Reflections on the Burrows Wheeler transform

Peter Fenwick

July 28, 2004

Technical Report 172
Department of Computer Science, The University of Auckland
Auckland, New Zealand

Abstract

This report presents some speculations on the nature of the Burrows Wheeler transform, from analogies with the better-known techniques of signal processing. These analogies, plus considerations of symmetry and inverse operations, suggest some possible developments to the Burrows Wheeler transform for lossless compression.

1 Introduction

Much of physics depends on viewing problems from two or more aspects, each one giving a particular insight into an otherwise difficult situation. One example is the particle/wave duality of quantum theory. Similarly, physics and mathematics often *transform* problems from one "space" into another space, again to facilitate computation or provide a complementary view. Simple examples here include conversion between rectangular and polar coordinate systems, or trigonometric functions and their inverses.

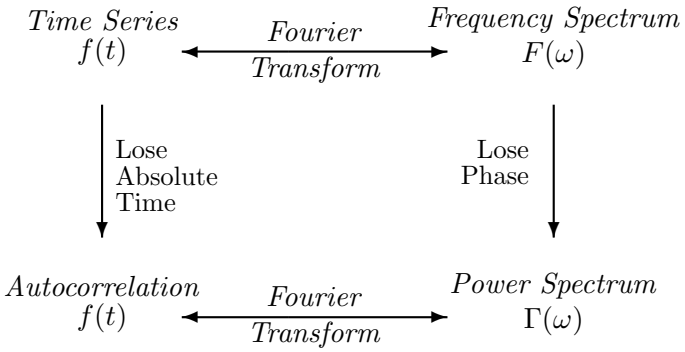

Figure 1: Time Series Transformations

A more complex example is the Fourier Transform, which converts between time space and frequency space. It was the structure and symmetry of the Fourier transform in time-series processing

that led to the proposed Burrows Wheeler structure. Figure 1 illustrates the complex of transforms between the time (t) and frequency (ω) domains or spaces. For now the main point, which inspired the Burrows-Wheeler reinterpretation, is the symmetry of the transformations.

The Fourier Transform is completely reversible, both at the top between time series and spectrum and at the bottom between autocorrelation and power spectrum. Vertically however the downward operations lose information and cannot be reversed. In contrast the Burrows-Wheeler transform and related operations *are* reversible vertically, but there is no "bottom connection".

2 The Burrows Wheeler transform

The Burrows Wheeler transform[1, 2] and its inverse convert the input data between "source order" and "context order", or between *source space* and *context space*. A major problem with improving Burrows Wheeler compression is that conventional compressors are designed to operate in source space and are extremely inefficient in context space. The normal structure on which they depend is largely destroyed by the Burrows Wheeler transformation. An alternative to MTF recoding that overcomes these problems for some files is the "inversion coding" of Arnavut[3].

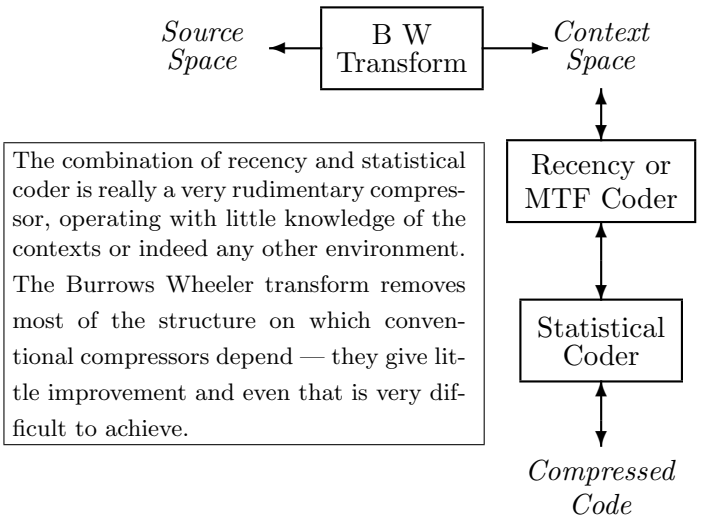

Figure 2: Basic Burrows Wheeler compressor.

Figures 2–4 illustrate the development of the new ideas. Figures 2 and 3 show the basic structure of first Burrows Wheeler compression and then PPM compression. Each figure includes comments to emphasise its important aspects.

The Burrows Wheeler compressor is seen to be rather naïve, especially in comparison with its PPM counterpart of Figure 3. PPM goes to a great deal of trouble to derive contexts from the source; it is these contexts that control the whole compression operation and give PPM is good performance.

The crucial point is that these PPM contexts, inferred from *source space*, are an approximation to the *context space* on the "other side". By symmetry, we propose the new Burrows Wheeler model of Figure 4 which, working in *context space*, builds approximations to the *source text* and uses these to control the detailed code generation. The source estimation is shown at the right-hand side of Figure 4, being symmetrical with the context estimation of PPM.

