

Coduri detectoare si corectoare de erori. Coduri Hamming

În domeniul compresiei de date (considerata în teoria codarii ca si "codarea sursei") codarea s-a facut în scopul reducerii dimensiunii reprezentarii, presupunând un canal de comunicatie ideal, fara pierderi. În cele ce urmeaza vom aborda pe scurt o problema diferita, si anume codarea în scopul detectiei si eventual corectiei erorilor ce pot apare pe un canal de comunicatie real, cu erori (considerata în teoria codarii ca si "codarea canalului").

În acest al doilea caz este evident ca, pentru a obtine detectie si corectie de erori, trebuie sa încarcam suplimentar fluxul cu biti dedicati acestui lucru, având de a face cu o crestere a dimensiunii reprezentarii (trebuie platit un pret...). În timp ce la compresia datelor se elimina cât mai mult posibil redundanta, codurile corectoare adauga acel nivel de redundanta necesar pentru a transmite datele în mod eficient si cu fidelitate pe un canal cu zgomot (cu erori).

1. Distanța Hamming

Pe lângă distantele prezentate într-un material anterior, o distanta de un interes deosebit în problema noastra este distanta Hamming, numita astfel dupa Richard Hamming, cel care a introdus-o în 1950. **Distanța Hamming** între doi vectori de dimensiuni egale este data de **numarul de pozitii în care acestia difera**. Ea masoara astfel numarul de schimbari care trebuie facute într-un vector pentru a îl obtine pe celalalt, sau reformulat numarul de *erori* care transforma un vector în celalalt.

Exemple:

vector 1	codare	126359	01101011
vector 2	notate	226389	01001110
distanța Hamming	3	2	3

Desi definirea este generala, în cele ce urmeaza vom considera doar cazul vectorilor cu elemente binare, fiind vorba de fluxuri de biti transmise pe canalul de comunicatie. În acest caz distanta este data de numarul de 1 din rezultatul obtinut prin XOR.

Pentru început vom considera doar cazul simplu al **erorilor singulare** - adica avem un singur bit eronat.

2. Un exemplu simplu de detectie de erori singulare

Sa consideram urmatoarea solutie simpla de codare – fiecare bit este codat prin repetarea sa de doua ori. Introducem astfel în mod evident o redundanta care însa ne va permite sa detectam erori singulare.

Daca la receptie obtinem coduri 00 respectiv 11 am receptionat corect 0 respectiv 1 iar daca obtinem 01 sau 10 am detectat o eroare singulara (fara a putea decide însa nimic în sensul corectiei ei).

Exemplu:

mesaj initial:	0.1.0.0.1.0.1.1.0.1
mesaj codat:	00.11.00.00.11.00.11.11.00.11
mesaj receptionat cu eroare:	00.11.00.00. 10 .00.11.11.00.11 => detectie de eroare

Remarcam faptul ca între oricare doua ”cuvinte de cod” valide avem o distanta Hamming 2.

3. Un exemplu simplu de corectie de erori singulare

Sa consideram urmatoarea solutie simpla de codare – fiecare bit este codat prin repetarea sa de trei ori. Introducem astfel în mod evident o redundanta si mai mare care însa ne va permite sa detectam si sa corectam erori singulare.

Daca la receptie obtinem coduri 000 respectiv 111 am receptionat corect 0 respectiv 1. Daca obtinem 001 sau 010 sau 100 am detectat o eroare singulara si putem **corecta** spre 000 (deci am receptionat 0) iar daca obtinem 011 sau 101 sau 110 am detectat o eroare singulara si putem **corecta** spre 111 (deci am receptionat 1).

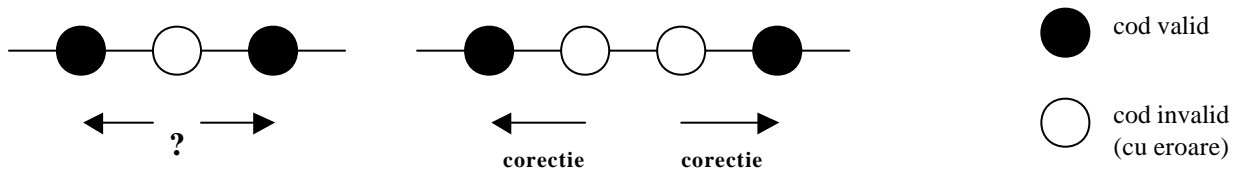
Exemplu:

mesaj initial:	0.1.0.0.1.0.1.1.0.1
mesaj codat:	000.111.000.000.111.000.111.111.000.111
mesaj receptionat cu eroare:	000. 101 .000.000.111.000.111.111. 001 .111
mesaj corectat:	000. 111 .000.000.111.000.111.111. 000 .111

=> **detectie** si corectie de erori singulare

Remarcam faptul ca între oricare doua ”cuvinte de cod” valide avem o distanta Hamming 3.

Din exemplele anterioare constatam ca, daca distanta Hamming între oricare cuvinte de cod valide este 2, putem detecta erori singulare dar nu le putem corecta - nu stim care cod valid aflat la distanta 1 este cel corect. Daca distanta Hamming între oricare cuvinte de cod valide este 3 putem detecta erori singulare si putem corecta înspre codul valid aflat la distanta 1 (remarcam faptul ca, în acest caz, putem detecta chiar erori duble ! – fara însa a le putea corecta).



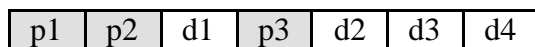
Daca scopul este doar de a detecta erori se pot folosi si alte solutii cum ar fi simpli biti de paritate, sume de control, coduri ciclice de tip CRC, functii hash criptografice, etc.

4. Codul Hamming (7,4)

Unul din cele mai cunoscute coduri detectoare si corectare de erori singulare este codul numit Hamming (7,4). Acesta notatie indica faptul ca avem un cod de 7 biti din care 4 sunt biti de date independenti (restul fiind biti redundanti, reprezentând paritatea a diferite combinatii a bitilor de date). Codul contine deci 4 biti de date d_1, d_2, d_3, d_4 si 3 biti de paritate p_1, p_2, p_3 . Bitii de paritate sunt calculati astfel:

- p_1 sumeaza d_1, d_2, d_4
- p_2 sumeaza d_1, d_3, d_4
- p_3 sumeaza d_2, d_3, d_4

iar organizarea bitilor în cuvântul de cod este urmatoarea:



De obicei se definesc doua matrici în legatura cu acest cod: matricea generatoare de cod G si matricea de detectie a paritatii H

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

Se constata în G ca liniile 1,2 si 4 calculeaza sumele aferente paritatilor respective iar liniile 3,5,6,7 simplu copiaza bitii de date. În H cele 3 linii calculeaza paritatile corespunzatoare.

La codare: se determina cuvântul de cod calculând paritatile corespunzatoare (se poate utiliza atât paritatea para cât si paritatea impara - în exemplele urmatoare vom folosi paritatea impara).

La decodare: se calculeaza paritatile corespunzatoare si se verifica cu cele corecte (în fapt se sumeaza si cu paritatile corecte si se verifica sa rezulte 0).

Exemplu de codare-decodare pentru codul Hamming (7,4)

La codare: fie cuvântul de cod 1001. Calculam:

$$p1 = 1+0+1 = 0$$

$$p2 = 1+0+1 = 0$$

$$p3 = 0+0+1 = 1$$

rezultând codul Hamming 0011001 (am reprezentat subliniat bitii de paritate).

La decodare (cazul 1): presupunem ca am receptionat 0011001 (deci fara erori)

Reconstituim bitii de date: 1 0 0 1

Verificam paritatile:

$$p1 + d1 + d2 + d4 = 0 + 1 + 0 + 1 = 0$$

$$p2 + d1 + d3 + d4 = 0 + 1 + 0 + 1 = 0$$

$$p3 + d2 + d3 + d4 = 1 + 0 + 0 + 1 = 0$$

Cum toate aceste valori sunt 0 rezulta ca nu am avut eroare.

La decodare (cazul 2): presupunem ca am receptionat 0011011 (deci cu o eroare în zona de date)

Reconstituim bitii de date: 1 0 1 1

Verificam paritatile:

$$p1 + d1 + d2 + d4 = 0 + 1 + 0 + 1 = 0$$

$$p2 + d1 + d3 + d4 = 0 + 1 + 1 + 1 = 1$$

$$p3 + d2 + d3 + d4 = 1 + 0 + 1 + 1 = 1$$

Cum aceste valori nu sunt toate 0 rezulta ca am avut eroare (deci am realizat **detectie** de eroare). În plus codul format din pozitiile eronate (aferent p3p2p1) având valoarea 110 indica bitul 6 ca fiind eronat deci al treilea bit de date trebuie corectat (deci am realizat **corectie** de eroare). Deci, bitii de date corectati sunt 1 0 0 1.

Pozitionarea initiala aparent ciudata a bitilor de paritate în cadrul codului Hamming este necesara pentru a obtine simplu, direct indicele bitului eronat.

La decodare (cazul 3): presupunem ca am receptionat 0111001 (deci cu o eroare în zona de paritate)

Reconstituim bitii de date: 1 0 0 1

Verificam paritatile:

$$p1 + d1 + d2 + d4 = 0 + 1 + 0 + 1 = 0$$

$$p2 + d1 + d3 + d4 = 1 + 1 + 0 + 1 = 1$$

$$p3 + d2 + d3 + d4 = 1 + 0 + 0 + 1 = 0$$

Cum aceste valori nu sunt toate 0 rezulta ca am avut eroare (deci am realizat **detectie** de eroare). În plus codul format din pozitiile eronate (aferent p3p2p1) având valoarea 010 indica bitul 2 ca fiind eronat deci al doilea bit de paritate era eronat iar bitii de date erau corecti (deci am realizat **corectie** de eroare).

Codul Hamming (7,4) descris anterior are distanta Hamming între oricare cuvintele de cod valide minim 3 si deci poate **detecta si corecta erori singulare** sau poate doar detecta erori duble (fara a putea corecta nici un fel de erori).

Calcululele anterioare se pot face si în forma matriciala considerând la codare înmultirea matricii G cu vectorul coloana al bitilor de date iar la verificarea paritatii înmultirea matricii H cu vectorul coloana al codului Hamming.

5. Coduri Hamming generale

Codul Hamming descris anterior a fost generalizat pentru orice numar de biti. Algoritmul general pentru proiectarea unui cod Hamming **corector de erori singulare** ("single error correcting" - SEC) este urmatorul:

- Numerotam bitii începând cu 1: 1, 2, 3, 4, etc.
- Fiecare pozitie care este putere a lui 2 reprezinta **bit de paritate** (pozitiile care au un singur bit în reprezentarea binara)
- Celelalte pozitii reprezinta **biti de date** (pozitiile care au mai mult de un bit pe 1 în reprezentarea binara).
- Fiecare bit de paritate acopera biti astfel: bitul k de paritate acopera acele pozitii care au bitul k cel mai semnificativ pe 1.

Deși modul de calcul al parității (pare sau impar) nu este esențial, de regulă însă se folosește paritatea impară (paritatea e 1 dacă numărul de biți de 1 este impar).

Algoritmul general poate fi înțeles mai bine din figura următoare:

poziție	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	...	
cod	p1	p2	d1	p3	d2	d3	d4	p4	d5	d6	d7	d8	d9	d10	d11	p5	d12	d13	...	
componenta biți paritate	p1	x		x		x		x		x		x		x		x		x		...
	p2		x	x			x	x			x	x			x	x			x	...
	p3				x	x	x	x					x	x	x	x				...
	p4								x	x	x	x	x	x	x	x				...
	p5																x	x	x	...

S-au reprezentat doar 5 biți de paritate și 13 biți de date având de a face cu codul care se notează Hamming(18,13).

Dacă avem m biți de paritate putem acoperi până la $2^m - 1$ biți. Dacă scădem cei m biți de paritate rămân $2^m - m - 1$ biți care pot fi folosiți pentru date. În funcție de valoarea lui m avem următoarele coduri Hamming:

Biți de paritate	Total biți	Biți date	Nume cod	Eficiența utilizare biți date
2	3	1	Hamming(3,1)	$1/3 = 0.333$
3	7	4	Hamming(7,4)	$4/7 = 0.571$
4	15	11	Hamming(15,11)	$11/15 = 0.733$
5	31	26	Hamming(31,26)	$26/31 = 0.839$
m	$2^m - 1$	$2^m - m - 1$	Hamming($2^m - 1$, $2^m - m - 1$)	$1 - m / (2^m - 1)$

Codurile Hamming descrise au distanța Hamming minimă 3 deci permit:

- detectie și corectie a erorilor simple
- detectie a erorilor duble dacă se renunță la corectie.

Uneori se adaugă un **bit suplimentar de paritate** a întregului cod făcând distanța minimă 4. În acest caz se poate distinge între erorile simple (care se pot corecta) și erorile duble care doar se

detecteaza. Codurile astfel rezultate se numesc SECDED ("single error correction, double error detection").

De un interes deosebit este codul (72,64) care reprezinta o versiune trunchiata a codului cu 7 biti de paritate (m=7) adica Hamming (127,120) varianta cu bit aditional de paritate. Acesta este folosit pe circuitele de memorie de 72 biti pentru detectia erorilor duble si corectia erorilor singulare. Codul respectiv are o eficienta de utilizare a bitilor de date de $64/72 = 0.8888$ identica cu situatia simpla a utilizarii unui bit de paritate pe fiecare byte, situatie corespunzatoare unui cod de tip (9,8). În plus fata de codul (9,8) care permite doar detectia erorilor singulare, codul (72,64) este de tip SECDEC adica permite detectia erorilor duble si corectia erorilor simple.

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/308967276>

Teoria Codurilor corectoare de erori

Book · February 2001

CITATION

1

READS

1,272

1 author:



[Adrian Constantin Atanasiu](#)
University of Bucharest

66 PUBLICATIONS 322 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Reaction System (Graduate Research Assistant is required) [View project](#)



Parikh Matrices [View project](#)

Coduri detectoare și corectoare de erori

Adrian Atanasiu

Editura Universității BUCUREȘTI

Prefață

Vă uitați la televizor – care transmite imagini prin satelit? Vorbiți la telefon (celular)? Folosiți Internetul? Ascultați muzică de pe DVD-uri?

Ați admirat fotografiile extrem de reușite ale planetelor Jupiter sau Saturn. Cum au fost obținute ele

Ați auzit de alfabetul Morse; v-ați întrebat de ce pentru ajutor apelul este S.O.S. ?

Sunt multe astfel de domenii intrate în cotidian, care folosesc coduri. De fapt, oricare din exemplele de mai sus se referă la un transfer de informație. Ceea ce se solicită (în mod neexplicit dar esențial) este ca informația cerută să ajungă nemodificată la beneficiarul transmisiei. Indiferent prin ce mediu (numit *canal*) se face transmisia (poate fi laser, cablu, unde etc) mesajul expeditorului trebuie să fie identic cu cel al destinatarului. Chiar dacă sunt “zgomote” care pot altera acest conținut.

Tocmai din cauza acestor posibile perturbații de canal, ceea ce se transmite (numit *mesaj de informație*) se completează cu elemente redondante (numite *caractere de control*), care nu aduc informație suplimentară dar o confirmă pe cea existentă. În acest fel, o modificare posibilă a mesajului (o zgârietură pe DVD, o interferență, un ecou de canal) poate fi eliminată la recepție de către decodor.

Să considerăm de exemplu un mesaj “*abac*” format din patru caractere. Îl vom codifica prin scrierea fiecărui caracter de trei ori. Deci vom transmite “*aaabbbbaaaccc*”. Dacă destinatarul primește “*caabbwaaacbb*”, la decodificare va separa textul în grupe de câte trei simboluri “*caa bbw aaa ccb*” și va păstra din fiecare grup caracterul care apare de cele mai multe ori. Va obține “*abac*”. Este o idee de decodificare folosită cu precădere de aproape toate sistemele, numită *decodificarea cea mai probabilă*.

Evident, pot apare perturbări grupate, care să altereze mai multe caractere consecutive. Cum se operează atunci? Pur și simplu se folosește altă codificare (sau supra-codificare, care este o combinație de mai multe codificări). S-a ajuns în acest moment să existe clase de coduri (turbo-codurile) capabile să corecteze aproape 75% din simbolurile alterate ale unui mesaj!

Pare fascinant. De fapt, așa și este!

Teoria codurilor este un domeniu dezvoltat relativ recent ca aparat matematic (are cam 70 ani vechime). Dacă se face abstracție de unele confuzii de termeni cu criptografia (care utilizează frecvent termenul de “cod” pentru unele sisteme clasice de criptare) sau de ideea că de fapt și scrierea sau vorbirea constituie sisteme de codificare ale ideilor, primul cod care merită acest nume în lumina obiectivelor menționate anterior, este codul Morse apărut în 1840 – de fapt un alfabet care să permită transmiterea mesajelor prin telegraf.

Explozia informațională din a doua jumătate a secolului XX a condus la apariția acestei discipline, ale cărei baze teoretice sunt puse în anii 50, are o dezvoltare fulgerătoare până la

sfârșitul anilor 60, stagnează aproape două decenii, după care renaște din 1985 și cunoaște astăzi din nou o creștere explozivă, explicabilă prin necesitatea transmiterii unui volum de informație nemaîntâlnit la o viteză aproape nebănuită până acum 10-15 ani.

Cea mai mare parte a noțiunilor prezentate aici constituie suportul unui curs ținut o vreme la Facultatea de Matematică a Universității București (întâi profesorul Silviu Guiasu în anii 67-75, apoi reluat de autor după 1990). Lucru explicabil, deoarece bazele teoriei codurilor sunt pur matematice (algebră liniară, teoria numerelor, corpuri finite, statistică). Aceasta nu face însă domeniul mai puțin atrăgător, dimpotrivă. Sugerează că pot fi gândite și alte abordări care să genereze sisteme de codificare noi, cu performanțe superioare. Deja ele apar din domeniul cuantic sau genetic. Ideea în sine constituie o provocare, un apel al societății, al vieții practice, adresat științelor fundamentale. Putem să îi răspundem? Aceasta este întrebarea.

Autorul este conștient că nu a putut acoperi și explica tot ce conține domeniul în sine. De asemenea, lucrarea poate cuprinde unele erori (nu numai de redactare). Este riscul pe care și-l asumă orice autor. De aceea, rugămintea către cititor(ul interesat) este de a nu ezita stabilirea unei colaborări. Suntem deschiși și receptivi la orice mesaj. În ciuda riscurilor unor posibile perturbații de canal.

Capitolul 1

Codificare și decodificare

1.1 Codificare

Definiția 1.1 Fiind date două mulțimi finite și nevide A (alfabetul sursă) și B (alfabetul cod), o **codificare** este o aplicație injectivă $\phi : A \rightarrow B^*$.

Elementele mulțimii $\phi(A) \subseteq B^*$ se numesc *cuvinte-cod*, iar mulțimea $\phi(A)$ se numește *cod*. Dacă B are numai două simboluri, codificarea ϕ se numește *binară*.

Exemplul 1.1 Fie $A = \{0, 1, \dots, 9\}$ și $B = \{0, 1\}$. Printre secvențele binare de lungime 5, numărul celor care au doi de 1 este $C_5^2 = 10$. Ele pot fi folosite pentru a codifica cifrele din scrierea zecimală (Tabelul 8.1).

Tabelul 1.1: Codul "doi-din-cinci"

Simbol zecimal	Cuvânt cod
1	11000
2	10100
3	01100
4	10010
5	01010
6	00110
7	10001
8	01001
9	00101
0	00011

Mesajul "173" are codul 110001000101100. De remarcat că între cuvinte - cod nu se lasă nici un spațiu, deoarece caracterul "spațiu" poate fi el însuși un simbol din alfabetul cod. Astfel de exemplu, codul Morse are alfabetul $B = \{., -, \text{spațiu}\}$.

Decodificarea se face foarte simplu: se împarte mesajul codificat în grupe de câte cinci caractere și se vede cifra din Tabelul 8.1 corespunzătoare grupei respective.

Aranjarea cuvintelor cod a fost făcută pentru a realiza și o decodificare pe baza unei formule. Astfel, dacă $a_0a_1a_2a_3a_4$ este un cuvânt - cod, el corespunde cifrei k rezultată din de algoritmul:

```
begin
   $x := a_1 + 2a_2 + 4a_3 + 7a_4$ ;
  if  $x = 11$  then  $k := 0$  else  $k := x$ ;
end.
```

Definiția 1.2 Pentru o codificare $\phi : A \rightarrow B^*$, se numește "codificare a mesajelor (textului) sursă" aplicația $\phi^* : A^* \rightarrow B^*$ definită recursiv prin:

- $\phi^*(\epsilon) = \epsilon$ (ϵ este cuvântul vid);
- $\phi^*(a\alpha) = \phi(a)\phi^*(\alpha)$, $\forall a \in A, \alpha \in A^*$.

Definiția 1.3 Codificarea ϕ este "unic decodabilă" dacă ϕ^* este injectivă.

Codificarea dată în Exemplul 2.1 este – după cum s-a observat – unic decodabilă. Acest lucru nu este totdeauna posibil. Dacă luăm de exemplu codificarea

$$\phi(a) = 10, \quad \phi(b) = 101, \quad \phi(c) = 110,$$

ea nu este unic decodabilă; astfel $\phi^*(ac) = \phi^*(ba) = 10110$.

Definiția 1.4

1. O codificare $\phi : A \rightarrow B^*$ în care toate cuvintele cod au lungimi n se numește "codificare-bloc de lungime n ", iar $\phi(A)$ este un "cod-bloc de lungime n ".
 2. O codificare $\phi : A \rightarrow B^*$ se numește "instantanee" dacă $\phi(A)$ are proprietatea prefixului (dacă $\alpha, \alpha\beta \in \phi(B)$ atunci $\beta = \epsilon$).
- Mulțimea $\phi(A)$ este numită "cod instantaneu".

Codul definit în Exemplul 2.1 este un cod - bloc de lungime 5.

Codurile bloc sunt eficiente în situația când simbolurile sursă au frecvențe egale de apariție; în caz contrar, ele devin greoaie și sunt preferabile codurile instantanee cu lungimi variabile ale cuvintelor cod.

Exemplul 1.2 Codul Morse, dat în Tabelul 8.2 este un cod instantaneu cu alfabetul cod $B = \{., -\}$. Deoarece spațiul este folosit numai la sfârșitul fiecărui cuvânt - cod, procedura de decodificare este simplă: orice cuvânt - cod se află între două spații, de la începutul mesajului până la primul spațiu, sau de la ultimul spațiu până la sfârșit. Motivul pentru care nu se folosește un cod - bloc este simplu: frecvențele literelor într-o limbă diferă foarte mult.

Tabelul 1.2: Codul Morse

A	. -	F	K	- - -	P	. - - .	U	. . -
B	- . . .	G	- - .	L	. - . . .	Q	- - - .	V -
C	- . - .	H	M	- -	R	. - .	W	. - -
D	- . .	I	. .	N	- .	S	. . .	X	- . . . -
E	.	J	. - - -	O	- - -	T	-	Y	- . - -
								Z	- - . .

Exemplul 1.3 *Un alt exemplu de cod - bloc este codul hexazecimal:*

0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

Exemplul 1.4 *Să presupunem că vrem să construim un cod binar pentru alfabetul $\{0, 1, 2, 3\}$ și observăm că 0 apare în mesajele sursă mai des decât orice alt simbol. Atunci următoarea schemă de codificare pare rezonabilă:*

$$\phi(0) = 0, \quad \phi(1) = 01, \quad \phi(2) = 011, \quad \phi(3) = 111.$$

Aici $\phi(0)$ are lungime minimă, iar algoritmul de decodificare este foarte simplu: se aplică recursiv regula:

"Fie sufixul 01^k ; el este codificarea numărului $k \pmod{3}$ $\underbrace{33\dots3}_{k \text{ div } 3}$ "

Totuși această codificare nu este instantanee. Într-adevăr, dacă se primește un mesaj lung de forma

$$0111111111111111\dots$$

nu vom ști dacă primul simbol sursă este 0, 1 sau 2 până nu se termină transmiterea întregului mesaj.

1.2 Exemple de coduri - bloc importante

Deși simple, codurile binare sunt de obicei lungi și deci greu de manipulat. Adesea este mai convenabil să grupăm simbolurile binare formând alfabetele mai complexe.

Astfel, grupuri de câte trei simboluri binare conduc la codurile octale. Reprezentarea în octal se indică de obicei prin indicele 8 așezat la sfârșit. De exemplu,

$$\phi(01) = (01)_8 = 000001$$

În mod similar, prin gruparea a câte patru simboluri binare se obține codul hexazecimal (Exemplul 2.3).

Un cod foarte important folosit în reprezentarea standard a simbolurilor alfabetice și numerice este codul *ASCII* (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)¹.

¹Acest cod a fost prezentat pe larg în cursul de Arhitectura sistemelor de Calcul.

Un ultim cod, folosit de toate editurile de pe glob este *Internațional Standard Book Number* (ISBN). El este un cod - bloc de lungime 10 (lungimea cuvintelor - cod crește prin folosirea simbolului e diverse poziții, dar acest caracter este ignorat la prelucrarea automată). Alfabetul cod este $B = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, X\}$, (X pentru numărul 10).

De exemplu, cartea *S. Lin, P. Costello - Teoria Codurilor* are codul

$$ISBN\ 0 - 13 - 283796 - X$$

Primul număr (0) reprezintă țara (SUA), 13 reprezintă editura (*Prentice-Hall*), iar următoarele șase cifre sunt asignate de editură ca număr de identificare al cărții. Ultimul simbol este de control (similar bitului de paritate) și definit astfel:

$$\text{Pentru codul ISBN } a_1 a_2 \dots a_{10}, \quad \sum_{i=1}^{10} i a_{11-i} = 0 \pmod{11}.$$

Astfel, în ISBN-ul de sus,

$$10 \cdot 0 + 9 \cdot 1 + 8 \cdot 3 + 7 \cdot 2 + 6 \cdot 8 + 5 \cdot 3 + 4 \cdot 7 + 3 \cdot 9 + 2 \cdot 6 + 1 \cdot 10 = 187 \equiv 0 \pmod{11}$$

Unele publicații au codul de identificare de trei cifre (de exemplu *Wiley-Inter-Science* are 471); în acest caz numărul pentru fiecare publicație are numai cinci simboluri.

Pentru România, codul de țară este 973.

1.3 Construcția codurilor instantanee

În acest paragraf ne punem problema construirii unui cod binar instantaneu peste alfabetul sursă $A = \{a_1, \dots, a_n\}$, ($n \geq 1$). Deci $B = \{0, 1\}$.

Inițial se specifică lungimile d_1, d_2, \dots, d_n ale cuvintelor cod. Fără a micșora generalitatea, putem presupune

$$1 \leq d_1 \leq d_2 \leq \dots \leq d_n$$

La primul pas se alege un cuvânt - cod binar arbitrar $\phi(a_1)$ de lungime d_1 .

În continuare se alege un cuvânt - cod arbitrar $\phi(a_2)$ din mulțimea secvențelor binare de lungime d_2 care nu au pe $\phi(a_1)$ ca prefix. Aceasta este totdeauna posibil pentru că numărul tuturor secvențelor binare de lungime d_2 este 2^{d_2} ; dintre acestea, numărul celor care au prefixul $\phi(a_1)$ este $2^{d_2-d_1}$.

Cum $2^{d_2} \geq 2^{d_2-d_1} + 1$, există cel puțin o alegere posibilă pentru $\phi(a_2)$ de lungime d_2 .

Va trebui să selectăm în continuare un cuvânt de lungime d_3 care nu are ca prefix $\phi(a_1)$ sau $\phi(a_2)$. Deci, din cele 2^{d_3} secvențe binare posibile trebuie eliminate cele $2^{d_3-d_1}$ secvențe cu prefixul $\phi(a_1)$ și $2^{d_3-d_2}$ secvențe cu prefixul $\phi(a_2)$. Aceasta este posibil dacă și numai dacă

$$2^{d_3} \geq 2^{d_3-d_2} + 2^{d_3-d_1} + 1$$

Împărțind această inegalitate cu 2^{d_3} , se obține

$$1 \geq 2^{-d_1} + 2^{-d_2} + 2^{-d_3}.$$

În mod analog se poate arăta în general inegalitatea

$$1 \geq 2^{-d_1} + 2^{-d_2} + \dots + 2^{-d_n}$$

Faptul că acest cod este instantaneu rezultă imediat din construcție.

Teorema 1.1 (Kraft) Fiind dat un alfabet sursă A de n simboluri și un alfabet cod B de k simboluri, se poate construi un cod instantaneu cu lungimile cuvintelor - cod d_1, d_2, \dots, d_n dacă și numai dacă este verificată inegalitatea:

$$k^{-d_1} + k^{-d_2} + \dots + k^{-d_n} \leq 1.$$

Demonstrație: "⇐": Fie $A = \{a_1, a_2, \dots, a_n\}$ și putem presupune relația $d_1 \leq d_2 \leq \dots \leq d_n$. Construim prin inducție codificarea instantanee ϕ , astfel:

- Se alege arbitrar $\phi(a_1) \in B^*$, $|\phi(a_1)| = d_1$ (se notează cu $|\alpha|$ lungimea secvenței α).
- Presupunem că au fost alese $\phi(a_1), \phi(a_2), \dots, \phi(a_{s-1})$ ($s > 1$). Atunci se va alege un cuvânt arbitrar $\phi(a_s) \in B^*$ care nu are ca prefix nici unul din cuvintele selectate anterior. Aceasta este posibil deoarece numărul cuvintelor cu prefixul $\phi(a_i)$ este $k^{d_s - d_i}$ ($1 \leq i \leq s - 1$); deci alegerea poate fi făcută din

$$k^{d_s} - \sum_{i=1}^{s-1} k^{d_s - d_i} \text{ elemente.}$$

Din inegalitatea lui Kraft avem

$$1 - \sum_{i=1}^{s-1} k^{-d_i} \geq k^{-d_s}$$

care, prin multiplicare cu k^{d_s} conduce la

$$k^{d_s} - \sum_{i=1}^{s-1} k^{d_s - d_i} \geq 1,$$

deci există cel puțin un element de lungime d_s care poate fi ales drept $\phi(a_s)$.

Afirmația reciprocă se demonstrează similar (pentru $k = 2$ ea a fost dată anterior).q.e.d.

Teorema 1.2 (McMillan) Orice codificare unic decodabilă satisface inegalitatea lui Kraft.

Demonstrație: Fie ϕ o codificare unic decodabilă. Notăm cu d_i lungimea cuvântului cod $\phi(a_i)$, ($1 \leq i \leq n$). Se observă că $\forall j$, ($j \geq 1$) se pot forma k^j cuvinte de lungime j peste alfabetul - cod B cu $|B| = k$. Din proprietatea de unic decodabilitate, numărul mesajelor sursă $\alpha = a_{i_1} a_{i_2} \dots a_{i_r}$ cu $\phi(\alpha) = j$ nu depășește k^j . Lungimea codului pentru α este $d_{i_1} + d_{i_2} + \dots + d_{i_r}$; deci numărul tuturor sumelor de forma

$$d_{i_1} + d_{i_2} + \dots + d_{i_r} = j$$

este cel mult k^j .

Rămâne de demonstrat că numărul $c = \sum_{i=1}^n k^{-d_i}$ este cel mult 1. Pentru aceasta, vom

arăta că $\forall r \geq 1$, $\frac{c^r}{r}$ este mărginit.

Să calculăm puterile lui c :

$$c^2 = \left(\sum_{i=1}^n k^{-d_i} \right) \left(\sum_{j=1}^n k^{-d_j} \right) = \sum_{i,j=1}^n k^{-(d_i+d_j)}$$

și în general,

$$c^r = \sum_{i_1, i_2, \dots, i_r=1}^n k^{-(d_{i_1}+d_{i_2}+\dots+d_{i_r})}$$

Această sumă se poate re-ordona grupând toți termenii de forma k^{-j} unde j satisface egalitatea anterioară. Cel mai mare j posibil este $j = d + d + \dots + d = rd$, unde $d = \max\{d_1, d_2, \dots, d_n\}$.

Numărul tuturor termenilor de forma k^{-j} din sumă este cel mult k^j . Deci,

$$c^r \leq \sum_{j=1}^{rd} k^j k^{-j} = \sum_{j=1}^{rd} 1 = rd.$$

Deci, $\frac{c^r}{r} \leq d$, de unde va rezulta $c \leq 1$ (pentru $c > 1$ șirul $a_r = \frac{c^r}{r} \rightarrow \infty$, deci nu este mărginit). q.e.d.

Corolarul 1.1 Pentru orice cod unic decodabil există un cod instantaneu care are toate cuvintele - cod de lungimi egale.

Demonstrație: Rezultă din demonstrația teoremei precedente.

Exemplul 1.5 Să considerăm alfabetul sursă $A = \{a, b, c\}$ și alfabetul cod $B = \{0, 1\}$; deci $n = |A| = 3$, $k = |B| = 2$. Vrem să construim o codificare instantanee $\phi : A \rightarrow B$ care are toate cuvintele - cod de lungime d .

Inegalitatea Kraft va da $2^{-d} + 2^{-d} + 2^{-d} \leq 1$, deci $2^{-d} \leq \frac{1}{3}$. Cel mai mic d care o verifică este $d = 2$. Deci orice mulțime de trei secvențe binare de lungime 2 va putea fi folosită drept cod. Sunt 4 astfel de mulțimi:

$$\{00, 01, 10\}, \quad \{00, 01, 11\}, \quad \{00, 10, 11\}, \quad \{01, 10, 11\}$$

1.4 Coduri Huffman

Am menționat anterior faptul că dacă frecvența simbolurilor sursă variază, atunci codurile instantanee sunt preferabile codurilor - bloc, deoarece simbolurile care apar mai frecvent vor fi codificate prin cuvinte - cod mai scurte.

Ne punem problema aflării unor codificări cât mai eficiente, în ipoteza că frecvențele simbolurilor sursă sunt cunoscute exact (de exemplu probabilitatea distribuției simbolurilor sursă în mesaje).

Definiția 1.5 O sursă de informație este o pereche $S = (A, P)$ unde

- $A = \{a_1, a_2, \dots, a_n\}$ este alfabetul sursă (mulțime ordonată);

- $P = \{P(a_1), P(a_2), \dots, P(a_n)\}$ este mulțimea ordonată a probabilităților elementelor lui A , deci

$$- 0 \leq P(a_i) \leq 1, (1 \leq i \leq n);$$

$$- \sum_{i=1}^n P(a_i) = 1.$$

Fie ϕ o codificare a unei surse de informație. Dacă se notează cu $d_i = |\phi(a_i)|$, se poate defini *lungimea medie* L a cuvintelor cod prin

$$L(\phi) = \sum_{i=1}^n d_i P(a_i).$$

O codificare este eficientă dacă lungimea medie a cuvintelor - cod este cât mai mică.

Definiția 1.6 Fiind dată o sursă de informație S și un alfabet cod, un cod Huffman este un cod instantaneu cu lungimea medie minimă.

Lungimea medie minimă a unui cod Huffman se notează cu $L_{min}(S)$.

Exemplul 1.6 Să se determine un cod Huffman binar pentru alfabetul sursă $A = \{a, b, c, d, e, f\}$ știind că 'a' apare de două ori mai des decât 'e' și 'e' de două ori mai des decât orice consoană.

Deci, vom avea sursa de informație

Simbol	a	b	c	d	e	f
Probabilitate	0,4	0,1	0,1	0,1	0,2	0,1

Putem asigna deci un cuvânt - cod de lungime 1 lui 'a' și unul de lungime doi lui 'e'. Atunci lungimile cuvintelor - cod rămase sunt egale cu 4, iar inegalitatea lui Kraft este saturată:

$$\frac{1}{2} + \frac{1}{2^2} + \frac{4}{2^4} = 1.$$

Un astfel de cod se poate construi efectiv:

$$\begin{array}{lll} \phi(a) = 0 & \phi(c) = 1101 & \phi(e) = 10 \\ \phi(b) = 1100 & \phi(d) = 1110 & \phi(f) = 1111 \end{array}$$

Lungimea sa medie este

$$L = 0,4 + 2 \times 0,2 + 4 \times 4 \times 0,1 = 2,4.$$

Deci, pentru acest exemplu, $L_{min}(S) \leq 2,4$.

1.4.1 Construcția codurilor Huffman binare

O sursă cu două simboluri are evident un cod Huffman de cuvinte - cod $\{0, 1\}$ (și deci $L_{\min}(S) = 1$).

O sursă cu trei simboluri $\{a_1, a_2, a_3\}$ în care a_1 are probabilitate maximă, poate fi redusă la cazul a două simboluri $\{a_1, a_{2,3}\}$ unde $P(a_{2,3}) = P(a_2) + P(a_3)$. Vom construi o codificare Huffman pentru sursa redusă

$$\phi(a_1) = 0, \quad \phi(a_{2,3}) = 1.$$

după care "spargem" cuvântul cod 1 în două cuvinte: 10 și 11; în acest fel se obține o codificare Huffman pentru sursa originală:

$$\begin{array}{ccc} a_1 & a_2 & a_3 \\ 0 & 10 & 11 \end{array}$$

În general, fie S o sursă de informație cu simbolurile $\{a_1, a_2, \dots, a_n\}$ ordonate descrescător după probabilități, adică:

$$P(a_1) \geq P(a_2) \geq \dots \geq P(a_n).$$

Construim o sursă redusă S^* cu simbolurile $\{a_1, \dots, a_{n-2}, a_{n-1,n}\}$ unde $a_{n-1,n}$ este un simbol nou, cu probabilitatea $P(a_{n-1,n}) = P(a_{n-1}) + P(a_n)$.

Dacă nu se poate construi un cod Huffman pentru S^* , se reia procedeul pentru această sursă (reordonând eventual simbolurile după probabilitate); în final se va ajunge la o sursă (pentru două simboluri problema a fost rezolvată) în care se poate construi codul Huffman.

Dacă se poate găsi o codificare Huffman ϕ^* pentru sursa redusă S^* , atunci codul din Tabelul 1.3 este un cod Huffman pentru S (vom demonstra această afirmație).

Tabelul 1.3:

a_1	a_2	\dots	a_{n-2}	a_{n-1}	a_n
$\phi^*(a_1)$	$\phi^*(a_2)$	\dots	$\phi^*(a_{n-2})$	$\phi^*(a_{n-1,n})0$	$\phi^*(a_{n-1,n})1$

Lema 1.1

$$L(\phi) = L(\phi^*) + P(a_{n-1}) + P(a_n).$$

Demonstrație: Fie $d_1, d_2, \dots, d_{n-2}, d^*$ lungimile cuvintelor cod corespunzătoare lui ϕ^* . Atunci lungimile cuvintelor cod pentru ϕ sunt $d_1, d_2, \dots, d_{n-2}, d^* + 1, d^* + 1$.

Efectuând calculele, se obține:

$$\begin{aligned} L(\phi) &= \sum_{i=1}^{n-2} d_i P(a_i) + (d^* + 1)P(a_{n-1}) + (d^* + 1)P(a_n) = \\ &= \sum_{i=1}^{n-2} d_i P(a_i) + d^* [P(a_{n-1}) + P(a_n)] + P(a_{n-1}) + P(a_n) = \\ &= L(\phi^*) + P(a_{n-1}) + P(a_n). \end{aligned}$$

q.e.d.

Teorema 1.3 Fie ϕ^* o codificare Huffman pentru o sursă de informație redusă S^* . Atunci codificarea ϕ definită de Tabelul 1.3 determină un cod Huffman pentru sursa de informație S .

Demonstrație: Fie a_1, a_2, \dots, a_n simbolurile sursă, ordonate descrescător după probabilitate. Deoarece teorema este evidentă pentru $P(a_n) = 0$, vom considera doar cazul $P(a_n) > 0$.

Demonstrația constă din trei pași:

- S admite o codificare Huffman ϕ_0 cu lungimile cuvintelor - cod ordonate:

$$d_1 \leq d_2 \leq \dots \leq d_n \quad (d_i = |\phi_0(a_i)|, \quad 1 \leq i \leq n).$$

Pentru a demonstra aceasta, plecăm de la o codificare Huffman arbitrară ϕ pentru S . Dacă există un simbol a_i astfel ca $d_i > d_{i+1}$, notăm cu ϕ' codificarea obținută din ϕ prin permutarea cuvintelor cod corespunzătoare lui a_i și a_{i+1} . Codul obținut prin codificarea ϕ' este evident un cod instantaneu, iar diferența dintre lungimile medii $L = L_{min}$ (al lui ϕ) și L' (al lui ϕ') este:

$$\begin{aligned} L_{min} - L' &= [d_i P(a_i) + d_{i+1} P(a_{i+1})] - [d_{i+1} P(a_i) + d_i P(a_{i+1})] = \\ &= (d_i - d_{i+1}) [P(a_i) - P(a_{i+1})]. \end{aligned}$$

Această expresie este produsul dintre două numere pozitive, deci $L_{min} \geq L'$, iar din proprietatea de minimalitate rezultă $L_{min} = L'$. Cu alte cuvinte, ϕ' este tot o codificare Huffman. Procedeu continuă până se obține codificarea ϕ_0 cerută.

- S admite o codificare Huffman ϕ_1 în care ultimele cuvinte cod, $\phi_1(a_{n-1})$ și $\phi_1(a_n)$ diferă doar prin ultimul simbol.

Fie ϕ_0 codificarea Huffman anterioară și $\tilde{\phi}_0$ codificarea rezultată din ϕ_0 eliminând ultimul simbol din $\phi_0(a_n)$. Lungimea medie a lui $\tilde{\phi}_0$ va fi evident mai mică decât cea a lui ϕ_0 (pentru că $P(a_n) > 0$), deci $\tilde{\phi}_0$ nu poate fi instantanee. Cuvântul - cod $\tilde{\phi}_0(a_i) = \phi_0(a_i)$, ($1 \leq i \leq n-1$) nu este prefixul nici unui alt cuvânt cod; deci există un i ($i \leq n-1$) astfel încât $\phi_0(a_n)$ este prefixul lui $\phi_0(a_i)$. Aceasta este posibil numai dacă $d_i = d_n$ și deci $\phi_0(a_i)$ diferă de $\phi_0(a_n)$ numai prin ultimul simbol. Dacă $i = n-1$, se ia $\phi_1 = \phi_0$. Altfel, se observă că $d_i = d_n$ implică $d_i = d_{i+1} = \dots = d_n$; deci se pot permuta cuvintele cod definite în ϕ_0 pentru a_i și a_{n-1} . Codificarea ϕ_1 astfel obținută are aceeași lungime medie ca și ϕ_0 , deci definește un cod Huffman.

- Să presupunem că se dă o codificare Huffman ϕ^* pentru sursa redusă S^* și definim o codificare ϕ pentru S conform Tabelului 1.3. Lungimile lor medii $L(\phi)$, $L(\phi^*)$ verifică relația din Lema 1.1.

Să folosim acum codificarea Huffman ϕ_1 construită anterior. Deoarece ultimele două cuvinte - cod diferă numai prin ultimul simbol, ϕ_1 poate fi obținut dintr-o codificare ϕ_1^* al lui S^* prin "spargerea" ultimului cuvânt - cod. În plus, ϕ_1^* este evident o codificare instantanee.

Prin calcule se ajunge la relația

$$L(\phi_1) - L(\phi_1^*) = P(a_{n-1}) + P(a_n).$$

Cum avem și $L(\phi) - L(\phi^*) = P(a_{n-1}) + P(a_n)$, rezultă

$$L(\phi) = L(\phi_1) - L(\phi_1^*) + L(\phi^*).$$

Acum, $L(\phi^*) = L_{\min}(S^*)$, deci $-L(\phi_1^*) + L(\phi^*) \leq 0$.

Rezultă $L(\phi) \leq L(\phi_1) = L_{\min}(S)$. Deci, codificarea ϕ definește un cod Huffman.

q.e.d.

1.5 Exerciții

Exercițiul 1.1 Care este cea mai mică lungime a unui cod - bloc cu alfabetul sursă $A = \{a, b, \dots, z\}$ și alfabetul cod $B = \{., -, spa'tiu\}$ (ca la codul Morse).

Dar dacă B are patru caractere ?

Exercițiul 1.2 Se definește codificarea

$$\begin{array}{ll} 1 \rightarrow 01 & 4 \rightarrow 1000 \\ 2 \rightarrow 011 & 5 \rightarrow 1100 \\ 3 \rightarrow 10 & 6 \rightarrow 0111 \end{array}$$

Este ea unic decodabilă ? Este instantanee ? Se poate găsi un cod instantaneu cu aceleași lungimi ale cuvintelor cod ?

Exercițiul 1.3 Se definește codificarea

$$\begin{array}{ll} a \rightarrow 1010 & d \rightarrow 0001 \\ b \rightarrow 001 & e \rightarrow 1101 \\ c \rightarrow 101 & f \rightarrow 1011 \end{array}$$

Este ea unic decodabilă ? Dacă nu, găsiți două mesaje sursă cu același cod.

Exercițiul 1.4 Este unic decodabilă codificarea:

$$\begin{array}{lll} 0 \rightarrow AA & 4 \rightarrow ABBAA & 7 \rightarrow AAAABB \\ 1 \rightarrow AABAB & 5 \rightarrow BABBA & 8 \rightarrow AAAABA \\ 2 \rightarrow ABBBBB & 6 \rightarrow BBBAB & 9 \rightarrow AAAAAB \\ 3 \rightarrow ABABA & & \end{array}$$

Exercițiul 1.5 Se poate decide unic decodabilitatea codificărilor

$$\begin{array}{ll} \phi(a) = 001 & \phi(a) = 00 \\ \phi(b) = 1001 & \phi(b) = 10 \\ \phi(c) = 0010 & \phi(c) = 011 \\ \phi(d) = 1110 & \phi(d) = 101 \\ \phi(e) = 1010 & \phi(e) = 111 \\ \phi(f) = 01110 & \phi(f) = 110 \\ \phi(g) = 0101 & \phi(g) = 010 \end{array}$$

folosind inegalitatea lui Kraft ?

Exercițiul 1.6 Să se construiască un cod binar instantaneu pentru următorul alfabet sursă cu lungimile corespunzătoare ale cuvintelor cod:

Simbol	A	B	C	D	E	F	G	H	I	J	K	L
Lungime	2	4	7	7	3	4	7	7	3	4	7	7

Exercițiul 1.7 Să se construiască un cod instantaneu pentru următorul alfabet sursă, cu lungimile corespunzătoare ale cuvintelor cod:

<i>Simbol</i>	1	2	3	4	5	6	7	8	9	0
<i>Lungime</i>	1	3	3	3	3	3	2	2	2	2

care este numărul minim e simboluri - cod necesare ?

