Codes

Jonathan L.F. King
University of Florida, Gainesville FL 32611-2082, USA
squash@ufl.edu
Webpage http://squash.1gainesville.com/
25 January, 2019 (at *02:22*)

(Folks, help me proofread and correct this.)

§An Overview

Formal languages

Use $\varnothing$ for the empty set. An ***alphabet*** $\mathbf{G}$ is a non-empty set, whose members are called "letters"; usually $2 \leqslant |\mathbf{G}| < \infty$. A ***word*** (over an alphabet $\mathbf{G}$) is a *finite* string of letters; Use $\mathbf{G}^\star$ for the set of all words, and use $\boldsymbol{\varepsilon}$ for the ***nullword***, the unique length-zero word. E.g if $\mathbf{G} = \{\mathsf{a}, \mathsf{b}\}$, then $\mathbf{G}^\star$ equals

$$\{\boldsymbol{\varepsilon}, \mathsf{a}, \mathsf{b}, \mathsf{aa}, \mathsf{ab}, \mathsf{ba}, \mathsf{bb}, \mathsf{aaa}, \mathsf{aab}, \ldots\}\,.$$

Write $\mathbf{G}^+$ for $\mathbf{G}^\star \smallsetminus \{\boldsymbol{\varepsilon}\}$.

Concatenation of words $\mathbf{v},\mathbf{z} \in \mathbf{G}^\star$ is written $\mathbf{v}\triangleright\mathbf{z}$ or just $\mathbf{vz}$. Thus $\mathsf{cat}\triangleright\mathsf{nip} = \mathsf{catnip}$. *So* $\mathbf{G}^\star$ *is a semigroup under concatenation, with* $\boldsymbol{\varepsilon}$ *the identity element.* Use $\mathrm{Len}(\mathbf{v})$ or $|\mathbf{v}|$ for the ***length*** of word $\mathbf{v}$, and have $\mathbf{v} \overset{|\cdot|}{>} 3$ mean that $|\mathbf{v}| > 3$. For n a natnum, let $\mathbf{v}^n$ mean the concatenation $\mathbf{v}\mathbf{v}\,.^n.\,\mathbf{v}$. So $\mathbf{v}^0 = \boldsymbol{\varepsilon}$.

A ***language*** is a subset $\mathcal{L} \subset \mathbf{G}^\star$. Here are six distinct languages:

$$\varnothing{=}\{\}\,,\ \{\boldsymbol{\varepsilon}\}\,,\ \{\mathsf{catnip}\}\,,\ \{\mathsf{cat},\mathsf{nip}\}\,,\ \{\boldsymbol{\varepsilon},\mathsf{cat},\mathsf{nip}\}$$
$$\text{and}\quad \{\mathsf{bc},\mathsf{bac},\mathsf{baac},\mathsf{baaac},\ldots\} = \{\mathsf{ba}^n\mathsf{c}\}_{n=0}^{\infty}\,.$$

The first five are finite languages, having cardinalities $0, 1, 1, 2, 3$. Call $\varnothing$ the ***void language*** and call $\{\boldsymbol{\varepsilon}\}$ the ***nullword language***. The "concatenation of *languages*" $\mathcal{K}, \mathcal{L} \subset \mathbf{G}^\star$ is

$$\mathcal{K}\mathcal{L} \;=\; \mathcal{K}\triangleright\mathcal{L} \quad :=\quad \{\mathbf{v}\triangleright\mathbf{w} \mid \mathbf{v}\in\mathcal{K} \text{ and } \mathbf{w}\in\mathcal{L}\}\,.$$

[So $\varnothing\triangleright\mathcal{L} = \varnothing = \mathcal{L}\triangleright\varnothing$ and $\{\boldsymbol{\varepsilon}\}\triangleright\mathcal{L} = \mathcal{L} = \mathcal{L}\triangleright\{\boldsymbol{\varepsilon}\}$.] Let $\mathcal{L}^n$ mean $\mathcal{L}\mathcal{L}\,.^n.\,\mathcal{L}$. Hence $\mathcal{L}^0 = \{\boldsymbol{\varepsilon}\}$, since the nullword language is the identity element for language-concatenation.[♡1]

Define the ***Kleene star*** operator by

$$\mathcal{L}^\star \quad :=\quad \bigcup_{n=0}^{\infty} \mathcal{L}^n\,.$$

In particular $\varnothing^\star = \{\boldsymbol{\varepsilon}\} = \{\boldsymbol{\varepsilon}\}^\star$. Similarly, the ***Kleene plus*** operator is

$$\mathcal{L}^+ \quad :=\quad \bigcup_{n=1}^{\infty} \mathcal{L}^n\,.$$

Hence $[\boldsymbol{\varepsilon}{\in}\mathcal{L}^+] \Leftrightarrow [\boldsymbol{\varepsilon}{\in}\mathcal{L}] \Leftrightarrow [\mathcal{L}^+{=}\mathcal{L}^\star]$. Each Kleene op is ***idempotent***: $[\mathcal{L}^\star]^\star = \mathcal{L}^\star$ and $[\mathcal{L}^+]^+ = \mathcal{L}^+$.

Prefix/Suffix. For words, say "$\mathbf{v}$ is a ***prefix*** of $\mathbf{w}$" if there exists a word $\mathbf{z}$ with $\mathbf{vz} = \mathbf{w}$; write $\mathbf{v} \preccurlyeq \mathbf{w}$ for this relation. If, also, $\mathbf{v} \neq \mathbf{w}$, then $\mathbf{v}$ is a ***proper prefix*** of $\mathbf{w}$, written $\mathbf{v} \prec \mathbf{w}$.

If $\exists \mathbf{z} \in \mathbf{G}^\star$ with $\mathbf{zv} = \mathbf{w}$, then "$\mathbf{v}$ is a ***suffix*** of $\mathbf{w}$". [However, we have no special symbol for the relation.]

