Arithmetic Coding

Huffman coding and the like use an integer number (k) of bits for each symbol, hence k is never less than 1. Sometimes, e.g., when sending a 1-bit image, compression becomes impossible.

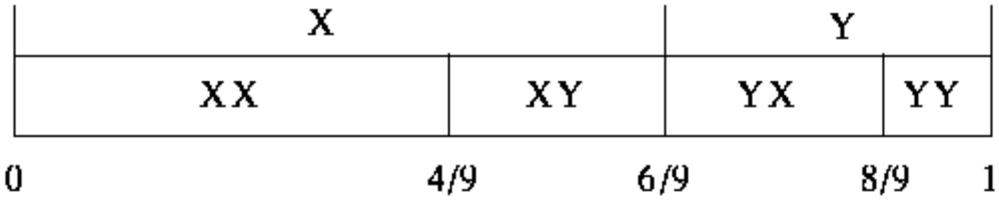
- Idea: Suppose alphabet was

$$X, Y$$

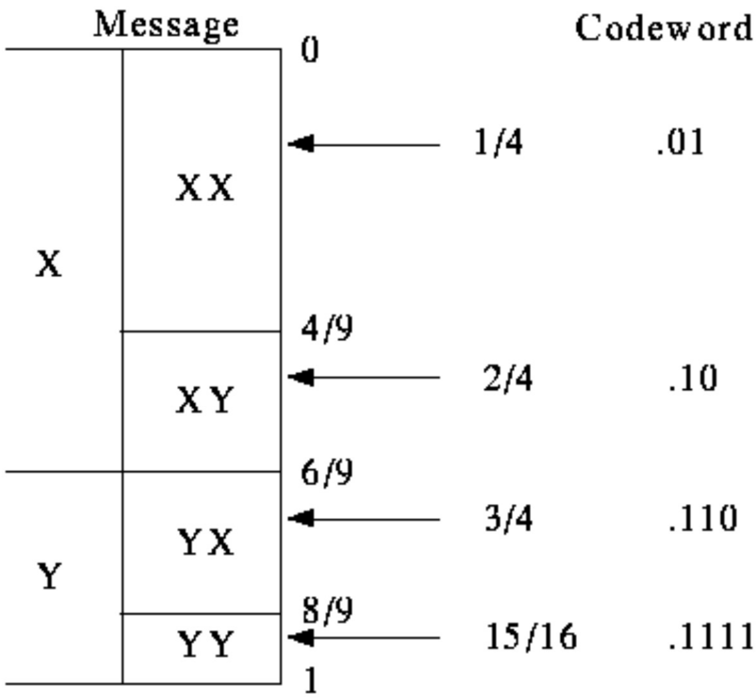
  and

```
prob(X) = 2/3
prob(Y) = 1/3
```