Exercițiul 1.8 Câte simboluri cod sunt necesare pentru ca următorul alfabet sursă să poată fi codificat într-un cod instantaneu cu lungimile cuvintelor cod date:

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>	<i>H</i>	<i>I</i>	<i>J</i>	<i>K</i>	<i>L</i>	<i>M</i>	<i>N</i>	<i>O</i>	<i>P</i>
1	2	2	2	1	2	2	2	1	2	2	2	2	2	1	2

Exercițiul 1.9 Demonstrați că pentru orice cod instantaneu în care inegalitatea Kraft este strictă, este posibil să se adauge un nou simbol sursă și să se extindă codul dat la un nou cod instantaneu (cu același alfabet cod). Demonstrați aceasta pentru codul definit la Exercițiul 2.7.

Capitolul 2

Coduri liniare

2.1 Matrice generatoare

Definiția 2.1 Fie q un număr prim și $n \in \mathbb{N}^*$ un număr natural nenul. Se numește ”cod liniar” orice subspațiu liniar al lui Z_q^n .

Un subspațiu k -dimensional ($k \leq n$) al lui Z_q^n se numește (n, k) -cod liniar peste Z_q .

Să notăm în general cu $A_{n,k}$ ($A_{n,k} \subseteq Z_q^n$) un (n, k) -cod liniar¹. Elementele sale se numesc *cuvinte-cod*.

Fie $n, k \in \mathbb{N}$, ($k < n$). O *codificare* este o aplicație injectivă $\phi : Z_q^k \rightarrow Z_q^n$, iar $A_{n,k} = \phi(Z_q^k)$. Elementele lui Z_q^k se numesc *mesaje de informație*.

Deci, $\mathbf{x} \in Z_q^k$ este un mesaj de informație scris cu caractere din alfabetul Z_q . Dacă transmitem succesiunea \mathbf{x} de semnale printr-un canal de comunicație, mesajul este supus diverselor perturbații ”de canal” care-l modifică. Ideea teoriei codurilor este următoarea: în loc de a transmite numai elementele lui Z_q^k , să ”lungim” mesajul informațional scufundând (prin intermediul aplicației ϕ) Z_q^k într-un spațiu liniar Z_q^n ($n > k$) astfel încât cele $n - k$ poziții noi - numite *poziții de control* - să asigure redondanța necesară refacerii mesajului de informație inițial (atunci când a fost alterat parțial).

Astfel, cu prețul lungirii mesajului, se câștigă protecția față de (anumite tipuri de) erori.

Observația 2.1

- Din definiție rezultă că un cod liniar de lungime n este un set C de cuvinte (secvențe, șiruri, vectori) de lungime n cu proprietățile:

$$- \forall \mathbf{a} = (a_1, \dots, a_n) \in C, \mathbf{b} = (b_1, \dots, b_n) \in C \implies \mathbf{a} + \mathbf{b} = (a_1 + b_1, \dots, a_n + b_n) \in C;$$

$$- \forall \mathbf{a} = (a_1, \dots, a_n) \in C, t \in Z_q \implies t \cdot \mathbf{a} = (ta_1, \dots, ta_n) \in C.$$

- Orice cod liniar conține cuvântul cod nul $\mathbf{0} = (0, 0, \dots, 0)$.

- $|A_{n,k}| = q^k$.

¹Dacă nu este necesar să specificăm dimensiunile n și k , vom mai folosi și notația C pentru a desemna un cod liniar.

$A_{n,k}$ fiind un spațiu liniar k -dimensional, admite o bază formată din k vectori cu n elemente fiecare. Fie

$$\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_k$$

o astfel de bază. Atunci orice cuvânt cod $\mathbf{v} \in A_{n,k}$ are forma

$$\mathbf{v} = \sum_{i=1}^k u_i \mathbf{e}_i$$

unde $(u_1, u_2, \dots, u_k) \in Z_q^k$ este unic determinat.

Cu alte cuvinte, k simboluri de informație $u_1, \dots, u_k \in Z_q$ determină în mod unic un cuvânt - cod $\mathbf{v} \in A_{n,k}$ prin relația de sus, și reciproc. Operația de codificare ϕ face această asociere biunivocă:

$$\mathbf{u} = (u_1, \dots, u_k) \in Z_q^k \iff \mathbf{v} = \sum_{i=1}^k u_i \mathbf{e}_i \in A_{n,k}$$

Fiecare cuvânt - cod într-un cod liniar va avea deci k simboluri de informație; celelalte $n - k$ simboluri se numesc *simboluri de control*.

Definiția 2.2 Fie $A_{n,k}$ un cod liniar cu baza $\mathbf{e}_1, \dots, \mathbf{e}_k$. Matricea

$$G_{k,n} = \begin{pmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \dots \\ \mathbf{e}_k \end{pmatrix}$$

se numește "matricea generatoare" a codului.

Deci operația de codificare este definită prin matricea generatoare G :

$$\forall \mathbf{u} \in Z_q^k, \quad \phi(\mathbf{u}) = \mathbf{u}G.$$

Exemplul 2.1 Matricea

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

generează un $(5, 2)$ -cod liniar peste Z_2 ($n = 5$, $k = 2$). Rangul ei este 2 (primele două coloane formează matricea unitate), deci cele două linii ale lui G sunt cuvinte - cod liniar independente.

Mulțimea mesajelor de informație este $Z_2^2 = \{00, 01, 10, 11\}$. Înmulțind fiecare mesaj cu matricea G se obține spațiul liniar al cuvintelor - cod

$$A_{5,2} = \{00000, 01101, 10011, 11110\}.$$

Funcția de codificare este:

$$00 \rightarrow 00000, \quad 01 \rightarrow 01101, \quad 10 \rightarrow 10011, \quad 11 \rightarrow 11110.$$

Un cod liniar poate fi generat de mai multe baze posibile. Deci se pot construi mai multe matrici generatoare pentru un (n, k) -cod liniar. Ele se pot transforma una în alta prin operațiile liniare obișnuite, definite pentru liniile unei matrici. Prin schimbarea matricii generatoare nu se schimbă spațiul liniar al cuvintelor - cod, ci numai modalitatea de codificare (funcția ϕ). Cum prin termenul *cod* se înțelege de obicei spațiul liniar $A_{n,k}$, rezultă că două matrici diferite care se deduc una din alta prin operații pe linii, reprezintă același cod.

Exemplul 2.2 *Reluând Exemplul 2.1, prin adunarea liniei doi la prima linie se obține matricea*

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

Ea generează același spațiu liniar $A_{5,2}$, dar codificarea diferă:

$$00 \rightarrow 00000, \quad 01 \rightarrow 01101, \quad 10 \rightarrow 11110, \quad 11 \rightarrow 10011$$

Exemplul 2.3 *Matricea*

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

generează un $(6, 3)$ -cod liniar peste Z_3 . Aducând matricea la forma canonică, se obține

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

Toate cele $3^3 = 27$ cuvinte ale acestui cod au următoarea proprietate: fiecare simbol este scris de două ori. Din acest motiv, astfel de cod este numit "cod cu repetiție".

Cea mai convenabilă regulă de codificare constă în scrierea simbolurilor de informație și suplimentarea lor cu $n - k$ simboluri de control. Aceasta corespunde matricii generatoare (unice)

$$G = (I|B)$$

unde I este matricea unitate de ordin k , iar B este o matrice cu k linii și $n - k$ coloane.

Definiția 2.3 *Un cod liniar este numit "sistematic" dacă admite o matrice generatoare de forma $G = (I|B)$ unde I este matricea unitate.*

Definiția 2.4 *Două coduri liniare C și C' de aceeași lungime n se numesc echivalente dacă există o permutare $\pi \in S_n$ astfel încât*

$$(v_1, v_2, \dots, v_n) \in C \iff v_{\pi(1)}v_{\pi(2)} \dots v_{\pi(n)} \in C'$$

(s-a notat cu S_n mulțimea permutărilor de n elemente).

Exemplul 2.4 Codul din Exemplul 2.1 este un cod sistematic. Din codificare se observă că mesajul de informație se află în cuvântul - cod pe primele două poziții.

Codul cu repetiție din Exemplul 2.3 nu este sistematic. Totuși, folosind permutarea $(1, 4, 6, 2, 5, 3)$ (se permută simbolurile doi cu patru și trei cu șase) se ajunge la un cod sistematic generat de matricea

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

Teorema 2.1 Orice cod liniar este echivalent cu un cod liniar sistematic.

Demonstrație: Matricea generatoare G a unui (n, k) - cod liniar $A_{n,k}$ are rangul k ; deci ea are k coloane liniar independente.

1. Să presupunem că primele k coloane ale lui G sunt liniar independente. Deci $G = (X|Y)$ unde X este o matrice $k \times k$ inversabilă. Există atunci o succesiune de operații de linii care transformă X în matricea unitate.

Aceeași succesiune de operații efectuate acum pentru toată matricea G va conduce la matricea $G' = (I|Y')$. Deoarece și G' este matrice generatoare pentru codul $A_{n,k}$, rezultă că acesta este sistematic.

2. Fie π permutarea care aduce coloanele liniar independente ale matricii G pe primele k poziții. Se obține în acest fel o nouă matrice G' . Fie $A'_{n,k}$ codul liniar obținut prin codificarea cu matricea generatoare G' . Atunci $A_{n,k}$ și $A'_{n,k}$ sunt echivalente, iar $A'_{n,k}$ este sistematic, conform cazului anterior. q.e.d.

2.2 Matrice de control

Definiția 2.5 Fie $A_{n,k}$ un cod liniar generat de matricea $G = G_{k,n}$. Se numește "matrice de control" o matrice $H = H_{n-k,n}$ cu proprietatea $GH^T = \mathbf{0}$.

Observația 2.2

- Din definiție rezultă că matricea de control H a unui cod liniar C are următoarea proprietate:

$$\mathbf{v} \in C \iff \mathbf{v}H^T = \mathbf{0}$$

Lăsăm ca exercițiu demonstrarea acestei echivalențe.

- Prin transpunere, relația de sus se poate scrie și $HG^T = \mathbf{0}$. Aceasta înseamnă că și H este matricea generatoare a unui $(n, n-k)$ - cod liniar peste corpul Z_q , cod pentru care G este matrice de control. Cele două coduri astfel definite se numesc "coduri duale".

Cuvintele - cod din cele două coduri duale sunt ortogonale (produsul lor scalar este 0). Într-adevăr, dacă C și C' sunt două coduri duale generate de matricile G respectiv H , iar $\mathbf{x} \in C, \mathbf{y} \in C'$ sunt cuvinte - cod arbitrare, există \mathbf{u}, \mathbf{v} cu $\mathbf{x} = \mathbf{u}G, \mathbf{y} = \mathbf{v}H$. În plus, $\mathbf{x}H^T = \mathbf{0}, \mathbf{y}G^T = \mathbf{0}$.

Atunci, $\mathbf{xy}^T = \mathbf{uGy}^T = \mathbf{u(yG^T)^T} = \mathbf{u0^T} = \mathbf{0}$.

Exemplul 2.5 $(7, 4)$ - codul liniar binar cu matricea generatoare

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

are drept matrice de control

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

care la rândul ei este matricea generatoare a unui $(7, 3)$ - cod liniar binar.

Se verifică imediat relația

$$GH^T = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Un cod care coincide cu codul său dual se numește *cod auto - dual*.

Teorema 2.2 Un cod liniar sistematic cu matricea generatoare $G = (I|B)$ admite ca matrice de control $H = (-B^T|I)$.

Demonstrație: Cele două matrici unitate din scrierea lui G și H sunt de ordin k respectiv $n - k$. Efectuând calculele, se obține:

$$GH^T = (I|B) \begin{pmatrix} -B \\ I \end{pmatrix} = -IB + BI = -B + B = \mathbf{0}. \quad q.e.d.$$

Corolarul 2.1 Matricea de control a unui (n, k) -cod liniar are rangul $n - k$.

Teorema de sus permite un algoritm de calcul extrem de simplu al matricii de control, atunci când se cunoaște matricea generatoare. Să arătăm aceasta pe un exemplu:

Exemplul 2.6 Plecând de la matricea generatoare construită în Exemplul 2.4, se poate construi matricea de control (calculele se fac în Z_3):

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

Aplicând acum permutarea inversă coloanelor lui H^* , se ajunge la matricea de control a codului definit în Exemplul 2.3:

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

Relația $\mathbf{x}H^T = \mathbf{0}$ pe care o verifică orice cuvânt - cod $\mathbf{x} \in A_{n,k}$ permite să definim un cod și sub forma unui sistem de ecuații.

Astfel, $A_{n,k}$ este un cod peste Z_q dacă și numai dacă elementele sale sunt soluții ale sistemului linear $\mathbf{x}H^T = \mathbf{0}$.

Exemplul 2.7 Codul cu matricea de control $H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \end{pmatrix}$

este definit ca mulțimea soluțiilor binare $(x_1, x_2, x_3, x_4, x_5)$ ale sistemului

$$x_1 + x_2 + x_4 = 0, \quad x_1 + x_3 + x_4 + x_5 = 0$$

Sistemul are 8 soluții, care formează codul binar

$$A_{5,2} = \{00000, 00101, 01110, 01011, 11100, 11001, 10010, 10111\}$$

2.3 Sindrom

Fie codul linear $A_{n,k}$ peste Z_q , cu matricea de control H și $\mathbf{a} \in Z_q^n$. Se numește *sindrom* al cuvântului \mathbf{a} vectorul \mathbf{z} cu $n - k$ componente obținut prin relația

$$\mathbf{z}^T = H\mathbf{a}^T,$$

sau - echivalent - $\mathbf{z} = \mathbf{a}H^T$.

Observația 2.3 $\mathbf{z} = \mathbf{0} \iff \mathbf{a} \in A_{n,k}$.

Dacă se transmite printr-un canal de comunicație un cuvânt \mathbf{a} pentru care sindromul corespunzător verifică relația $\mathbf{z} = \mathbf{a}H^T \neq \mathbf{0}$, înseamnă că s-a detectat faptul că în timpul transmisiei au apărut erori.

Teorema 2.3 În Z_2 , sindromul recepționat este egal cu suma coloanelor din matricea de control, corespunzătoare pozițiilor perturbate.

Demonstrație: Fie $\mathbf{a} = (a_1, a_2, \dots, a_n) \in A_{n,k}$ cuvântul-cod transmis. Fără a restrânge generalitatea, să presupunem că au intervenit trei erori, pe pozițiile i, j, k , fiind recepționat vectorul

$$\mathbf{a}' = (a_1, \dots, a_{i-1}, a'_i, a_{i+1}, \dots, a_{j-1}, a'_j, a_{j+1}, \dots, a_{k-1}, a'_k, a_{k+1}, \dots, a_n)$$

unde $a'_i \neq a_i$, $a'_j \neq a_j$, $a'_k \neq a_k$. Avem

$$\mathbf{0} = \mathbf{a}H^T = a_1(h_1) + \dots + a_i(h_i) + \dots + a_j(h_j) + \dots + a_k(h_k) + \dots + a_n(h_n)$$

pentru că $\mathbf{a} \in A_{n,k}$, iar $(h_1), \dots, (h_n)$ sunt coloanele matricii H .

Avem, de asemenea

$$\mathbf{a}'H^T = a_1(h_1) + \dots + a'_i(h_i) + \dots + a'_j(h_j) + \dots + a'_k(h_k) + \dots + a_n(h_n).$$

Prin adunarea acestor două egalități se obține:

$$\mathbf{z}' = \mathbf{a}'H^T = \mathbf{a}H^T + \mathbf{a}'H^T = (0) + \dots + (0) + (h_i) + \dots + (h_j) + \dots + (h_k) + (0) + \dots + (0) = (h_i) + (h_j) + (h_k). \text{ q.e.d.}$$

2.4 Pondere, distanță Hamming

Pentru orice cuvânt $\mathbf{x} \in Z_q^n$, se numește *pondere* numărul $w(\mathbf{x})$ de componente nenule ale lui \mathbf{x} . Evident, $0 \leq w(\mathbf{x}) \leq n$.

Pentru două cuvinte $\mathbf{x}, \mathbf{y} \in Z_q^n$, se numește *distanța Hamming* între ele, numărul

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

Propoziția 2.1 *d este o distanță definită pe Z_q^n .*

Demonstrație: Se verifică ușor proprietățile unei distanțe:

1. $d(\mathbf{x}, \mathbf{x}) = 0$;
2. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$;
3. $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z})$.

Ca o remarcă, deoarece $\mathbf{x} - \mathbf{y} \in Z_q^n$, rezultă că distanța Hamming dintre două cuvinte este ponderea unui cuvânt din Z_q^n .

Cu ajutorul lui d , mulțimea Z_q^n se poate structura ca spațiu metric.

Definiția 2.6 *Se numește distanță (Hamming) a codului liniar $A_{n,k} \subset Z_q^n$ cea mai mică distanță (Hamming) dintre elementele codului $A_{n,k}$, adică*

$$d = \min_{\mathbf{x}, \mathbf{y} \in A_{n,k}, \mathbf{x} \neq \mathbf{y}} d(\mathbf{x}, \mathbf{y})$$

Folosind proprietatea anterioară și faptul că $\mathbf{0} \in A_{n,k}$, rezultă că

$$d = \min_{\mathbf{x} \in A_{n,k}, \mathbf{x} \neq \mathbf{0}} w(\mathbf{x}).$$

Adesea valoarea d este introdusă printre parametrii generali ai codului, folosind notația de (n, k, d) -cod.

Teorema 2.4 *Fie H matricea de control a unui cod liniar $A_{n,k}$. Codul are distanța minimă d dacă și numai dacă orice combinație liniară de $d - 1$ coloane ale lui H este liniar independentă și există cel puțin o combinație liniară de d coloane liniar dependente.*

Demonstrație: Pentru orice cuvânt \mathbf{a} , cu $w(\mathbf{a}) = s$, $\mathbf{a}H^T$ este o combinație liniară de s coloane ale lui H . Cum există un cuvânt - cod de pondere minimă egală cu distanța d a codului, rezultă că avem cel puțin o combinație de d coloane liniar dependente ale lui H . O combinație de mai puțin de d coloane liniar dependente ar conduce la găsirea unui cuvânt - cod de pondere mai mică decât d , deci contradicție. q.e.d.

Definiția 2.7 *Pentru $r > 0$ și $\mathbf{x} \in Z_q^n$, definim sfera de rază r și centru \mathbf{x} ca*

$$S_r(\mathbf{x}) = \{\mathbf{y} \mid d(\mathbf{x}, \mathbf{y}) \leq r\}$$

Teorema 2.5 *Fie $A_{n,k} \subseteq Z_q^n$ un cod liniar cu distanța Hamming d . Dacă $r = \left\lfloor \frac{d-1}{2} \right\rfloor$, atunci*

$$\forall \mathbf{x}, \mathbf{y} \in A_{n,k} \quad [\mathbf{x} \neq \mathbf{y} \implies S_r(\mathbf{x}) \cap S_r(\mathbf{y}) = \emptyset].$$

Demonstrație: Presupunem prin absurd că există $\mathbf{z} \in Z_q^n$, $\mathbf{z} \in S_r(\mathbf{x}) \cap S_r(\mathbf{y})$. Atunci $d(\mathbf{x}, \mathbf{z}) \leq r$, $d(\mathbf{y}, \mathbf{z}) \leq r$ deci, conform inegalității triunghiului, $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{y}, \mathbf{z}) \leq 2r = 2 \left\lfloor \frac{d-1}{2} \right\rfloor < d$, ceea ce contrazice afirmația că d este distanța minimă a codului $A_{n,k}$. q.e.d.

Evident, în fiecare astfel de sferă există un singur cuvânt - cod: cel aflat în centrul sferei.

Definiția 2.8 *Un cod liniar $A_{n,k} \subset Z_q^n$ de distanță minimă d este perfect dacă*

$$Z_q^n = \bigcup_{\mathbf{x} \in A_{n,k}} S_r(\mathbf{x})$$

$$\text{unde } t = \left\lfloor \frac{d-1}{2} \right\rfloor.$$

2.5 Detectare și corectare de erori

Fie $A_{n,k} \subset Z_q^n$ un cod liniar de distanță minimă d și $t = \left\lfloor \frac{d-1}{2} \right\rfloor$. Să presupunem că s-a recepționat cuvântul $\mathbf{z} \in Z_q^n$; va exista cel mult un cuvânt $\mathbf{x} \in A_{n,k}$ astfel încât $\mathbf{z} \in S_t(\mathbf{x})$ (în cazul codurilor perfecte, acest cuvânt există totdeauna).

În cazul (ideal) când $\mathbf{z} = \mathbf{x}$, cuvântul a fost transmis fără erori (sau cu erori nedetectabile).

Dacă $\mathbf{z} \neq \mathbf{x}$, atunci mesajul a fost perturbat (și avem o detectare de erori); în ipoteza că numărul de erori apărute este minim și există un $\mathbf{x} \in A_{n,k}$ astfel ca $d(\mathbf{x}, \mathbf{z}) \leq t$, atunci \mathbf{z} provine din cuvântul - cod \mathbf{x} - și se va transforma în acesta prin corectarea corespunzătoare a erorilor.

În celelalte cazuri, \mathbf{z} sau nu se poate corecta, sau se corectează greșit, în alt cuvânt cod.

Această ultimă situație nu apare la codurile perfecte.

Metoda de detectare și corectare a erorilor descrisă mai sus se numește *decodificarea cea mai probabilă*. Pentru marea majoritate a codurilor liniare aceasta este singura metodă utilizată.

Fie $C \subseteq Z_q^n$ un cod liniar; pentru orice cuvânt $\mathbf{e} \in Z_q^n$ formăm mulțimea

$$V_{\mathbf{e}} = \mathbf{e} + C = \{\mathbf{e} + \mathbf{x} \mid \mathbf{x} \in C\}$$

$V_{\mathbf{e}}$ este mulțimea cuvintelor $\mathbf{w} = \mathbf{e} + \mathbf{x}$ care pot fi recepționate la transmiterea cuvântului - cod \mathbf{x} , atunci când a acționat un vector - eroare \mathbf{e} . \mathbf{e} va fi numit *eroare - tip*.

Se verifică imediat că $V_{\mathbf{e}}$ este subspațiu liniar al lui Z_q^n (a se vedea Observația 2.1).

În particular, $C = \mathbf{0} + C = V_{\mathbf{0}}$.

Propoziția 2.2 *Pentru orice $\mathbf{a} \in V_{\mathbf{e}}$, $V_{\mathbf{a}} = V_{\mathbf{e}}$.*

Demonstrație: Din $\mathbf{a} \in V_{\mathbf{e}}$, rezultă că există $\mathbf{x} \in C$ cu $\mathbf{a} = \mathbf{e} + \mathbf{x}$. Deoarece $\mathbf{x} + C = C$ (evident, C fiind subspațiu liniar), avem $V_{\mathbf{a}} = \mathbf{a} + C = \mathbf{e} + \mathbf{x} + C = \mathbf{e} + C = V_{\mathbf{e}}$. q.e.d.

Vom utiliza această propoziție pentru definirea unei tehnici generale de decodificare.

Dacă vrem să detectăm o anumită eroare - tip \mathbf{e} care modifică cuvintele din C în cuvinte ale subspațiului $V_{\mathbf{e}}$, atunci vor fi mai ușor de depistat cuvintele din $V_{\mathbf{e}}$ cu ponderea minimă.

Pentru fiecare $V_{\mathbf{e}}$ se alege un cuvânt numit *reprezentantul* lui $V_{\mathbf{e}}$; acesta este un element cu cea mai mică pondere nenulă din $V_{\mathbf{e}}$.

Se construiește următorul tablou (numit *tablou standard*):

1. Pe prima linie se scriu cuvintele - cod, începând cu $\mathbf{0}$ (reprezentantul lui $C = V_{\mathbf{0}}$);
2. Pe prima coloană se scriu reprezentanții $\mathbf{0}, \mathbf{e}_1, \mathbf{e}_2, \dots$;
3. Pe linia cu reprezentantul \mathbf{e}_i , sub cuvântul cod \mathbf{x}_j se scrie cuvântul $\mathbf{e}_i + \mathbf{x}_j \pmod{q}$.

Exemplul 2.8 Fie matricea generatoare $G = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$ peste Z_2 . Ea va codifica $Z_2^2 = \{00, 01, 10, 11\}$ în $A_{4,2} = \{0000, 1001, 0111, 1110\}$.

Calculul mulțimilor $V_{\mathbf{e}}$ conduce la

$$\begin{aligned} V_{0000} &= V_{1001} = V_{0111} = V_{1110} \\ V_{1000} &= V_{0001} = V_{0110} = V_{1111} \\ V_{0100} &= V_{1101} = V_{0011} = V_{1010} \\ V_{0010} &= V_{1011} = V_{0101} = V_{1100} \end{aligned}$$

Alegem ca reprezentanți pe 0000, 1000 (se poate și 0001), 0100, 0010. Tabloul standard va fi:

0000	1001	0111	1110
1000	0001	1111	0110
0100	1101	0011	1010
0010	1011	0101	1100

Pentru destinatar, acest tablou este ca un dicționar care se utilizează astfel: cuvântul primit \mathbf{a} se decodifică în cuvântul - cod din capul coloanei pe care se află \mathbf{a} .

De exemplu, dacă se recepționează 1101, el se va decodifica în 1001.

Evident, cuvintele de pe prima coloană se decodifică în ele însele (sunt cuvinte - cod și nu au fost perturbate de nici o eroare).

Pentru orice cod liniar, un dicționar complet de tipul celui de mai sus constituie cea mai simplă metodă de decodificare.

Problema apare atunci când într-o mulțime $V_{\mathbf{e}}$ sunt mai multe cuvinte de pondere minimă. Atunci tabela va decodifica corect numai eroarea - tip aleasă ca reprezentant.

Astfel, revenind la Exemplul 2.8, cuvântul recepționat 1111 se decodifică în 0111 (considerând că a fost alterat primul caracter).

Dacă se ia însă drept reprezentant pe linia a doua 0001 în loc de 1000, a doua linie din tabloul standard este

0001 1000 0110 1111

Atunci, 1111 se decodifică în 1110 (considerînd ultimul caracter ca fiind cel alterat de canalul de transmisie). Care este cuvântul - cod corect transmis ?

Acest lucru nu poate fi decis. Singurul lucru care poate fi făcut este să se aleagă drept reprezentanți erorile - tip cele mai probabile.

Pentru un (n, k) - cod peste Z_q , un dicționar complet constă din toate cele q^n cuvinte posibile, lucru destul de dificil deoarece în practică codurile sunt destul de lungi ($n \geq 100$).

De aceea este utilizată o altă manieră de lucru care reduce mult mărimea tabloului standard.

Fie H matricea de control a unui (n, k) - cod liniar $A_{n,k}$; putem considera (Corolarul 2.1) că liniile lui H sunt vectori liniar independenți.

Teorema 2.6 Pentru orice $\mathbf{e} \in Z_q^n$, toate cuvintele din $V_{\mathbf{e}}$ au același sindrom.

Demonstrație: Fie $\mathbf{v} \in A_{n,k}$ arbitrar și $\mathbf{w} = \mathbf{e} + \mathbf{v}$. Avem

$$\mathbf{s} = \mathbf{w}H^T = (\mathbf{e} + \mathbf{v})H^T = \mathbf{e}H^T + \mathbf{v}H^T = \mathbf{e}H^T + \mathbf{0} = \mathbf{e}H^T.$$

Deci toate cuvintele din $V_{\mathbf{e}}$ au sindromul egal cu sindromul lui \mathbf{e} . q.e.d.

Invers, pentru fiecare sindrom - deci pentru fiecare cuvânt \mathbf{s} de lungime $n - k$, se poate determina un vector - eroare \mathbf{e} având sindromul \mathbf{s} .

Mai mult, \mathbf{s} va fi ales astfel încât să aibă pondere minimă (conform decodificării cele mai probabile). Pentru aceasta, se rezolvă sistemul de ecuații liniare $H\mathbf{e}^T = \mathbf{s}^T$, care are soluție, deoarece liniile lui H sunt liniar independente.

Din mulțimea soluțiilor alegem una de pondere minimă, cu ajutorul căreia construim mulțimea $V_{\mathbf{e}}$ a tuturor cuvintelor de sindrom \mathbf{s} .

Pe baza celor de mai sus, se poate folosi următoarea procedură de decodificare:

1. La recepționarea unui cuvânt \mathbf{w} , se calculează sindromul \mathbf{s} :
$$\mathbf{s}^T = H\mathbf{w}^T.$$
2. Se află eroarea - tip \mathbf{e} cu sindromul \mathbf{s} .
3. Se consideră cuvântul - cod corect ca fiind $\mathbf{v} = \mathbf{w} - \mathbf{e}$.

În acest fel, nu mai este necesar să se rețină tot tabloul standard; este suficient să se știe doar reprezentanții subspațiilor $V_{\mathbf{e}}$ și sindromurile corespunzătoare.

Exemplul 2.9 Reluând codul definit în Exemplul 2.8, el are ca matrice de control

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

Calculând sindromurile reprezentanților, se ajunge la tabloul:

<i>Sindrom</i>	<i>Reprezentant</i>
00	0000
01	1000
10	0010
11	0100

care reprezintă o reducere cu 50% a datelor stocate (comparativ cu tabloul standard).
De exemplu, la recepționarea cuvântului 1101, se calculează sindromul

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

Reprezentantul sindromului 11 este 0100. Efectuând operația

$$1101 - 0100 = 1101 + 0100 = 1001$$

(în Z_2 scăderea este de fapt adunare) se ajunge la cuvântul transmis, anume 1001.

Exemplul 2.10 Să considerăm codul liniar $A_{5,3}$ peste Z_3 definit de matricea de control

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

Sindromurile sunt cuvinte de lungime 2 peste Z_3 , deci în total nouă cuvinte. Lista sindromurilor și a reprezentanților este:

<i>Sindrom</i>	<i>Reprezentant</i>
00	00000
10	10000
01	01000
22	10010
11	11000
02	00001
20	20000
12	00010
21	00100

De remarcat că un tablou standard pentru acest cod are dimensiunea 9×27 și conține 243 cuvinte; volumul de date s-a redus deci cu 92%.

Să presupunem că s-a trimis cuvântul cod 01221 și s-a recepționat 01201.

Sindromul este

$$\begin{pmatrix} 1 & 0 & 2 & 1 & 0 \\ 0 & 1 & 1 & 2 & 2 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 2 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

deci $\mathbf{e} = 00010$; decodificarea este

$$\mathbf{v} = \mathbf{w} - \mathbf{e} = 01201 - 00010 = 01221.$$

Decodificarea a fost corectă deoarece eroarea - tip apărută a fost una din cele selectate prin reprezentanți.

2.6 Exerciții

2.1 Să se construiască coduri liniare care să transforme cuvântul $(a_1, a_2, \dots, a_n) \in Z_q^n$ în:

1. $(a_1, a_1, a_1, a_2, a_2, a_2, \dots, a_n, a_n, a_n)$;
2. $(a_1, b_1, a_2, b_2, \dots, a_n, b_n)$ unde $b_1 = a_1, b_2 = a_1 + a_2, b_n = a_1 + \dots + a_n$;
3. $(a_1, 2a_n, 3a_2, 4a_{n-1}, \dots)$.

2.2 Un cod liniar peste Z_5 are următoarea matrice generatoare:

$$G = \begin{pmatrix} 1 & 2 & 3 & 1 & 2 \\ 2 & 2 & 4 & 1 & 0 \\ 1 & 1 & 2 & 2 & 1 \end{pmatrix}$$

Să se afle matricea de control.

2.3 Aceeași problemă pentru Z_7 .

2.4 Determinați dacă următorul cod liniar binar este sistematic sau nu:

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

Dacă nu, găsiți un cod sistematic echivalent.

2.5 Găsiți matricea generatoare a unui cod liniar binar care are matricea de control:

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

2.6 Se dă matricea de control

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

a unui cod liniar binar.

Să se decodifice mesajele 110110, 010100.

2.7 Descrieți dualul $(6, 3)$ - codului binar descris de ecuațiile:

$$x_1 = x_4, \quad x_2 = x_5, \quad x_3 = x_6.$$

2.8 Fie A un cod liniar binar obținut din toate sumele posibile ale cuvintelor 101011, 011101, 011010.

1. Aflați o matrice de control a codului.
2. Aflați un tablou standard și decodificați 111011.

2.9 Demonstrați că codul liniar binar descris de ecuațiile

$$x_3 = x_1 + x_2, \quad x_4 = x_1, \quad x_5 = x_1 + x_2$$

corectează o eroare. Construiți tabloul standard.

2.10 Construiți o tabelă de sindromuri de decodificare pentru:

1. Codul cu repetiție de lungime 7;
2. Codul din Exercițiul 2.7;
3. Codul peste Z_3 cu matricea generatoare:

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

2.11 Fie un cod liniar $A_{n,k} \subseteq Z_q^n$ și tabela de sindromuri de decodificare corespunzătoare. Definim o operație de decodificare $\phi : Z_q^n \rightarrow A_{n,k}$ cu proprietatea: $\forall \mathbf{v} \in A_{n,k}, \forall \mathbf{e}$ reprezentant din tabelă, $\phi(\mathbf{v} + \mathbf{e}) = \mathbf{v}$. (ϕ corectează erorile - tip din tabela standard). Să se arate că atunci ϕ nu corectează nici o altă eroare - tip: dacă \mathbf{e} nu este un reprezentant din tabelă, atunci există $\mathbf{v} \in A_{n,k}$ cu $\phi(\mathbf{v} + \mathbf{e}) \neq \mathbf{v}$.

(Altfel spus, decodificarea cu sindromuri este optimală).

Capitolul 3

Coduri liniare - II

3.1 Capacități de detectare și corectare a erorilor

După cum am văzut până acum, un cod liniar este un spațiu liniar, codul dual este complementul său ortogonal etc. Ceea ce interesează aici este depistarea unor coduri cu proprietăți deosebite în detectarea și corectarea erorilor. Aceste proprietăți sunt legate în special de distanța minimă a codului. Vom prezenta pentru început câteva rezultate legate de diverse margini relativ la distanță, numărul de simboluri de control, informație etc.

Teorema 3.1 *Un cod liniar $A_{n,k} \subseteq Z_q^n$ are distanța minimă d dacă și numai dacă poate detecta orice combinație de maxim $d - 1$ erori.*

Demonstrație: Dacă se transmite un cuvânt $\mathbf{a} \in A_{n,k}$ și apar t ($0 \leq t < d$) erori, se va recepționa cuvântul $\mathbf{a} + \mathbf{e} \in Z_q^n$ cu $w(\mathbf{e}) = t$.

Deoarece $d(\mathbf{a}, \mathbf{a} + \mathbf{e}) = t$, rezultă $\mathbf{a} + \mathbf{e} \notin A_{n,k}$, deci se detectează eroare.

Afirmația reciprocă este evidentă. q.e.d.

Teorema 3.2 *Un cod liniar $A_{n,k} \subseteq Z_q^n$ are distanța minimă d dacă și numai dacă poate corecta orice combinație de $t \leq \left\lfloor \frac{d-1}{2} \right\rfloor$ erori.*

Demonstrație: Fie t un număr întreg pozitiv; să presupunem că $d \leq 2t$ și vom arăta că există erori - tip de pondere t (numite și *pachete de t erori independente*) care nu pot fi corectate de codul $A_{n,k}$.

Aceasta va duce la concluzia că $d \geq 2t + 1$, adică $A_{n,k}$ poate corecta orice pachet de $t \leq \left\lfloor \frac{d-1}{2} \right\rfloor$ erori.

Fie $\mathbf{a}, \mathbf{b} \in A_{n,k}$ cu $d(\mathbf{a}, \mathbf{b}) = d$ și i_1, i_2, \dots, i_d toți indicii în care \mathbf{a} diferă de \mathbf{b} . Alegând $\frac{d}{2} \leq t \leq \frac{d+1}{2}$, avem $d \leq 2t$ (t astfel ales este minim).

Să presupunem că se trimite \mathbf{a} și se recepționează cuvântul $\mathbf{a}' = (a'_1, a'_2, \dots, a'_n)$ unde

$$a'_i = \begin{cases} a_i (= b_i) & \text{dacă } i \neq i_1, i_2, \dots, i_d, \\ b_i & \text{dacă } i = i_1, i_3, \dots, \\ a_i & \text{dacă } i = i_2, i_4, \dots \end{cases}$$

Atunci, evident $d(\mathbf{a}, \mathbf{a}') = \left\lceil \frac{d+1}{2} \right\rceil = t$ și $d(\mathbf{a}', \mathbf{b}) = \left\lfloor \frac{d}{2} \right\rfloor \leq t = d(\mathbf{a}', \mathbf{a})$. Aceasta va duce la decodificarea lui \mathbf{a} în \mathbf{b} - incorect.

Să presupunem acum $d \geq 2t + 1$. Atunci codul $A_{n,k}$ poate corecta orice pachet de t erori.

Pentru a arăta aceasta, să presupunem că se trimite un cuvânt - cod \mathbf{a} și se primește un cuvânt \mathbf{a}' cu $d(\mathbf{a}, \mathbf{a}') \leq t$. Pentru orice cuvânt - cod \mathbf{b} (deci cu $d(\mathbf{a}, \mathbf{b}) \geq d \geq 2t + 1$) avem, conform inegalității triunghiului:

$$d(\mathbf{a}, \mathbf{a}') + d(\mathbf{a}', \mathbf{b}) \geq d(\mathbf{a}, \mathbf{b}) \geq 2t + 1$$

de unde rezultă $d(\mathbf{a}', \mathbf{b}) \geq 2t + 1 - d(\mathbf{a}, \mathbf{a}') \geq 2t + 1 - t = t + 1 > d(\mathbf{a}', \mathbf{a})$.

Deci, în ipoteza celei mai probabile decodificări, \mathbf{a}' se va decodifica în \mathbf{a} . q.e.d.

Teorema 3.3 *Distanța minimă a unui (n, k) - cod liniar verifică relația*

$$d \leq n - k + 1.$$

Demonstrație: Vom separa demonstrația în două părți:

A: Fie $A_{n,k}$ un cod sistematic. Atunci primele k simboluri din orice cuvânt - cod pot fi alese arbitrar (ele formează mesajul de informație, un element arbitrar din Z_q^k). Fie $\mathbf{v} \in A_{n,k}$ cuvântul de forma $\mathbf{v} = 100 \dots 0v_{k+1}v_{k+2} \dots v_n$. Evident, $0 < w(\mathbf{v}) \leq n - k + 1$.

Cum d este cea mai mică pondere a unui cuvânt - cod nenul, rezultă inegalitatea cerută.

B: Fie $A_{n,k}$ un cod liniar arbitrar și $A'_{n,k}$ codul liniar sistematic echivalent. Se observă că cele două coduri au aceiași parametri n, k, d . Folosind acum **A**, inegalitatea se obține din nou. q.e.d.

Teorema 3.4 *În orice cod liniar $A_{n,k} \subseteq Z_q^n$ avem*

$$d \leq \frac{nq^{k-1}(q-1)}{q^k-1} \quad (\text{marginea Plotkin})$$

Demonstrație: Să considerăm elementele din $A_{n,k}$ așezate ca linii ale unui tablou. Se obține un tablou cu q^k linii și n coloane. Fiecare componentă nenulă din Z_q apare pe fiecare coloană în q^{k-1} linii. Atunci, suma ponderilor tuturor cuvintelor - cod este egală cu $nq^{k-1}(q-1)$ deoarece fiecare componentă nenulă apare de q^{k-1} ori în fiecare coloană și avem $q-1$ componente nenule distribuite pe n coloane.

Distanța minimă a codului nu poate să depășească ponderea medie a cuvintelor codului, adică

$$d \leq \frac{nq^{k-1}(q-1)}{q^k-1}$$

deoarece în $A_{n,k}$ sunt $q^k - 1$ cuvinte nenule. q.e.d.

Teorema 3.5 *Fie $A_{n,k}$ un cod liniar peste Z_q care corectează orice combinație de maxim t erori. Într-un asemenea cod sunt necesare cel puțin*

$$n - k \geq \log_q[1 + C_n^1(q-1) + C_n^2(q-1)^2 + \dots + C_n^t(q-1)^t]$$

poziții de control (marginea Hamming).

Demonstrație: Pentru ca $A_{n,k}$ să corecteze orice combinație de cel mult t erori, este necesar ca fiecare astfel de eroare - tip să fie un reprezentant în tabloul standard; deci să fie caracterizată printr-un sindrom distinct.

Sunt q^{n-k} sindromuri distincte, deci acesta este numărul maxim de erori care pot fi corectate de cod. Din cele q^{n-k} sindromuri, 1 trebuie să fie pentru 0 erori, $C_n^1(q-1)$ - pentru erori - tip simple (cuvinte \mathbf{e} cu o singură componentă nenulă), $C_n^2(q-1)^2$ - pentru erori duble etc.

Deci, este necesar ca

$$q^{n-k} \geq 1 + C_n^1(q-1) + C_n^2(q-1)^2 + \dots + C_n^t(q-1)^t.$$

Apoi se logaritmează. q.e.d.

Teorema 3.6 *Dacă*

$$n - k \geq \log_q[1 + C_{n-1}^1(q-1) + C_{n-1}^2(q-1)^2 + \dots + C_{n-1}^{d-2}(q-1)^{d-2}]$$

atunci există un cod liniar $A_{n,k} \subseteq Z_q^n$ cu distanța minimă d (marginea Varșamov - Gilbert).

Demonstrație: Pentru ca să existe un cod liniar $A_{n,k}$ peste Z_q cu distanța minimă d este suficient (Teorema 2.4) ca orice coloană din matricea de control $H_{n-k,n}$ să nu fie combinație liniară a altor $d-2$ coloane, în acest fel ne-existând nici o combinație liniară între $d-1$ coloane ale lui H .

Această condiție este echivalentă cu

$$q^{n-k} - 1 \geq C_{n-1}^1(q-1) + C_{n-1}^2(q-1)^2 + \dots + C_{n-1}^{d-2}(q-1)^{d-2}.$$

Aici, $q^{n-k} - 1$ reprezintă numărul total de coloane distincte nenule care pot apare în matricea H . Semnificația termenilor din membrul drept este evidentă; astfel, de exemplu $C_{n-1}^2(q-1)^2$ reprezintă numărul combinațiilor liniare cu coeficienți nenuli a două din cele $n-1$ coloane etc.

Apoi se logaritmează. q.e.d.

Fie $A_{n,k} \subseteq Z_q^n$ un cod liniar pentru care $d \geq 2t + 1$ (deci cu capacitatea de a corecta orice combinație de maxim t erori independente). Reamintim că în prelegerea precedentă am definit pentru orice $\mathbf{x} \in A_{n,k}$ sfera centrată în \mathbf{x} prin

$$S_t(\mathbf{x}) = \{\mathbf{y} \in Z_q^n \mid d(\mathbf{x}, \mathbf{y}) \leq t\}.$$

Mai introducem și suprafața (scoarța) acestei sfere, definită:

$$A_t(\mathbf{x}) = \{\mathbf{y} \in Z_q^n \mid d(\mathbf{x}, \mathbf{y}) = t\}.$$

Vom nota numărul de elemente ale fiecăreia din cele două mulțimi prin

$$S_t = |S_t(\mathbf{x})|, \quad A_t = |A_t(\mathbf{x})|$$

(datorită proprietății de omogenitate a spațiilor liniare, valorile S_t și A_t sunt aceleași pentru orice $\mathbf{x} \in Z_q^n$).

Au loc relațiile evidente:

$$A_t \leq S_t, \quad S_t = \sum_{i=0}^t A_i.$$

Deoarece sferele de rază t centrate în cuvintele codului $A_{n,k}$ sunt disjuncte, avem

$$q^k S_t \leq q^n.$$

Aici, q^k reprezintă numărul de cuvinte - cod, iar q^n - numărul total de cuvinte din Z_q^n .
Din această relație se obține imediat

$$n - k \geq \log_q S_t,$$

cunoscută sub numele de *inegalitatea volumului*.

Ea mai poate fi găsită și sub forma

$$\frac{k}{n} \leq 1 - \frac{1}{n} \log_q S_t.$$

Raportul $\frac{k}{n}$ se numește *rata de informație* și dă o măsură a cantității de informație pe care o poartă un cuvânt - cod. O rată de informație mică (mai multe simboluri de control) asigură o securitate mai mare a datelor transmise. În schimb, condiții practice de eficiență cer o rată de informație cât mai mare (mai multă informație pe unitatea de mesaj). Aceasta este una din solicitările contradictorii ale teoriei codurilor.

3.2 Modificări ale codurilor liniare

Adesea este imposibil să se utilizeze un cod bun deoarece el nu satisface anumite restricții tehnice, cum ar fi lungimea sau rata de informație. De aceea este util să se aplice anumite prelucrări asupra codurilor, care să nu afecteze proprietățile principale de detectare și corectare a erorilor.

Definiția 3.1 Numim *extensie* a unui (n, k) - cod liniar $A_{n,k} \subseteq Z_q^n$, $(n+1, k)$ - codul liniar $A_{n+1,k}^*$ obținut din $A_{n,k}$ prin adăugarea la fiecare cuvânt - cod $a_1 a_2 \dots a_n$ a unui

simbol nou a_{n+1} , cu proprietatea $\sum_{i=1}^{n+1} a_i = 0 \pmod{q}$.

Observația 3.1

- În cazul binar, noul caracter a_{n+1} poartă numele bit de paritate.

Dacă H este matricea de control a codului $A_{n,k}$, atunci codul extins $A_{n+1,k}^*$ are matricea de control

$$H^* = \left(\begin{array}{cccc|c} & & & & 0 \\ & & & & \vdots \\ & & & & 0 \\ \hline 1 & 1 & \dots & 1 & 1 \end{array} \right)$$

De fapt, ultima linie reprezintă ecuația $\sum_{i=1}^n x_i = (q-1)x_{n+1}$.

- Dacă un cod liniar binar $A_{n,k}$ are o distanță minimă impară d , atunci codul extins are distanța minimă $d + 1$. Într-adevăr, fie $\mathbf{a} = a_1 a_2 \dots a_n \in A_{n,k}$ cu $w(\mathbf{a}) = d$. Cum d este impar, rezultă $\sum_{i=1}^n a_i = 1$ (în Z_2), deci $a_{n+1} = 1$. Cuvântul $\mathbf{a}' = a_1 a_2 \dots a_n 1 \in A_{n+1,k}^*$, $w(\mathbf{a}') = d + 1$ și nu se poate construi un alt cuvânt - cod în $A_{n+1,k}^*$ de pondere mai mică.

Definiția 3.2 Fie $A_{n,k} \subseteq Z_q^n$.