Codes

For the time being, a ***code*** $\mathcal{C}$ means a non-void subset $\mathcal{C} \subset \mathbf{G}^+$; usually $2 \leqslant |\mathcal{C}| < \infty$. [Occasionally it is convenient to consider collections $\mathcal{C}$ which *might* own $\boldsymbol{\varepsilon}$. So if all we know is that $\mathcal{C} \subset \mathbf{G}^\star$, then we call $\mathcal{C}$ a ***nullishcode***. If we can later on prove that $\mathcal{C} \not\ni \boldsymbol{\varepsilon}$, then we'll have shown $\mathcal{C}$ to be a code.]

Call $\mathcal{C}$ a ***block code*** if all its codewords have the same length. E.g, $\{\mathsf{FBI}, \mathsf{CIA}\}$ is a blockcode, whereas $\{\mathsf{Go}, \mathsf{Gators}\}$ is *not* a blockcode, –although it *is* (see below) a prefixcode. [Caveat: "block code" is used with

[♡1] Aside: We already knew that $(\mathbf{G}^\star\,,\triangleright\,,\boldsymbol{\varepsilon})$ is a [non-commutative] semigroup. And letting $\mathbb{L}{=}\mathbb{L}_{\mathbf{G}}$ denote the set of *all* languages over $\mathbf{G}$, this generalizes to a [non-commutative] semigroup

$$(\mathbb{L}\,,\triangleright\,,\{\boldsymbol{\varepsilon}\})\,.$$

Of course, we also have the two commutative semigroups $(\mathbb{L}\,,\cup\,,\varnothing)$ and $(\mathbb{L}\,,\cap\,,\mathbf{G}^\star)$.

slightly different meanings in the literature. Perhaps ***constant-length code*** is a more accurate term.]

A code $\mathcal{C}$ is ***uniquely decodable*** (a ***UD-code***) if each code-message $\mathbf{z} \in \mathcal{C}^\star$ has a unique decomposition w.r.t $\mathcal{C}$. That is, if words $\mathbf{v}_j, \mathbf{w}_k \in \mathcal{C}$ satisfy

1.1: *If* $\mathbf{v}_1\mathbf{v}_2 \dots \mathbf{v}_J = \mathbf{z} = \mathbf{w}_1\mathbf{w}_2 \dots \mathbf{w}_K$ *then* $J = K$ *and* $\forall i$: $\mathbf{v}_i = \mathbf{w}_i$.

A ***prefix code*** $\mathcal{C}$ (more accurately called a "prefix-free code") has no codeword being a proper prefix of another. Prefix-codes are UD-codes since, stronger than (1.1), they have the ***RI-UD property*** (the "right-infinite-UD property") that

$$1.2: \quad \left[\begin{array}{c} \mathbf{v}_1\mathbf{v}_2\mathbf{v}_3\cdots = \mathbf{w}_1\mathbf{w}_2\mathbf{w}_3\cdots \\ \text{with each } \mathbf{v}_j, \mathbf{w}_k \in \mathcal{C} \end{array}\right] \Rightarrow \left[\begin{array}{c} \forall i \in \mathbb{Z}_+ : \\ \mathbf{v}_i = \mathbf{w}_i \end{array}\right].$$

We have these non-reversible implications

$$1.2': \quad \text{Block} \implies \text{Prefixcode} \overset{*1}{\implies} \text{RI-UD} \overset{*2}{\implies} \text{UD}.$$

A code showing $(*2)$ non-reversible has these words

$$1.2'': \qquad \mathbf{v} := \mathsf{b}\,,\ \mathbf{w} := \mathsf{ba}\,,\ \mathbf{z} := \mathsf{aa}\,.$$

It is uniquely decodable (Exer. E1), yet fails (1.2), since $\mathbf{vzzz}\cdots$ equals $\mathbf{wzzz}\cdots$. Finally, that $(*1)$ is non-reversible will be shown by (1.5), the "Chris code".

A ***suffix code*** (no codeword is a proper suffix of another) is automatically a UD-code. Dually to $(1.2')$ we have non-reversible implications

$$1.3': \quad \text{Block} \implies \text{Suffixcode} \implies \text{LI-UD} \implies \text{UD},$$

where a ***left-infinite-UD–code*** (a ***LI-UD–code***) satisfies

$$1.3: \quad \left[\begin{array}{c} \cdots \mathbf{v}_{-2}\mathbf{v}_{-1} = \cdots \mathbf{w}_{-2}\mathbf{w}_{-1} \\ \text{with each } \mathbf{v}_j, \mathbf{w}_k \in \mathcal{C} \end{array}\right] \Rightarrow \left[\begin{array}{c} \forall i \in \mathbb{Z}_- : \\ \mathbf{v}_i = \mathbf{w}_i \end{array}\right].$$

Note $(1.2'')$ is an example of a suffixcode which is not a prefixcode.

Bi-infinite. A bi-∞ $\mathbf{G}$-string σ can be viewed as a map $\sigma{:}\mathbb{Z}\to\mathbf{G}$. A ***$\mathcal{C}$-parsing*** of σ is a sequence

$$\cdots < k_{-2} < k_{-1} < k_0 < k_1 < k_2 < k_3 < \cdots$$

of integers st. each substring $\sigma\downarrow_{[k_\ell \,..\, k_{\ell+1})}$ is a codeword, that is, lies $\mathcal{C}$. Write sequence $(k_\ell)_{\ell\in\mathbb{Z}}$ as $\vec{\mathbf{k}}$.

Say that $\mathcal{C}$ has the ***bi-infinite-UD property*** (is ***BI-UD***) if

1.4: *Each bi-∞ string σ which has a $\mathcal{C}$-parsing, has only <u>one</u> $\mathcal{C}$-parsing. I.e, with $\vec{\boldsymbol{\jmath}}$ and $\vec{\mathbf{k}}$ two $\mathcal{C}$-parsings of σ, then the <u>sets</u> $\{j_i\}_{i\in\mathbb{Z}}$ and $\{k_\ell\}_{\ell\in\mathbb{Z}}$ are equal.*

Slightly weaker, consider two parsings $\vec{\boldsymbol{\jmath}}$ and $\vec{\mathbf{k}}$, and let $\mathbf{v}_\ell := \sigma\downarrow_{[j_\ell \,..\, j_{\ell+1})}$ and $\mathbf{w}_\ell := \sigma\downarrow_{[k_\ell \,..\, k_{\ell+1})}$. The ***weak-BI-UD*** property asserts

For each σ and parsings as above, there exists a translation $T \in \mathbb{Z}$ so that:

$$1.4^{\text{weak}}: \qquad \forall \ell \in \mathbb{Z}: \quad \mathbf{v}_{\ell+T} = \mathbf{w}_\ell\,.$$

(I.e, one parsing may be a shift of the other, but the codeword sequences are the same.)

