CSC 310S -- Information Theory

Arithmetic Coding

Pseudocode for Encode and Decode at Three Levels

Notation:

Si =	ith symbol of source alphabet
Pi =	probability of symbol Si
B(Si) =	sum from j = 1 to i- 1 of Pi
T(Si) =	sum from j = 1 to i of Pi
Xi =	ith source symbol to be encoded
H =	upper bound on range
L =	lower bound on range
D =	current width of range
V =	(remaining) value to be decoded

1. Monolithic Message

```
Encode (Xvec)
     L <- 0.0 ;  H <- 1.0
     for j from 1 to length (Xvec)
          D <- H - L
          H <- L + D * T(Xj)
          L <- L + D * B(Xj)
     end for

Decode (V)
     L <- 0.0 ;  H <- 1.0
     M <- ""
     while true do,
          find S* such that
               B(S*) <= (V-L)/(H-L) < T(S*)
          M <- M || S*
          D <- H - L
          H <- L + D * T(S*)
          L <- L + D * B(S*)
     end while
```

In section 3, V, H, L and D, as well as B(Xj) and T(Xj) will be treated as integer representations of bit strings that represent binary fractions. Consequently, the comparisons are based on the higher order bits, and arithmetic operations become shifts or logical operations applied to the bit strings.

2. Continuous Message - Unbounded Word Length

```
Encode (Xvec)
     L <- 0.0 ;  H <- 1.0
     for the next source symbol, Xj,
          D <- H - L
          H <- L + D * T(Xj)
          L <- L + D * B(Xj)
          while L >= .5 or H < .5 do
               if L >= .5 then
                    send "1"
                    L <- 2 * (L - .5)
                    H <- 2 * (H - .5)
               else if H < .5 then
                    send "0"
                    L <- 2 * L
                    H <- 2 * H
               end if
          end while
     end for

Decode (V)
     L <- 0.0 ;  H <- 1.0
     V <- code bits received so far (treated as a binary fraction)
     while received bits remain,
          find S* such that
               B(S*) <= (V-L)/(H-L) < T(S*)
          emit (S*)
          D <- H - L
          H <- L + D * T(S*)
          L <- L + D * B(S*)
          while L >= .5 or H < .5 do
               if L >= .5 then
                    V <- 2 * (V - .5) + next received bit
                    L <- 2 * (L - .5)
                    H <- 2 * (H - .5)
               else if H < .5 then
                    V <- 2 * V  + next received bit
                    L <- 2 * L
                    H <- 2 * H
               end if
          end while
     end while
```

3. Continuous Message - Bounded Word Length (e.g., 8 below)

```
Encode (Xvec)
     L <- "00000000" ;  H <- "11111111"
     C <- 0
     for the next source symbol, Xj,
          D <- H - L
          H <- L + D * T(Xj)
          L <- L + D * B(Xj)
          while L >= "10000000" or H < "10000000"
                     or ( L >= "01000000" and H < "11000000" ) do
               if L >= "10000000" then
                    send "1"
                    send C "0"'s ; C <- 0
                    L <- leftshift(L) with low order "0"
                    H <- leftshift(H) with low order "1"
               else if H < "10000000" then
                    send "0"
                    send C "1"'s ; C <- 0
                    L <- leftshift(L) with low order "0"
                    H <- leftshift(H) with low order "1"
               else
                    assert L >= "01000000" and H <= "11000000"
                    C <- C + 1
                    L <- leftshift(L) AND "01111111" with low order "0"
                    H <- leftshift(H) OR "10000000" with low order "1"
               end if
          end while
     end for

Decode (V)
     L <- "00000000" ;  H <- "11111111"
     V <- code bits received so far (treated as a binary fraction)
     while received bits remain,
          find S* such that
               B(S*) <= (V-L)/(H-L) < T(S*)
          emit (S*)
          D <- H - L
          H <- L + D * T(S*)
          L <- L + D * B(S*)
          while L >= "10000000" or H < "10000000" do
               if L >= "10000000" or H < "10000000" then
                    V <- leftshift(V) with next received bit
                    L <- leftshift(L) with low order "0"
                    H <- leftshift(H) with low order "1"
               else
                    assert L >= "01000000" and H <= "11000000"
                    V <- shiftleft(V) AND "01111111" with next received bit
                    L <- leftshift(L) AND "01111111" with low order "0"
                    H <- leftshift(H) OR "10000000" with low order "1"
               end if
          end while
     end while
```