1. "Relaxarea" lui $A_{n,k}$ este un cod liniar $\bar{A}_{n-1,k}$ obținut prin ștergerea ultimului simbol din cuvintele lui $A_{n,k}$;
2. "Completarea" lui $A_{n,k}$ este un cod definit $A_{n,k}^* = A_{n,k} \cup (A_{n,k} + \mathbf{1})$ (unde $\mathbf{1}$ este cuvântul cu toate elementele 1, iar suma se face modulo q);
3. "Expurgarea" lui $A_{n,k}$ este codul $A'_{n,k} = \{\mathbf{a} \in A_{n,k} \mid w(\mathbf{a}) \equiv 0 \pmod{2}\}$.

Observația 3.2

- Relaxarea este operația inversă extensiei.
- Prin completarea și expurgarea codurilor liniare se obțin coduri liniare numai în cazul binar. În celelalte cazuri, noile mulțimi rezultate nu sunt spații liniare.

Propoziția 3.1 Prin completarea unui cod liniar binar $A_{n,k}$ se obține un cod liniar binar $A_{n,k+1}$ cu un număr dublu de cuvinte - cod.

Demonstrație: Completarea unui cod binar C înseamnă adăugarea la cuvintele - cod ale lui C a tuturor cuvintelor obținute prin complementare (schimbarea lui 0 în 1 și a lui 1 în 0).

Fie $G_{k,n}$ matricea generatoare a codului $A_{n,k}$. Se verifică ușor că matricea

$$G'_{k+1,n} = \begin{pmatrix} & & G & & \\ 1 & 1 & \dots & 1 & 1 \end{pmatrix}$$

generează $A_{n,k} \cup (\mathbf{1} + A_{n,k})$. Acest cod are $k + 1$ poziții de informație și lungime n . Fiecare din cele două submulțimi are un număr egal de elemente. q.e.d.

Propoziția 3.2 Orice cod liniar binar are sau toate cuvintele - cod de pondere pară, sau numărul cuvintelor - cod de pondere pară este egal cu al celor de pondere impară.

Demonstrație: Fie C un cod liniar binar cu un cuvânt \mathbf{v}_1 de pondere impară. Să presupunem că $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_s$ sunt toate cuvintele lui C ; atunci $C = C + \mathbf{v}_1$.

Pentru orice cuvânt - cod \mathbf{v}_i de pondere pară (impară), $\mathbf{v}_1 + \mathbf{v}_i$ are pondere impară (pară). Pentru a justifica această afirmație, să presupunem că $w(\mathbf{v}_1) = 2p + 1$, $w(\mathbf{v}_i) = 2q$ iar \mathbf{v}_i și \mathbf{v}_1 au 1 pe r poziții comune. Atunci $w(\mathbf{v}_1 + \mathbf{v}_i) = w(\mathbf{v}_i) + w(\mathbf{v}_1) - 2r$ (pentru că $1 + 1 = 0$) $= 2p + 1 + 2q - 2r = 2s + 1$.

Similar se argumentează și în cazul când \mathbf{v}_i are pondere impară.

Deci adunarea cu \mathbf{v}_1 definește o corespondență biunivocă între cuvintele - cod de pondere pară și cele de pondere impară, ceea ce completează demonstrația. q.e.d.

Corolarul 3.1 *Expurgarea unui cod liniar binar C este tot C sau un cod liniar binar având ca elemente jumătate din elementele lui C .*

Matricea generatoare G_{exp} a lui C_{exp} se poate obține din matricea G a lui C astfel: dacă toate liniile lui G sunt vectori de pondere pară, atunci cele două coduri coincid.

Altfel, fie $G = [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_s, \mathbf{e}_{s+1}, \dots, \mathbf{e}_k]^T$ în care - fără a micșora generalitatea, putem presupune că primele s au pondere impară, iar celelalte $k - s$ au pondere pară. Atunci $G_{exp} = [\mathbf{0}, \mathbf{e}_2 + \mathbf{e}_1, \dots, \mathbf{e}_s + \mathbf{e}_1, \mathbf{e}_{s+1}, \dots, \mathbf{e}_k]^T$. q.e.d.

Exemplul 3.1 *Să construim codul $A_{4,2}$ peste Z_3 de matrice generatoare*

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

Ea codifică cele 9 elemente din Z_3^2 în

$$A_{4,2} = \{0000, 0111, 0222, 1010, 1121, 1202, 2020, 2101, 2212\}.$$

Codul liniar relaxat $A_{3,2} = \{000, 011, 022, 101, 112, 120, 202, 210, 221\}$ este generat de matricea

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

Construcția a fost posibilă deoarece prin eliminarea ultimei coloane, liniile rămase sunt tot liniar independente.

Dacă acest lucru nu este realizabil, se caută k cuvinte - cod în $A_{n,k}$ cu proprietatea că după eliminarea ultimei componente, ele sunt liniar independente. Acestea formează liniile noii matrici generatoare.

Codul completat este

$$\begin{array}{cccccccccc} 0000 & 0111 & 0222 & 1010 & 1121 & 1202 & 2020 & 2101 & 2212 \\ 1111 & 1222 & 1000 & 2121 & 2202 & 2010 & 0101 & 0212 & 0020 \end{array}$$

De remarcat că el nu este un spațiu liniar (nu este închis la adunarea din Z_3).

Codul expurgat are cinci elemente: $\{0000, 1010, 1121, 2020, 2212\}$. Nici acesta nu este cod liniar.

Exemplul 3.2 *Să reluăm matricea generatoare din Exemplul 19.4, dar pentru un cod liniar peste Z_2 . Codul generat de G este $A_{4,2} = \{0000, 0111, 1010, 1101\}$.*

Toate codurile modificate sunt în acest caz coduri liniare. Astfel

- *Codul relaxat $A_{rel} = \{000, 011, 101, 110\}$ este generat de aceeași matrice G_{rel} din Exemplul 19.4.*
- *Codul completat $A_{com} = \{0000, 0111, 1010, 1101, 1111, 1000, 0101, 0010\}$ este un cod liniar generat de matricea*

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

- *Codul expurgat $A_{exp} = \{0000, 1010\}$ este un cod liniar generat de matricea*

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

3.3 Detectarea și corectarea simultană a erorilor

Să începem cu un exemplu.

Exemplul 3.3 Fie $(7, 4)$ - codul liniar binar având matricea de control

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

El are distanța minimă $d = 3$, deci poate detecta 2 erori și poate corecta o eroare (Teoremele 19.1, 3.2). Totuși, codul nu poate realiza simultan ambele deziderate.

Mai precis, atunci când codul este utilizat pentru corectare de erori, erorile duble scapă nedetectate. Astfel, dacă se trimite 0000000 și se recepționează 1010000, sindromul este 010. Tabloul standard conduce la corectarea celui de-al doilea bit, și decodifică (incorect) în cuvântul - cod 111000.

Uneori însă, se solicită în mod explicit un cod capabil să detecteze și să corecteze erori în același timp.

Definiția 3.3 Un cod C de lungime n corectează t erori și detectează s erori simultan dacă orice cuvânt - cod \mathbf{v} are următoarea proprietate:

$$\forall \mathbf{x} \in Z_q^n [d(\mathbf{x}, \mathbf{v}) \leq s \implies \forall \mathbf{a} \in C \setminus \{\mathbf{v}\}, d(\mathbf{x}, \mathbf{a}) > t].$$

În această situație, detectarea și corectarea simultană a erorilor se realizează astfel: la recepționarea unui cuvânt $\mathbf{x} \in Z_q^n$ se caută cel mai apropiat cuvânt - cod \mathbf{v} (în sensul distanței Hamming). Dacă $d(\mathbf{x}, \mathbf{v}) \leq t$ atunci cuvântul se corectează în \mathbf{v} ; altfel, se anunță că cel puțin s simboluri sunt modificate.

Justificarea acestui procedeu rezultă imediat din definiție.

Teorema 3.7 Un cod liniar corectează t erori și detectează s erori simultan dacă și numai dacă

$$d \geq t + s + 1.$$

Demonstrație: **A:** Să presupunem $d \geq t + s + 1$.

Fie \mathbf{v} un cuvânt - cod și $\mathbf{x} \in Z_q^n$ cu $d(\mathbf{v}, \mathbf{x}) \leq s$. Pentru orice cuvânt - cod \mathbf{v}' ($\mathbf{v}' \neq \mathbf{v}$) avem $d(\mathbf{v}, \mathbf{v}') \geq d \geq t + s + 1$.

Folosind inegalitatea triunghiului,

$$d(\mathbf{v}, \mathbf{x}) + d(\mathbf{x}, \mathbf{v}') \geq d(\mathbf{v}, \mathbf{v}') \geq t + s + 1,$$

se deduce

$$d(\mathbf{x}, \mathbf{v}') \geq t + s + 1 - d(\mathbf{v}, \mathbf{x}) \geq t + s + 1 - s = t + 1.$$

Deci, condiția din Definiția 20.3 este îndeplinită.

B: Să presupunem prin absurd $d < t + s + 1$.

Fie \mathbf{v}, \mathbf{v}' două cuvinte - cod cu $d(\mathbf{v}, \mathbf{v}') = d \leq t + s$. Construim cuvântul \mathbf{x} din \mathbf{v} în felul următor: dacă $d \leq s$ atunci $\mathbf{x} = \mathbf{v}'$; altfel se înlocuiesc primele s simboluri în care acesta diferă de \mathbf{v}' , cu valorile lor din \mathbf{v}' . Atunci $d(\mathbf{v}, \mathbf{x}) = s$ și $d(\mathbf{v}', \mathbf{x}) = d - s \leq t + s - s = t$, ceea ce contrazice condiția din Definiția 20.3. q.e.d.

Exemplul 3.4 Să considerăm $(8, 4)$ - codul liniar binar generat de matricea

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

El are distanța minimă $d = 4$, deci - conform Teoremei 3.2 poate corecta maxim o eroare, iar conform Teoremei 3.7 poate corecta o eroare și detecta simultan două erori.

Astfel recepționarea cuvântului 11110010 conduce la corectarea sa în 10110010 (deoarece $d(11110010, 10110010) = 1$ și 10110010 este cuvânt - cod).

În schimb recepționarea cuvântului 00001111 anunță că au apărut cel puțin două erori. În această situație, nu mai puțin de patru cuvinte - cod (00010111, 00101011, 01001101, 10001110) sunt situate la distanța 2 de cuvântul primit.

Exemplul 3.5 Codul binar cu repetiție de lungime 7 (care are 2 elemente: 0000000 și 1111111) poate realiza una din condițiile:

- Corectează 3 erori;
- Detectează 6 erori;
- Corectează 2 erori și detectează 4 erori simultan.

3.4 Probabilitatea nedectării erorilor

Să ne punem următoarea problemă: care este probabilitatea ca la transmiterea unui cuvânt - cod \mathbf{a} să fie recepționat alt cuvânt - cod \mathbf{b} ($\mathbf{b} \neq \mathbf{a}$). Altfel spus, care este probabilitatea ca o eroare să scape nedectată ?

Notând cu $\mathbf{e} = \mathbf{b} - \mathbf{a}$, o eroare este nedectată dacă și numai dacă \mathbf{e} este un cuvânt - cod nenul.

Vom considera un canal de transmisie *binar simetric*, adică un canal în care singurele simboluri transmise sunt 0 și 1, iar probabilitatea p ($0 \leq p \leq 1$) ca la transmiterea lui 0 să fie recepționat 1 este egală cu probabilitatea ca la transmiterea lui 1 să se recepționeze 0.

Într-un astfel de canal, dacă $w(\mathbf{e}) = i$ (adică au fost perturbate la transmisie i caractere), probabilitatea de apariție a erorii - tip \mathbf{e} este $p^i q^{n-i}$, unde $q = 1 - p$.

Notând cu A_i numărul cuvintelor - cod cu ponderea i , probabilitatea P_{ned} a unei erori nedectabile este suma probabilităților $p^i q^{n-i}$, fiecare termen apărând de A_i ori pentru $i = 1, 2, \dots, n$.

Formal,

$$P_{ned} = \sum_{i=1}^n A_i p^i q^{n-i}$$

Cum $A_1 = A_2 = \dots = A_{d-1} = 0$, suma se reduce la $P_{ned} = \sum_{i=d}^n A_i p^i q^{n-i}$.

Exemplul 3.6 Să considerăm un $(7, 4)$ - cod liniar binar definit în Exemplul 19.6. Cele $2^4 = 16$ cuvinte - cod ale sale sunt: un cuvânt de pondere 0, câte șapte cuvinte de pondere 3 și respectiv 4 și un cuvânt de pondere 7. Atunci

$$P_{ned} = 7p^3q^4 + 7p^4q^3 + p^7$$

Dacă folosim acest cod într-un canal binar simetric cu eroare de probabilitate $p = 0.01$, avem

$$P_{ned} = 7(0.01)^3(0.99)^4 + 7(0.01)^4(0.99)^3 + (0.01)^7 \approx 7 \times 10^{-6}$$

deci - în medie - apar cam șapte erori nedetectabile la un milion de cuvinte transmise.

Definiția 3.4 Polinomul de variabilă $x \in [0, 1]$ definit

$$P(x) = \sum_{i=0}^n A_i x^i$$

unde A_i este numărul de cuvinte de pondere i din codul liniar $A_{n,k}$, se numește "numărătorul de ponderi" al codului $A_{n,k}$.

Exemplul 3.7 Codul definit în Exemplul 19.5 are numărătorul de ponderi

$$P(x) = 1 + x^2 + 2x^3$$

Propoziția 3.3 Fie $A_{n,k}$ un cod liniar binar cu numărător de ponderi $P(x)$. Probabilitatea apariției unei erori nedetectabile la folosirea codului $A_{n,k}$ într-un canal binar simetric este

$$P_{ned} = q^n \left[P\left(\frac{p}{q}\right) - 1 \right].$$

Demonstrație: Relația de definiție a lui P_{ned} se poate rescrie

$$q^n \sum_{i=1}^n A_i p^i q^{-i} = q^n \sum_{i=1}^n A_i \left(\frac{p}{q}\right)^i.$$

Deoarece $A_0 = 1$ (singurul cuvânt - cod de pondere 0 este cuvântul - cod $\mathbf{0}$), expresia devine în continuare

$$P_{ned} = q^n \left[\sum_{i=0}^n A_i \left(\frac{p}{q}\right)^i - 1 \right] = q^n \left[P\left(\frac{p}{q}\right) - 1 \right]. \quad \text{q.e.d.}$$

3.5 Identitatea MacWilliams

În acest paragraf vom arăta un rezultat care face posibilă determinarea numărătorului de ponderi $P_{C^\perp}(x)$ al dualului C^\perp unui cod liniar C , direct din numărătorul de ponderi $P_C(x)$ al codului C .

Propoziția 3.4 Fie $\mathbf{x} \in \mathbb{Z}_2^n$ și $A_{n,k}$ un (n, k) - cod liniar binar. Atunci are loc egalitatea

$$\frac{1}{2^k} \sum_{\mathbf{v} \in A} (-1)^{\mathbf{v}\mathbf{x}} = \begin{cases} 1 & \text{dacă } \mathbf{x} \in A_{n,k}^\perp \\ 0 & \text{altfel} \end{cases}$$

Demonstrație: Dacă $\mathbf{x} \in A_{n,k}^\perp$ atunci evident, $(-1)^{\mathbf{v}\mathbf{x}} = (-1)^0 = 1$ și suma este egală cu numărul de cuvinte - cod din $A_{n,k}$, care este 2^k .

Să presupunem acum că $\mathbf{x} \notin A_{n,k}^\perp$; deci există un cuvânt - cod $\mathbf{v}_0 \in A$ cu $\mathbf{v}_0\mathbf{x} = 1$.

Vom arăta că în acest caz, numărul cuvintelor - cod ortogonale pe \mathbf{x} este egal cu cel al cuvintelor - cod ne-ortogonale pe \mathbf{x} (și deci suma din formulă este 0).

Fie $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r$ toate cuvintele - cod ortogonale pe \mathbf{x} . Facem afirmația că atunci $\mathbf{v}_1 + \mathbf{v}_0, \mathbf{v}_2 + \mathbf{v}_0, \dots, \mathbf{v}_r + \mathbf{v}_0$ sunt toate cuvintele - cod ne-ortogonale pe \mathbf{x} .

Într-adevăr:

1. $\forall i (1 \leq i \leq r) \quad (\mathbf{v}_i + \mathbf{v}_0)\mathbf{x} = \mathbf{v}_i\mathbf{x} + \mathbf{v}_0\mathbf{x} = 0 + 1 = 1$;
2. Dacă $\mathbf{v} \in A_{n,k}$ verifică relația $\mathbf{v}\mathbf{x} = 1$, atunci $\mathbf{v} - \mathbf{v}_0$ este un cuvânt cod - ortogonal pe \mathbf{x} , deci $\mathbf{v} - \mathbf{v}_0 = \mathbf{v}_i$ pentru un anumit i . q.e.d.

Teorema 3.8 (*Identitatea MacWilliams*) Pentru orice (n, k) - cod liniar binar C are loc relația

$$P_{C^\perp}(x) = \frac{(1+x)^n}{2^k} P_C\left(\frac{1-x}{1+x}\right).$$

Demonstrație: Să rescriem numărătorul de ponderi sub o formă puțin diferită:

$$B(x, y) = \sum_{i=0}^n A_i x^i y^{n-i}.$$

Evident, deoarece $P(x) = B(x, 1)$ și $B(x, y) = y^n P\left(\frac{x}{y}\right)$, cele două expresii sunt echivalente.

În notația cu polinomul B , identitatea MacWilliams se scrie

$$B_{C^\perp}(x) = \frac{1}{2^k} B_C(y-x, y+x).$$

Cu ajutorul ponderii cuvintelor - cod, numărătorul de ponderi are forma

$$B_C(x, y) = \sum_{i=0}^n A_i x^i y^{n-i} = \sum_{\mathbf{a} \in C} x^{w(\mathbf{a})} y^{n-w(\mathbf{a})}$$

unde $\mathbf{a} = (a_1, a_2, \dots, a_n)$.

Prelucrând membrul drept al identității MacWilliams, avem:

$$B_C(y-x, y+x) = \sum_{\mathbf{a} \in C} (y-x)^{w(\mathbf{a})} (y+x)^{n-w(\mathbf{a})} = \sum_{\mathbf{a} \in C} \prod_{i=1}^n [y + (-1)^{a_i} x].$$

În mod analog, membrul stâng se scrie:

$$B_{C^\perp}(x, y) = \sum_{\mathbf{a} \in C^\perp} x^{w(\mathbf{a})} y^{n-w(\mathbf{a})}.$$

Folosind Propoziția 3.4, el se poate reformula:

$$B_{C^\perp}(x, y) = \sum_{\mathbf{a} \in Z_2^n} \left[\frac{1}{2^k} \sum_{\mathbf{v} \in C} (-1)^{\mathbf{a}\mathbf{v}} \right] x^{w(\mathbf{a})} y^{n-w(\mathbf{a})}.$$

Expresia din paranteze este 0 pentru toate cuvintele \mathbf{a} care nu sunt în C^\perp . Deci

$$B_{C^\perp}(x, y) = \frac{1}{2^k} \sum_{\mathbf{v} \in C} \sum_{\mathbf{a} \in Z_2^n} (-1)^{\mathbf{v}\mathbf{a}} x^{w(\mathbf{a})} y^{n-w(\mathbf{a})}.$$

Suma interioară se face după toate secvențele binare \mathbf{a} de lungime n . Vom ordona această sumă după ponderile lui \mathbf{a} : pentru $\mathbf{a} = 0$ sumandul este y^n ; pentru cuvintele \mathbf{a} de pondere 1 avem: $[(-1)^{a_1} + (-1)^{a_2} + \dots + (-1)^{a_n}]xy^{n-1}$ etc.

În final, pentru ponderea n avem $[(-1)^{a_1} + \dots + (-1)^{a_n}]x^n$.

Suma tuturor acestor sumanzi $\sum_{k=0}^n [(-1)^{a_{i_1}} + \dots + (-1)^{a_{i_k}}]x^k y^{n-k}$ este egală cu

$$[y + (-1)^{a_1}x][y + (-1)^{a_2}x] \dots [y + (-1)^{a_n}x] = \prod_{i=1}^n [y + (-1)^{a_i}x].$$

$$\text{Deci } B_{C^\perp}(x, y) = \frac{1}{2^k} \sum_{\mathbf{a} \in C} \prod_{i=1}^n [y + (-1)^{a_i}x] = \frac{1}{2^k} B_C(y - x, y + x). \quad \text{q.e.d.}$$

Exemplul 3.8 Să considerăm codul $A_{4,2}$ din Exemplul 19.5 al cărui numărător de ponderi a fost dat în Exemplul 3.7. Pentru codul dual, numărătorul de ponderi este

$$\begin{aligned} P_{A_{4,2}^\perp}(x) &= \frac{(1+x)^4}{2^2} P_{A_{4,2}}\left(\frac{1-x}{1+x}\right) = \frac{(1+x)^4}{4} \left[1 + \left(\frac{1-x}{1+x}\right)^2 + 2\left(\frac{1-x}{1+x}\right)^3 \right] = \\ &= \frac{(1+x)^4 + (1+x)^2(1-x)^2 + 2(1+x)(1-x)^3}{4} = \frac{4 + 4x^2 + 8x^3}{4} = 1 + x^2 + 2x^3. \end{aligned}$$

Deci cele două coduri au același numărător de ponderi.

Aceasta nu înseamnă însă că cele două coduri coincid (și deci codul ar fi auto-dual); a avea același numărător de ponderi este doar o condiție necesară, nu și suficientă pentru ca un cod să coincidă cu dualul său.

Ca o verificare, $A_{4,2} = \{0000, 0111, 1010, 1101\}$, iar $A_{4,2}^\perp = \{0000, 0101, 1011, 1110\}$.

Exemplul 3.9 Fie codul liniar cu repetiție de lungime pară n $C = \{00\dots 0, 11\dots 1\}$; numărătorul lui de ponderi este $P_C(x) = 1 + x^n$. Codul dual are numărătorul de ponderi

$$P_{C^\perp}(x) = \frac{(1+x)^n}{2} \left[1 + \left(\frac{1-x}{1+x}\right)^n \right] = 1 + C_n^2 x^2 + C_n^4 x^4 + \dots + x^n$$

Rezultă din această formă că dualul codului liniar cu repetiție este codul liniar binar al tuturor cuvintelor de lungime n și pondere pară.

3.6 Exerciții

3.1 În Z_2^n notăm cu $\bar{\mathbf{x}}$ cuvântul obținut din \mathbf{x} prin permutarea caracterelor 0 și 1 între ele. Să se arate că pentru orice $\mathbf{a}, \mathbf{b} \in Z_2^n$:

1. $\bar{\mathbf{a}} + \bar{\mathbf{b}} = \mathbf{a} + \mathbf{b}$;
2. $\mathbf{a} + \bar{\mathbf{b}} = \bar{\mathbf{a}} + \mathbf{b} = \overline{\mathbf{a} + \mathbf{b}}$;
3. $d(\mathbf{a}, \bar{\mathbf{b}}) = d(\bar{\mathbf{a}}, \mathbf{b}) = w(\overline{\mathbf{a} + \mathbf{b}})$.

3.2 Descrieți codurile modificate obținute din codul liniar binar cu matricea genera-toare

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

3.3 Aceeași problemă pentru codul peste Z_3 definit prin

$$G = \begin{pmatrix} 1 & 0 & 0 & 2 & 2 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

3.4 Fie $A_{n,k}$ un (n, k) - cod liniar binar și $A'_{n,k-1}$ - codul obținut din A prin expurgare. Ce relație există între matricile de control ale celor două coduri ?

3.5 Arătați cum poate codul binar cu repetiție de lungime 7 să corecteze două erori și să detecteze 4 erori simultan. Câte erori poate detecta dacă corectează o eroare ?

3.6 Fie un $(15, 4)$ - cod liniar binar în care fiecare coloană i din matricea generatoare este scrierea binară a lui i sub forma unui vector cu 4 componente. Să se determine distanța minimă, numărătorul de ponderi și numărătorul de ponderi al codului dual.

3.7 Fie C un $(2k + 1, k)$ - cod binar astfel ca $C^\perp \subset C$. Descrieți $C^\perp \setminus C$.

Capitolul 4

Clase de coduri liniare

4.1 Coduri Hamming

Fie H matricea de control a unui cod liniar binar. Dacă se transmite un cuvânt - cod \mathbf{a} și se recepționează $\mathbf{a} + \mathbf{e}$ (deci cu eroarea - tip \mathbf{e}), atunci sindromul este $\mathbf{e}H^T$. Acest sindrom este egal cu suma coloanelor lui H care corespund pozițiilor afectate de erori (Prelegerea II, Teorema 2.3).

În particular, o eroare care apare pe o poziție corespunzătoare unei coloane nule din H nu influențează sindromul. Deci, o astfel de eroare nu este detectată.

Dacă H are două coloane identice și se întâmplă ca pe pozițiile corespunzătoare lor să apară simultan erori, acestea se anulează reciproc în calculul sindromului - și deci nu pot fi detectate.

Pe de-altă parte, dacă toate coloanele lui H sunt distincte și nenule, o eroare singulară pe poziția s va face ca sindromul să fie egal cu coloana numărul s din H . În acest caz, erorile singulare pot fi detectate și corectate foarte ușor.

Pe baza acestor observații am demonstrat teorema:

Teorema 4.1 *Un cod liniar binar poate corecta o eroare dacă și numai dacă matricea sa de control are toate coloanele nenule și distincte.*

Pentru a se putea corecta toate erorile simple, trebuie să existe sindromuri distincte pentru fiecare eroare - tip; deci, conform marginii Hamming (Prelegerea III, Teorema 3.5),

$$2^{n-k} \geq n + 1.$$

Pe baza acestor considerații se definește codul Hamming binar:

Definiția 4.1 *Codul liniar binar în care coloanele matricii H sunt reprezentarea binară a numerelor $1, 2, \dots, 2^r - 1$ este numit cod Hamming binar.*

Deci, pentru orice număr natural r ($r \geq 2$) se poate construi un (n, k) - cod liniar binar în care $n = 2^r - 1$, $k = 2^r - r - 1$.

De remarcat că definiția nu determină pentru fiecare r în mod unic matricea de control a codului Hamming. De obicei se consideră acea matrice H în care coloana i reprezintă scrierea în binar a numărului i . Deoarece codul este echivalent cu un cod sistematic (există coloane pentru $2^0, 2^1, \dots, 2^{r-1}$), toate celelalte reprezentări au aceleași proprietăți.

Exemplul 4.1 Pentru $r = 3$ avem codul Hamming de lungime $n = 2^3 - 1 = 7$ cu $k = 2^3 - 3 - 1 = 4$ simboluri de informație și 3 simboluri de control. Matricea de control este:

$$H_{3,7} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

De aici rezultă că el este determinat de soluțiile sistemului liniar:

$$\begin{cases} x_4 + x_5 + x_6 + x_7 = 0 \\ x_2 + x_3 + x_6 + x_7 = 0 \\ x_1 + x_3 + x_5 + x_7 = 0 \end{cases}$$

Să determinăm matricea generatoare a acestui cod. Pentru aceasta, construim întâi codul sistematic echivalent, permutând coloanele pentru a aduce matricea de control la forma eșalonată canonic:

$$H_{3,7}^* = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

De aici se obține matricea generatoare eșalonată canonic:

$$G_{4,7}^* = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Aplicând permutarea inversă asupra coloanelor, se obține matricea generatoare a $(7, 4)$ - codului Hamming binar:

$$G_{4,7} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Coloanele corespunzătoare matricii unitate $(3, 5, 6, 7)$, reprezintă pozițiile simbolurilor de informație în fiecare cuvânt - cod.

Deci simbolurile x_1, x_2, x_4 sunt simboluri de control. Sistemul de sus poate fi rearanjat pentru a permite calculul simbolurilor de control plecând de la simbolurile de informație:

$$\begin{cases} x_1 = x_3 + x_5 + x_7 \\ x_2 = x_3 + x_6 + x_7 \\ x_4 = x_5 + x_6 + x_7 \end{cases}$$

Toate cuvintele codului sunt:

Informație	Cuvânt cod	Informație	Cuvânt cod
0000	0000000	0110	0110011
1000	1000011	0101	0101010
0100	0100101	0011	0011001
0010	0010110	1110	1110000
0001	0001111	1101	1101001
1100	1100110	1011	1011010
1010	1010101	0111	0111100
1001	1001100	1111	1111111

Teorema 4.2 *Orice cod Hamming binar are distanța minimă 3.*

Demonstrație: Evident, orice două coloane din matricea de control sunt liniar independente. În plus, se pot găsi trei coloane (de exemplu primele trei) a căror sumă să fie $\mathbf{0}$. Conform Teoremei 2.4 (Prelegerea II), distanța minimă a codului este 3. q.e.d.

Deci orice cod Hamming binar poate corecta o eroare sau poate detecta două erori. El nu poate realiza acest lucru simultan (nu verifică condiția $d \geq s + t + 1$ din Teorema 3.7, Prelegerea III).

Decodificarea se realizează foarte simplu, conform următorului algoritm:

Algoritm A:

Fie \mathbf{a} vectorul recepționat.

1. Se calculează sindromul $\mathbf{s} = \mathbf{a}H^T$.
2. Dacă $\mathbf{s} = \mathbf{0}$, nu a apărut nici o eroare (sau eroarea este nedetectabilă), deci $\mathbf{v} = \mathbf{a}$, STOP.
3. Altfel, eroarea este pe poziția i , unde i este numărul a cărei reprezentare în binar este sindromul \mathbf{s} .

Decodificarea este $\mathbf{v} = \mathbf{a} + \mathbf{e}_i$, unde \mathbf{e}_i este vectorul care are 1 pe poziția i și 0 în rest.

Exemplul 4.2 *Să considerăm din nou (7,4) - codul Hamming binar din Exemplul 19.4 și să presupunem că s-a recepționat cuvântul $\mathbf{x} = 0011101$. Calculul sindromului conduce la valoarea*

$$\mathbf{x}H^T = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

care este scrierea în binar a numărului 5. Deci a intervenit o eroare simplă pe poziția a cincea. Corectăm această poziție – schimbând 1 cu 0 și se obține cuvântul - cod 0011001, care pe pozițiile 3, 5, 6, 7 conține mesajul de informație: 1001.

Codul Hamming binar poate fi îmbunătățit prin extensie. Această operație conduce la un $(2^r, 2^r - r - 1)$ - cod liniar, cu toate cuvintele - cod de pondere pară.

Exemplul 4.3 *Prin extensia (7,4) - codului Hamming se obține codul cu matricea de control*

$$H^* = \left(\begin{array}{ccccccc|c} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array} \right).$$

De remarcat că H^ are rangul 4; în plus, toate liniile acestei matrici sunt cuvinte - cod în codul Hamming binar extins. Deci H^* poate fi considerată matrice generatoare a acestui cod. Rezultă că (8,4) - codul Hamming binar extins este auto - dual.*

Un cod Hamming binar extins este soluția unui sistem linear de $n - k + 1$ ecuații, ecuația suplimentară $\sum_{i=1}^{n+1} x_i = 0$ fiind *ecuația de control a parității*.

Propoziția 4.1 *Un cod Hamming binar extins are $d = 4$.*

Demonstrație: Fie $\mathbf{a} = a_1 a_2 \dots a_n$ un cuvânt - cod de pondere $d = 3$ din codul Hamming binar. Trecând la codul extins, cuvântul $\mathbf{a}' = a_1 a_2 \dots a_n a_{n+1}$ verifică relația suplimentară $\sum_{i=1}^{n+1} a_i = 0$ (reamintim, sumele se fac modulo 2).

Cum $\sum_{i=1}^n a_i = 1$ ($w(\mathbf{a}) = 3$), rezultă $a_{n+1} = 1$.

Noul cuvânt are evident pondere minimă, și aceasta este $3 + 1 = 4$. q.e.d.

Codurile Hamming binare extinse corectează o eroare simplă și detectează 2 erori simultan. Algoritmul prezentat este bazat pe verificarea celor $n - k + 1$ ecuații de control:

Algoritm B:

1. Dacă nu sunt verificate ecuația de control a parității și cel puțin una din primele $n - k$ ecuații, înseamnă că a apărut o eroare simplă, care se corectează cu Algoritmul A;
2. Dacă ecuația de control a parității este verificată dar cel puțin una din primele $n - k$ ecuații de control nu se verifică, s-a detectat o eroare dublă;
3. În celelalte situații nu au apărut erori (sau eroarea este nedetectabilă).

Faptul că se poate lua totdeauna o decizie, se bazează pe următorul rezultat:

Teorema 4.3 *Codul Hamming binar este perfect.*

Demonstrație: Reamintim (Prelegerea II, Definiția 2.8) că un cod binar C este perfect dacă

$$Z_q^n = \bigcup_{\mathbf{x} \in C} S_t(\mathbf{x}), \quad \text{unde } t = \left\lceil \frac{d-1}{2} \right\rceil.$$

Scriind această relație în funcție de numărul de elemente din fiecare sferă și ținând cont că toate sferile conțin un număr egal de elemente, avem

$$q^k [1 + C_n^1(q-1) + \dots + C_n^t(q-1)^t] = q^n.$$

Pentru cazul codurilor Hamming, $q = 2$, $n = 2^r - 1$, $k = 2^r - r - 1$, $t = 1$, deci totul revine la verificarea egalității $2^k(1+n) = 2^n$. q.e.d.

De remarcat că rata de informație a codurilor Hamming

$$R = \frac{k}{n} = 1 - \frac{r}{2^r - 1}$$

crește rapid spre 1.

Evident însă că odată cu această creștere scade protecția față de erori.

4.1.1 Coduri Hamming nebinare

Definiția 4.2 Fie q un număr prim, r ($r \geq 2$) un număr întreg și $n = \frac{q^r - 1}{q - 1}$. Se numește cod Hamming nebinar un $(n, n - r)$ - cod liniar peste Z_q , în care matricea de control are orice pereche de două coloane liniar independente (nici o coloană nu este multiplu scalar al altei coloane).

Mulțimea coloanelor unui astfel de cod formează o mulțime maximală de vectori liniar independenți doi câte doi.

Exemplul 4.4 Să considerăm $q = 3, r = 2$. Atunci $n = \frac{3^2 - 1}{3 - 1} = 4$. Un $(4, 2)$ - cod Hamming ternar poate fi definit prin de matricea de control

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

Decodificarea se poate face folosind tabela de sindromuri, în care s-au luat ca reprezentanți toate combinațiile posibile de o eroare:

Sindrom	Reprezentant	Sindrom	Reprezentant
01	1000	12	0001
02	2000	20	0200
10	0100	21	0002
11	0010	22	0020

Dacă se recepționează de exemplu 2222, calculul sindromului dă $\mathbf{s} = 02$. Reprezentantul este 2000. Se calculează $2222 - 2000 = 2222 + 1000 = 0222$ deci cuvântul - cod transmis a fost 2021.

Cum matricea generatoare a acestui cod este $G = \begin{pmatrix} 2 & 2 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{pmatrix}$ ultimele două caractere formează mesajul de informație; deci s-a codificat mesajul 22.

Despre codurile Hamming nebinare se pot stabili următoarele rezultate:

- Deoarece pentru q și r fixați, codurile Hamming corespunzătoare sunt echivalente, se poate alege o anumită matrice de control. Uzual se folosește matricea în care se scriu pe coloane toate elementele din Z_q^r , care satisfac cerința ca primul element nenul (de sus în jos) să fie 1.
- Codurile Hamming nebinare au $d = 3$ (evident, din construcția matricii de control de mai sus). Deci ele pot corecta o eroare.
- Proprietatea de a fi coduri perfecte se păstrează. Într-adevăr, deoarece un cod Hamming conține $q^k = q^{n-r}$ cuvinte - cod iar $n = \frac{q^r - 1}{q - 1}$, egalitatea stabilită în demonstrația Teoremei 4.3 se scrie $q^{n-r}[1 + n(q - 1)] = q^n$, care se verifică imediat.

Exemplul 4.5 Să considerăm $(13, 10)$ - codul Hamming ternar. Acest cod are o aplicație interesantă în problema Pronosportului. După cum se știe, un buletin Pronosport conține rezultatele (notate cu 1, 2, X) a 13 meciuri. Pentru a avea sigur 13 rezultate exacte trebuie completate 3^{13} buletine.

Câte buletine sunt însă necesare pentru a avea sigur 12 rezultate exacte ?

La prima vedere s-ar părea că 3^{12} . Completând însă buletinele cu elementele codului Hamming ternar $(13, 10)$ (cu 0 în loc de X) – care sunt în număr de 3^{10} , se atinge scopul dorit. Într-adevăr, acesta fiind un cod perfect corector de o eroare, orice element din Z_3^{13} diferă prin cel mult o poziție de un cuvânt - cod.

Astfel, numărul buletinelor se reduce de nouă ori.

4.2 Codul Golay

Al doilea cod liniar prezentat are capacitatea de corecție pentru maxim 3 erori.

Vom construi întâi varianta extinsă a codului, deoarece algoritmul de decodificare este mai simplu și ușor de aplicat ulterior la codul Golay normal.

4.2.1 Codul Golay binar extins

Acest cod a fost folosit de programul spațial *Voyager* la începutul anilor '80 pentru transmiterea fotografiilor planetelor Jupiter și Saturn.

Să considerăm matricea 12×12 din Figura 4.1:

Tabelul 4.1:

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

Fie matricea 12×24 $G = (I_{12}|B)$. Codul liniar binar generat de G se numește *codul Golay extins* și va fi notat C_{24} .

Observația 4.1

- Matricea B este mai ușor de construit decât pare la prima vedere. Astfel, eliminând ultima linie și coloană, matricea rămasă – să spunem B_1 – este generată ciclic (spre stânga) de cuvântul binar 11011100010. Deci $B = \begin{pmatrix} B_1 & \mathbf{1}^T \\ \mathbf{1} & 0 \end{pmatrix}$, unde $\mathbf{1} = 1111111111$. Evident, B este simetrică ($B^T = B$).
- C_{24} are $n = 24$, $k = 12$ și $2^{12} = 4096$ cuvinte - cod.
- Conform Teoremei 2.2, o matrice de control a codului este $H = (B|I_{12})$.

Teorema 4.4 $H = (I_{12}|B)$ este de asemenea matrice de control pentru C_{24} .

Demonstrație: Liniile din B au pondere impară (7 sau 11); deci produsul (scalar) al unei linii cu ea însăși este 1. O verificare simplă arată că produsul primei linii cu oricare altă linie din B este 0. Structura ciclică a lui B_1 asigură că atunci produsul scalar al oricăror două linii este 0.

În concluzie, $BB^T = I_{12}$. Dar $B^T = B$, așa că putem scrie:

$$GH^T = (I|B) \begin{pmatrix} I \\ B \end{pmatrix} = I^2 + B^2 = I + BB^T = I + I = \mathbf{0}.$$

Vom folosi ambele matrici de control pentru decodificarea codului C_{24} . q.e.d.

Corolarul 4.1

- A.** C_{24} admite ca matrice generatoare și pe $G = (B|I_{12})$.
B. Codul Golay extins este auto - dual ($C_{24} = C_{24}^\perp$).

Demonstrație: Se verifică imediat. q.e.d.

Teorema 4.5 C_{24} are distanța minimă $d = 8$.

Demonstrație: Vom demonstra afirmația în trei pași.

1. Ponderea cuvintelor din C_{24} este multiplu de 4.

Să observăm că liniile lui G au pondere 8 sau 12. Fie $\mathbf{v} \in C_{24}$, scris ca sumă de două linii din G : $\mathbf{v} = \mathbf{r}_i + \mathbf{r}_j$.

Cum B are liniile ortogonale, rezultă că și liniile lui G sunt ortogonale. Deci \mathbf{r}_i și \mathbf{r}_j au un număr par (să zicem $2x$) de elemente 1 în comun. Atunci $w(\mathbf{v}) = w(\mathbf{r}_i) + w(\mathbf{r}_j) - 2(2x)$, care este multiplu de 4.

Fie acum $\mathbf{v} \in C_{24}$ reprezentat ca sumă de trei linii din G : $\mathbf{v} = \mathbf{r}_i + \mathbf{r}_j + \mathbf{r}_s$. Notăm $\mathbf{v}_1 = \mathbf{r}_i + \mathbf{r}_j$. Deoarece $\mathbf{v}_1, \mathbf{r}_s \in C_{24}$, iar C_{24} este auto - dual (deci oricare două cuvinte - cod sunt ortogonale), rezultă că \mathbf{v}_1 și \mathbf{r}_s au în comun un număr par (să zicem $2y$) de elemente 1.

Deci $w(\mathbf{v}) = w(\mathbf{v}_1) + w(\mathbf{r}_s) - 2(2y)$ care este multiplu de 4.

Folosind acum un procedeu de inducție, cum orice cuvânt - cod este combinație liniară de linii din G , ponderea sa va fi multiplu de 4.

2. Primele 11 linii din G sunt cuvinte - cod de pondere 8, deci distanța minimă a codului C_{24} este 4 sau 8.

3. C_{24} nu are cuvinte de pondere 4.

Să presupunem că există $\mathbf{v} \in C_{24}$ cu $w(\mathbf{v}) = 4$. Cum pentru C_{24} am considerat două matrici generatoare, vor exista două mesaje de informație $u_1, u_2 \in Z_2^{12}$ cu $\mathbf{v} = \mathbf{u}_1(I|B)$, $\mathbf{v} = \mathbf{u}_2(B|I)$. Una din cele două jumătăți din \mathbf{v} are cel puțin doi de 1; deci $w(\mathbf{u}_1) \leq 2$ sau $w(\mathbf{u}_2) \leq 2$.

Pe de altă parte, o linie din B are pondere minim 7, deci \mathbf{v} nu este o linie a matricii B . Dacă \mathbf{v} este suma a două linii din B , ea nu poate avea o pondere mai mică de 4; deci $w(\mathbf{v}) = w(\mathbf{u}_1) + w(\mathbf{u}_2) \geq 2 + 2 > 4$, contradicție. q.e.d.

4.2.2 Decodificarea codului Golay extins

Conform Teoremei 4.5, un cod Golay extins poate corecta orice combinație de maxim 3 erori independente.

În această secțiune vom nota cu $\mathbf{a} \in Z_{24}$ cuvântul recepționat, cu \mathbf{v} cuvântul - cod cel mai apropiat, și cu \mathbf{e} eroarea - tip ($\mathbf{v} = \mathbf{a} + \mathbf{e}$).

Deoarece capacitatea de corecție este de 3 erori, vom considera $w(\mathbf{e}) \leq 3$.

Pentru orice cuvânt $\mathbf{a} \in Z_{24}^2$, vom separa cu o virgulă prima jumătate a cuvântului de cea de-a doua: $\mathbf{a} = [\mathbf{a}_1, \mathbf{a}_2]$ cu $\mathbf{a}_1, \mathbf{a}_2 \in Z_{12}^2$. Similar, eroarea - tip va fi notată $\mathbf{e} = [\mathbf{e}_1, \mathbf{e}_2]$. Evident, condiția $w(\mathbf{e}) \leq 3$ implică $w(\mathbf{e}_1) \leq 1$ sau $w(\mathbf{e}_2) \leq 1$.

Folosind cele două matrici de control posibile, se pot defini două sindromuri pentru \mathbf{a} :

$$\begin{aligned} \mathbf{s}_1 &= \mathbf{e} \begin{pmatrix} I_{12} \\ B \end{pmatrix} = [\mathbf{e}_1, \mathbf{e}_2] \begin{pmatrix} I_{12} \\ B \end{pmatrix} = \mathbf{e}_1 + \mathbf{e}_2 B, \\ \mathbf{s}_2 &= \mathbf{e} \begin{pmatrix} B \\ I_{12} \end{pmatrix} = [\mathbf{e}_1, \mathbf{e}_2] \begin{pmatrix} B \\ I_{12} \end{pmatrix} = \mathbf{e}_1 B + \mathbf{e}_2. \end{aligned}$$

De aici rezultă următoarea observație: dacă $w(\mathbf{e}_2) \leq 1$, atunci \mathbf{s}_1 este sau un cuvânt de pondere maxim 3 (dacă $w(\mathbf{e}_2) = 0$), sau o linie a lui B cu cel mult doi biți schimbați (dacă $w(\mathbf{e}_2) = 1$).

Similar, dacă $w(\mathbf{e}_1) \leq 1$, atunci \mathbf{s}_2 este sau un cuvânt de pondere maxim 3 sau o linie a lui B cu cel mult doi biți schimbați.

Dacă se folosește și faptul că $\mathbf{s}_2 = \mathbf{e}_1 B + \mathbf{e}_2 = (\mathbf{e}_1 + \mathbf{e}_2 B)B = \mathbf{s}_1 B$ (deci se poate folosi doar prima matrice de control), putem defini următorul algoritm de decodificare a codurilor Golay extinse¹:

¹S-a notat cu \mathbf{u}_i un cuvânt de lungime 12 cu 1 pe poziția i și 0 în rest.

Algoritm C:

1. Se calculează sindromul $\mathbf{s} = \mathbf{a}H^T$, unde $H = (I_{12} \mid B)$;
2. Dacă $w(\mathbf{s}) \leq 3$, atunci $\mathbf{e} = [\mathbf{s}, \mathbf{0}]$, STOP.
3. Dacă există o linie \mathbf{b}_i a lui B cu $w(\mathbf{s} + \mathbf{b}_i) \leq 2$, atunci $\mathbf{e} = [\mathbf{s} + \mathbf{b}_i, \mathbf{u}_i]$, STOP.
4. Dacă $w(\mathbf{s}B) \leq 3$, atunci $\mathbf{e} = [\mathbf{0}, \mathbf{s}B]$, STOP.
5. Dacă există o linie \mathbf{b}_i a lui B cu $w(\mathbf{s}B + \mathbf{b}_i) \leq 2$, atunci $\mathbf{e} = [\mathbf{u}_i, \mathbf{s}B + \mathbf{b}_i]$, STOP.
6. Dacă \mathbf{e} nu a fost determinat încă, se cere retransmiterea.

După determinarea erorii \mathbf{e} , cuvântul - cod transmis se determină prin $\mathbf{v} = \mathbf{a} + \mathbf{e}$.

Exemplul 4.6 Să decodificăm cuvântul $\mathbf{a} = [101111101111, 010010010010]$.

Sindromul este

$$\mathbf{s} = \mathbf{a}H^T = 101111101111 + 001111101110 = 100000000001.$$

Deoarece $w(\mathbf{s}) = 2 \leq 3$, se găsește $\mathbf{e} = [\mathbf{s}, \mathbf{0}] = [100000000001, 000000000000]$, deci s-a transmis cuvântul $\mathbf{v} = \mathbf{a} + \mathbf{u} = [001111101110, 010010010010]$.

