

# **Apunts de Teoria de la Informació i Codificació**

**Rafel Farré i Cirera**

**tardor 2003**



# Índex

<b>1</b>	<b>Informació i Entropia</b>	<b>5</b>
1.1	Repàs de probabilitat . . . . .	5
1.2	Informació i Entropia . . . . .	6
1.3	El lema de Gibbs . . . . .	8
<b>2</b>	<b>Codis: desigualtat de Kraft</b>	<b>9</b>
2.1	Codis . . . . .	9
2.1.1	Codis prefixos . . . . .	10
2.2	La desigualtat de Kraft . . . . .	11
2.2.1	El Teorema de Kraft . . . . .	11
2.2.2	El Teorema de McMillan . . . . .	12
<b>3</b>	<b>Compressió estadística: Huffman</b>	<b>15</b>
3.1	Fonts sense memòria . . . . .	15
3.2	Codis de Huffman . . . . .	16
3.2.1	Longitud mitja i entropia . . . . .	18
3.3	El primer Teorema de Shannon . . . . .	19
3.3.1	Extensions d'una font . . . . .	19
3.3.2	El primer teorema de Shannon . . . . .	19
<b>4</b>	<b>Mètodes de diccionari: LZ77 i LZ78</b>	<b>21</b>
4.1	LZ77 . . . . .	21
4.1.1	GNU zip . . . . .	23
4.2	LZ78 . . . . .	23
<b>5</b>	<b>Codis de Bloc</b>	<b>25</b>
5.1	Distància . . . . .	25
5.2	Paràmetres d'un codi . . . . .	26
5.3	Detecció i correcció d'errors . . . . .	27
5.4	Exemple: un codi que corregeix 2 errors . . . . .	29
5.5	Cotes dels paràmetres d'un codi . . . . .	32

<b>6</b>	<b>Cossos finits</b>	<b>35</b>
6.1	El cos $\mathbb{F}_p$	35
6.2	Polinomis irreductibles a $\mathbb{F}_p$	36
6.3	Els cossos $\mathbb{F}_q$	38
<b>7</b>	<b>Codis lineals</b>	<b>43</b>
7.1	Codis lineals: primeres propietats	43
7.2	Matriu generadora d'un codi lineal	44
7.2.1	codificació a partir d'una matriu generadora	46
7.3	matriu de control	47
7.3.1	Relació entre matriu generadora i matriu de control	48
7.3.2	Distància mínima i matriu de control	49
7.3.3	Detecció i correcció per síndrome	50
<b>8</b>	<b>Codis Polinomials</b>	<b>53</b>
8.1	Ràfegues d'errors	53
8.2	Matrius generadores i de control	54
8.2.1	a partir de $g(x)$	54
8.2.2	sistemàtica	55
8.2.3	a partir del zeros de $g(x)$	56
8.3	Codis Cíclics	56
8.3.1	Polinomi de control	57
<b>9</b>	<b>Codis de Reed-Solomon</b>	<b>59</b>
9.1	Definició i propietats bàsiques	59
9.2	Polinomis localitzador i avaluador	60
9.3	El mètode PGZ millorat	61
9.4	El mètode de Suniyama o Euclides Estès	64

# Capítol 1

## Informació i Entropia

### 1.1 Repàs de probabilitat

Un espai de probabilitat finit és un conjunt finit  $\Omega$  més una funció  $P : 2^\Omega \rightarrow [0, 1]$  satisfent les propietats següents:

- 1  $P(\Omega) = 1$
- 2  $P(A \cup B) = P(A) + P(B)$  si  $A$  i  $B$  són subconjunts de  $\Omega$  disjunts.

Aquí  $2^\Omega$  denota el conjunt de les parts (subconjunts) de  $\Omega$ . Els subconjunts de  $\Omega$  reben el nom de successos. En general quan escrivim  $P(A)$  se suposarà que  $A$  és un succés.

En un espai de probabilitat (com que només treballarem amb finits no ho repetirem més) se satisfan les propietats següents:

- $P(A^C) = 1 - P(A)$ .
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- $P(A_1 \cup \dots \cup A_r) = P(A_1) + \dots + P(A_r)$  si els  $A_i$  són disjunts dos a dos.

Si  $\Omega = \{x_1, \dots, x_n\}$  i  $p_i := P(x_i)$ , els  $p_i$  compliran

$$p_1 + \dots + p_n = 1, p_i \geq 0 \quad (1.1.1)$$

i si  $A = \{x_{i_1}, \dots, x_{i_k}\}$ , la probabilitat de  $A$  ve donada per

$$P(A) = p_{i_1} + \dots + p_{i_k}. \quad (1.1.2)$$

Quan  $\{p_1, \dots, p_n\}$  satisfan (1.1.1) els anomenem distribució de probabilitat. De fet, donar un espai de probabilitat és equivalent a donar un conjunt i una distribució que tinguin el mateix nombre d'elements (més la manera en que es corresponen). Se sobreentén que  $p_i$  és la probabilitat del succés elemental  $x_i$ . Llavors la probabilitat d'un succés es calcula mitjançant (1.1.2).

Per exemple, si llancem dos daus diferents,  $\Omega = \{1, 2, 3, 4, 5, 6\} \times \{1, 2, 3, 4, 5, 6\}$  i si suposem que els daus no tenen defectes la probabilitat de que surti  $(i, j)$  és igual a  $\frac{1}{36}$ . Aquest és un exemple de distribució uniforme: tots el  $p_i$  valen igual i per (1.1.1) tenim que  $p_i = \frac{1}{n}$ . Si volem calcular la probabilitat de que la suma dels daus valgui 4, el succés és  $A = \{(1, 3), (2, 2), (3, 1)\}$  i  $P(A) = \frac{3}{36} = \frac{1}{12}$ .

És habitual presentar els espais de probabilitat com a variables aleatòries. La manera més senzilla de fer-ho és la següent: Es considera una funció  $X$ , que anomenarem variable aleatòria, i que pren tots els valors de  $\Omega$  (la seva imatge o rang és  $\Omega$ ). Si  $A$  és un subconjunt de  $\Omega$ , la probabilitat  $P(X \in A)$  de que  $X$  prengui valors en  $A$  és  $P(A)$ , en particular  $P(X = x_i) = p_i$ .

### probabilitat condicionada

Quan tenim dos successos  $A$  i  $B$ ,  $P(A | B)$ , la probabilitat de  $A$  condicionada a  $B$  és la probabilitat de que surti  $A$  sabent que ha sortit  $B$  i es calcula per la fórmula

$$P(A | B) = \frac{P(A \cap B)}{P(B)}.$$

Aquí se sobreentén que  $P(B) \neq 0$ .

Si tenim una partició  $B_1, \dots, B_k$  de  $\Omega$ , es poden recuperar la probabilitat de  $A$  i les probabilitats de  $B_i$  condicionada a  $A$  a partir dels valors  $P(B_i)$  i  $P(A | B_i)$  mitjançant les fórmules següent:

- $P(A) = \sum_{i=1} P(A | B_i)P(B_i)$  fórmula de les probabilitats totals
- $P(B_i | A) = \frac{P(A | B_i)P(B_i)}{\sum_{j=1} P(A | B_j)P(B_j)}$  fórmula de Bayes

## 1.2 Informació i Entropia

Els logaritmes, si no diem el contrari són en base 2, i.e.,  $\log x$  indica el logaritme en base 2 de  $x$ . En termes del logaritme neperià  $\ln$ ,  $\log x = \frac{\ln x}{\ln 2}$ .

**Definició 1.2.1.** La incertesa o quantitat d'informació d'un succés  $A$  és  $I(A) := -\log p(A)$ .

Justifiquem una mica aquesta definició. Observem que la funció  $I(x) = -\log x$  té tres propietats bàsiques:

- és decreixent
- $I(0) = \infty$  i  $I(1) = 0$
- $I(xy) = I(x) + I(y)$ .

Justificar la 'incertesa' és relativament fàcil, un succés és més incert com menys probable sigui. D'altra banda, un succés segur (de probabilitat 1) té incertesa 0 i un succés impossible (de probabilitat 0) té incertesa infinita. Podríem, però,

haver pres en la definició d'incertesa qualsevol funció amb les dues primeres propietats anteriors. Justificar la 'informació' és un pèl més delicat. D'entrada podem argumentar que saber que ha ocorregut un succés molt probable, no ens esta donant gaire informació. Per exemple, en un país com el nostre on el bon temps és freqüent, que demà farà sol no és notícia. Sí que ho és, en canvi, el fet de que nevi, un succés poc probable. La tercera propietat ens diu que si dos successos  $A$ ,  $B$  són independents llavors  $I(A \cap B) = I(A) + I(B)$ . Això ens diu que la quantitat d'informació que tenim al saber que han succeït  $A$  i  $B$  és la suma de les informacions de cada un d'ells.

Per exemple si l'experiment consisteix en tirar dos daus diferents, la incertesa del succés  $(4, 2)$  és  $I(4, 2) = -\log(\frac{1}{36}) = \log 36$  i la informació del succés 'la suma dels daus és 4' val  $-\log \frac{1}{12} = \log 12$ . Aquesta incertesa està mesurada en bits. Si uséssim altres logaritmes estariem canviant les unitats d'informació (multipliquem per una constant). L'unitat d'informació usant logaritmes neperians és el *nat* i usant logaritmes decimals és el *dècit* o *Hartley*. Per exemple 1 nat són  $\log e = 1.4427$  bits i un dècit són  $\log 10 = 3.322$  bits. Per tenir una idea de la mesura del bit observen que en l'experiment corresponent a tirar una moneda, la informació del succés cara és  $-\log \frac{1}{2} = 1$ .

L'entropia d'una variable aleatòria és la mitjana de les incerteses dels valors que pren:

**Definició 1.2.2.** Si la variable aleatòria  $X$  pren els valors  $x_1, \dots, x_n$  amb probabilitats  $P(X = x_i) = p_i$ , l'entropia de  $X$ , que denotarem per  $H(X)$  es defineix

$$H(X) := -\sum_{i=1}^n p(X = x_i) \log P(X = x_i) = -\sum_{i=1}^n p_i \log p_i.$$

L'entropia només depèn de la distribució de probabilitat  $p_1, \dots, p_n$  i posarem  $H(p_1, \dots, p_n) = -\sum_{i=1}^n p_i \log p_i$ .

Per exemple, l'entropia de l'experiment  $X$  corresponent a tirar dos daus és  $H(X) = 36(-\frac{1}{36} \log(\frac{1}{36})) = \log 36$ .

Si una variable aleatòria que pren  $n$  valors té distribució uniforme (tots els successos  $x_i$  són equiprobables) llavors  $p_i = \frac{1}{n}$  i

$$H(X) = -\sum_{i=1}^n \frac{1}{n} \log \frac{1}{n} = -\frac{n}{n} \log \frac{1}{n} = \log n.$$

A vegades ens convindrà mesurar l'entropia en altres unitat a part del bit. Quan prenem logaritmes en base  $q$  escriurem

$$H_q(X) = H_q(p_1, \dots, p_n) = -\sum_{i=1}^n p_i \log_q(p_i).$$

La base  $q$  del logaritme ha de més gran que 1. Recordem que  $\log_q x = \ln x / \ln q = \log x / \log q$ . Així quan canviem de base en el logaritme, multipliquem (o dividim) per un nombre positiu.

### 1.3 El lema de Gibbs

El següent resultat, conegut com a lema de Gibbs el farem servir per demostrar propietats de l'entropia. Recordem aquí un parell de propietats del logaritme neperià que són força clares si feu la gràfica de la funció  $\ln x$ :

- $\ln x \leq x - 1$  per qualsevol  $x \geq 0$
- $\ln x = x - 1$  si i només si  $x = 1$ .

**Lema 1.3.1 (Lema de Gibbs).** Si  $p_1, \dots, p_n$  és una distribució de probabilitat i  $q_1, \dots, q_n$  satisfan

$$q_1 + \dots + q_n \leq 1, \quad q_i \geq 0,$$

llavors

$$-\sum_{i=1}^n p_i \log_q p_i \leq -\sum_{i=1}^n p_i \log_q q_i.$$

A més val la igualtat si i només si  $p_i = q_i$  per tots els  $i$ .

*Demostració.* Podem suposar que tots els  $p_i > 0$ , sinó eliminem els nuls de la suma. També podem suposar el mateix per tots els  $q_i$ , altrament la suma de la dreta és  $\infty$ .

Hem de veure que  $-\sum_{i=1}^n p_i \log_q p_i + \sum_{i=1}^n p_i \log_q q_i \leq 0$ , i que val zero si i només si els  $p_i$  són iguals als  $q_i$ .

Podem reemplaçar  $\log_q$  pel logaritme neperià, ja que ambdós costats de la desigualtat només quedaran multiplicats per una constant positiva.

Ara bé,

$$\begin{aligned} -\sum_{i=1}^n p_i \ln p_i + \sum_{i=1}^n p_i \ln q_i &= \sum_{i=1}^n p_i \ln \frac{q_i}{p_i} \leq \\ \sum_{i=1}^n p_i \left( \frac{q_i}{p_i} - 1 \right) &= \sum_{i=1}^n (q_i - p_i) = \sum_{i=1}^n q_i - 1 \leq 0 \end{aligned}$$

per les propietats del logaritme abans esmentades. A més hi ha igualtat si i només si  $\ln \frac{q_i}{p_i} = \frac{q_i}{p_i} - 1$  per cada  $i$ , i això només passa quan coincideixen  $p_i$  i  $q_i$ .  $\square$

El màxim de l'entropia s'assoleix amb la distribució de probabilitat uniforme:

**Proposició 1.3.2.** Si  $X$  és una variable aleatòria que pren  $n$  valors, llavors  $H(X) \leq \log n$  i hi ha igualtat si i només si  $X$  segueix la distribució uniforme.

*Demostració.* Apliquem el lema de Gibbs amb  $q_1 = \dots = q_n = 1/n$  i tenim  $H(X) \leq -\sum_{i=1}^n p_i \log \frac{1}{n} = \log n \sum_{i=1}^n p_i = \log n$  i hi ha igualtat si i només si tots els  $p_i = \frac{1}{n}$ .  $\square$

De fet es pot veure que al apropar-se a la distribució uniforme, l'entropia augmenta.

## Capítol 2

# Codis: desigualtat de Kraft

### 2.1 Codis

Un alfabet és un conjunt finit de símbols que també anomenarem lletres. Per exemple l'alfabet llatí és  $\{a, b, \dots, y, z\}$  i l'alfabet binari és  $\{0, 1\}$ . Les paraules d'un alfabet  $A$  són seqüències de lletres de  $A$ . Al conjunt de paraules de  $A$  el denotarem per  $A^*$ , incloent la paraula buida. Si  $C \subseteq A^*$ ,  $C^*$  denotarà el conjunt de paraules de  $A^*$  que s'obtenen concatenant paraules de  $C$ . Observem que  $C^* \subseteq A^*$ .

Es general, quan codifiquem (i descodifiquem), el que fem es reemplaçar certes seqüències de símbols per altres. Més concretament la situació és la següent: tenim dos alfabet  $A$  i  $B$  i reemplaçem paraules de  $B$  per paraules de  $A$ , és a dir tenim una funció  $f : B^* \rightarrow A^*$ , que en principi no té perquè estar definida sobre tot  $B^*$ . A més, per tal de poder descodificar demanarem que  $f$  sigui injectiva. Normalment la  $f$  ve donada per el que s'anomena *esquema de codificació*. El cas més senzill consisteix a reemplaçar les lletres de  $B$  per paraules prefixades de  $A$ . Per reemplaçar paraules de  $B$  el que fem és concatenar. Molt sovint això es fa per blocs de lletres enlloc de lletres.

Per exemple, si  $B = \{a, b, c, d\}$  i  $A = \{0, 1\}$  i considerem l'esquema de codificació

$$\begin{aligned} a &\mapsto 0 \\ b &\mapsto 01 \\ c &\mapsto 10 \\ d &\mapsto 11, \end{aligned} \tag{2.1.1}$$

llavors  $f(abcd) = 0011011 = f(aadad)$  i  $f$  no és injectiva. El problema és que la paraula 0011011 la podem descomposar de dues maneres diferents com a concatenació de paraules de  $\{0, 01, 10, 11\}$ :  $0011011 = 0/01/10/11 = 0/0/11/0/11$ .

Si  $B = \{b_1, \dots, b_n\}$  i l'esquema és  $b_i \mapsto c_i \in A^*$   $f$  serà injectiva quan tota paraula de  $\{c_1, \dots, c_n\}^*$  s'escriu de manera única com a concatenació de paraules de  $\{c_1, \dots, c_n\}$ . Això és el que anomenarem codi.

**Definició 2.1.1.** Un *codi*  $C$  de l'alfabet  $A$  és un subconjunt de  $A^*$  amb la

propietat següent: si  $c_1/\dots/c_r = d_1/\dots/d_s$  amb tots els  $c_i, d_i \in \mathcal{C}$  llavors  $r = s$  i  $c_i = d_i$  per  $i = 1, \dots, r$

Un exemple important de codi és quan totes les paraules de  $\mathcal{C}$  tenen la mateixa longitud. Aquests codis s'anomenen *codis de bloc* i es fan servir per detectar i corregir errors. La descodificació es fa trencant la paraula en blocs de la mateixa longitud (la de les paraules de codi) i mirant quina lletra els hi correspon (si n'hi ha). Si en l'exemple anterior canviem l'esquema per

$$\begin{aligned} a &\mapsto 00 \\ b &\mapsto 01 \\ c &\mapsto 10 \\ d &\mapsto 11 \end{aligned} \tag{2.1.2}$$

la paraula 01000010101100 es descomposa en 01/00/00/10/10/11/00 i per tant prové de *baaccdc*.

### 2.1.1 Codis prefixos

Un altre cas important de codis són els anomenats *prefixos* o *instantanis*. Una paraula  $v$  de  $A$  és un prefix de  $w$  quan hi ha  $x \in A^*$  amb  $w = vx$ .

**Definició 2.1.2.** Un subconjunt finit  $\mathcal{C}$  de  $A^*$  es diu *prefix* si no hi ha dues paraules a  $\mathcal{C}$  que siguin prefixa l'una de l'altre.

**Proposició 2.1.3.** *Els conjunts prefixos són codis.*

*Demostració.* Si  $c_1/\dots/c_r = d_1/\dots/d_s$  llavors  $c_1$  és prefix de  $d_1$  o a l'inrevés, i per tant han de ser iguals. Esborrant-les tenim  $c_2/\dots/c_r = d_2/\dots/d_s$  i un altre cop  $c_2 = d_2$ . Si ho repetim arribem a que  $r = s$  i cada  $c_i$  és igual a  $d_i$ .  $\square$

Aquests codis es diuen *instantanis* perquè per descodificar anem llegint els primers símbols (d'esquerra a dreta) fins que ens trobem una paraula del codi, que la descodifiquem per la lletra corresponent i tornem a començar. Si ens estan enviant un missatge llarg, això permet anar descodificant sobre la marxa i no tenir que esperar a rebre tot el missatge per descodificar. Per exemple, l'esquema de codificació

$$\begin{aligned} a &\mapsto 0 \\ b &\mapsto 10 \\ c &\mapsto 110 \\ d &\mapsto 1110 \end{aligned} \tag{2.1.3}$$

és prefix, i si hem rebut la paraula 0110010011100, aquesta es descomposa com 0/110/0/10/0/1110/0 i per tant prové de la paraula *acabada*. Si observem aquest codi veiem que no tota paraula es pot descomposar com a concatenació de paraules del codi. Per exemple la paraula 1111 no es pot descomposar.

També hi ha codis no prefixos:

*Exemple 2.1.4.*  $\{0, 01\}$  i  $\{0, 010, 0110, 01110, 011110\}$  són codis no prefixos.

## 2.2 La desigualtat de Kraft

Un alfabet  $q$ -ari és un alfabet de  $q$  lletres. Un codi  $q$ -ari és un codi sobre un alfabet  $q$ -ari.

Quan volem comprimir el que farem és usar un codi amb les paraules ‘al més curt possibles’ en un cert sentit que precisarem més endavant. Òbviament un codi no pot tenir moltes paraules i que totes siguin curtes. Per exemple, en un alfabet binari hi ha com a molt dues paraules de longitud 1, 4 de longitud 2, 8 de longitud 3, ... En aquest sentit, la següent pregunta és important: donats  $\ell_1, \dots, \ell_n$ , hi ha algun codi que tingui les paraules precisament d’aquestes longituds? La desigualtat (2.2.1) que s’anomena desigualtat de Kraft juga un paper essencial al respecte, i la resposta a la pregunta és la següent: si i només si es satisfà la desigualtat de Kraft. Això és el que es demostra a les proposicions 2.2.2 i 2.2.3.

### 2.2.1 El Teorema de Kraft

**Lema 2.2.1.** *Si  $\mathcal{C} = \{c_1, \dots, c_n\}$  és un codi  $q$ -ari prefix on  $\ell_1 \leq \dots \leq \ell_n$  són les longituds de las paraules de  $\mathcal{C}$  i*

$$\sum_{i=1}^n q^{-\ell_i} < 1$$

*llavors hi ha una paraula  $c$  de longitud  $\ell_n$  tal que  $\{c_1, \dots, c_n, c\}$  és codi prefix.*

*Demostració.* Tot el que hem de fer es veure que podem trobar una paraula de longitud  $\ell_n$  que no té com a prefix cap de les paraules de  $\mathcal{C}$  i afegir-l’hi.

Cada paraula  $c_i$  és prefix de  $q^{\ell_n - \ell_i}$  paraules de longitud  $\ell_n$  i una paraula no pot tenir com a prefix dues paraules de  $\mathcal{C}$  diferents. Per tant, el nombre de paraules de longitud  $\ell_n$  que no són prefix de cap paraula de  $\mathcal{C}$  és:

$$q^{\ell_n} - \sum_{i=1}^n q^{\ell_n - \ell_i} = q^{\ell_n} \left( 1 - \sum_{i=1}^n q^{-\ell_i} \right) > 0.$$

□

**Proposició 2.2.2 (Teorema de Kraft).** *Si tenim  $\ell_1, \dots, \ell_n$  satisfent la desigualtat següent*

$$\sum_{i=1}^n q^{-\ell_i} \leq 1 \tag{2.2.1}$$

*llavors hi ha un codi  $q$ -ari prefix format per paraules de longituds  $\ell_1, \dots, \ell_n$ .*

*Demostració.* Suposem que  $\ell_1 \leq \ell_2 \leq \dots \leq \ell_n$  i fem la demostració per inducció sobre  $n$ .

Per  $n = 1$  prenem qualsevol paraula de longitud  $\ell_1$ . Si  $n > 1$  com que  $\sum_{i=1}^{n-1} q^{-\ell_i} < \sum_{i=1}^n q^{-\ell_i} \leq 1$ , per hipòtesi d’inducció hi ha un codi prefix  $\mathcal{C} =$

$\{c_1, \dots, c_{n-1}\}$  amb longituds  $\ell_1, \dots, \ell_{n-1}$ . Per 2.2.1 el podem ampliar amb una paraula de longitud  $\ell_{n-1}$ . Com que  $\ell_n \geq \ell_{n-1}$  aquesta nova paraula la podem allargar a una de longitud  $\ell_n$ .  $\square$

La demostració de 2.2.2 ens proporciona un algorisme per construir el codi. Primer ordenem les longituds  $\ell_1 \leq \ell_2 \leq \dots \leq \ell_n$  i prenem una paraula de longitud  $\ell_1$  qualsevol. A continuació anem afegint successivament paraules de longituds  $\ell_2, \dots, \ell_n$ . La única restricció és que en cada pas hem d'afegir una paraula que no tingui com a prefix cap de les anteriors.