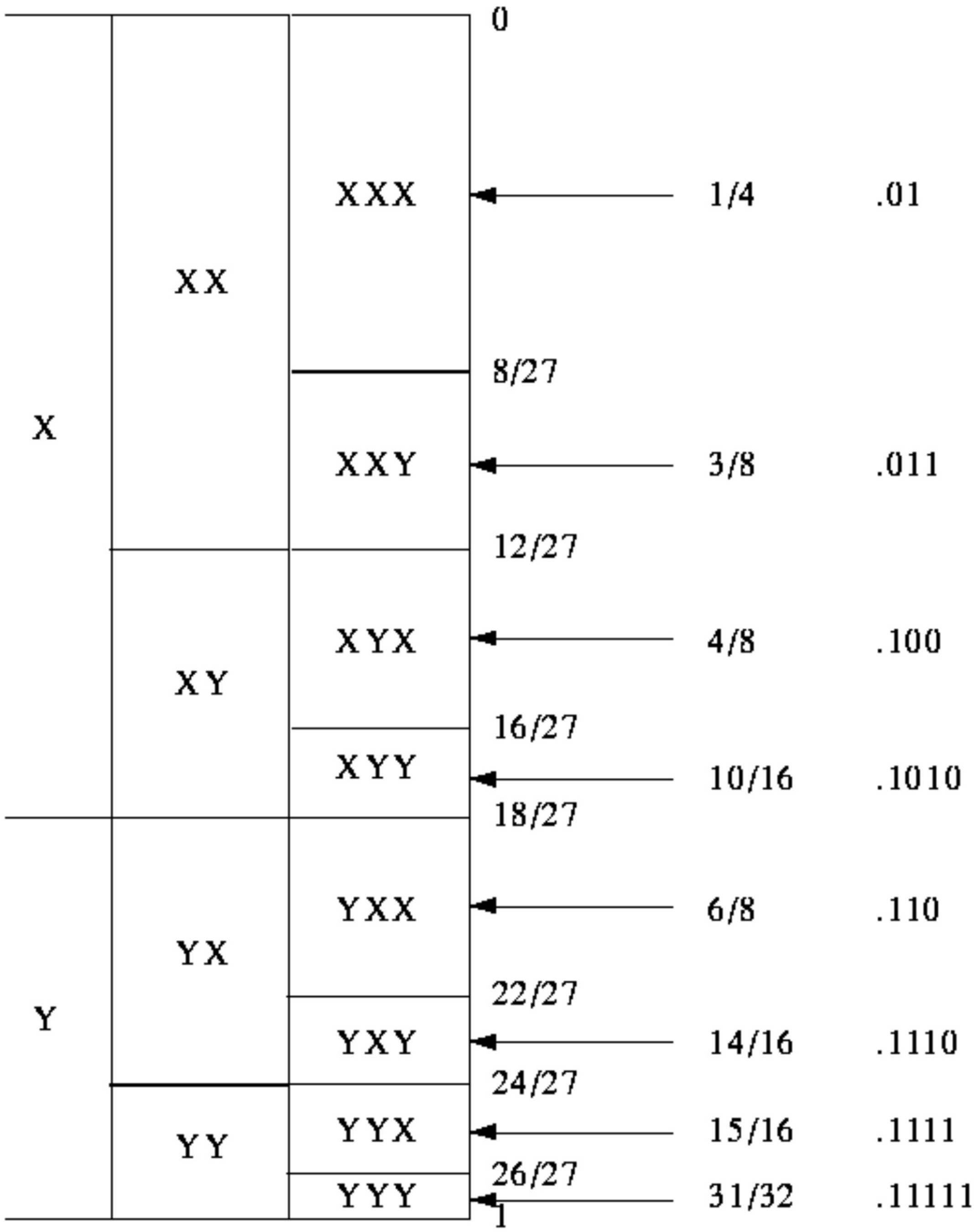
- If we are only concerned with encoding length 2 messages, then we can map all possible messages to intervals in the range [0..1]:



- To encode message, just send enough bits of a binary fraction that uniquely specifies the interval.



- Similarly, we can map all possible length 3 messages to intervals in the range [0..1]:



- Q: How to encode X Y X X Y X ?

  Q: What about an alphabet with 26 symbols, or 256 symbols, ...?
- In general, number of bits is determined by the size of the interval.

  Examples:
  - first interval is 8/27, needs 2 bits -> 2/3 bit per symbol (X)
  - last interval is 1/27, need 5 bits
- In general, need $-\log p$ bits to represent interval of size *p*. Approaches optimal encoding as message length got to infinity.
- Problem: how to determine probabilities?
  - Simple idea is to use adaptive model: Start with guess of symbol frequencies.

Update frequency with each new symbol.

  - Another idea is to take account of intersymbol probabilities, e.g., Prediction by Partial Matching.

- Implementation Notes: Can be CPU and memory intensive; patented.

Next Up Previous

Next: Lempel-Ziv-Welch (LZW) Algorithm **Up:** Lossless Compression Algorithms (Entropy **Previous:** Adaptive Huffman Coding

Dave Marshall
10/4/2001