Deoarece $G = (I_{12} \mid B)$ este în forma eşalonat canonică, mesajul de informație (orice cuvânt din Z_2^{12}) apare pe primele 12 poziții ale cuvântului - cod. Astfel, în exemplul de sus, mesajul de informație a fost 001111101110.

Exemplul 4.7 Se cere decodificarea cuvântului $\mathbf{a} = [001001001101, 101000101000]$.

Sindromul este

$$\mathbf{s} = \mathbf{a}H^T = 001001001101 + 111000000100 = 110001001001.$$

Deoarece $w(\mathbf{s}) = 5$, se trece la pasul 3 al Algoritmului C și se calculează:

$$\begin{aligned} \mathbf{s} + \mathbf{b}_1 &= 000110001100 \\ \mathbf{s} + \mathbf{b}_2 &= 011111000010 \\ \mathbf{s} + \mathbf{b}_3 &= 101101011110 \\ \mathbf{s} + \mathbf{b}_4 &= 001001100100 \\ \mathbf{s} + \mathbf{b}_5 &= 000000010010 \end{aligned}$$

Deoarece $w(\mathbf{s} + \mathbf{b}_5) \leq 2$, se determină

$$\mathbf{e} = [\mathbf{s} + \mathbf{b}_5, \mathbf{e}_5] = [000000010010, 000010000000]$$

și se decide că s-a transmis cuvântul - cod

$$\mathbf{v} = \mathbf{a} + \mathbf{e} = [001001011111, 101010101000].$$

Exemplul 4.8 Să decodificăm cuvântul $\mathbf{a} = [000111000111, 011011010000]$.

Sindromul este

$$\mathbf{s} = \mathbf{a}H^T = \mathbf{e}_1 + \mathbf{e}_2B = 000111000111 + 101010101101 = 101101101010$$

care are ponderea 7. Trecând la pasul 3 se găsește $w(\mathbf{s} + \mathbf{b}_i) \geq 3$ pentru toate liniile lui B ; deci se continuă cu pasul 4: al doilea sindrom este $\mathbf{s}B = 111001111101$ cu ponderea 8. Pasul 5 va da:

$$\begin{aligned}
\mathbf{s}B + \mathbf{b}_1 &= 001110111000 \\
\mathbf{s}B + \mathbf{b}_2 &= 010111110110 \\
\mathbf{s}B + \mathbf{b}_3 &= 100101101010 \\
\mathbf{s}B + \mathbf{b}_4 &= 000001010000
\end{aligned}$$

S-a ajuns la $w(\mathbf{s}B + \mathbf{b}_4) \leq 2$, deci se poate determina eroarea:

$$\mathbf{e} = [\mathbf{u}_4, \mathbf{s}B + \mathbf{b}_4] = [000100000000, 000001010000].$$

Cuvântul - cod transmis a fost

$$\mathbf{v} = \mathbf{a} + \mathbf{e} = [000011000111, 011010000000]$$

4.2.3 Codul Golay binar

Prin relaxarea codului Golay extins (eliminarea ultimului bit din fiecare cuvânt - cod) se ajunge la *Codul Golay binar*.

Fie \hat{B} matricea 12×11 obținută din B prin eliminarea ultimei coloane. Definim $\hat{G} = (I_{12} | \hat{B})$. Codul linear binar generat de \hat{G} se numește *cod Golay* și este notat cu C_{23} . Caracteristicile sale sunt:

$$n = 23, \quad k = 12, \quad \text{număr de cuvinte - cod: } 2^{12} = 4096.$$

Evident, extensia lui C_{23} este C_{24} .

Teorema 4.6 *Distanța minimă a codului Golay este $d = 7$.*

Demonstrație: Demonstrația se poate face fie direct (similar celei de la Teorema 4.5) fie folosind faptul că C_{23} este relaxarea codului C_{24} , care are distanța minimă 8. q.e.d.

În consecință, un cod Golay va corecta orice combinație de maxim 3 erori.

Teorema 4.7 *Codul Golay este perfect.*

Demonstrație: Se verifică relația:

$$2^{12}(C_{23}^0 + C_{23}^1 + C_{23}^2 + C_{23}^3) = 2^{12}(1 + 23 + 253 + 1771) = 2^{12}2^{11} = 2^{23}. \text{ q.e.d.}$$

Rezultă că orice cuvânt $\mathbf{a} \in Z_2^{23}$ se află la o distanță cel mult 3 de un cuvânt - cod. Astfel, dacă se adaugă la sfârșit 0 sau 1, formând $\mathbf{a}0$ respectiv $\mathbf{a}1$ pentru a obține un cuvânt de pondere impară, acest cuvânt este la distanță maxim 3 de un cuvânt - cod $\mathbf{c} \in C_{24}$. Se folosește Algoritmul C pentru a obține acest cuvânt - cod, apoi se elimină ultimul caracter din \mathbf{c} ; se ajunge astfel la cel mai apropiat cuvânt - cod din C_{23} față de \mathbf{a} .

Algoritm D:

1. Se formează cuvântul extins de pondere impară $\mathbf{a}0$ sau $\mathbf{a}1$;
2. Se decodifică $\mathbf{a}i$ folosind Algoritmul C și se obține $\mathbf{c} \in C_{24}$;
3. Se elimină ultimul caracter din \mathbf{c} .

Exemplul 4.9 *Să decodificăm $\mathbf{a} = [001001001001, 11111110000]$.*

Deoarece \mathbf{a} are pondere impară, se construiește

$$\mathbf{a}0 = 001001001001, 111111100000.$$

Sindromul acestui cuvânt este $\mathbf{s}_1 = 100010111110$.

Pentru că $\mathbf{s}_1 = \mathbf{b}_6 + \mathbf{e}_9 + \mathbf{e}_{12}$, $\mathbf{a}0$ se decodifică în $[001001000000, 111110100000]$, așa că \mathbf{a} este decodificat în $[001001000000, 11111010000]$.

4.3 Unicitatea codurilor perfecte binare

Ambele tipuri de coduri liniare prezentate până acum sunt coduri perfecte.

Se observă imediat că pentru corectarea unei simple erori, singurele coduri binare perfecte sunt codurile Hamming.

Vom mai arăta că această singularitate este valabilă și în cazul codurilor Golay; anume, singurul cod binar perfect corector de 3 erori este codul Golay.

Pentru aceasta sunt necesare două leme:

Lema 4.1 *O condiție necesară pentru existența unui (n, k) - cod binar perfect corector de t erori este $\sum_{i=0}^t C_n^i = 2^p$ pentru un anumit p .*

Demonstrație: Rezultă imediat din relația scrisă în demonstrația Teoremei 4.3, în care se ia $q = 2$. q.e.d.

Lema 4.2 $\sum_{i=0}^t C_n^i = \frac{n+1}{t!} R_t(n)$

unde t este număr natural impar, $R_t(X) \in Z[X]$, $gr(R_t(X)) = t - 1$.

Demonstrație: Pentru $t = 1$ se verifică imediat.

Presupunem adevărată afirmația pentru t și o demonstrăm pentru $t + 2$. Avem

$$\sum_{i=0}^{t+2} C_n^i = \frac{n+1}{t!} + C_n^{t+1} + C_n^{t+2} = \frac{n+1}{(t+2)!} \left[(t+1)(t+2)R_t(n) + \prod_{i=0}^t (n-i) \right]$$

Expresia din paranteza dreaptă este un polinom de gradul $t + 1$; notându-l cu $R_{t+2}(n)$, afirmația este demonstrată. q.e.d.

Să considerăm acum cazul $t = 3$. Pentru ca să existe un cod binar perfect corector de 3 erori, cu lemele de sus, trebuie ca $(n+1)(n^2 - n + 6) = 3 \cdot 2^s$, sau

$$(n+1)[(n+1)^2 - 3(n+1) + 8] = 3 \cdot 2^s.$$

Considerată ca o ecuație în $n+1$, singurele soluții întregi pozitive sunt de forma $n+1 = 2^r p$ unde $p = 1$ sau $p = 3$. Înlocuind, se ajunge la

$$2^{2r} p^3 - 2^r \cdot 3p^2 + 8p = 2^{s-r} \cdot 3 \quad (1)$$

Pentru $r \leq 3$ și $p = 1, 3$ verificările se fac imediat. Pentru $r \geq 4$ se ajunge la contradicție.

Singurele valori care verifică ecuația sunt:

- $n = 0, 1, 2$ – nu corespund nici unui cod.
- $n = 3$ – codul trivial cu un singur cuvânt - cod de lungime 3.
- $n = 7$ – codul (trivial) cu repetiție $\{0000000, 1111111\}$.
- $n = 23$ – codul Golay.

În acest mod am demonstrat teorema:

Teorema 4.8 *Codul binar Golay este singurul cod binar perfect netrivial corector de 3 erori.*

Lemele 15.1 și 4.2 pot fi folosite și pentru alte valori impare ale lui t . Cercetările nu au condus la alte coduri perfecte binare, dar nici nu s-a demonstrat că nu există nici un cod perfect binar corector de t ($t > 3$ impar) erori. Afirmatia este valabilă deocamdată pentru $t < 20$.

Un alt caz interesant de studiu este $q = 2$, $t = 2$. Aici se poate da teorema:

Teorema 4.9 *Nu există nici un cod netrivial binar perfect corector de 2 erori.*

Demonstrație: Lema 15.1 conduce la relația

$$(2n + 1)^2 = 2^{s+3} - 7.$$

Ecuatia $x^2 + 7 = 2^m$ a fost studiată în multe articole (vezi Math. Rev. 26, #74).

Singurele soluții sunt $x = 1, 3, 5, 11, 181$ cărora le corespund:

$n = 0, 1$ – fără coduri.

$n = 2$ – codul trivial cu un singur cuvânt.

$n = 5$ – codul cu repetiție $\{00000, 11111\}$.

$n = 90$.

Acest ultim caz este eliminat de următorul rezultat (Van Lindt - Coding theory, pp. 95):

Dacă există un cod binar perfect corector de t erori, atunci $\frac{n+1}{t+1}$ este număr întreg.
q.e.d.

4.4 Exerciții

4.1 Fie $(8, 4)$ - codul Hamming binar extins. Decodificați cuvintele

$$10101010 \quad 11010110 \quad 11111111.$$

4.2 Să se demonstreze că toate cuvintele - cod ale codului Hamming binar extins $(2^r, 2^r - r - 1)$ au pondere pară.

4.3 Construiți codurile Hamming ternare pentru $r = 2, 3$ și determinați decodificarea pe baza sindromurilor.

4.4 Construiți $(5, 3)$ - codul Hamming peste Z_4 . Determinați toate cuvintele - cod și tabela de decodificare cu sindromuri.

Decodificați cuvintele: 11223, 32101 2222 1100.

4.5 Demonstrați afirmațiile din Corolarul 15.1.

4.6 Arătați că C_{24} conține un cuvânt cu toate componentele egale cu 1 și nici un cuvânt de pondere 20.

Demonstrați că numărătorul de ponderi al lui C_{24} este:

$$1 + 759X^8 + 2576X^{12} + 759X^{16} + X^{24}.$$

4.7 De ce în Algoritmul D se face o extensie a cuvintelor la cuvinte de pondere impară? Dați un exemplu pentru a justifica raționamentul.

4.8 În codul Golay extins C_{24} să se decodifice – dacă este posibil – cuvintele:

$$\begin{array}{ll} [111000000000, 011011011011] & [111111000000, 100011100111] \\ [111111000000, 101011100111] & [111111000000, 111000111000] \\ [111000000000, 110111001101] & [110111001101, 111000000000] \\ [000111000111, 101000101101] & [110000000000, 101100100000] \end{array}$$

4.9 Să se determine cea mai probabilă eroare - tip pentru cuvinte având sindromurile:

$$\begin{array}{ll} \mathbf{s}_1 = 010010000000, & \mathbf{s}_2 = 011111010000 \\ \mathbf{s}_1 = 010010100101, & \mathbf{s}_2 = 001000110000 \\ \mathbf{s}_1 = 111111000101, & \mathbf{s}_2 = 111100010111 \\ \mathbf{s}_1 = 111111111011, & \mathbf{s}_2 = 010010001110 \\ \mathbf{s}_1 = 001101110110, & \mathbf{s}_2 = 111110101101 \\ \mathbf{s}_1 = 010111111001, & \mathbf{s}_2 = 100010111111 \end{array}$$

4.10 Folosind C_{23} , decodificați cuvintele:

$$\begin{array}{ll} [101011100000, 10101011011] & [101010000001, 11011100010] \\ [100101011000, 11100010000] & [011001001001, 01101101111] \end{array}$$

4.11 Detaliați demonstrația Teoremei 4.6.

4.12 Rezolvați ecuația (1).

Capitolul 5

Coduri Reed - Muller

Vom introduce o nouă clasă de coduri binare, caracterizate printr-o tehnică de decodificare deosebit de simplă: codurile *Reed - Muller* ($R - M$). Ele au fost definite de Reed, iar Muller a construit modalitatea de decodificare și – implicit – de detectare și corectare a erorilor. Unul din aceste coduri ($\mathcal{RM}(1, 5)$) a fost folosit în 1969 de sonda Mariner pentru transmiterea de imagini de pe Lună. Fiecare pixel din imagine avea asignat una din $2^6 = 64$ grade de umbră, iar cei șase biți de informație erau codificați într-un cuvânt de lungime 32. Codul $\mathcal{RM}(1, 5)$ poate corecta până la 7 erori.

5.1 Definirea prin funcții booleene

5.1.1 Funcții și polinoame booleene

Definiția 5.1 O funcție booleană de m ($m \geq 1$) variabile este o aplicație

$$f : Z_2^m \rightarrow Z_2.$$

O modalitate simplă folosită pentru definirea unei funcții booleene este asocierea unei *tabele de adevăr*: un tablou $(m + 1) \times 2^m$ care conține toate combinațiile posibile de m valori binare, cărora li se asociază valoarea funcției (de asemenea o valoare binară). Prin convenție, primii m biți de pe coloana i ($0 \leq i \leq 2^m - 1$) reprezintă scrierea în baza 2 a numărului i .

Exemplul 5.1 Următoarea tabelă de adevăr definește o funcție booleană de 3 variabile:

\mathbf{x}_0	0	1	0	1	0	1	0	1
\mathbf{x}_1	0	0	1	1	0	0	1	1
\mathbf{x}_2	0	0	0	0	1	1	1	1
\mathbf{f}	0	1	1	0	1	1	1	0

Observăm că o astfel de tabelă definește un cuvânt binar de lungime 8. Afirmatia este adevărată și invers: orice cuvânt binar de lungime 8 este definit printr-o tabelă de adevăr a unei funcții booleene de 3 variabile. Astfel, se pot identifica funcțiile booleene de 3 variabile prin cuvintele binare de lungime 8. În tabela de sus, cuvântul 01101110 este pus în corespondență biunivocă cu funcția \mathbf{f} .

În cele ce urmează, orice cuvânt binar $\mathbf{f} = f_0 f_1 \dots f_{2^m-1}$ de lungime 2^m este considerat ca o funcție booleană de m variabile, unde

$$\begin{aligned} \mathbf{f}(0, 0, \dots, 0, 0) &= f_0, \\ \mathbf{f}(0, 0, \dots, 0, 1) &= f_1, \\ \mathbf{f}(0, 0, \dots, 1, 0) &= f_2, \\ &\vdots \\ \mathbf{f}(1, 1, \dots, 1, 1) &= f_{2^m-1} \end{aligned}$$

În general, $f_i = \mathbf{f}(i_{m-1}, \dots, i_1, i_0)$, unde $i = \sum_{k=0}^{m-1} i_k 2^k$
 ($i_{m-1} \dots i_1 i_0$ este scrierea în binar a lui i).

Exemplul 5.2 Există două funcții booleene constante:

$$\mathbf{1} = 11 \dots 11, \quad \mathbf{0} = 00 \dots 00.$$

Exemplul 5.3 Orice variabilă poate fi tratată ca o funcție booleană. De exemplu, \mathbf{x}_0 este funcția booleană care asignează fiecărui m -tuplu $(x_0, x_1, \dots, x_{m-1})$ valoarea primei coordonate x_0 . Deci, valoarea este 0 pentru toate numerele pare și 1 pentru toate numerele impare: $\mathbf{x}_0 = 0101 \dots 01$ (vezi prima linie din Exemplul 20.3 pentru cazul $m = 3$).

În general,

\mathbf{x}_i este cuvântul binar în care pe poziția k ($0 \leq k \leq 2^m - 1$) este 1 atunci și numai atunci când scrierea binară a lui k conține 1 pe poziția i .

Această observație rezultă din modul de scriere al tabelor de adevăr.

De exemplu, $\mathbf{x}_1 = 00110011 \dots 0011$ și $\mathbf{x}_{m-1} = \underbrace{00 \dots 00}_{2^{m-1}} \underbrace{11 \dots 11}_{2^{m-1}}$.

Pentru $m = 4$, cele patru variabile sunt descrise în Tabelul 5.1:

Tabelul 5.1:

\mathbf{x}_0	0	1	0	1	0	1	0	1	0	1	0	1	0	1		
\mathbf{x}_1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
\mathbf{x}_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
\mathbf{x}_3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Pe mulțimea funcțiilor booleene se definesc două operații:

- **Suma logică (sau exclusiv):** $\mathbf{f} + \mathbf{g} = \mathbf{h}$
unde $h_i = f_i + g_i \pmod{2}$, $0 \leq i \leq 2^m - 1$.
- **Produsul logic (și):** $\mathbf{f} \mathbf{g} = \mathbf{h}$
unde $h_i = f_i g_i \pmod{2}$, $0 \leq i \leq 2^m - 1$.

Observația 5.1

1. Produsul logic verifică relația $\mathbf{ff} = \mathbf{f}$.

Deci în reprezentarea funcțiilor nu vor apare exponenți mai mari de 1.

2. Există și alte operații care pot fi exprimate cu ajutorul sumei și produsului logic. Astfel,

- Negația $\bar{\mathbf{f}} = \mathbf{1} + \mathbf{f}$;
- \vee (sau disjunctiv): $\mathbf{f} \vee \mathbf{g} = \mathbf{f} + \mathbf{g} + \mathbf{fg}$.

Definiția 5.2 Un polinom boolean de m nedeterminate este o sumă de termeni din mulțimea $\{\mathbf{0}, \mathbf{1}\} \cup \{\mathbf{x}_{i_1} \mathbf{x}_{i_2} \dots \mathbf{x}_{i_k} \mid 0 \leq i_1 < i_2 < \dots < i_k \leq m-1, k \geq 1\}$.

Funcția $\mathbf{0}$ este numită *polinom boolean de gradul -1* , funcția $\mathbf{1}$ este numită *polinom boolean de gradul 0*, iar orice alt polinom boolean are gradul k unde k este numărul maxim de factori dintr-un termen al lui \mathbf{f} .

Exemplul 5.4 Polinomul boolean $\mathbf{1} + \mathbf{x}_0 \mathbf{x}_1$ de 3 nedeterminate are gradul 2. El este negația polinomului

$$\mathbf{x}_0 \mathbf{x}_1 = (01010101)(00110011) = 00010001.$$

Deci $\mathbf{1} + \mathbf{x}_0 \mathbf{x}_1 = 11101110$.

Același polinom, considerat ca funcție de 4 nedeterminate, este cuvântul 1110111011101110.

Exemplul 5.5 Polinomul $\mathbf{x}_i \mathbf{x}_j$ ($i \neq j$) este cuvântul binar în care pe poziția k este 1 dacă și numai dacă reprezentarea binară a lui k are 1 pe pozițiile i și j . Numărul acestor situații este 2^{m-2} (deoarece pot fi alese arbitrar $m-2$ valori din reprezentarea binară a lui k pe m poziții). Deci $w(\mathbf{x}_i \mathbf{x}_j) = 2^{m-2}$.

Mai general, se poate arăta prin inducție

<p>Dacă i_1, i_2, \dots, i_r sunt valori distincte din $[0, m-1]$, atunci</p> $w(\mathbf{x}_{i_1} \mathbf{x}_{i_2} \dots \mathbf{x}_{i_r}) = 2^{m-r}. \quad (*)$
--

Fiecare polinom boolean de m variabile determină un cuvânt binar de lungime 2^m : pentru o singură nedeterminată, se folosește Exemplul 14.4, după care se operează adunările și multiplicările necesare.

Invers, orice cuvânt binar $\mathbf{f} = f_0 f_1 \dots f_{2^m-1}$ poate fi translatat într-un polinom boolean pe baza următoarei propoziții:

Propoziția 5.1 Dacă \mathbf{f} este o funcție booleană de m variabile, atunci:

$$f(x_0, \dots, x_{m-2}, x_{m-1}) = f(x_0, \dots, x_{m-2}, 0) + [f(x_0, \dots, x_{m-2}, 0) + f(x_0, \dots, x_{m-2}, 1)] \mathbf{x}_{m-1}.$$

Demonstrație: Deoarece x_{m-1} poate lua doar două valori (0, 1), este suficient să verificăm identitatea pentru acestea. Cazul $x_{m-1} = 0$ se verifică banal. Pentru $x_{m-1} = 1$ avem:

$$f(x_0, \dots, x_{m-2}, 0) + [f(x_0, \dots, x_{m-2}, 0) + f(x_0, \dots, x_{m-2}, 1)] = f(x_0, \dots, x_{m-2}, 1). \text{ q.e.d.}$$

Exemplul 5.6 Să translatăm $\mathbf{f} = 01101110$ într-un polinom boolean de 3 variabile. Vom aplica pe etape formula din Propoziția 20.1:

$$\begin{aligned} \mathbf{f} &= 0110 + [0110 + 1110]\mathbf{x}_2 = 0110 + 1000\mathbf{x}_2 = \\ &= (01 + [01 + 10]\mathbf{x}_1) + (10 + [10 + 00]\mathbf{x}_1)\mathbf{x}_2 = 01 + 11\mathbf{x}_1 + 10\mathbf{x}_2 + 10\mathbf{x}_1\mathbf{x}_2 = \\ &= (0 + [0 + 1]\mathbf{x}_0) + (1 + [1 + 1]\mathbf{x}_0)\mathbf{x}_1 + (1 + [1 + 0]\mathbf{x}_0)\mathbf{x}_2 + (1 + [1 + 0]\mathbf{x}_0)\mathbf{x}_1\mathbf{x}_2 = \\ &= \mathbf{x}_0 + \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_0\mathbf{x}_2 + \mathbf{x}_1\mathbf{x}_2 + \mathbf{x}_0\mathbf{x}_1\mathbf{x}_2. \end{aligned}$$

Teorema 5.1 Spațiul liniar Z_2^n , ($n = 2^m$) are o bază formată din toate monoamele booleene:

$$\begin{aligned} &\mathbf{1} \\ &\mathbf{x}_i \quad (0 \leq i \leq m-1) \\ &\mathbf{x}_i\mathbf{x}_j \quad (0 \leq i, j \leq m-1, \quad i \neq j) \\ &\vdots \\ &\mathbf{x}_0\mathbf{x}_1 \dots \mathbf{x}_{m-1} \end{aligned}$$

Demonstrație: Fiecare cuvânt de lungime $n = 2^m$ este o funcție booleană de m nedeterminate, care poate fi exprimată printr-un polinom boolean. Monoamele booleene pot fi considerate în Z_2^n și sunt evident liniar independente. În plus, deoarece pentru fiecare $k = 0, 1, \dots, m$ există C_m^k monoame de grad k , numărul lor total va fi $\sum_{k=0}^m C_m^k = 2^m = n$, adică dimensiunea spațiului liniar. q.e.d.

5.1.2 Coduri Reed - Muller

Definiția 5.3 Se numește cod Reed - Muller de lungime $n = 2^m$ și grad r ($0 \leq r \leq m$), codul liniar $\mathcal{RM}(r, m)$ al tuturor cuvintelor binare de lungime n care au - ca polinoame booleene - gradul maxim r .

Exemplul 5.7 $\mathcal{RM}(0, m)$ este format din toate polinoamele de grad cel mult 0, adică $\mathbf{0}$ și $\mathbf{1}$. Deci $\mathcal{RM}(0, m)$ este codul cu repetiție de lungime 2^m .

Exemplul 5.8 $\mathcal{RM}(1, m)$ are baza $\mathbf{1}, \mathbf{x}_0, \dots, \mathbf{x}_{m-1}$; orice polinom de grad cel mult 1 se poate scrie ca sumă de o parte din aceste $m + 1$ polinoame (liniar independente). Deci, $\mathcal{RM}(1, m)$ este un $(2^m, m + 1)$ - cod liniar.

De exemplu, $\mathcal{RM}(1, 3)$ are matricea generatoare:

$$G = \begin{pmatrix} \mathbf{1} \\ \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

care generează codul Hamming extins $(8, 4)$.

Similar, $\mathcal{RM}(1, 4)$ este un $(16, 5)$ - cod liniar binar, iar $\mathcal{RM}(1, 5)$ este un $(32, 6)$ - cod liniar, folosit în 1969 de programul Mariner, după cum s-a menționat anterior.

Propoziția 5.2 $\mathcal{RM}(r, m)$ are $k = C_m^0 + C_m^1 + \dots + C_m^r$ simboluri de informație.

Demonstrație: Codul $\mathcal{RM}(r, m)$ are ca bază monoamele booleene

$$\{\mathbf{1}\} \cup \{\mathbf{x}_{i_1} \mathbf{x}_{i_2} \dots \mathbf{x}_{i_s} \mid s \leq r, 0 \leq i_1 < i_2 < \dots < i_s < m\}.$$

Numărul acestor monoame este $C_m^0 + C_m^1 + \dots + C_m^r$. Cum ele sunt liniar independente, vor forma liniile matricii generatoare G a codului; ori numărul de linii este egal cu numărul de simboluri de informație. q.e.d.

Propoziția 5.3 *Dualul codului $\mathcal{RM}(r, m)$ este codul $\mathcal{RM}(m - r - 1, m)$.*

Demonstrație: Trebuie arătat că cele două coduri au baze ortogonale și suma dimensiunilor lor este egală cu suma întregului spațiu.

A: Fie $\mathbf{v}_{i_1} \mathbf{v}_{i_2} \dots \mathbf{v}_{i_p}$ ($p \leq r$) un monom din baza codului $\mathcal{RM}(r, m)$ și $\mathbf{v}_{j_1} \mathbf{v}_{j_2} \dots \mathbf{v}_{j_s}$ ($s \leq m - r - 1$) un monom din baza codului $\mathcal{RM}(m - r - 1, m)$. Produsul lor are t ($t \leq r + m - r - 1 = m - 1$) variabile distincte (variabilele comune apar o singură dată), deci ponderea lui este (conform $(*)$) $2^{m-t} \geq 2$. Fiind un număr par, rezultă că produsul scalar (adică suma în binar a termenilor) este 0.

B: Suma dimensiunilor celor două coduri este:

$$\sum_{i=0}^r C_m^i + \sum_{i=0}^{m-r-1} C_m^i = \sum_{i=0}^r C_m^i + \sum_{i=r+1}^m C_m^i = \sum_{i=0}^m C_m^i = 2^m = n. \text{ q.e.d.}$$

Exemplul 5.9 $\mathcal{RM}(m - 2, m)$ este codul Hamming extins de lungime 2^m .

Într-adevăr, codul său dual este $\mathcal{RM}(m - (m - 2) - 1, m) = \mathcal{RM}(1, m)$, deci $\mathcal{RM}(m - 2, m)$ are matricea de control

$$H = \begin{pmatrix} \mathbf{1} \\ \mathbf{x}_0 \\ \vdots \\ \mathbf{x}_m \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & \dots & 0 & 1 & 0 & 1 \\ & & & & \vdots & & & & \\ 0 & 0 & 0 & 0 & \dots & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Dacă adunăm prima linie la toate celelalte și apoi o permutăm cu ultima linie, se obține altă matrice de control a codului:

$$H \sim \left(\begin{array}{cccc|cccc} 1 & 0 & 1 & 0 & \dots & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & \dots & 1 & 1 & 0 & 0 \\ & & & & \vdots & & & & \\ 1 & 1 & 1 & 1 & \dots & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 \end{array} \right).$$

După eliminarea ultimei linii și coloane se obține un $(2^m - 1, m)$ - cod ale cărui coloane sunt nenule și diferite două câte două, adică matricea de control H_0 a unui cod Hamming binar. Deci H este matricea de control a unui cod Hamming extins.

Codificarea mesajelor cu un cod $R - M$ se realizează normal, înmulțind mesajul de informație cu matricea generatoare. În acest fel biții de informație devin coeficienții polinomului boolean corespunzător. De exemplu, în $\mathcal{RM}(1, m)$ codificarea celor $m + 1$ caractere de informație este:

$$(a_1, a_2, \dots, a_{m+1}) \begin{pmatrix} \mathbf{1} \\ \mathbf{x}_0 \\ \vdots \\ \mathbf{x}_{m-1} \end{pmatrix} = a_1 \mathbf{1} + a_2 \mathbf{x}_0 + \dots + a_{m+1} \mathbf{x}_{m-1}.$$

5.2 Definirea recursivă a codurilor R - M

Să introducem o altă modalitate de definire a codurilor Reed - Muller, nu prin polinoame booleene, ci prin construcție recursivă.

Definiția 5.4 Fie $m \geq 0$ un număr natural. Se definește codul Reed - Muller $\mathcal{RM}(r, m)$ de ordin r ($0 \leq r \leq m$) și lungime $n = 2^m$ astfel:

- $\mathcal{RM}(0, m) = \{00 \dots 0, 11 \dots 1\}$, $\mathcal{RM}(m, m) = Z_2^n$.
- $\mathcal{RM}(p, m) = \{[\mathbf{a}, \mathbf{a} + \mathbf{b}] \mid \mathbf{a} \in \mathcal{RM}(p, m - 1), \mathbf{b} \in \mathcal{RM}(p - 1, m - 1)\}, 0 < p < r$.

S-a notat cu $[\mathbf{x}, \mathbf{y}]$ un cuvânt de lungime 2^m scris ca alăturare de două subcuvinte de lungimi egale (2^{m-1}), separate prin virgulă (similar notației de la codurile Golay).

Exemplul 5.10 $\mathcal{RM}(0, 0) = \{0, 1\}$

$$\mathcal{RM}(0, 1) = \{00, 11\}, \quad \mathcal{RM}(1, 1) = \{00, 01, 10, 11\} = Z_2^2$$

$$\mathcal{RM}(0, 2) = \{0000, 1111\}, \quad \mathcal{RM}(2, 2) = Z_2^4$$

$$\begin{aligned} \mathcal{RM}(1, 2) &= \{[\mathbf{a}, \mathbf{a} + \mathbf{b}] \mid \mathbf{a} \in \{00, 01, 10, 11\}, \mathbf{b} \in \{00, 11\}\} = \\ &= \{0000, 0011, 0100, 0111, 1000, 1011, 1100, 1111\}. \end{aligned}$$

În mod similar se poate da o definiție recursivă a matricii generatoare $G(r, m)$ pentru codul $\mathcal{RM}(r, m)$.

- $G(0, m) = (11 \dots 1)$;
- pentru $0 < p < r < m$, $G(p, m) = \begin{pmatrix} G(p, m - 1) & G(p, m - 1) \\ \mathbf{0} & G(p - 1, m - 1) \end{pmatrix}$;
- $G(m, m) = \begin{pmatrix} G(m - 1, m) \\ 00 \dots 01 \end{pmatrix}$.

Teorema 5.2 $G(r, m)$ este matrice generatoare pentru codul $\mathcal{RM}(r, m)$.

Demonstrație: Se verifică prin inducție după r . q.e.d.

Exemplul 5.11 Să considerăm $r = 2$; atunci lungimea este $n = 2^2 = 4$ și pentru $r = 1, 2$ avem

$$G(1, 2) = \begin{pmatrix} G(1, 1) & G(1, 1) \\ \mathbf{0} & G(0, 1) \end{pmatrix} \quad G(2, 2) = \begin{pmatrix} G(1, 2) \\ 0001 \end{pmatrix}.$$

Din definiție, matricile generatoare pentru $\mathcal{RM}(0, 1)$ și $\mathcal{RM}(1, 1)$ sunt

$$\begin{aligned} G(0, 1) &= (1 \ 1), \quad G(1, 1) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \text{ așa că} \\ G(1, 2) &= \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad G(2, 2) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \end{aligned}$$

Propoziția 5.4 $\mathcal{RM}(r-1, m) \subseteq \mathcal{RM}(r, m)$.

Demonstrație: Să considerăm inițial matricea

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

Pentru că $\mathbf{1}$ este prima linie a lui $G(1, m-1)$, cuvântul $[\mathbf{1}, \mathbf{1}]$ formează prima linie a matricii $(G(1, m-1) \ G(1, m-1))$. Deci $\mathcal{RM}(0, m) = \{\mathbf{0}, \mathbf{1}\}$ este conținut în codul $\mathcal{RM}(1, m)$.

În general, deoarece $G(r-1, m-1)$ este submatrice a lui $G(r, m-1)$ și $G(r-2, m-1)$ este o submatrice a lui $G(r-1, m-1)$, este evident că

$$G(r-1, m) = \begin{pmatrix} G(r-1, m-1) & G(r-1, m-1) \\ \mathbf{0} & G(r-2, m-1) \end{pmatrix}$$

este o submatrice a lui $G(r, m)$, deci $\mathcal{RM}(r-1, m)$ este subcod al lui $\mathcal{RM}(r, m)$. q.e.d.

Teorema 5.3 $\mathcal{RM}(r, m)$ are distanța $d = 2^{m-r}$.

Demonstrație: Vom folosi o inducție după r :

Pentru $r = 0$, evident: $\mathcal{RM}(0, m)$ fiind codul cu repetiție, distanța sa este $d = n = 2^m$.

La pasul II, deoarece

$\mathcal{RM}(r, m) = \{[\mathbf{x}, \mathbf{x} + \mathbf{y}] | \mathbf{x} \in \mathcal{RM}(r, m-1), \mathbf{y} \in \mathcal{RM}(r-1, m-1)\}$ și $\mathcal{RM}(r-1, m-1) \subseteq \mathcal{RM}(r, m-1)$ (Propoziția 11.3), rezultă $\mathbf{x} + \mathbf{y} \in \mathcal{RM}(r, m-1)$.

Dacă $\mathbf{x} \neq \mathbf{y}$, conform ipotezei de inducție $w(\mathbf{x} + \mathbf{y}) \geq 2^{m-1-r}$. Cum și $w(\mathbf{x}) \geq 2^{m-1-r}$, putem scrie $w([\mathbf{x} + \mathbf{y}, \mathbf{x}]) = w(\mathbf{x} + \mathbf{y}) + w(\mathbf{x}) \geq 2^{m-r}$.

Dacă $\mathbf{x} = \mathbf{y}$, atunci $[\mathbf{x}, \mathbf{x} + \mathbf{y}] = [\mathbf{y}, \mathbf{0}]$; dar $\mathbf{y} \in \mathcal{RM}(r-1, m-1)$ și deci $w([\mathbf{y}, \mathbf{0}]) = w(\mathbf{y}) \geq 2^{m-r}$.

Cum orice linie a matricii generatoare este cuvânt - cod, iar ultima linie are ponderea exact 2^{m-r} , demonstrația este încheiată. q.e.d.

5.3 Definirea geometrică a codurilor R-M

Codurile Reed - Muller mai pot fi definite și geometric - prin folosirea spațiilor afine. Avantajul acestei reprezentări constă în modalitatea mai simplă de aplicare a algoritmilor de decodificare.

Pentru ușurința descrierii am construit întâi cazul tri-dimensional. De asemenea, pentru a vedea echivalența cu definirea anterioară a codurilor Reed - Muller, vom face permanent legătura cu polinoamele booleene (sau cu funcțiile lor caracteristice).

5.3.1 Cazul 3 -dimensional

Spațiul euclidian 3 - dimensional binar este mulțimea $\{(a, b, c) | a, b, c \in Z_2\}$. Spre deosebire de spațiul euclidian obișnuit - unde cele trei coordonate luau valori în \mathcal{R} - aici numărul punctelor este finit: numai 8. Ele pot fi listate, renotându-le astfel:

Punct	Funcție caracteristică
$\mathbf{p}_0 = 000$	00000001
$\mathbf{p}_1 = 001$	00000010
$\mathbf{p}_2 = 010$	00000100
$\mathbf{p}_3 = 011$	00001000
$\mathbf{p}_4 = 100$	00010000
$\mathbf{p}_5 = 101$	00100000
$\mathbf{p}_6 = 110$	01000000
$\mathbf{p}_7 = 111$	10000000

Dreptele pot fi definite în geometria euclidiană prin expresii de forma

$$\mathbf{a} + t\mathbf{b} \quad \mathbf{a}, \mathbf{b} \in Z_2^3, \mathbf{b} \neq \mathbf{0}.$$

unde t este un parametru binar ($t \in \{0, 1\}$). Deci o dreaptă în spațiul 3 - dimensional binar are numai 2 puncte: $\mathbf{a}, \mathbf{a} + \mathbf{b}$. Invers, orice pereche de două puncte distincte \mathbf{a}, \mathbf{a}' formează o dreaptă, anume $\mathbf{a} + t(\mathbf{a}' - \mathbf{a})$. Putem astfel să considerăm dreptele ca fiind totalitatea celor $C_8^2 = 28$ submulțimi de câte două puncte

$$\{\mathbf{p}_0, \mathbf{p}_1\}, \{\mathbf{p}_0, \mathbf{p}_2\}, \dots, \{\mathbf{p}_6, \mathbf{p}_7\}.$$

În mod similar, *planele* din geometria euclidiană sunt definite

$$\mathbf{a} + t_1\mathbf{b} + t_2\mathbf{c}, \quad \mathbf{a}, \mathbf{b}, \mathbf{c} \in Z_2^3, \quad \mathbf{b}, \mathbf{c} \text{ liniar independente,}$$

unde t_1, t_2 sunt parametri binari.

Un plan este format deci din patru puncte: $\mathbf{a}, \mathbf{a} + \mathbf{b}, \mathbf{a} + \mathbf{c}, \mathbf{a} + \mathbf{b} + \mathbf{c}$. Aparent, deși numărul planelor în geometria euclidiană binară 3 - dimensională ar trebui să fie $C_8^3 = 70$, condiția de liniar independență reduce acest număr la 14:

Plan	Funcție caracteristică	Polinom boolean
$\{\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_5, \mathbf{p}_7\}$	10101010	\mathbf{x}_0
$\{\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_6, \mathbf{p}_7\}$	11001100	\mathbf{x}_1
$\{\mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_6, \mathbf{p}_7\}$	11110000	\mathbf{x}_2
$\{\mathbf{p}_0, \mathbf{p}_2, \mathbf{p}_4, \mathbf{p}_6\}$	01010101	$\mathbf{1} + \mathbf{x}_0$
$\{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_4, \mathbf{p}_5\}$	00110011	$\mathbf{1} + \mathbf{x}_1$
$\{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$	00001111	$\mathbf{1} + \mathbf{x}_2$
$\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_5, \mathbf{p}_6\}$	01100110	$\mathbf{x}_0 + \mathbf{x}_1$
$\{\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_6\}$	01011010	$\mathbf{x}_0 + \mathbf{x}_2$
$\{\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5\}$	00111100	$\mathbf{x}_1 + \mathbf{x}_2$
$\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_4, \mathbf{p}_7\}$	10010110	$\mathbf{x}_0 + \mathbf{x}_1 + \mathbf{x}_2$
$\{\mathbf{p}_0, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_7\}$	10011001	$\mathbf{1} + \mathbf{x}_0 + \mathbf{x}_1$
$\{\mathbf{p}_0, \mathbf{p}_2, \mathbf{p}_5, \mathbf{p}_7\}$	10100101	$\mathbf{1} + \mathbf{x}_0 + \mathbf{x}_2$
$\{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_6, \mathbf{p}_7\}$	11000011	$\mathbf{1} + \mathbf{x}_1 + \mathbf{x}_2$
$\{\mathbf{p}_0, \mathbf{p}_3, \mathbf{p}_5, \mathbf{p}_6\}$	01101001	$\mathbf{1} + \mathbf{x}_0 + \mathbf{x}_1 + \mathbf{x}_2$

Tabelul 5.2

În general, un plan este descris de ecuația generală

$$h_0\mathbf{x}_0 + h_1\mathbf{x}_1 + h_2\mathbf{x}_2 = c$$

care definește un subspațiu 2 - dimensional al lui Z_2^3 .

Din faptul că orice dreaptă este o intersecție de două plane, ea poate fi descrisă printr-un sistem de două ecuații:

$$h_0\mathbf{x}_0 + h_1\mathbf{x}_1 + h_2\mathbf{x}_2 = c, \quad h'_0\mathbf{x}_0 + h'_1\mathbf{x}_1 + h'_2\mathbf{x}_2 = c'.$$

Dreptele și planele sunt exemple de spații afine. Un *spațiu afin* în Z_2^3 este o mulțime de forma

$$C_a = \mathbf{a} + C = \{\mathbf{a} + \mathbf{b} \mid \mathbf{b} \in C\}$$

unde $\mathbf{a} \in Z_2^3$ iar C este un subspațiu linear din Z_2^3 . Dacă C este un spațiu de dimensiune s , atunci C_a este un s - *spațiu afin*.

În particular, dreptele sunt 1 - spații afine, iar planele: 2 - spații afine. Pentru fiecare punct \mathbf{p}_i avem un 0 - spațiu afin, și – în sfârșit – există un 3 - spațiu afin unic - Z_2^3 .

Orice spațiu afin L poate fi descris de un cuvânt binar $\mathbf{f}_L = f_7 \dots f_1 f_0$ definit prin

$$f_i = \begin{cases} 1 & \text{dacă } \mathbf{p}_i \in L, \quad (0 \leq i \leq 7) \\ 0 & \text{altfel} \end{cases}$$

Cuvântul \mathbf{f}_L (sau funcția booleană de trei variabile corespunzătoare) se numește *funcția caracteristică* a spațiului afin L (tabelele anterioare listează aceste funcții pentru 0 și 2 - spații afine).

Fiind date două spații afine L, L' , intersecția lor $L \cap L'$ este caracterizată de produsul logic $\mathbf{f}_L \mathbf{f}_{L'}$.

Exemplul 5.12 Primele două plane din Tabelul 5.2 se intersectează după linia $\{\mathbf{p}_3, \mathbf{p}_7\}$. Produsul logic al funcțiilor lor caracteristice este $\mathbf{x}_0\mathbf{x}_1 = 10001000$, care se poate verifica imediat ca fiind funcția caracteristică a dreptei $\{\mathbf{p}_3, \mathbf{p}_7\}$.

5.3.2 Cazul m - dimensional

Vom extinde construcțiile anterioare la un cadru mai general, al geometriei euclidiene m - dimensionale binare. Punctele (elementele lui Z_2^m) pot fi renotate după extensia binară a indicilor; mai clar, vom scrie $Z_2^m = \{\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_{2^m-1}\}$ unde \mathbf{p}_i reprezintă scrierea în binar (pe m biți) a numărului i ($0 \leq i \leq 2^m - 1$). Deci

$$\mathbf{p}_0 = 000 \dots 00, \quad \mathbf{p}_1 = 000 \dots 01, \quad \mathbf{p}_2 = 000 \dots 10, \quad \dots, \quad \mathbf{p}_{2^m-1} = 111 \dots 11.$$

Definiția 5.5 Fie C un subspațiu linear r - dimensional al lui Z_2^m și $\mathbf{a} \in Z_2^m$. Mulțimea

$$C_{\mathbf{a}} = \mathbf{a} + C = \{\mathbf{a} + \mathbf{b} \mid \mathbf{b} \in C\}$$

se numește r - *spațiu afin modulo C în geometria euclidiană m - dimensională*.

Un $(m - 1)$ - *spațiu afin* se numește "hiperplan".

Dacă $\mathbf{b}_1, \dots, \mathbf{b}_r$ este o bază a lui C , r - spațiul afin $C_{\mathbf{a}}$ se notează

$$\mathbf{a} + t_1 \mathbf{b}_1 + \dots + t_r \mathbf{b}_r.$$

El are 2^r puncte (date de variantele de alegere ale parametrilor binari t_i , $1 \leq i \leq r$).

O altă modalitate de notare a r - spațiilor afine se realizează cu ajutorul sistemelor de ecuații liniare: astfel, dacă C este definit ca mulțimea soluțiilor sistemului $H\mathbf{x}^T = \mathbf{0}^T$, atunci $C_{\mathbf{a}}$ este dat de mulțimea soluțiilor sistemului

$$H\mathbf{x}^T = H\mathbf{a}^T.$$

Acest sistem are $m - r$ ecuații. În particular, un hiperplan este definit printr-o singură ecuație:

$$h_0 x_0 + h_1 x_1 + \dots + h_{m-1} x_{m-1} = c.$$

Exemplul 5.13 Un 0 - spațiu afin cuprinde un singur punct. Există deci 2^m 0 - spații afine distincte: $\{\mathbf{p}_0\}, \{\mathbf{p}_1\}, \dots, \{\mathbf{p}_{2^m-1}\}$.

Similar, orice 1 - spațiu afin (sau "dreaptă") este o mulțime formată din două puncte

$$\mathbf{a} + t\mathbf{b} \equiv \{\mathbf{a}, \mathbf{a} + \mathbf{b}\},$$

și invers, orice mulțime de două puncte distincte formează un 1 - spațiu afin.

Există deci $C_{2^m}^2 = 1$ - spații afine.

Exemplul 5.14 Fie P_i spațiul afin definit de ecuația $x_i = 1$. Deci P_i este mulțimea punctelor \mathbf{p}_k care au 1 pe poziția i . De exemplu $P_0 = \{\mathbf{p}_1, \mathbf{p}_3, \dots, \mathbf{p}_{2^m-1}\}$.

Fiecare P_i este un hiperplan și - deoarece \mathbf{p}_{2^i} are un singur 1 pe poziția i și 0 în rest, putem scrie

$$P_i \equiv \mathbf{p}_{2^i} + C$$

unde C este spațiul liniar definit de ecuația $x_i = 0$, (deci tot un hiperplan).

Propoziția 5.5 Există $2(2^m - 1)$ hiperplane.

Demonstrație: Deoarece se pot construi $2^m - 1$ ecuații cu coeficienți binari (nu toți nuli) și variabile x_0, x_1, \dots, x_{m-1} , Z_2^m are $2^m - 1$ subspații C de dimensiune $m - 1$. Fiecare din ele are 2^{m-1} puncte, deci vor exista $\frac{2^m}{2^{m-1}} = 2$ spații afine modulo C . q.e.d.