Veiem-ho en un exemple. A l'alfabet binari  $\{0, 1\}$  hi ha un codi de longituds 1, 2, 3, 4, 4, ja que  $1/2 + 1/4 + 1/8 + 1/16 + 1/16 = 1$ . Comencem per la paraula 0, per exemple. Així totes les altres hauran de començar per 1. A continuació afegim 10, per exemple. Les paraules següents hauran de començar per 11. Ara afegim 110. Les que queden hauran de començar per 111 forçosament. Finalment no tenim altra tria que prendre 1110 i 1111 i ja tenim el codi:  $\mathcal{C} = \{0, 10, 110, 1110, 1111\}$ . El fet que a l'últim pas no hi hagi tria possible (hem pres 1111, la única paraula de longitud 4 que no és prefix de cap de les anteriors) ens diu que no es poden afegir paraules al codi, i això passa perquè en realitat tenim una igualtat a (2.2.1). Això passarà sempre per la proposició 2.2.3.

## 2.2.2 El Teorema de McMillan

**Proposició 2.2.3 (Teorema de McMillan).** *Si  $\mathcal{C} = \{c_1, \dots, c_n\}$  és un codi  $q$ -ari i  $\ell_1, \dots, \ell_n$  són les longituds de las paraules de  $\mathcal{C}$  llavors:*

$$\sum_{i=1}^n q^{-\ell_i} \leq 1$$

*Demostració.* Si notem per  $l$  el màxim dels  $\ell_i$ ,

$$\left( \sum_{i=1}^n q^{-\ell_i} \right)^k = \sum_{i_1=1}^n \dots \sum_{i_k=1}^n q^{-(\ell_{i_1} + \dots + \ell_{i_k})} = \sum_{r=1}^{kl} \frac{a_r}{q^r},$$

on  $a_r$  és el nombre de vegades que podem obtenir  $r$  com a suma  $\ell_{i_1} + \dots + \ell_{i_k}$  variant  $(i_1, \dots, i_k) \in \{1, \dots, n\}^k$ . Com que  $\mathcal{C}$  és un codi, resulta que  $a_r$  és igual al nombre de paraules de longitud  $r$  de l'alfabet que es poden obtenir concatenant  $k$  paraules de  $\mathcal{C}$ . Així resulta que  $a_r$  és menor o igual al nombre de paraules de longitud  $r$  de l'alfabet, i.e.,  $a_r \leq q^r$ . Així

$$\left( \sum_{i=1}^n q^{-\ell_i} \right)^k \leq \sum_{r=1}^{kl} \frac{q^r}{q^r} = kl.$$

Traient arrels

$$\sum_{i=1}^n q^{-\ell_i} \leq (kl)^{1/k}$$

i prenent límit per  $k \rightarrow \infty$  arribem a

$$\sum_{i=1}^n q^{-\ell_i} \leq 1.$$

□

**Corol·lari 2.2.4.** *Donat un codi qualsevol sempre hi ha un altre codi sobre el mateix alfabet amb paraules de les mateixes longituds i a més prefix.*

*Demostració.* Per 2.2.2 i 2.2.3. Observem que 2.2.3 s'aplica a un codi qualsevol i a 2.2.2 es construeix un codi prefix. □

Com que a l'hora de comprimir el que realment importa d'un codi són les longituds de les paraules, a la pràctica només treballarem amb codis prefixos.

El fet que en la desigualtat de Kraft hi hagi igualtat té molt a veure amb el fet que cada paraula de  $A^*$  es pugui descomposar com a concatenació de paraules de  $\mathcal{C}$ . De fet no podem esperar això exactament (per exemple, en el codi (2.1.2) les paraules de longitud senar no pertanyen a  $\mathcal{C}^*$ ) sinó que tota paraula de  $A^*$  sigui prefix d'una de  $\mathcal{C}^*$ . A l'exemple (2.1.2) si afegim un bit a una paraula de longitud senar obtenim una paraula de  $\mathcal{C}^*$ . Observem que per aquest codi se satisfà la igualtat:  $2^{-2} + 2^{-2} + 2^{-2} + 2^{-2} = 1$ . En canvi, en el codi (2.1.3) tenim  $2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} = 15/16$  i la paraula 1111 no és prefix de cap paraula de  $\mathcal{C}^*$ .

**Proposició 2.2.5.** *Si  $\mathcal{C}$  és un codi prefix, són equivalents:*

1.  $\mathcal{C}$  satisfà la igualtat de Kraft
2.  $\mathcal{C}$  no es pot estendre
3. tota paraula de  $A^*$  és prefix d'una paraula de  $\mathcal{C}^*$ .

*Demostració.* (1)→(2) és clar perquè si tenim igualtat en Kraft i afegim una paraula, llavors el sumatori és més gran que 1.

(2)→(3). Ho fem per reducció a l'absurd. Suposem que hi ha alguna paraula de  $A^*$  que no és prefix de cap paraula de  $\mathcal{C}^*$ . Sigui  $x$  una tal paraula de longitud mínima. Com que  $x$  no és prefix de cap paraula de  $\mathcal{C}$  i  $\mathcal{C} \cup \{x\}$  no és prefix, llavors alguna paraula de  $\mathcal{C}$ , posem  $c$ , és prefix de  $x$ , i.e.  $x = c/y$ . Com que  $y$  és de longitud més curta que  $x$ ,  $y$  ha de ser prefix d'una paraula  $w$  de  $\mathcal{C}^*$ , i.e.  $w = y/z$ . Llavors  $x/z = c/y/z = c/w \in \mathcal{C}^*$ , contradicció.

(3)→(1). Si tinguéssim desigualtat estricta en Kraft, per 2.2.1 podem ampliar el codi afegint-li una paraula  $c$ . Llavors  $c$  no és prefix de  $\mathcal{C}^*$ . □



## Capítol 3

# Compressió estadística: Huffman

### 3.1 Fonts sense memòria

Una *font sense memòria* és una distribució de probabilitat sobre un alfabet. Es pot pensar també com una variable aleatòria que pren valors en un alfabet. La idea és pensar que tenim un mecanisme que va emetent les lletres de manera aleatòria e independentment segons la seva probabilitat. Suposarem que a la distribució totes les probabilitats són  $> 0$ , altrament suprimim de la font les lletres amb probabilitat nul·la, ja que no seran mai emeses per la font. Quan engeguem aquest mecanisme ens produeix paraules aleatòriament. Si  $B = \{b_1, \dots, b_n\}$  és l'alfabet i  $p_1, \dots, p_n$  és la distribució de probabilitats, la probabilitat que la font emeti la paraula  $b_{i_1}/\dots/b_{i_r}$  és  $p_{i_1} \dots p_{i_r}$ .

Aquestes paraules les haurem de codificar mitjançant un esquema de codificació en un alfabet  $A$  (normalment el binari) i enviar-les o emmagatzemar-les. L'objectiu de la compressió és que les paraules un cop codificades siguin al més curtes possible. La idea serà codificar lletres amb probabilitat alta mitjançant paraules curtes i a l'inrevés, tot això amb una mica de gràcia. Si l'esquema de codificació és  $b_i \mapsto c_i \in A^*$  el codi resultant és  $\mathcal{C} = \{c_1, \dots, c_n\}$  i la longitud mitja d'una lletra codificada és

$$\tilde{\ell}(\mathcal{C}) = \sum_{i=1}^n p_i \ell(c_i).$$

A  $\tilde{\ell}(\mathcal{C})$  l'hi direm longitud mitja del codi. Això és una mica imprecís, doncs la longitud mitja no depèn només del codi sinó de l'esquema de codificació. De fet, en tota aquesta secció parlarem de codis quan en realitat hauríem de dir esquemes de codificació.

## 3.2 Codis de Huffman

Quan comprimim voldrem doncs que la longitud mitja del codi sigui la mínima possible. Per 2.2.4 ens podem restringir a codis prefixos.

**Definició 3.2.1.** Un codi de Huffman d'una font en un alfabet  $A$  és un codi prefix de  $A$  que fa mínima la longitud mitja.

A continuació veurem que hi ha codis de Huffman, i el que és més important, donarem un algorisme per calcular-los.

**Lema 3.2.2.** Per a qualsevol font  $S$  i alfabet  $A$ , hi ha codis de Huffman.

*Demostració.* Sigui  $L$  la longitud mitja d'un codi qualsevol. N'hi ha prou amb veure que hi ha un nombre finit de codis amb longitud mitja menor o igual que  $L$ . Si  $\ell(\mathcal{C}) = \sum_{i=1}^n p_i \ell(c_i) \leq L$  llavors  $p_i \ell(c_i) \leq L$  i per tant  $\ell(c_i) \leq L/p_i$ . Com que la longitud de cada paraula del codi està fitada, només n'hi pot haver un nombre finit.  $\square$

**Lema 3.2.3.** Si  $\{c_1, \dots, c_n\}$  és un codi de Huffman d'una font amb probabilitats  $p_1, \dots, p_n$  i  $p_j > p_i$  llavors  $\ell(c_j) \leq \ell(c_i)$ .

*Demostració.* Considerem el codi  $\mathcal{C}'$  obtingut a partir de  $\mathcal{C}$  intercanviant les paraules  $c_i$  i  $c_j$ . Com que  $\mathcal{C}$  és Huffman  $0 \leq \tilde{\ell}(\mathcal{C}') - \tilde{\ell}(\mathcal{C}) = p_i \ell(c_j) + p_j \ell(c_i) - p_i \ell(c_i) - p_j \ell(c_j) = (p_j - p_i)(\ell(c_i) - \ell(c_j))$ . De  $p_j - p_i > 0$  deduïm  $\ell(c_i) - \ell(c_j) \geq 0$ .  $\square$

**Lema 3.2.4.** Si  $\mathcal{C}$  és un codi de Huffman i  $c \in \mathcal{C}$  és una paraula de longitud màxima llavors hi ha una altra paraula  $d \in \mathcal{C}$  de longitud màxima que només difereix de  $c$  en la última lletra.

*Demostració.* Notem per  $c^-$  la paraula que s'obté esborrant la última lletra de  $c$ , i considerem  $\mathcal{C}'$  obtingut a partir de  $\mathcal{C}$  reemplaçant  $c$  per  $c^-$  i deixant la resta igual. Com que  $\tilde{\ell}(\mathcal{C}') = \tilde{\ell}(\mathcal{C}) - p_n < \tilde{\ell}(\mathcal{C})$ ,  $\mathcal{C}'$  no pot ser prefix i per tant alguna paraula de  $\mathcal{C}'$  és prefix d'una altre. Com que  $\mathcal{C}$  és prefix la única possibilitat és que  $c^-$  sigui prefix d'alguna altra paraula  $d$  de  $\mathcal{C}'$  (i per tant de  $\mathcal{C}$ ). Tampoc pot ser  $c^- = d$  doncs llavors  $d$  seria prefix de  $c$ . Així  $\ell(d) \geq \ell(c^-) + 1 = \ell(c)$ . Com que  $c$  era de longitud màxima,  $\ell(d) \leq \ell(c)$  i per tant  $c$  i  $d$  són de la mateixa longitud i només difereixen en la última lletra.  $\square$

El teorema següent ens proporciona un algorisme recursiu per construir codis de Huffman en el cas binari.

**Teorema 3.2.5.** Suposem que tenim una font  $S$  amb distribució de probabilitat  $p_1, \dots, p_n$  i  $p_1 \geq \dots \geq p_{n-1} \geq p_n$ . Si  $\{c_1, \dots, c_{n-1}\}$  és un codi de Huffman binari d'una font amb probabilitats  $p_1, \dots, p_{n-2}, p_{n-1} + p_n$  llavors

$$\{c_1, \dots, c_{n-2}, c_{n-1}/0, c_{n-1}/1\}$$

és un codi de Huffman binari de  $S$ .

*Demostració.* És fàcil veure que  $\mathcal{C} = \{c_1, \dots, c_{n-2}, c_{n-1}/0, c_{n-1}/1\}$  és prefix perquè  $\mathcal{C}' = \{c_1, \dots, c_{n-1}\}$  ho és. Observem també que

$$\tilde{\ell}(\mathcal{C}) = \tilde{\ell}(\mathcal{C}') + p_{n-1} + p_n. \quad (3.2.1)$$

Per 3.2.2 prenem  $\mathcal{C}_1$  un codi de Huffman binari per  $S$ . Per 3.2.3 i 3.2.4,  $\mathcal{C}_1$  és de la forma  $\{d_1, \dots, d_{n-2}, d_{n-1}/0, d_{n-1}/1\}$ , on  $d_{n-1}/0$  i  $d_{n-1}/1$  són les paraules de longitud màxima que corresponen a  $p_{n-1}$  i  $p_n$ . Ara veiem que  $\mathcal{C}'_1 = \{d_1, \dots, d_{n-2}, d_{n-1}\}$  és un codi prefix. Com que  $\mathcal{C}_1$  és prefix tot el que hem de verificar és que  $d_{n-1}$  no és prefix de cap  $d_i$  amb  $i < n - 1$ . Això no pot ser perquè si  $d_{n-1} = d_i$  llavors  $d_i$  seria prefix de  $d_{n-1}/0$  i si  $d_{n-1}$  és un prefix de  $d_i$  de menor longitud, llavors  $d_i$  hauria de ser o bé  $d_{n-1}/0$  o bé  $d_{n-1}/1$ , perquè aquestes són de longitud màxima. També

$$\tilde{\ell}(\mathcal{C}_1) = \tilde{\ell}(\mathcal{C}'_1) + p_{n-1} + p_n. \quad (3.2.2)$$

Per veure que  $\mathcal{C}$  és de Huffman ho fem per reducció a l'absurd. Si  $\mathcal{C}$  no fos de Huffman,  $\tilde{\ell}(\mathcal{C}_1) < \tilde{\ell}(\mathcal{C})$  i per (3.2.1) i (3.2.2) obtindríem  $\tilde{\ell}(\mathcal{C}'_1) < \tilde{\ell}(\mathcal{C}')$ , absurd perquè  $\mathcal{C}'$  és de Huffman.  $\square$

L'algorisme es fa en dues etapes. A la primera anem reduint la mida de la font sumant les dues probabilitats més baixes fins que arribem a una font de 2 lletres. Un codi de Huffman per una font de dues lletres és  $\{0, 1\}$  i anem construint els successius codis de Huffman segons el teorema anterior fins que arribem a la font original. Veiem-ne un exemple.

Considerem l'alfabet  $\{a, b, c, d, e\}$  amb probabilitats  $\{0.4, 0.3, 0.1, 0.1, 0.1\}$ . A la primera etapa anem construint les noves distribucions de probabilitat. Les escrivim en columna i posem en negreta la suma dels valors de la columna anterior. A la segona etapa construïm el codi començant per la dreta:

$a$	0.4	0.4	0.4	<b>0.6</b>	$a$	1	1	1	<b>0</b>
$b$	0.3	0.3	0.3	0.4	$b$	00	00	00	1
$c$	0.1	<b>0.2</b>	<b>0.3</b>		$c$	011	<b>010</b>	<b>01</b>	
$d$	0.1	0.1			$d$	0100	011		
$e$	0.1				$e$	0101			

Si el codi no és binari, es fa de manera anàloga amb una petita variació. En el cas binari com que en cada pas sumem dues probabilitats la font s'ens redueix en 1 caràcter. D'aquesta manera sempre podem acabar amb una font de dos caràcters. Si l'alfabet amb el volem codificar té  $q$  lletres, en cada pas sumarem  $q$  probabilitats i per tant en reduïm  $q-1$ . Pot ser que d'aquesta manera no arribem a  $q$  caràcters exactament a l'últim pas. A l'exemple anterior, si volem codificar amb l'alfabet  $\{0, 1, 2\}$  la mida de les fonts va sent 6, 4, 2 en cada pas. El que es fa és ajuntar-ne 2 en lloc de tres al començar, i a continuació ajuntem de tres en tres fins arribar a 3. La mida de les fonts serà 6, 5, 3. Si la font té  $n$  caràcters i volem codificar amb un alfabet de  $q$  lletres, per 'quadrar' necessitaríem que  $n \equiv q \pmod{q-1}$ . Si en el primer pas sumem  $r$  probabilitat, amb  $0 < r \leq q$ ,

per que a l'últim pas en quedin  $q$  exactament, caldrà que  $n - (r - 1) \equiv q \pmod{q - 1}$ , és a dir,  $r \equiv n - (q - 1) \equiv n \pmod{q - 1}$ .

Resumint, en el primer pas hem de sumar  $r$  probabilitats, on  $r$  és l'únic enter que satisfà:

$$0 < r \leq q, \quad r \equiv n \pmod{q - 1}.$$

### 3.2.1 Longitud mitja i entropia

La proposició següent ens diu, entre altres, que l'entropia és una cota inferior de la longitud mitja. El primer Teorema de Shannon ens dirà que podem codificar una font de manera que la longitud mitja s'apropi a l'entropia tant com vulguem.

Quan codifiquem en un alfabet  $q$ -ari (amb  $q$  lletres) hem de prendre, logaritmes en base  $q$  al calcular l'entropia. Farem servir la notació següent:

$$H_q(p_1, \dots, p_n) = - \sum_{i=1}^n p_i \log_q p_i.$$

Si  $S$  és una font amb distribució  $p_1, \dots, p_n$ ,  $H_q(S) = H_q(p_1, \dots, p_n)$ .

**Proposició 3.2.6.** *Si  $\mathcal{C}$  és un codi  $q$ -ari de Huffman de la font  $S$  llavors:*

$$H_q(S) \leq \tilde{\ell}(\mathcal{C}) < H_q(S) + 1. \quad (3.2.3)$$

*Demostració.* Si  $p_1, \dots, p_n$  és la distribució de la font i  $\ell_1, \dots, \ell_n$  són les longituds de les paraules de  $\mathcal{C}$  llavors, per 2.2.3, es satisfà la desigualtat de Kraft:  $\sum_{i=1}^n q^{-\ell_i} \leq 1$ . Aplicant el lema de Gibbs 1.3.1

$$H_q(S) = - \sum_{i=1}^n p_i \log_q p_i \leq - \sum_{i=1}^n p_i \log_q q^{-\ell_i} = \sum_{i=1}^n p_i \ell_i = \tilde{\ell}(\mathcal{C}).$$

Si prenem  $r_i := \lceil -\log_q(p_i) \rceil$  llavors  $-\log_q(p_i) \leq r_i < -\log_q(p_i) + 1$ . Així  $q^{-r_i} \leq p_i$  i per tant  $\sum_{i=1}^n q^{-r_i} \leq \sum_{i=1}^n p_i = 1$ . Prenem, per 2.2.2, un codi  $\mathcal{C}'$  amb longituds  $r_1, \dots, r_n$ . Com que  $\mathcal{C}$  és de Huffman

$$\tilde{\ell}(\mathcal{C}) \leq \tilde{\ell}(\mathcal{C}') = \sum_{i=1}^n p_i r_i < \sum_{i=1}^n p_i (-\log_q(p_i) + 1) = H_q(S) + \sum_{i=1}^n p_i = H_q(S) + 1.$$

□

La demostració d'aquesta proposició ens proporciona un altre mètode per a construir un codi satisfent (3.2.3): apliquem l'algorisme de la proposició 2.2.2 a les longituds següents:

$$\lceil -\log_q(p_1) \rceil, \dots, \lceil -\log_q(p_n) \rceil,$$

que satisfaran la desigualtat de Kraft. Per exemple, si la distribució és

$$S := \{.3, .2, .1, .1, .05, .05, .05, .05, .05, .025, .025\}$$

les longituds que surten són  $\{2, 3, 4, 4, 5, 5, 5, 5, 6, 6\}$  i si calculem la longitud mitja d'aquest codi és 3.550. Si calculem l'entropia d'aquesta distribució dóna  $H(S) = 2.996439345$ . Si calculem un codi de Huffman per a aquesta distribució obtenim 10, 110, 000, 001, 0100, 0101, 0110, 0111, 1110, 11110, 11111 i la seva longitud mitja és 3.050.

### 3.3 El primer Teorema de Shannon

#### 3.3.1 Extensions d'una font

L'estratègia per tal d'apropar la longitud mitja del codi a l'entropia consisteix en considerar blocs de  $k$  lletres de l'alfabet de la font i codificar aquests blocs amb un codi de Huffman. Si  $B = \{b_1, \dots, b_n\}$  és l'alfabet i  $p_1, \dots, p_n$  és la distribució de probabilitats, la probabilitat que la font emeti la paraula  $b_{i_1} / \dots / b_{i_r}$  és  $p_{i_1} \dots p_{i_r}$ . S'anomena extensió  $k$ -èsima de la font  $S$  a la font  $S^k$  a la font que té per alfabet  $B^k$  (les paraules de longitud  $k$  de  $B$ ) i per distribució l'esmentada anteriorment.

**Lema 3.3.1.**

$$H_q(S^k) = kH_q(S).$$

*Demostració.*

$$\begin{aligned} H_q(S^k) &= - \sum_{i_1, \dots, i_k} p_{i_1} \dots p_{i_k} \log_q(p_{i_1} \dots p_{i_k}) = \\ &= - \sum_{i_1, \dots, i_k} p_{i_1} \dots p_{i_k} (\log_q(p_{i_1}) + \dots + \log_q(p_{i_k})) = \\ &= - \sum_{i_1, \dots, i_k} p_{i_1} \dots p_{i_k} \log_q(p_{i_1}) - \dots - \sum_{i_1, \dots, i_k} p_{i_1} \dots p_{i_k} \log_q(p_{i_k}) = \\ &= -k \sum_{i_1, \dots, i_k} p_{i_1} \dots p_{i_k} \log_q(p_{i_k}) = k \sum_{i_1, \dots, i_{k-1}} p_{i_1} \dots p_{i_{k-1}} \left( - \sum_{i_k=1}^n p_{i_k} \log_q(p_{i_k}) \right) = \\ &= kH_q(S) \sum_{i_1, \dots, i_{k-1}} p_{i_1} \dots p_{i_{k-1}} = kH_q(S). \end{aligned}$$

□

#### 3.3.2 El primer teorema de Shannon

El primer teorema de Shannon, conegut com a teorema de codificació sense soroll (o de codificació de font) diu que codificant adequadament la font podem apropar el nombre de bits per símbol a l'entropia tant com vulguem. L'estratègia consisteix a codificar blocs de lletres suficientment llargs.

**Teorema 3.3.2 (Primer Teorema de Shannon).** Si  $\mathcal{C}_k$  és un codi de Huffman per a l'extensió  $k$ -èsima d'una font  $S$ , llavors:

$$\lim_{k \rightarrow \infty} \frac{\tilde{\ell}(\mathcal{C}_k)}{k} = H_q(S).$$

*Demostració.* Per 3.2.6 i 3.3.1

$$kH_q(S) = H_q(S^k) \leq \tilde{\ell}(\mathcal{C}_k) < H_q(S^k) + 1 = kH_q(S) + 1,$$

i dividint per  $k$

$$H_q(S) \leq \frac{\tilde{\ell}(\mathcal{C}_k)}{k} < H_q(S) + \frac{1}{k}.$$

□

Veiem-ne un exemple.

