CMPSC250 Final Project Proposal

Aubrey Collins and Hawk Weisman

March 13, 2016

2-3 finger trees are a data structure used to implement other common data structures in a purely functional manner.[1] Finger trees are immutable and persistent, and can be used to implement structures such as sequences, indexed sequences, and other non-tree data structures, providing improved performance over other immutable data structures such as linked lists. In this proposal, we discuss our plan to develop a finger tree implementation as a library in the Scala programming language, and to demonstrate its performance and effectiveness.

[1] Ralf Hinze and Ross Paterson. Finger trees: A simple general-purpose data structure. *J. Funct. Program.*, 16(2): 197–217, March 2006. ISSN 0956-7968. DOI: 10.1017/S0956796805005769. URL `http://dx.doi.org/10.1017/S0956796805005769`

Motivation

IN FUNCTIONAL PROGRAMMING, the use of *pure functions* is either encouraged or enforced. Pure functions are those which do not cause modifications to external state;[2] rather, any state changes caused by a function must be encapsulated in its return value.[3] While it may at first seem counterintuitive, requiring programmmers to use only pure functions can have many benefits, such as making code more modular and easier to debug, eliminating common classes of concurrent-access errors, and enabling a wide variety of static analyses and compile-time optimizations.

[2] Often referred to as 'side effects'.

[3] Amr Sabry. What is a purely functional language? *Journal of Functional Programming*, 8:1–22, 1998. ISSN 1469-7653. URL `http://journals.cambridge.org/article_S0956796897002943`

In order to facilitate this style of programming, we should like to implement purely-functional data structures. Such data structures should be *immutable*, meaning that rather than being modified in-place, they are modified by creating a copy of the data structure with the changes applied.[4] As choosing to use purely-functional data structures rather than mutable ones can have performance disadvantages, we should take care to ensure that our implementations of these structures are are as efficient as possible.

[4] Chris Okasaki. *Purely functional data structures*. PhD thesis, Cambridge University Press, 1999

2-3 FINGER TREES, first proposed by Hinze and Paterson, provide a way to implement sequence data structures which provide amortized $\mathcal{O}(1)$ access and $\mathcal{O}(\log n)$ split and append operations. They may be easily adapted to implement indexed oI'r ordered sequences as well.[5] Sequence types are often implemented with linked list structures, for which accessing an arbitrary position in the sequence has a worst-case time complexity of $\mathcal{O}(n)$. Thus, we can see that the use of 2-3 finger trees is likely to offer significant performance advantages.

[5] Ralf Hinze and Ross Paterson. Finger trees: A simple general-purpose data structure. *J. Funct. Program.*, 16(2): 197–217, March 2006. ISSN 0956-7968. DOI: 10.1017/S0956796805005769. URL `http://dx.doi.org/10.1017/S0956796805005769`

In the Haskell programming language, the standard library's generic sequence type (`Data.Sequence`) is implemented using finger trees.[6] While the Scala programming language offers similar generic immutable sequence types, they are implemented using either linked-lists or arrays.[7]

[6] Ross Paterson, Louis Wasserman, Bertram Felgenhauer, David Feuer, and Milan Straka. *Data.Sequence*, 2014. URL `http://hackage.haskell.org/package/containers-0.5.7.1/docs/Data-Sequence.html`. Haskell Standard Library Documentation

[7]

Proposal

WE INTEND TO IMPLEMENT AND EVALUATE 2-3 finger trees in the Scala programming language.

References

Ralf Hinze and Ross Paterson. Finger trees: A simple general-purpose data structure. *J. Funct. Program.*, 16(2):197–217, March 2006. ISSN 0956-7968. DOI: 10.1017/S0956796805005769. URL `http://dx.doi.org/10.1017/S0956796805005769`.

Chris Okasaki. *Purely functional data structures*. PhD thesis, Cambridge University Press, 1999.

Ross Paterson, Louis Wasserman, Bertram Felgenhauer, David Feuer, and Milan Straka. *Data.Sequence*, 2014. URL `http://hackage.haskell.org/package/containers-0.5.7.1/docs/Data-Sequence.html`. Haskell Standard Library Documentation.

Amr Sabry. What is a purely functional language? *Journal of Functional Programming*, 8:1–22, 1998. ISSN 1469-7653. URL `http://journals.cambridge.org/article_S0956796897002943`.