Exemplul 5.15 Pentru $i \neq j$, intersecția $P_i \cap P_j$ (mulțimea punctelor care au 1 pe pozițiile i și j) este un $(m - 2)$ - spațiu afin. Într-adevăr, dacă se ia $\mathbf{a} = \mathbf{p}_{2^i+2^j}$, avem

$$P_i \cap P_j \equiv \mathbf{a} + C,$$

unde C este determinat de ecuațiile $x_i = 0$, $x_j = 0$ (deci C are dimensiunea $m - 2$).

Definiția 5.6 Funcția caracteristică a unui r - spațiu afin L este cuvântul binar $\mathbf{f}_L = f_{2^m-1} \dots f_1 f_0$ definit prin

$$f_i = \begin{cases} 1 & \text{dacă } \mathbf{p}_i \in L, \\ 0 & \text{altfel} \end{cases} \quad (0 \leq i \leq 2^m - 1)$$

Funcția caracteristică poate fi interpretată ca un polinom boolean $f_L(x_0, \dots, x_{m-1})$. Din proprietățile acestor polinoame rezultă

$$a_0 a_1 \dots a_{m-1} \in L \iff f_L(a_0, a_1, \dots, a_{m-1}) = 1$$

Observația 5.2

1. Singurul m - spațiu afin (Z_2^m) are funcția caracteristică $\mathbf{1} = 11 \dots 1$.
2. Un hiperplan P_i are funcția $f_{P_i} = x_i$.
3. Fie L un hiperplan definit de ecuația $h_0 x_0 + \dots + h_{m-1} x_{m-1} = c$. Funcția sa caracteristică va fi un polinom boolean de gradul 1, anume

$$f_L(x_0, \dots, x_{m-1}) = h_0 x_0 + \dots + h_{m-1} x_{m-1} + c + 1$$

Această relație rezultă din faptul că un punct $a_0 \dots a_{m-1}$ este în plan dacă și numai dacă $h_0 a_0 + \dots + h_{m-1} a_{m-1} = c$, adică $f_L(a_0, \dots, a_{m-1}) = 1$.

4. Pentru două spații afine L, L' , funcția caracteristică a intersecției $L \cap L'$ este $\mathbf{f}_L \mathbf{f}_{L'}$. Astfel, pentru $P_i \cap P_j$ (care este un $(m-2)$ - spațiu afin) funcția caracteristică este $\mathbf{x}_i \mathbf{x}_j$.
Mai general, polinomul boolean $\mathbf{x}_{i_1} \mathbf{x}_{i_2} \dots \mathbf{x}_{i_s}$ este funcția caracteristică a unui $(m-s)$ - spațiu afin.

Teorema 5.4 Funcția caracteristică a unui r - spațiu afin este un polinom boolean de gradul $m - r$.

Demonstrație: Un r - spațiu afin L este definit ca soluția sistemului de ecuații $H\mathbf{x}^T = \mathbf{c}^T$, sau, detaliind,

$$\sum_{j=0}^{m-1} h_{ij} x_j = c_i, \quad 1 \leq i \leq m - r.$$

Aceste ecuații se pot scrie

$$\sum_{j=0}^{m-1} h_{ij} x_j + c_i + 1 = 1, \quad 1 \leq i \leq m - r.$$

Atunci, polinomul boolean de grad $m - r$

$$f(x_0, \dots, x_{m-1}) = \prod_{i=1}^{m-r} \left(\sum_{j=0}^{m-1} h_{ij} x_j + c_i + 1 \right)$$

poate fi considerat funcția caracteristică a lui L . q.e.d.

Definiția 5.7 $\mathcal{RM}(r, m)$ este codul liniar generat de toate funcțiile caracteristice ale spațiilor afine de dimensiune cel puțin $m - r$ în geometria euclidiană m - dimensională peste Z_2 .

Faptul că aceasta coincide cu definiția anterioară a codurilor Reed - Muller rezultă din construcția spațiilor afine: $\mathcal{RM}(r, m)$ conține toate funcțiile caracteristice ale s - spațiilor afine, unde $s \geq m - r$. Faptul că aceste funcții generează tot spațiul $\mathcal{RM}(r, m)$ rezultă din Observația 5.2, punctul 4.

Exemplul 5.16 Codul $\mathcal{RM}(1, 3)$ este generat de funcțiile caracteristice ale tuturor planelor. Orice astfel de funcție este un polinom de trei variabile de gradul 1.

Codul $\mathcal{RM}(2, 3)$ este generat de funcțiile caracteristice ale tuturor planelor și liniilor. Cum o linie este intersecția a două plane, funcția sa caracteristică este produsul a două polinoame de gradul 1, deci un polinom de gradul 2 (cuvânt - cod din $\mathcal{RM}(2, 3)$).

5.4 Exerciții

5.1 Ce polinom boolean are ultima linie a tabelului de adevăr:

10100110 1010011010100110 0101001110011100

5.2 Determinați tabela de adevăr a polinomului boolean $\mathbf{1} + \mathbf{x}_0 + \mathbf{x}_1\mathbf{x}_2$:

1. Ca funcție de trei variabile;
2. Ca funcție de patru variabile.

5.3 Demonstrați afirmația (*).

5.4 Găsiți un $(15, 5)$ - cod liniar binar corector de 3 erori (folosiți un cod $R - M$ relaxat).

5.5 Fiind dat codul $\mathcal{RM}(1, 3)$, codificați toate mesajele de informație posibile. Același lucru pentru codul $\mathcal{RM}(2, 3)$.

5.6 Demonstrați Teorema 5.2

5.7 Construiți matricile generatoare $G(1, 3)$, $G(2, 3)$, $G(r, 4)$, $r = 0, 1, 2$.

5.8 Demonstrați că $G(r, m)$ are $k + 1$ linii, unde k este dat de Propoziția 20.2

5.9 Calculați numărul de 2 - spații afine în geometria euclidiană peste Z_2^m .

5.10 Este orice funcție booleană funcția caracteristică a unui anumit spațiu afin? Caracterizați astfel de funcții.

5.11 Orice funcție caracteristică a unui $(r + 1)$ - spațiu afin aparține codului dual lui $\mathcal{RM}(r, m)$.

5.12 Să se arate că $\mathcal{RM}(2, 5)$ este auto - dual. Să se determine toate codurile \mathcal{RM} auto - duale.

Capitolul 6

Decodificarea codurilor Reed - Muller

Avantajul principal al codurilor Reed - Muller constă în facilitatea decodificării, facilitate bazată pe o tehnică diferită de cea de până acum. Această tehnică, numită *decodificare majoritară* nu apelează la ideea de sindrom, ci corectează direct biții modificați, folosind diverse proprietăți ale cuvântului recepționat.

6.1 Decodificarea majoritară

Să prezentăm pe scurt principiile generale ale decodificării majoritare.

Vom începe cu un exemplu foarte simplu:

Exemplul 6.1 Fie codul binar cu repetiție de lungime $2n + 1$:

$$C = \{\underbrace{00 \dots 0}_{2n+1}, \underbrace{11 \dots 1}_{2n+1}\}.$$

Ca sistem de ecuații de control se poate lua

$$\begin{cases} x_1 + x_2 & = 0 \\ x_1 + x_3 & = 0 \\ & \vdots \\ x_1 + x_{2n+1} & = 0 \end{cases}$$

Dacă se recepționează cuvântul $\mathbf{y} = \mathbf{x} + \mathbf{e}$, la înlocuirea lui în sistem, acesta devine

$$y_1 + y_i = e_1 + e_i, \quad (2 \leq i \leq 2n + 1)$$

Dacă mai mult de n din cele $2n$ expresii $y_1 + y_i$ iau valoarea 1, aceasta înseamnă că:

- Au apărut mai puțin de n erori, printre care și pe prima poziție ($e_1 = 1$), sau
- Au apărut mai mult de n erori, dar nu pe prima poziție ($e_1 = 0$).

Din aceste două variante, prima este mai probabilă. Deci valoarea majoritară pe care o iau cele n valori $y_1 + y_i$ va fi valoarea lui e_1 .

Definiția 6.1 Fie $A_{n,k}$ un cod liniar. Un set de ecuații de control pentru $A_{n,k}$ este ortogonal pe mulțimea de poziții $P \subset \{1, 2, \dots, n\}$ dacă și numai dacă:

1. $\forall i \in P$, termenul x_i apare (cu coeficient nenul) în fiecare ecuație;
2. $\forall i \notin P$, termenul x_i apare cel mult într-o ecuație.

Decodificarea se realizează caracter cu caracter, după următorul procedeu:

Fie $\mathbf{a} = a_1 \dots a_n$ cuvântul recepționat și i ($1 \leq i \leq n$) o poziție.
 Se construiește mulțimea maximală de ecuații de control ortogonale pe poziția i .
 Fie e_i valoarea obținută în majoritatea acestor ecuații. Al i -lea caracter decodificat este $a_i + e_i$.

Dacă fiecare ecuație de control ortogonală pe poziția i se scrie

$$a_i = f(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n),$$

atunci decodificarea majoritară revine la a decodifica pe a_i prin valoarea care apare cel mai frecvent în evaluările expresiilor f .

Exemplul 6.2 Fie dualul (15, 11) - codului Hamming cu matricea de control

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

Pentru fiecare pereche de coloane distincte din H există o a treia coloană astfel încât suma celor trei coloane este $\mathbf{0}$.

Deci, fiecare cuvânt de pondere 3 al codului Hamming dă o ecuație de control cu trei termeni pentru codul dual; în acest fel se obține pentru fiecare caracter x_i câte un sistem format din 7 ecuații ortogonale pe poziția i . Astfel,

- pentru x_1 ele sunt: $x_1 = x_2 + x_3 = x_4 + x_5 = x_6 + x_7 = x_8 + x_9 = x_{10} + x_{11} = x_{12} + x_{13} = x_{14} + x_{15}$;

- pentru x_2 : $x_2 = x_1 + x_3 = x_4 + x_6 = x_5 + x_7 = x_8 + x_{10} = x_9 + x_{11} = x_{12} + x_{14} = x_{13} + x_{15}$ etc.

Codul dual are distanța minimă $d = 8$; deci el poate corecta maxim 3 erori.

Dacă în mesajul primit \mathbf{x} apar cel mult 3 erori, atunci un sistem de ecuații ortogonal pe x_1 va avea cel mult trei ecuații care să dea pentru x_1 o valoare greșită și cel puțin cinci cu valoarea calculată corect. Valoarea găsită majoritar va fi cea în care se decodifică primul caracter.

Procedeu se reia pentru x_2, x_3, \dots

Să presupunem de exemplu că s-a recepționat mesajul 001011110101011. Pentru decodificarea primului caracter vom calcula sumele

$$0 + 1, 0 + 1, 1 + 1, 1 + 0, 1 + 0, 1 + 0, 1 + 1.$$

Se obțin 5 valori de 1 și două de 0; deci, primul simbol este 1.

Procedând similar pentru fiecare poziție, se ajunge la cuvântul 101010110101010.

Această metodă este nu numai ușor de implementat, dar are adesea și alte completări (gen "bitul de control al parității") care fac posibilă corectarea mai multor erori.

În această prelegere vom construi trei algoritmi de decodificare a codurilor $R-M$. Doi din ei sunt bazați pe modurile de reprezentare (geometric și algebric) ale acestor coduri; al treilea este un algoritm care folosește definiția recursivă a codurilor $R-M$ și este prezentat numai pentru cazul $r=1$.

Toți algoritmi sunt foarte ușor de implementat, utilizarea unuia sau a altuia depinzând doar de criterii particulare în alegerea parametrilor codului.

6.2 Algoritm geometric de decodificare majoritară a codurilor $R-M$

Deoarece distanța unui cod $\mathcal{RM}(r, m)$ este $d = 2^{m-r}$, el va fi capabil să corecteze până la $2^{m-r-1} - 1$ erori. În cele ce urmează vom presupune că s-a recepționat cuvântul $\mathbf{y} = y_{2^m-1} \dots y_1 y_0$, în care au fost modificate maxim $2^{m-r-1} - 1$ poziții. Problema este de a determina pentru fiecare i ($0 \leq i \leq 2^m - 1$) dacă bitul y_i trebuie corectat sau nu. Sau – altfel spus – de a vedea dacă poziția lui \mathbf{y} corespunde 0 - spațiului afin $\{\mathbf{p}_i\}$ trebuie sau nu corectată.

Pentru a folosi proprietățile codurilor Reed - Muller exprimate de spațiile afine, vom reformula totul în maniera următoare:

Pentru fiecare s - spațiu afin L ($0 \leq s \leq r+1$) vom cerceta dacă pozițiile cuvântului recepționat \mathbf{y} corespund punctelor lui L (adică acei biți y_i pentru care $\mathbf{p}_i \in L$) sunt modificați sau nu.

Vom da întâi câteva noțiuni și notații ajutătoare pentru a simplifica demonstrațiile.

Definiția 6.2 Fie G un grup, C un subgrup al său și $x \in G$. Se numește "subgrup modulo C " al lui G mulțimea

$$C_x = x + C = \{x + a \mid a \in C\}.$$

Lema 6.1 Subgrupurile unui grup modulo un subgrup arbitrar C , au următoarele proprietăți:

1. $\forall a \in G, \exists x \in G, a \in C_x$;
2. Dacă $x \neq y$ atunci $C_x \cap C_y = \emptyset$ sau $C_x = C_y$;
3. Dacă $a, b \in C_x$ atunci $a - b \in C$;
4. $\forall x, \text{card}(C_x) = \text{card}(C)$.

Demonstrație: Exercițiu.

Teorema principală a acestui paragraf este:

Teorema 6.1 Orice s - spațiu afin din geometria binară m - dimensională este conținut în exact $2^{m-s} - 1$ ($s+1$) - spații afine distincte. În plus, orice punct din afara lui L se află în unul și numai unul din aceste ($s+1$) - spații afine.

Demonstrație: **I:** Să arătăm întâi că orice s - subspațiu liniar $C \subseteq Z_2^m$ este conținut în exact $2^{m-s} - 1$ subspații distincte de dimensiune $s+1$.

Orice ($s+1$) - spațiu care conține C este de forma

$$\overline{C} = C + t\mathbf{b} \equiv \{\mathbf{a} + t\mathbf{b} \mid \mathbf{a} \in C, t = 0, 1\},$$

unde $\mathbf{b} \notin C$ este un punct arbitrar fixat. Aceasta rezultă imediat din faptul că orice bază a lui C poate fi extinsă la o bază a lui \overline{C} .

Pentru două puncte $\mathbf{b}, \mathbf{b}' \notin C$, spațiile liniare $C + t\mathbf{b}$ și $C + t\mathbf{b}'$ coincid dacă și numai dacă \mathbf{b} și \mathbf{b}' sunt în același subspațiu modulo C (Lema 15.1).

Din aceeași leamnă rezultă că sunt în total $\frac{2^m}{2^s}$ subspații modulo C . Unul este chiar C , iar celelalte conțin numai puncte dinafara lui C . Deci – înafară de C – există $2^{m-s} - 1$ spații distincte $C + t\mathbf{b}$ pentru $\mathbf{b} \notin C$.

II: Orice s - spațiu afin $L = \mathbf{a} + C$ ($\dim C = s$) este conținut în $2^{m-s} - 1$ ($s + 1$) - spații afine distincte de forma $\mathbf{a} + \overline{C}$.

Cu **I**, orice spațiu care conține L este de forma $\mathbf{b} + \overline{C}$. Deoarece L este de forma $\mathbf{a} + C$, rezultă că $\mathbf{a} \in \mathbf{b} + \overline{C}$, deci $\mathbf{a} - \mathbf{b} \in \overline{C}$, de unde rezultă (Lema 15.1) că punctele \mathbf{a}, \mathbf{b} sunt în același subspațiu modulo C .

III: Orice punct $\mathbf{b} \notin L = \mathbf{a} + C$ este într-un ($s + 1$) - spațiu afin care conține L , anume $\mathbf{a} + \overline{C}$ unde $\overline{C} = C + t(\mathbf{b} - \mathbf{a})$.

Într-adevăr, alegând $t = 1$ și $\mathbf{0} \in C$, avem $\mathbf{b} = \mathbf{a} + [\mathbf{0} + (\mathbf{b} - \mathbf{a})]$.

Pentru a verifica că \overline{C} are dimensiunea $s + 1$ este suficient de arătat că $\mathbf{b} - \mathbf{a} \notin C$; dacă prin absurd $\mathbf{b} - \mathbf{a} \in C$, atunci $\mathbf{a} + (\mathbf{b} - \mathbf{a}) = \mathbf{b} \in C$, contradicție.

În final, ar mai trebui arătat că acest ($s + 1$) - spațiu afin care îl conține pe \mathbf{b} este unic. Orice ($s + 1$) - spațiu afin care conține $\mathbf{a} + C$ are forma $\mathbf{a} + \overline{C}$ unde $\dim(\overline{C}) = s + 1$.

Dacă $\mathbf{b} \in \mathbf{a} + \overline{C}$, atunci $\mathbf{b} - \mathbf{a} \in \overline{C}$, deci \overline{C} conține spațiul linear $C + (\mathbf{b} - \mathbf{a})$. Cum ambele spații au aceeași dimensiune ($s + 1$), ele coincid. q.e.d.

Corolarul 6.1 Dacă numărul de erori din cuvântul recepționat \mathbf{y} este $t < 2^{m-r-1}$, atunci pentru fiecare s - spațiu afin L ($0 \leq s \leq r$) majoritatea ($s + 1$) - spațiilor afine care conțin L au aceeași paritate a erorilor ca L .

Demonstrație: Conform Teoremei 20.1, un s -spațiu afin L este inclus în $2^{m-r} - 1 > 2t$ ($s + 1$) - spații afine L' , fiecare L' fiind unic determinat de un punct din afara lui L . Să considerăm toate punctele $\mathbf{p}_i \notin L$ pentru care caracterul y_i este modificat. Lor le corespund cel mult t ($s + 1$) - spații afine L' . Toate celelalte spații L' rămase au proprietatea că nu conțin nici un punct \mathbf{p}_i din afara lui L pentru care y_i este greșit. Deci aceste spații au aceeași paritate de erori ca și L , iar numărul lor este cel puțin $(2^{m-r} - 1) - t > t$, deci majoritar. q.e.d.

Algoritm de decodificare pentru $\mathcal{RM}(r, m)$:

1. (*Inițializare*): La recepționarea unui cuvânt $\mathbf{y} \in Z_2^m$ se consideră toate ($r + 1$) - spațiile afine L . Un spațiu L se numește *impar* dacă $\mathbf{y}\mathbf{f}_L = 1$ (reamintim, \mathbf{f}_L este funcția caracteristică a spațiului afin L). În caz contrar L este *par*.

2. (*Inducție*): Pentru fiecare $s = r, r - 1, \dots, 0$ unde ($s + 1$) - spațiile afine au fost definite drept pare sau impare, se consideră toate s - spațiile afine. Un s - spațiu afin L este par dacă majoritatea ($s + 1$) - spațiilor afine care conțin pe L sunt pare; altfel L este impar.

3. (*Final*): Pentru $i = 0, 1, \dots, 2^m - 1$, se corectează y_i dacă și numai dacă 0 - spațiul afin $\{\mathbf{p}_i\}$ este impar.

Tabelul 6.1: Primul pas al decodificării lui 11101010

Plan	Paritate	Plan	Paritate
$\{\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_5, \mathbf{p}_7\}$	par	$\{\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_6\}$	impar
$\{\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_6, \mathbf{p}_7\}$	impar	$\{\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5\}$	par
$\{\mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_6, \mathbf{p}_7\}$	impar	$\{\mathbf{p}_0, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_7\}$	par
$\{\mathbf{p}_0, \mathbf{p}_2, \mathbf{p}_4, \mathbf{p}_6\}$	impar	$\{\mathbf{p}_0, \mathbf{p}_2, \mathbf{p}_5, \mathbf{p}_7\}$	par
$\{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_4, \mathbf{p}_5\}$	par	$\{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_6, \mathbf{p}_7\}$	impar
$\{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$	par	$\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_4, \mathbf{p}_7\}$	par
$\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_5, \mathbf{p}_6\}$	impar	$\{\mathbf{p}_0, \mathbf{p}_3, \mathbf{p}_5, \mathbf{p}_6\}$	impar

Tabelul 6.2: Al doilea pas al decodificării lui 11101010

Dreaptă	Paritate	Dreaptă	Paritate	Dreaptă	Paritate
$\{\mathbf{p}_0, \mathbf{p}_1\}$	par	$\{\mathbf{p}_1, \mathbf{p}_5\}$	par	$\{\mathbf{p}_3, \mathbf{p}_5\}$	par
$\{\mathbf{p}_0, \mathbf{p}_2\}$	par	$\{\mathbf{p}_1, \mathbf{p}_6\}$	par	$\{\mathbf{p}_3, \mathbf{p}_6\}$	impar
$\{\mathbf{p}_0, \mathbf{p}_3\}$	par	$\{\mathbf{p}_1, \mathbf{p}_7\}$	par	$\{\mathbf{p}_3, \mathbf{p}_7\}$	par
$\{\mathbf{p}_0, \mathbf{p}_4\}$	par	$\{\mathbf{p}_2, \mathbf{p}_3\}$	par	$\{\mathbf{p}_4, \mathbf{p}_5\}$	par
$\{\mathbf{p}_0, \mathbf{p}_5\}$	par	$\{\mathbf{p}_2, \mathbf{p}_4\}$	par	$\{\mathbf{p}_4, \mathbf{p}_6\}$	impar
$\{\mathbf{p}_0, \mathbf{p}_6\}$	impar	$\{\mathbf{p}_2, \mathbf{p}_5\}$	par	$\{\mathbf{p}_4, \mathbf{p}_7\}$	par
$\{\mathbf{p}_0, \mathbf{p}_7\}$	par	$\{\mathbf{p}_2, \mathbf{p}_6\}$	impar	$\{\mathbf{p}_5, \mathbf{p}_6\}$	impar
$\{\mathbf{p}_1, \mathbf{p}_2\}$	par	$\{\mathbf{p}_2, \mathbf{p}_7\}$	par	$\{\mathbf{p}_5, \mathbf{p}_7\}$	par
$\{\mathbf{p}_1, \mathbf{p}_3\}$	par	$\{\mathbf{p}_3, \mathbf{p}_4\}$	par	$\{\mathbf{p}_6, \mathbf{p}_7\}$	impar
$\{\mathbf{p}_1, \mathbf{p}_4\}$	par				

Exemplul 6.3 Să considerăm $\mathcal{RM}(1, 3)$ în care s-a recepționat cuvântul
 $\mathbf{y} = 11101010$

La primul pas trebuie să decidem care plane (2-spații afine) sunt pare și care sunt impare. De exemplu planul $L = \{\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_5, \mathbf{p}_7\}$ este par deoarece

$$\mathbf{y}\mathbf{f}_L = 11101010 \cdot 10101010 = 0$$

(vezi Prelegerea V, Paragraful 5.3.1). Se obține Tabelul 8.1.

La pasul următor trebuie să decidem paritatea fiecărei drepte (1-spațiu afin). Orice dreaptă este inclusă în $2^{3-1} - 1 = 3$ plane distincte.

De exemplu, linia $\{\mathbf{p}_0, \mathbf{p}_1\}$ este conținută în trei plane:

$$\{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_4, \mathbf{p}_5\} \text{ (par)}, \quad \{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\} \text{ (par)}, \quad \{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_6, \mathbf{p}_7\} \text{ (impar)}$$

deci, prin majoritate, $\{\mathbf{p}_0, \mathbf{p}_1\}$ este pară.

În mod similar se efectuează calculele pentru toate liniile, obținându-se Tabelul 8.2.

Se poate trece acum la corectarea erorilor. $\mathcal{RM}(1, 3)$ poate corecta maxim $2^{m-r-1} - 1 = 2^{3-1-1} - 1 = 1$ erori. $\{\mathbf{p}_0\}$ este conținut în șapte linii: șase pare și una impară, deci y_0 este corect etc. Singurul bit care trebuie corectat este y_6 deoarece $\{\mathbf{p}_6\}$ este în șapte linii, din care șase impare. Cuvântul - cod trimis a fost deci 10101010.

Algoritmul de decodificare se bazează pe următoarea observație:

Fie codul $\mathcal{RM}(r, m)$ și $\mathbf{y} \in Z_2^m$ un cuvânt recepționat.
 Un $(r + 1)$ - spațiu afin L este par $\iff \mathbf{y}\mathbf{f}_L = 0$.

Într-adevăr, dacă \mathbf{y} este cuvânt - cod, atunci $\mathbf{y}\mathbf{f}_L = 0$ (Prelegerea V, Exercițiul 5.11). Acum, dacă au fost perturbate un număr par de caractere din \mathbf{y} , corespunzătoare unor puncte din L , valoarea produsului scalar $\mathbf{y}\mathbf{f}_L$ nu se modifică (atenție ! se lucrează într-un corp de caracteristică 2). Pentru un număr impar de poziții perturbate, valoarea produsului scalar este 1.

6.3 Algoritm algebric de decodificare majoritară

În afară de algoritmul prezentat anterior, care se bazează pe reprezentarea geometrică a codurilor $R - M$, se poate da și o variantă algebrică de decodificare, folosind direct definiția decodificării majoritare.

În cele ce urmează să considerăm un cod $\mathcal{RM}(r, m)$ fixat. El are $k = 1 + C_m^1 + \dots + C_m^r$ simboluri de informație și lungimea $n = 2^m$.

În matricea $U_{m,n} = \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{m-1} \end{pmatrix}$ care conține pe coloane toate elementele spațiului liniar Z_2^m , vom nota cu \mathbf{u}_i ($0 \leq i \leq m-1$) coloanele care reprezintă baza (naturală) a lui Z_2^m .

Deci, orice coloană \mathbf{w}_j ($0 \leq j \leq n-1$) din Z_2^m se poate scrie

$$\mathbf{w}_j = \sum_{i=0}^{m-1} t_{ij} \mathbf{u}_i, \quad t_{ij} \in \{0, 1\}.$$

Orice cuvânt $\mathbf{v} \in Z_2^n$ poate fi considerat *indicatorul* unei mulțimi de cuvinte din Z_2^m : acele coloane din $U_{m,n}$ care corespund pozițiilor din \mathbf{v} unde se află valoarea 1.

Exemplul 6.4 Pentru orice $j = 0, \dots, n-1$, cuvântul $\mathbf{e}_j = 00 \dots 010 \dots 0$, cu 1 pe poziția j corespunde cuvântului de pe coloana j din $U_{m,n}$.

Cuvântul $\mathbf{1} = 11 \dots 1$ corespunde întregului spațiu Z_2^m .

Propoziția 6.1

$$\mathbf{e}_j = \prod_{i=0}^{m-1} [\mathbf{x}_i + (1 + t_{ij})\mathbf{1}], \quad \forall j, 0 \leq j \leq 2^m - 1.$$

Demonstrație: Să arătăm întâi că $\mathbf{x}_i + (1 + t_{ij})\mathbf{1}$ reprezintă indicatorul acelor coloane din U care au aceeași componentă i ca și coloana $\mathbf{w}_j = \sum_{i=0}^{m-1} t_{ij} \mathbf{u}_i$.

Într-adevăr:

1. Dacă $t_{ij} = 1$, atunci $\mathbf{x}_i + (1 + t_{ij})\mathbf{1} = \mathbf{x}_i + (1 + 1)\mathbf{1} = \mathbf{x}_i$ care este indicatorul mulțimii coloanelor cu 1 (ca și \mathbf{w}_j) pe linia i .

2. Dacă $t_{ij} = 0$, atunci $\mathbf{x}_i + (1 + t_{ij})\mathbf{1} = \mathbf{x}_i + \mathbf{1}$ care are componentele lui \mathbf{x}_i cu 0 și 1 permutate, fiind deci indicatorul mulțimii coloanelor care au 0 (ca și \mathbf{w}_j) pe linia i .

Cuvântul $\prod_{i=0}^{m-1} [\mathbf{x}_i + (1 + t_{ij})\mathbf{1}]$ reprezintă indicatorul mulțimii acelor coloane din matricea $U_{m,n}$ care au toate componentele egale cu cele ale coloanei \mathbf{w}_j . Cum coloanele matricii U sunt distincte, această mulțime este chiar $\{\mathbf{w}_j\}$, al cărei indicator este \mathbf{e}_j . q.e.d.

Fie mesajul de informație $a_1 a_2 \dots a_k \in Z_2^k$. Ele se codifică cu ajutorul $(2^m, k)$ - codului Reed - Muller în cuvântul - cod

$$\mathbf{f} = (f_0, f_1, \dots, f_{n-1}) = a_1 \mathbf{1} + a_2 \mathbf{x}_0 + \dots + a_{m+1} \mathbf{x}_{m-1} + a_{m+2} \mathbf{x}_0 \mathbf{x}_1 + \dots + a_k \mathbf{x}_{m-r} \mathbf{x}_{m-r+1} \dots \mathbf{x}_{m-1}.$$

Putem enunța acum rezultatul principal pe baza căruia se construiește algoritmul algebric de decodificare majoritară pentru codurile $R - M$.

Teorema 6.2 Pentru fiecare componentă a_s din mesajul de informație care se înmulțește - la codificare - cu un produs de forma $\mathbf{x}_{i_1} \mathbf{x}_{i_2} \dots \mathbf{x}_{i_r}$, există 2^{m-r} sume disjuncte (cu termeni diferiți) egale cu a_s , conținând fiecare 2^r componente ale cuvântului - cod corespunzător $\mathbf{f} = f_0 f_1 \dots f_{n-1}$.

Demonstrație: Ținând cont de Propoziția 6.1 și de faptul că $\mathbf{e}_0, \mathbf{e}_1, \dots, \mathbf{e}_{n-1}$ constituie de asemenea o bază pentru Z_2^n , putem scrie:

$$\mathbf{f} = (f_0, f_1, \dots, f_{n-1}) = \sum_{j=0}^{n-1} f_j \mathbf{e}_j = \sum_{j=0}^{n-1} f_j \prod_{i=0}^{m-1} [\mathbf{x}_i + (1 + t_{ij})\mathbf{1}].$$

Prin urmare, ca un produs de forma $\mathbf{x}_{i_1} \dots \mathbf{x}_{i_t}$ să fie în această sumă, trebuie ca $t_{ij} = 0$ pentru $i \notin \{i_1, i_2, \dots, i_t\}$. Dacă notăm

$$\begin{aligned} C(i_1, \dots, i_j) &= \{p \mid p = \sum_{i=0}^{m-1} t_{ip} 2^i, t_{ip} = 0 \text{ dacă } i \notin \{i_1, \dots, i_j\}\} = \\ &= \{p \mid p = t_{i_1 p} 2^{i_1} + t_{i_2 p} 2^{i_2} + \dots + t_{i_j p} 2^{i_j}, t_{i_s p} \in \{0, 1\}, s = 1, 2, \dots, j\}, \end{aligned}$$

vom avea

$$\mathbf{f} = (f_0, f_1, \dots, f_{n-1}) = \sum_{t=1}^r \sum_{i_1 < i_2 < \dots < i_t} \left(\sum_{j \in C(i_1, \dots, i_t)} f_j \right) \mathbf{x}_{i_1} \mathbf{x}_{i_2} \dots \mathbf{x}_{i_t}$$

Pe de-altă parte,

$$\mathbf{f} = (f_0, \dots, f_{n-1}) = a_1 \mathbf{1} + a_2 \mathbf{x}_0 + \dots + a_{m+1} \mathbf{x}_{m-1} + a_{m+2} \mathbf{x}_0 \mathbf{x}_1 + \dots + a_k \mathbf{x}_{m-r} \dots \mathbf{x}_{m-1}.$$

Prin identificare, rezultă că elementul a_s , coeficient al produsului $\mathbf{x}_{i_1} \mathbf{x}_{i_2} \dots \mathbf{x}_{i_r}$ este

$$a_s = \sum_{j \in C(i_1, \dots, i_r)} f_j \quad (1)$$

Fie $z \notin \{i_1, \dots, i_r\}$. Deoarece baza codului $\mathcal{RM}(r, m)$ nu conține monoame de forma $\mathbf{x}_{i_1} \dots \mathbf{x}_{i_r} \mathbf{x}_{i_r+1}$, vom obține prin codificare

$$\sum_{j \in C(i_1, \dots, i_r, z)} f_j = 0. \quad (2)$$

Însă,

$$C(i_1, \dots, i_r, z) = C(i_1, \dots, i_r) \cup [C(i_1, \dots, i_r) + 2^z],$$

unde s-a notat $C + \alpha = \{x + \alpha \mid x \in C\}$. Într-adevăr, putem scrie

$$\begin{aligned} C(i_1, \dots, i_r, z) &= \{j \mid j = t_{i_1 j} 2^{i_1} + \dots + t_{i_r j} 2^{i_r} + t_{z j} 2^z\} = \\ &= \{j \mid j = t_{i_1 j} 2^{i_1} + \dots + t_{i_r j} 2^{i_r}\} \cup \{j \mid j = t_{i_1 j} 2^{i_1} + \dots + t_{i_r j} 2^{i_r} + 2^z\} = \\ &= C(i_1, \dots, i_r) \cup [C(i_1, \dots, i_r) + 2^z], \end{aligned}$$

cele două mulțimi ale reuniunii fiind disjuncte.

Deci relația (2) se scrie $\sum_{j \in C(i_1, \dots, i_r)} f_j + \sum_{j \in C(i_1, \dots, i_r) + 2^z} f_j = 0$ și – ținând cont de (1), avem

$$a_s + \sum_{j \in C(i_1, \dots, i_r) + 2^z} f_j = 0 \quad (3)$$

Din (1) și (3) rezultă următoarea afirmație: dacă $m = r + 1$, pentru orice simbol de informație a_s înmulțit la codificare cu un produs de forma $\mathbf{x}_{i_1} \dots \mathbf{x}_{i_r}$ există $2^{m-r} = 2$ sume disjuncte ((1) și (3)), fiecare de câte 2^r componente ale lui \mathbf{f} .

Aceste două relații se numesc *relații de determinare* a componentei a_s .

Fie acum $z_1, z_2 \notin \{i_1, \dots, i_r\}$. Deoarece baza codului nu are produse de forma $\mathbf{x}_{i_1} \dots \mathbf{x}_{i_r} \mathbf{x}_{i_r+1} \mathbf{x}_{i_r+2}$, vom avea

$$\sum_{j \in C(i_1, \dots, i_r, z_1, z_2)} f_j = 0. \quad (4)$$

Dar

$$C(i_1, \dots, i_r, z_1, z_2) = C(i_1, \dots, i_r) \cup [C(i_1, \dots, i_r) + 2^{z_1}] \cup [C(i_1, \dots, i_r) + 2^{z_2}] \cup [C(i_1, \dots, i_r) + 2^{z_1} + 2^{z_2}]$$

Deci (4) se descompune în

$$\sum_{j \in C(i_1, \dots, i_r)} f_j + \sum_{j \in C(i_1, \dots, i_r) + 2^{z_1}} f_j + \sum_{j \in C(i_1, \dots, i_r) + 2^{z_2}} f_j + \sum_{j \in C(i_1, \dots, i_r) + 2^{z_1} + 2^{z_2}} f_j = 0$$

ceea ce – ținând cont de (1) și (3) devine $a_s + a_s + a_s + \sum_{j \in C(i_1, \dots, i_r) + 2^{z_1} + 2^{z_2}} f_j = 0$.

deci

$$a_s = \sum_{j \in C(i_1, \dots, i_r) + 2^{z_1} + 2^{z_2}} f_j \quad (5)$$

Am arătat astfel că, dacă $m = r + 2$, atunci există $2^{m-r} = 2^2 = 4$ sume ((1), de două ori (3) pentru z_1 respectiv z_2 , și (5)) care dau pe a_s , fiecare sumă conținând 2^r componente ale lui \mathbf{f} . Deci, pentru a_s există în acest caz patru relații de determinare.

Teorema rezultă în continuare printr-un procedeu de inducție finită. q.e.d.

Să vedem cum se folosește Teorema 11.1 la decodificarea majoritară a codurilor Reed - Muller:

După cum s-a văzut, pentru fiecare componentă a_s corespunzătoare unui produs de forma $\mathbf{x}_{i_1} \dots \mathbf{x}_{i_r}$ există 2^{m-r} sume disjuncte, conținând fiecare 2^r componente ale lui \mathbf{f} care – în absența erorilor – vor fi toate egale cu a_s .

Deoarece sumele sunt disjuncte (deci termenii lor sunt elemente diferite ale cuvântului - cod), o eroare singulară va afecta o singură relație de determinare a lui a_s . Prin urmare, dacă în transmiterea cuvântului - cod \mathbf{f} apar cel mult $2^{m-r-1} - 1$ erori independente, putem determina – prin majoritate – pe a_s , identificându-l cu valoarea a *jumătate plus unu* din relațiile de determinare corespunzătoare.

După ce se determină în acest mod coeficienții tuturor produselor $\mathbf{x}_{i_1} \dots \mathbf{x}_{i_r}$, se scade din vectorul recepționat combinația liniară

$$a_p \mathbf{x}_0 \dots \mathbf{x}_{r-1} + \dots + a_k \mathbf{x}_{m-r} \dots \mathbf{x}_{m-1}.$$

În acest fel problema se reduce în continuare la decodificarea într-un cod de ordin $r - 1$ și lungime 2^m ; procedeu este similar, fiind determinate aici componentele de informație care se codifică prin înmulțire cu produse de forma $\mathbf{x}_{j_1} \dots \mathbf{x}_{j_{r-1}}$ ($j_1 < j_2 < \dots < j_{r-1}$).

Exemplul 6.5 Fie codul $\mathcal{RM}(2, 4)$. El are lungimea $n = 16$, ordinul $r = 2$, $k = 1 + C_4^1 + C_4^2 = 11$ simboluri de informație și $n - k = 5$ simboluri de control. Deci el este un $(16, 11)$ - cod liniar, capabil să corecteze cel mult o eroare (pentru că $2^{4-2-1} - 1 = 1$).

Orice mesaj de informație $(a_1, a_2, \dots, a_{11}) \in Z_2^{11}$ se codifică în cuvântul

$$\mathbf{f} = (f_0, \dots, f_{15}) = a_1 \mathbf{1} + a_2 \mathbf{x}_0 + a_3 \mathbf{x}_1 + a_4 \mathbf{x}_2 + a_5 \mathbf{x}_3 + a_6 \mathbf{x}_0 \mathbf{x}_1 + a_7 \mathbf{x}_0 \mathbf{x}_2 + a_8 \mathbf{x}_0 \mathbf{x}_3 + a_9 \mathbf{x}_1 \mathbf{x}_2 + a_{10} \mathbf{x}_1 \mathbf{x}_3 + a_{11} \mathbf{x}_2 \mathbf{x}_3.$$

Să stabilim de exemplu relațiile de determinare pentru a_9 , care se codifică prin înmulțire cu $\mathbf{x}_1 \mathbf{x}_2$. Trebuie calculate mulțimile:

$$\begin{aligned} C(1, 2) &= \{j \mid j = t_{1j} 2^1 + t_{2j} 2^2, \quad t_{ij} \in \{0, 1\}\} = \{0, 2, 4, 6\} \\ C(1, 2) + 2^0 &= \{1, 3, 5, 7\} \\ C(1, 2) + 2^3 &= \{8, 10, 12, 14\} \\ C(1, 2) + 2^0 + 2^3 &= \{9, 11, 13, 15\}. \end{aligned}$$

Deci relațiile de determinare corespunzătoare lui a_9 devin

$$\begin{aligned} a_9 &= f_0 + f_2 + f_4 + f_6 & a_9 &= f_1 + f_3 + f_5 + f_7 \\ a_9 &= f_8 + f_{10} + f_{12} + f_{14} & a_9 &= f_9 + f_{11} + f_{13} + f_{15} \end{aligned}$$

Dacă intervine o eroare în timpul transmiterii cuvântului - cod \mathbf{f} , va fi afectată o singură relație de determinare, celelalte trei relații dând valoarea corectă a lui a_9 .

După ce s-a determinat valorile $a_6, a_7, a_8, a_9, a_{10}, a_{11}$, se trece la etapa a doua a decodificării, scăzând din cuvântul recepționat combinația

$$a_6 \mathbf{x}_0 \mathbf{x}_1 + a_7 \mathbf{x}_0 \mathbf{x}_2 + a_8 \mathbf{x}_0 \mathbf{x}_3 + a_9 \mathbf{x}_1 \mathbf{x}_2 + a_{10} \mathbf{x}_1 \mathbf{x}_3 + a_{11} \mathbf{x}_2 \mathbf{x}_3$$

și aplicând cuvântului obținut procedeul de corectare a erorilor pentru codul $\mathcal{RM}(1, 4)$.

În final, a_1 se decodifică în codul $\mathcal{RM}(0, 4)$, ceea ce înseamnă că el este egal cu valoarea majoritară (0 sau (1)) luată de elementele f_0, f_1, \dots, f_{15} .

6.4 Decodificarea codurilor $\mathcal{RM}(1, m)$

Pentru cazul $r = 1$ se poate da și un alt algoritm de decodificare, extrem de eficient. El folosește transformarea Hadamard pentru aflarea celui mai apropiat cuvânt - cod (în distanță Hamming).

Vom da inițial câteva noțiuni ajutătoare.

Definiția 6.3 Fie $A_{m,n}, B_{p,q}$ două matrici. Se definește produsul Kronecker $A \times B$ ca fiind matricea $C_{mp,nq} = [a_{ij}B]$.

Acest produs este evident asociativ și necomutativ; el va fi utilizat pe larg și pentru alte construcții.

Exemplul 6.6 Fie $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, $I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Atunci

$$I_2 \times H = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}, \quad H \times I_2 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}.$$

Se consideră șirul de matrici definit

$$H_m^i = I_{2^{m-i}} \times H \times I_{2^{i-1}}, \quad i = 1, 2, \dots$$

unde H este matricea (Hadamard) definită în Exemplul 6.6.

Exemplul 6.7 Fie $m = 2$. Atunci

$$H_2^1 = I_2 \times H \times I_1 = I_2 \times H, \quad H_2^2 = I_1 \times H \times I_2 = H \times I_2.$$

Exemplul 6.8 Fie $m = 3$. Atunci:

$$H_3^1 = I_4 \times H \times I_1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix},$$

$$H_3^2 = I_2 \times H \times I_2 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix},$$

$$H_3^3 = I_1 \times H \times I_4 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix}.$$

Modalitatea recursivă de construcție a codurilor $\mathcal{RM}(r, m)$ (Prelegerea V, secțiunea 5.2) sugerează existența unui algoritm similar de decodificare. Acest algoritm există numai pentru $\mathcal{RM}(1, m)$ și îl prezentăm fără a demonstra corectitudinea lui.

Algoritm de decodificare a codurilor $\mathcal{RM}(1, m)$:

Fie $\mathbf{y} \in Z_2^n$ ($n = 2^m$) cuvântul recepționat și $G(1, m)$ matricea generatoare a codului.

1. Se înlocuiește 0 cu -1 în \mathbf{y} ; fie \mathbf{y}' noul cuvânt;
2. Se calculează $\mathbf{y}_1 = \mathbf{y}'H_m^1$, $\mathbf{y}_i = \mathbf{y}_{i-1}H_m^i$ ($2 \leq i \leq m$);
3. Se determină poziția j a celei mai mari componente (în valoare absolută) a lui \mathbf{y}_m .

Fie $v(j) \in Z_2^n$ reprezentarea binară a lui j pe m biți (începând cu biții cei mai semnificativi). Cuvântul decodificat este:

$(1, v(j))$ dacă a j - a componentă a lui \mathbf{y}_m este pozitivă,

$(0, v(j))$ dacă a j - a componentă a lui \mathbf{y}_m este negativă.

Exemplul 6.9 Fie $m = 3$ și $G(1, 3)$ matricea generatoare a codului $\mathcal{RM}(1, 3)$. Să presupunem că s-a recepționat cuvântul $\mathbf{y} = 10101011$.

Primul pas al algoritmului transformă acest cuvânt în $\mathbf{y}' = (1, -1, 1, -1, 1, -1, 1, 1)$.

Se calculează apoi:

$$\begin{aligned} \mathbf{y}_1 &= \mathbf{y}'H_3^1 = (0, 2, 0, 2, 0, 2, 2, 0) \\ \mathbf{y}_2 &= \mathbf{y}_1H_3^2 = (0, 4, 0, 0, 2, 2, -2, 2) \\ \mathbf{y}_3 &= \mathbf{y}_2H_3^3 = (2, 6, -2, 2, -2, 2, 2, -2). \end{aligned}$$

Cea mai mare componentă a lui \mathbf{y}_3 este 6 pe poziția 1. Cum $v(1) = 100$ și $6 > 0$, rezultă că a fost codificat mesajul de informație 1100.

Dacă s-a recepționat cuvântul $\mathbf{y} = 10001111$, atunci $\mathbf{y}' = (1, -1, -1, -1, 1, 1, 1, 1)$ și

$$\begin{aligned} \mathbf{y}_1 &= \mathbf{y}'H_3^1 = (0, 2, -2, 0, 2, 0, 2, 0), \\ \mathbf{y}_2 &= \mathbf{y}_1H_3^2 = (-2, 2, 2, 2, 4, 0, 0, 0), \\ \mathbf{y}_3 &= \mathbf{y}_2H_3^3 = (2, 2, 2, 2, -6, 2, 2, 2). \end{aligned}$$

Cea mai mare componentă a lui \mathbf{y}_3 (în valoare absolută) este -6 aflată pe poziția 4. Deoarece $v(4) = 001$ și $-6 < 0$, mesajul transmis este 0001.

6.5 Exerciții

6.1 În codul Hamming $(15, 11)$ să se decodifice mesajele
111100000000000, 011101000111001.

6.2 Demonstrați Lema 15.1.

6.3 Să se decodifice cuvântul 01111100 în $\mathcal{RM}(1, 3)$. Verificați corectitudinea decodificării.

6.4 Să se decodifice 0111100 în $\mathcal{RM}(2, 3)$.

6.5 Să se scrie toate relațiile de determinare din codul $\mathcal{RM}(2, 4)$.

6.6 Aceeași problemă pentru codul $\mathcal{RM}(1, 3)$.

6.7 Să se decodifice 111111011111111 în $\mathcal{RM}(2, 4)$.

6.8 Arătați că orice hiperplan L în geometria euclidiană peste Z_2^m are drept complement $Z_2^m \setminus L$ tot un hiperplan.

6.9 În codul $\mathcal{RM}(1, 3)$, să se decodifice cuvintele
01011110, 01100111, 00010100, 11001110.

6.10 Să se calculeze H_4^i pentru $i = 1, 2, 3, 4$.

6.11 În codul $\mathcal{RM}(1, 4)$, să se decodifice cuvintele:
1011011001101001, 1111000001011111.

Capitolul 7

Alte clase de coduri elementare

Înafara codurilor Hamming și Reed - Muller se pot construi și alte clase de coduri cu proprietăți remarcabile. Vom prezenta numai câteva astfel de coduri – nu toate liniare – punând în evidență la fiecare clasă caracteristicile sale specifice.