*Exemple 3.3.3.* Si la font  $S$  emet  $\{a, b\}$  amb probabilitats  $\{.9, .1\}$  respectivament,  $H(S) = 0.4689955936$ . Codificant  $S$  directament obtindrem una longitud mitja de 1 bit per símbol. Un codi de Huffman per  $S^2 = aa, ab, ba, bb$  és  $\{0, 110, 11, 111\}$ . Com que la seva longitud mitja és de 1.29 bits per símbol (de  $S^2$ ) necessitem  $\frac{1.29}{2} = 0.645$  bits per símbol de  $S$ . Si treballem amb  $S^3$ , un codi de Huffman seria  $\{0, 100, 111, 10100, 10110, 10111, 1010, 10101\}$  que té longitud mitja 1.598 i per tant necessitaríem  $\frac{1.598}{3} = 0.5326666667$  bits per símbol de  $S$ . El nombre de bits per símbol de les diferents extensions de  $S$  queda reflectit a la taula següent:

$S^i$	$S$	$S^2$	$S^3$	$S^4$	$S^5$	$S^6$
bits/símbol de $S^i$	1	1.290	1.598	1.970	2.401	2.821
bits/símbol de $S$	1	.6450	.5327	.4926	.4802	.4702

  

$S^7$	$S^8$	$S^9$	$S^{10}$	$S^{11}$	$S^{12}$	$S^{13}$	$S^{14}$	$S^{15}$
3.320	3.806	4.275	4.767	5.206	5.640	6.128	6.607	7.084
.4743	.4758	.4750	.4767	.4733	.4700	.4714	.4719	.4723

## Capítol 4

# Mètodes de diccionari: LZ77 i LZ78

En general els mètodes estadístics (incloent Huffman amb memòria i compressió aritmètica) assoleixen bones taxes de compressió, però a costa de ser mes costosos en recursos (temps i memòria). Generalment requereixen un estudi estadístic previ a la compressió. Els mètodes de diccionari, encara que aconseguen taxes de compressió en general no tan bones solen ser menys costosos en recursos. Probablement és això el que els ha fet tant populars. No requereixen estudi previ i són mecanismes de compressió ‘on the fly’. La idea es anar generant sobre la marxa un diccionari on anar emmagatzemant repeticions de paraules i reemplaçar una paraula per un ‘pointer’ al diccionari. Hi ha una bona colla de variants, però es poden reunir en dues famílies bàsiques: els denominats LZ77 i LZ78. De cada família hi ha una bona colla d’implementacions. Per exemple, els compressors LHarc, PKzip, GNUzip, Info-ZIP i el format PNG (successor del GIF) usen algorismes que pertanyen a la família LZ77.

### 4.1 LZ77

En aquest mètode, conegut també com a ‘finestra corredora’ hi ha dues finestres, d’amplada fixa que van corrent sobre el text. A la finestra de la dreta es busca el string més llarg que també estigui a la finestra de l’esquerra. La finestra de l’esquerra és més llarga que la de la dreta. Si el string és prou llarg (dos caràcters normalment, a vegades tres) es reemplaça per el parell  $(d, l)$ , on  $d$  indica el nombre de llocs que ens hem de desplaçar a l’esquerra per trobar el string i  $l$  indica la llargada del string. La finestra està inicialment situada a l’origen i es va desplaçant una o  $l$  ( si hi ha reemplaçament) posicions a la dret en cada pas.

Per exemple, si estem intentant comprimir

El tinter tenia tanta tinta que la tonta es va tacar

l'algorisme ens anirà produint:

El tinter(7,2)enia(6,2)a(13,2)(6,3)(19,3)(6,2)que l(13,3)o(13,4)es v(31,4)car

De fet no és exactament això el resultat de la compressió. Normalment es treballa amb blocs de 8 bits (1 byte), és a dir un alfabet de 256 caràcters. Per tal de distingir entre un parell i un caràcter s'afegeix un bit. Si fixem els valors de les finestres en  $2^5$  i  $2^2$  respectivament (el valor òptim per a l'exemple anterior) i representem els parell  $(d, l)$  escrivint  $d-1$  i a continuació  $l-1$  en binari necessitarem  $1+5+2$  bits per a cada parell i  $1+8$  bits per a cada literal. Així, els parells comprimeixen mentre que els literals fan el contrari. El que s'espera és que hi hagi suficients repeticions i el resultat sigui de compressió. A l'exemple anterior, la versió comprimida ocupa el 73.6% de la versió no comprimida. Si treballem amb valors alts per als paràmetres de les finestres obtindrem més parells, però també necessitarem més bits per emmagatzemar-los. Cal trobar un compromís, i els valors típics per a la finestra de l'esquerra estan entre  $2^{12}$  i  $2^{15}$  i entre  $2^4$  i  $2^5$  per a la finestra de la dreta.

Les característiques essencials del LZ77 són:

- Factor de compressió relativament elevat ,al menys si es compara amb altre mètodes de diccionari.
- Compressió relativament lenta a menys que s'implementin mètodes per accelerar la cerca de 'matching'.
- Descompressió molt ràpida.

A la pràctica es fan modificacions destinades o bé a augmentar la velocitat de compressió (a costa de disminuir-ne el factor de compressió) o petites modificacions que augmenten el factor de compressió.

Citem-ne algunes d'elles:

- Ús de funcions o cadenes de hash per tal d'accelerar la cerca durant la compressió.
- A vegades la cerca del 'matching' més llarg no fa assolir la millor compressió. Com que la cerca de la millor combinació entre parells ordenats i literals és intractable el que es fa és implementar una 'lazy evaluation'.
- Com que les longituds més curtes en els parells són més freqüents, es pot millorar la compressió en un segon estadi utilitzant codis de Huffman per les longituds. Això també es pot fer tant per al desplaçament com per els caràcter.

A continuació descriurem breument el GNU zip o *gzip* una implementació concreta del LZ77.

### 4.1.1 GNU zip

El *gzip* reserva 15 i 8 bits per al desplaçament i llargada respectivament. Això vol dir que un parell necessita  $1 + 15 + 8 = 24$  bits i un caràcter  $1 + 8 = 9$  bits, així que la cerca del 'matching' només surt rentable a partir de 3 caràcters. La llargada màxima serà de  $2^8 + 2 = 258$  caràcters. Manté una cadena de hash per a la cerca de 'matching' llargs. Fa servir 'Lazy evaluation': després de buscar el 'matching' més llarg corresponent a una posició busca si hi ha un 'matching' més llarg per a la posició següent. Si el troba envia el caràcter corresponent com a literal i continua.

A més dona 10 opcions (durant la compressió) que afecten la cerca a través de la cadena de hash i la 'lazy evaluation'. De fet cada una d'aquestes opcions determina un valor dels següents paràmetres:

***good\_length*** Si el 'match' actual té llargada superior llavors la profunditat màxima per la cerca es redueix (divideix *max\_chain* per 2)

***max\_lazy*** No executa 'lazy evaluation' més enllà d'aquesta longitud

***nice\_length*** Acaba la cerca més enllà d'aquesta llargada

***max\_chain*** Límit de la profunditat de la cadena de cerca hash

## 4.2 LZ78

LZ78, a diferència del LZ77 manté un diccionari més estructurat i que va creixent gradualment al llarg de la compressió. Això fa la compressió més àgil però a costa de disminuir la velocitat de descompressió. A la pràctica el diccionari té un límit de mida i presenta un problema quan l'assolim.

Comença amb el diccionari buit i a cada pas busca en el lookahead la frase més llarg que hi ha en el diccionari i retorna el parell  $(n, c)$ , on  $n$  és el 'pointer' per aquesta frase i  $c$  és el caràcter que va a continuació. Al diccionari també s'afegeix la frase representada per  $n$  seguida de  $c$ . De fet el diccionari té forma de arbre.

Al codificar

El tinter tenia tanta tinta que la tonta es va tacar

produïm

El tin(4,e)r(3,t)(6,i)a(9,a)(6,t)(12, )(4,i)(14,a)(2,q)u(7, )l(10,t)o(10, )(2,s)  
v(11,a)car



# Capítol 5

## Codis de Bloc

Recordem que un codi de bloc és  $\mathcal{C}$  un codi de longitud fixa. Si  $n$  és la longitud i  $A$  l'alfabet llavors  $\mathcal{C} \subseteq A^n$ . Aquí estudiarem els codis de bloc des del punt de vista de la correcció i detecció d'errors.

### 5.1 Distància

En general quan es codifica un missatge  $m$  es reemplaça per paraules de  $\mathcal{C}$  i s'envia a través d'un canal. El que es desitja és poder detectar i corregir el màxim nombre d'errors que es puguin produir durant la transmissió. Per això és fonamentals que les paraules de  $\mathcal{C}$  estiguin prou 'allunyades' les unes de les altres. Començarem introduint el concepte de distància entre paraules. Una paraula  $\mathbf{x}$  de longitud  $n$  l'escriurem indistintament com  $\mathbf{x} = x_1 \cdots x_n$  o bé  $\mathbf{x} = (x_1, \dots, x_n)$ .

**Definició 5.1.1.** Si  $\mathbf{x}, \mathbf{y} \in A^n$  la distància entre  $\mathbf{x}$  i  $\mathbf{y}$  és el nombre de posicions en les que són diferents:

$$d(\mathbf{x}, \mathbf{y}) = |\{1 \leq i \leq n \mid x_i \neq y_i\}|.$$

Per exemple  $d(11001101, 10101010) = 7$  i  $d(\text{ric}, \text{ruc}) = 1$ . La distància satisfà les propietats d'una mètrica:

**Proposició 5.1.2.** 1.  $d(\mathbf{x}, \mathbf{y}) = 0$  si i només si  $\mathbf{x} = \mathbf{y}$ .

2.  $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ .

3.  $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$

*Demostració.* (1) i (2) són evidents. Per demostrar (3) només cal observar que si  $x_i \neq z_i$  llavors o bé  $x_i \neq y_i$  o bé  $y_i \neq z_i$ .  $\square$

En general un codi tindrà més capacitat detectora i correctora com més lluny estiguin les paraules les unes de les altres. Això ho mesura precisament la distància mínima.

**Definició 5.1.3.** Si  $\mathcal{C} \subseteq A^n$

$$\delta(\mathcal{C}) := \min \{d(\mathbf{u}, \mathbf{v}) \mid \mathbf{u}, \mathbf{v} \in \mathcal{C}, \mathbf{u} \neq \mathbf{v}\}$$

## 5.2 Paràmetres d'un codi

Els paràmetres fonamentals d'un codi són

- La longitud, que denotarem per  $n$ .
- La mida, que denotarem per  $M := |\mathcal{C}|$ .
- La distància mínima  $\delta = \delta(\mathcal{C})$ .
- La mida de l'alfabet, que denotarem per  $q := |A|$ .

Quan aquests són els paràmetres d'un codi  $\mathcal{C}$  ho expressarem dient que  $\mathcal{C}$  és un  $(n, M, \delta)_q$ -codi. Com és habitual quan  $q = 2$  el subíndex no es posa. Altres paràmetres importants són

- La *dimensió*  $k := \log_q M$ .
- La *taxa de transmissió*  $k/n$ .
- La *taxa de redundància*  $1 - k/n$ .

La taxa de transmissió mesura quina part de la informació que es transmet a través del canal correspon al missatge original i la taxa de redundància mesura la part que correspon a la redundància que l'hi hem afegit.

En general buscarem que un codi tingui una taxa de transmissió al més alta possible i una distància mínima prou alta per tal de poder assolir una determinada capacitat correctora o detectora.

*Exemple 5.2.1.* 1. El codi trivial (consta d'una sola paraula) és un  $(n, 1, 0)_q$ -codi. Té taxa de transmissió 0.

2. El codi total  $\mathcal{C} = A^n$  és un  $(n, q^n, 1)_q$ -codi. Té taxa de transmissió 1.

3. El codi de repetició  $\mathcal{R}_q(n) = \{a \cdots a \mid a \in A\}$  és un  $(n, q, n)_q$ -codi. té taxa de transmissió  $1/n$ .

4. El codi binari que consisteix en les paraules 00000000, 11111111, 10101010, 11010000 i 11100100 és un  $(8, 5, 3)$ -codi i la seva taxa de transmissió és  $\frac{\log(5)}{8} = 0.2902410118$ .

5. El codi binari que consisteix en les paraules del codi anterior i totes les seves permutacions cícliques és un  $(8, 20, 3)$ -codi i la seva taxa de transmissió és  $\frac{\log(20)}{8} = 0.5402410119$ .

El primer codi és inútil perquè no pot transmetre informació i el segon no serveix als nostres propòsits perquè no permet detectar ni corregir res (enviem

la informació tal qual). El tercer té una distància mínima molt alta, però una taxa de transmissió molt baixa quan  $n$  és gran. El quart i el cinquè tenen la mateixa capacitat detectora i correctora (detecten 2 errors i en corregeixen 1), però (5) és millor perquè té taxa de transmissió més alta.

### 5.3 Detecció i correcció d'errors

Com que pel canal només s'envien paraules del codi, l'estratègia per detectar errors consisteix a mirar si la paraula rebuda  $\mathbf{y}$  és del codi. Si  $\mathbf{y}$  és del codi s'accepta i si no es rebutja. Depenent del canal es pot demanar retransmissió o es marca com a errònia tota la paraula (esborrall). La *capacitat detectora* d'un codi és el màxim nombre d'errors (a qualsevol posició) que l'estratègia esmentada detecta.

A l'exemple 5.2.1 (4) si l'emissor ha enviat  $\mathbf{x} = 11010000$  i s'han produït errors a la primera i segona posició, el receptor rep  $\mathbf{y} = 00010000$ . Com que  $\mathbf{y}$  no és del codi, el receptor pot detectar l'error. Si es produeixen errors a les posicions primer, segona i quarta llavors  $\mathbf{y} = 00000000$ , el receptor decideix que no hi ha error quan s'han produït tres errors. En aquest cas es veu que el màxim nombre d'errors que es detecten és 2.

**Proposició 5.3.1.** *Un codi detecta fins a un màxim de  $\delta - 1$  errors.*

*Demostració.* Si s'ha rebut  $\mathbf{y}$  al enviar  $\mathbf{x} \in \mathcal{C}$  i s'ha produït  $< \delta$  errors llavors  $\mathbf{y} \notin \mathcal{C}$  i l'estratègia detecta l'error. En general no es detecten errors en  $\delta$  posicions: si  $\mathbf{x}$  i  $\mathbf{y}$  són dues paraules del codi que estan a distància  $\delta$ , pot passar que s'envii  $\mathbf{x}$  i es rebí  $\mathbf{y}$ . En aquest cas s'han produït  $\delta$  errors i l'estratègia no els ha detectat.  $\square$

Pel que fa a la correcció l'estratègia usual és la següent: si s'ha rebut  $\mathbf{y}$ , es busca una paraula  $\mathbf{u}$  del codi que estigui a distància mínima de  $\mathbf{y}$ ; el descodificador llavors decideix que la paraula enviada era  $\mathbf{u}$ . Això funcionarà correctament només quan  $\mathbf{u} = \mathbf{x}$ . aquesta estratègia de correcció rep el nom de descodificació per mínima distància. La *capacitat correctora* d'un codi és el màxim nombre d'errors (a qualsevol posició) que l'estratègia esmentada corregeix correctament.

A l'exemple 5.2.1 (4) si l'emissor ha enviat  $\mathbf{x} = 11010000$  i s'ha produït un error a la primera posició, el receptor rep  $\mathbf{y} = 01010000$ . Com que la paraula el codi més propera a  $\mathbf{y}$  és  $\mathbf{u} = \mathbf{x}$ , el receptor descodifica  $\mathbf{y}$  com la paraula  $\mathbf{u}$  i corregeix l'error. Si es produeixen dos errors a les posicions primera i segona, llavors el receptor rep  $\mathbf{y} = 00010000$ . Com que la paraula del codi més propera a  $\mathbf{y}$  és  $\mathbf{u} = 00000000$  el receptor descodifica  $\mathbf{y}$  com  $\mathbf{u}$  i no corregeix correctament. En aquest cas es veu que el màxim nombre d'errors que pot corregir és 1.

**Lema 5.3.2.** *Si  $\mathbf{x}, \mathbf{y}$  són dues paraules de la mateixa longitud i  $d(\mathbf{x}, \mathbf{y}) = r + s$  existeix una paraula  $\mathbf{z}$  a distància  $r$  de  $\mathbf{x}$  i distància  $s$  de  $\mathbf{y}$ .*

*Demostració.* Considerem la paraula  $\mathbf{z}$  obtinguda reemplaçant en  $\mathbf{x}$   $r$  de les  $r + s$  posicions en que  $\mathbf{x}$  difereix de  $\mathbf{y}$  per el seu valor en  $\mathbf{y}$ . És clar que  $d(\mathbf{x}, \mathbf{z}) = r$  i  $d(\mathbf{z}, \mathbf{y}) = s$ .  $\square$

**Proposició 5.3.3.** *El descodificador per mínima distància corregeix fins a un màxim de  $\rho := \lfloor \frac{\delta-1}{2} \rfloor$  errors.*

*Demostració.* Si a l'enviar  $\mathbf{x}$  s'han produït com a molt  $\rho$  errors i es rep  $\mathbf{y}$  resulta que  $d(\mathbf{y}, \mathbf{x}) \leq \rho$ , però  $\mathbf{x}$  és la única paraula de  $\mathcal{C}$  que està a distància  $\leq \rho$  de  $\mathbf{y}$ : si  $\mathbf{u}$  en fos una altre llavors  $d(\mathbf{x}, \mathbf{u}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{u}) \leq 2\rho \leq \delta - 1$  i tindriem dues paraules del codi a distància menor que  $\delta$ , impossible.

Veiem ara que l'estratègia pot fallar si es produeixen  $\rho+1$  errors. Si  $\delta$  és senar  $\delta = 2\rho + 1$  i si  $\delta$  és parell  $\delta = 2(\rho + 1)$ . En qualsevol cas posem  $\delta = (\rho + 1) + \mu$  amb  $\mu = \rho$  o  $\mu = \rho + 1$  depenent de si  $\delta$  és senar o parell. Siguin  $\mathbf{x}, \mathbf{u}$  són dues paraules del codi a distància  $\delta$  i  $\mathbf{y}$  una paraula a distància  $\rho + 1$  de  $\mathbf{x}$  i  $\mu$  de  $\mathbf{u}$ , que existeix pel lema previ. Quan s'envii  $\mathbf{x}$ , es rebí  $\mathbf{y}$  (s'han produït  $\rho + 1$  errors) i el descodificador tria  $\mathbf{u}$  (en el cas en que  $\rho$  és parell és una de les tries possibles del descodificador, mentre que en el cas senar aquesta és la única tria possible) no corregeix bé.  $\square$

Els algorismes de descodificació solen ser molt costosos i la dificultat augmenta quan volem corregir més errors. A vegades resulta més ràpid demanar i esperar la retransmissió del missatge que corregir molts errors. Per exemple, si tenim un codi amb distància mínima 5, pot corregir fins a dos errors. Ara bé, pot passar que el nostre descodificador sigui força eficient quan corregeix un error i en canvi poc eficient al corregir-ne dos, de manera que en aquest cas és millor demanar retransmissió. En aquests casos s'adopta una estratègia diferent: es decideix corregir fins a un determinat nombre d'errors  $t$  i en el cas en que s'en produeixin més de  $t$  es demana retransmissió. Més precisament, l'estratègia és la següent: Si es rep  $\mathbf{y}$  el descodificador busca una paraula  $\mathbf{u}$  del codi a distància com a molt  $t$  de  $\mathbf{y}$ . Si la troba, decideix descodificar amb aquesta paraula. Altrament marca la paraula com a esborrall i demana retransmissió. Si  $t \leq r$ , es diu que un codi *detecta  $r$  errors i en corregeix  $t$*  si aquesta estratègia funciona correctament per a totes les retransmissions on s'han produït com a molt  $r$  errors. És a dir, si s'ha enviat  $\mathbf{x}$  i s'han produït com a molt  $r$  errors, el descodificador mai triarà una paraula  $\mathbf{u}$  que no sigui  $\mathbf{x}$ .

**Proposició 5.3.4.** *Un codi detecta  $r$  errors i en corregeix  $t$  sii  $t + r < \delta$ .*

*Demostració.* Suposem que  $s + r < \delta$ . Si al enviar  $\mathbf{x}$  es rep  $\mathbf{y}$  i s'han produït com a molt  $t$  errors llavors  $d(\mathbf{x}, \mathbf{y}) \leq t$ . Com que  $2t \leq t + r < \delta$  llavors  $\mathbf{x}$  és la única paraula del codi a distància com a molt  $t$  de  $\mathbf{y}$  i el descodificador tria  $\mathbf{u} = \mathbf{x}$ . En el cas en que s'ha produït un nombre d'errors entre  $t$  i  $r$  llavors  $t < d(\mathbf{x}, \mathbf{y}) \leq r$ . Hem de veure que en aquest cas no hi ha cap paraula  $\mathbf{u}$  del codi a distància com a molt  $t$  de  $\mathbf{y}$ . Si  $\mathbf{u}$  és del codi i  $d(\mathbf{u}, \mathbf{y}) \leq t$  llavors  $d(\mathbf{x}, \mathbf{u}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{u}) \leq r + t < \delta$ , contradicció.

Veiem ara que quan es produeixen  $r$  errors amb  $t + r = \rho$  a vegades l'estratègia falla. Siguin  $\mathbf{x}, \mathbf{u}$  són dues paraules del codi a distància  $\delta$  i  $\mathbf{y}$  una paraula a distància  $t$  de  $\mathbf{u}$  i distància  $r$  de  $\mathbf{x}$ . Quan s'envia  $\mathbf{x}$  i es rep  $\mathbf{y}$  es produeixen  $r$  errors i el descodificador tria la paraula  $\mathbf{u}$  (de fet  $\mathbf{u}$  és una de les tries possibles, n'hi poden haver més), corregint incorrectament.  $\square$

Per exemple un codi amb distància mínima 7 el podem fer servir de les maneres següents:

- Per a detectar fins a 6 errors.
- Per a corregir 1 error i detectar-ne fins a 5 errors.
- Per a corregir 2 errors i detectar-ne fins a 4 errors.
- Per a corregir 3 errors.

De fet les proposicions 5.3.1 i 5.3.3 són un cas particular de la proposició 5.3.4

En certes ocasions el senyal rebut no és clar i no es pot distingir quina és la lletra que està en una certa posició. Llavors es marca aquesta posició com un esborrall, que indicarem per \*. Si hem rebut  $\mathbf{y} = y_1 y_2 \cdots * \cdots * \cdots y_n$  l'estratègia de correcció d'esborralls consisteix a buscar una paraula del codi que coincideixi amb  $\mathbf{y}$  fora del esborralls.