Immediately,

$$1.4': \quad \text{BI-UD} \overset{*3}{\implies} \text{weak-BI-UD} \overset{*4}{\implies} \left[\begin{array}{c}\text{Both LI-UD} \\ \text{and RI-UD}\end{array}\right].$$

The code $\{\mathsf{bbb}\}$ produces $\sigma := \cdots\mathsf{bbbb}\cdots$, which is its only bi-$\infty$ string. This σ has *three* parsings, since the cutpoints $\vec{\boldsymbol{\jmath}}$ can all be mod-3 congruent to -1 or 0 or 1. Yet each parsing yields the *same* codeword sequence, namely $\cdots\boxed{\mathsf{bbb}}\boxed{\mathsf{bbb}}\boxed{\mathsf{bbb}}\cdots$. Hence $(*3)$ is *not reversible*.

The "Pirate code" $\{\mathsf{OH}, \mathsf{HO}\}$ is trivially LI-UD and RI-UD, since it is a blockcode. Yet the Pirate code admits bi-∞ string $\cdots\mathsf{HOHOHOH}\cdots$, which can be parsed as $\cdots\boxed{\mathsf{OH}}\boxed{\mathsf{OH}}\boxed{\mathsf{OH}}\cdots$ or as $\cdots\boxed{\mathsf{HO}}\boxed{\mathsf{HO}}\boxed{\mathsf{HO}}\cdots$, two different codeword sequences. Yup; (*4) *ain't reversible either.*

The "Chris code" (evidently a cry for help)

1.5: $$\{\mathsf{S}, \mathsf{SOS}\}$$

is BI-UD, since each occurrence of "O" must lie in $\boxed{\mathsf{SOS}}$, and every other codeword must be $\boxed{\mathsf{S}}$. Not being a prefixcode, (1.5) proves $(*1)$ not reversible.

Trees. Here, a (rooted) ***tree*** is a set T of nodes, equipped with two operators: Root(T) is the root-node of T. For each node $v \in T$, let Kids(v) be the *set* of children of v. A node w is a ***leaf-node*** if: The set Kids(w) is empty. A tree has the property that, from the root-node, one can get to an arbitrary node, by applying the Kids$(\cdot)$ operator finitely-many times.

Trees T and S are (***tree-***)***isomorphic*** if *there exists* a bijection $f{:}T\to S$ such that:

TI 1: $f(\text{Root}(T)) = \text{Root}(S)$.

TI 2: For each $v \in T$:
$$\{f(k) \mid k \in \text{Kids}(v)\} \;=\; \text{Kids}(f(v)).$$

For a $\Gamma \in \mathbb{Z}_+$, a tree is ***Γ-bounded*** if each node has at most Γ many children. The tree is ***Γ-full*** if every node is either has *no* children [is a ***leaf-node***], or has precisely Γ many children; otherwise, the tree is ***Γ-deficient***.

Inequalities

Kraft proved (2a) for *prefix-codes*, as well as its converse, (2b). McMillan strengthened (2a) to UD-codes.

2: Kraft-McMillan Inequality. *Consider a countable code $\mathcal{C}$ over finite alphabet $\mathbf{G}$. If $\mathcal{C}$ is a UD-code then*

2a: $$\sum\nolimits_{\mathbf{v}\in\mathcal{C}} 1/\Gamma^{\text{Len}(\mathbf{v})} \;\leqslant\; 1\,,$$

where Γ is the number of letters in $\mathbf{G}$.
Conversely, consider posints $\vec{\ell} = (\ell_1, \ell_2, \ldots, \ell_R)$.

2b: *If $\sum_{j=1}^{R} 1/\Gamma^{\ell_j} \;\leqslant\; 1$ then there exists a prefix $\mathbf{G}$-code $\mathcal{C} = (\mathbf{v}_1, \ldots, \mathbf{v}_R)$ with each $\text{Len}(\mathbf{v}_j) = \ell_j$.*

[*The also result holds for infinite tuples $\vec{\ell} = (\ell_1, \ell_2, \ell_3, \ldots,)$ that satisfy $[\sum_{j=1}^{\infty} 1/\Gamma^{\ell_j}] \leqslant 1$.*] ♢

Exer. E2. Give an example of a code, $\mathcal{X}$, that violates (2a). [So $\mathcal{X}$ must fail to be UD.] □

Defn. A code $\mathcal{C}$ is ***weakly-UD*** if the following holds. *For each posint N and words $\mathbf{v}_i, \mathbf{w}_i \in \mathcal{C}$:*

1.1′: *If $\mathbf{v}_1\mathbf{v}_2\ldots\mathbf{v}_N \;=\; \mathbf{w}_1\mathbf{w}_2\ldots\mathbf{w}_N$ then $\forall i$: $\mathbf{v}_i = \mathbf{w}_i$.*

Contrast this with the (1.1) defn of ***UD***. □

Exer. E3. Posting race: Who can be the first to post a code which is weakly-UD, but not UD? □

Soln to E3. Impossible. Consider a non-UD. So there exist codewords $\mathbf{v}_j, \mathbf{w}_k$ and bounds $J \neq K$ for which

$$\mathbf{v}_1\mathbf{v}_2\ldots\mathbf{v}_J \;=\; \mathbf{w}_1\mathbf{w}_2\ldots\mathbf{w}_K \;=:\; \mathbf{z}\,.$$

WLOG $\mathbf{v}_1 \neq \mathbf{w}_1$. [If they are equal, discard both. Continue. You *won't* discard *all* the words, since $J \neq K$.] Let $N := J + K$. Notice that the word $\mathbf{zz}$ has *two* decompositions, each using precisely N many codewords:

$$\mathbf{v}_1\ldots\mathbf{v}_J\,\mathbf{w}_1\ldots\mathbf{w}_K \;=\; \mathbf{w}_1\ldots\mathbf{w}_K\,\mathbf{v}_1\ldots\mathbf{v}_J\,.$$

Moreover, these decompositions differ, since $\mathbf{v}_1 \neq \mathbf{w}_1$. Hence the code turned out *not* be weakly-UD. ♦

Preliminaries for (2a). The below proof uses $S_{n,\ell}$, the number of length-ℓ strings which are concatenations of n many codewords. E.g, consider a code $\mathcal{C} = \{\mathbf{v}, \mathbf{w}, \mathbf{z}\}$ have lengths $5, 7, 8$, respectively.

