Error Detection and Correction: Hamming Code; Reed-Muller Code

Greg Plaxton
Theory in Programming Practice, Spring 2005
Department of Computer Science
University of Texas at Austin

Hamming Code: Motivation

- Assume a word size of k
- Recall parity check coding
  - Send one additional bit per word, the parity bit
  - Allows detection (but not correction) of a single error (bit flip) in the $k+1$ bits transmitted
- Hamming code
  - Send ℓ additional bits per word, called the check bits
  - Allows correction of a single error in the $k+\ell$ bits transmitted

Hamming Code: Determining The Number of Check Bits

- We choose ℓ as the least positive integer such that the binary representation of $k+\ell$ has ℓ bits
  - Exercise: Prove that such an ℓ is guaranteed to exist
  - Examples: If $k = 1$, we set ℓ to 2 since $k+\ell = 3 = 11_2$; if $k = 2$, we set ℓ to 3 since $k+\ell = 5 = 101_2$; if $k = 4$, we set ℓ to 3 since $k+\ell = 7 = 111_2$
- What is the maximum number of data bits k corresponding to a given number of check bits ℓ?
  - The positive numbers with ℓ-bit binary representations range from $2^{\ell-1}$ to $2^{\ell}-1$
  - So we need $k+\ell \leq 2^{\ell}-1$, i.e., $k \leq 2^{\ell}-\ell-1$

Hamming Code: The Construction

- Index the $k + \ell$ bit positions from 1 to $k + \ell$
- Put the ℓ check bits in positions with indices that are powers of 2, i.e., $2^0 = 1 = 1_2$, $2^1 = 2 = 10_2$, $2^2 = 4 = 100_2$, $2^3 = 8 = 1000_2$, . . .
- Put the k data bits in the remaining positions (preserving their order, say)
- Choose values for the check bits so that the XOR of the indices of all 1 bits is zero
  - Can we always find such a setting of the check bits?
  - Is this setting unique?

Hamming Code: Decoding

- We'd like to argue that if 0 or 1 bit flips occur in transmission of the encoded bit string of length $k + \ell$, then the decoder can uniquely determine the original k data bits
- The decoder first computes the XOR of the indices of all 1 bits in the (possibly corrupted) string of length $k + \ell$ that it receives
  - If no errors occurred in transmission, the XOR is zero
  - If a 0 flipped to a 1 in bit position i, the XOR is i
  - If a 1 flipped to a 0 in bit position i, the XOR is i
- So what rule should the decoder use to determine the original k data bits?

Reed-Muller Code: Motivation

- So far we've seen efficient codes for detecting a single error (parity check code) and for correcting a single error (Hamming code)
- What if we want to be able to detect or correct a large number of errors?
  - We need to find a set of codewords such that the minimum Hamming distance between any two codewords is large
- For any nonnegative integer n, the Reed-Muller code defines 2^n codewords of length 2^n such that the Hamming distance between any two codewords is exactly 2^{n-1}
  - How many errors can be detected (as a function of n)?
  - How many errors can be corrected (as a function of n)?

Reed-Muller Code: Hadamard Matrices

- The Reed-Muller code is based on Hadamard matrices
- We now inductively define a $2^n \times 2^n$ Hadamard matrix H_n for each nonnegative integer n
  - $H_0 = [1]$
  - H_{n+1} is formed by putting a copy of H_n into each quadrant, and complementing the copy placed in the lower-right quadrant
- For any nonnegative integer n, the 2^n codewords of length 2^n of the corresponding Reed-Muller code are simply the rows of H_n
  - It remains to argue that the Hamming distance between any two codewords is exactly 2^{n-1}

Reed-Muller Code: Proof of the Hamming Distance Property

- We prove the claim by induction on $n \geq 0$
- Base case: H_0 has only one row, so any claim regarding all pairs of rows holds vacuously
- Induction hypothesis: Assume that for some nonnegative integer n, the Hamming distance between any two rows of H_n is 2^{n-1}
- Induction step
  - Consider rows i and j (numbering from 1, say) of H_{n+1}, where $i < j$
  - Verify that the Hamming distance between rows i and j is 2^n in each of the following cases: (1) $j \leq 2^n$; (2) $i > 2^n$; (3) $i \leq 2^n$ and $j = 2^n + i$; (4) $i \leq 2^n$ and $j \geq 2^n$ and $j \neq 2^n + i$