**Proposició 5.3.5.** *Un codi corregeix fins a  $\delta - 1$  esborralls.*

*Demostració.* Primer veiem que fins a  $\delta - 1$  esborralls els corregeix correctament. Si hem enviat  $\mathbf{x}$  i hem rebut  $\mathbf{y}$  i s'han produït menys de  $\delta$  esborralls llavors  $d(\mathbf{x}, \mathbf{y}) \leq \delta - 1$ . Però  $\mathbf{x}$  és la única paraula del codi que coincideix amb  $\mathbf{y}$  fora dels esborralls: si  $\mathbf{u}$  n'és una altre llavors  $\mathbf{x}$  i  $\mathbf{u}$  coincideixen fora del esborralls i per tant  $d(\mathbf{x}, \mathbf{u}) \leq \delta - 1$ , contradicció.

Veiem ara que en general no en corregeix  $\delta$  o més. Prenem  $\mathbf{x}, \mathbf{u} \in \mathcal{C}$  a distància  $\delta$ . Si enviem  $\mathbf{x}$  i rebem  $\mathbf{y}$  definit per

$$y_i := \begin{cases} x_i, & \text{si } x_i = u_i; \\ *, & \text{si } x_i \neq u_i. \end{cases}$$

s'han produït  $\delta$  esborralls i l'estratègia pot triar  $\mathbf{u}$ . □

Observem que un esborrall no és més que un error del qual coneixem la posició. Comparant les proposicions 5.3.3 i 5.3.5 veiem que el coneixement de les posicions dels errors és molt important: la capacitat de correcció es duplica com a mínim.

## 5.4 Exemple: un codi que corregeix 2 errors

A continuació presentem l'exemple d'un codi que corregeix 2 errors. L'alfabet és  $\mathbb{F}_7 = \mathbb{Z}_7$  i

$$\mathcal{C} := \left\{ \mathbf{x} \in (\mathbb{Z}_7)^7 \mid \sum_{i=1}^7 x_i = 0, \sum_{i=1}^7 i x_i = 0, \sum_{i=1}^7 i^2 x_i = 0, \sum_{i=1}^7 i^3 x_i = 0 \right\}.$$

Aquestes operacions s'efectuen a  $\mathbb{F}_7$ . Si Notem per  $H$  la matriu (amb coeficient  $\mathbb{Z}_7$ ) següent

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1^2 & 2^2 & 3^2 & 4^2 & 5^2 & 6^2 & 7^2 \\ 1^3 & 2^3 & 3^3 & 4^3 & 5^3 & 6^3 & 7^3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & -3 & -2 & -1 & 0 \\ 1 & -3 & 2 & 2 & -3 & 1 & 0 \\ 1 & 1 & -1 & 1 & -1 & 1 & 0 \end{pmatrix},$$

Tenim que les paraules del codi són les solucions del sistema homogeni amb matriu  $H$ . És a dir,

$$\mathbf{x} \in \mathcal{C} \text{ si i } H\mathbf{x}^t = 0.$$

L'estratègia per a detectar errors amb aquest codi consisteix a calcular  $H\mathbf{y}^t$  i mirar si es igual a zero. A  $H\mathbf{y}^t = 0$  se l'hi diu *la síndrome* de  $\mathbf{y}$  i es denota per  $s(\mathbf{y})$ . Per exemple, si el missatge rebut és  $\mathbf{y} = (1, 6, 3, 0, 5, 6, 0)$ ,  $s(\mathbf{y}) = (0, -1, 1, 0)$ . Com que la síndrome no és zero, hem detectat errors.

Com que  $H$  té rang 4, el sistema homogeni té tres graus de llibertat, i per tant la mida de codi és  $7^3 = 343$ . Més endavant veurem que aquest codi té distància mínima 5 i per tant pot corregir 2 errors. 343 paraules són massa per a mirar quina d'aquestes és la més propera a  $\mathbf{y}$ . Ara veurem un algorisme de descodificació més eficient.

Suposem que s'han produït dos errors durant la transmissió. Ho podem expressar així:  $\mathbf{y} = \mathbf{x} + \mathbf{e}$ , on  $\mathbf{e}$  és el vector d'errors. El vector  $\mathbf{e}$ , de longitud 7, té totes les coordenades nulles llevat de dues, la  $i$ -èsima i la  $j$ -èsima on hi ha els valors  $a$  i  $b$  respectivament. Com que  $\mathbf{x}$  és una paraula del codi,  $H\mathbf{x}^t = 0$  i per tant

$$s(\mathbf{y}) = H\mathbf{y}^t = H(\mathbf{x} + \mathbf{e})^t = H\mathbf{x}^t + H\mathbf{e}^t = H\mathbf{e}^t. \quad (5.4.1)$$

Tenint em compte que  $H\mathbf{e}^t = (a+b, ai+bj, ai^2+bj^2, ai^3+bj^3)$ , hem de resoldre les equacions

$$a + b = s_1 \quad (5.4.2)$$

$$ai + bj = s_2 \quad (5.4.3)$$

$$ai^2 + bj^2 = s_3 \quad (5.4.4)$$

$$ai^3 + bj^3 = s_4 \quad (5.4.5)$$

multiplicant l'equació (5.4.2) per  $i$  i restant-li l'equació (5.4.3) obtenim

$$b(i - j) = is_1 - s_2 \quad (5.4.6)$$

Si fem el mateix amb el parell d'equacions (5.4.3) i (5.4.4) i també amb (5.4.4) i (5.4.5) ens queda

$$b^2(i - j) = is_2 - s_3 \quad (5.4.7)$$

$$b^3(i - j) = is_3 - s_4. \quad (5.4.8)$$

Multiplicant (5.4.6) per (5.4.8) tenim que

$$b^4(i - j)^2 = (is_1 - s_2)(is_3 - s_4), \quad (5.4.9)$$

i elevant al quadrat (5.4.7),

$$b^4(i-j)^2 = (is_2 - s_3)^2. \quad (5.4.10)$$

Igualant les equacions (5.4.9) i (5.4.10), i simplificant, obtenim

$$(s_2^2 - s_1s_3)i^2 + (s_1s_4 - s_2s_3)i + (s_3^2 - s_2s_4) = 0.$$

És a dir,  $i$  és un zero del polinomi

$$p(X) = AX^2 + BX + C, \quad \text{on} \quad \begin{cases} A = s_2^2 - s_1s_3, \\ B = s_1s_4 - s_2s_3, \\ C = s_3^2 - s_2s_4. \end{cases} \quad (5.4.11)$$

Aquest polinomi  $p(X)$  rep el nom de *polinomi localitzador*. Com que els papers de  $i$  i  $j$  són simètrics,  $j$  també ha de ser un zero del polinomi localitzador. Això ens proporciona un mètode per a calcular les posicions d'error  $i$ ,  $j$ . Un cop trobades, usant les equacions (5.4.6) i (5.4.2), deduïm que:

$$b = \frac{is_1 - s_2}{i - j}, \quad a = s_1 - b. \quad (5.4.12)$$

Aquestes fórmules serviran per a calcular les magnituds d'error.

Ara ho apliquem al missatge anterior  $\mathbf{y} = (1, 6, 3, 0, 5, 6, 0)$ , del que ja hem calculat la síndrome:  $s(\mathbf{y}) = (0, -1, 1, 0)$ . Mitjançant (5.4.11) calculem el polinomi localitzador, que és  $p(X) = X^2 + X + 1$ . Aquest polinomi té dos zeros a  $\mathbb{Z}_7$ : 2 i 4. Per tant, si s'han produït 2 errors, aquests estan a les posicions  $i = 2, j = 4$ . Usant (5.4.12) obtenim  $b = 3, a = 4$  i ja tenim l'error que s'ha produït:  $\mathbf{e} = (0, 4, 0, 3, 0, 0, 0)$ . Finalment el missatge enviat és  $\mathbf{x} = \mathbf{y} - \mathbf{e} = (1, 2, 3, 4, 5, 6, 0)$ .

Aquestes fórmules funcionen sempre que hi ha exactament dos errors. El cas d'un error és encara més fàcil. Si s'a produït un error de magnitud  $a$  a la posició  $i$  llavors  $H\mathbf{e}^t = (a, ai, ai^2, ai^3)$ , i hen de resoldre les equacions

$$\begin{aligned} a &= s_1 \\ ai &= s_2 \\ ai^2 &= s_3 \\ ai^3 &= s_4, \end{aligned}$$

que tenen com a solució

$$a = s_1, \quad i = s_2/s_1. \quad (5.4.13)$$

Observem que si calculem els coeficients del polinomi localitzador en aquest cas són tots nuls. Per exemple,  $A = s_2^2 - s_1s_3 = (ai)^2 - aai^2 = 0$ . Això servirà per a distingir els casos de un o dos errors. Per exemple si rebem  $\mathbf{y}_1 = (3, 2, 1, 0, 1, 5, 4)$ , la seva síndrome és  $s(\mathbf{y}_1) = (2, 3, 1, 5)$ ; això ja ens diu que s'ha produït algun error. Si calculem el coeficient de grau 2 del polinomi localitzador obtenim  $A = 0$ . Això ens diu que no s'han produït exactament 2

errors. Només queda la possibilitat d'un error o bé més de dos. Com que en el segon cas no podem fer res, suposem que hi ha un error. Per (5.4.13) tenim que  $i = 5$ ,  $a = 2$ , i per tant  $\mathbf{e} = (0, 0, 0, 0, 2, 0, 0)$ . Finalment el missatge enviat és  $\mathbf{x} = \mathbf{y} - \mathbf{e} = (3, 2, 1, 0, 6, 5, 4)$ .

L'algorisme complet de descodificació per a aquest codi és:

1. Calculem la síndrome  $s(\mathbf{y})$  de  $\mathbf{y}$ . si  $s(\mathbf{y}) = 0$  decidim que no hi ha error. Altrament:
2. Calculem els coeficients  $A, B, C$  mitjançant les fórmules (5.4.11). Si  $A = B = C = 0$  decidim que hi ha un error i calculem  $\mathbf{e}$  usant les fórmules (5.4.13) (si  $s_1$  o  $s_2$  són zero és que hi ha més de dos errors i no podem corregir). Finalment corregim:  $\mathbf{x} = \mathbf{y} - \mathbf{e}$ . Si  $A = 0$  i  $B$  o  $C$  no són zero llavors sabem que s'han produït més de dos errors. Altrament:
3. Calculem els zeros del polinomi localitzador  $p(X)$ . Si no en té o en té un de doble hi ha més de dos errors i no podem corregir. Si en té dos de diferent  $i, j$  llavors calculem  $a$  i  $b$  mitjançant (5.4.12) i finalment corregim:  $\mathbf{x} = \mathbf{y} - \mathbf{e}$ .

Aquest algorisme només funciona si s'han produït com a molt dos errors. Observem que en alguns casos en que s'han produït més de dos errors ho podem detectar, però no sempre. Per exemple si enviem la paraula  $\mathbf{x} = (3, 2, 1, 0, 6, 5, 4)$ , i es produeixen el tres errors  $\mathbf{e} = (0, 0, 1, 3, 6, 0, 0)$  llavors la paraula rebuda és  $\mathbf{y} = (3, 2, 2, 3, 5, 5, 4)$ . Si descodifiquem aplicant l'algorisme anterior obtenim  $\mathbf{u} = (3, 2, 2, 3, 5, 1, 5)$ . La raó és que  $\mathbf{u}$  és una paraula del codi que està a distància 2 de  $\mathbf{y}$ .

*Exercici 5.4.1.* Introduïu un, dos, i tres errors a la paraula del codi  $\mathbf{x} = (1, 2, 3, 4, 5, 6, 0)$  i apliqueu l'algorisme de descodificació. Proveu amb tres o més errors diverses vegades i observeu què passa.

## 5.5 Cotes dels paràmetres d'un codi

En general un codi serà més bo quan més altes siguin la distància mínima i la taxa de transmissió (o equivalentment la mida). El problema està en que quan volem fer créixer un d'aquests paràmetres moltes vegades ens veiem obligats a fer disminuir l'altre. El problema de trobar un bon codi el podem plantejar millor de la manera següent: De entre tots els codis que tenen la mateixa longitud  $n$  i distància mínima  $d$ , quins són els que tenen mida més gran? A aquests codis els hi direm optimals i la seva mida la denotarem per  $A_q(n, d)$ :

**Definició 5.5.1.** Sigui  $1 \leq d \leq n$ .

1.  $A_q(n, d) := \max \{|\mathcal{C}| \mid \mathcal{C} \in A^n, \delta(\mathcal{C}) = d\}$ .
2. Un  $(n, M, d)_q$ -codi és optimal si  $M = A_q(n, d)$ .

El càlcul de  $A_q(n, d)$  és molt difícil i es coneix exactament en pocs casos. En general només es coneixen fites superiors i inferiors dels valors de  $A_q(n, d)$ .

Es coneixen unes quantes fites superiors de  $A_q(n, d)$  i una fita inferior. Aquesta fita inferior és la de Gilbert (o Gilbert-Varshamov) que veurem a continuació. Per a valors particulars de  $n$  i  $d$  la única manera de millorar la fita de Gilbert és construir un codi amb els paràmetres adequats. De fites superiors només en veurem tres: la de Hamming i la de Singleton i la de Plotkin.

Si  $\mathbf{x} \in A^n$  i  $d$  és un enter  $0 \leq d \leq n$ , la bola de centre  $\mathbf{x}$  i radi  $d$  és:

$$B_d(\mathbf{x}) := \{\mathbf{y} \in A^n \mid d(\mathbf{y}, \mathbf{x}) \leq d\}.$$

Per calcular el nombre de paraules de la bola  $B_d(\mathbf{x})$  hem de comptar la paraula  $\mathbf{x}$  més les que estan a distància 1 més les que estan a distància 2 ... mes les que estan a distància  $d$  de  $\mathbf{x}$ . Aquest càlcul ens dona:

$$|B_d(\mathbf{x})| = 1 + n(q-1) + \binom{n}{2}(q-1)^2 + \dots + \binom{n}{r}(q-1)^d = \sum_{i=0}^d \binom{n}{i} (q-1)^i.$$

Aquest nombre el denotarem per  $V_q(n, d)$ .

**Proposició 5.5.2 (fita de Gilbert).**

$$A_q(n, d) \geq \frac{q^n}{V_q(n, d-1)}.$$

*Demostració.* Sigui  $\mathcal{C}$  és un codi optimal amb paràmetres  $(n, M, d)_q$ . Tota paraula de  $A^n$  ha d'estar a distància com a molt  $d-1$  d'alguna paraula de  $\mathcal{C}$ ; altrament la podem afegir a  $\mathcal{C}$  sense reduir la distància mínima, contradient la optimalitat de  $\mathcal{C}$ . Així, les boles de radi  $d-1$  centrades en paraules del codi recobreixen tot  $A^n$ :  $A^n = \bigcup_{\mathbf{u} \in \mathcal{C}} B_{d-1}(\mathbf{u})$  i per tant

$$q^n \leq \sum_{\mathbf{u} \in \mathcal{C}} |B_{d-1}(\mathbf{u})| = MV_q(n, d-1),$$

d'on

$$A_q(n, d) = M \geq \frac{q^n}{V_q(n, d-1)}.$$

□

**Proposició 5.5.3 (fita de Hamming).**

$$A_q(n, d) \leq \frac{q^n}{V_q(n, \lfloor \frac{d-1}{2} \rfloor)}.$$

*Demostració.* Sigui  $\mathcal{C}$  és un codi optimal amb paràmetres  $(n, M, d)_q$ . Veiem que no hi ha cap paraula  $\mathbf{x}$  de  $A^n$  que estiguin a distància com a molt  $\lfloor \frac{d-1}{2} \rfloor$  de dues paraules diferents  $\mathbf{u}, \mathbf{v}$  de  $\mathcal{C}$ : si  $d(\mathbf{x}, \mathbf{u}) \leq \lfloor \frac{d-1}{2} \rfloor$  i  $d(\mathbf{x}, \mathbf{v}) \leq \lfloor \frac{d-1}{2} \rfloor$  llavors

$d(\mathbf{u}, \mathbf{v}) \leq d(\mathbf{u}, \mathbf{x}) + d(\mathbf{x}, \mathbf{v}) \leq 2\lfloor \frac{d-1}{2} \rfloor < d$ , contradicció. Llavors les boles de radi  $\lfloor \frac{d-1}{2} \rfloor$  centrades a les paraules de  $\mathcal{C}$  són disjunes, i per tant

$$\sum_{\mathbf{u} \in \mathcal{C}} \left| B_{\lfloor \frac{d-1}{2} \rfloor}(\mathbf{u}) \right| = MV_q(n, \lfloor \frac{d-1}{2} \rfloor) \leq q^n,$$

d'on

$$A_q(n, d) = M \leq \frac{q^n}{V_q(n, \lfloor \frac{d-1}{2} \rfloor)}.$$

□

**Proposició 5.5.4 (fita de Singleton).**

$$A_q(n, d) \leq q^{n-d+1}.$$

*Demostració.* Sigui  $\mathcal{C}$  és un codi optimal amb paràmetres  $(n, M, d)_q$ . El codi  $\mathcal{C}' \subseteq A^{n-d+1}$  obtingut esborrant els  $d-1$  primers caràcters de cada paraula de  $\mathcal{C}$  té el mateix nombre de paraules que  $\mathcal{C}$ , doncs dues paraules de  $\mathcal{C}$  difereixen en com a mínim  $d$  caràcters i per tant continuen sent diferents després d'esborrar-ne  $d-1$ . Així  $A_q(n, d) = |\mathcal{C}'| \leq q^{n-d+1}$ . □

**Proposició 5.5.5 (fita de Plotkin).** Notem per  $\theta = \frac{q-1}{q}$ .

Si  $d \geq \theta n$  llavors

$$A_q(n, d) \leq \frac{d}{d - \theta n}.$$

*Demostració.* Sigui  $\mathcal{C}$  és un codi optimal amb paràmetres  $(n, M, d)_q$ . □

## Capítol 6

# Cossos finits

En aquest capítol introduïrem els cossos finits amb el mínim de teoria possible i posant exemples. L'objectiu és que l'estudiant aprengui a construir cossos finits i fer operacions amb ells. Les propietats que caldrà conèixer les enunciarèm en forma de proposició i sense demostració.

### 6.1 El cos $\mathbb{F}_p$

En aquesta secció introduïm el primer exemple de cos:  $\mathbb{Z}_p$  el cos de les classes de congruència mòdul un nombre primer  $p$ . Podem triar representant positius menors que  $p$  i llavors  $\mathbb{Z}_p = \{0, 1, \dots, p-1\}$ . Les operacions de suma i multiplicació a  $\mathbb{Z}_p$  s'efectuen com a enters i a continuació es redueixen mòdul  $p$ . Per exemple, la taula de la multiplicació a  $\mathbb{Z}_5$  és:

$$\begin{array}{c|ccccc} \cdot & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 2 & 3 & 4 \\ 2 & 0 & 2 & 4 & 1 & 3 \\ 3 & 0 & 3 & 1 & 4 & 2 \\ 4 & 0 & 4 & 3 & 2 & 1. \end{array} \tag{6.1.1}$$

Les operacions de suma i multiplicació tenen les propietats usuals: associativa, commutativa, hi ha neutres per a la suma (el 0) i pel producte (l'1), distributiva de la multiplicació respecte a la suma i existència d'invers respecte la suma ( $-a$  és l'invers respecte a la suma de  $a$ ). Quan tenim dues operacions amb aquestes propietats se l'hi diu *Anell*. Per exemple, les classes de congruència mòdul un enter qualsevol són un anell.

**Definició 6.1.1.** 1. L'invers d'un element  $a$  respecte a la multiplicació és un element  $b$  tal que  $ab = 1$ .

2. Un cos és un anell on tot element no nul admet invers respecte la multiplicació.

És fàcil veure que en un cos l'invers d'un element no nul  $a$  és únic i el denotarem per  $a^{-1}$ .

**Proposició 6.1.2.** *Si  $p$  és primer,  $\mathbb{Z}_p$  és un cos que denotarem per  $\mathbb{F}_p$ .*

Si observem les files de la taula (6.1.1) veurem que no hi ha repeticions. L'invers d'un element el trobem buscant el 1 a la seva fila. Per exemple, l'invers del 3 és el 2; en símbols:  $3^{-1} = 2$ . Normalment, però, no tindrem la taula. Si el mòdul és petit, podem trobar l'invers 'a ull' (és a dir, provant). Per exemple, l'invers del 4 a  $\mathbb{Z}_7$  és 2 perquè  $2 \cdot 4 = 1$  a  $\mathbb{Z}_7$ . Quan el mòdul és gran caldrà utilitzar l'algorisme d'Euclides Estès per trobar una identitat de Bézout.

Una identitat de Bézout per a dos enters  $a, b$  és una expressió de la forma

$$ax + by = (a, b), \quad \text{amb } x, y \in \mathbb{Z}$$

on  $(a, b)$  denota el màxim comú divisor de  $a$  i  $b$ . L'algorisme d'Euclides Estès consisteix en obtenir, a partir de  $a$  i  $b$  successions  $(r_i, 0 \leq i \leq n)$ ,  $(q_i, 1 \leq i \leq n-1)$ ,  $(x_i, 0 \leq i \leq n)$  i  $(y_i, 0 \leq i \leq n)$  que definim a continuació. S'inicialitzen així:

$$r_0 := a, r_1 := b, x_0 = 1, x_1 = 0, y_0 := 0, y_1 := 1.$$

Els  $r_{i+1}$  i  $q_i$  són el reste i el quocient de la divisió Euclidiana de  $r_{i-1}$  per  $r_i$ . L'algorisme acaba quan obtenim un reste nul. El màxim comú divisor del parell  $a, b$  és l'últim reste no nul  $r_n$ . Els  $x$ 's i els  $y$ 's es calculen mitjançant les fórmules

$$x_{i+1} = x_i q_i - x_{i-1}, \quad y_{i+1} = y_i q_i - y_{i-1}$$

Per aplicar l'algorisme suposarem que  $a, b > 0$  i normalment es pren  $a > b$ .

Per exemple, el màxim comú divisor de 23 i 5 és 1. A la taula següent hem aplicat l'algorisme d'Euclides Estès:

r	23	5	3	2	1
q		4	1	1	
x	1	0	1	-1	2
y	0	1	-4	5	-9

Fent  $x = x_n$  i  $y = y_n$  s'obté una identitat de Bézout per  $a$  i  $b$ . A l'exemple anterior obtenim  $23 \cdot 2 + 5 \cdot (-9) = 1$ . Aquesta identitat, mòdul 23, ens dona  $5 \cdot (-9) = 1$  a  $\mathbb{Z}_{23}$ , i per tant ja hem calculat l'invers de 5 a  $\mathbb{Z}_{23}$ :  $5^{-1} = -9 = 14$ . En general, per a calcular l'invers de  $b$  mòdul  $p$  haurem de fer una identitat de Bézout per al parell  $(p, b)$  i quedar-nos amb la  $y$ ; la  $x$  no cal calcular-la. Observem que l'existència d'identitats de Bézout demostra la proposició 6.1.2.

## 6.2 Polinomis irreductibles a $\mathbb{F}_p$

En aquesta secció usarem els polinomis a coeficients  $\mathbb{F}_p$  per a construir nous cossos finits. El conjunt dels polinomis a coeficients  $\mathbb{F}_p$  el denotarem per  $\mathbb{F}_p[x]$ . Podem efectuar les operacions de suma i producte de polinomis de  $\mathbb{F}_p[x]$  de

manera anàloga a com ho fem amb els polinomis a coeficient reals, la única diferència està en que les operacions entre els coeficients les fem dins  $\mathbb{F}_p$ .

