Pretty Printing with Partial Continuations

Olaf Chitil

University of Kent, UK

Extended Abstract

A pretty printer converts tree structured data, for example the syntax tree of a program or an XML document, into nicely formatted text with a given line width limit. A pretty printing library provides the functionality common to a large class of pretty printers, thus enabling a programmer to easily implement a specific pretty printer. A pretty printing library enables the programmer to *compositionally* describe alternative layouts for components of the data to be printed. The pretty printing algorithm then chooses an *optimal* layout from this set of layouts. There is a trade-off between the available choice of alternative layouts, the optimality criterion and the efficiency of the pretty printing algorithm.

In 1980 Oppen [4] published an imperative pretty printer that allows the description of nice layouts, adequate for many purposes, and that is very efficient. The algorithm takes time linear in the size of the input, independent of the line-width limit. Furthermore, the algorithm is *bounded*, that is, it produces parts of the output already after having processed only limited parts of its input. Oppen's work inspired numerous pretty printing libraries, in particular several Haskell libraries [3, 5, 7, 1, 2, 6], because lazy evaluation seems to be a natural basis for implementing boundedness.

In [1, 2] I presented the first purely functional algorithm that has all the nice efficiency properties of Oppen's algorithm. This algorithm uses an intricate lazy coupling of two double-ended queues. Thus it demonstrates the power of laziness but is rather complex and requires a specially modified implementation of double-ended queues.

Subsequently Doaitse Swierstra [6] showed that the two special double-ended queues can be replaced by one double-ended queue and a lazy list, thus giving a simpler solution that also reuses a standard double-ended queue implementation.

Now I present a new purely functional algorithm that is both linear-time and bounded and is faster than any previous functional implementation. It is relatively simple and explicitly expresses all issues in pretty printing, many of which are hidden in previous implementations based on implicit lazy evaluation. Correctness and linear runtime of the new algorithm do not rely on lazy evaluation. However, lazy evaluation ensures that input and output are only evaluated as demanded and thus the algorithm only requires a small bounded amount of space.

In the new algorithm partial continuations express explicitly the necessary switching between producing output and processing the input. While traversing

the document that is to be formatted we update a standard double-ended queue of partial continuations. Each partial continuation can output a part of the document. The double ended queue serves as a buffer for those parts of the document that have been traversed but for which the correct formatting cannot yet be decided. We need a double ended queue to both process new input and to check continously if we can already produce more output. Finally, specialisation reduces the number of costly operations on the double-ended queue and thus improves performance.

Acknowledgements

Thanks to Bernd Braßel and Michael Hanus for discussions about how logical variables could simplify the implementation of pretty printing. The need for deferring output until logical variables are bound gave me the idea of using continuations.

References

1. Olaf Chitil. Pretty printing with lazy dequeues. In *Preliminary Proceedings of the 2001 ACM SIGPLAN Haskell Workshop*, pages 183–201. Universiteit Utrecht, 2001. UU-CS-2001-23.
2. Olaf Chitil. Pretty printing with lazy dequeues. *Transactions on Programming Languages and Systems (TOPLAS)*, 27(1):163–184, January 2005.
3. John Hughes. The design of a pretty-printing library. In J. Jeuring and E. Meijer, editors, *Advanced Functional Programming*, LNCS 925. Springer Verlag, 1995.
4. Dereck C Oppen. Prettyprinting. *ACM Transactions on Programming Languages and Systems*, 2(4):465–483, 1980.
5. Simon L Peyton Jones. A pretty printer library in Haskell. Part of the GHC distribution at http://www.haskell.org/ghc, 1997.
6. S. Doaitse Swierstra. Linear, online, functional pretty printing (corrected and extended version). Technical Report UU-CS-2004-025a, Utrecht University, 2004.
7. Philip Wadler. A prettier printer. In *The Fun of Programming*, chapter 11, pages 223–244. Palgrave Macmillan, 2003.