$S_{1,15} = |\varnothing| = 0.$ $\quad S_{2,15} = |\{\mathbf{wz}, \mathbf{zw}\}| = \,?$
$S_{3,15} = |\{\mathbf{vvv}\}| = 1.$ $\quad S_{4,15} = |\varnothing| = 0.$

Indeed, $S_{n,15}$ is zero for each $n \geqslant 4$. As for $S_{2,15}$: If $\mathbf{wz} = \mathbf{zw}$ then $S_{2,15} = 1$, else $S_{2,15} = 2$. □

Proof of (2a). WLOGenerality, $\mathcal{C}$ is finite. (Exer. E4)
With $\Gamma := |\mathbf{G}|$, our goal is

2a′: $$\sum\nolimits_{\mathbf{v}\in\mathcal{C}} 1/\Gamma^{\text{Len}(\mathbf{v})} \;\overset{?}{\leqslant}\; 1\,.$$

WELOG, suppose the shortest and longest words in $\mathcal{C}$ have lengths 3 and 7. For $n = 1, 2, \ldots,$ each string in $\mathcal{C}^n$ has a length, ℓ, in $[3n\,..\,7n]$; let $S_{n,\ell}$ be the number of such strings. Certainly $S_{n,\ell} \leqslant \Gamma^\ell$, the number of *all* length-ℓ strings over $\mathbf{G}$. So the "generating function"

$$F_n(x) \;:=\; \sum_{\ell=3n}^{7n} [S_{n,\ell}\cdot x^\ell]$$

satisfies, for $x > 0$, that $F_n(x) \leqslant \sum_{\ell=3n}^{7n} \Gamma^\ell \cdot x^\ell$. Thus

$$*\text{:}\;\; F_n(\tfrac{1}{\Gamma}) \;\leqslant\; \sum_{\ell=3n}^{7n} \Gamma^\ell \cdot \tfrac{1}{\Gamma^\ell} \overset{\text{note}}{=\!=} 1 + 7n - 3n \overset{\text{note}}{\leqslant} 5n\,,$$

for each posint n.

Using uniqueness. Fix n and an $\ell \in [3n \,..\, 7n]$.

The coefficient of x^ℓ in $[F_1(x)]^n$ is the number of $\mathcal{C}$-n-parsings of length-ℓ strings, whereas $S_{n,\ell}$ is the number of length-ℓ strings which admit a $\mathcal{C}$-n-parsing.

The UD-hypothesis [actually, only "weakly-UD" is being used] says these two numbers are equal. Hence our two polynomials are equal,

$$\begin{aligned} [F_1(x)]^n &= F_n(x). \quad \textit{So } (*) \textit{ implies} \\ [F_1(\tfrac{1}{\Gamma})]^n &\leqslant 5n\,. \end{aligned}$$

The LhS is exponential in n, whilst the RhS is linear. Thus $F_1(\frac{1}{\Gamma}) \leqslant 1$. Finally, observe that $F_1(\frac{1}{\Gamma})$ is a rewriting of LhS($2a$). ♦

Proof of (2b). We'll show the idea for $\Gamma = 2$. Arrange the lengths as $\ell_1 \leqslant \ell_2 \leqslant \ldots \leqslant \ell_R$. On the full binary-tree of depth $D := \ell_R$, put weight $1/2^D$ on each leaf-node. All the nodes start as ***free***; we will iteratively mark some as ***busy*** as we create words $\mathbf{v}_1, \mathbf{v}_2, \ldots$. Call a node ***very-free*** if it and all its children are *free*, i.e not *busy*.

Let $\mathbf{v}_1$ be the leftmost path down to depth ℓ_1; so $\mathbf{v}_1 = 000\,.^{\ell_1}.\,0$. Mark $\mathbf{v}_1$ and all its children as *busy*. This action creates busy *leaf*-nodes of total weight.

$$2^{D-\ell_1} \cdot \tfrac{1}{2^D} \overset{\text{note}}{=\!=} 1/2^{\ell_1}\,.$$

With $d := \ell_1$, note that

$*$: *Each free node at depth $\geqslant d$ is very-free.*

Let $\mathbf{v}_2$ be the leftmost path to a free node at depth ℓ_2. [So $\mathbf{v}_2$ has $\ell_1 - 1$ many 0s, then a 1, then $\ell_2 - \ell_1$ many 0s.] Mark $\mathbf{v}_2$ and its descendants as *busy*. Now the total weight of busy leaf-nodes is

$$\frac{1}{2^{\ell_1}} + \frac{1}{2^{\ell_2}}\,.$$

Moreover, with $d := \ell_2$, note $(*)$ holds, since $\ell_2 \geqslant \ell_1$.

We'd like to continue using depth ℓ_3, depth ℓ_4, ..., depth ℓ_k, ... The only obstruction at a stage k, is if there is no *free* node at depth ℓ_k. But the total leaf-weight we've used up so far, is

$$W \;:=\; \sum_{j=1}^{k-1} 1/2^{\ell_j}\,.$$

Since this sum is *strictly* less than 1, there exists a free-node at depth ℓ_{k-1}. (*Indeed, the number of such free-nodes is precisely* $[1 - W]/2^{k-1}$.) Finally, since $\ell_k \geqslant \ell_{k-1}$, there is certainly a free-node at depth ℓ_k. ♦

2c: ***Defn.*** For a Γ-code with lengths $\vec{\boldsymbol{\ell}} = (\ell_1, \ldots, \ell_R)$, use

$$\Sigma(\vec{\boldsymbol{\ell}}) \;:=\; \Sigma_\Gamma(\vec{\boldsymbol{\ell}}) \;:=\; \sum_{j=1}^{R} 1/\Gamma^{\ell_j}$$

for its ***Kraft-sum***. Kraft's thm says –if the code is UD– that $\Sigma(\vec{\boldsymbol{\ell}}) \leqslant 1$. If equality, then the code (ditto the tuple) is ***complete***, otherwise it is ***redundant***; more precisely, Γ-***complete*** and Γ-***redundant***.