A  $\mathbb{F}_5[x]$ , per exemple, la suma dels polinomis  $f(x) = x^2 + 3x + 4$  i  $g(x) = 3x^3 + 4x^2 + 3$  és

$$f(x) + g(x) = 3x^3 + 3x + 2$$

i el seu producte  $f(x)g(x)$  és

$$\begin{aligned} (x^2 + 3x + 4)(3x^3 + 4x^2 + 3) &= \\ 3x^5 + 13x^4 + 19x^3 + 24x^2 + 9x + 12 &= 3x^5 + 3x^4 + 4x^2 + 4x^3 + 4x + 2. \end{aligned}$$

Amb aquestes operacions,  $\mathbb{F}_p[x]$  és un anell molt semblant a l'anell dels enters  $\mathbb{Z}$ .

**Definició 6.2.1.** Un polinomi  $f(x)$  de  $\mathbb{F}_p[x]$  és *irreductible* (o també *primer*) si no és igual al producte de dos polinomis de grau menor.

Per exemple, els polinomis de grau 1 sempre són irreductibles, encara que ens interessaran polinomis irreductibles de grau com a mínim 2. Els polinomis irreductibles a  $\mathbb{F}_p[x]$  juguen un paper anàleg al dels nombres primers a  $\mathbb{Z}$ . Sabem que tot enter (no nul) es pot descomposar com a producte de primers i eventualment (en el cas que sigui negatiu) el factor  $-1$ .

**Proposició 6.2.2.** *Tot polinomi de  $\mathbb{F}_p[x]$  diferent de 0 descomposa com a producte de polinomis irreductibles.*

Com que el fet de multiplicar un polinomi per una constant no nul·la (un element de  $\mathbb{F}_p$  diferent de 0) no canvia el fet de ser irreductible, els prendrem sempre amb coeficient 1 en el monomi de grau més alt (multiplicant eventualment per l'invers d'aquest coeficient). Aquests polinomis reben el nom de *polinomis monics*.

Una altra analogia amb els enters és que amb els polinomis també tenim una divisió Euclidea (una divisió amb reste). La divisió de dos polinomis de  $\mathbb{F}_p[x]$  es fa també de la manera habitual amb la única diferència que les operacions amb els coeficients dels polinomis es fan a  $\mathbb{F}_p$ . El reste sempre és un polinomi de grau menor que el del divisor.

Per exemple, si fem la divisió de  $f(x) = x^4 + 3x^2 + 4x + 1$  per  $g = x^2 + x + 2$  obtenim com a quocient  $x^2 + 4x + 2$  i reste  $4x + 2$ .

Si dividim un polinomi  $f(x) \in \mathbb{F}_p[x]$  per  $(x - a)$  (on  $a \in \mathbb{F}_p$ ) obtenim com a reste la constant  $f(a)$ , ja que si  $f(x) = q(x)(x - a) + r$  amb  $r \in \mathbb{F}$  i substituïm  $a$  a l'expressió, obtenim  $f(a) = q(a) \cdot 0 + r = r$ . Així, un polinomi és dividit per  $x - a$  si i *sols*  $f(a) = 0$ , i per tant un polinomi té un divisor de grau 1 si i *sols* té algun zero.

D'aquesta manera els polinomis de grau 2 irreductibles són els que no tenen zeros. Per exemple,  $x^2 + 1$  és un polinomi irreductible a  $\mathbb{F}_3[x]$ , mentre que

$$x^2 + x + 1$$

és l'únic polinomi irreductible de grau 2 a  $\mathbb{F}_2[x]$  (ja que és l'únic que no té zeros).

Els polinomis de grau 3 irreductibles també són els que no tenen zeros, ja que si fos reductible hauria de tenir un divisor de grau 1.

Farem servir això per a calcular tots els polinomis irreductibles de grau 3 a  $\mathbb{F}_2[x]$ . Com que no volem que 0 sigui un zero caldrà que tingui terme independent 1. Si volem que 1 tampoc sigui un zero caldrà que tingui un nombre senar de monomis. Això només ens deixa dues possibilitats:

$$x^3 + x + 1, \quad x^3 + x^2 + 1.$$

Per a grau 4 no n'hi haurà prou amb que un polinomi no tingui zeros per tal de ser irreductible. Per exemple, el polinomi de  $\mathbb{F}_2[x]$   $(x^2+x+1)^2 = x^4+x^2+1$  no té zeros i òbviament no és irreductible. Però a  $\mathbb{F}_2[x]$  és l'únic, ja que un polinomi de grau 4 no irreductible i sense zeros ha de ser producte de dos irreductibles de grau 2. Així, la llista de tots els polinomis irreductibles de grau 4 a  $\mathbb{F}_2[x]$  és la següent:

$$x^4 + x + 1, \quad x^4 + x^3 + 1, \quad x^4 + x^3 + x^2 + x + 1.$$

Si fem el mateix per a grau 5, a més d'eliminar els que tenen zeros caldrà eliminar tots els productes d'un irreductible de grau 2 per un irreductible de grau 3, dels quals només n'hi ha 2:  $(x^2 + x + 1)(x^3 + x + 1) = x^5 + x^4 + 1$  i  $(x^2 + x + 1)(x^3 + x^3 + 1) = x^5 + x + 1$ . Fent la llista de tots els irreductibles de grau 5 a  $\mathbb{F}_2[x]$  obtenim

$$x^5 + x^2 + 1, \quad x^5 + x^3 + 1, \\ x^5 + x^3 + x^2 + x + 1, \quad x^5 + x^4 + x^2 + x + 1, \quad x^5 + x^4 + x^3 + x + 1, \quad x^5 + x^4 + x^3 + x^2 + 1.$$

*Exercici 6.2.3.* Trobeu tots els polinomis irreductibles de  $\mathbb{F}_2[x]$  de grau 6.

Aquesta construcció de polinomis irreductibles la podríem seguir, però hi ha millors mètodes per a obtenir polinomis irreductibles. De fet a molts llibres de codis hi ha llistes bastant extenses de polinomis irreductibles. El que és important és el fet següent:

**Proposició 6.2.4.** *A  $\mathbb{F}_p[x]$  hi ha polinomis irreductibles de tots els graus.*

## 6.3 Els cossos $\mathbb{F}_q$

De manera anàloga a com fem congruències en els enters ara farem congruències a  $\mathbb{F}_p[x]$  mòdul un polinomi.

**Definició 6.3.1.** Siguin  $f(x), g(x), h(x)$  polinomis de  $\mathbb{F}_p[x]$ .

$g(x)$  és congruent amb  $h(x)$  mòdul  $f(x)$  sii  $f(x)$  divideix  $g(x) - h(x)$ .

Així, dos polinomis seran congruents mòdul  $f(x)$  quan tinguin el mateix reste al dividir-los per  $f(x)$ . Al conjunt de classes de congruència mòdul  $f(x)$  el denotarem per

$$\mathbb{F}_p[x]/(f).$$

Com que els restes de dividir per  $f(x)$  són polinomis de grau menor que  $f(x)$ , triarem sempre els representants d'aquesta manera. Així  $\mathbb{F}_p[x]/(f)$  conté tots els polinomis de grau menor que el de  $f(x)$ . Per exemple

$$\mathbb{F}_2[x]/(x^2 + x + 1) = \{0, 1, x, x + 1\}.$$

Si  $f(x)$  té grau  $n$ ,  $\mathbb{F}_p[x]/(f)$  té cardinal  $p^n$ .

A  $\mathbb{F}_p[x]/(f)$  podem efectuar les operacions de suma i producte de manera habitual. En el segon cas hem de fer la reducció mòdul  $f(x)$  després d'efectuar la multiplicació. Per exemple, les taules de la suma i multiplicació a  $\mathbb{F}_2[x]/(x^2 + x + 1)$  són les següents:

+	0	1	$x$	$x + 1$
0	0	1	$x$	$x + 1$
1	1	0	$x + 1$	$x$
$x$	$x$	$x + 1$	0	1
$x + 1$	$x + 1$	$x$	1	0

·	0	1	$x$	$x + 1$
0	0	0	0	0
1	0	1	$x$	$x + 1$
$x$	0	$x$	$x + 1$	1
$x + 1$	0	$x + 1$	1	$x$

Per a calcular, per exemple, el producte de  $x$  per  $x + 1$  fem  $x(x + 1) = x^2 + x$  i després, reduint mòdul  $x^2 + x + 1$ , ens queda 1. A la taula de la multiplicació podem comprovar fàcilment que  $\mathbb{F}_2[x]/(x^2 + x + 1)$  és un cos: a cada fila (excepte la del 0) apareixen tots els elements una sola vegada. Això passarà sempre que prenguem un polinomi irreductible. De fet, tenim el següent anàleg a la proposició 6.1.2:

**Proposició 6.3.2.** *Si  $f \in \mathbb{F}_p[x]$  és irreductible de grau  $n$ ,  $\mathbb{F}_p[x]/(f)$  és un cos que té  $p^n$  elements i que denotarem per  $\mathbb{F}_{p^n}$ .*

Per tal de no confondre els polinomis amb els elements de  $\mathbb{F}_4 = \mathbb{F}_2[x]/(x^2 + x + 1)$ , farem servir la notació següent:  $\alpha$  denota  $x$  (o millor dit:  $x$  mòdul  $x^2 + x + 1$ ) i reemplaçem  $x$  per  $\alpha$  a tot arreu. Llavors  $\alpha^2 + \alpha + 1 = 0$  (ja que el reste de  $x^2 + x + 1$  mòdul ell mateix és zero) i podem aïllar  $\alpha^2$ :

$$\alpha^2 = \alpha + 1. \tag{6.3.1}$$

L'equació (6.3.1) ens serveix per a fer els càlculs. Per exemple,  $\alpha(\alpha + 1) = \alpha^2 + \alpha = \alpha + 1 + \alpha = 1$ . Les taules anteriors, expressades usant  $\alpha$  enlloc de  $x$  queden:

+	0	1	$\alpha$	$\alpha + 1$
0	0	1	$\alpha$	$\alpha + 1$
1	1	0	$\alpha + 1$	$\alpha$
$\alpha$	$\alpha$	$\alpha + 1$	0	1
$\alpha + 1$	$\alpha + 1$	$\alpha$	1	0

·	0	1	$\alpha$	$\alpha + 1$
0	0	0	0	0
1	0	1	$\alpha$	$\alpha + 1$
$\alpha$	0	$\alpha$	$\alpha + 1$	1
$\alpha + 1$	0	$\alpha + 1$	1	$\alpha$

Ara construïrem  $\mathbb{F}_8$  utilitzant un polinomi irreductible de grau 3:  $f(x) = x^3 + x + 1$ ;  $\mathbb{F}_8 = \mathbb{F}_2[x]/(x^3 + x + 1)$ . Si notem  $x$  (de fet  $x$  mòdul  $x^3 + x + 1$ ) per  $\alpha$ , els elements de  $\mathbb{F}_8$  són tots els polinomis en  $\alpha$  de grau com a molt 2:

$$\mathbb{F}_8 = \{0, 1, \alpha, \alpha + 1, \alpha^2, \alpha^2 + 1, \alpha^2 + \alpha, \alpha^2 + \alpha + 1\}.$$

Per tal d'efectuar la multiplicació usarem que  $\alpha^3 + \alpha + 1 = 0$ ; és a dir:

$$\alpha^2 = \alpha + 1. \quad (6.3.2)$$

Per exemple,

$$(\alpha + 1)(\alpha^2 + 1) = \alpha^3 + \alpha^2 + \alpha + 1 = (\alpha + 1) + \alpha^2 + \alpha + 1 = \alpha^2.$$

També

$$(\alpha^2 + \alpha)(\alpha^2 + 1) = \alpha^4 + \alpha^3 + \alpha^2 + \alpha,$$

i com que  $\alpha^4 = \alpha\alpha^3 = \alpha(\alpha + 1) = \alpha^2 + \alpha$ , llavors

$$(\alpha^2 + \alpha)(\alpha^2 + 1) = (\alpha^2 + \alpha) + (\alpha + 1) + \alpha^2 + \alpha = \alpha + 1.$$

Si calculem les successives potències de  $\alpha$  anem obtenint:

$$\begin{aligned} \alpha^3 &= \alpha + 1 \\ \alpha^4 &= \alpha\alpha^3 = \alpha(\alpha + 1) = \alpha^2 + \alpha \\ \alpha^5 &= \alpha\alpha^4 = \alpha(\alpha^2 + \alpha) = \alpha^3 + \alpha^2 = (\alpha + 1) + \alpha^2 = \alpha^2 + \alpha + 1 \\ \alpha^6 &= \alpha\alpha^5 = \alpha(\alpha^2 + \alpha + 1) = \alpha^3 + \alpha^2 + \alpha = (\alpha + 1) + \alpha^2 + \alpha = \alpha^2 + 1 \\ \alpha^7 &= \alpha\alpha^6 = \alpha(\alpha^2 + 1) = \alpha^3 + \alpha = (\alpha + 1) + \alpha = 1. \end{aligned}$$

Si continuem calculant potències de  $\alpha$  obtindrem repeticions del que ja hem calculat. Això és degut a que

$$\alpha^7 = 1.$$

Per exemple,  $\alpha^{23} = \alpha^{7 \cdot 3 + 2} = (\alpha^7)^3 \alpha^2 = \alpha^2$ . En general, per a calcular  $\alpha^m$  només caldrà calcular  $m$  mòdul 7. El càlcul anterior el podem posar en forma de taula:

$\alpha$	$\alpha^2$	$\alpha^3$	$\alpha^4$	$\alpha^5$	$\alpha^6$	$\alpha^7$	(6.3.3)
$\alpha$	$\alpha^2$	$\alpha + 1$	$\alpha^2 + \alpha$	$\alpha^2 + \alpha + 1$	$\alpha^2 + 1$	1	

Amb l'ajuda d'aquesta taula podem efectuar qualsevol operació a  $\mathbb{F}_8$ : si hem de sumar (o restar) usem la notació de la fila inferior, mentre que si hem de multiplicar o dividir usem la fila superior. Per exemple per a calcular l'invers de  $\alpha + 1$  posem  $(\alpha + 1)^{-1} = (\alpha^3)^{-1} = \alpha^{-3} = \alpha^4 = \alpha^2 + \alpha$ . Com que a la taula (6.3.3) hi apareixen tots els elements de  $\mathbb{F}_8$  menys el 0,

$$\mathbb{F}_8 = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}.$$

Això ens permet treballar amb dues representacions dels elements de  $\mathbb{F}_8$  (el que haurem d'evitar és barrejar-les totes dues alhora; en aquests apunts usarem preferentment la notació en potències de  $\alpha$ ).

**Definició 6.3.3.** Un element  $\beta \in \mathbb{F}_q$  és *primitiu* si tot element no nul de  $\mathbb{F}_q$  és de la forma  $\beta^r$  per algun  $r$ .

A la construcció que hem donat anteriorment de  $\mathbb{F}_8$  l'element  $\alpha$  era primitiu i això ens ha permès obtenir la taula (6.3.3) amb tots els elements no nuls. A vegades, quan construïm un cos finit com  $\mathbb{F}_p[x]/(f)$ , l'element  $\alpha = x \pmod{f(x)}$  no és primitiu. En aquests casos haurem de reemplaçar  $\alpha$  per un altre element  $\beta$  que sigui primitiu. Això sempre és possible:

**Proposició 6.3.4.** *En tot cos finit sempre hi ha elements primitius.*

El concepte següent és força útil per a buscar elements primitius.

**Definició 6.3.5.** L'ordre d'un element  $\beta \in \mathbb{F}_q$  no nul és

$$\text{ord } \beta = \min \{n \geq 1 \mid \beta^n = 1\}.$$

Com que el conjunt  $\{\beta^n \mid n \in \mathbb{N}\}$  és finit, sempre hi haurà dos exponents  $m < n$  tals que  $\beta^m = \beta^n$  i per tant  $\beta^{n-m} = 1$ . És a dir, sempre hi ha un  $n \geq 1$  tal que  $\beta^n = 1$  i per tant la definició anterior té sentit.

Per exemple, a la construcció de  $\mathbb{F}_8$  que hem fet,  $\text{ord}(\alpha) = 7$ . L'ordre té les propietats següents:

**Proposició 6.3.6.** *Si  $\beta \in \mathbb{F}_q$  no nul.*

1.  $\beta$  és primitiu sii  $\text{ord } \beta = q - 1$ .
2.  $\text{ord } \beta$  divideix  $q - 1$ .
3.  $\text{ord}(\beta^r) = \frac{\text{ord } \beta}{(\text{ord } \beta, r)}$ .

No hi ha cap mètode específic per a trobar elements primitius, però les propietats anteriors ens ajuden a buscar-los. Per exemple, a  $\mathbb{F}_7$  l'ordre del 2 és 3, ja que  $2^3 = 1$ . Això ens diu que 2 no és primitiu. En canvi, com que ni  $3^2 = 2$  ni  $3^3 = 6$  són 1, l'ordre de 3 no és ni 2 ni 3, i per tant (ha de ser un divisor de 6) és 6. Així 3 és primitiu a  $\mathbb{F}_7$ . Un cop trobat un element primitiu  $\beta$ , com que tots els elements no nuls són de la forma  $\beta^r$ , per la tercera propietat anterior, tots els primitius són de la forma  $\beta^r$  amb  $(r, q - 1) = 1$ . Això ens diu que hi ha  $\varphi(q - 1)$  elements primitius a  $\mathbb{F}_q$  ( $\varphi$  és la funció Phi de Euler;  $\varphi(n)$  és el nombre d'enters menors que  $n$  primers amb  $n$ ). Per exemple, tots els primitius de  $\mathbb{F}_7$  són  $3^1 = 3$  i  $3^5 = 5$ .

*Exercici 6.3.7.* Trobeu tots els elements primitius de  $\mathbb{F}_3$ ,  $\mathbb{F}_5$ ,  $\mathbb{F}_{11}$  i  $\mathbb{F}_{13}$ . Feu una taula anàloga a (6.3.3) per a un d'ells en cada cos. Utilitzeu la taula per a calcular  $2^{100000}$  en aquests quatre cossos.

*Exemple 6.3.8.* Anem a construir  $\mathbb{F}_9$  usant el polinomi  $f(x) = x^2 + 1 \in \mathbb{F}_3[x]$ . Com que  $f$  no té zeros a  $\mathbb{F}_3$ , és irreductible a  $\mathbb{F}_3[x]$ . Llavors  $\mathbb{F}_9 = \mathbb{F}_3[x]/(x^2 + 1) = \{0, 1, 2, \alpha, \alpha + 1, \alpha + 2, 2\alpha, 2\alpha + 1, 2\alpha + 2\}$ . Com que  $\alpha^2 = -1$  llavors  $\alpha^4 = 1$  i  $\alpha$  no és primitiu. Cal trobar un altre element primitiu. Ja podem descartar  $\alpha$ ,  $\alpha^2 = 2$  i  $\alpha^3 = 2\alpha$ . Si prenem  $\beta = \alpha + 1$ , llavors  $\beta^2 = \alpha^2 + 2\alpha + 1 = 2\alpha$  i  $\beta^4 = (2\alpha)^2 = 2^2\alpha^2 = 2 \neq 1$ . Per tant  $\beta$  té ordre 8 (ha de tenir ordre divisor de

8 i no té ordre 2 ni 4) i és primitiu. Si construïm la taula de les potències de  $\beta$  obtenim

$\beta$	$\beta^2$	$\beta^3$	$\beta^4$	$\beta^5$	$\beta^6$	$\beta^7$	$\beta^8$
$\alpha + 1$	$2\alpha$	$2\alpha + 1$	$2$	$2\alpha + 2$	$\alpha$	$\alpha + 2$	$1$

(6.3.4)

*Exercici 6.3.9.* Construïu  $\mathbb{F}_8$  usant el polinomi  $x^3 + x^2 + 1$  i comproveu que  $\alpha = x \pmod{x^3 + x^2 + 1}$  és primitiu. Demostreu que tot element de  $\mathbb{F}_8$  (diferent de 0 i 1) és primitiu.

*Exemple 6.3.10.* Ara construïm  $\mathbb{F}_{16}$  usant el polinomi  $x^4 + x + 1$ . Notant, com habitualment  $\alpha = x \pmod{x^4 + x + 1}$ , tindrem que

$$\mathbb{F}_{16} = \{0, 1, \alpha, \alpha + 1, \alpha^2, \alpha^2 + 1, \alpha^2 + \alpha, \alpha^2 + \alpha + 1, \alpha^3, \alpha^3 + 1, \alpha^3 + \alpha, \alpha^3 + \alpha + 1, \alpha^3 + \alpha^2, \alpha^3 + \alpha^2 + 1, \alpha^3 + \alpha^2 + \alpha, \alpha^3 + \alpha^2 + \alpha + 1\},$$

i les operacions s'efectuen tenint en compte l'equació:

$$\alpha^4 = \alpha + 1.$$

Llavors  $\alpha^5 = \alpha\alpha^4 = \alpha^2 + \alpha \neq 1$  i com que  $\alpha^3 \neq 1$  llavors l'ordre de  $\alpha$  és 15 i per tant és primitiu. Si construïm la taula obtenim:

$\alpha$	$\alpha^2$	$\alpha^3$	$\alpha^4$	$\alpha^5$	$\alpha^6$	$\alpha^7$	$\alpha^8$	$\alpha^9$
$\alpha$	$\alpha^2$	$\alpha^3$	$\alpha + 1$	$\alpha^2 + \alpha$	$\alpha^3 + \alpha^2$	$\alpha^3 + \alpha + 1$	$\alpha^2 + 1$	$\alpha^3 + \alpha$
$\alpha^{10}$		$\alpha^{11}$		$\alpha^{12}$		$\alpha^{13}$	$\alpha^{14}$	$\alpha^{15}$
$\alpha^2 + \alpha + 1$		$\alpha^3 + \alpha^2 + \alpha$		$\alpha^3 + \alpha^2 + \alpha + 1$		$\alpha^3 + \alpha^2 + 1$	$\alpha^3 + 1$	$1$

(6.3.5)

De les proposicions 6.2.4 i 6.3.2 s'en segueix que per a cada nombre  $q$  que sigui potència de primer hi ha un cos finit de cardinal  $q$ . Es pot demostrar que és únic llevat d'isomorfia. No és difícil veure, a més, que no hi ha cossos finits en cap altra cardinalitat. Tot això ho enunciem sense demostració:

**Teorema 6.3.11.** *Per a cada nombre natural  $q$  que sigui potència de primer, hi ha un únic cos llevat d'isomorfia de cardinal  $q$ . No hi ha cossos finits en altres cardinalitats.*

És a dir, independentment del polinomi irreductible  $f(x) \in \mathbb{F}_p[x]$  de grau  $n$  que prenguem,  $\mathbb{F}_p[x]/(f)$  és el mateix cos  $\mathbb{F}_{p^n}$ .

