Title: Rewriting Haskell Strings
Author: Don Stewart, University of New South Wales

The Haskell String type is notoriously inefficient. In this talk I will introduce a new data type, ByteString, based on lazy lists of byte arrays, combining the speed benefits of strict arrays with lazy evaluation, along with equational transformations based on term rewriting are used to deforest intermediate ByteStrings automatically.

Novel fusion combinators with improved expressiveness and performance over previous functional array fusion strategies are described, and provide the foundation for the Data.ByteString library, a Haskell library providing a purely functional interface to unboxed byte arrays, approaching the speed of low-level mutable arrays in C.

The main contribution is to introduce a new system for fusion, based on ``streams'', offering greater expressiveness and generality than has been possible with previous work on functional array fusion. Secondly, we describe a full scale, successful implementation of stream fusion for byte arrays. The implementation utilises existential types, the Haskell foreign function interface and compiler rewrite rules, while presenting the user with a familiar, purely functional interface. The fusion techniques presented are not restricted to arrays or to Haskell, and should be generally applicable to sequence-like data structures, including lists.

The use of fusible array combinators dramatically improves both the time and space performance of I/O and string-based Haskell programs. Indeed, we are finally able to realise the performance promise of declarative programming in Haskell.