Given tuples $\vec{\boldsymbol{\ell}} = (\ell_1, \ldots, \ell_R)$ and $\vec{\mathbf{s}} = (s_1, \ldots, s_R)$, write $\vec{\boldsymbol{\ell}} \preccurlyeq \vec{\mathbf{s}}$ if $\boxed{\forall j\colon \ell_j \leqslant s_j}$ and write $\vec{\boldsymbol{\ell}} \prec \vec{\mathbf{s}}$ if at least one of these inequalities is strict. □

Exer. E4.8. A finite Γ-bounded tree T with R many leaves, yields a length-spectrum $\vec{\boldsymbol{\ell}} = (\ell_1, \ldots, \ell_R)$; so terms "$\Gamma$-complete" and "$\Gamma$-redundant" makes sense for the tree. Prove:

2d: Completeness Lemma. *A finite Γ-bounded tree, T, is Γ-complete IFF it is Γ-full.* ◊

In (2e), below, we consider only *binary* prefix-codes; $\Gamma = 2$.

2e: K-M Completeness corollary. *Given a tuple $\vec{\mathbf{s}}$, there exists a complete prefix-code with tuple $\vec{\boldsymbol{\ell}} \preccurlyeq \vec{\mathbf{s}}$.* ◊

Proof. We need but produce a complete $\vec{\boldsymbol{\ell}} \preccurlyeq \vec{\mathbf{s}}$, since Kraft's thm will hand us a prefix-code with lengths $\vec{\boldsymbol{\ell}}$.

It suffices, given a redundant $\vec{\mathbf{s}}$, to produce an $\vec{\boldsymbol{\ell}} \prec \vec{\mathbf{s}}$; after all, there are only finitely-many tuples $\prec\vec{\mathbf{s}}$, so iterating will eventually halt, at a complete tuple.

WLOG, $T := s_1$ is a max length in $\vec{\mathbf{s}}$; so each $1/2^{s_j}$ is a multiple of $1/2^T$, hence so is $\Sigma(\vec{\mathbf{s}})$. As $\vec{\mathbf{s}}$ is redundant, the gap $1 - \Sigma(\vec{\mathbf{s}})$ dominates $1/2^T$. So define $\vec{\boldsymbol{\ell}}$ by $\ell_2 := s_2, \ldots, \ell_R := s_R$, and $\ell_1 := s_1 - 1$. ♦

Exer. E5. POSTING RACE: Does (2e) hold for larger alphabet-sizes? If so, how does the proof need to be modified? □

Expected coding-length

A ***probability distribution*** on a codeword-set $\mathcal{C}$ is a map $\mathsf{P}\colon\mathcal{C}\to[0,1]$ st.

3a: $$\sum_{\mathbf{v}\in\mathcal{C}} \mathsf{P}(\mathbf{v}) \;=\; 1\,.$$

We will usually discard from the code all probability-zero words. In practice, then, a "probability distribution" is a map $\mathsf{P}{:}\mathcal{C}\to(0,1)$ fulfilling (3a)

The ***expected***[♡2] ***coding length*** of $\mathcal{C}$ is

3b: $$\mathrm{ECL}(\mathcal{C}) \;:=\; \sum_{\mathbf{v}\in\mathcal{C}} \mathsf{P}(\mathbf{v})\cdot \mathrm{Len}(\mathbf{v})\,.$$

E.g, consider code $\mathcal{C} := \{\mathbf{w}_1,\ldots,\mathbf{w}_4\}$ where

3c: $\mathbf{w}_1 := 00$, $\mathbf{w}_2 := 010$, $\mathbf{w}_3 := 011$, $\mathbf{w}_4 := 1$,

where $\mathsf{P}(\mathbf{w}_4)=\frac{1}{2}$, and the other three words have probability $\frac{1}{6}$. Then $\mathrm{ECL}(\mathcal{C})$ is then

$$\tfrac{1}{2}\cdot 1 \;+\; \tfrac{1}{6}\cdot[2+3+3] \;=\; \tfrac{11}{6}\,.$$

Codemap. A ***source alphabet*** Ω, also called a "message set", might be

$$\{\mathsf{a},\mathsf{b},\ldots,\mathsf{z},\,.\,,\mathit{Space}\}\,,$$

or might be $\{\mathsf{tank},\mathsf{ship},\ldots,\mathsf{plane}\}$. Fixing a ***code-alphabet*** $\mathbf{G}$, a map $f{:}\Omega\to\mathbf{G}^+$ is a ***codemap*** (or ***cipher***) if

i: f is injective, *and*

ii: $\mathcal{C} := \mathrm{Range}(f)$ is a code. [Phrased this way, so that if we change our defn of "code" for a given context, then the defn of ***codemap*** changes with it.]

Every adjective applying to a code, also applies to a codemap; e.g, "a ***block/prefix/UD*** codemap".

ECL. Consider a [finite or countably-infinite] message set Ω and a probability distribution $\mathsf{P}{:}\Omega\to[0,1]$. A codemap $f{:}\Omega\to\mathbf{G}^+$ puts a probability-distribution on $\mathcal{C} := \mathrm{Range}(f)$ by assigning, for $\mathbf{w}\in\mathcal{C}$,

4a: $$\mathsf{P}(\mathbf{w}) \;:=\; \mathsf{P}\big(f^{-1}(\mathbf{w})\big)\,.$$

Thus the code has an *expected coding-length*, which we may write as

$$\mathrm{ECL}(\mathcal{C}) \quad\text{or}\quad \mathrm{ECL}(f)\,.$$

[♡2] "Expected" is what probabilists use for "average".