# Capítol 7

## Codis lineals

### 7.1 Codis lineals: primeres propietats

A partir d'ara l'alfabet serà un cos finit  $\mathbb{F}_q$ . Quan no vulguem fer referència al seu cardinal posarem senzillament  $\mathbb{F}$ . L'avantatge addicional que això ens proporciona és que podem operar amb les lletres de l'alfabet. L'inconvenient és que limita la seva mida:  $q = |\mathbb{F}|$  ha de ser potència de primer. Recordem que  $\mathbb{F}^n$  és un  $\mathbb{F}$ -espai vectorial amb la suma component a component i el producte per escalar multiplicant totes les components: si  $\mathbf{x} = (x_1, \dots, x_n)$ ,  $\mathbf{y} = (y_1, \dots, y_n)$  i  $\lambda \in \mathbb{F}$ ,

$$\begin{aligned}\mathbf{x} + \mathbf{y} &:= (x_1 + y_1, \dots, x_n + y_n) \\ \lambda \mathbf{x} &:= (\lambda x_1, \dots, \lambda x_n).\end{aligned}$$

**Definició 7.1.1.** Un codi lineal de longitud  $n$  és un subespai vectorial de  $\mathbb{F}^n$ .

Els codis lineals són doncs codis de bloc amb les propietats addicionals següents:

- Si  $\mathbf{u}, \mathbf{v} \in \mathcal{C}$  llavors  $\mathbf{u} + \mathbf{v} \in \mathcal{C}$ .
- Si  $\mathbf{u} \in \mathcal{C}$  y  $\lambda \in \mathbb{F}$  llavors  $\lambda \mathbf{u} \in \mathcal{C}$

Més en general, tot codi lineal és tancat per combinació lineal de paraules del codi:

$$\text{si } \mathbf{u}_1, \dots, \mathbf{u}_n \in \mathcal{C} \text{ i } \lambda_1, \dots, \lambda_n \in \mathbb{F} \text{ llavors } \sum_{i=1}^n \lambda_i \mathbf{u}_i \in \mathcal{C}.$$

Els paràmetres d'un codi lineal són exactament els mateixos que per un codi de bloc. Observem ara que la dimensió tal com l'havíem definit per codis de bloc coincideix amb la dimensió com a espai vectorial: si la dimensió com a espai vectorial és  $k$  i  $\mathbf{v}_1, \dots, \mathbf{v}_k$  és una base de  $\mathcal{C}$  llavors tot  $\mathbf{u} \in \mathcal{C}$  s'expressa de manera única  $\mathbf{u} = \sum_{i=1}^k \lambda_i \mathbf{v}_i$  amb  $\lambda_i \in \mathbb{F}$ . Això vol dir que la mida  $M$  és  $q^k$  i per tant la dimensió del codi és  $\log_q M = k$ . Com que la dimensió sempre serà un enter s'acostuma a donar la dimensió en lloc de la mida. En aquest cas direm que  $\mathcal{C}$  és un  $[n, k, \delta]_q$ -codi i això ja indicarà per conveni que és lineal.

*Exemple 7.1.2.* 1.  $\mathcal{C}_1 = \langle 1100, 0011 \rangle = \{0000, 1100, 0011, 1111\}$  binari. És un  $[4, 2, 2]$ -codi.

2.  $\mathcal{C}_2 = \langle 1100, 0011 \rangle_3 = \{0000, 1100, 2200, 0011, 0022, 1111, 1122, 2211, 2222\}$  ternari. És un  $[4, 2, 2]_3$ -codi.

3.  $\mathcal{C}_3 = \langle 1100, 0011, 0110 \rangle = \{0000, 1100, 0011, 0110, 1111, 1010, 0101, 1001\}$  binari. És un  $[4, 3, 2]$ -codi.

Els codis lineals presenten una sèrie d'avantatges sobre els codis de bloc que anirem veient. La primera és que el càlcul de la distància mínima es simplifica bastant: només cal calcular les distàncies de les paraules del codi a la paraula  $\mathbf{0} = (0, \dots, 0)$ , que sempre és del codi. Això s'anomena pes d'una paraula.

**Definició 7.1.3 (pes de Hamming).** Si  $\mathbf{x} \in \mathbb{F}^n$  el pes de  $\mathbf{x}$  és  $\|\mathbf{x}\| = d(\mathbf{x}, \mathbf{0})$ .

El pes de  $\mathbf{x}$  és el nombre de coordenades no nul·les de  $x$ .

*Observació 7.1.4.* 1.  $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{x} + \mathbf{z}, \mathbf{y} + \mathbf{z})$ .

2.  $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$ .

3.  $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ .

*Demostració.* 1. és obvi, 2. s'obté prenent  $\mathbf{z} = -\mathbf{y}$  a 1. i 3. és la desigualtat triangular amb 0,  $x$  i  $x + y$ .  $\square$

**Proposició 7.1.5.** Si  $\mathcal{C}$  és un codi lineal llavors

$$\delta(\mathcal{C}) = \min \{\|\mathbf{u}\| \mid \mathbf{u} \in \mathcal{C}, \mathbf{u} \neq \mathbf{0}\}.$$

*Demostració.* Veiem que de fet

$$\{\|\mathbf{u}\| \mid \mathbf{u} \in \mathcal{C}, \mathbf{u} \neq \mathbf{0}\} = \{d(\mathbf{u}, \mathbf{v}) \mid \mathbf{u}, \mathbf{v} \in \mathcal{C}, \mathbf{u} \neq \mathbf{v}\}.$$

La inclusió d'esquerra a dreta surt perquè si  $\mathbf{u} \in \mathcal{C}$ , llavors  $\|\mathbf{u}\| = d(\mathbf{u}, \mathbf{0})$  i  $\mathbf{0} \in \mathcal{C}$ . D'altra banda si  $\mathbf{u}, \mathbf{v} \in \mathcal{C}$  llavors  $\mathbf{u} - \mathbf{v} \in \mathcal{C}$  i  $d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|$ , d'on s'obté l'altre inclusió.  $\square$

## 7.2 Matriu generadora d'un codi lineal

Si  $\mathcal{C}$  és un  $[n, k]$ -codi i  $\mathbf{g}_1, \dots, \mathbf{g}_k \in \mathbb{F}^n$  n'és una base, sabem que tot  $\mathbf{u} \in \mathcal{C}$  és de la forma

$$\mathbf{u} = m_1 \mathbf{g}_1 + \dots + m_k \mathbf{g}_k \quad (7.2.1)$$

amb  $m_i \in \mathbb{F}$  únics. Si  $g_i = (g_{i,1}, \dots, g_{i,n})$  l'equació (7.2.1) s'expressa matricialment així:

$$(m_1 \quad \dots \quad m_k) \begin{pmatrix} g_{1,1} & \dots & g_{1,n} \\ \vdots & & \vdots \\ g_{k,1} & \dots & g_{k,n} \end{pmatrix} \quad (7.2.2)$$

Si notem  $\mathbf{m} = (m_1 \cdots m_k)$  i  $G = \begin{pmatrix} g_{1,1} & \cdots & g_{1,n} \\ \vdots & & \vdots \\ g_{k,1} & \cdots & g_{k,n} \end{pmatrix}$ , (7.2.2) s'escriu

$$\mathbf{u} = \mathbf{m}G \quad (7.2.3)$$

A la matriu  $G$  l'hi direm matriu generadora:

**Definició 7.2.1.** Una *matriu generadora* d'un codi lineal  $\mathcal{C}$  és una matriu que té per files una base de  $\mathcal{C}$ .

*Observació 7.2.2.* 1. Tota matriu generadora té rang igual al nombre de files (direm que té rang màxim). El nombre de files és la dimensió i el nombre de columnes és la longitud.

2. Si  $G$  és matriu generadora de  $\mathcal{C}$  llavors  $\mathcal{C} = \{\mathbf{m}G \mid \mathbf{m} \in \mathbb{F}^k\}$

Com que la matriu  $G$  determina el codi, moltes vegades, per donar el codi el que fem és donar una matriu generadora. La única condició que ha de satisfer una matriu per tal de ser matriu generadora d'un codi lineal és que tingui com a mínim tantes columnes com files i que sigui de rang màxim.

*Exemple 7.2.3.* A l'exemple 7.1.2, la matriu

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

és una matriu generadora del primer codi  $\mathcal{C}_1$ . La mateixa matriu també és una matriu generadora del segon codi  $\mathcal{C}_2$ . La matriu

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

és una matriu generadora de  $\mathcal{C}_3$ .

Recordem que les transformacions elementals per files són d'un dels tres tipus següents:

- permutar files
- multiplicar una fila per un escalar no nul
- sumar a una fila una combinació lineal de les altres

Si fem transformacions elementals per files de  $G$ , obtindrem altres matrius generadores del mateix codi, ja que aquests tres operacions elementals transformen bases en bases. El mètode de Gauss-Jordan permet, mitjançant transformacions elementals per files i **eventualment permutant columnes** arribar a una matriu de la forma

$$(I_k \mid B), \quad (7.2.4)$$

on  $I_k$  és la identitat  $k \times k$  i  $B$  és una matriu  $k \times (n - k)$ . Si hem tingut que efectuar permutacions de columnes la matriu així obtinguda no té perquè generar el mateix codi. Per exemple, si hem permutat la segona i la quarta columna, hem permutat la segona i la quarta posició de totes les paraules del codi. Si volem que generi el mateix codi haurem de desfer les permutacions de columnes que haguem fet durant el procés. Això demostra que tot codi admet una matriu generadora com (7.2.4) llevat de permutació de columnes. Quan la matriu generadora tingui exactament la forma (7.2.4) direm que és *sistemàtica a l'esquerra*. Quan tingui la identitat a la dreta en direm *sistemàtica a la dreta*. No tots els codis admeten matrius generadores sistemàtiques. Als codis que admeten una matriu generadora sistemàtica reben el nom de *codis sistemàtics*.

### 7.2.1 codificació a partir d'una matriu generadora

Un cop tenim una matriu generadora  $G$  d'un codi  $\mathcal{C}$ , la codificació es fa de la manera següent. El missatge que volem enviar el trenquem en blocs de longitud  $k$ . Si  $\mathbf{m} \in \mathbb{F}^k$  és un bloc de longitud  $k$ , el codificarem així:

$$\begin{aligned} \mathbb{F}^k &\rightarrow \mathbb{F}^n \\ \mathbf{m} &\mapsto \mathbf{x} = \mathbf{m}G, \end{aligned}$$

i enviem  $\mathbf{x}$ .

Observem que la codificació converteix blocs de longitud  $k$  en blocs de longitud  $n$ . Aquí es veu que la taxa de transmissió és  $k/n$  i que la taxa de redundància és  $(n - k)/n$ .

Quan  $G$  és sistemàtica a l'esquerra i  $\mathbf{x} = \mathbf{m}G$  els primers  $k$  caràcters de  $\mathbf{x}$  són els de  $\mathbf{m}$ . Si  $G$  és sistemàtica a la dreta  $\mathbf{m}$  ocupa les últimes  $k$  posicions de  $\mathbf{x}$ . Un dels avantatges de les codificacions sistemàtiques és la facilitat de recuperar  $\mathbf{m}$  a partir de  $\mathbf{x}$ .

Tornant als exemples de 7.1.2:

*Exemple 7.2.4.* 1. El codi binari  $\mathcal{C}_1 = \langle 1100, 0011 \rangle$  no admet una forma sistemàtica. Observem però que tenim la identitat situada a la segona i tercera columna. Si utilitzem la matriu generadora

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix},$$

El missatge  $\mathbf{m} = (1, 1)$  és codifica com

$$\mathbf{m}G = (1, 1) \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} = (1, 1, 1, 1).$$

2. Una matriu generadora del codi binari  $\mathcal{C}_3 = \langle 1100, 0011, 0110 \rangle$  és la següent:

$$G_1 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}.$$

Fent transformacions per files arribem a la forma sistemàtica:

$$G_2 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

Si codifiquem  $\mathbf{m} = (0, 1, 1)$  amb la primera obtenim

$$\mathbf{m}G_1 = (0, 1, 1) \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} = (0, 1, 0, 1),$$

mentre que si el codifiquem amb la segona obtenim

$$\mathbf{m}G_2 = (0, 1, 1) \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} = (0, 1, 1, 0).$$

### 7.3 matriu de control

Un subespai  $\mathcal{C}$  de  $\mathbb{F}^n$  també el podem definir com el conjunt de solucions d'un sistema lineal homogeni:

$$\begin{cases} h_{1,1}x_1 + \cdots + h_{1,n}x_n = 0 \\ \vdots \\ h_{r,1}x_1 + \cdots + h_{r,n}x_n = 0 \end{cases} \quad (7.3.1)$$

Expressat matricialment

$$\begin{pmatrix} h_{1,1} & \cdots & h_{1,n} \\ \vdots & & \vdots \\ h_{r,1} & \cdots & h_{r,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \quad (7.3.2)$$

Si notem  $\mathbf{x} = (x_1, \dots, x_n)$  i  $H = \begin{pmatrix} h_{1,1} & \cdots & h_{1,n} \\ \vdots & & \vdots \\ h_{r,1} & \cdots & h_{r,n} \end{pmatrix}$ , l'equació (7.3.2) s'escriu:

$$H\mathbf{x}^t = \mathbf{0}. \quad (7.3.3)$$

Si  $k$  és la dimensió de  $\mathcal{C}$  llavors  $k = n - \text{rang}(H)$ . Eliminant files podem suposar que  $r = \text{rang}(H)$  i per tant  $k + r = n$ . A la matriu  $H$  l'hi direm matriu de control. D'ara en endavant usarem  $r$  per denotar  $n - k$  i l'anomenarem *codimensió*.

**Definició 7.3.1.** Una *matriu de control* d'un codi lineal  $\mathcal{C}$  és una matriu  $H$  de rang màxim tal que

$$\mathbf{x} \in \mathcal{C} \text{ sii } H\mathbf{x}^t = \mathbf{0},$$

Pel que hem dit abans les matrius de control tenen  $r$  files i  $n$  columnes.

### 7.3.1 Relació entre matriu generadora i matriu de control

La proposició següent ens diu quina relació hi ha entre les matrius generadores i les matrius de control d'un mateix codi.

**Proposició 7.3.2.**  $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathbb{F}^n$  codis lineals de la mateixa longitud i dimensió. Suposem que  $G$  és una matriu generadora de  $\mathcal{C}_1$  i  $H$  és una matriu de control de  $\mathcal{C}_2$ . Llavors  $\mathcal{C}_1 = \mathcal{C}_2$  si i  $HG^t = 0$

*Demostració.* Si  $HG^t = 0$  és fàcil comprovar que  $\mathcal{C}_1 \subseteq \mathcal{C}_2$ :  $H(\mathbf{m}G)^t = HG^t\mathbf{m} = 0\mathbf{m} = 0$ . Com que tenen la mateixa dimensió han de ser iguals. Recíprocament, si són iguals i prenem  $\mathbf{m} \in \mathbb{F}^k$  llavors  $HG^t\mathbf{m}^t = H(\mathbf{m}G)^t = 0$ . Com que  $\mathbf{m}$  és qualsevol, llavors  $HG^t = 0$ .  $\square$

Com que la matriu trasposta de  $HG^t$  és  $GH^t$ , la condició

$$HG^t = 0 \tag{7.3.4}$$

és equivalent a  $GH^t = 0$ . Quan volem trobar  $G$  a partir de  $H$  (per trobar  $H$  a partir de  $G$  es fa anàlogament) el que hem de fer es trobar una base de solucions del sistema:

$$Hx^t = 0 \tag{7.3.5}$$

(o del sistema  $Gx^t = 0$  si volem trobar  $H$  a partir de  $G$ ). És a dir, el pas de  $G$  a  $H$  i a l'inrevés es redueix a trobar una base de solucions d'un sistema homogeni.

Si  $G = (I_k \mid B)$  està en forma sistemàtica, és fàcil comprovar que

$$H = (-B^t \mid I_{n-k}) \tag{7.3.6}$$

és una matriu de control. Com que  $H$  és de rang màxim amb les dimensions adequades veiem que se satisfà (7.3.4):

$$HG^t = (-B^t \mid I_{n-k}) \begin{pmatrix} I_k \\ - \\ B^t \end{pmatrix} = -B^t I_k + I_{n-k} B^t = -B^t + B^t = 0.$$

Un bon mètode per passar de  $G$  a  $H$  (i a l'inrevés) consisteix a aplicar Gauss-Jordan a  $G$ , **permutant columnes eventualment**, i aplicar la fórmula (7.3.6). Al final **cal desfer a  $H$  les permutacions de columnes que hàgim fet a  $G$** , és a dir, aplicar a  $H$  la permutació inversa de la que hem aplicat a  $G$ . Aquest mètode és equivalent a resoldre (7.3.5) per Gauss-Jordan.

Tornem als exemples de 7.1.2.

*Exemple 7.3.3.* 1. Sabem que

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

és una matriu generadora de  $\mathcal{C}_1$ . En aquest cas, si fem el procés obtenim la mateixa matriu.

2. Sabem que

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

és una matriu generadora de  $\mathcal{C}_3$ . Fent Gauss-Jordan arribem a

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

i per tant  $H = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}$  és una matriu de control.

Anàlogament a com ho fem a  $G$ , podem aplicar transformacions elementals per files a  $H$  i obtenim noves matrius de control del codi. La raó és que aplicar aquestes transformacions a les equacions d'un sistema lineal no en canvia el conjunt de les seves solucions. Tenint en compte la simetria de l'equació (7.3.4), podem intercanviar els papers de  $G$  i  $H$  i per tant el mateix sistema serveix per a obtenir  $G$  a partir de  $H$ .

### 7.3.2 Distància mínima i matriu de control

El càlcul de la distància mínima d'un codi es pot simplificar gràcies al fet següent.

**Proposició 7.3.4.**  *$\mathcal{C}$  un codi lineal amb matriu de control  $H$ .  $\delta(\mathcal{C})$  és igual al mínim nombre de columnes de  $H$  que són linealment dependents.*

*Demostració.* Notem per  $s$  el mínim nombre de columnes dependents i per  $\mathbf{h}_1, \dots, \mathbf{h}_n$  les columnes de  $H$ . Si  $\mathbf{x} \in \mathcal{C}$  i té pes  $\delta$ ,  $H\mathbf{x}^t = x_1\mathbf{h}_1 + \dots + x_n\mathbf{h}_n = 0$  i per tant les  $\delta$  columnes de  $H$  que corresponen a les coordenades no nul·les de  $\mathbf{x}$  són linealment dependents. Això demostra que  $\delta \geq s$ . Inversament, si tenim  $s$  columnes de  $H$  dependents  $\lambda_{i_1}\mathbf{h}_{i_1} + \dots + \lambda_{i_s}\mathbf{h}_{i_s} = 0$  llavors tots els  $\lambda_{i_j}$  són no nuls (altrament hi hauria menys de  $s$  columnes dependents) i la paraula  $\mathbf{x}$  definida per  $x_{i_j} = \lambda_{i_j}$  per  $j = 1 \dots s$  i zero a les restants coordenades és del codi i té pes  $s$ . Això demostra que  $s \geq \delta$ .  $\square$

**Corol·lari 7.3.5.** *En tot codi lineal  $\delta \leq r + 1$ .*

De fet no és gaire difícil de demostrar que aquesta desigualtat també val per als codis no lineals, i és coneguda com fita de Singleton. Als codis que satisfan  $\delta = r + 1$  en direm *separables a màxima distància*, en anglès MDS.

Aquests codis són un cas particular del que s'anomena codis optimals. Un codi és *optimal* quan té la màxima dimensió possible entre tots els codis de la mateixa longitud i mateixa distància mínima. L'interès dels codis optimals està en que, per a una longitud i capacitat correctora fixada, són els que tenen màxima taxa de transmissió.

És interessant expressar el corol·lari 7.3.5 en termes de la capacitat detectora i correctora:

- La capacitat detectora és com a molt  $r$ .

- La capacitat correctora  $\rho$  és com a molt  $r/2$ .

*Exemple 7.3.6.* 1. Sigui  $\mathcal{C}_1$  el codi binari que té per matriu generadora

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Una matriu de control és

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Com que no hi ha columnes nul·les ( $\delta > 1$ ) ni dues columnes iguals ( $\delta > 2$ ) i la primera és la suma de la cinquena i la sisena,  $\delta = 3$ . És un  $[7, 4, 3]$ -codi.

2. Sigui  $\mathcal{C}_2$  el codi que té per matriu generadora

$$G = \begin{pmatrix} \alpha^2 & \alpha & 1 & 0 & 0 \\ \alpha & \alpha^2 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix},$$

sobre  $\mathbb{F}_4 = \mathbb{F}_2[x]/(x^2 + x + 1)$ , on  $\alpha = \bar{x}$ . Una matriu de control de  $\mathcal{C}_2$  és

$$H = \begin{pmatrix} 1 & 0 & \alpha^2 & \alpha & 1 \\ 0 & 1 & \alpha & \alpha^2 & 1 \end{pmatrix}.$$

Es veu que  $\delta = 3$  ja que no hi ha dues columnes dependents i que qualssevol 3 columnes ho són, ja que sobrepassen el rang de  $H$ . És un  $[5, 3, 3]_4$ -codi.

### 7.3.3 Detecció i correcció per síndrome

Les matrius de control s'utilitzen tant per a la detecció com per a la correcció d'errors. Un cop triada una matriu de control  $H$ , si  $\mathbf{y} \in \mathbb{F}^n$  s'anomena *síndrome de  $\mathbf{y}$*  al vector  $S(\mathbf{y}) = H\mathbf{y}^t$ . La detecció d'errors és molt simple: es mira si la síndrome del missatge rebut  $\mathbf{y}$  és zero o no.

Pel que fa a la correcció, observem que si s'ha enviat  $\mathbf{x}$  i s'ha rebut  $\mathbf{y} = \mathbf{x} + \mathbf{e}$ , on  $\mathbf{e}$  denota el vector d'errors

$$S(\mathbf{y}) = H(\mathbf{x} + \mathbf{e})^t = H\mathbf{x}^t + H\mathbf{e}^t = H\mathbf{e}^t,$$

ja que  $\mathbf{x}$  és una paraula del codi. Com que el nombre d'errors és el pes de  $\mathbf{e}$ , el descodificador per mínima distància ha de buscar  $\mathbf{e}$  de pes com a molt  $\rho$  (que serà de pes mínim) tal que

$$H\mathbf{e}^t = S(\mathbf{y}). \quad (7.3.7)$$

Si mirem (7.3.7) com un sistema amb  $\mathbf{e}$  com a incògnita ( $H$  i  $S(\mathbf{y})$  són conegudes), la solució de (7.3.7) de pes mínim rep el nom de *líder* (o líder de la

classe de les solucions dels sistema). Casa síndrome té un líder. Buscar el líder correspon a buscar el mínim nombre de columnes de  $H$  de les quals  $S(\mathbf{y})$  n'és combinació lineal. Aquest mètode en general és massa lent i el que se sol fer és tenir precalculada una taula de síndromes i líders. Com que només necessitem els líders quan aquests tenen pes com a molt  $\rho$ , això en principi (si no tenim en compte el cost en temps i espai) és senzill: només cal calcular  $H\mathbf{e}^t$  per a tots  $\mathbf{e}$  amb  $\|\mathbf{e}\| \leq \rho$ . En general, però, això és massa costós en espai doncs la taula haurà de tenir tantes entrades com paraules de longitud  $n$  i pes com a molt  $\rho$ . Aquest nombre és

$$\sum_{i=1}^{\rho} \binom{n}{i} (q-1)^i,$$

i en els codis que s'empren actualment resulta impracticable. Per exemple, en televisió digital per satèl·lit, actualment s'utilitza un codi Reed-Solomon  $RS(204, 188)$  (utilitzant la notació del capítol 9). Aquest és un codi lineal amb paràmetres  $[204, 188, 17]_{256}$  que corregeix fins a 8 errors. Una taula de síndromes i líders per a aquest codi tindria

$$\sum_{i=1}^8 \binom{204}{i} (255)^i = 1157599170407964753465733575201420 \simeq 1.1576 \cdot 10^{33}$$

entrades!