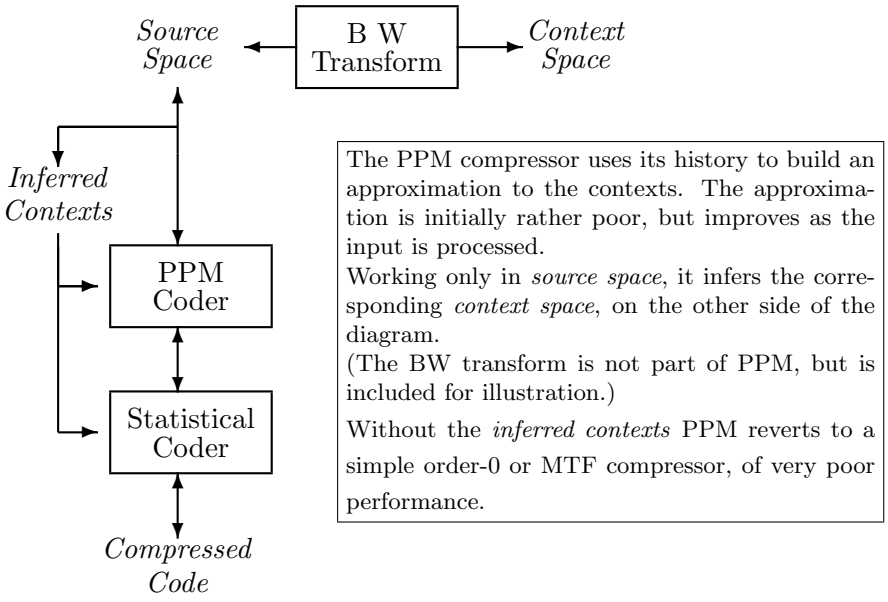

Figure 3: PPM compressor.

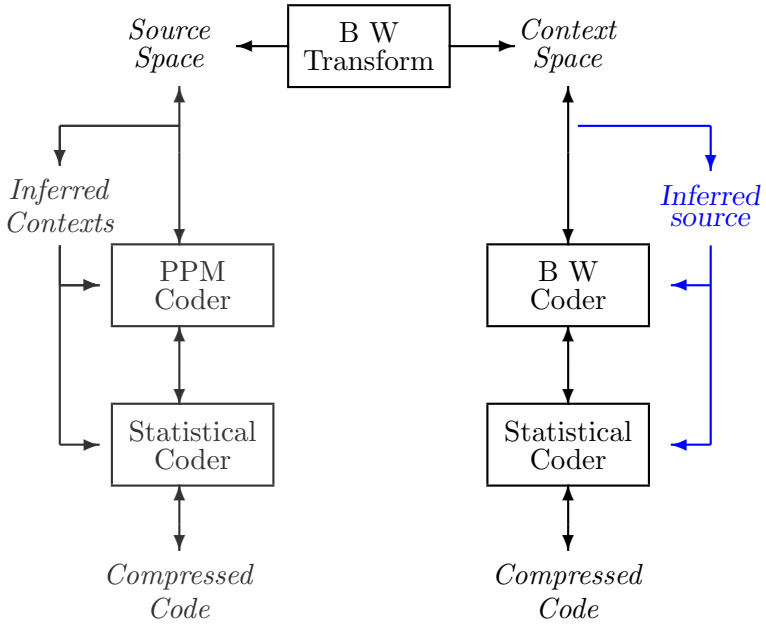

Figure 4: Enhanced Burrows Wheeler compressor.

3 Possible implementation

This section describes one possible implementation of the proposed compressor. In contrast to most previous Burrows Wheeler work, it concentrates on the *reverse* transform, breaking it open and incorporating it into the compressor. The overhead is that we must encode the counts of all symbols, sending information that is usually extracted as an early stage of the reverse transform. As most text files have about 80 symbols, each requiring 10–12 bits to encode, the overhead is probably around 1000 bits (say 2500 for binary files).

1. Encode into the compressed file the frequencies of all symbols in the source alphabet. This duplicates (or anticipates) the *first* step of the reverse Burrows Wheeler transform, the step that usually follows the recovery of the permuted symbols.

2. As the transformed text is processed, build links for each symbol to its context; this anticipates the *second* stage of the reverse transform.

3. Eventually the separate links will start to combine, building longer source fragments that can be used as contexts to enable better prediction of symbols.

(Because we know the frequency of each symbol, we can also build a simple model for each order 1 context and know when to clear it at the start of the next context.)

The growing knowledge of the source can be used to guide selection of the likely symbols, increasing the efficiency of the recency or MTF process (or replacing it completely). We can also envisage an exact parallel to PPM, where the inferred source can predict probabilities of the known symbols and control the statistical coder. It also immediately raises those perennial problems of PPM, concerning exclusion, zero frequencies and escape probabilities, problems that are usually avoided with Burrows Wheeler compression. But experience with PPM may be of great help here.

4 Conclusions

This report has attempted to develop some understandings of Burrows Wheeler compression by analogy with time-series transforms and suggests directions for future work. The results may be relevant and useful. (But then again they might not be ....) In any case I hope that it may encourage further work in unconventional directions. I have long felt that the real underlying structure of the Burrows Wheeler transform was not understood and hope that this report may help remedy that deficiency.

References

[1] Burrows M., Wheeler, D.J. (1994) "A Block-sorting Lossless Data Compression Algorithm", *SRC Research Report* 124, Digital Systems Research Center, Palo Alto.
`gatekeeper.dec.com/pub/DEC/SRC/research-reports/SRC-124.ps.Z`

[2] Fenwick, P.M., "The Burrows-Wheeler Transform for Block Sorting Text Compression – Principles and Improvements", *The Computer Journal*, Vol 39, No 9, pp 731–740, 1996.

[3] Arnavut, Z., "Move-to-front and Inversion coding", *Proc. IEEE Data Compression Conference*, March 2000, IEEE Computer Society Press, Los Alamitos, California pp193–202