MECL. Use ***MECL*** for Minimum ECL. Consider a *finite* prob-vector $\vec{\mathbf{p}} = (p_1,\ldots,p_L)$. A code [for the moment, assume a binary code] $\mathcal{C} = (\mathbf{v}_1,\ldots,\mathbf{v}_L)$ has

3b′: $$\mathrm{ECL}(\mathcal{C}) \;=\; \sum_{j=1}^{L} p_j\cdot\mathrm{Len}(\mathbf{v}_j)\,.$$

The minimum of (3b′) taken over *all* prefix-codes, or over all UD-codes, we will call

4b: $$\text{PC-MECL}(\vec{\mathbf{p}}) \quad\text{and}\quad \text{UD-MECL}(\vec{\mathbf{p}})\,,$$

respectively. Evidently

4c: $$\text{PC-MECL}(\vec{\mathbf{p}}) \;\geqslant\; \text{UD-MECL}(\vec{\mathbf{p}})$$

since, for UD-codes, we are taking a minimum over the larger collection of codes. By the way, I'll sometimes use $\mathrm{MECL}(\vec{\mathbf{p}})$ as a synonym for $\text{UD-MECL}(\vec{\mathbf{p}})$.

The minimum in (3b′) *depends on* $\Gamma := |\mathbf{G}|$, the number of letters in our code alphabet. [We can compress English more by coding into a 3-letter alphabet, rather than a 2-letter alphabet.] To indicate the dependency on cardinality Γ, we may write

4d: $$\text{PC-MECL}_\Gamma(\vec{\mathbf{p}}) \quad\text{and}\quad \text{UD-MECL}_\Gamma(\vec{\mathbf{p}})\,.$$

Huffman codes

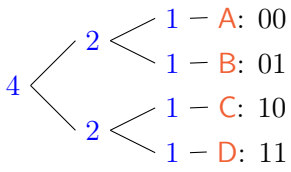
(Binary HCs will be described in class.)

Interpret a tuple such as (3:A 1:B 5:C) as putting prob-distribution $(\frac{3}{9},\frac{1}{9},\frac{5}{9})$ on letters (A, B, C); the 9 is the sum of the ***weights***, $3+1+5$.

Our convention is that the branch going up-right is labeled with bit 0, and the up-left with bit 1.

Non-uniqueness of Huffman Codes. Frequency-tuple $F := (1{:}\mathsf{A}\ 1{:}\mathsf{B}\ 1{:}\mathsf{C}\ 1{:}\mathsf{D})$ admits HC

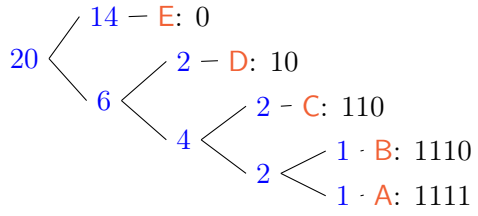
5a:



But F also admits each other permutation of $\{\mathsf{A},\mathsf{B},\mathsf{C},\mathsf{D}\}$ being attached to those leaves. So this Freq-tuple admits several HCs.

For a more interesting example, consider Frequency-tuple $F' := (1{:}A\ 1{:}B\ 2{:}C\ 2{:}D\ 14{:}E)$. This admits HC $\mathcal{C}_1$:

5b:



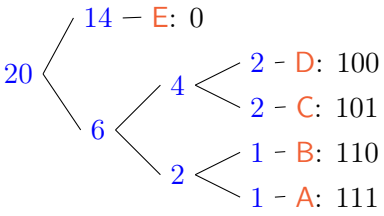
So $20{\cdot}\mathrm{ECL}(\mathcal{C}_1)$ equals

$$\overbrace{1{\cdot}2{\cdot}4}^{B,A} + \overbrace{2{\cdot}1{\cdot}3}^{C} + \overbrace{2{\cdot}1{\cdot}2}^{D} + \overbrace{14{\cdot}1{\cdot}1}^{E} \;=\; 32\,.$$

Thus $\mathrm{ECL}(\mathcal{C}_1) = \frac{32}{20} = \frac{8}{5}$ bits-per-letter.

Our F' also admits HC $\mathcal{C}_2$:

5c:



Thus $20{\cdot}\mathrm{ECL}(\mathcal{C}_2)$ equals

$$\overbrace{1{\cdot}2{\cdot}3}^{B,A} + \overbrace{2{\cdot}2{\cdot}3}^{D,C} + \overbrace{14{\cdot}1{\cdot}1}^{E} \;=\; 32\,.$$

We see that $\mathrm{ECL}(\mathcal{C}_2) = \mathrm{ECL}(\mathcal{C}_1)$. It is worth noticing that codes $\mathcal{C}_1$ and $\mathcal{C}_2$ are not only different, they are not even tree-isomorphic. □

6: HC-same-ECL Thm. *Fix a probability L-vec $\vec{\mathbf{p}}$, with $L \geqslant 2$. Then all $\vec{\mathbf{p}}$-HCs have the same* ECL. ♢

Proof. We proceed by induction on L, with proposition

$R(L)$: *For every prob. L-vec $\vec{\mathbf{q}}$: Each two $\vec{\mathbf{q}}$-HCs have the same* ECL.

The base $L{=}2$ case is easy, since the only Huffman-tree is $\mathrm{Root} \lessdot^{\text{Prob.}}_{\text{Prob.}}$ whose ECL is 1.

Induction step. Fix an $L \geqslant 3$ st. $R(L{-}1)$.

Let $J := L{-}2$. Given $\vec{\mathbf{p}}$, let α, β denote its two lowest probabilities,[♡3] and write $\vec{\mathbf{p}}$ as $(\alpha, \beta, p_1, \ldots, p_J)$.