Els mètodes aquí descrits funcionen amb qualsevol matriu de control  $H$ , encara que el càlcul de les síndromes i els líders depenen de la  $H$  triada. Cal fixar una matriu de control per a tot el procediment.

*Exemple 7.3.7.* 1. el codi  $\mathcal{C}_1$  de l'exercici 7.3.6 corregeix un error. Si codifiquem  $\mathbf{m} = (1, 0, 1, 0)$  amb  $G$  obtenim  $\mathbf{x} = \mathbf{m}G = (1, 0, 1, 0, 1, 0, 1)$ . Si es produeix l'error  $\mathbf{e} = (0, 0, 0, 1, 0, 0, 0)$  rebem  $\mathbf{y} = (1, 0, 1, 1, 1, 0, 1)$ . calculant la síndrome amb  $H$  obtenim  $S(\mathbf{y}) = (1, 1, 1)$ . Com que  $(1, 1, 1)$  és la quarta columna de  $H$ , el descodificador detecta que hi ha un error a la quarta posició i el corregeix. La taula de síndromes i líders, en aquest cas, consisteix en les set columnes de  $H$  acompanyades amb el corresponent vector de pes 1 que indica la columna:

Líder	1000000	0100000	0010000	0001000
Síndrome	110	101	011	111
	0000100	0000010	0000001	
	100	010	001	

2. el codi  $\mathcal{C}_2$  de l'exercici 7.3.6 corregeix un error. Si hem rebut la paraula  $\mathbf{y} = (\alpha^2, 0, \alpha^2, \alpha, \alpha^2)$ , calculant la síndrome amb  $H$  obtenim  $S(\mathbf{y}) = (1, \alpha^2)$  i deduïm que s'han produït errors. Com que  $(1, \alpha^2) = \alpha(\alpha^2, \alpha)$  el descodificador detecta que hi ha un error de magnitud  $\alpha$  a la tercera posició, i.e.,  $\mathbf{e} = (0, 0, \alpha, 0, 0)$  i per tant  $\mathbf{x} = \mathbf{y} - \mathbf{e} = (\alpha^2, 0, 1, \alpha, \alpha^2)$ .

Si hem usat  $G$  per codificar llavors el missatge està a les tres últimes posicions de  $\mathbf{x}$ ,  $\mathbf{m} = (1, \alpha, \alpha^2)$ . Si calculem la taula de síndromes i líders en aquest cas obtenim:

Líder	10000	$\alpha 0000$	$\alpha^2 0000$	01000	$0\alpha 000$	$0\alpha^2 000$	00100
Síndrome	10	$\alpha 0$	$\alpha^2 0$	01	$0\alpha$	$0\alpha^2$	$\alpha^2 \alpha$
$00\alpha 00$	$00\alpha^2 00$	00010	000 $\alpha 0$	000 $\alpha^2 0$	00001	0000 $\alpha$	0000 $\alpha^2$
$1\alpha^2$	$\alpha 1$	$\alpha\alpha^2$	$\alpha^2 1$	$1\alpha$	11	$\alpha\alpha$	$\alpha^2 \alpha^2$

El càlcul de  $\mathbf{e}$  a partir de  $S(\mathbf{y}) = (1, \alpha^2)$  el podem fer buscant la síndrome a la taula anterior i obtenim  $\mathbf{e} = 00\alpha 00$ .

## Capítol 8

# Codis Polinomials

D'ara en endavant, les paraules d'un codi les designarem amb les lletres  $\mathbf{u}, \mathbf{v}, \mathbf{w}, \dots$  i identificarem paraules  $\mathbf{u} = (u_0, \dots, u_{n-1}) \in \mathbb{F}^n$  de longitud  $n$  amb polinomis  $u(x) = u_0 + u_1x + \dots + u_{n-1}x^{n-1}$  de grau com a molt  $n - 1$  amb coeficients a  $\mathbb{F}$ .  $\mathbb{F}[x]_n$  denotarà els polinomis a coeficients dins  $\mathbb{F}$  de grau menor que  $n$ . Estem identificant  $\mathbb{F}^n$  amb  $\mathbb{F}[x]_n$ .

**Definició 8.0.8.** Si  $g(x) = g_0 + g_1x + \dots + g_rx^r \in \mathbb{F}[x]$  és un polinomi de grau  $r$  (és a dir,  $g_r \neq 0$ ) amb terme independent no nul (és a dir,  $g(0) = g_0 \neq 0$ ) i  $n > r$ , el codi polinomial generat per  $g$  de longitud  $n$  és, per definició

$$C_{g,n} := \{u(x) \in \mathbb{F}[x]_n \mid g(x) \text{ divideix } u(x)\}.$$

A vegades la longitud del codi queda clara pel context i ometem la  $n$  del subíndex. Si  $u(x) \in C_{g,n}$  llavors  $u(x) = m(x)g(x)$  on  $m$  denota el quocient de  $u(x)$  per  $g(x)$ , que ha de ser de grau menor que  $n - r$ . Així

$$C_{g,n} := \{m(x)g(x) \mid m(x) \in \mathbb{F}[x]_{n-r}\}.$$

*Observació 8.0.9.*  $C_{g,n}$  és un codi lineal.

*Demostració.* Si  $u_1(x), u_2(x) \in C_{g,n}$  llavors  $u_i(x) = m_i(x)g(x)$  per a certs  $m_i(x) \in \mathbb{F}[x]_{n-r}$ . Si  $\lambda_1, \lambda_2 \in \mathbb{F}$  llavors  $\lambda_1(m_1(x)g(x)) + \lambda_2(m_2(x)g(x)) = (\lambda_1m_1(x) + \lambda_2m_2(x))g(x)$ , i per tant  $\lambda_1(m_1(x)g(x)) + \lambda_2(m_2(x)g(x)) \in C_{g,n}$ .  $\square$

### 8.1 Ràfegues d'errors

Una de les gràcies dels codis polinomials és la seva utilitat per a detectar ràfegues d'errors. Una ràfega d'errors és, essencialment, una colla d'errors en posicions consecutives. Més precisament, és un vector de la forma

$$(0, \dots, 0, e_i, \dots, e_j, 0, \dots, 0),$$

amb  $e_i \neq 0$  i  $e_j \neq 0$  (les posicions intermèdies poden ser nul·les). La longitud de la ràfega anterior és  $j - i + 1$  (el nombre de posicions que hi ha entre  $i$  i  $j$  comptant aquestes).

Si fem notació polinomial i  $u(x) = u_i x^i + \dots + u_j x^j$  amb  $u_i, u_j \neq 0$ ,  $u(x)$  és una ràfega de longitud  $j - i + 1$ .

**Proposició 8.1.1.** *Tot codi polinomial amb polinomi generador de grau  $r$  detecta totes les ràfegues d'errors de longitud com a molt  $r$ .*

*Demostració.* Veiem primer que tota paraula del codi diferent de 0 és una ràfega de longitud més gran que  $r$ . Tota paraula  $u(x) \neq 0$  del codi és de la forma  $u(x) = m(x)g(x)$  per un cert  $m(x) = m_s x^s + \dots + m_t x^t$  amb  $m_s \neq 0, m_t \neq 0$ . Així  $u(x) = (m_s x^s + \dots + m_t x^t)(g_0 + \dots + g_r x^r) = m_s g_0 x^s + \dots + m_t g_r x^{r+t}$  que, vist com a ràfega, té longitud  $r + 1 + (t - s) > r$ , ja que  $m_s g_0 \neq 0, m_t g_r \neq 0$ .

Si hem enviat  $u$  i hem rebut  $v(x) = u(x) + e(x)$  i s'ha produït una ràfega d'errors  $e(x) \neq 0$  de longitud com a molt  $r$ , com que  $e(x)$  no és del codi llavors  $v(x)$  tampoc serà del codi i per tant detectarem l'error. □

## 8.2 Matrius generadores i de control

### 8.2.1 a partir de $g(x)$

Com que

$$C_{g,n} = \{m(x)g(x) \mid m(x) \in \mathbb{F}[x]_{n-r}\},$$

els polinomis  $g(x), xg(x), x^2g(x), \dots, x^{n-r-1}g(x)$  formen un sistema generador de  $C_{g,n}$ . Si els posem en fila, obtenim la matriu següent:

$$\begin{pmatrix} g_0 & \cdots & g_r & 0 & \cdots & 0 \\ 0 & g_0 & \cdots & g_r & \cdots & 0 \\ \vdots & \ddots & \ddots & & \ddots & \vdots \\ 0 & \cdots & 0 & g_0 & \cdots & g_r \end{pmatrix} \quad (8.2.1)$$

El menor de mides  $(n - r) \times (n - r)$  situat a l'esquerra de la matriu(8.2.1) és

$$\begin{pmatrix} g_0 & g_1 & \cdots & \\ 0 & g_0 & \cdots & \\ \vdots & \ddots & \ddots & \\ 0 & \cdots & 0 & g_0 \end{pmatrix},$$

que té determinant no nul perquè  $g_0 \neq 0$ . Això ens diu que (8.2.1) és de rang màxim i per tant és una matriu generadora del codi. A més, aquest codi admet una codificació sistemàtica. De la mateixa manera, si observem el menor situat a la dreta també té determinant no nul ( $g_r \neq 0$ ), per tant els codis polinomials admeten codificacions sistemàtiques tant a la dreta com a l'esquerra.

Això també ens diu que la dimensió d'aquest codi és  $k = n - r$ . Dit d'una altra manera: el grau del polinomi generador coincideix amb la codimensió.

No és difícil comprovar que la codificació consistent multiplicar per (8.2.1) coincideix amb la codificació següent, consistent en multiplicar  $m(x)$  per  $g(x)$ :

$$\begin{aligned} \mathbb{F}[x]_k &\rightarrow \mathbb{F}[x]_n \\ m(x) &\mapsto u(x) = m(x)g(x). \end{aligned}$$

### 8.2.2 sistemàtica

Ja hem vist que  $C_{g,n}$  admet codificacions sistemàtiques a la dreta i l'esquerra. La codificació sistemàtica a la dreta és la següent:

$$\begin{aligned} \mathbb{F}[x]_k &\rightarrow \mathbb{F}[x]_n \\ m(x) &\mapsto u(x) = m(x)x^r + a(x), \end{aligned}$$

on,  $a(x)$  és el reste de dividir  $-m(x)x^r$  per  $g(x)$ , ja que  $m(x)x^r + a(x) \equiv 0 \pmod{g(x)}$  per ser  $u \in C_{g,n}$ .

Si notem per  $a_i(x)$  el reste de dividir  $-x^{i+r}$  per  $g$ , aquesta codificació envia  $x^i$  a  $x^{i+r} + a_i(x)$ , i la matriu generadora d'aquesta codificació és, per tant,

$$(A \mid I_k) \quad (8.2.2)$$

on  $A$  és la matriu que conté els  $a_i$  posats en fila. Com que  $a_i$  és congruent amb  $-x^{r+i}$  mòdul  $g$ , (8.2.2) podem notar-la així:

$$\left( \begin{array}{ccc|c} \text{---} & -x^r \text{ mod } g & \text{---} & \\ \vdots & \vdots & \vdots & \\ \text{---} & -x^{n-1} \text{ mod } g & \text{---} & I_k \end{array} \right) \quad (8.2.3)$$

Aplicant la fórmula (7.3.6) obtenim la següent matriu de control:

$$H = \left( \begin{array}{c|ccc} I_k & & & \\ \hline & x^r \text{ mod } g & \dots & x^{n-1} \text{ mod } g \\ & | & \dots & | \\ & & & \end{array} \right) \quad (8.2.4)$$

*Exemple 8.2.1.*  $\mathbb{F} = \mathbb{F}_2$  i  $g = x^3 + x + 1$ . En el codi generat per  $g$  de longitud 7,  $C_{g,7}$ , la matriu corresponent a codificar multiplicat per  $g$  és

$$G_1 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Si calculem  $x^3, x^4, x^5, x^6 \pmod{g}$  ens va donant  $1 + x, x + x^2, x^2 + x^3 \equiv 1 + x + x^2, x + x^2 + x^3 \equiv 1 + x^2$ , i la matriu generadora sistemàtica a la dreta és

$$G_2 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Usant (7.3.6) tenim que

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

és una matriu de control. És fàcil comprovar que el mínim nombre de columnes dependents és 3 i per tant 3 és la distància mínima.

Si  $m = (1, 0, 0, 1)$  i usem la primera codificació,  $m(x) = 1 + x^3$ ,  $u(x) = m(x)g(x) = (1 + x^3)(1 + x + x^3) = 1 + x + x^4 + x^6$ . Obtindriem el mateix multiplicant per  $G_1$ .

### 8.2.3 a partir del zeros de $g(x)$

Un cas important és quan el polinomi generador  $g(x)$  factoritza linealment sense zeros múltiples, és a dir, quan

$$g(x) = (x - \alpha_1) \cdots (x - \alpha_r)$$

amb  $\alpha_1, \dots, \alpha_r$  diferents dos a dos. En aquest cas  $g(x)$  divideix  $u(x)$  si  $u(\alpha_i) = 0$  per cada  $i$ . Així

$$C_{g,n} = \{u(x) \in \mathbb{F}[x]_n \mid u(\alpha_i) = 0, i = 1 \dots r\},$$

i com que  $u(\alpha) = u_0 + u_1\alpha + \dots + u_{n-1}\alpha^{n-1}$  resulta que

$$H = \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha_r & \alpha_r^2 & \cdots & \alpha_r^{n-1} \end{pmatrix} \quad (8.2.5)$$

és una nova matriu de control de  $C_{g,n}$ . Calcular la síndrome  $s = S(v)$  amb aquesta matriu és el mateix que substituir els  $\alpha_i$  en  $v(x)$ : si  $s = (s_0, \dots, s_{r-1})$  llavors  $s_i = v(\alpha_{i+1})$ . Aquesta matriu de control és la que utilitzarem en els codis de Reed-Solomon per a corregir errors.

## 8.3 Codis Cíclics

Si  $\mathbf{u} = (u_0, \dots, u_{n-1}) \in \mathbb{F}^n$ , el desplaçament cíclic de  $\mathbf{u}$  és

$$\sigma(\mathbf{u}) = (u_{n-1}, u_0, u_1, \dots, u_{n-2}).$$

**Definició 8.3.1.** Un codi cíclic és un codi lineal tancat per desplaçament cíclic, i.e, un codi lineal que a més verifica la propietat següent:

$$\text{Si } \mathbf{u} \in \mathcal{C} \text{ llavors } \sigma(\mathbf{u}) \in \mathcal{C}.$$

Observem que un codi cíclic  $\mathcal{C}$  és tancat per desplaçament cíclic invers; i.e., si  $\mathbf{u} = (u_0, \dots, u_{n-1}) \in \mathcal{C}$ , llavors  $(u_1, \dots, u_{n-2}, u_{n-1}, u_0) \in \mathcal{C}$ . Això és degut a que un desplaçament cíclic invers coincideix a fer  $n - 1$  desplaçaments cíclics:  $(u_1, \dots, u_{n-2}, u_{n-1}, u_0) = \sigma^{(n-1)}(\mathbf{u})$ .

Si fem servir la notació polinomial, i  $u(x) \in \mathbb{F}[x]_n$  el desplaçament cíclic de  $u(x)$  es calcula així:

$$\begin{aligned} \sigma(u(x)) &= \sigma(u_0 + \dots + u_{n-1}x^{n-1}) = u_0x + \dots + u_{n-2}x^{n-1} + u_{n-1} = \\ &u_0x + \dots + u_{n-2}x^{n-1} + u_{n-1}x^n - u_{n-1}x^n + u_{n-1} = xg(x) - u_{n-1}(x^n - 1). \end{aligned}$$

És a dir:

$$\sigma(u(x)) = xg(x) - u_{n-1}(x^n - 1).$$

Anem a veure que els codis cíclics són un cas particular de codis polinomials. Més precisament veurem que els codis cíclics són precisament els codis polinomials tals que el seu polinomi generador  $g(x)$  satisfà  $g(x) \mid x^n - 1$ , on  $n$  és la longitud del codi.

**Proposició 8.3.2.** *Sigui  $\mathcal{C} \subseteq \mathbb{F}^n$  un codi lineal. Llavors  $\mathcal{C}$  és cíclic si i  $\mathcal{C} = C_{g,n}$  per algun polinomi  $g(x)$  amb terme independent no nul i tal que  $g(x) \mid x^n - 1$ .*

*Demostració.* Primer veiem que un codi de la forma  $C_{g,n}$  amb  $g(x) \mid x^n - 1$  és cíclic. Si  $u(x) \in C_{g,n}$ , llavors  $\sigma(u(x)) = xu(x) - u_{n-1}(x^n - 1) \in C_{g,n}$ , ja que  $g(x)$  divideix  $u(x)$  i  $x^n - 1$  i per tant  $g(x) \mid xu(x) - u_{n-1}(x^n - 1)$ .

Suposem ara que  $\mathcal{C}$  és cíclic. Veiem primer que  $\mathcal{C}$  és polinomial. Prenem  $0 \neq g(x) \in \mathcal{C}$  de grau mínim i sigui  $r$  el seu grau.  $g(0) \neq 0$ , altrament, aplicant un desplaçament cíclic invers tindríem un polinomi de grau menor que  $r$ . Per ser  $\mathcal{C}$  cíclic també  $xg(x), \dots, x^{n-r-1}g(x) \in \mathcal{C}$  i per linealitat  $C_{g,n} = \{m(x)g(x) \mid m(x) \in \mathbb{F}[x]_{n-r}\} \subseteq \mathcal{C}$ . Per tal de veure  $C_{g,n} = \mathcal{C}$  només cal veure  $\mathcal{C} \subseteq C_{g,n}$ . Sigui  $u(x) \in \mathcal{C}$  qualsevol. Si fem la divisió Euclídia de  $u(x)$  per  $g(x)$  obtenim una expressió  $u(x) = q(x)g(x) + r(x)$  amb  $q(x)$  de grau menor que  $n - r$  i  $r(x)$  de grau menor que  $r$ . Com que tant  $u(x)$  com  $q(x)g(x)$  estan a  $\mathcal{C}$  (aquest últim perquè està a  $C_{g,n}$ ) també  $r(x)$  ha de ser de  $\mathcal{C}$ . Però com que  $r(x)$  és de grau menor que  $r$  ha de ser zero. És a dir,  $u(x) \in C_{g,n}$ .

Ara només queda veure que  $g(x)$  divideix  $x^n - 1$ . Sigui  $u(x) \in C_{g,n}$  tal que  $u_{n-1} \neq 0$ . Això sempre es pot aconseguir per desplaçament cíclic. Com que  $\sigma(u(x)) = xu(x) - u_{n-1}(x^n - 1) \in C_{g,n}$  llavors  $g(x)$  divideix  $xu(x) - u_{n-1}(x^n - 1)$ . Com que  $g(x)$  també divideix  $u(x)$  arribem a la conclusió que  $g(x)$  divideix  $u_{n-1}(x^n - 1)$ . Com que  $u_{n-1} \neq 0$  llavors  $g(x)$  divideix  $x^n - 1$ .  $\square$

### 8.3.1 Polinomi de control

Sigui  $\mathcal{C}$  és un codi cíclic amb polinomi generador  $g(x)$  monic. com que  $g(x)$  divideix  $x^n - 1$ , el quocient  $x^n - 1$  per  $g(x)$  el denotarem per  $h(x)$ . És a dir,  $h(x)$  és el polinomi monic tal que  $g(x)h(x) = x^n - 1$ , i rep el nom de polinomi de control. Com que  $h(x)$  té grau  $k$ , és de la forma:

$$h(x) = h_0 + h_1x + \dots + h_kx^k.$$

Ara veurem que el polinomi de control ens dona una matriu de control.

**Proposició 8.3.3.** Si  $\mathcal{C} \subseteq \mathbb{F}^n$  és un codi cíclic amb polinomi de control  $h(x)$ , llavors

$$H = \begin{pmatrix} 0 & \cdots & 0 & 0 & h_k & \cdots & h_0 \\ 0 & \cdots & 0 & h_k & \cdots & h_0 & 0 \\ \vdots & & & & & & \vdots \\ h_k & \cdots & h_0 & 0 & 0 & \cdots & 0 \end{pmatrix} \quad (8.3.1)$$

és una matriu de control de  $\mathcal{C}$ .

*Demostració.* Obviament  $H$  té les dimensions adequades i rang màxim, perquè tant  $h_0$  com  $h_k$  són no nuls. Només cal veure que  $GH^t = 0$ . Però

$$GH^t = \begin{pmatrix} g_0 & \cdots & g_r & 0 & \cdots & 0 \\ 0 & g_0 & \cdots & g_r & \cdots & 0 \\ \vdots & \ddots & \ddots & & \ddots & \vdots \\ 0 & \cdots & 0 & g_0 & \cdots & g_r \end{pmatrix} \begin{pmatrix} 0 & 0 & \cdots & 0 & h_k \\ \vdots & & & & \vdots \\ 0 & h_h & & & h_1 \\ h_k & h_{k-1} & & & h_0 \\ \vdots & \vdots & & & \vdots \\ h_1 & h_0 & 0 & \cdots & 0 \\ h_0 & 0 & 0 & \cdots & 0 \end{pmatrix}. \quad (8.3.2)$$

Només cal adonar-se que el resultat de multiplicar la fila  $i$  de  $G$  per la columna  $j$  de  $H^t$  és el coeficient d'índex  $n + 1 - (i + j)$  del polinomi  $g(x)h(x)$ . Com que  $1 \leq i \leq k$  i  $1 \leq j \leq r$  això ens dona tots els coeficients de grau entre 1 i  $n - 1$  de  $g(x)h(x)$ . Com que  $g(x)h(x) = x^n - 1$  són tots igual a zero.  $\square$

## Capítol 9

# Codis de Reed-Solomon

### 9.1 Definició i propietats bàsiques

**Definició 9.1.1.** Siguin  $k < n$  i  $\beta \in \mathbb{F}$  d'ordre més gran o igual que  $n$ ,  $r = n - k$ . El codi de Reed-Solomon  $RS_\beta(n, k)$  és el codi polinomial de longitud  $n$  amb polinomi generador  $g(x) = (x - \beta)(x - \beta^2) \cdots (x - \beta^r)$ .

