ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
School of Computer and Communication Sciences

Handout 17 Information Theory and Coding
Midterm Solutions Oct. 28, 2014

PROBLEM 1.

(a) Since C is a function of (T, K), $H(C|T, K) = 0$. Similarly, $H(T|C, K) = 0$.

(b) Expanding $H(CT|K)$ by the chain rule in two ways we find $H(C|K) + H(T|CK) = H(T|K) + H(C|TK)$. From (a) it follows that $H(C|K) = H(T|K)$.

(c) From (b) and that conditioning reduces entropy $H(C) \geq H(C|K) = H(T|K) = H(T)$ where the last equality is because T and K are independent.

(d) As C is a function of T and K we have $H(C|T) \leq H(T, K|T) = H(K|T) \leq H(K)$. Note that the assumption that T and K is independent is not necessary for this conclusion.

(e) From (c) we have $H(C) \geq H(T)$, and from (d) we have $H(K) \geq H(C|T)$. Moreover, we have $H(C|T) = H(T)$ from the independence of C and T, Consequently $H(K) \geq H(C) \geq H(T)$.

PROBLEM 2.

(a) The words of $\mathcal{D}_n$ are $(w_i = b^{i-1}a,\ i = 1, \dots, n)$ and $w_{n+1} = b^n$. Consider an infinite sequence $u_1u_2\dots$. Denote by $k = 0, 1, \dots$ be the number of b's at the beginning of $u_1u_2\dots$. If $k < n$ the sequence $u_1u_2\dots$ is parsed as $w_{k+1}\dots$, and if $k \geq n$ the sequence is parsed as $w_{n+1}\dots$. Thus the dictionary is valid. Also, the prefixes of any w_i are of the form b^j with $j < n$. Since no dictionary word is of the form b^j with a j stricty less than n the dictionary is prefix free.

(b) Since the source is memoryless and since the dictionary is valid and prefix free we know that $(W_i,\ i = 1, 2, \dots)$ are i.i.d., so it is sufficient to consider the statistics of $W = W_1$. Note that length(W) is never strictly larger than n and for $i = 1, \dots, n$, length$(W) \geq i$ if and only if $U_1U_2\dots$ start with b^{i-1}. Thus

$$\Pr(\text{length}(W) \geq i) = \begin{cases} (1-p)^{i-1} & 1 \leq i \leq n \\ 0 & i > n. \end{cases}$$

Thus $E[\text{length}(W)] = \sum_{i=1}^{n}(1-p)^{i-1} = [1-(1-p)^n]/p$.

(c) We know that $H(W) = H(U)E[\text{length}(W)]$. Since $H(U) = h_2(p) = p\log\frac{1}{p} + (1-p)\log\frac{1}{1-p}$, we find $H(W) = h_2(p)\frac{1-(1-p)^n}{p}$.

(d) The Huffman code $\mathcal{C}_n$ for W has the required property.

(e) Parsing the sequence $U_1U_2\dots$ using $\mathcal{D}_n$ and encoding the words by $\mathcal{C}_n$ will yield a scheme that uses

$$\frac{E[\text{length}(\mathcal{C}_n(W)]}{E[\text{length}(W)]} \quad \text{bits/letter.}$$

As

$$\frac{E[\text{length}(\mathcal{C}_n(W)]}{E[\text{length}(W)]} \le \frac{H(W)+1}{E[\text{length}(W)]} = H(U) + \frac{1}{E[\text{length}(W)]} = H(U) + \frac{p}{1-(1-p)^n}$$

and since as n gets large $(1-p)^n \to 0$, we see that we can make the number of bits per letter as close to $H(U)+p$ as desired by taking a large enough n.

PROBLEM 3.

(a) The number of binary sequences of length n that have a given substring of length $m \le n$ is 2^{n-m}: for each of the $n-m$ positions outside the substring we have 2 choices. Consequenty the number of words in A_j that have $C(i)$ as an initial substring (prefix) is $2^{l_j-l_i}$ and similarly for the number of words that have $C(i)$ as a suffix.

(b) The words removed in $(*)$ and $(**)$ are precisely those discussed in (a). As some of those may have been removed in a prior step, and since the words in $(*)$ and $(**)$ may overlap, the number of words removed is at most $2 \cdot 2^{l_j-l_i} = 2^{l_j-l_i+1}$.

(c) The number of words removed from A_i at the time we test $A_i \ne \emptyset$ is at most

$$\sum_{m=1}^{i-1} 2^{l_i-l_m+1} = 2^{l_i} 2 \sum_{m=1}^{i-1} 2^{-l_m} < 2^{l_i}$$

since $\sum_{m=1}^{i-1} 2^{-l_m} < \sum_{m=1}^{k} 2^{-l_m} \le \frac{1}{2}$. As the initial size of A_i was 2^{l_i} we see that A_i is not empty at the time of the test, and thus the algorithm will not fail.

(d) We know from (c) that algorithm will not fail. Since $\mathcal{C}(i)$ is chosen from A_i it is of length l_i. Also, steps $(*)$ and $(**)$ ensure that $\mathcal{C}(i)$ is neither a prefix nor a suffix of $\mathcal{C}(j)$ for $j > i$. On the other hand since $l_1 \le \dots \le l_k$, $\mathcal{C}(i)$ can not be a prefix or suffix of $\mathcal{C}(j)$ for $j < i$ either. So the returned code is fix-free.

(e) Choosing $l(u) = \lceil \log \frac{1}{p(u)} \rceil + 1$ yields

$$\log \frac{1}{p(u)} + 1 \le l_i \le \log \frac{1}{p(u)} + 2.$$

The right hand side inequality ensures $E[l(U)] \le H(U) + 2$, whereas the left hand side inequality ensures $2^{-l(u)} \le p(u)/2$ and thus $\sum_u 2^{-l(u)} \le 1/2$ and consequently the existence of a fix-free code $\mathcal{C}$ with these lengths.