- Suponhamos que $\mathbf{p}$ é uma distribuição de probabilidades a q símbolos e que $r < q$.
- Existe um único $r' \in \{2, 3, \ldots, r\}$ tal que

$$r' \equiv q \pmod{r-1},$$

e é este o número das probabilidades mais pequenas que agrupamos em $\mathbf{p}$ para obter o $\mathbf{p}'$ seguinte, no qual temos $q' = q - r' + 1$ probabilidades.

- Note-se que, no caso de $r = 2$, temos sempre $r' = 2$, ou seja para a construção de códigos de Huffman binários, agrupamos sempre as duas probabilidades mais pequenas.
- Note-se também que $q' = q - r' + 1 \equiv 1 \pmod{r-1}$, pelo que na etapa seguinte teremos $r'' = r$. Por outras palavras, só precisamos de calcular r' na primeira etapa, sendo nas seguintes o agrupamento feito para as r probabilidades mais pequenas.

Proposição 3.9 *Seja $q > 2$ e seja $\mathbf{p} = (p_1, p_2, \ldots, p_q)$ uma distribuição de probabilidades a q símbolos com $p_1 \geqslant p_2 \geqslant \cdots \geqslant p_q$. Então existe um código prefixo compacto r-ário $c_1, c_2, \ldots, c_q$ satisfazendo as seguintes propriedades:*

1. $|c_1| \leqslant |c_2| \leqslant \cdots \leqslant |c_q|$;
2. *as últimas r' palavras-código (com $r' \in \{2, \ldots, r\}$ e $r' \equiv q \pmod{r-1}$) coincidem exceto na sua última letra e não há nenhuma outra palavra-código que junto com elas tenha a mesma propriedade.*

Prova. Dado um código $C = (c_1, c_2, \ldots, c_q)$, se se tiver $|c_i| > |c_j|$ com $i < j$, então trocando c_i e c_j não aumentamos o comprimento médio $\sum_k p_k |c_k|$. Logo, a partir de qualquer código compacto podemos obter um código compacto que satisfaz a condição (1) por simples permutação das palavras-código.
De entre os códigos compactos para a nossa distribuição de probabilidades da fonte, consideremos um para o qual $\sum_i |c_i|$ é mínimo. Pelo Teorema McMillan, vale a desigualdade de Kraft, ou seja temos $\sum_i r^{-|c_i|} \leqslant 1$, pelo que

$$\Delta = r^{|c_q|} - \sum_{i=1}^{q} r^{|c_q| - |c_i|} \geqslant 0.$$

$$\Delta = r^{|c_q|} - \sum_{i=1}^{q} r^{|c_q|-|c_i|} \geqslant 0.$$

Se for $\Delta \geqslant r-1$, então os comprimentos $(|c_1|, |c_2|, \ldots, |c_{q-1}|, |c_q|-1)$ também satisfazem a desigualdade de Kraft, pelo que há algum código prefixo r-ário com estes comprimentos, sendo o seu comprimento médio no máximo o de C, pelo que continua a ser compacto (e $p_q = 0$), o que contradiz a minimalidade da soma $\sum_i |c_i|$.
Logo $0 \leqslant \Delta \leqslant r-2$, ou seja $r - \Delta \in \{2, 3, \ldots, r\}$, uma vez que Δ é um inteiro.
Seja t o número de palavras em C de comprimento $|c_q|$.