7.1 Codurile MacDonal

Fie $A_{n,k}$ un cod linear peste Z_2 și G matricea sa generatoare.

Vom nota cu $M_{k,2^k-1}$ o matrice care conține toate coloanele nenule posibile de k elemente, care se pot construi peste Z_2 . Ordinea acestor coloane este arbitrară dar fixată. Coloana i din matricea M se va numi *coloană de tip i* . Pentru ușurința notației se admite aserțiunea că tipul de coloană i este reprezentarea în binar a numărului zecimal i .

Deoarece codul $A_{n,k}$ nu este influențat de operațiile elementare efectuate asupra coloanelor din matricea generatoare (în particular permutarea de coloane), este suficient să definim acest cod indicând doar numărul coloanelor de fiecare tip care intră în componența matricii generatoare.

Se obține astfel *reprezentarea modulară a codului $A_{n,k}$* :

$$N = (n_1, n_2, \dots, n_{2^k-1})$$

unde n_i este numărul coloanelor de tip i care apar (sau nu) în matricea G ,

$$\sum_{i=1}^{2^k-1} n_i = n.$$

Exemplul 7.1 Să luăm $k = 2$, $n = 3$ și matricea generatoare $G = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$.

Avem matricea $M = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$ (coloanele sunt reprezentările binare ale numerelor 1, 2, 3).

Reprezentarea modulară a codului generat de matricea G este atunci $N = (2, 0, 1)$ (pentru că în G există două coloane de tipul 1, nici una de tipul 2 și una de tipul 3).

Să introducem matricea

$$K_{2^k-1,n} = M^T G$$

K are drept linii toate cuvintele nenule ale codului $A_{n,k}$.

Fie de asemenea matricea pătrată simetrică:

$$C_{2^k-1,2^k-1} = M^T M$$

Astfel, pentru Exemplitul 19.4,
$$K = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

Teorema 7.1 *O listă a ponderilor cuvintelor codului $A_{n,k}$ se obține înmulțind (în \mathcal{R}) matricea de reprezentare modulară cu matricea C :*

$$W_{2^k-1} = NC \quad (1)$$

Demonstrație: Să scriem matricea M sub forma unei matrici linie

$$M = (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{2^k-1}),$$

unde \mathbf{v}_i ($1 \leq i \leq 2^k - 1$) este coloana de tip i din matricea M .

Matricea generatoare corespunzătoare reprezentării modulare $N = (n_1, n_2, \dots, n_{2^k-1})$ este

$$G = (\underbrace{\mathbf{v}_1, \dots, \mathbf{v}_1}_{n_1}, \underbrace{\mathbf{v}_2, \dots, \mathbf{v}_2}_{n_2}, \dots, \underbrace{\mathbf{v}_{2^k-1}, \dots, \mathbf{v}_{2^k-1}}_{n_{2^k-1}})$$

unde $n_1 + n_2 + \dots + n_{2^k-1} = n$.

Se obține deci

$$C = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_{2^k-1} \end{pmatrix} (\mathbf{v}_1, \dots, \mathbf{v}_{2^k-1}) = \begin{pmatrix} \mathbf{v}_1 \mathbf{v}_1 & \mathbf{v}_1 \mathbf{v}_2 & \dots & \mathbf{v}_1 \mathbf{v}_{2^k-1} \\ \mathbf{v}_2 \mathbf{v}_1 & \mathbf{v}_2 \mathbf{v}_2 & \dots & \mathbf{v}_2 \mathbf{v}_{2^k-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{v}_{2^k-1} \mathbf{v}_1 & \mathbf{v}_{2^k-1} \mathbf{v}_2 & \dots & \mathbf{v}_{2^k-1} \mathbf{v}_{2^k-1} \end{pmatrix},$$

unde produsele scalare sunt efectuate în Z_2 .

Acum, $W = (w_1, w_2, \dots, w_{2^k-1}) = (n_1, n_2, \dots, n_{2^k-1})C$, înmulțire efectuată în \mathcal{R} .

Prin identificare rezultă

$$w_i = n_1 \mathbf{v}_1 \mathbf{v}_i + n_2 \mathbf{v}_2 \mathbf{v}_i + \dots + n_{2^k-1} \mathbf{v}_{2^k-1} \mathbf{v}_i.$$

Deoarece toate produsele scalare $\mathbf{v}_j \mathbf{v}_i$ sunt 0 sau 1, w_i este un număr natural. El este chiar ponderea cuvântului cod care ocupă linia i în matricea K .

Într-adevăr, avem

$$K_{2^k-1,n} = \begin{pmatrix} \mathbf{v}_1 \mathbf{v}_1 & \mathbf{v}_1 \mathbf{v}_1 & \dots & \mathbf{v}_1 \mathbf{v}_{2^k-1} \\ \mathbf{v}_2 \mathbf{v}_1 & \mathbf{v}_2 \mathbf{v}_1 & \dots & \mathbf{v}_2 \mathbf{v}_{2^k-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{v}_{2^k-1} \mathbf{v}_1 & \mathbf{v}_{2^k-1} \mathbf{v}_1 & \dots & \mathbf{v}_{2^k-1} \mathbf{v}_{2^k-1} \end{pmatrix},$$

produsele scalare fiind în Z_2 .

Deci ponderea cuvântului situat pe linia i în matricea K este:

$$\underbrace{\mathbf{v}_i \mathbf{v}_1 + \dots + \mathbf{v}_i \mathbf{v}_1}_{n_1} + \dots + \underbrace{\mathbf{v}_i \mathbf{v}_{2^k-1} + \dots + \mathbf{v}_i \mathbf{v}_{2^k-1}}_{n_{2^k-1}} = n_1 \mathbf{v}_i \mathbf{v}_1 + \dots + n_{2^k-1} \mathbf{v}_i \mathbf{v}_{2^k-1}. \quad \text{q.e.d.}$$

Ne punem problema de a determina un cod atunci când se dă vectorul ponderilor cuvintelor sale (deci și ponderea minimă, adică distanța Hamming).

Formal, din (1) se obține

$$N = WC^{-1}$$

înmulțire efectuată în \mathcal{R} .

Formula este adevărată numai dacă demonstrăm că matricea C este inversabilă.

În acest sens avem:

Teorema 7.2 *Matricea C este nesingulară. Inversa ei se calculează înlocuind în C pe 0 cu -1 și împărțind toate elementele cu 2^{k-1} .*

Demonstrație: I: Fiecare linie (deci și coloană) din C conține 1 pe 2^{k-1} poziții. Două linii (coloane) distincte din C au în comun 1 în 2^{k-2} poziții.

Într-adevăr, liniile lui C – la care se adaugă linia nulă – formează un spațiu liniar cu 2^k cuvinte binare. Să considerăm submulțimea liniilor care au 1 în componentele i și j . Ele formează un subspațiu liniar cu $\frac{2^k}{2^2} = 2^{k-2}$ elemente deoarece, factorizând cu acest spațiu se obțin patru clase de resturi, corespunzătoare componentelor (i, j) de forma $(0, 0), (0, 1), (1, 0), (1, 1)$.

Pentru a arăta că 1 apare pe fiecare linie din C de 2^{k-1} ori se raționează similar.

II: În această situație, putem scrie

$$C^2 = \begin{pmatrix} 2^{k-1} & 2^{k-2} & \dots & 2^{k-2} \\ 2^{k-2} & 2^{k-1} & \dots & 2^{k-2} \\ & & \vdots & \\ 2^{k-2} & 2^{k-2} & \dots & 2^{k-1} \end{pmatrix} = 2^{k-2}(I + J)$$

unde I este matricea unitate iar J este matricea pătrată cu toate elementele egale cu 1, ambele de ordin 2^k .

Se mai observă că $CJ = 2^{k-1}J$. Deci

$$I = \frac{1}{2^{k-1}}(2C^2 - 2^{k-1}J) = \frac{1}{2^{k-1}}(2C^2 - CJ) = C \frac{2C - J}{2^{k-1}} = CC^{-1}.$$

De aici rezultă că C este inversabilă și $C^{-1} = \frac{1}{2^{k-1}}(2C - J)$. q.e.d.

Exemplul 7.2 *Fie $k = 3, n = 5$. Vom avea:*

$$M = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}, \quad C = M^T M = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix},$$

$$C^{-1} = \frac{1}{4} \begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & -1 & 1 & -1 \\ -1 & 1 & 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{pmatrix}.$$

Dacă luăm codul binar cu matricea generatoare $G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$, el are reprezentarea modulară $N = (1, 1, 0, 1, 1, 1, 0)$. Vectorul ponderilor cuvintelor - cod este $W = (2, 2, 4, 3, 3, 3, 3)^1$. Într-adevăr, cuvintele - cod sunt liniile matricii

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

Să determinăm acum reprezentarea modulară a unui cod linear binar în care toate cuvintele - cod au aceeași pondere w (și deci distanța minimă va fi $d = w$, iar numărătorul de ponderi $P(X) = 1 + (2^k - 1)X^w$). Vom avea prin ipoteză $W = \underbrace{(w, w, \dots, w)}_{2^{k-1}}$.

După cum am remarcat în demonstrația Teoremei 20.2, 1 apare pe fiecare linie din C în 2^{k-1} poziții. Rezultă (Teorema 20.2) că în C^{-1} , 1 va apare pe fiecare linie (coloană) tot în 2^{k-1} poziții, în rest fiind -1 . Deci

$$N = WC^{-1} = \underbrace{(w2^{-(k-1)}, \dots, w2^{-(k-1)})}_{2^{k-1}}$$

Problema enunțată are soluție dacă w se divide cu 2^{k-1} ; în acest caz, reprezentarea modulară este de forma $N = \underbrace{(r, r, \dots, r)}_{2^{k-1}}$, adică fiecare tip de coloană apare în matricea generatoare de $r = w \cdot 2^{-(k-1)}$ ori.

Plecând de la aceste considerente, McDonald a introdus o clasă de coduri pentru care reprezentarea modulară este de forma $0^i 1^j$. Mai exact,

n	d	N
$2^k - 1$	2^{k-1}	$(1, 1, 1, \dots, 1)$
$2^k - 2$	$2^{k-1} - 1$	$(0, 1, 1, \dots, 1)$
$2^k - 3$	$2^{k-1} - 2$	$(0, 0, 1, \dots, 1)$
\dots	\dots	\dots
$2^k - 2^u$	$2^{k-1} - 2^{u-1}$	$\underbrace{(0, \dots, 0)}_{2^{u-1}}, \underbrace{(1, \dots, 1)}_{2^k - 2^u}$

Aceste coduri au proprietatea că prezintă o distanță minimă foarte apropiată de marginea Plotkin.

Să luăm de exemplu al treilea cod din tabelul de sus. Reamintim, marginea Plotkin este

$$d \leq \frac{nq^{k-1}(q-1)}{q^k - 1}.$$

În cazul codului McDonald avem $q = 2$, $n = 2^k - 3$, $d = 2^{k-1} - 2$, deci

$$\frac{nq^{k-1}(q-1)}{q^k - 1} - d = \frac{(2^k - 3)2^{k-1}}{2^k - 1} - (2^{k-1} - 2) = \frac{2^k - 2}{2^k - 1} < 1.$$

¹De aici se poate obține și numărătorul de ponderi: $P(X) = 1 + 2X^2 + 4X^3 + X^4$.

7.2 Coduri derivate din matricile Hadamard

În Capitolul anterior am folosit o matrice Hadamard pentru a construi un algoritm de decodificare al codurilor $\mathcal{RM}(1, m)$. De fapt matricile Hadamard pot fi utilizate în mod direct la construcția de coduri.

Definiția 7.1 Se numește matrice Hadamard \mathcal{H}_n o matrice pătrată de ordinul n cu elemente 1 și -1 , ale cărei linii sunt ortogonale două câte două.

Teorema 7.3 Fiind dată matricea Hadamard \mathcal{H}_n , (n par) se poate construi un cod de distanță minimă $d = \frac{n}{2}$, format din $2n$ cuvinte de lungime n .

Demonstrație: În matricea \mathcal{H}_n vom înlocui 1 cu 0 și -1 cu 1 și vom nota $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ vectorii care formează liniile noii matrici.

Codul căutat este

$$C = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n, -\mathbf{v}_1, -\mathbf{v}_2, \dots, -\mathbf{v}_n\}.$$

Acest cod (care nu este neapărat liniar) are $2n$ cuvinte de lungime n .

Mai trebuie arătat că distanța sa minimă este $n/2$.

Deoarece \mathbf{v} și $-\mathbf{v}$ diferă în toate componentele, distanța Hamming dintre ele este n . Pe de altă parte, din proprietatea matricilor Hadamard rezultă

$$\mathbf{v}_i(\pm\mathbf{v}_j) = 0, \quad \forall i, j \in \{1, \dots, n\}, \quad i \neq j.$$

Pentru ca această egalitate să fie adevărată, este necesar ca \mathbf{v}_i și \mathbf{v}_j să coincidă în jumătate din componente și să difere în cealaltă jumătate (atenție, calculele se fac în \mathcal{R} , nu în $Z_2!$). Rezultă că distanța Hamming dintre cele două cuvinte este $\frac{n}{2}$. Deci $d = \frac{n}{2}$. q.e.d.

Pentru $n = 32$ un astfel de cod a fost folosit de programul spațial Mariner.

Teorema 7.4 Dacă \mathcal{H}_n este matrice Hadamard, atunci $\mathcal{H}_{2n} = \begin{pmatrix} \mathcal{H}_n & \mathcal{H}_n \\ \mathcal{H}_n & -\mathcal{H}_n \end{pmatrix}$ este matrice Hadamard.

Demonstrație: Evident, \mathcal{H}_{2n} este o matrice $2n \times 2n$ cu elemente ± 1 . Mai trebuie arătat că liniile sale sunt ortogonale două câte două.

Avem: $[\mathbf{v}_i, \mathbf{v}_i][\mathbf{v}_i, -\mathbf{v}_i] = \mathbf{v}_i\mathbf{v}_i - \mathbf{v}_i\mathbf{v}_i = n - n = 0$.

De asemenea, pentru $i \neq j$, $[\mathbf{v}_i, \mathbf{v}_i][\mathbf{v}_j, \pm\mathbf{v}_j] = [\mathbf{v}_i, \mathbf{v}_j] \pm [\mathbf{v}_i, \mathbf{v}_j] = 0 \pm 0 = 0$. q.e.d.

Pe baza acestei teoreme, putem construi coduri Hadamard a căror distanță să fie oricât de mare. Este suficient să găsim o matrice Hadamard inițială, pe care (folosind Teorema 20.3) să o "dilatăm" ulterior cât este nevoie.

O astfel de matrice Hadamard inițială a fost dată în Capitolul 6:

$$\mathcal{H}_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Pentru construcția altor matrici (și coduri) Hadamard mai generale au fost elaborate diverse tehnici. Una din cele mai cunoscute este cea a lui Paley și Hall.

Definiția 7.2 O matrice pătrată M de ordin n cu 0 pe diagonala principală și ± 1 înafara ei, astfel ca $MM^T = (n-1)I_n$ se numește matrice de conjunctură (conference matrix).

Exemplul 7.3 Matricile $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $\begin{pmatrix} 0 & 1 & 1 & 1 \\ -1 & 0 & -1 & 1 \\ -1 & 1 & 0 & -1 \\ -1 & -1 & 1 & 0 \end{pmatrix}$

sunt matrici de conjunctură.

Fie q un număr prim impar. În corpul Z_q definim funcția reziduu χ astfel:

$$\chi(x) = \begin{cases} 0 & \text{dacă } x = 0 \\ 1 & \text{dacă } x \text{ este un pătrat nenul} \\ -1 & \text{în rest} \end{cases}$$

Teorema 7.5 Matricea Paley $S_q = (s_{ij})$ de ordin q definită prin $s_{ij} = \chi((q+i-j) \bmod q)$, ($i, j \in Z_q$) are următoarele proprietăți:

1. $S_q J_q = J_q S_q = \mathbf{0}$;
2. $S_q S_q^T = qI_q - J_q$;
3. $S_q^T = (-1)^{\frac{q-1}{2}} S_q$.

Toate calculele sunt în Z . S-a notat cu I_n matricea unitate de ordin n și cu J_n matricea pătrată de ordin n cu toate elementele egale cu 1.

Demonstrație: Este lăsată ca exercițiu.

Exemplul 7.4 Pentru $q = 3$, matricea Paley este $S_3 = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$,

$$\text{iar pentru } q = 5, \quad S_5 = \begin{pmatrix} 0 & 1 & -1 & -1 & 1 \\ 1 & 0 & 1 & -1 & -1 \\ -1 & 1 & 0 & 1 & -1 \\ -1 & -1 & 1 & 0 & 1 \\ 1 & -1 & -1 & 1 & 0 \end{pmatrix}.$$

De remarcat că $S_3^T = -S_3$, $S_5^T = S_5$ (ceea ce verifică și Teorema 7.5, punctul 3).

Teorema 7.6 Fie q un număr prim impar astfel încât $q \equiv 3 \pmod{4}$ (număr Blum). Atunci există o matrice Hadamard de ordin $q+1$.

Demonstrație: Din ipoteză și Teorema 7.5, se poate defini matricea Paley S_q . Construim matricea M de ordin $q+1$ astfel:

$$M = \begin{pmatrix} 0 & 1 & \dots & 1 \\ -1 & & & \\ -1 & & S_q & \\ -1 & & & \end{pmatrix}$$

Se verifică faptul că M este o matrice de conjunctură cu proprietatea $M^T = -M$ ($q \equiv 3 \pmod{4}$ conduce la relația $S^T = -S$).

Fie $H = I_{q+1} + M$. Avem $HH^T = (I_{q+1} + M)(I_{q+1} + M)^T = (I_{q+1} + M)(I_{q+1} + M^T) = I_{q+1} + M + M^T + MM^T = I_{q+1} + qI_{q+1} = (q+1)I_{q+1}$, deci orice două linii distincte ale lui H sunt ortogonale.

Rezultă că H este matrice Hadamard de ordin $q+1$. q.e.d.

Plecând de la matricile Paley se pot construi și alte coduri Hadamard (modificând puțin Teorema 7.3).

Fie S_n o matrice Paley de ordin n (conform Teoremei 7.5). Un cod Hadamard de ordin n poate fi definit cu $\mathbf{0}, \mathbf{1}$ și cu liniile matricilor $\frac{1}{2}(S_n + I_n + J_n)$, $\frac{1}{2}(-S_n + I_n + J_n)$. Acesta este un cod care conține $2n+2$ cuvinte - cod de lungime n și are distanța minimă $d = \frac{n-1}{2}$.

Exemplul 7.5 Pentru $n = 9 = 3^2$, codul Hadamard este format din cele 20 cuvinte care formează liniile matricii

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & J & & P^2 & & & P & & \\ & P & & J & & & P^2 & & \\ & P^2 & & P & & & J & & \\ & I & & J - P^2 & & & J - P & & \\ & J - P & & I & & & J - P^2 & & \\ & J - P^2 & & J - P & & & I & & \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

unde $P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$, iar I și J sunt de ordin 3.

7.3 Coduri produs

Să considerăm un cod de lungime $n = n_1 n_2$. În loc să scriem cuvintele sale sub formă de linii de lungime n , le putem reprezenta ca matrici cu n_1 linii și n_2 coloane. Un mod simplu de a realiza acest lucru este de a scrie cuvântul cod $\mathbf{a} = a_0 a_1 \dots a_{n-1}$ ca o matrice $X = [x_{ij}]$, $0 \leq i \leq n_1 - 1$, $0 \leq j \leq n_2 - 1$, unde $x_{ij} = a_{in_2+j}$.

Aceasta va fi numită *scrierea canonică*.

Definiția 7.3 Fie C_1, C_2 două coduri liniare de lungime n_1 și respectiv n_2 . Se construiește codul C de lungime $n_1 n_2$ ca mai sus (folosind scrierea canonică). Spunem că $C = C_1 \times C_2$ este codul produs al lui C_1 cu C_2 dacă și numai dacă C constă din toate cuvintele - cod în care scrierea canonică verifică proprietățile:

- Fiecare coloană a unei matrici este un cuvânt - cod din C_1 ;
- Fiecare linie a unei matrici este un cuvânt - cod din C_2 .

De remarcat că prin permutarea lui C_1 cu C_2 se obține un cod echivalent.

Teorema 7.7 *Dacă C_1 este un (n_1, k_1) - cod liniar și C_2 este un (n_2, k_2) - cod liniar, atunci $C_1 \times C_2$ este un $(n_1 n_2, k_1 k_2)$ - cod liniar.*

Demonstrație: Fie G_i ($i = 1, 2$) matricile generatoare ale celor două coduri, pe care le presupunem în forma eșalonat canonică; deci $G_i = [I_{k_i} B_{n_i - k_i}]$.

Vom nota cu $\mathbf{g}_i^{(1)}$ ($1 \leq i \leq k_1$) respectiv $\mathbf{g}_i^{(2)}$ ($1 \leq i \leq k_2$) liniile celor două matrici.

Definim matricile X_{ij} ($1 \leq i \leq k_1$, $1 \leq j \leq k_2$) de dimensiune $n_1 \times n_2$, astfel:

- Primele k_1 linii sunt $\mathbf{0}$ cu excepția liniei i care este $\mathbf{g}_j^{(2)}$;
- Primele k_2 coloane sunt $\mathbf{0}$ cu excepția coloanei j care este $\mathbf{g}_i^{(1)T}$.
- Pentru $k > k_1$, linia k este $g_{ik}^{(1)} \mathbf{g}_j^{(2)}$;
- Pentru $k > k_2$, coloana k este $g_{jk}^{(2)} \mathbf{g}_i^{(1)T}$.

Schematic, o astfel de matrice arată astfel:

$$X_{ij} = \left(\begin{array}{ccc|ccc} 0 & \dots & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ & & 1 & x & x & x \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & \dots & 0 \\ \hline 0 & x & 0 & & & \\ \dots & x & \dots & & X & \\ 0 & x & 0 & & & \end{array} \right)$$

Evident, X_{ij} reprezintă un cuvânt - cod din $C_1 \times C_2$ și orice alt cuvânt - cod este o combinație liniară formată din aceste matrici. Deci X_{ij} ($1 \leq i \leq k_1$, $1 \leq j \leq k_2$) formează o bază a codului $C_1 \times C_2$.

Submatricile formate din primele k_1 linii și k_2 coloane formează mulțimea mesajelor de informație. q.e.d.

Exemplul 7.6 *Să considerăm codurile liniare binare C_1, C_2 generate de matricile*

$$G_1 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \text{ respectiv } G_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Deci $n_1 = n_2 = 3$, $k_1 = k_2 = 2$. Sunt 4 matrici X_{ij} , toate de dimensiune 3×3 :

$$\begin{array}{cc} X_{11} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} & X_{12} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \\ X_{21} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} & X_{22} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}. \end{array}$$

Să considerăm mesajul de informație 1011 pe care îl așezăm sub forma unei matrici

$$2 \times 2: \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

Matricea - cod va fi

$$X_{11} + X_{21} + X_{22} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Se observă că liniile sunt cuvinte - cod din C_2 , iar coloanele sunt cuvinte - cod din C_1 .

Propoziția 7.1 Matricea generatoare a codului produs $C_1 \times C_2$ este $G_1 \times G_2$ unde G_1 și G_2 sunt matricile generatoare ale celor două coduri, iar "×" este produsul Kronecker.

Demonstrație: Fie $(n_1, k_1), (n_2, k_2)$ cele două coduri, cu matricile generatoare G_1 respectiv G_2 , și să considerăm matricea $G_1 \times G_2$ obținută prin produsul Kronecker (Capitolul 6, Definiția 6.3).

Dacă se ia linia $k_2i + j$ din această matrice și se reprezintă sub formă canonică ca o matrice $n_1 \times n_2$, se obține exact matricea X_{ij} din construcția Teoremei 7.7. Cum X_{ij} reprezintă o bază pentru codul $C_1 \times C_2$, rezultă că $G_1 \times G_2$ este matricea sa generatoare². q.e.d.

Exemplul 7.7 Matricea generatoare a codului produs $A_1 \times A_2$ din Exemplul 7.6 este:

$$G = G_1 \times G_2 = \begin{pmatrix} G_2 & \mathbf{0} & G_2 \\ \mathbf{0} & G_2 & G_2 \end{pmatrix} = \left(\begin{array}{ccc|ccc|ccc} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array} \right).$$

Se observă imediat că prin "plierea" celor patru linii din această matrice se obțin matricile X_{ij} din Exemplul 7.6.

Să considerăm din nou mesajul de informație 1011. Prin înmulțirea cu matricea G se ajunge la cuvântul - cod 101111010 care - scris ca o matrice 3×3 - coincide cu rezultatul din Exemplul 7.6.

Teorema 7.8 Distanța minimă a codului $C_1 \times C_2$ este egală cu $d_1 \cdot d_2$ (d_1 și d_2 fiind distanțele minime ale codurilor C_1 respectiv C_2).

Demonstrație: Dacă C_i are lungimea n_i ($i = 1, 2$), să considerăm o matrice X reprezentând un cuvânt - cod nenul din $C_1 \times C_2$ (am văzut că asemenea cuvinte există).

Putem găsi deci cel puțin o linie cu minim d_2 elemente nenule. Fiecare astfel de element se află pe o coloană cu minim d_1 elemente nenule; deci matricea X are minim $d_1 \cdot d_2$ elemente nenule. q.e.d.

Exemplul 7.8 Cele două coduri din Exemplul 7.6 au $d_1 = d_2 = 2$ (se găsesc imediat cuvintele - cod și se observă că ponderea lor nenulă minimă este 2). Deci codul produs va avea distanța minimă $d = 2 \cdot 2 = 4$. De remarcat că - deși C_1 și C_2 nu pot corecta erori, $C_1 \times C_2$ este capabil să corecteze o eroare.

Codurile produs - cu toate că au fost definite de la începutul anilor '60, au devenit foarte importante odată cu folosirea lor în transmisiile telefonice fără fir și codificarea informației pe CD - uri și DVD-uri. Vom relua ulterior - pe larg - analiza acestor coduri.

²Din acest motiv, în primii ani, codurile produs erau numite și coduri Kronecker.

7.4 Coduri optimal maximale

Fie mulțimea $C \subseteq Z_q^n$. Se numește *diametrul* mulțimii C numărul

$$d(C) = \min \{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}\}.$$

De remarcat că, dacă C este cod liniar, atunci $d(C)$ este chiar distanța sa minimă.

Definiția 7.4 C este mulțime optimală de distanță minimă d dacă:

- $d(C) = d$;
- $\forall \mathbf{x} \in Z_q^n \setminus C, d(C \cup \{\mathbf{x}\}) < d$.

Exemplul 7.9 În Z_2^6 , următoarele mulțimi sunt optimale de distanță minimă 3:

$$A = \{000000, 010101, 101010, 111111\}$$

$$B = \{000000, 101010, 011001, 000111, 110100, 111111\}$$

Definiția 7.5 Se numește mulțime optimală cardinal maximală de distanță minimă d o mulțime optimală din Z_q^n având un număr maxim de elemente.

Vom nota acest număr cu $a(n, d, q)$.

Exemplul 7.10 Mulțimile date în Exemplul 7.9 sunt optimale dar nu cardinal maximale pentru distanța 3. O mulțime optimală cardinal maximală din Z_2^6 pentru $d = 3$ este

$$C = \{000000, 001011, 010101, 011110, 100110, 101101, 110011, 111000\},$$

deci $a(6, 3, 2) = 8$.

Definiția 7.6 Un cod liniar C de distanță minimă d este optimal dacă C este mulțime optimală cardinal maximală de distanță minimă d .

Spunem că un astfel de cod satisface *condiția max - min*, deoarece are un număr maxim de vectori aflați la distanța minimă d .

Nu se cunosc algoritmi de construcție a unor astfel de coduri. În această secțiune vom da numai câteva condiții necesare pentru definirea codurilor optimale.

Teorema 7.9 Într-un cod optimal C de lungime n și distanță minimă d ,

$$a(n, d, q) \leq q \cdot a(n - 1, d, q).$$

Demonstrație: Putem presupune – fără a micșora generalitatea – că primul caracter al codului este caracter de informație.

Fie K mulțimea cuvintelor din C care au 0 pe prima componentă. Cum C este cod liniar, iar pe prima componentă pot apare q caractere distincte, va rezulta

$$\text{card}(K) = \frac{a(n, d, q)}{q}.$$

K este tot un cod liniar cu distanța minimă d , în care toate cuvintele au 0 pe prima poziție. Dacă eliminăm această componentă, se obține un cod din Z_q^{n-1} , tot de distanță minimă d , cu același număr de cuvinte - cod.

Acest cod nu este neapărat optimal; deci condiția va fi

$$a(n-1, d, q) \geq \frac{a(n, d, q)}{q}$$

Mai trebuie făcută observația că $a(n-1, n-1, q) = q$ pentru că un cod cu repetiție este optimal de distanță minimă egală cu lungimea cuvintelor - cod. q.e.d.

Corolarul 7.1 Dacă $j \geq d$, atunci $q^{n-j}a(j, d, q) \geq a(n, d, q)$.

Demonstrație: Din Teorema 7.9 rezultă inegalitățile:

$$\begin{aligned} q \cdot a(n-1, d, q) &\geq a(n, d, q) \\ q \cdot a(n-2, d, q) &\geq a(n-1, d, q) \\ \dots \\ \underline{q \cdot a(n-i, d, q) &\geq a(n-i+1, d, q)} \end{aligned}$$

Prin înmulțire se obține $q^i \cdot a(n-i, d, q) \geq a(n, d, q)$, după care se face notația $n-i = j$. q.e.d.

Teorema 7.10 Fie $A_{n,k} \subseteq Z_q^n$ un cod liniar optimal de distanță minimă d .

Dacă $n < \frac{qd}{q-1}$, atunci numărul k al simbolurilor de informație verifică inegalitatea

$$k \leq n - \frac{dq-1}{q-1} + 1 + \log_q d.$$

Demonstrație: Deoarece $A_{n,k}$ este cod liniar, numărul de simboluri de informație va fi dat de relația $a(n, d, q) = q^k$.

Rescriem inegalitatea lui Plotkin (Capitolul 3) sub forma $d(q^k - 1) \leq nq^{k-1}(q-1)$, sau

$$q^{k-1}(dq + n - nq) \leq n.$$

Deoarece - din ipoteză - $dq + n - nq > 0$, avem $q^k \leq \frac{dq}{dq + n - nq}$, sau

$$a(n, d, q) \leq \frac{dq}{dq + n - nq}.$$

Fie $\frac{dq-1}{q-1} = i + f$ unde $i = \left\lfloor \frac{dq-1}{q-1} \right\rfloor$, $f = \left\{ \frac{dq-1}{q-1} \right\}$.

Deci $qd - 1 = (q-1)i + (q-1)f$ și avem

$$a(i, d, q) \leq \frac{dq}{dq + i - iq} = \frac{dq}{dq - (q-1)i} = \frac{dq}{1 + (q-1)f}.$$

Folosind Corolarul 15.1 și $i \geq d$ (deoarece $\left\lceil \frac{dq-1}{q-1} \right\rceil = \left\lceil d + \frac{d-1}{q-1} \right\rceil \geq d$), rezultă

$$a(n, d, q) \leq q^{n-i} a(i, d, q) \leq \frac{q^{n-\frac{dq-1}{q-1}+f}}{1+(q-1)f} qd.$$

Dezvoltând în serie Taylor, $q^f \leq 1 + (q-1)f$ ($f < 1$), și deci

$$a(n, d, q) \leq q^{n-\frac{dq-1}{q-1}} qd,$$

sau $q^k \leq q^{n-\frac{dq-1}{q-1}} qd$, de unde – prin logaritmare – se obține relația din enunț. q.e.d.

7.5 Exerciții

7.1 Să se construiască vectorii ponderilor pentru codurile Hamming (7, 4) și $\mathcal{RM}(1, 3)$.

7.2 Folosind vectorul ponderilor, să se construiască numărătorul de ponderi al unui cod liniar.

7.3 Să se construiască matricile generatoare și matricile K pentru codurile McDonald cu $k = 2, 3, 4$.

7.4 Să se construiască matricile și codurile Hadamard pentru $n = 4$ și $n = 8$.

7.5 Să se arate că dacă \mathcal{H}_n este o matrice Hadamard, atunci $\mathcal{H}_n \mathcal{H}_n^T = nI_n$.

7.6 Să se construiască o matrice Hadamard de ordin 12.

7.7 Să se arate că matricea Hadamard \mathcal{H}_8 generează (8, 4) - codul Hamming binar extins.

7.8 Să se construiască codurile produs pentru codul cu repetiție de lungime n (C_1) și codul cu repetiție de lungime p (C_2).

7.9 Să se construiască matricea generatoare a codului produs $C \times C$ unde C este codul Hamming (7, 4).

7.10 Aceeași problemă pentru codul $\mathcal{RM}(1, 3)$.

7.11 Demonstrați Teorema 7.5.

7.12 Folosind matricile Paley, să se construiască codurile Hadamard de ordin 5, 7 și respectiv 10.

7.13 Să se arate că un cod binar liniar optimal de distanță minimă d are cel puțin $2d - 1 - \log_2 d$ simboluri de control.

Capitolul 8

Circuite liniare și extensii Galois

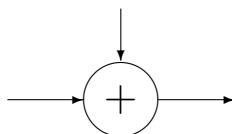
8.1 Circuite liniare pentru operații elementare

În această secțiune presupunem că toate calculele se realizează într-un corp de caracteristică q (q număr prim).

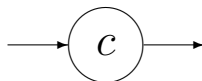
Un *circuit liniar* este un graf orientat, cu trei tipuri de noduri (conectori):

- *Sumator*: este un conector cu cel puțin două intrări și o ieșire. Efectul este acela de a scoate suma modulo q a elementelor aflate la intrare. Operația se execută instantaneu.

Un sumator (cu 2 intrări) se notează astfel:



- *Multiplicator*: este un conector cu o singură intrare și o ieșire unde se obține rezultatul înmulțirii cu $c \in Z_q$ a elementului de la intrare. Operația se execută (de asemenea) instantaneu. Notăție:



- *Element de înmagazinare*: este un registru *DFF* cu o intrare și o ieșire; efectul este întârzierea cu un tact a intrării: elementul $a \in Z_q$ intră în momentul k și iese în momentul $k + 1$. Notăție:



Aceste trei tipuri de conectori se pot grupa arbitrar – folosind eventual și noduri obișnuite. Utilizarea lor conduce la obținerea de circuite liniare care pot realiza automat operații uzuale cu polinoame: adunări, scăderi, înmulțiri și împărțiri.

În cele ce urmează, vom identifica un vector cu $n + 1$ componente

$$\mathbf{a} = (a_0, a_1, \dots, a_n)$$

cu polinomul de gradul n cu o variabilă

$$a(X) = a_0 + a_1X + \dots + a_nX^n$$

În cursul transmisiei mesajului \mathbf{a} , ordinea de prelucrare (transmisie) se face după puterile descrescătoare ale lui X ; deci, sensul de circulație a semnalelor va fi:

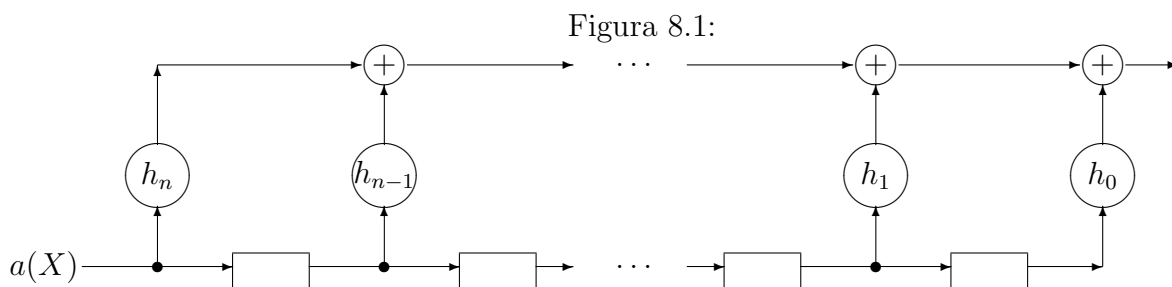
$$a_0 \quad a_1 \quad \dots \quad a_n \longrightarrow$$

8.1.1 Construcția unor circuite liniare uzuale

A *Circuit de înmulțire cu un polinom.*

Fie $h(X) = h_0 + h_1X + \dots + h_nX^n$ un polinom fixat și $a(X) = a_0 + a_1X + \dots + a_kX^k$ un polinom arbitrar. Coeficienții ambelor polinoame pot fi luați într-un inel arbitrar (din considerente pur aplicative, acesta este de obicei Z_q , dar construcțiile rămân valabile și în cazul general).

Un circuit de înmulțire cu $h(X)$ este:



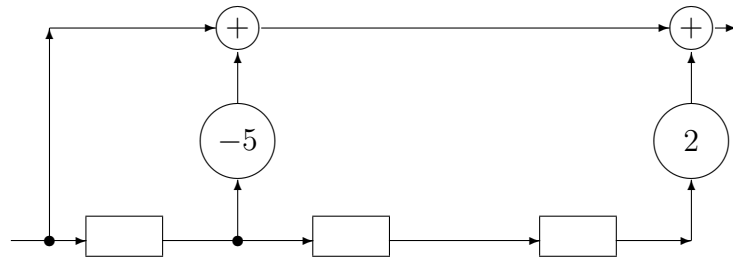
Inițial elementele de înmagazinare conțin 0; coeficienții polinomului $a(X)$ intră în stânga jos – după puterile descrescătoare ale lui X – câte unul la un tact, iar coeficienții produsului ies în dreapta sus. Circuitul funcționează $n + k + 1$ tacti; după $k + 1$ tacti, la intrare se introduc zerouri în cei n tacti rămași, până ce elementele de înmagazinare conțin din nou numai 0.

De remarcat că un element de înmagazinare funcționează după principiile unei variabile de memorie (introducerea unei valori face să dispară vechea valoare din locație).

Circuitul lucrează după formula de înmulțire obișnuită:

$$a(X)h(X) = a_0h_0 + (a_0h_1 + a_1h_0)X + \dots + (a_{k-1}h_n + a_kh_{n-1})X^{k+n-1} + a_kh_nX^{n+k}$$

Exemplul 8.1 Să luăm polinomul $h(X) = X^3 - 5X^2 + 2 \in Z[X]$ (vectorul corespunzător este $\mathbf{h} = (2, 0, -5, 1)$). Circuitul de înmulțire cu $h(X)$ va fi:



De remarcat că multiplicarea cu 1 se face prin arc simplu (fără conector), iar multiplicarea cu 0 revine la suprimarea arcului. În cazul operațiilor cu polinoame din $Z_2[X]$, toți multiplicatorii se reduc la aceste două cazuri.

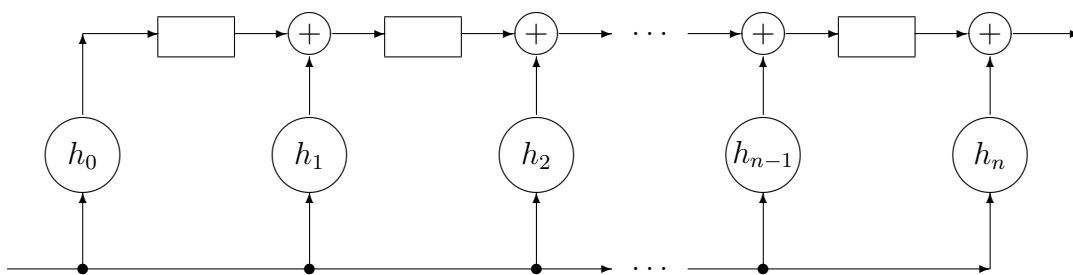
Dacă dorim să înmulțim $h(X)$ cu polinomul $a(X) = 2X + 8$ (vector $\mathbf{a} = (8, 2)$), circuitul linear va funcționa 5 tacti (până când toate elementele de înmagazinare revin la 0); comportarea sa este dată de tabelul:

Nr. tact	Intrare	Înmagazinare			Ieșire
0	0	0	0	0	0
1	2	0	0	0	2
2	8	2	0	0	-2
3	0	8	2	0	-40
4	0	0	8	2	4
5	0	0	0	8	16
6	0	0	0	0	0

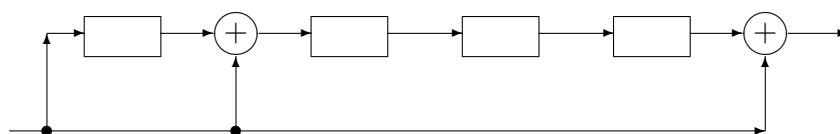
Deci, polinomul produs este $2X^4 - 2X^3 - 40X^2 + 4X + 16$; sau - în termeni de vectori, $(16, 4, -40, -2, 2)$.

Aceași operație de înmulțire cu un polinom fixat $h(X)$ este realizată și de circuitul linear:

Figura 8.2:



Exemplul 8.2 Circuitul linear care realizează înmulțirea cu polinomul $h(X) = 1 + X + X^4$ peste Z_2 este

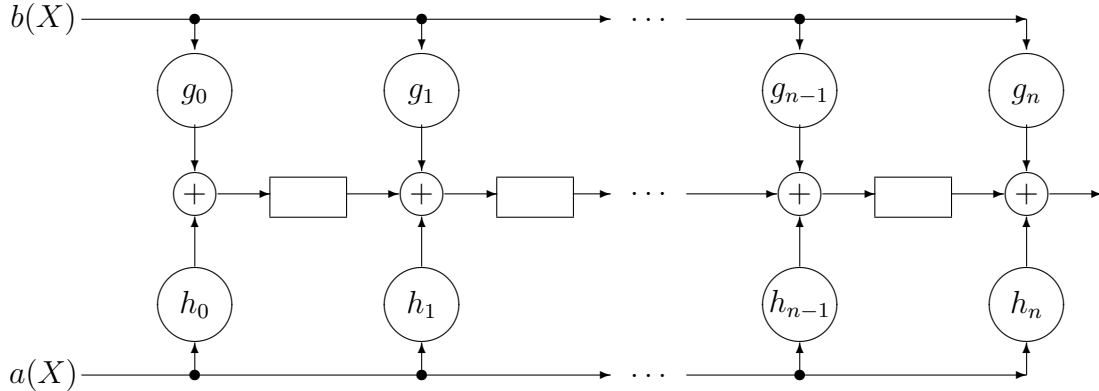


deoarece vectorul corespunzător polinomului $h(X)$ este $\mathbf{h} = (1, 1, 0, 0, 1)$.

B. Fie $h(X) = h_0 + h_1X + \dots + h_nX^n$ și $g(X) = g_0 + g_1X + \dots + g_nX^n$ două polinoame fixate. Un circuit liniar care să realizeze operația

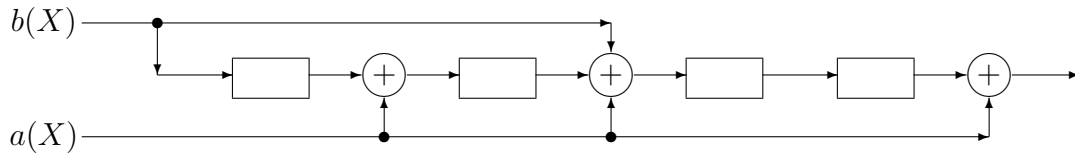
$$a(X)h(X) + b(X)g(X)$$

pentru două polinoame arbitrare $a(X), b(X) \in Z_q[X]$, este:



De remarcat că polinoamele $g(X)$ și $h(X)$ pot avea grade diferite; în acest caz se alege n ca fiind gradul cel mai mare și se completează celălalt polinom cu coeficienți nuli până la gradul n .

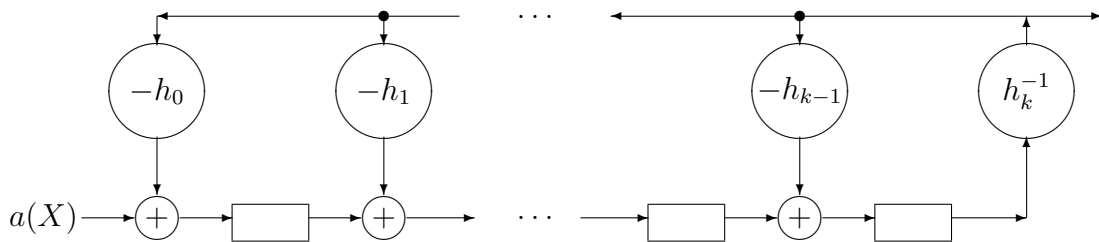
Exemplul 8.3 Fie polinoamele $g(X) = 1 + X^2$, $h(X) = X + X^2 + X^4$ cu coeficienți în Z_2 ; deci vectorii corespunzători vor fi $\mathbf{h} = (0, 1, 1, 0, 1)$, $\mathbf{g} = (1, 0, 1, 0, 0)$. Circuitul liniar care realizează expresia $a(X)h(X) + b(X)g(X)$ este:



C. Circuit liniar de împărțire cu un polinom.

Fie polinomul (fixat) $h(X) = h_0 + h_1X + \dots + h_kX^k$ unde $h_k \neq 0$. Un circuit liniar care realizează împărțirea unui polinom arbitrar $a(X) = a_0 + a_1X + \dots + a_nX^n$ la $h(X)$ este:

Figura 8.3:

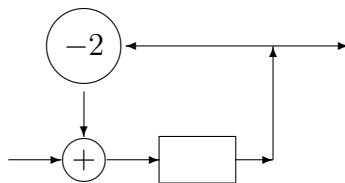


Inițial, toate cele k elemente de înmagazinare conțin 0. La intrare, timp de $n + 1$ tacti se introduc coeficienții polinomului $a(X)$ (în ordinea descrescătoare a puterilor). În primii k tacti, ieșirea va scoate numai 0. Primul coeficient nenul apare la ieșire la momentul $k + 1$ și este $a_n h_k^{-1}$ (operații în inelul coeficienților celor două polinoame) – coeficientul termenului X^{n-k} din cât.

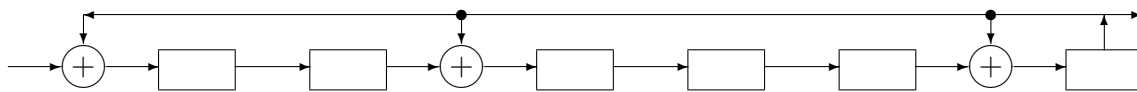
Pentru fiecare coeficient q_j al câtului, polinomul $q_j h(X)$ trebuie scăzut din deîmpărțit; această operație se realizează printr-un procedeu de conexiune inversă (feedback).

