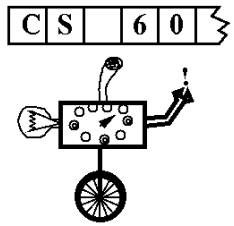


More Binary Encodings

November 7, 2001

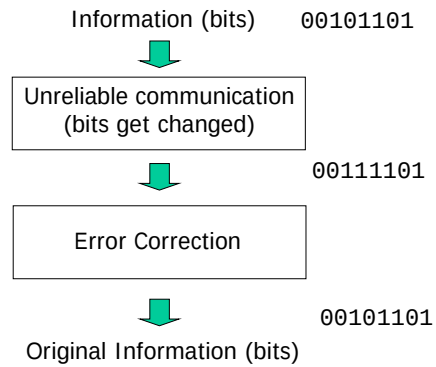
More Encoding Examples

- Encode the set {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
  - Encoding #1 (BCD: binary-coded decimal)
    - `0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001`
  - Encoding #2 (7-segment encoding)
    - `1110111, 0010010, 1011101, 1011011, 0111010, 1101011, 1101111, 1010010, 1111111, 1111010`

Encoding Examples

- Encoding #2 (7-segment encoding)
  - 1110111, 0010010, 1011101, 1011011, 0111010, 1101011, 1101111, 1010010, 1111111, 1111010

Error-Correcting Codes



Error-Correcting Codes

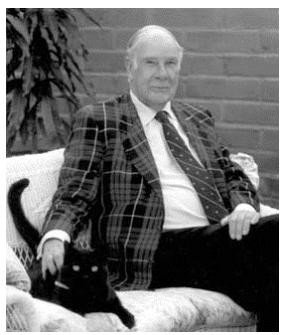
- Uses:
  - Communication lines
  - Computer memory
- The trick:
  - Transmit extra bits
  - But how?
  - What if *those* bits are corrupted too?

Hamming Codes

- The objective:
  - Tolerate bit *flips* due to **errors** in communication, memory loss, etc.
- The approach:
  - Add **redundant** bits
  - If one bit is flipped, the original can still be recovered based on the resulting configuration
  - The clever part is that the "redundant" bits can also be flipped without harm.

Richard W. Hamming, 1915-1998

worked with Shannon at Bell Labs

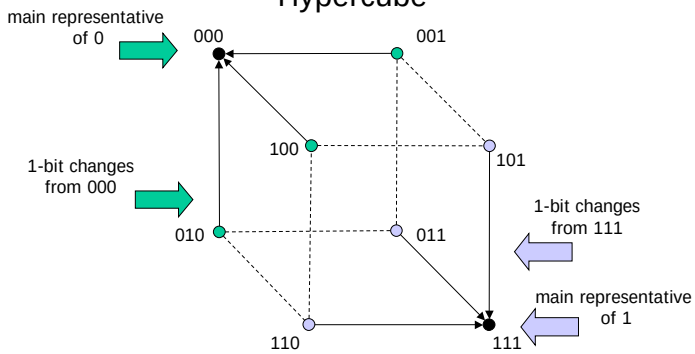


"The purpose of computing is insight, not numbers."

Hamming Code Example

- Encode set {0, 1}

Hamming Code Visualization: Hypercube



Efficiency

- The Hamming code may look inefficient (3 bits to encode 1 bit).
- However, it gets better for more bits.
- Among n bits total, only about $\log_2(n)$ are required to do support the error-correcting part.

Analysis

- Let b be the total number of bits in the code.
- Let d be the number of data bits (2^d data words).
- For each data word, we need $1+b$ combinations for error correction (any of b bits could be flipped).
- Therefore $2^d(1+b)$ combinations in all are required, so $2^b \geq 2^d(1+b)$.

```
    size(d) = find(1, d);
    find(b, d) = pow(2, b) >= pow(2, d)*(1+b) ?
                     b : find(b+1, d);
map(size, range(1, 20)) ==>
    [3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17,
     18, 19, 20, 21, 22, 23, 24, 25]
```

- e.g. 1 million code words can be handled with 25 bits.

7-bit Hamming Code
(4 bits of data, 3 bits redundancy)

		data	data	data	parity	data	parity	parity
decimal	binary	b_7	b_6	b_5	b_4	b_3	b_2	b_1
0	0000	**0**	**0**	**0**	0	**0**	0	0
1	0001	**0**	**0**	**0**	0	**1**	1	1
2	0010	**0**	**0**	**1**	1	**0**	0	1
3	0011	**0**	**0**	**1**	1	**1**	1	0
4	0100	**0**	**1**	**0**	1	**0**	1	0
5	0101	**0**	**1**	**0**	1	**1**	0	1
6	0110	**0**	**1**	**1**	0	**0**	1	1
7	0111	**0**	**1**	**1**	0	**1**	0	0
8	1000	**1**	**0**	**0**	1	**0**	1	1
9	1001	**1**	**0**	**0**	1	**1**	0	0
10	1010	**1**	**0**	**1**	0	**0**	1	0
11	1011	**1**	**0**	**1**	0	**1**	0	1
12	1100	**1**	**1**	**0**	0	**0**	0	1
13	1101	**1**	**1**	**0**	0	**1**	1	0
14	1110	**1**	**1**	**1**	1	**0**	0	0
15	1111	**1**	**1**	**1**	1	**1**	1	1

These are the representatives.

7 bits = 2^7 combinations total, 2^4 of which are representatives.