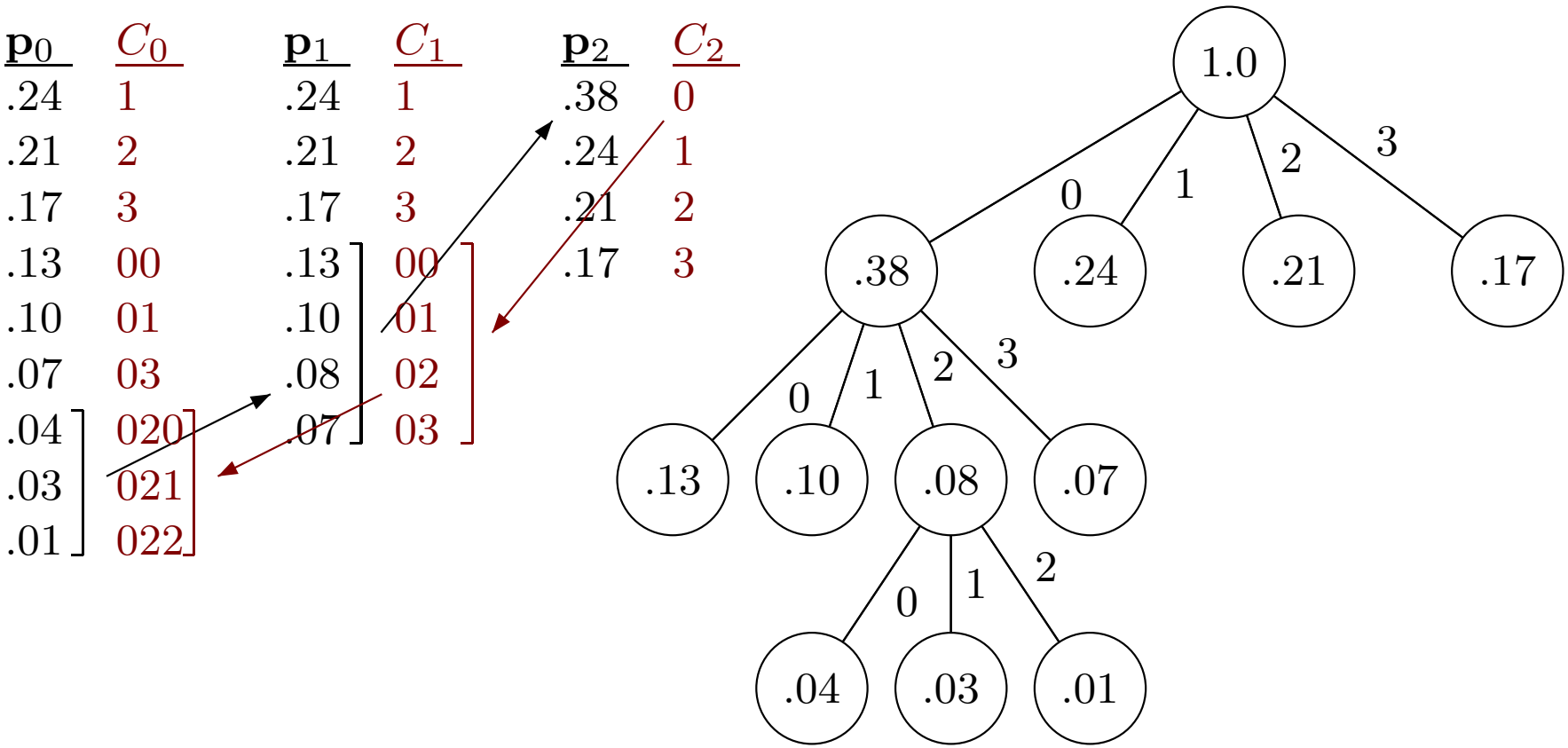
Se fosse $t = 1$, então poderíamos apagar a última letra de c_q mantendo um código prefixo, o que novamente estaria em contradição com a minimalidade da soma $\sum_i |c_i|$.
Logo, $t \geqslant 2$. Note-se que $\Delta = r^{|c_q|} - \sum_{i=1}^{q} r^{|c_q|-|c_i|} \equiv -t \pmod{r}$, ou seja $t \equiv r - \Delta \pmod{r}$. Como $2 \leqslant r - \Delta \leqslant r$ e $t \geqslant 2$, concluímos que $t \geqslant r - \Delta$.

Por outro lado, temos $\Delta = r^{|c_q|} - \sum_{i=1}^{q} r^{|c_q|-|c_i|} \equiv 1 - q \pmod{r-1}$, ou seja $r - \Delta \equiv q \pmod{r-1}$. Logo, $r - \Delta = r'$, onde r' é definido no enunciado.
Como $t \geqslant r - \Delta$, segue que $t \geqslant r'$, ou seja há pelo menos r' palavras-código de comprimento máximo.