La matriu de control que farem servir per descodificar serà la matriu donada per (8.2.5), que en aquest cas es converteix en:

$$H = \begin{pmatrix} 1 & \beta & \beta^2 & \cdots & \beta^{n-1} \\ 1 & \beta^2 & \beta^4 & \cdots & \beta^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \beta^r & \beta^{2r} & \cdots & \beta^{r(n-1)} \end{pmatrix} \quad (9.1.1)$$

**Proposició 9.1.2.**  $RS_\beta(n, k)$  és MDS i per tant corregeix fins a  $\rho = \lfloor r/2 \rfloor$  errors.

*Demostració.* Hem de veure que cada  $r$  columnes de  $H$  són independents. Si  $0 \leq i_1 < i_2 < \cdots < i_r \leq n - 1$

$$\begin{vmatrix} \beta^{i_1} & \cdots & \beta^{i_r} \\ \beta^{2i_1} & \cdots & \beta^{2i_r} \\ \vdots & \vdots & \vdots \\ \beta^{ri_1} & \cdots & \beta^{ri_r} \end{vmatrix} = \beta^{i_1} \cdots \beta^{i_r} \begin{vmatrix} 1 & \cdots & 1 \\ \beta^{i_1} & \cdots & \beta^{i_r} \\ \vdots & \vdots & \vdots \\ \beta^{(r-1)i_1} & \cdots & \beta^{(r-1)i_r} \end{vmatrix}$$

i aquest últim determinant és el determinant de Vandermonde de  $\beta^{i_1}, \dots, \beta^{i_r}$ , que és no nul perquè tots aquests elements són diferent ja que l'ordre de  $\beta$  és més gran o igual a  $n$ .  $\square$

Normalment es pren  $r$  parell (o equivalentment  $\delta$  senar). En aquest cas tot és més senzill:  $\rho = r/2$  i  $r - \rho = \rho$ . això simplifica la forma de les matrius que intervenen al sistema de descodificació PGZ.

## 9.2 Polinomis localitzador i avaluador

Suposem que s'han produït  $t$  errors amb  $t \leq \rho$  i que l'error és  $e(x) = e_{i_1}x^{i_1} + e_{i_t}x^{i_t}$ . Si denotem per  $P = \{i_1, \dots, i_t\}$  el conjunt de les posicions d'error, podem expressar el polinomi  $e(x)$  així:  $e(x) = \sum_{i \in P} e_i x^i$ .

**Definició 9.2.1 (polinomis localitzador i avaluador).** Els polinomis localitzador  $\lambda$  i avaluador  $w$  de l'error  $e(x)$  són, respectivament:

$$\lambda(x) = \prod_{i \in P} (1 - \beta^i x) \quad (9.2.1)$$

$$\omega(x) = \sum_{i \in P} e_i \beta^i \prod_{\substack{j \in P \\ j \neq i}} (1 - \beta^j x) \quad (9.2.2)$$

Observem que el polinomi localitzador té grau  $t$  i el avaluador grau  $t - 1$  i que el terme independent de  $\lambda$  és 1. Els escriurem  $\lambda(x) = 1 + \lambda_1 x + \dots + \lambda_t x^t$  i  $\omega(x) = \omega_0 + \omega_1 x + \dots + \omega_{t-1} x^{t-1}$ .

El polinomi localitzador serveix per calcular les posicions d'error: els seus zeros són  $\{\beta^{-i} \mid i \in P\}$ . El polinomi avaluador serveix per calcular les magnituds  $e_i$  de l'error mitjançant la fórmula següent, coneguda com a fórmula de Forney.

**Proposició 9.2.2 (Fórmula de Forney).** Si  $i \in P$ ,

$$e_i = \frac{-\omega(\beta^{-i})}{\lambda'(\beta^{-i})} \quad (9.2.3)$$

*Demostració.* Si calculem la derivada  $\lambda'$  de  $\lambda$  obtenim

$$\lambda'(x) = \sum_{i \in P} -\beta^i \prod_{\substack{j \in P \\ j \neq i}} (1 - \beta^j x)$$

i substituint  $\beta^i$ ,

$$\lambda'(\beta^i) = -\beta^i \prod_{\substack{j \in P \\ j \neq i}} (1 - \beta^{j-i}).$$

Substituint  $\beta^i$  a  $\omega$ ,

$$\omega(\beta^i) = e_i \beta^i \prod_{\substack{j \in P \\ j \neq i}} (1 - \beta^{j-i}),$$

d'on surt (9.2.3). □

La síndrome  $s(x) = s_0 + s_1 x + \dots + s_{r-1} x^{r-1}$  d'un missatge rebut  $v(x)$  serà la que es calcula multiplicant per la matriu (9.1.1). També es pot calcular fent  $s_j = v(\beta^{j+1})$ . Com que  $v(x) = u(x) + e(x)$  i  $u(\beta^{j+1}) = 0$  resulta que  $s_j = v(\beta^{j+1}) = e(\beta^{j+1}) = \sum_{i \in P} e_i \beta^{i(j+1)}$  per  $j = 0, \dots, r - 1$ .

**Teorema 9.2.3 (Identitat fonamental).**

$$\omega(x) \equiv s(x)\lambda(x) \pmod{x^r} \quad (9.2.4)$$

*Demostració.*

$$\begin{aligned} \frac{\omega(x)}{\lambda(x)} &= \sum_{i \in P} \frac{e_i \beta^i}{1 - \beta^i x} = \sum_{i \in P} e_i \beta^i (1 + \beta^i x + (\beta^i x)^2 + (\beta^i x)^3 + \dots) = \\ &= \left( \sum_{i \in P} e_i \beta^i \right) + \left( \sum_{i \in P} e_i \beta^{2i} \right) x + \left( \sum_{i \in P} e_i \beta^{3i} \right) x^2 + \left( \sum_{i \in P} e_i \beta^{4i} \right) x^3 + \dots = \\ &= e(\beta) + e(\beta^2)x + e(\beta^3)x^2 + e(\beta^4)x^3 + \dots = \\ &= s(x) + \text{Termes de Grau Superior a } (r-1). \end{aligned}$$

Hem vist doncs que

$$\frac{\omega(x)}{\lambda(x)} = s(x) + x^r R(x),$$

on  $R(x)$  és una sèrie de potències. Multiplicant per  $\lambda(x)$  obtenim  $\omega(x) = s(x)\lambda(x) + x^r \lambda(x)R(x)$ , on ara  $\lambda(x)R(x)$  és un polinomi.  $\square$

## 9.3 El mètode PGZ millorat

El mètode PGZ (Peterson-Gorenstein-Zierler) és el primer que es va publicar per a la descodificació (correcció d'errors) dels codis de Reed-Solomon. Encara que quan el nombre d'errors a corregir és gran resulta poc eficient, per a pocs errors resulta factible, fins i tot, fent càlculs 'a mà'.

Si escrivim les equacions de (9.2.4) corresponents als monomis de graus  $t, \dots, r-1$  obtenim:

$$\begin{cases} 0 = s_0 \lambda_t + s_1 \lambda_{t-1} + \dots + s_{t-1} \lambda_1 + s_t \\ 0 = s_1 \lambda_t + s_2 \lambda_{t-1} + \dots + s_t \lambda_1 + s_{t+1} \\ \vdots \\ 0 = s_{r-t-1} \lambda_t + s_{r-t} \lambda_{t-1} + \dots + s_{r-2} \lambda_1 + s_{r-1}. \end{cases} \quad (9.3.1)$$

Si pensem (9.3.1) com un sistema lineal amb incògnites

$$(-\lambda_t, -\lambda_{t-1}, \dots, -\lambda_1),$$

la matriu d'aquest sistema és

$$\left( \begin{array}{cccc|c} s_0 & s_1 & \cdots & s_{t-1} & s_t \\ s_1 & s_2 & \cdots & s_t & s_{t+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ s_{t-1} & s_t & \cdots & s_{2t-2} & s_{2t-1} \\ \hline s_t & s_{t+1} & \cdots & s_{2t-1} & s_{2t} \\ \vdots & \vdots & & \vdots & \vdots \\ s_{r-t-1} & s_{r-t} & \cdots & s_{r-2} & s_{r-1} \end{array} \right) \quad (9.3.2)$$

Veurem que el sistema (9.3.2) és compatible i determinat. De fet el menor  $t \times t$  situat a l'extrem superior esquerra és no nul. El problema d'aquest sistema és que necessitem saber el nombre d'errors  $t$  per tal d'escriure'l. En lloc de (9.3.2) usarem la matriu  $S$  definida a continuació<sup>1</sup>, que no té aquest problema. La única diferència entre aquestes matrius és la forma. (9.3.2) té més files que columnes en general, al revés que (9.3.3); però ambdues tenen en comú la caixa de dimensions  $t \times (t + 1)$  situada a l'extrem superior esquerra :

$$S := \left( \begin{array}{cccc|c|cc} s_0 & s_1 & \cdots & s_{t-1} & s_t & \cdots & s_\rho \\ s_1 & s_2 & \cdots & s_t & s_{t+1} & \cdots & s_{\rho+1} \\ \vdots & \vdots & \cdots & \vdots & \vdots & & \vdots \\ s_{t-1} & s_t & \cdots & s_{2t-2} & s_{2t-1} & \cdots & s_{\rho+t-1} \\ \hline s_t & s_{t+1} & \cdots & s_{2t-1} & s_{2t} & \cdots & s_{\rho+t} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ s_{(r-\rho)-1} & s_{(r-\rho)} & \cdots & s_{(r-\rho)+t-2} & s_{(r-\rho)+t-1} & \cdots & s_{2\rho-1} \end{array} \right) \quad (9.3.3)$$

Denotem per  $S_t$  el menor  $t \times t$  de  $S$  situat a la part superior esquerra de (9.3.2) i (9.3.3):

$$S_t = \left( \begin{array}{cccc} s_0 & s_1 & \cdots & s_{t-1} \\ s_1 & s_2 & \cdots & s_t \\ \vdots & \vdots & \ddots & \vdots \\ s_{t-1} & s_t & \cdots & s_{2t-2} \end{array} \right) \quad (9.3.4)$$

**Teorema 9.3.1.** *La matriu  $S$  té rang  $t$  i el determinant de  $S_t$  és no nul.*

*Demostració.* Comencem veient que  $S_t$  té determinant no nul. Per fer això primer veurem que  $S_t$  és igual al producte matricial següent:

$$\left( \begin{array}{ccc} \beta^{i_1} & \cdots & \beta^{i_t} \\ \beta^{2i_1} & \cdots & \beta^{2i_t} \\ \vdots & \ddots & \vdots \\ \beta^{ti_1} & \cdots & \beta^{ti_t} \end{array} \right) \left( \begin{array}{ccc} e_{i_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & e_{i_t} \end{array} \right) \left( \begin{array}{ccc} 1 & \cdots & 1 \\ \beta^{i_1} & \cdots & \beta^{i_t} \\ \vdots & \ddots & \vdots \\ \beta^{(t-1)i_1} & \cdots & \beta^{(t-1)i_t} \end{array} \right)^t, \quad (9.3.5)$$

<sup>1</sup>en el cas en que  $r$  és parell  $r - \rho = r$ ,  $2\rho = r$  i la matriu té  $\rho$  files i  $\rho + 1$  columnes

on  $P = \{i_1, \dots, i_t\}$  són les posicions d'error. L'element situat a la fila  $u$  columna  $v$  de (9.3.5) és  $a_{u,v} = \sum_{i \in P} e_i \beta^{i(u+v-1)} = e(\beta^{u+v-1}) = s_{u+v-2}$ ; però  $s_{u+v-2}$  és precisament l'element situat a la fila  $u$ , columna  $v$  de  $S_t$ . Si calculem el determinant de  $S_t$  mitjançant (9.3.5) obtenim  $\det(S_t) = \det(V(\beta^{i_1}, \dots, \beta^{i_t}))^2 \prod_{i \in P} e_i \beta^i$ , que no s'anulla, doncs la matriu de Vandermonde  $V(\beta^{i_1}, \dots, \beta^{i_t})$  té determinant no nul. Ara veurem que tot menor  $A_l$  d'ordre  $l > t$  de  $S$  té determinant nul. Suposem que hem pres les files  $j_1, \dots, j_l$  i les columnes  $k_1, \dots, k_l$ . Primer veiem que  $A_l$  es pot expressar matricialment:

$$\begin{pmatrix} \beta^{j_1 i_1} & \dots & \beta^{j_1 i_t} \\ \beta^{j_2 i_1} & \dots & \beta^{j_2 i_t} \\ \vdots & \ddots & \vdots \\ \beta^{j_l i_1} & \dots & \beta^{j_l i_t} \end{pmatrix} \begin{pmatrix} e_{i_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & e_{i_t} \end{pmatrix} \begin{pmatrix} \beta^{(k_1-1)i_1} & \dots & \beta^{(k_1-1)i_t} \\ \beta^{(k_2-1)i_1} & \dots & \beta^{(k_2-1)i_t} \\ \vdots & \ddots & \vdots \\ \beta^{(k_l-1)i_1} & \dots & \beta^{(k_l-1)i_t} \end{pmatrix}^t, \quad (9.3.6)$$

L'element situat a la fila  $u$  columna  $v$  de (9.3.6) és  $a_{u,v} = \sum_{i \in P} e_i \beta^{i(j_u + k_v - 1)} = s_{k_u + k_v - 2}$ , que és precisament l'element situat a la fila  $u$ , columna  $v$  de  $A_l$ . Com que la matriu de l'esquerra de (9.3.6) té  $t$  columnes, el producte no pot tenir rang superior a  $t$ , i per tant el determinant de  $A_l$  s'anulla.  $\square$

Com que  $S_t$  és el menor superior esquerra de la matriu del sistema (9.3.2), aquest sistema és compatible i determinat. A més, per tal de calcular el polinomi localitzador n'hi ha prou amb resoldre el sistema següent:

$$\left( \begin{array}{cccc|c} s_0 & s_1 & \dots & s_{t-1} & s_t \\ s_1 & s_2 & \dots & s_t & s_{t+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ s_{t-1} & s_t & \dots & s_{2t-2} & s_{2t-1} \end{array} \right), \quad (9.3.7)$$

la solució del qual serà doncs:

$$(-\lambda_t, -\lambda_{t-1}, \dots, -\lambda_1).$$

Ara bé, com que  $S$  conté (9.3.7) a la part superior esquerra, quan fem Gauss-Jordan amb la matriu  $S$  (sense permutar columnes, no cal), pel Teorema 9.3.1 arribarem a una matriu del tipus

$$\left( \begin{array}{ccc|c|ccc} 1 & \dots & 0 & -\lambda_t & \dots & \dots \\ \vdots & \ddots & \vdots & \vdots & \dots & \dots \\ 0 & \dots & 1 & -\lambda_1 & \dots & \dots \\ \hline 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & \dots & 0 \end{array} \right), \quad (9.3.8)$$

d'on s'obté el nombre de zeros  $t$  i el polinomi localitzador tot d'una.

Aquest és el mètode PGZ (Peterson-Gorenstein-Zierler) millorat<sup>2</sup>: Primer calclem la síndrome  $s(x)$  mitjançant la fórmula  $s_i = v(\beta^{i+1})$  per  $i = 0 \dots r-1$ . La manera més eficaç de fer aquestes substitucions és mitjançant Ruffini. A continuació fem Gauss-Jordan a  $S$  fins arribar a una matriu de la forma (9.3.8). Amb això obtenim  $t$  i  $\lambda(x)$ . El polinomi avaluador es calcula a partir de la identitat fonamental (9.2.4). A continuació es calculen els zeros de  $\lambda(x)$  per tal d'obtenir les posicions d'error  $P$ . El mètode consisteix en provar successivament els valors  $\beta^{-i}$  per  $i = 0 \dots n-1$ . Finalment amb la fórmula de Forney (proposició 9.2.2) obtenim les magnituds dels errors.

## 9.4 El mètode de Suniyama o Euclides Estès

Un altre mètode és l'algorisme de Suniyama o Euclides Estès. si escrivim l'equació fonamental (9.2.4) com

$$\omega(x) = \lambda(x)s(x) + \mu(x)x^r \quad (9.4.1)$$

té una certa semblança a una identitat de Bézout pels polinomis  $s(x)$  i  $x^r$ , però  $\omega(x)$  no és el mcd de  $s(x)$  i  $x^r$ , que és 1 (0 no és un zero de  $\omega(x)$ ). El mètode de Suniyama consisteix a aplicar l'algorisme d'Euclides Estès per tal de calcular una identitat de Bézout, però aturant-se quan el reste sigui de grau menor que  $\rho$ . Veiem-ho amb una mica més de detall.

De la mateixa manera que l'algorisme d'Euclides, calcularem restes de divisions euclides, que denotarem per  $w_0(x)$ ,  $w_1(x)$ ,  $w_2(x)$ ,  $\dots$ ,  $w_j(x)$  definits de la manera següent:

$$\left\{ \begin{array}{l} \omega_0 = x^r \\ \omega_1 = s(x) \\ \omega_{i-1} = \omega_i q_i + \omega_{i+1}, \end{array} \right. \quad (9.4.2)$$

$\omega_{i+1}$  i  $q_i$  s'obtenen com el reste i quocient de dividir  $w_{i-1}$  per  $w_i$ . Els graus dels polinomis  $w_i$  decreixen (estrictament) durant tot el procés ja que comencen per  $\deg(w_0) = r$  i  $\deg(w_1) \leq r-1$ . A més  $\deg(q_i) = \deg(w_{i-1}) - \deg(w_i) > 0$ . El procés l'aturarem quan haguem obtingut  $w_j(x)$ , que serà el primer reste amb grau menor que  $\rho$ . Com que  $s(x)$  i  $x^r$  són primers entre si, sempre hi podem arribar (podem arribar a grau zero, el grau del mcd). També calcularem  $\lambda_i(x)$  per  $i = 0 \dots j$  mitjançant:

$$\left\{ \begin{array}{l} \lambda_0 = 0 \\ \lambda_1 = 1 \\ \lambda_{i+1} = \lambda_{i-1} - \lambda_i q_i. \end{array} \right. \quad (9.4.3)$$

<sup>2</sup>El sistema PGZ que es troba als llibres de text, primer calcula el nombre d'errors  $t$  calculant el determinants de les matrius  $l \times l$  situades a l'extrem superior esquerra de  $S$  començant per  $l = \rho$  i anant-lo disminuint fins trobar el primer amb determinant no nul. Llavors es resol el sistema (9.3.2) sense les equacions que sobren, és a dir, amb només les  $t$  primeres files.

Els graus dels polinomis  $\lambda_i$  creixen (estrictament), ja que  $\deg(\lambda_{i+1}) = \deg(\lambda_i) + \deg(q_i)$  i aquest últim té grau no nul. Els polinomis  $\mu_i(x)$  per  $i = 0 \dots j$  que definim a continuació són auxiliars per a la demostració del Teorema 9.4.2; a l'hora de calcular els polinomis localitzador i avaluador no els necessitarem i per tant no caldrà calcular-los:

$$\begin{cases} \mu_0 = 1 \\ \mu_1 = 0 \\ \mu_{i+1} = \mu_{i-1} - \mu_i q_i \end{cases}$$

**Lema 9.4.1.** *Els polinomis definits anteriorment satisfan:*

1.  $\lambda_i s + \mu_i x^r = w_i$ , per  $i = 0 \dots j$ .
2.  $\lambda_{i+1} \mu_i - \mu_{i+1} \lambda_i = (-1)^i$ , per  $i = 0 \dots j - 1$ .
3.  $\deg(\lambda_{i+1}) + \deg(\omega_i) = r$ , per  $i = 0 \dots j - 1$ .

*Demostració.* Es fan totes tres per inducció. Comencem per (1). Els casos  $i = 0$  i  $i = 1$  són clars. Suposem que l'equació val per  $i - 1$  i  $i$ . Llavors  $\lambda_{i+1} s + \mu_{i+1} x^r = (\lambda_{i-1} - \lambda_i q_i) s + (\mu_{i-1} - \mu_i q_i) x^r = \lambda_{i-1} s + \mu_{i-1} x^r - q_i (\lambda_i s + \mu_i x^r) = \omega_{i-1} - q_i \omega_i = \omega_{i+1}$ .

Per  $i = 0$  (2) també és clar; i si val per a  $i - 1$  llavors  $\lambda_{i+1} \mu_i - \mu_{i+1} \lambda_i = (\lambda_{i-1} - \lambda_i q_i) \mu_i - (\mu_{i-1} - \mu_i q_i) \lambda_i = \lambda_{i-1} \mu_i - \mu_{i-1} \lambda_i - (-1)^{i-1} = (-1)^i$ .

Veiem ara (3). Per  $i = 1$  és clar. El pas d'inducció és immediat tenint en compte que  $\deg(\lambda_{i+1}) = \deg(\lambda_i) + \deg(q_i)$  i  $\deg(\omega_{i-1}) = \deg(\omega_i) + \deg(q_i)$ .  $\square$

**Teorema 9.4.2.** *Si  $\lambda_j(x)$  i  $\omega_j(x)$  són els polinomis obtinguts mitjançant l'algorisme descrit anteriorment, llavors*

$$\lambda(x) = a \lambda_j(x) \text{ i } \omega(x) = a \omega_j(x)$$

per a un cert  $a \in \mathbb{F}^*$ .

*Demostració.* Multiplicant (9.4.1) per  $\lambda_j(x)$ , multiplicant l'equació (1) del Lema 9.4.1 per  $\lambda(x)$  i restant obtenim

$$\lambda_j \omega - \lambda \omega_j = (\lambda_j \mu - \lambda \mu_j) x^r. \quad (9.4.4)$$

Ara be, per (3) del Lema 9.4.1 i com que  $\deg(w_{j-1}) \geq \rho$ ,  $\deg(\lambda_j) \leq r - \rho$  i per tant  $\deg(\lambda_j \omega) < r - \rho + t \leq r$ . Com que  $\deg(\lambda \omega_j) < t + \rho \leq r$ , el polinomi  $\lambda_j \omega - \lambda \omega_j$  té grau menor que  $r$  i per tant és el polinomi nul. De

$$\lambda_j(x) \omega(x) = \lambda(x) \omega_j(x)$$

i com que  $\lambda(x)$  i  $\omega(x)$  són primers entre si, deduïm  $\lambda(x) \mid \lambda_j(x)$ . Així  $\lambda_j(x) = a(x) \lambda(x)$  i  $\omega_j(x) = a(x) \omega(x)$  per a un cert polinomi  $a(x)$ . Per 9.4.4 també

$$\lambda_j(x) \mu(x) = \lambda(x) \mu_j(x).$$

Ara bé, per (2) del Lema 9.4.1  $\lambda_j$  i  $\mu_j$  són primers entre si i per tant  $\lambda_j(x)$  divideix  $\lambda(x)$ . Tot plegat  $\lambda$  i  $\lambda_j$  tenen el mateix grau i per tant el polinomi  $a(x)$  de fet és una constant.  $\square$

El mètode de Suniyama (o Euclides Estès) és el següent. Primer calculem la síndrome  $s(x)$  mitjançant la fórmula  $s_i = v(\beta^{i+1})$  per  $i = 0 \dots r-1$ . La manera més eficaç de fer aquestes substitucions és mitjançant Ruffini. A continuació obtenim les successions  $w_i$  i  $\lambda_i$  per  $i = 1 \dots j$  definides a (9.4.2) i (9.4.3), on  $j$  és el primer índex on el grau de  $w_j$  és (estrictament) menor que  $\rho$ . Tenint en compte que el terme independent del polinomi localitzador és 1 podem calcular la constant  $a$  del Teorema 9.4.2 i per tant obtenim els polinomis localitzador i avaluador. A continuació es calculen els zeros de  $\lambda$  per tal d'obtenir les posicions d'error  $P$ . El mètode consisteix en provar successivament els valors  $\beta^{-i}$  per  $i = 0 \dots n-1$ . Finalment amb la fórmula de Forney (9.2.3) obtenim les magnituds dels errors.