform

In this project, I aim to develop a haskell module for powerlist data structure and use it to demonstrate how we can express common algorithms in the powerlist paradigm. I will also implement parallel versions of them and compare the performance with their sequential counterparts, demonstrating possible improvements. Due to time constraint, it might not be possible to implement all of them, so I will pick a few from the above list.

2 Proposal

I plan to execute the following:

- Develop a module for powerlist structure (and probably upload a package on hackage for others to use).
- Add algorithms for prefix-sum, batcher and rank sort using powerlist.
- Parallelize these algorithms using different strategies, some of which are described by Niculescu [2].
- Benchmark them against the sequential versions.
- Add more algorithms if time permits.

References

[1] J. Misra, "Powerlist: A structure for parallel recursion," *ACM Trans. Program. Lang. Syst.*, vol. 16, p. 1737–1767, nov 1994.

[2] V. Niculescu, "Data-distributions in powerlist theory," in *Theoretical Aspects of Computing – ICTAC 2007* (C. B. Jones, Z. Liu, and J. Woodcock, eds.), (Berlin, Heidelberg), pp. 396–409, Springer Berlin Heidelberg, 2007.