Se $c_q = wa$ para uma letra a, então podemos acrescentar a C todas as palavras que ainda lá não estejam da forma wb sem violar a condição de termos um código prefixo.
Uma vez que $t \equiv r' \pmod{r}$, eliminando, se necessário, palavras a mais e reordenando as que ficam, conseguimos assim satisfazer a condição (2) do enunciado.
□

Exemplo

O exemplo anterior do processo de codificação de Huffman pode ser resumido na seguinte árvore:



A redução de Huffman

- Dada uma distribuição de probabilidades $\mathbf{p}$ sobre um conjunto finito, designemos por $L_r(\mathbf{p})$ o comprimento médio de um código compacto r-ário para a fonte com distribuição de probabilidades $\mathbf{p}$.
- Dados $\mathbf{p} = (p_1, p_2, \ldots, p_q)$ com $p_1 \geqslant p_2 \geqslant \cdots \geqslant p_q$ e $r \leqslant q$, a *redução r-ária de* $\mathbf{p}$ é definida como sendo

$$\mathbf{p}' = (p_1, p_1, \ldots, p_{q-r'}, p_{q-r'+1} + \cdots + p_q)$$

  onde $r' \in \{2, 3, \ldots, r\}$ é tal que $r' \equiv q \pmod{r-1}$.
- Formalizando o método de Huffman apresentado no exemplo anterior, temos o seguinte resultado.

Proposição 3.10 *Seja* $\mathbf{p} = (p_1, p_2, \ldots, p_q)$ *uma distribuição de probabilidades com* $p_1 \geqslant p_2 \geqslant \cdots \geqslant p_q$ *e seja* $\mathbf{p}'$ *a sua redução de Huffman* r*-ária. Seja* $C' = \{c_1, c_2, \ldots, c_{q-r'}, c\}$ *um código prefixo* r*-ário compacto para* $\mathbf{p}'$ *com os comprimentos das palavras-código em ordem crescente. Então*

$$C = \{c_1, c_2, \ldots, c_{q-r'}, c0, c1, \ldots, c\langle r' - 1\rangle\}$$

é um código compacto para $\mathbf{p}$*. Além disso, tem-se*

$$L_r(\mathbf{p}) = L_r(\mathbf{p}') + p_{q-r'+1} + \cdots + p_q.$$

Prova. Começamos por notar que C é um código prefixo e que o seu comprimento médio é $L_r(\mathbf{p}') + p$ onde $p = p_{q-r'+1} + \cdots + p_q$. Em particular, tem-se

$$L_r(\mathbf{p}) \leqslant L_r(\mathbf{p}') + p. \tag{7}$$

Pela Proposição 3.9, existe um código prefixo compacto r-ário para $\mathbf{p}$, $D = \{d_1, \ldots, d_q\}$, tal que as últimas r' palavras-código têm um prefixo comum d de comprimento $|d_q| - 1$ e todas as restantes são mais curtas ou não têm d como prefixo. Então $\{d_1, \ldots, d_{q-r'}, d\}$ é um código prefixo para $\mathbf{p}'$ cujo comprimento médio é $L_r(\mathbf{p}) - p$ e, portanto, tem-se

$$L_r(\mathbf{p}') \leqslant L_r(\mathbf{p}) - p. \tag{8}$$

Combinando as desigualdades (7) e (8), obtemos a igualdade $L_r(\mathbf{p}) = L_r(\mathbf{p}') + p$ do enunciado. Logo C é um código compacto para $\mathbf{p}$ pois o seu comprimento médio é $L_r(\mathbf{p})$. □

Algoritmo de Huffman

- O algoritmo (recursivo) de Huffman para construir um código prefixo compacto r-ário para uma fonte q-ária com distribuição de probabilidades $\mathbf{p}$ pode agora ser formalizado como segue:
  - se $q \leqslant r$, tomamos o código $\{0, 1, \ldots, q-1\}$ e temos $L_r(\mathbf{p}) = 1$;
  - se $q > r$ então, aplicando o algoritmo à fonte com $q - r'$ símbolos e distribuição de probabilidades $\mathbf{p}'$, obtemos um código prefixo compacto r-ário C' para esta fonte; a transformação $C' \mapsto C$ da Proposição 3.10 fornece então um código prefixo compacto r-ário para $\mathbf{p}$.
- ***Exercício:*** mostre que o número de reduções de Huffman a aplicar no algoritmo de Huffman é $\lceil (q-r)/(r-1) \rceil$; mais precisamente, supondo que $q \geqslant r$, ao fim deste número de reduções obtemos uma fonte com r símbolos e, portanto, uma codificação r-ária.