Consider two HCs, $\mathcal{C}$ and $\mathcal{X}$, with length-spectra that I have written above and below $\vec{\mathbf{p}}$, here.

$$\begin{array}{rc} \mathcal{C}: & D\ \ D\ \ d_1\ \ d_2\ \ \ldots\ \ d_J \\ & (\alpha,\ \beta,\ p_1,\ p_2,\ \ldots,\ p_J) \\ \mathcal{X}: & Y\ \ Y\ \ y_1\ \ y_2\ \ \ldots\ \ y_J\,. \end{array}$$

So code $\mathcal{C}$ assigns length-D codewords to the first two nodes it joins, which have probs α and β. Computing

†:
$$\begin{aligned} \mathrm{ECL}(\mathcal{C}) &= D{\cdot}\alpha + D{\cdot}\beta + \textstyle\sum_{i=1}^{J}[d_i \cdot p_i]\,; \\ \mathrm{ECL}(\mathcal{X}) &= Y{\cdot}\alpha + Y{\cdot}\beta + \textstyle\sum_{i=1}^{J}[y_i \cdot p_i]\,. \end{aligned}$$

After joining two nodes, the codes now recursively act on $\vec{\mathbf{q}} := (\alpha{+}\beta, p_1, p_2, \ldots, p_J)$ and assign length-spectra as follows:

$$\begin{array}{rc} \mathcal{C}: & D{-}1\ \ d_1\ \ d_2\ \ \ldots\ \ d_J \\ & (\alpha{+}\beta,\ p_1,\ p_2,\ \ldots,\ p_J) \\ \mathcal{X}: & Y{-}1\ \ y_1\ \ y_2\ \ \ldots\ \ y_J\,. \end{array}$$

Since $\vec{\mathbf{q}}$ is an $[L{-}1]$-vector, proposition $R(L{-}1)$ says that the above two ECLs are equal, i.e

‡:
$$\begin{aligned} &[D{-}1]{\cdot}[\alpha{+}\beta] + \textstyle\sum_{i=1}^{J}[d_i \cdot p_i] \\ &\quad = [Y{-}1]{\cdot}[\alpha{+}\beta] + \textstyle\sum_{i=1}^{J}[y_i \cdot p_i]\,. \end{aligned}$$

And this implies equality in the two RhSs of (†). ♦

7a: Depth Lemma. *Fix a probability L-vector $\vec{\mathbf{p}}$, and a $\vec{\mathbf{p}}$–PC-MECL. Consider two leaf-nodes with probabilities x and α, at depths D and D', respectively. If $x > \alpha$, then necessarily $D \leqslant D'$.* ♢

Exer. E6. Prove the above Depth Lemma. □

7b: Huffman's theorem.

i: *HCs are PC-MECLs.*

ii: *HCs are UD-MECLs.* ♢

[♡3] They might be equal; indeed, perhaps $\beta = \alpha$, with 8 nodes all having probability α. We are not picking two *nodes*; we are picking two **probabilities**. In particular, I am not assuming that HCs $\mathcal{C}$ and $\mathcal{X}$ join the same two nodes, at the first step.

Pf of (i). We induct on L, with proposition

HUFF(L): *Each probability L-vector $\vec{\mathbf{q}}$, admits a Huffman Code which is a PC-MECL.*

The base L=2 case is immediate, since the only tree is Root $\prec$ Prob. Prob., which *is* a Huffman-tree.

Induction step. Fix an $L \geqslant 3$ st. HUFF(L−1). Fix $\vec{\mathbf{p}}$, a prob. L-vector, and consider a $\vec{\mathbf{p}}$–PC-MECL, viewed as a tree.

Let $\alpha \leqslant \beta$ denote the two smallest probabilities of $\vec{\mathbf{p}}$. At the tree's deepest level, D, consider two joined leaf-nodes, and call their probabilities x and y. It suffices to show:

$*$: *We can permute the probabilities of the leaves, without changing the* ECL, *so that, now, these two nodes have probabilities α and β.*

For then, we collapse these two into a single node, producing prob.-vec $\vec{\mathbf{q}} := (\alpha+\beta, p_2, p_3, \ldots, p_{L-1})$. By the induction hypothesis, there is a $\vec{\mathbf{q}}$-HC which is a $\vec{\mathbf{q}}$–PC-MECL. Expanding the collapsed node back into $\prec \begin{smallmatrix}\alpha\\ \beta\end{smallmatrix}$ automatically produces a Huffman-tree[♡4]. which is a $\vec{\mathbf{p}}$–PC-MECL. And all HCs have the same ECL, by (6).

Establishing $(*)$**.** If $x = \alpha$, then leave that leaf-node alone. Otherwise, $x > \alpha$. Now the Depth Lemma, (7a), says that α can't be shallower than x, so [since x is at max depth], every α-node has to be at D, the deepest level. Switch some α-leaf with our x-leaf.

This does not change the ECL, *since the nodes are at the same depth.*

Now our joined-pair is $\prec \begin{smallmatrix}\alpha\\ y\end{smallmatrix}$. Do the same operation with y w.r.t β. *Now* our joined-pair is $\prec \begin{smallmatrix}\alpha\\ \beta\end{smallmatrix}$, as desired. ♦

Exer. E7. Prove (ii), that every HC is a UD-MECL. □

Exer. E8. POSTING RACE: Give Proof or CEX: *If prefix code $\mathcal{C}$ is a PC-MECL, then $\mathcal{C}$ is a Huffman code.* □

Entropy/Distropy

Define $\boldsymbol{\eta}: [0,1] \to [0,\infty)$ by $\boldsymbol{\eta}(x) := x \cdot \log_2(1/x)$, and extend by continuity, so that $\boldsymbol{\eta}(0) = 0$. (Use l'Hôpital's rule, if you like.)

The ***distribution entropy***, which I call ***distropy***, of a probability-vector $\vec{\mathbf{v}}$ is

$$\mathcal{H}(\vec{\mathbf{v}}) := \sum_{p \in \vec{\mathbf{v}}} \boldsymbol{\eta}(p)\,.$$

For a probability-distr $\mathsf{P}()$ on a code[♡5] $\mathcal{C}$, then, $\mathcal{H}(\mathsf{P})$ equals $\sum_{\mathbf{v}\in\mathcal{C}} \boldsymbol{\eta}(\mathsf{P}(\mathbf{v}))$.