După momentul $n + 1$ (când au intrat toți coeficienții polinomului $a(X)$) la ieșire s-a obținut câtul $q(X)$ al împărțirii lui $a(X)$ la $h(X)$, iar în elementele de înmagazinare se găsesc coeficienții restului.

Exemplul 8.4 Un circuit liniar de împărțire (peste Q) la polinomul $h(X) = 2 + X$ este:



Exemplul 8.5 Fie polinomul $h(X) = 1 + X^2 + X^5 + X^6 \in \mathbb{Z}_2[X]$. Circuitul care realizează împărțirea este:



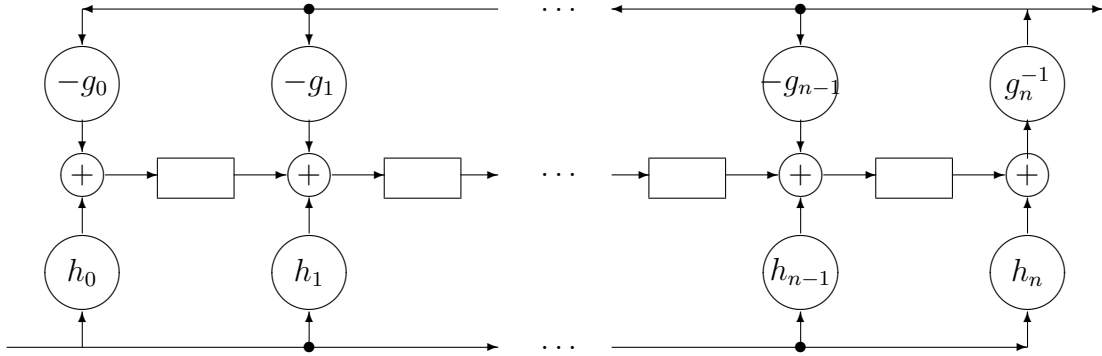
Să luăm polinomul $a(X) = 1 + X^4 + X^5 + X^7 + X^9 + X^{10}$ pe care să-l împărțim în \mathbb{Z}_2 la polinomul $h(X)$. Se obține câtul $X^4 + X$ și restul $X^5 + X^3 + X + 1$.

Circuitul funcționează 11 tacti. Etapele prin care trece circuitul pentru efectuarea acestei împărțiri sunt reprezentate de tabelul următor:

Nr. tact	Intrare	Elemente de înmagazinare	Ieșire
0	—	0 0 0 0 0 0	—
1	1	1 0 0 0 0 0	0
2	1	1 1 0 0 0 0	0
3	0	0 1 1 0 0 0	0
4	1	1 0 1 1 0 0	0
5	0	0 1 0 1 1 0	0
6	1	1 0 1 0 1 1	0
7	1	0 1 1 1 0 0	1
8	0	0 0 1 1 1 0	0
9	0	0 0 0 1 1 1	0
10	0	1 0 1 0 1 0	1
11	1	1 1 0 1 0 1	0

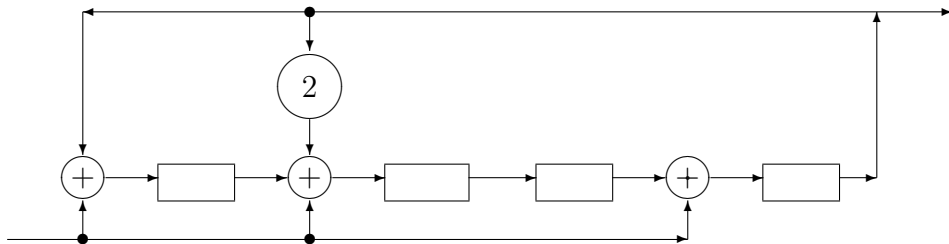
Vectorii $(1, 1, 0, 1, 0, 1)$ - de pe ultima linie și $(0, 1, 0, 0, 1)$ - scris îngroșat și în ordine inversă pe ultima coloană - reprezintă restul $1 + X + X^3 + X^5$ respectiv câtul $X + X^4$ împărțirii lui $a(X)$ la $h(X)$.

D. Circuitul liniar care realizează înmulțirea cu polinomul $h(X) = h_0 + h_1X + \dots + h_nX^n$ urmată de împărțirea la polinomul $g(X) = g_0 + g_1X + \dots + g_nX^n$ este:



Când gradele celor două polinoame sunt diferite, se construiește întâi circuitul liniar corespunzător polinomului de grad maxim, după care se completează cu circuitul corespunzător celuilalt polinom.

Exemplul 8.6 Să luăm polinoamele $h(X) = 1 + X + X^3$, $g(X) = 2 + X + X^4$ cu coeficienți în Z_3 . Circuitul liniar care realizează înmulțirea cu $h(X)$ și împărțirea la $g(X)$ este



Deoarece calculele se fac în Z_3 , $-2 = 1$, $-1 = 2$, și $1^{-1} = 1$.

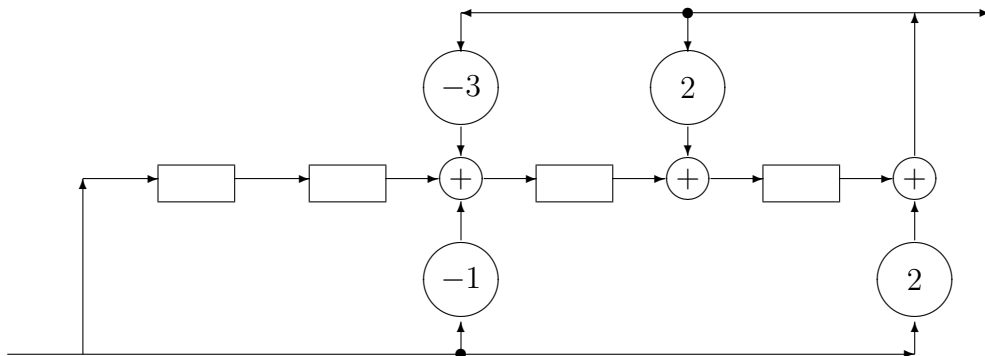
Pentru intrarea $a(X) = 1 + 2X + 2X^2$, comportarea circuitului este următoarea:

-	0	0	0	0	-
2	2	2	0	2	-
2	1	2	2	2	2
1	0	0	2	0	2

cea ce corespunde câtului $2 + 2X$ și restului $2X^2$.

De remarcat că primul coeficient (de grad maxim) al câtului este scos de circuit cu o întârziere de $\text{grad}(g(X)) - \text{grad}(h(X)) = 1$ tact.

Exemplul 8.7 Un circuit liniar pentru înmulțirea cu $h(X) = 1 - X^2 + 2X^4$ și împărțirea la $g(X) = 3 - 2X + X^2$ - ambele din $Z[X]$ - este:



Fie intrarea $a(X) = -2 + X$; comportarea circuitului este:

–	0	0	0	0	–
1	1	0	–7	4	2
–2	–2	1	2	–7	0
0	0	–2	22	–12	–7
0	0	0	34	–2	–12

deci câțul este $2X^3 - 7X - 12$ iar restul $34 - 2X$.

De remarcat că circuitul este lăsat să funcționeze $\text{grad}(h(X)) - \text{grad}(g(X)) = 2$ tați în plus (prin introducerea de zero-uri), pentru golirea elementelor de înmagazinare necontrolate de împărțitor.

8.2 Extensii Galois

Fie q un număr prim și $h(X) = h_0 + h_1X + \dots + h_nX^n \in Z_q[X]$. Vom considera algebra polinoamelor modulo $h(X)$.

Fiind dat un polinom oarecare din $Z_q[X]$, clasa sa de resturi modulo $h(X)$ se găsește împărțind polinomul respectiv la $h(X)$. Restul împărțirii specifică clasa de resturi respectivă.

Două polinoame cărora le corespunde același rest la împărțirea cu $h(X)$ sunt echivalente, intrând în aceeași clasă de resturi modulo $h(X)$.

Fiecare clasă de resturi modulo $h(X)$ se reprezintă prin polinomul (unic) de grad strict mai mic decât n , care aparține clasei respective. Toți acești reprezentanți pot fi considerați ca polinoame de gradul $n - 1$, având eventual coeficienți nuli.

Vom nota cu $\{f(X)\}$ clasa de resturi a polinomului $f(X)$ modulo $h(X)$. Deci $\{f(X)\}$ va corespunde unui vector (distinct) de lungime n , cu componente din Z_q (prin completare eventual cu zerouri).

Operațiile în algebra claselor de resturi sunt:

$$\begin{aligned} \{a(X)\} + \{b(X)\} &= \{a(X) + b(X)\} \\ c\{a(X)\} &= \{ca(X)\}, \quad c \in Z_q \\ \{a(X)\}\{b(X)\} &= \{a(X)b(X)\} \end{aligned}$$

toate calculele fiind făcute modulo $h(X)$.

Evident, $\{0\} = \{h(X)\}$, unde $\{0\}$ este polinomul de grad $n - 1$ cu toți coeficienții nuli (sau vectorul cu n componente zero). Vom nota $\{0\} = \mathbf{0}$.

Pentru un polinom $s(X) \in Z_q[X]$, clasa de resturi modulo $h(X)$ se obține foarte ușor. Teorema împărțirii cu rest dă:

$$s(X) = a(X)h(X) + r(X)$$

unde $\text{grad}(r(X)) < \text{grad}(h(X)) = n$. Atunci:

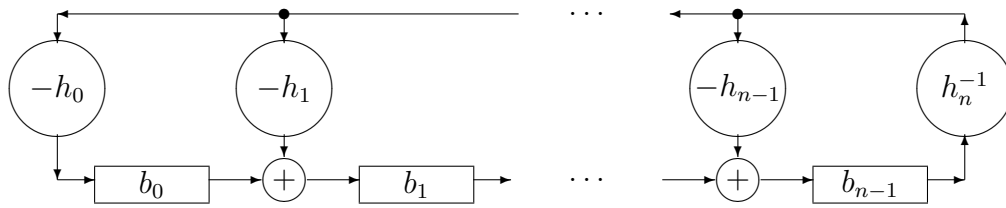
$$\{s(X)\} - \{r(X)\} = \{s(X) - r(X)\} = \{a(X)h(X)\} = \{a(X)\}\{h(X)\} = \mathbf{0},$$

deci $\{s(X)\} = \{r(X)\}$.

Să reluăm circuitul liniar de împărțire definit în Figura 8.3, în care facem următoarele modificări (vezi Figura 8.4):

- Se neglijează intrarea și ieșirea;
- La momentul inițial elementele de înmagazinare conțin coeficienții unui polinom $b(X) = b_0 + b_1X + \dots + b_{n-1}X^{n-1}$.

Figura 8.4:



La următorul tact, în elementele de înmagazinare se obțin coeficienții polinomului:

$$\begin{aligned} b'(X) &= -h_0h_n^{-1}b_{n-1} + (b_0 - h_1h_n^{-1}b_{n-1})X + (b_1 - h_2h_n^{-1}b_{n-1})X^2 + \dots + \\ &\quad + (b_{n-2} - h_{n-1}h_n^{-1}b_{n-1})X^{n-1} = \\ &= b_0X + b_1X^2 + \dots + b_{n-1}X^n - b_{n-1}X^n - h_n^{-1}b_{n-1}(h_0 + h_1X + \dots + h_{n-1}X^{n-1}) = \\ &= Xb(X) - h_n^{-1}h_{n-1}g(X). \end{aligned}$$

Dacă notăm $\alpha = \{X\}$ ($\{X\}$ este clasa de resturi modulo $h(X)$ a polinomului X), din relația de mai sus se obține:

$$b'(\alpha) = \{b'(X)\} = \{Xb(X)\} = \{X\}\{b(X)\} = \{X\}b(\{X\}) = \alpha b(\alpha).$$

Deci circuitul de împărțire cu polinomul $h(X)$ poate fi utilizat pentru înmulțirea lui $b(\alpha)$ cu α .

Teorema 8.1 În algebra polinoamelor modulo $h(X)$, $\text{grad}(h(X)) = n$, avem:

1. $h(\alpha) = \mathbf{0}$ unde $\alpha = \{X\}$;
2. h este polinomul de grad minim care are pe α ca rădăcină.

Demonstrație:

1. Fie $h(X) = h_0 + h_1X + \dots + h_nX^n \in Z_q[X]$, $h_n \neq 0$.

$$\begin{aligned} \text{Vom avea } h(\alpha) &= h_0 + h_1\alpha + \dots + h_n\alpha^n = h_0 + h_1\{X\} + \dots + h_n\{X\}^n = \\ &= h_0 + h_1\{X\} + \dots + h_n\{X^n\} = \{h_0 + h_1X + \dots + h_nX^n\} = \{h(X)\} = \mathbf{0}. \end{aligned}$$

2. Folosind același raționament, se obține pentru orice polinom $f(X)$ cu $\text{grad}(f(X)) < n$, că $f(\alpha) = \{f(X)\} \neq \mathbf{0}$, deoarece toate polinoamele ne-identice nule de grad strict mai mic decât n aparțin la clase de resturi distincte. q.e.d.

Să presupunem acum că $h(X) \in Z_q[X]$ este un polinom ireductibil de gradul n . Conform Teoremei 19.1, $\alpha = \{X\}$ este rădăcină a lui $h(X)$. Acest element α poate fi considerat fie ca un polinom de gradul $n - 1$ (cu toți coeficienții nuli înafară de cel al lui X), fie ca un vector cu n componente $(0, 1, 0, \dots, 0)$.

Vom nota cu $GF(q^n)$ mulțimea vectorilor din Z_q^n , sau – echivalent:
 - mulțimea polinoamelor de grad $n - 1$ cu coeficienți în Z_q , sau
 - mulțimea claselor de resturi modulo polinomul $h(X) \in Z_q[X]$ de gradul n , ireductibil peste Z_q .
 $GF(q^n)$ se numește *extensia Galois de grad n* a lui Z_q .

Spunem că $GF(q^n)$ se obține prin adăugarea la Z_q a unei rădăcini a polinomului $h(X) \in Z_q[X]$ de grad n , ireductibil peste corpul de bază Z_q .

Observația 8.1 *Procedeul de trecere de la Z_q la $GF(q^n)$ este similar trecerii de la corpul numerelor reale \mathcal{R} la cel complex \mathcal{C} , prin adăugarea rădăcinii $i = \{X\}$ în algebra claselor de resturi modulo polinomul $1 + X^2$, ireductibil peste \mathcal{R} . Putem considera mulțimea \mathcal{C} a numerelor complexe fie ca mulțimea polinoamelor de grad $2 - 1 = 1$, $a + bi$ cu $a, b \in \mathcal{R}$, fie ca mulțimea vectorilor (a, b) cu două componente reale.*

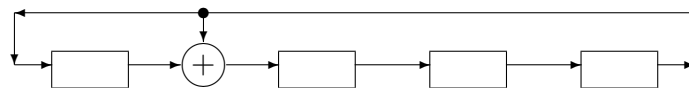
Dacă polinomul $h(X) \in Z_q[X]$ este ireductibil peste Z_q și primitiv, puterile lui $\alpha = \{X\}$ vor genera toate elementele lui $GF(q^n) \setminus \{0\}$ unde $n = \text{grad}(h(X))$.

Acest lucru se poate realiza cu circuitul liniar din Figura 8.4, introducând la momentul inițial în elementele de înmagazinare coeficienții polinomului $b(X) = 1$ (adică vectorul $(1, 0, \dots, 0)$).

Conform celor observate mai sus, la momentul următor se obține $\{Xb(X)\} = \{X\} = \alpha$, după care urmează pe rând $\alpha^2, \alpha^3, \dots$.

Cum $h(X)$ a fost ales primitiv, el va genera toate elementele nenule din $GF(q^n)$.

Exemplul 8.8 *Fie polinomul $1 + X + X^4$ ireductibil peste Z_2 și $\alpha = \{X\}$ în algebra claselor de resturi modulo $1 + X + X^4$. Circuitul liniar care va genera toate elementele nenule din $GF(2^4)$ este:*



unde, la momentul inițial, în elementele de înmagazinare este vectorul $(1, 0, 0, 0)$.

Conform Teoremei 19.1, α este o rădăcină a polinomului $1 + X + X^4$, adică

$$1 + \alpha + \alpha^4 = 0.$$

Lăsând circuitul să funcționeze, în elementele de înmagazinare se obțin toate elementele lui $GF(2^4)$, pe care le interpretăm fie ca vectori cu patru componente peste Z_2 , fie ca polinoame de gradul 3 în α cu coeficienți în Z_2 .

α^0	=	1				1	0	0	0				
α^1	=		α			0	1	0	0				
α^2	=			α^2		0	0	1	0				
α^3	=				α^3	0	0	0	1				
α^4	=	1	+	α		1	1	0	0				
α^5	=			α	+	α^2	0	1	1	0			
α^6	=				α^2	+	α^3	0	0	1	1		
α^7	=	1	+	α				α^3	1	1	0	1	
α^8	=	1			+	α^2			1	0	1	0	
α^9	=			α				+	α^3	0	1	0	1
α^{10}	=	1	+	α	+	α^2				1	1	1	0
α^{11}	=			α	+	α^2	+	α^3	0	1	1	1	1
α^{12}	=	1	+	α	+	α^2	+	α^3	1	1	1	1	1
α^{13}	=	1			+	α^2	+	α^3	1	0	1	1	1
α^{14}	=	1					+	α^3	1	0	0	1	1
α^{15}	=	1							1	0	0	0	0

Vedem deci că polinomul $1 + X + X^4$ este primitiv, puterile lui α epuizând toți vectorii nenuli din $GF(2^4) = \{0, \alpha^0, \alpha, \dots, \alpha^{14}\}$.

Exemplul 8.9 Tabelul din exemplul precedent descrie operația de înmulțire din extensia galios $GF(2^4)$ generată de rădăcina α a polinomului primitiv $1 + X + X^4$. Dacă s-ar alege α ca rădăcină a altui polinom primitiv – de exemplu $1 + X^3 + X^4$, se va genera tot $GF(2^4)$, dar într-o altă ordine.

Lăsăm ca exercițiu această generare.

În $GF(2^4)$ se poate defini și operația de adunare, însumând modulo 2 componentele celor doi operanzi. Astfel, dacă reluăm $GF(2^4)$ generat în Exemplul 8.8 de rădăcina polinomului $1 + X + X^4$, tabela de adunare va fi:

+	0	α^0	α	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}
0	0	α^0	α	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}
α^0	α^0	0	α^4	α^8	α^{14}	α	α^{10}	α^{13}	α^9	α^2	α^7	α^5	α^{12}	α^{11}	α^6	α^3
α	α	α^4	0	α^5	α^9	α^0	α^2	α^{11}	α^{14}	α^{10}	α^3	α^8	α^6	α^{13}	α^{12}	α^7
α^2	α^2	α^8	α^5	0	α^6	α^{10}	α	α^3	α^{12}	α^0	α^{11}	α^4	α^9	α^7	α^{14}	α^{13}
α^3	α^3	α^{14}	α^9	α^6	0	α^7	α^{11}	α^2	α^4	α^{13}	α	α^{12}	α^9	α^{10}	α^8	α^0
α^4	α^4	α	α^0	α^{10}	α^7	0	α^8	α^{12}	α^3	α^5	α^{14}	α^2	α^{13}	α^6	α^{11}	α^9
α^5	α^5	α^{10}	α^2	α	α^{11}	α^8	0	α^9	α^{13}	α^4	α^6	α^0	α^3	α^{14}	α^7	α^{12}
α^6	α^6	α^{13}	α^{11}	α^3	α^2	α^{12}	α^9	0	α^{10}	α^{14}	α^3	α^7	α	α^4	α^0	α^8
α^7	α^7	α^9	α^{14}	α^{12}	α^4	α^3	α^{13}	α^{10}	0	α^{11}	α^0	α^6	α^8	α^2	α^9	α
α^8	α^8	α^2	α^{10}	α^0	α^{13}	α^5	α^4	α^{14}	α^{11}	0	α^{12}	α	α^7	α^9	α^3	α^6
α^9	α^9	α^7	α^3	α^{11}	α	α^{14}	α^6	α^5	α^0	α^{12}	0	α^{13}	α^2	α^8	α^{10}	α^4
α^{10}	α^{10}	α^5	α^8	α^4	α^{12}	α^2	α^0	α^7	α^6	α	α^{13}	0	α^{14}	α^3	α^9	α^{11}
α^{11}	α^{11}	α^{12}	α^6	α^9	α^5	α^{13}	α^3	α	α^8	α^7	α^2	α^{14}	0	α^0	α^4	α^{10}
α^{12}	α^{12}	α^{11}	α^{13}	α^7	α^{10}	α^6	α^{14}	α^4	α^2	α^9	α^8	α^3	α^0	0	α	α^5
α^{13}	α^{13}	α^6	α^{12}	α^{14}	α^8	α^{11}	α^7	α^0	α^5	α^3	α^{10}	α^9	α^4	α	0	α^2
α^{14}	α^{14}	α^3	α^7	α^{13}	α^0	α^9	α^{12}	α^8	α	α^6	α^4	α^{11}	α^{10}	α^5	α^2	0

Fie acum $a(X) = a_0 + a_1X + \dots + a_{k-1}X^{k-1} \in Z_q[X]$. Valoarea $a(\alpha)$ se obține folosind circuitul de împărțire dat în Figura 8.3, cu următoarele observații:

- Se ignoră ieșirea;
- Coeficienții lui $a(X)$ se introduc la intrare în ordinea descrescătoare a puterilor;
- Inițial, elementele de înmagazinare conțin valorile $(0, 0, \dots, 0)$.

Pentru calculul unei valori de forma $a(\alpha)\alpha^j$ ($j \geq 0$ fixat), se pot construi circuite liniare care să realizeze direct acest produs.

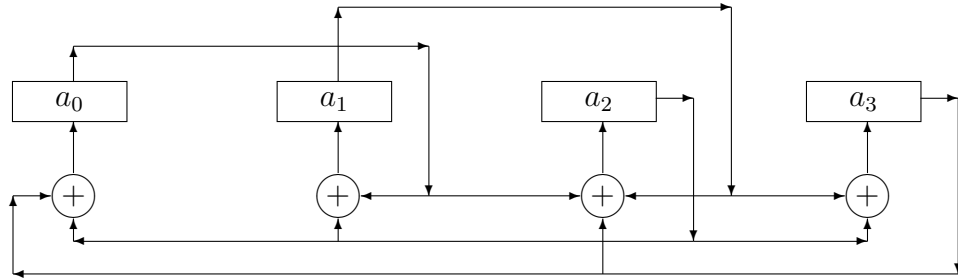
Exemplul 8.10 Fie $\alpha \in GF(2^4)$ element primitiv, soluție a ecuației $1 + X + X^4 = 0$. Polinomul $a(X)$ va avea atunci gradul maxim 3. Fie $a(X) = a_0 + a_1X + a_2X^2 + a_3X^3$.

Să construim un circuit liniar care să efectueze $a(\alpha)\alpha^5$.

Folosind tabelul din Exemplul 8.8, avem:

$$(a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3)\alpha^5 = a_0\alpha^5 + a_1\alpha^6 + a_2\alpha^7 + a_3\alpha^8 = a_0(\alpha + \alpha^2) + a_1(\alpha^2 + \alpha^3) + a_2(1 + \alpha + \alpha^2) + a_3(1 + \alpha^2) = (a_2 + a_3) + (a_0 + a_2)\alpha + (a_0 + a_1 + a_3)\alpha^2 + (a_1 + a_2)\alpha^3$$

Pentru acest polinom se va construi circuitul liniar:



La momentul inițial, în elementele de înmagazinare se găsesc coeficienții lui $a(\alpha)$; după un tact, aici vor fi coeficienții polinomului $a(\alpha)\alpha^5$.

8.3 Exerciții

8.1 Să se realizeze circuite liniare (în ambele modalități de construcție) care să efectueze înmulțiri cu polinoamele:

$$\begin{aligned}h(X) &= 1 + 2X + 3X^2 + 4X^3 \in Z[X] \\h(X) &= 1 + 2X + X^5 \in Z_7[X]\end{aligned}$$

8.2 Să se reprezinte sub formă de tabel comportamentul circuitelor realizate anterior, la înmulțirea cu polinoamele:

$$a(X) = 2 + 5X + X^3, \quad a(X) = X - X^3 - 2X^4, \quad a(X) = 0.$$

8.3 Se dă vectorul $\mathbf{h} = (1, 0, 0, -1, 0, 1)$. Să se realizeze circuitul de înmulțire cu polinomul corespunzător și să se reprezinte comportamentul acestui circuit la înmulțirea cu polinoamele reprezentate prin vectorii $\mathbf{a} = (1)$ și $\mathbf{a} = (1, 1, 1, 0, 0, 0, 1)$.

8.4 Să se realizeze un circuit liniar care realizează produsul $a(X)g(X) + b(X)h(X)$ pentru polinoamele:

$$\begin{aligned}g(X) &= 1 + X + X^3, & h(X) &= 1 - 5X - 2X^2 + 3X^3; \\g(X) &= -4X^3 + X^5 - 2X^8, & h(X) &= X^2 + 2X^4 + X^6; \\g(X) &= X - X^3 + 3X^4, & h(X) &= 2X^3 + X^5 + X^9.\end{aligned}$$

8.5 Să se facă împărțirea în $Q[X]$ la polinomul $h(X) = X^2 - 2X + 1$ a polinoamelor $a(X) = X^5 - 4X^3 - 2X + 5$, $a(X) = X + 7$.

8.6 Aceeași problemă pentru polinoamele $h(X) = 2X - 1$ și respectiv $a(X) = X^3 - 3$.

- (1) În $Q[X]$;
- (2) În $Z_3[X]$.

8.7 Să realizeze un circuit care să efectueze (în Q) înmulțirea cu polinomul $g(X) = 1 - 2X + X^3$ și împărțirea cu polinomul $h(X) = g(X)$.

8.8 Aceeași problemă pentru polinoamele:

$$\begin{aligned}g(X) &= X - 2X^3 - X^4, & h(X) &= 2 + X + X^3 \quad \text{în } Z_5[X]; \\g(X) &= -1 + X - X^6, & h(X) &= X^2 - 4X^3 - 2X^5 + 3X^7 \quad \text{în } Z_{11}[X].\end{aligned}$$

8.9 Să se rezolve problema enunțată în Exemplul 18.2.

8.10 Să se genereze toate puterile lui α , rădăcină a polinomului

- (1) $1 + X^2 + X^5 \in Z_2[X]$
- (2) $1 + X + X^3 \in Z_3[X]$.

8.11 Folosind elementele din Exemplul 9.8, să se construiască circuite de generare pentru $a(\alpha)\alpha$, $a(\alpha)\alpha^2$ și $a(\alpha)\alpha^7$.

8.12 Să se construiască circuite de generare în $GF(2^5)$ pentru $a(\alpha)$ și $a(\alpha)\alpha^{10}$, unde α este rădăcină a polinomului $1 + X^2 + X^5 \in Z_2[X]$.

Capitolul 9

Coduri ciclice

9.1 Relații de recurență liniară

Teorema 9.1 Fie q un număr prim și $f(X) \in Z_q[X]$, $\text{grad}(f(X)) = n$. În algebra polinoamelor modulo $f(X)$, fie I un ideal și $g(X)$ polinomul de grad minim a cărui clasă de resturi aparține lui I : $\{g(X)\} \in I$. Atunci:

1. $\{s(X)\} \in I \iff g(X)|s(X)$;
2. $g(X)|f(X)$.

Demonstrație:

(1) Fie $\{s(X)\} \in I$. Folosind identitatea împărțirii, avem

$$s(X) = b(X)g(X) + r(X), \quad \text{grad}(r(X)) < \text{grad}(g(X))$$

Trecând la clasele de resturi respective, se obține $\{s(X)\} = \{b(X)\}\{g(X)\} + \{r(X)\}$. Cum $\{s(X)\} \in I$, rezultă $\{b(X)\}\{g(X)\} \in I$ (deoarece I este ideal), deci $\{r(X)\} \in I$, absurd, deoarece $g(X)$ este polinomul de grad minim a cărui clasă de resturi aparține idealului I . Singura variantă rămâne $r(X) \equiv 0$, adică $s(X) = b(X)g(X)$.

Reciproca este imediată: din $s(X) = b(X)g(X)$ rezultă $\{s(X)\} = \{b(X)\}\{g(X)\} \in I$.

(2) Prin împărțirea polinomului $f(X)$ cu $g(X)$ se obține

$$f(X) = c(X)g(X) + r(X) \quad \text{unde} \quad \text{grad}(r(X)) < \text{grad}(g(X)).$$

Avem $\mathbf{0} = \{f(X)\} = \{c(X)\}\{g(X)\} + \{r(X)\}$ de unde rezultă $\{r(X)\} \in I$, absurd; deci $r(X) \equiv 0$. q.e.d.

Teorema 9.2 În algebra polinoamelor modulo $f(X)$, pentru orice ideal I există un polinom normat unic $g(X)$ de grad minim, cu $\{g(X)\} \in I$.

Reciproc, pentru orice divizor normat al lui $f(X)$ există un ideal generat de acel divizor.

$g(X)$ se numește *generatorul* idealului I și scriem $I = (\{g(X)\})$.

Demonstrație: Fie $h(X)$ un polinom de grad minim cu $\{h(X)\} \in I$; vom nota $g(X) = h_n^{-1}h(X)$ unde h_n este coeficientul termenului de grad maxim din polinomul $h(X)$.

Să presupunem că $g(X)$, $g'(X)$ sunt două astfel de polinoame de grad minim, ale căror clase de resturi sunt în I . Conform Teoremei 20.1, $g(X)$ și $g'(X)$ se divid unul pe altul

și – având același grad – diferă printr-o constantă. Polinoamele fiind normate, această constantă este 1. Deci $g(X) = g'(X)$.

Reciproc, fie $g(X)$ un divizor normat al polinomului $f(X)$ și $(\{g(X)\})$ idealul generat de el. Un element al acestui ideal este $\{a(X)\} \in (\{g(X)\})$, deci are forma $\{a(X)\} = \{b(X)\}\{g(X)\} = \{b(X)g(X)\}$. q.e.d.

Teorema 9.3 Fie q un număr prim, $f(X) \in Z_q[X]$ un polinom normat, cu $\text{grad}(f(X)) = n$ și fie $f(X) = g(X)h(X)$. Dacă $\text{grad}(h(X)) = k$, atunci în algebra polinoamelor modulo $f(X)$, idealul $(\{g(X)\})$ are dimensiunea k .

Demonstrație: Să observăm că vectorii (asociați polinoamelor)

$$\{g(X)\}, \{Xg(X)\}, \dots, \{X^{k-1}g(X)\}$$

sunt liniar independenți. Într-adevăr, pentru orice k elemente $c_0, \dots, c_{k-1} \in Z_q$, nu toate nule, avem

$$c_0\{g(X)\} + c_1\{Xg(X)\} + \dots + c_{k-1}\{X^{k-1}g(X)\} = \{(c_0 + c_1X + \dots + c_{k-1}X^{k-1})g(X)\} \neq 0$$

deoarece s-a obținut un polinom de grad cel mult $n - k + k - 1 = n - 1 < n$.

În același timp, pentru orice $\{s(X)\} \in (\{g(X)\})$ avem $\{s(X)\} = \{a(X)g(X)\} = \{(a_0 + a_1X + \dots + a_{k-1}X^{k-1})g(X)\} = a_0\{g(X)\} + a_1\{Xg(X)\} + \dots + a_{k-1}\{X^{k-1}g(X)\}$, unde unii din coeficienții a_0, a_1, \dots, a_{k-1} pot fi nuli.

Deci vectorii de sus formează o bază a subspațiului liniar $(\{g(X)\})$, care are deci dimensiunea k . q.e.d.

Teorema 9.4 Fie $f(X) \in Z_q[X]$ un polinom normat și $f(X) = g(X)h(X)$. În algebra polinoamelor modulo $f(X)$,

$$\{a(X)\} \in (\{g(X)\}) \iff \{a(X)\} \text{ este în spațiul nul al idealului } (\{h(X)\}).$$

Demonstrație: " \implies ": Fie $\{a(X)\} \in (\{g(X)\})$, deci $a(X) = v(X)g(X)$.

Fie de asemenea $\{b(X)\} \in (\{h(X)\})$, deci $b(X) = w(X)h(X)$. Vom avea $\{a(X)\}\{b(X)\} = \{v(x)w(X)g(X)h(X)\} = \{v(X)w(X)f(X)\} = \{v(X)w(X)\}\{f(X)\} = \mathbf{0}$.

" \impliedby ": Să considerăm $\{a(X)\}$ în spațiul nul al idealului $(\{h(X)\})$. Atunci $\{a(X)\}\{h(X)\} = \mathbf{0}$, adică $a(X)h(X) = c(X)f(X) = c(X)g(X)h(X)$ de unde rezultă $a(X) = c(X)g(X)$. Deci $\{a(X)\} \in (\{g(X)\})$. q.e.d.

Definiția 9.1 Se numește relație de recurență liniară egalitatea

$$\sum_{j=0}^k h_j a_{i+j} = 0, \tag{1}$$

unde $i \geq 0$, $h_k = 1$, $h_0 \neq 0$, $h_j \in Z_q$, $a_{i+j} \in Z_q$.

Relațiile de recurență liniară se pot scrie și

$$a_{i+k} = - \sum_{j=0}^{k-1} h_j a_{i+j}, \quad i = 0, 1, \dots$$

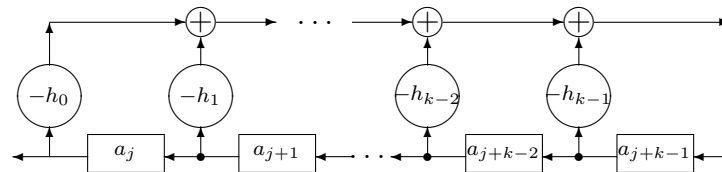
O soluție a unei astfel de relații de recurență liniară este orice succesiune infinită de forma $a_0, a_1, \dots, a_p, \dots$ care verifică relația dată, cu h_0, h_1, \dots, h_k fixate, $h_0 \neq 0, h_k = 1$. Relația va determina succesiv pe a_k din a_0, \dots, a_{k-1} , apoi pe a_{k+1} din a_1, \dots, a_k ș.a.m.d.

Altfel spus, ”condițiile inițiale” a_0, a_1, \dots, a_{k-1} determină o soluție a relației de recurență liniară.

Observația 9.1

- Orice combinație liniară de soluții ale unei relații de recurență liniară este tot o soluție.
- Soluțiile pentru care condițiile inițiale sunt respectiv $(1, 0, \dots, 0), (0, 1, \dots, 0), \dots, (0, 0, \dots, 1)$ determină orice altă soluție. Deci, spațiul soluțiilor relației de recurență (1) are dimensiunea cel mult k .

Fiind date condițiile inițiale (arbitrare) a_0, a_1, \dots, a_{k-1} , soluția relației de recurență liniară (1) corespunzătoare lor se poate obține folosind circuitul liniar



în care, la momentul inițial, în elementele de înmagazinare se găsesc a_0, a_1, \dots, a_{k-1} .

Să considerăm acum polinomul (fixat) $h(X) \in \mathbb{Z}_q[X]$, $h(X) = h_0 + h_1X + \dots + h_kX^k$, $h_0 \neq 0, h_k = 1$, și fie n cel mai mic număr natural pentru care $X^n - 1$ se divide cu $h(X)$. Notăm

$$g(X) = \frac{X^n - 1}{h(X)} \tag{2}$$

Pe baza acestor date construim relația de recurență liniară

$$a_{i+k} = - \sum_{j=0}^{k-1} h_j a_{i+j}, \quad i \geq 0 \tag{3}$$

Demonstrația existenței unui astfel de număr n se face folosind principiul cutiei. Găsim două numere n_1, n_2 ($n_1 > n_2$) pentru care polinoamele $X^{n_1} - 1$ și $X^{n_2} - 1$ dau același rest la împărțirea cu $h(X)$. Va rezulta că $X^{n_2} (X^{n_1-n_2} - 1)$ se divide cu $h(X)$. Cum $h(X)$ are termen liber nenul, el va divide pe $X^{n_1-n_2} - 1$.

Teorema 9.5

1. Soluțiile relației de recurență (3) sunt periodice, de perioadă n .
2. Mulțimea formată din prima perioadă a fiecărei soluții, scrisă ca polinom $a(X) = a_0X^{n-1} + \dots + a_{n-2}X + a_{n-1}$ constituie idealul $(\{g(X)\})$ în algebra polinoamelor modulo $X^n - 1$.

Demonstrație: (a) Vom arăta că oricărui element $\{a(X)\} \in (\{g(X)\})$ cu $a(X) = a_0X^{n-1} + \dots + a_{n-2}X + a_{n-1}$ îi corespunde o soluție periodică

$$a_1, a_1, \dots, a_{n-1}, a_0, a_1, \dots, a_{n-1}, a_0, \dots$$

a relației de recurență (3). Evident, nu este obligatoriu ca toți coeficienții a_i să fie nenuli, gradul real al polinomului $f(X)$ putând fi chiar zero.

Fie $\{a(X)\}\{h(X)\} = \{c(X)\}$. Vom nota $c(X) = c_0 + c_1X + \dots + c_{n-1}X^{n-1}$. Avem:

$$\text{- pentru } k \leq p \leq n-1, \quad c_p = h_0a_{n-1-p} + h_1a_{n-1-p+1} + \dots + h_ka_{n-1-p+k}; \quad (4)$$

$$\text{- pentru } 0 \leq p < k, \quad c_p = h_0a_{n-1-p} + h_1a_{n-1-p+1} + \dots + h_pa_{n-1} + h_{p+1}a_0 + \dots + h_ka_{k-p-1}. \quad (5)$$

Conform Teoremei 11.6, dacă $\{a(X)\} \in (\{g(X)\})$ atunci $\{a(X)\}\{h(X)\} = \mathbf{0}$, adică $c_p = 0$ ($0 \leq p \leq n-1$).

Considerând $(a_i)_i$ ca o succesiune periodică, vom avea evident $a_{i+n} = a_i$, $i = 0, 1, \dots$. Ținând cont de aceasta introducând $c_p = 0$ în (4) și (5) se obține (3).

(b) Idealul $(\{g(X)\})$ are dimensiunea $k = \text{grad}(h(X))$ (Teorema 20.5) și fiecare element al acestui ideal (polinom sau vector – după cum este scris) va da (conform cu (a)) o soluție a relației de recurență liniară (3). Dar spațiul soluțiilor relației (3) are dimensiunea cel mult k . Deci idealul $(\{g(X)\})$ va da toate soluțiile relației de recurență liniară (3). Mai trebuie arătat că perioada acestor soluții este chiar n .

Din cele de până acum a rezultat că perioada este cel mult n . Vom arăta că soluția corespunzătoare clasei de resturi $\{g(X)\}$ are perioada n .

Să presupunem prin absurd că soluția corespunzătoare lui $\{g(X)\}$ are perioada $m < n$. Dar n este divizibil cu m și deci coeficienții polinomului $g(X)$ – considerat ca fiind de grad $n-1$ (prin completare cu coeficienți nuli) formează $\frac{n}{m}$ blocuri care se repetă.

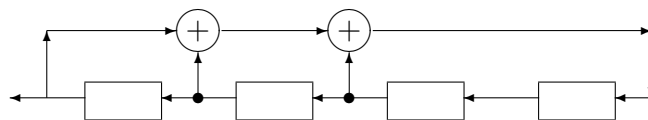
Deci $g(X) = q(X)(1 + X^m + X^{2m} + \dots + X^{n-m})$ unde $q(X)$ este un polinom de gradul $m-1$. Relația se poate scrie și $g(X) = q(X) \frac{X^n - 1}{X^m - 1}$.

Vom avea acum $(X^n - 1)(X^m - 1) = g(X)h(X)(X^m - 1) = q(X)h(X)(X^n - 1)$ adică $X^m - 1 = q(X)h(X)$, ceea ce contrazice condiția că n este cel mai mic număr pentru care $X^n - 1$ se divide cu $h(X)$. Deci $m = n$. q.e.d.

Exemplul 9.1 Să considerăm polinomul $h(X) = 1 + X + X^2 + X^4 \in \mathbb{Z}_2[X]$. Relația de recurență liniară asociată este

$$a_{i+4} = a_{i+2} + a_{i+1} + a_i, \quad (i \geq 0).$$

Circuitul liniar corespunzător are forma



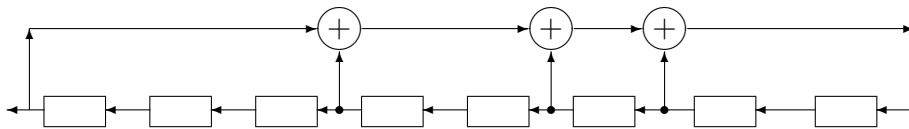
Cel mai mic n pentru care $X^n - 1$ se divide cu $h(X)$ este $n = 7$. Cum $g(X) = \frac{X^7 - 1}{h(X)} = X^3 + X + 1$, rezultă că circuitul va genera cuvintele idealului $(\{g(X)\})$. Fiecare cuvânt este caracterizat de cele patru valori binare inițiale din elementele de înmagazinare. Vor fi deci $2^4 = 16$ cuvinte de lungime 7. Ele corespund tuturor polinoamelor de grad maxim 6 din $\mathbb{Z}_2[X]$, care se divid cu $g(X)$.

De exemplu, pentru valorile inițiale $(1, 0, 1, 1)$, funcționarea circuitului timp de șapte tacți este:

Ieșire				
—	1	0	1	1
1	0	1	1	0
0	1	1	0	0
1	1	0	0	0
1	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	1	0	1	1

În elementele de înmagazinare se regăsesc valorile inițiale, iar la ieșire s-a obținut polinomul $f(X) = X^6 + X^4 + X^3 = X^3g(X)$.

Exemplul 9.2 Circuitul liniar corespunzător polinomului $h(X) = 1 + X^3 + X^5 + X^6 + X^8 \in Z_2[X]$ este



El dă soluțiile relației de recurență liniară $a_i + a_{i+3} + a_{i+5} + a_{i+6} + a_{i+8} = 0$, care formează idealul generat de polinomul $g(X) = \frac{X^{255} - 1}{h(X)}$ ideal compus din $2^8 = 256$ cuvinte de lungime 255.

9.2 Definirea codurilor ciclice

Fie n ($n \geq 2$) un număr natural. Să notăm cu A_n algebra polinoamelor din $Z_q[X]$, modulo $X^n - 1$. După cum s-a convenit, identificăm cuvântul $a_0a_1 \dots a_{n-1}$ cu vectorul $(a_0, a_1, \dots, a_{n-1})$ și cu polinomul $a(X) = a_0 + a_1X + \dots + a_{n-1}X^{n-1} \in Z_q[X]$.

În cadrul dualismului *vector - polinom* vom face o deosebire: anularea produsului a două polinoame nu înseamnă ortogonalitatea vectorilor corespunzători. Pentru această situație se folosește următoarea propoziție:

Propoziția 9.1 *Produsul a două polinoame este zero dacă și numai dacă toate produsele scalare dintre vectorul unui polinom și permutările ciclice ale vectorului celuilalt polinom sunt zero.*

Demonstrație: Fie $a(X), b(X) \in Z_q[X]$, $a(X) = \sum_{i=0}^{n-1} a_iX^i$, $b(X) = \sum_{i=0}^{n-1} b_iX^i$.

Atunci $\{c(X)\} = \{a(X)\}\{b(X)\} = \{c_0 + c_1X + \dots + c_{n-1}X^{n-1}\}$ unde

$$c_j = \sum_{i=0}^j a_i b_{j-i} + \sum_{i=j+1}^{n-1} a_i b_{j+n-i} = (a_0, a_1, \dots, a_{n-1}) \cdot (b_j, b_{j-1}, \dots, b_0, b_{n-1}, b_{n-2}, \dots, b_{j+1})^T$$

și Propoziția rezultă din faptul că $c_j = 0 \forall j$.

q.e.d.

Definiția 9.2 Un subspațiu liniar $V_n \subseteq A_n$ se numește ciclic dacă

$$(a_0, a_1, \dots, a_{n-1}) \in V_n \quad \implies \quad (a_{n-1}, a_0, \dots, a_{n-2}) \in V_n.$$

Teorema 9.6 Un subspațiu liniar $V_n \subseteq A_n$ este ciclic dacă și numai dacă este ideal.

Demonstrație: Înmulțirea cu clasa de resturi $\{X\}$ înseamnă de fapt permutarea ciclică a componentelor cu o unitate spre dreapta, pentru că:

$$\{X\}\{a_0 + a_1X + \dots + a_{n-1}X^{n-1}\} = \{a_{n-1} + a_0X + \dots + a_{n-2}X^{n-1}\}.$$

" \Leftarrow ": Dacă $V_n \subseteq A_n$ este ideal, atunci pentru orice $\mathbf{v} \in V_n$ avem $\mathbf{v}' = \{X\}\mathbf{v} \in V_n$ (s-a notat tot cu \mathbf{v} clasa de resturi modulo $X^n - 1$ corespunzătoare polinomului ai cărui coeficienți sunt componentele vectorului \mathbf{v}). Deci V_n este ciclic.

" \Rightarrow ": Presupunem că subspațiul liniar $V_n \subseteq A_n$ este ciclic și fie $\mathbf{v} \in V_n$. Atunci, din $\{X\}\mathbf{v} \in V_n$ rezultă $\{X^j\}\mathbf{v} \in V_n, \forall j = 1, \dots, n-1$.

Deci $\{c_0 + c_1X + \dots + c_{n-1}X^{n-1}\}\mathbf{v} \in V_n$, adică V_n este ideal în A_n . q.e.d.

Definiția 9.3 Se numește cod ciclic un ideal propriu $L \subset A_n$.

Codurile ciclice au fost introduse de Prange (1957), care a evidențiat bogăția de informație rezultată din structura lor algebrică.

9.3 Generarea codurilor ciclice

Pentru a construi un cod ciclic se folosește structura idealelor în algebra polinoamelor modulo $X^n - 1$. Fie I un ideal propriu în A_n și $g(X)$ polinomul normat de grad minim cu $\{g(X)\} \in I$. Atunci (Teorema 20.1) $\{a(X)\} \in I$ dacă și numai dacă $g(X)|a(X)$.

De asemenea, $g(X)|X^n - 1$. Oricărui divizor $g(X)$ al lui $X^n - 1$ ($\text{grad}(g(X)) < n$) îi corespunde un ideal propriu în A_n , generat de $g(X)$.

Pentru a defini un cod ciclic este suficient să dăm generatorul său $g(X)$, divizor al lui $X^n - 1$. Fie $g(X) = g_0 + g_1X + \dots + g_{n-k}X^{n-k}$ ($k < n$).