Exemplo

Seja $\mathbf{p} = (.9, .1)$ e seja $r = 2$. Temos $H_2(\mathbf{p}) \simeq 0.469$. Considerando as sucessivas n-ésimas extensões da fonte, com distribuições de probabilidade $\mathbf{p}^n$, e aplicando o algoritmo de Huffman, obtemos os seguintes comprimentos médios por símbolo:

$$L_2(\mathbf{p}) = 1, \frac{1}{2}L_2(\mathbf{p}^2) = 0.645, \frac{1}{3}L_2(\mathbf{p}^3) \simeq 0.533, \frac{1}{4}L_2(\mathbf{p}^4) \simeq 0.493.$$

Eis um pequeno programa recursivo em *Mathematica* para fazer estes cálculos:

Programa no *Mathematica* para o cálculo de $L_r(\mathbf{p})$

```
controle=False;
huff[r_,prob_]:=
    Module[{q=Length[prob],rlinha=r,novo},
           If[!controle,res=1;rlinha=Mod[q,r-1,2]];
           If[q<=r,
                controle=False;
                res,
              controle=True;
              novo=Plus@@Take[prob,-rlinha];
              res=res+novo;
              huff[r,
                   Sort[Join[Take[prob,{1,q-rlinha}],{novo}],
                        Greater]]]];

ext[n_,x_]:=ext[n,x]=
    If[n===1,Sort[x,Greater],
       Module[{y=ext[n-1,x],t},
              t=Table[y[[i]]x[[j]],{i,Length[y]},{j,Length[x]}];
              Sort[Flatten[t],Greater]]];
```

Para o exemplo anterior, `Table[huff[2,ext[n,{.9,.1}]]/n,{n,4}]` produz o seguinte resultado:

$$\{1, 0.645, 0.532667, 0.49255\}.$$

Codificação aritmética

- Os códigos de Huffman são compactos mas, a menos que a fonte q-ária seja especial, o que em geral não há razões para esperar que seja o caso, podem estar longe de serem 100% eficientes.
- Pelo Teorema de Shannon, esta dificuldade pode ser ultrapassada trabalhando com a extensão da fonte de ordem n para n suficientemente grande.
- Mas dessa forma temos de codificar uma fonte q^n-ária, o que em particular, se um canal de comunicação tiver de ser usado, nos obriga a usá-lo para transmitir uma tabela de codificação com q^n entradas.
- Caso, adicionalmente, se verifiquem alterações das probabilidades da fonte, i.e., ela não seja estacionária, o processo poderá ter de ser repetido com regularidade.

- Uma alternativa consiste em dispor de um sistema que proceda à codificação/descodificação sem necessitar de ter à partida uma tabela para o efeito.
- Adicionalmente, a restrição que até aqui temos assumido da fonte não ter memória é excessivamente restritiva.
- Para o sistema de *codificação aritmética* que passamos a descrever, assumimos que temos uma fonte em que, para N fixado, se conhece a probabilidade $P(s_{ij}|s_{i1}s_{i2}\cdots s_{i,j-1})$ (para $j = 1, \ldots, N$) de ser emitido o símbolo s_{ij} tendo acabado ser emitida a mensagem $s_{i1}s_{i2}\cdots s_{i,j-1}$.

- Na codificação aritmética binária, a cada mensagem $s_{i1}s_{i2}\cdots s_{iN}$, via a sua probabilidade, corresponde um subintervalo de $[0,1[$, no qual é escolhido de forma canónica um elemento da forma $0.a_1a_2\cdots a_n$, na base 2, sendo $a_1a_2\cdots a_n$ a mensagem correspondente.
- A subdivisão do intervalo $[0,1[$ para a construção dos subintervalos em causa prossegue de forma iterativa à medida que a mensagem vai sendo produzida:
  - inicialmente, a subdivisão é feita atribuindo sucessivamente aos símbolos $s_1,\ldots,s_q$ os intervalos, começando o primeiro em 0, de comprimentos $P(s_1),\ldots,P(s_q)$;
  - no passo seguinte, se no primeiro passo tiver sido recebido o símbolo s_i, o subintervalo correspondente a esse s_i é subdividido em subintervalos, começando no extremo inferior com comprimentos $P(s_1|s_i)P(s_i),\ldots,P(s_q|s_i)P(s_i)$.
  - e assim sucessivamente, cada novo símbolo determinando um novo subintervalo dentro do subintervalo anterior.