8: Jensen's inequality. *On an interval $J \subset \mathbb{R}$, consider points $Q_{\mathbf{v}} \in J$, for each $\mathbf{v}$ in a countable indexing-set $\mathcal{C}$. We have a probability-distr $\mathsf{P}()$ on $\mathcal{C}$. Then for each convex-down fnc $\mathscr{L}: J \to \mathbb{R}$*

8a: $$\mathscr{L}\Big(\sum_{\mathbf{v}\in\mathcal{C}} \mathsf{P}(\mathbf{v}) \cdot Q_{\mathbf{v}}\Big) \geqslant \sum_{\mathbf{v}\in\mathcal{C}} \mathsf{P}(\mathbf{v}) \cdot \mathscr{L}(Q_{\mathbf{v}})\,.$$

Now suppose $\mathscr{L}$ is strictly convex-down. Then:

8b: *Equality in* (8a) *IFF the probability-distr is concentrated on a single point.*

IOWords, having removed all zero-probability elements from $\mathcal{C}$, the map $\mathbf{v} \mapsto Q_{\mathbf{v}}$ is constant. ♢

Proof. See picture on blackboard. ♦

9: Distropy UD-code Inequality. *Fix a binary code $\mathcal{C}$ and probability distribution $\mathsf{P}: \mathcal{C} \to (0,1)$. If $\mathcal{C}$ is uniquely decodable, then*

9a: $$\mathrm{ECL}(\mathcal{C}) \geqslant \mathcal{H}(\mathsf{P})\,.$$

There is equality in (9a) *IFF*

9b: $$\forall \mathbf{v} \in \mathcal{C}:\ \mathsf{P}(\mathbf{v}) = 1/2^{\widehat{\mathbf{v}}}\,.$$ ♢

[♡4] The permuting of probabilities, because it is done recursively, can permute interior-nodes of the tree. So the final Huffman-tree can be non-isomorphic to the original PC-MECL tree. This kind of argument is called ***tree surgery***.

[♡5] For comparison with (binary) distropy/entropy, we will usually be examining a ***binary code***; a code over a 2-symbol alphabet, $\mathbb{B}$. (Typically, $\mathbb{B} = \{0,1\}$.) So a *binary code* is a subset $\mathcal{C} \subset \mathbb{B}^+$.

Pf of (9a). Let "$\sum_{\mathbf{v}}$" mean "$\sum_{\mathbf{v}\in\mathcal{C}}$" and $\widehat{\mathbf{v}}$ mean $\text{Len}(\mathbf{v})$. With $\mathscr{L}() := \log_2()$, note $\text{ECL}(\mathcal{C})$ equals $\sum_{\mathbf{v}} \mathsf{P}(\mathbf{v})\widehat{\mathbf{v}}$, which equals $\sum_{\mathbf{v}} \mathsf{P}(\mathbf{v})\mathscr{L}(2^{\widehat{\mathbf{v}}})$. Consequently, we can write $\mathcal{H}(\mathsf{P}) - \text{ECL}(\mathcal{C})$ as

$$\begin{aligned}&\Big[\textstyle\sum_{\mathbf{v}} \mathsf{P}(\mathbf{v})\mathscr{L}\big(\frac{1}{\mathsf{P}(\mathbf{v})}\big)\Big] - \Big[\textstyle\sum_{\mathbf{v}} \mathsf{P}(\mathbf{v})\mathscr{L}\big(2^{\widehat{\mathbf{v}}}\big)\Big]\\ &= \textstyle\sum_{\mathbf{v}} \mathsf{P}(\mathbf{v})\,\mathscr{L}\Big(\frac{1}{\mathsf{P}(\mathbf{v})}\cdot\frac{1}{2^{\widehat{\mathbf{v}}}}\Big)\,.\end{aligned}$$

Since $\mathscr{L}()$ is strictly convex-down, Jensen's inequality applies to say

†:
$$\begin{aligned}\mathcal{H}(\mathsf{P}) - \text{ECL}(\mathcal{C}) &\leqslant \mathscr{L}\Big(\textstyle\sum_{\mathbf{v}} \mathsf{P}(\mathbf{v})\cdot\frac{1}{\mathsf{P}(\mathbf{v})}\frac{1}{2^{\widehat{\mathbf{v}}}}\Big)\\ &\overset{\text{note}}{=\!=} \mathscr{L}\Big(\textstyle\sum_{\mathbf{v}} 1\Big/2^{\widehat{\mathbf{v}}}\Big)\,.\end{aligned}$$

By (2a) the Kraft-McMillan inequality, $\sum_{\mathbf{v}} 1/2^{\widehat{\mathbf{v}}} \leqslant 1$. And $\mathscr{L}()$ is order-preserving. Thus the above yields

$$\mathcal{H}(\mathsf{P}) - \text{ECL}(\mathcal{C}) \;\leqslant\; \mathscr{L}(1) \;=\; 0\,,$$

as desired. ♦

Pf of (9b). Suppose $\text{ECL}(\mathcal{C}) = \mathcal{H}(\mathsf{P})$. This forces equality in Kraft, so $\sum_{\mathbf{v}} 1/2^{\widehat{\mathbf{v}}} = 1$, and in Jensen's, so the map $\mathbf{v} \mapsto \frac{1}{\mathsf{P}(\mathbf{v})}\cdot\frac{1}{2^{\widehat{\mathbf{v}}}}$ is constant; say κ. Thus $\mathsf{P}(\mathbf{v})\cdot\kappa = 1/2^{\widehat{\mathbf{v}}}$, for each $\mathbf{v}$. Summing over all $\mathbf{v} \in \mathcal{C}$ implies that $1\cdot\kappa = 1$. Hence $\kappa = 1$. ♦

Sardinas-Patterson Algorithm. An example of a UD-code [indeed, it is a suffixcode], for which the SarPat algorithm eventually cycles (as it must), but not with the empty prefix-list, is

$$\{\mathsf{bc}\,,\,\mathsf{b}\,,\,\mathsf{Xc}\,,\,\mathsf{cX}\}\,.$$

(On hold. . .)

Decoding-delay for UD-codes. Consider a long word $\mathbf{w}$ which is the initial part. . .
(On hold. . .)

§Index for "JK Codes notes"

Filename: `Problems/NumberTheory/jk-codes.latex`
As of: *Tuesday 15Mar2016.* Typeset: *25Jan2019* at *02:22.*