Effects of Finite-Precision Arithmetic in Enumerative Coding

Kees A. Schouhamer Immink and Augustus Janssen

Abstract — The storage requirements of conventional enumerative schemes can be reduced by using floating point arithmetical operations instead of the conventional fixed point operations. The new enumeration scheme incurs a small coding loss. A simple relationship between storage requirements and coding loss is derived.

I. INTRODUCTION

Translation using enumeration has the virtue that the complexity (weight storage) grows polynomially with the code-word length contrasting the complexity of direct look-up which grows exponentially. Here we use a floating point instead of a fixed point representation of the weights. The floating-point notation is convenient for representing numbers that differ many orders of magnitude. In this notation, each weight is represented by s bits, and as a result, the hardware required for storage grows linearly with the codeword length n. The penalty attached to the finite-precision representation of the weights is that it will entail a (small) loss in code rate. A quantitative trade-off between the precision of the number representation and concomitant code redundancy will be detailed. We will apply the theory to the enumeration of \((dkr)\) sequences.

II. ENUMERATION USING FLOATING-POINT ARITHMETIC

The hardware for implementing the enumeration algorithm comprises a (binary) adder, a subtracter, a comparator, and a look-up table of the pre-computed set of weights. The binary fixed-point representation of a single weight requires \(Rn\) bits per weight, where \(R, 0 < R < 1\), is a constant. For the overall scheme, we need therefore storage proportional to \(Rn^2\). We develop an enumeration method where the weights are specified in floating-point notation.

We employ a two-part radix-2 representation

\[ I = (m, e) \]

to express the weight

\[ I = m \times 2^e, \]

where \(I, m, \) and \(e\) are non-negative integers. The two components \(m\) and \(e\) are usually called mantissa and exponent of the integer \(I\), respectively. The translation of a weight into \((m, e)\) is easily accomplished. It is assumed that the exponent of each weight is represented by \(e_p\) bits.

Let \(I\) be the short-hand notation of one of the weights \(N^0(i)\). It is well known that the non-negative integer \(I, I < 2^n\), can be uniquely represented by a binary \(n\)-tuple \(x\) = \((x_{n-1}, \ldots, x_0)\), where

\[ I = \sum_{i=0}^{n-1} x_i 2^i. \]