Exemplo

Para a fonte sem memória

Símbolo	s_1	s_2	s_3
$P(s_i)$	0.2	0.5	0.3

os subintervalos sucessivos correspondentes à mensagem $s_2 s_2 s_1$ são

Letra seguinte	subintervalo
s_2	$[0.2, 0.7[$
s_2	$[0.3, 0.55[$
s_1	$[0.3, 0.35[$

Em binário, o intervalo correspondente à mensagem é

$$[0.01001100,\ 0.0101001[$$

no qual o número 0.0101 é aquele que tem a expansão binária mais curta.
Podemos então codificar $s_2 s_2 s_1$ por 0101.

Descodificação aritmética

Para a descodificação da mensagem $a_1 \cdots a_n$,

- vamos sucessivamente acumulando as probabilidades $P(s_1), \ldots, P(s_i)$ até que $\sum_{k=1}^{i-1} P(s_k) \leqslant 0.a_1 \cdots a_n < \sum_{k=1}^{i} P(s_k)$, o que determina o símbolo inicial s_i;
- supondo que o segmento inicial $s_{i1} \cdots s_{i,j-1}$ da mensagem já foi identificado pela determinação do intervalo $[m, M[$ correspondente, acumulamos as probabilidades $\sum_{k=1}^{i-1} P(s_{kj} | s_{i1} \cdots s_{i,j-1})$ enquanto este número não exceder $\frac{0.a_1 \cdots a_n - m}{M-m}$, o que determina o símbolo seguinte s_{ij}.

Exemplo

No exemplo anterior, da fonte sem memória $\{s_1, s_2, s_3\}$ com probabilidades $\mathbf{p} = (0.2, 0.5, 0.3)$, a mensagem 0101, pode ser descodificada como segue:

- ao número em binário 0.0101 corresponde na base 10 a dízima $\frac{1}{4} + \frac{1}{16} = 0.25 + 0.0625 = 0.3125$;
- de $0.2 \leqslant 0.3125 < 0.2 + 0.5$, concluímos que o primeiro símbolo é s_2;
- passamos a trabalhar com o número $\frac{0.3125-0.2}{0.5} = 0.225$ e de $0.2 \leqslant 0.225 < 0.2 + 0.5$ concluímos que o segundo símbolo também é s_2;
- o número seguinte é $\frac{0.225-0.2}{0.5} = 0.05$ e de $0 \leqslant 0.05 < 0.2$ concluímos que o símbolo seguinte é s_1;
- o número seguinte é $\frac{0.05-0}{0.2} = 0.25$, pelo que o símbolo seguinte seria s_2;
- o número seguinte é $\frac{0.25-0.2}{0.5} = 0.1$, pelo que o símbolo seguinte seria s_1;
- o número seguinte é $\frac{0.1-0}{0.2} = 0.5$, pelo que o símbolo seguinte seria s_2;
- o número seguinte é $\frac{0.5-0.2}{0.5} = 0.6$, pelo que o símbolo seguinte seria s_2;
- o número seguinte é $\frac{0.6-0.2}{0.5} = 0.8$, pelo que o símbolo seguinte seria s_3;
- o número seguinte é $\frac{0.8-0.7}{0.3} \simeq 0.333\cdots$, pelo que o símbolo seguinte seria s_2; ETC???

Afinal, onde termina a mensagem?
Para o saber, podemos seguir um de vários mecanismos:

- enviando um símbolo tipo *EOF* (*end of file*) no final da mensagem;
- começando por enviar o número de símbolos da mensagem (o que poderá ser confuso se a própria mensagem puder ser constituída exclusivamente por dígitos);
- enviando sempre mensagens constituídas por blocos do mesmo comprimento (como os pacotes da Internet).

Este último mecanismo tem ainda a vantagem de nos permitir determinar com que precisão devemos calcular as frações que aparecem nos sucessivos passos do algoritmo de descodificação.

Com um destes tipos de mecanismos para parar o processo de descodificação, obtemos o seguinte resultado:

Teorema 3.11 *O processo de codificação/descodificação aritmética descrito acima permite recuperar as mensagens iniciais e envolve um número de operações proporcional ao comprimento da mensagem.*