O bază a idealului ($\{g(X)\}$) se poate alege (Teorema 20.5)

$$\{g(X)\}, \{Xg(X)\}, \dots, \{X^{k-1}g(X)\}.$$

Matricea generatoare corespunzătoare acestui cod ciclic va fi

$$G_{k,n} = \begin{pmatrix} g_0 & g_1 & \dots & g_{n-k} & 0 & 0 & \dots & 0 \\ 0 & g_0 & \dots & g_{n-k-1} & g_{n-k} & 0 & \dots & 0 \\ & & & & \vdots & & & \\ 0 & 0 & \dots & 0 & g_0 & g_1 & \dots & g_{n-k} \end{pmatrix}$$

Rezultă că un cod ciclic poate fi organizat ca un (n, k) -cod liniar, unde n este gradul polinomului $X^n - 1$ iar k este gradul polinomului $h(X) = \frac{X^n - 1}{g(X)}$.

Exemplul 9.3 Codul cu repetiție este un cod ciclic al cărui polinom generator este $g(X) = 1 + X + X^2 + \dots + X^{n-1}$.

Exemplul 9.4 Să considerăm codul ciclic binar de lungime 7, cu polinomul generator $g(X) = 1 + X + X^3$ (vezi și Exemplul 9.1). El are matricea generatoare

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

Exemplul 9.5 Fie codul ciclic de lungime 6 peste Z_3 , de polinom generator $g(X) = 2 + X^2$. Matricea sa generatoare este

$$G = \begin{pmatrix} 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 1 \end{pmatrix}.$$

El este deci un $(6, 4)$ - cod ternar.

Sunt 14 coduri ciclice ternare de lungime 6, obținute din factorii polinomului $X^6 - 1 = (X + 1)^3(X + 2)^3$. Este un exercițiu interesant obținerea matricilor generatoare pentru toate aceste coduri.

Idealul generat de $\{g(X)\}$ este spațiul nul al idealului $\{h(X)\}$ unde $h(X) = \frac{X^n - 1}{g(X)}$. Acest ideal $\{h(X)\}$ se construiește luând ca bază polinoamele

$$\{h(X)\}, \{Xh(X)\}, \dots, \{X^{n-k-1}h(X)\}.$$

Ținând cont de Propoziția 9.1, putem construi matricea de control H a codului ciclic $\{g(X)\}$, luând drept linii ale matricii cele $n - k$ polinoame de sus, cu ordinea componentelor inversată.

Exemplul 9.6 Să reluăm codul din Exemplul 9.4. Deoarece $X^7 - 1 = (1 + X)(1 + X + X^3)(1 + X^2 + X^3)$, pentru acest cod, avem $h(X) = (1 + X)(1 + X + X^3) = 1 + X^2 + X^3 + X^4$.

Matricea generatoare a codului $\{h(X)\}$ este

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

deci matricea de control asociată codului din Exemplul 9.4 va fi

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

Deoarece coloanele sale sunt nenule și distincte două câte două, codul astfel construit este echivalent cu un $(7, 4)$ cod Hamming binar.

Pentru a obține un cod ciclic sistematic, este convenabil să alegem o altă bază pentru idealul $\{g(X)\}$. Să observăm că pentru $i = n - k, n - k + 1, \dots, n - 1$, putem scrie

$$X^i = q_i(X)g(X) + r_i(X) \text{ cu } \text{grad}(r_i(X)) < \text{grad}(g(X)) = n - k.$$

Deci $\{X^i - r_i(X)\} \in \{g(X)\}$, $i = n - k, n - k + 1, \dots, n - 1$.

Această mulțime de vectori este liniar independentă și conduce la o matrice generatoare de forma

$$G = [-R \ I]$$

unde linia j din matricea R este vectorul coeficienților lui $r_j(X)$ pentru $j = n-k, \dots, n-1$. Codul obținut este deci sistematic, iar matricea sa de control se scrie imediat:

$$H = [I \ R^T].$$

De remarcat că matricea H^T are pe linii componentele resturilor $r_i(X)$ pentru $i = 0, 1, \dots, n-1$.

Exemplul 9.7 Să luăm din nou $X^7 - 1 = g(X)h(X)$ peste Z_2 , unde $g(X) = 1 + X^2 + X^3$, $h(X) = 1 + X^2 + X^3 + X^4$ (deci $n = 7$, $k = 4$). Avem

$$\begin{aligned} X^0 &= && 0g(X) &+& 1 \\ X^1 &= && 0g(X) && + X \\ X^2 &= && 0g(X) && + X^2 \\ X^3 &= && g(X) &+& 1 && + X^2 \\ X^4 &= && (1+X)g(X) &+& 1 &+& X &+& X^2 \\ X^5 &= && (1+X+X^2)g(X) &+& 1 &+& X \\ X^6 &= && (X+X^2+X^3)g(X) && + X &+& X^2 \end{aligned}$$

Baza corespunzătoare idealului ($\{g(X)\}$) este

$$\{1 + X^2 + X^3\}, \{1 + X + X^2 + X^4\}, \{1 + X + X^5\}, \{X + X^2 + X^6\}$$

care conduce la matricea generatoare a unui cod sistematic:

$$G_{4,7} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} = [-R_{4,3} \ I_4].$$

Matricea de control corespunzătoare se scrie simplu:

$$H_{3,7} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} = [I_3 \ R_{3,4}^T].$$

9.4 Implementarea generării codurilor ciclice

Codurile ciclice sunt coduri a căror implementare este mult facilitată de circuitele liniare. Cu ajutorul acestora se poate realiza automat codificarea, calculul sindromului, detectarea și corectarea erorilor.

În construcția practică vom distinge două cazuri:

9.4.1 Circuit cu k elemente de înmagazinare

Este o metodă de generare a codurilor ciclice, avantajoasă dacă sunt mai puține simboluri de informație decât simboluri de control: $k < n - k$.

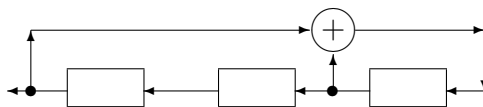
Fie codul ciclic $(\{g(X)\}) \subset A_n$ și $h(X) = \frac{X^n - 1}{g(X)} = h_0 + h_1X + \dots + h_kX^k$ cu $h_0 \neq 0, h_k = 1$. Conform Teoremei 9.5, idealul $(\{g(X)\})$ este generat de circuitul liniar care construiește soluțiile relației de recurență liniară

$$\sum_{j=0}^k h_j a_{i+j} = 0, \quad i = 0, 1, \dots$$

Mesajul de informație care trebuie codificat – conținând k simboluri pe fiecare bloc – se introduce la momentul inițial în elementele de înmagazinare ale circuitului, sub formă de ”condiții inițiale”. Lăsând circuitul să funcționeze, obținem după n momente cuvântul -cod corespunzător, aparținând idealului $(\{g(X)\})$ și având pe primele k poziții elementele de informație.

Exemplul 9.8 *Circuitul liniar construit în Exemplul 9.1 este un circuit de codificare pentru codul generat de polinomul $g(X) = 1 + X + X^3$. Exemplul descrie și un mod de funcționare pentru cuvântul de informație 1011.*

Dacă s-ar lua drept polinom generator $1 + X^2 + X^3 + X^4$, idealul generat de el este dat de circuitul liniar



deoarece $h(X) = \frac{X^7 - 1}{1 + X^2 + X^3 + X^4} = 1 + X^2 + X^3$.

9.4.2 Circuit cu $n - k$ elemente de înmagazinare

Este o strategie avantajos de utilizat în cazul $n - k < k$.

Dacă interpretăm mesajul de informație ca un polinom de gradul $k - 1$, atunci codificarea se poate face utilizând un circuit de înmulțire cu polinomul generator $g(X)$ (vezi Prelegerea anterioară). La decodificare se recapătă mesajul inițial (dacă nu au apărut erori) folosind un circuit de împărțire cu $g(X)$.

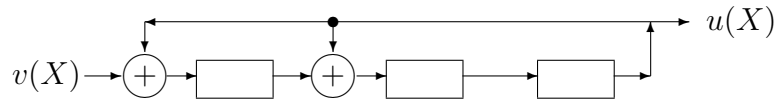
Exemplul 9.9 *Codul ciclic de polinom generator $g(X) = 1 + X + X^3$ poate codifica mesajele de informație folosind circuitul liniar:*



Astfel, mesajul de informație $\mathbf{u} = 1001$ se codifică în $\mathbf{v} = 1100101$, conform calculelor:

$$v(X) = u(X)g(X) = (1 + X^3)(1 + X + X^3) = 1 + X + X^4 + X^6$$

Pentru decodificare se folosește circuitul

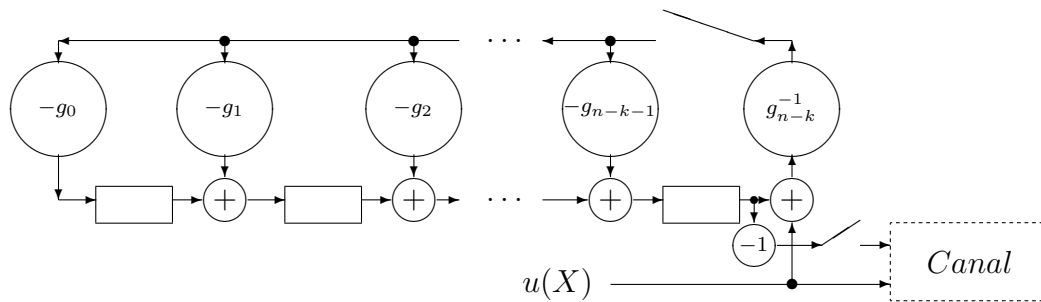


Dezavantajul unei asemenea metode îl constituie faptul că – nefiind un cod sistematic – prin codificare pozițiile de informație se pierd. Pentru a obține un cod sistematic, procedăm în felul următor: mesajul de informație este considerat un polinom $u_0(X)$ de gradul $n - 1$ având pozițiile de informație drept coeficienții lui X^{n-k}, \dots, X^{n-1} , restul coeficienților fiind nuli. Atunci avem $u_0(X) = q(X)g(X) + r(X)$ cu $\text{grad}(r(X)) < \text{grad}(g(X)) = n - k$, de unde

$$\{u_0(X) - r(X)\} \in (\{g(X)\}).$$

$\{u_0(X) - r(X)\}$ reprezintă un cuvânt - cod în care coeficienții lui X^{n-k}, \dots, X^{n-1} sunt pozițiile de informație nealterate, iar coeficienții lui $r(X)$ cu semnul schimbat (deci coeficienții lui $X^0, X^1, \dots, X^{n-k-1}$) sunt caracterele de control.

Circuitul care realizează această codificare este următorul:

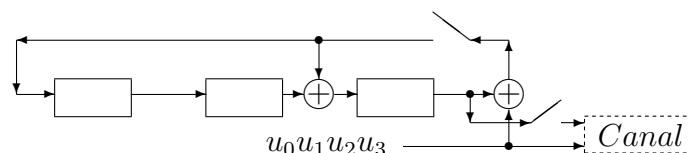


El funcționează astfel:

- Inițial cele $n - k$ elemente de înmagazinare conțin 0, întrerupătorul de ieșire este decuplat iar cel de feedback - cuplat.
- Se introduc cele k elemente ale mesajului de informație atât în circuit cât și în canalul de comunicație. După k momente, în elementele de înmagazinare avem coeficienții restului $r(X)$.
- Se decuplează feedbackul și se cuplează circuitul la canalul de comunicație.
- Coeficienții restului – cu semn schimbat – se transmit în canal imediat după pozițiile de informație.

Deoarece provine din circuitul de împărțire cu $g(X)$, același circuit poate fi folosit și pentru detectarea erorilor la recepție. Pentru aceasta, după decuplarea întrerupătorului de ieșire și cuplarea celui de feedback, se introduce în circuit cuvântul recepționat. Dacă la momentul n restul nu este nul (cel puțin un element de înmagazinare conține un element nenul) înseamnă că a apărut cel puțin o eroare în transmisie.

Exemplul 9.10 Codul ciclic binar de lungime 7 generat de $g(X) = 1 + X^2 + X^3$, poate fi construit folosind circuitul liniar:



9.5 Exerciții

9.1 Să se determine toate codurile ciclice de lungimi $n = 5$ și $n = 6$

(a) peste Z_2 ; (b) peste Z_3 .

9.2 Construiți o bază pentru cel mai mic cod ciclic de lungime n care conține cuvântul:

(a) $\mathbf{v} = 1101000$, $n = 7$;

(b) $\mathbf{v} = 010101$, $n = 6$;

(c) $\mathbf{v} = 11011000$, $n = 8$.

9.3 Pentru fiecare din cuvintele de mai jos, găsiți polinomul generator al celui mai mic cod ciclic care conține cuvântul respectiv:

010010

01100110

0101100

001000101110000

000010010000000

010111010000000

9.4 Să se afle polinomul generator al codului ciclic C știind o bază S a sa:

$S = \{010, 011, 111\}$;

$S = \{1010, 0101, 1111\}$;

$S = \{0101, 1010, 1100\}$;

$S = \{1000, 0100, 0010, 0001\}$;

$S = \{11000, 01111, 11110, 01010\}$.

9.5 Într-o codificare sistematică, codificați mesajul de informație 1101 într-un cu-vânt - cod al unui cod binar ciclic de lungime 7 cu polinomul generator $g(X) = 1 + X^2 + X^3$.

9.6 Pentru codul definit mai sus codificați mesajele de informație date de polinoamele: $1 + X^2$, X , $X + X^2 + X^3$.

Fiind date cuvintele - cod $X^2 + X^4 + X^5$, $1 + X + X^2 + X^4$, $X^2 + X^3 + X^4 + X^6$, găsiți mesajele de informație corepunzătoare.

9.7 Construiți circuite liniare pentru codificare și decodificare ale codului ciclic binar de lungime 7 generat de polinomul $g(X) = (1 + X^2 + X^3)(1 + X)$.

9.8 Care este lungimea minimă a unui cod ciclic binar de polinom generator $g(X) = 1 + X^2 + X^3 + X^4$?

Construiți circuite liniare pentru codificarea și decodificarea sa.

9.9 Construiți un circuit liniar pentru codificarea (15, 11) - codului Hamming binar.

9.10 Construiți matricea de control a codului ciclic binar de lungime n și polinom generator $g(X)$:

$n = 6$, $g(X) = 1 + X^2$;

$n = 8$, $g(X) = 1 + X^2$;

$n = 9$, $g(X) = 1 + X^3 + X^6$;

$n = 15$, $g(X) = 1 + X + X^4$ (generează codul Hamming);

$n = 23$, $g(X) = 1 + X + X^5 + X^6 + X^7 + X^9 + X^{11}$ (generează codul Golay);

$n = 15$, $g(X) = 1 + X^4 + X^6 + X^7 + X^8$.

9.11 Sunt ciclice codurile liniare binare definite de matricile generatoare definite mai jos ?

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

9.12 Arătați că dacă polinomul generator al unui cod ciclic binar se divide cu polinomul $1 + X$, atunci toate cuvintele sale au pondere pară.

Reciproca este adevărată ?

9.13 Dați o condiție necesară și suficientă pentru polinomul generator al unui cod ciclic de lungime impară pentru ca $11\dots 1$ să fie cuvânt - cod.

9.14 Arătați că dacă $g(X) \in Z_q[X]$ generează un (n, k) - cod ciclic, atunci $g(X^p)$ generează un (pn, pk) - cod ciclic.

Descrieți codurile ciclice binare pentru $g(X) = 1 + X$ și $g(X) = 1 + X + X^3$ ($n = 7$).

9.15 Arătați că intersecția a două coduri ciclice de aceeași lungime este tot un cod ciclic. Care este polinomul său generator ?

Capitolul 10

Decodificarea codurilor ciclice

10.1 Detectarea și corectarea erorilor independente

După cum se știe de la codurile liniare, o modalitate de caracterizare a tipurilor de erori care pot apare în transmisia mesajelor este dată pe baza sindromului. În cazul codurilor ciclice este convenabil să lucrăm cu un *sindrom polinomial*.

Definiția 10.1 Fie C un cod ciclic de lungime n , cu polinomul generator $g(X)$. Se numește "sindrom polinomial" $s(X)$ al cuvântului \mathbf{a} de lungime n , restul împărțirii polinomului corespunzător $a(X)$ la $g(X)$.

Deci, dacă se transmite cuvântul - cod $q(X)g(X)$ și se primește $a(X)$, sindromul va fi

$$s(X) = a(X) - q(X)g(X), \quad \text{grad}(s(X)) < \text{grad}(g(X)).$$

Observația 10.1 Fie \mathbf{e} vectorul eroare apărut la transmisie.

- Deoarece $a(X) - e(X)$ este divizibil cu $g(X)$, rezultă că resturile împărțirii lui $a(X)$ și $e(X)$ la $g(X)$ coincid. Deci se poate vorbi despre "tipul de eroare" \mathbf{e} .
- Similar cu raționamentul folosit la codurile liniare, vor fi considerate sindromuri toate polinoamele din $Z_q[X]$ de grad $< n - k$; pentru fiecare sindrom $s(X)$ putem alege o eroare - tip $e(X)$ de pondere minimă care corespunde aceluși sindrom.

Structura algebrică a codurilor ciclice permite construcția de metode de decodificare (cu corectarea posibilelor erori aferente) mai eficiente decât algoritmi similari de la codurile liniare.

Vom începe cu prezentarea unui exemplu.

Exemplul 10.1 Fie C codul Hamming de lungime 7, care are polinomul generator $g(X) = 1 + X + X^3$ (Prelegerea 9, Exemplele 9.4 și 9.6).

Dacă se recepționează cuvântul $\mathbf{a} = 1011001$, sindromul lui este restul împărțirii lui $1 + X^2 + X^3 + X^6$ la $1 + X + X^3$, adică $s(X) = 1 + X$.

Se știe că orice cod Hamming corectează o eroare, deci va fi util să avem un tabel cu sindromurile tuturor erorilor - tip $e(X) = X^i$ ($0 \leq i \leq 6$).

<i>Eroare - tip</i>	<i>Sindrom</i>
0	0
1	1
X	X
X^2	X^2
X^3	$1 + X$
X^4	$X + X^2$
X^5	$1 + X + X^2$
X^6	$1 + X^2$

Pentru că $X + 1$ este sindromul lui $e(X) = X^3$, tragem concluzia că a fost perturbat al treilea bit și vom decodifica

$$a(X) - e(X) = 1011001 - 0001000 = 1010001.$$

O dificultate în această metodă de decodificare bazată pe sindrom constă în necesitatea de a lista toate erorile - tip corespunzătoare sindromurilor, și de a le parcurge la recepția fiecărui cuvânt. O simplificare este însă propusă de Meggitt (1960), bazată pe structura ciclică a codurilor. Vom face corecția numai pentru ultimul caracter (coeficientul termenului de grad maxim) al cuvântului \mathbf{a} recepționat. După aceea se efectuează o permutare ciclică și se studiază din nou ultimul caracter.

În acest fel, după n permutări ciclice, toate pozițiile au fost corectate.

Metoda prezintă două avantaje majore:

1. Se folosește doar lista erorilor - tip de grad $n - 1$. (Astfel, în Exemplul 20.3 o singură eroare-tip are gradul 6).
2. Calculul sindromului se efectuează o singură dată – la început – folosind circuitul liniar de împărțire cu $g(X)$. După ce a funcționat n momente, în elementele de înmagazinare se găsește sindromul. După aceea se neglijează ieșirea și se lasă să funcționeze circuitul. La fiecare pas (Prelegerea 8, Figura 8.4) sindromul se înmulțește cu X , această operație având ca efect permutarea sa ciclică cu o poziție.

Propoziția 10.1 *Dacă un cuvânt $\mathbf{a} = (a_0, a_1, \dots, a_{n-1})$ are sindromul $s(X)$, atunci permutarea sa ciclică are sindromul $s'(X)$, care este restul împărțirii lui $Xs(X)$ la polinomul generator $g(X)$.*

Demonstrație: Sindromul $s(X)$ este definit prin identitatea împărțirii

$$a(X) = q(X)g(X) + s(X)$$

unde $q(X)$ este câtul împărțirii lui $a(X)$ la $g(X)$. Deoarece se lucrează în algebra polinoamelor modulo $X^n - 1$, permutarea ciclică a lui $\{a(X)\}$ este $\{a'(X)\} = \{Xa(X)\}$, construit astfel:

$$\begin{aligned} a'(X) &= Xa(X) - a_{n-1}X^n + a_{n-1} = Xa(X) - a_{n-1}(X^n - 1) = \\ &= Xa(X) - a_{n-1}g(X)h(X) = Xq(X)g(X) + Xs(X) - a_{n-1}g(X)h(X) = \\ &= Xs(X) + g(X)[Xq(X) - a_{n-1}h(X)]. \end{aligned}$$

Rezultă de aici că resturile împărțirii lui $a'(X)$ și $Xs(X)$ la polinomul generator $g(X)$ sunt aceleași. q.e.d.

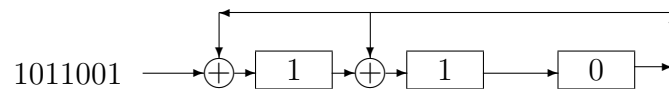
Se poate da acum

Algoritmul de decodificare Meggitt pentru (n, k) - codurile ciclice binare:

1. Se listează toate sindromurile corespunzătoare erorilor-tip \mathbf{e} reprezentate prin polinoame de grad $n - 1$;
 2. Cuvântul recepționat \mathbf{a} este introdus în circuitul liniar de împărțire cu $g(X)$, care va funcționa n tacti;
 3. Dacă în elementele de înmagazinare se obține un polinom care se regăsește în lista de sindromuri, se modifică cel mai din dreapta bit al cuvântului primit;
 4. Se permută ciclic cuvântul primit și – în același timp – se lasă să funcționeze un tact circuitul liniar (ignorând ieșirea), după care se reia Pasul (3).
- Se repetă acest procedeu de $n - 1$ ori.

Exemplul 10.2 *Revoluând codul Hamming din Exemplul 20.3, algoritmul va lucra astfel:*

- Singurul sindrom din listă este $1 + X^2$ (de vector 101).
- Dacă se recepționează de exemplu $\mathbf{a} = 1011001$, el se introduce în circuitul liniar



După 7 tacti, în elementele de înmagazinare se obține sindromul 110.

- 110 este diferit de 101 (singurul sindrom din listă), deci a_6 a fost recepționat corect.
- Se permută circular \mathbf{a} în 1101100 și se lasă circuitul să funcționeze un tact; elementele de înmagazinare vor conține acum 011.
- Nici acest sindrom nu este pe listă, deci a_5 a fost și el corect ș.a.m.d.

Pașii de lucru ai algoritmului sunt reprezentați în tabloul:

Pas	Bit	Sindrom
7	a_6	110
8	a_5	011
9	a_4	111
10	a_3	101
11	a_2	100
12	a_1	010
13	a_0	001

Deci singurul bit corectat (unde se transformă 1 în 0) este a_3 . Mesajul primit este 1010001.

Exemplul 10.3 *Să considerăm codul ciclic binar generat de polinomul $g(X) = 1 + X^4 + X^6 + X^7 + X^8$. El este un $(15, 7)$ - cod ciclic de distanță minimă $d = 5$ (după cum vom vedea mai târziu) deci poate corecta cel mult 2 erori independente. Lista completă a sindromurilor (deci a tuturor erorilor - tip de pondere 0, 1, 2) are $C_{15}^0 + C_{15}^1 + C_{15}^2 = 121$ elemente. Folosind algoritmul Meggitt, numărul lor se reduce la 15, anume*

Tip eroare	Sindrom
X^{14}	$X^7 + X^6 + X^5 + X^3$
$1 + X^{14}$	$X^7 + X^6 + X^5 + X^3 + 1$
$X + X^{14}$	$X^7 + X^6 + X^5 + X$
$X^2 + X^{14}$	$X^7 + X^6 + X^5 + X^3 + X^2$
$X^3 + X^{14}$	$X^7 + X^6 + X^5$
$X^4 + X^{14}$	$X^7 + X^6 + X^5 + X^4 + X^3$
$X^5 + X^{14}$	$X^7 + X^6 + X^3$
$X^6 + X^{14}$	$X^7 + X^5 + X^3$
$X^7 + X^{14}$	$X^6 + X^5 + X^3$
$X^8 + X^{14}$	$X^5 + X^4 + X^3 + 1$
$X^9 + X^{14}$	$X^7 + X^4 + X^3 + X + 1$
$X^{10} + X^{14}$	$X^3 + X^2 + X$
$X^{11} + X^{14}$	$X^7 + X^6 + X^5 + X^4 + X^2 + X$
$X^{12} + X^{14}$	$X^7 + X^6 + X^4 + X$
$X^{13} + X^{14}$	$X^7 + X^4 + X^3 + X^2$

Deci, la recepția unui cuvânt \mathbf{a} de lungime 15, vom calcula sindromul și – dacă acesta se află în tabelul de sus – vom modifica bitul a_{14} . Apoi se face o permutare ciclică și se corectează a_{13} , ș.a.m.d.

De remarcat că în Exemplul 14.4 decodificarea nu este completă: algoritmul Meggitt corectează în total 121 erori - tip. Privit însă ca un cod liniar tabela de decodificare va avea $2^{15}/2^7 = 256$ linii, ceea ce înseamnă că acest cod are posibilitatea de a corecta mult mai multe erori. Cu algoritmul Meggitt se vor corecta toate erorile simple sau duble, ignorând complet posibilitatea existenței altor tipuri de erori.

Vom încerca în continuare să îmbunătățim această situație.

10.2 Pachete standard de erori

Fie $C \subseteq A_n$ un (n, k) - cod ciclic peste Z_q , capabil să corecteze t erori.

Definiția 10.2 Pentru un polinom $e(X) = X^s e_0(X) \in Z_q[X]$ cu $e_0(0) = 1$, spunem că "lungimea-pachet" a lui \mathbf{e} este $1 + \text{grad}(e_0(X))$.

Cuvântul $\mathbf{e} \in Z_q^n$ este un "pachet standard de lungime j " dacă gradul minim al polinoamelor $\{X^i e(X)\}$ ($0 \leq i \leq n-1$) este $j-1$.

Dacă se trimite un cuvânt \mathbf{v} și se recepționează \mathbf{a} , spunem că erorile au afectat j poziții consecutive dacă eroarea - tip $\mathbf{e} = \mathbf{a} - \mathbf{v}$ este un pachet standard de lungime j .

Exemplul 10.4 Fie $n = 7$ și $\mathbf{e} = 0101100$. Atunci $e(X) = X + X^3 + X^4 = X(1 + X^2 + X^3)$. Efectuând toate permutările sale ciclice, se obține $\{X^6 e(X)\} = \{1 + X^2 + X^3\}$ - corespunzător unui polinom de gradul 3; toate celelalte permutări conduc la polinoame de grad mai mic. Deci \mathbf{e} este un pachet standard de lungime 4.

Dacă se ia $\mathbf{e} = 1000100$, polinomul asociat $(1 + X^4)$ are gradul 4; efectuând permutările sale ciclice, se obține $\{X^3(1 + X^4)\} = \{1 + X^3\}$, de gradul 3, deci și acest \mathbf{e} este un pachet standard de lungime 4.

Reamintim că la codurile liniare se construia un tablou standard în care pe fiecare linie se aflau toate cuvintele cu același sindrom. Codul era corector de t erori independente dacă toate cuvintele din Z_q^n de pondere cel mult t erau așezate pe linii diferite; aceste cuvinte constituiau reprezentanții sindromurilor și apăreau pe prima coloană, fiind considerate erori - tip pentru fiecare linie.

Astfel de tabele se pot construi și pentru codurile ciclice; aici se iau ca reprezentanți în fiecare linie, pachetele standard de erori de lungime minimă. În acest fel, un cod liniar corector de t erori (independente) este corector de pachete de j erori consecutive dacă toate cuvintele de lungime - pachet cel mult j sunt reprezentanți.

Definiția 10.3 Un (n, k) - cod ciclic corectează pachetele de j erori consecutive dacă orice pachet standard de lungime cel mult j este selectat ca sindrom.

Lema 10.1 Dacă un cod C este corector de t erori independente și corector de pachete de j erori consecutive, atunci $t \leq j$.

Demonstrație: Exercițiu. q.e.d.

Exemplul 10.5 Să considerăm toate pachetele standard nenule de lungime cel mult 3 în Z_2^{15} . Fiecare astfel de cuvânt este de forma

$$\{e(X)\} = \{X^i e_0(X)\} \text{ cu } 0 \leq i \leq 14 \text{ și } e_0(X) \in \{1, 1 + X, 1 + X^2, 1 + X + X^2\}.$$

În total sunt $15 \cdot 4 = 60$ astfel de pachete de erori.

Exemplul 10.6 Fie $g(X) = 1 + X + X^2 + X^3 + X^6$ polinomul generator al unui $(15, 9)$ - cod ciclic binar. El nu este un cod corector de 3 erori independente pentru că sunt 576 tipuri de erori de pondere cel mult 3 și numai $2^{15}/2^9 = 64$ sindromuri. În schimb sunt numai 61 pachete standard de lungime maxim 3 (Exemplul 18.2, la care se adaugă pachetul nul); deci acest cod poate corecta orice pachet de maxim 3 erori consecutive. Faptul că acesta este un cod ciclic corector de pachete de 3 erori consecutive rezultă din calculul sindromurilor polinoamelor $\{X^i e_0(X)\}$ unde $0 \leq i \leq 14$ și $e_0(X) \in \{1, 1 + X, 1 + X^2, 1 + X + X^2\}$.

10.2.1 Detectarea pachetelor de erori

Teorema 10.1 Un (n, k) - cod ciclic detectează orice pachet de maxim $n - k$ erori.

Demonstrație: Dacă $\{e(X)\}$ este o eroare - tip de lungime cel mult $n - k$, atunci $e(X) = X^i e_0(X)$ cu $\text{grad}(e_0(X)) < n - k$. Fie $g(X)$ polinomul generator al codului, cu $\text{grad}(g(X)) = n - k$.

Știm că $\{e(X)\}$ este cuvânt - cod $\iff g(X) | e(X)$.

Cum $g(X) | X^n - 1$, el va fi prim cu X^i ; deci $\{e(X)\}$ este cuvânt - cod dacă și numai dacă $g(X) | e_0(X)$, absurd deoarece $\text{grad}(e_0(X)) < n - k = \text{grad}(g(X))$. q.e.d.

Teorema 10.2 Proporția pachetelor standard de $j > n - k$ erori pe care nu le poate detecta un (n, k) - cod ciclic este

$$\begin{cases} \frac{q^{-(n-k-1)}}{q-1} & \text{dacă } j = n - k + 1, \\ q^{-(n-k)} & \text{dacă } j > n - k + 1 \end{cases}$$

Demonstrație: Fie $\{e(X)\} = \{X^s e_0(X)\}$ unde $\text{grad}(e_0(X)) = j - 1$. Să numărăm câte asemenea pachete standard de erori sunt posibile. Ca prim și ultim coeficient al lui $e_0(X)$ poate fi orice element din Z_q^* (în număr de $q - 1$), iar coeficienții intermediari pot fi orice elemente din Z_q (în număr de q). Deci sunt $(q - 1)^2 q^{j-2}$ pachete standard de erori care încep și se termină în aceeași poziție.

Un pachet de erori $\{e(X)\}$ nu este detectat de un (n, k) - cod ciclic dacă și numai dacă $\{e(X)\}$ este cuvânt - cod, adică $e_0(X) = g(X)h(X)$, unde $\text{grad}(h(X)) = \text{grad}(e_0(X)) - \text{grad}(g(X)) = j - 1 - (n - k)$. Deci polinomul $h(X)$ are $j - n + k$ coeficienți.

Apar două situații:

1. $\text{grad}(h(X)) = 0$, adică $j = n - k + 1$. Atunci $h(X)$ se reduce la o constantă și există $q - 1$ asemenea polinoame $h(X)$. Deci, raportul dintre numărul pachetelor standard de lungime j nedetectabile și numărul total al pachetelor standard de lungime j care pot fi localizate este

$$\frac{q - 1}{(q - 1)^2 q^{j-2}} = \frac{q^{-(n-k-1)}}{q - 1}.$$

2. Dacă $\text{grad}(h(X)) > 0$, adică $j > n - k + 1$, vom avea $(q - 1)^2 q^{j-n+k-2}$ polinoame $h(X)$ posibile (pachete standard de lungime j) nedetectabile. Rezultă că raportul de sus este în acest caz

$$\frac{(q - 1) q^{j-n+k-2}}{(q - 1)^2 q^{j-2}} = q^{-(n-k)}.$$

q.e.d.

Exemplul 10.7 Codul ciclic definit în Exemplul 18.7 detectează orice pachet standard de maxim 6 erori.

Dintre pachetele standard de 7 erori posibile care pot apare, doar $1/2^5$ nu pot fi detectate, iar dintre pachetele standard de 8 - 15 erori nu pot fi detectate $1/2^6$.

10.2.2 Corectarea pachetelor de erori

Algoritmul de corectare a pachetelor de erori este o variantă a Algoritmului Meggitt.

Fie $C = (\{g(X)\})$ un (n, k) - cod ciclic binar corector de pachete de maxim j erori consecutive, și $\mathbf{a} \in Z_q^n$ un cuvânt recepționat.

1. Se calculează sindromul $\{s(X)\}$;
2. Pentru fiecare $i \geq 0$ se calculează $s_i(X) = X^i s(X) \text{ mod } g(X)$, până se ajunge la un p cu $\text{grad}(s_p(X)) < j - 1$.
Atunci eroarea standard este $\{e(X)\} = \{X^{n-p} s_p(X)\}$ și se generează cuvântul - cod $\mathbf{a} + \mathbf{e}$.

Exemplul 10.8 Conform Exemplului 18.7, polinomul $g(X) = 1 + X + X^2 + X^3 + X^6$ generează un $(15, 9)$ - cod ciclic binar corector de pachete standard de cel mult 3 erori. Să folosim algoritmul Meggitt pentru a decodifica cuvântul $\mathbf{a} = 111100100001010$.

$$\begin{aligned}
s(X) &= 1 + X + X^2 + X^3 + X^6 + X^{11} + X^{13} \pmod{g(X)} = 1 + X^3 + X^4 + X^5. \\
s_1(X) &= Xs(X) \pmod{g(X)} = 1 + X^2 + X^3 + X^4 + X^5, \\
s_2(X) &= X^2s(X) \pmod{g(X)} = 1 + X^2 + X^4 + X^5, \\
s_3(X) &= X^3s(X) \pmod{g(X)} = 1 + X^2 + X^5, \\
s_4(X) &= X^4s(X) \pmod{g(X)} = 1 + X^2.
\end{aligned}$$

S-a ajuns la grad($s_4(X)$) = 2 ≤ 3 - 1. Deci eroarea - tip este
 $\{e(X)\} = \{X^{15-4}s_4(X)\} = X^{11} + X^{13}.$

Cuvântul - cod transmis a fost

$$\mathbf{v} = \mathbf{a} + \mathbf{e} = 111100100001010 + 000000000001010 = 111100100000000.$$

10.3 Transpunere

O metodă eficientă de utilizare a capacității de corectare a pachetelor standard de erori de către codurile ciclice binare este utilizarea de transpuneri (*interleaving*).

În mod natural, mesajele de informație $\mathbf{m}_1, \mathbf{m}_2, \dots$ sunt codificate în cuvintele - cod $\mathbf{c}_1, \mathbf{c}_2, \dots$ și transmise prin canal în ordinea codificării. Transmisia se poate realiza însă și după o rearanjare a caracterelor de informație din mai multe cuvinte - cod consecutive.

Formal, prin *transpunere la adâncimea s* se înțelege scrierea a s cuvinte - cod $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_s$ ca linii ale unei matrici

$$X_{s,n} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & \dots & c_{1n} \\ c_{21} & c_{22} & c_{23} & \dots & c_{2n} \\ & & \vdots & & \\ c_{s1} & c_{s2} & c_{s3} & \dots & c_{sn} \end{pmatrix}$$

și transmiterea a n mesaje de lungime s , formate din coloanele matricii X_{sn} :

$$c_{11}c_{21} \dots c_{s1}, \quad c_{12}c_{22} \dots c_{s2}, \quad \dots, \quad c_{1n}c_{2n} \dots c_{sn}$$

Exemplul 10.9 Să considerăm codul liniar generat de matricea $G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$

și șase mesaje codificate $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_6$, unde

$$\begin{aligned}
\mathbf{c}_1 &= 100110, & \mathbf{c}_2 &= 010101, & \mathbf{c}_3 &= 111000, \\
\mathbf{c}_4 &= 010101, & \mathbf{c}_5 &= 100110, & \mathbf{c}_6 &= 111000.
\end{aligned}$$

Deci - la nivel de caracter - secvența va arăta:

$$100110 \quad 010101 \quad 111000 \quad 010101 \quad 100110 \quad 111000.$$

Printr-o transpunere la adâncimea 3, forma mesajelor este:

$$101 \quad 011 \quad 001 \quad 110 \quad 100 \quad 010 \quad 011 \quad 101 \quad 001 \quad 110 \quad 010 \quad 100$$

iar cu o transpunere la adâncimea 6:

$$101011 \quad 011101 \quad 001001 \quad 110110 \quad 100010 \quad 010100.$$

Ce efect are o transpunere la adâncimea s asupra capacității de corecție de pachete standard de erori ale codului? Răspunsul este dat de

Teorema 10.3 *Fie C un cod binar corector de pachete standard de j erori. Dacă C este transpus la adâncimea s , atunci toate pachetele de cel mult sj erori pot fi corectate, în ipoteza că fiecare cuvânt - cod este afectat de cel mult un pachet standard de j erori.*

Demonstrație: Dacă primul caracter al cuvântului - cod \mathbf{c} este al i -lea caracter transmis, atunci celelalte caractere apar pe pozițiile $i + s, i + 2s, \dots, i + (n - 1)s$. Orice pachet de maxim sj erori va produce un pachet standard de maxim j erori în \mathbf{c} , deci \mathbf{c} va putea fi regăsit la recepție (dacă el nu a mai fost cumva afectat și de alte pachete de erori). q.e.d.

Restricția ca fiecare cuvânt - cod să fie afectat de cel mult un pachet de erori impune condiția ca pachetele de erori să fie separate de perioade în care transmisia este corectă, perioade suficient de lungi pentru evitarea situației ca un bloc de s cuvinte să fie afectat de două pachete distincte de erori. În acest fel, crescând s , crește și lungimea pachetului de erori care poate fi corectat, dar crește și lungimea perioadelor necesare de transmisie fără erori.

Exemplul 10.10 *Codul din Exemplul 18.5 corectează o eroare. Folosind o transpunere la adâncimea 3, el poate corecta orice pachet standard de maxm 3 erori consecutive.*

Un dezavantaj al transpunerii este acela că nu se poate face transmisia până nu au fost codificate toate cele s cuvinte, fapt care nu va permite folosirea directă a circuitelor liniare.

Pentru a evita acest neajuns se va folosi un s -cadru de transpunere întârziată, care aranjează caracterele codificate, nu sub forma unei matrici X_{sn} ci sub formă de "scară" - ca în tabelul:

$$\begin{array}{cccccccccccc}
 c_{1,1} & c_{2,1} & \dots & c_{s+1,1} & c_{s+2,1} & \dots & c_{2s+1,1} & c_{2s+2,1} & \dots & c_{(n-1)s+1,1} & \dots \\
 & & & c_{1,2} & c_{2,2} & \dots & c_{s+1,2} & c_{s+2,2} & \dots & c_{(n-2)s+1,2} & \dots \\
 & & & & & & c_{1,3} & c_{2,3} & \dots & c_{(n-3)s+1,3} & \dots \\
 & & & & & & & & & \vdots & \\
 & & & & & & & & & c_{1,n} & \dots
 \end{array}$$

Deoarece transmisia se face pe coloane, spațiile libere trebuie marcate. Acest lucru se realizează introducând pe aceste poziții un caracter special $*$ $\notin Z_q$.

Exemplul 10.11 *Să reluăm mesajul codificat $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_6$ din Exemplul 18.5.*

Un 1 - cadru de transpunere întârziată va fi

$$\begin{array}{cccccccccccc}
 1 & 0 & 1 & 0 & 1 & 1 & \dots & & & & & \\
 * & 0 & 1 & 1 & 1 & 0 & 1 & \dots & & & & \\
 * & * & 0 & 0 & 1 & 0 & 0 & 1 & \dots & & & \\
 * & * & * & 1 & 1 & 0 & 1 & 1 & 0 & \dots & & \\
 * & * & * & * & 1 & 0 & 0 & 0 & 1 & 0 & \dots & \\
 * & * & * & * & * & 0 & 1 & 0 & 1 & 0 & 0 & \dots
 \end{array}$$

Scvența de caractere transmise este

$$1****00****110***0101**1111*10000\dots$$

Dacă se folosește un 2 - cadru de transpunere întârziată, avem

```

1 0 1 0 1 1 ...
* * 0 1 1 1 0 1 ...
* * * * 0 0 1 0 0 1 ...
* * * * * * 1 1 0 1 1 0 ...
* * * * * * * * 1 0 0 0 1 0 ...
* * * * * * * * * * * 0 1 0 1 0 0 ...

```

iar secvența de caractere transmise este

```
1*****0*****10*****01*****110***110***...
```

Este ușor de găsit o teoremă analogă cu Teorema 20.2:

Teorema 10.4 *Fie C un cod capabil să corecteze toate pachetele standard de cel mult j erori consecutive. Dacă C folosește un s - cadru de transpunere întârziată, atunci se poate corecta orice pachet standard de maxim $j(sn + 1)$ erori, în ipoteza că orice cuvânt - cod este afectat de cel mult un pachet standard de j erori.*

Demonstrație: Exercițiu.

Ca o altă facilitate, pentru codificarea unui mesaj se folosesc adesea două coduri. De exemplu, pentru codificarea muzicii pe compact - discuri se utilizează două coduri Reed - Solomon, iar NASA și Agenția Spațială Europeană folosesc două coduri convoluționale.

Fie A_i două (n_i, k_i) ($i = 1, 2$) coduri liniare. *Transpunerea încrucișată* a lui A_1 cu A_2 se realizează astfel:

1. Mesajele de informație sunt codificate cu A_1 iar cuvintele - cod rezultate sunt transpuse la adâncimea k_2 .
2. Coloanele obținute în acest proces de transpunere (care sunt de lungime k_2) - privite ca mesaje de informație - sunt codificate de codul A_2 .
3. Cuvintele - cod rezultate sunt transpuse la o adâncime s - fixată sau folosind un s - cadru de transpunere întârziată.

Avantajul principal al celui de al doilea proces de codificare este următorul:

Fie d_1, d_2 distanțele minime ale celor două coduri. A_2 poate detecta $d_2 - 1$ erori (nu ne punem problema corectării lor). Dacă s-au detectat erori pentru un cuvânt - cod din C_2 , atunci toate caracterele acestui cuvânt sunt marcate și tratate drept simboluri care pot fi incorecte. Se consideră apoi cuvintele - cod din A_1 . Dacă $n_1 - d_1 + 1$ caractere dintr-un cuvânt - cod din A_1 sunt corecte, atunci se pot determina unic celelalte $d_1 - 1$ caractere (deoarece este imposibil ca două cuvinte - cod distincte din A_1 să coincidă pe $n_1 - d_1 + 1$ poziții, ele diferind pe minim d_1 poziții; afirmația este bazată și pe faptul că pozițiile caracterelor corecte sunt cunoscute).

Deci, dacă un cuvânt - cod din A_1 are cel mult $d_1 - 1$ simboluri marcate și se știe că toate caracterele greșite sunt marcate, cuvintele - cod pot fi decodificate corect.

Cât de mare este un pachet de erori pe care îl poate corecta această schemă? Să presupunem că A_2 a fost transpus la adâncimea s , și că orice cuvânt - cod (atât din A_1 cât și din A_2) a fost afectat de cel mult un pachet de erori. Dacă apare un pachet de cel

mult $s(d_2 - 1)$ erori consecutive, atunci el va afecta maxim $d_2 - 1$ simboluri din fiecare cuvânt - cod din A_2 . Deci aceste erori vor fi detectate și conduc la marcarea tuturor caracterelor cuvintelor - cod implicate. Dacă $s \leq d_1 - 1$, atunci orice cuvânt - cod din A_1 are cel mult $d_1 - 1$ simboluri marcate.

În ipoteza că nu există decât maxim un pachet de erori care l-a perturbat, caracterele marcate pot fi corectate.

Am arătat astfel:

Teorema 10.5 *Fie o codificare care folosește transpunerea încrucișată a (n_1, k_1) - codului A_1 cu (n_2, k_2) - codul A_2 , A_2 fiind transpus la adâncimea s ($s \leq d_1 - 1$). Dacă fiecare cuvânt - cod este afectat de cel mult un pachet standard de erori, atunci toate pachetele standard de erori de lungime maxim $s(d_2 - 1)$ pot fi corectate. (d_1, d_2 sunt distanțele minime ale celor două coduri).*

Exemplul 10.12 *Fie A_1 și A_2 codurile binare, definite respectiv de matricile generatoare*

$$G_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \quad G_2 = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Deci, $n_1 = 8, k_1 = 4, d_1 = 4, n_2 = 6, k_2 = 3, d_2 = 3$.

Să codificăm mesajele de informație $\mathbf{m}_1 = 1000$, $\mathbf{m}_2 = 1100$, $\mathbf{m}_3 = 1010$ folosind o transpunere încrucișată a lui A_1 cu A_2 , A_2 fiind transpus la o adâncime $s = 3 = d_1 - 1$.

Codificarea mesajelor de informație cu codul A_1 dă:

$$\mathbf{c}_1 = \mathbf{m}_1 G_1 = 10001110, \quad \mathbf{c}_2 = \mathbf{m}_2 G_1 = 11000011, \quad \mathbf{c}_3 = \mathbf{m}_3 G_1 = 10100101.$$

Transpunerea acestor cuvinte la adâncimea $k_2 = 3$ conduce la mesajul

$$111 \ 010 \ 001 \ 000 \ 100 \ 101 \ 110 \ 011$$

Aceste 8 mesaje de informație sunt codificate cu A_2 în:

$$\begin{aligned} \mathbf{c}'_1 &= 111000, & \mathbf{c}'_2 &= 010101, & \mathbf{c}'_3 &= 001011, & \mathbf{c}'_4 &= 000000, \\ \mathbf{c}'_5 &= 100110, & \mathbf{c}'_6 &= 101101, & \mathbf{c}'_7 &= 110011, & \mathbf{c}'_8 &= 011110. \end{aligned}$$

și apoi transpuse la adâncimea $s = 3$ (\mathbf{c}'_6 și \mathbf{c}'_7 vor fi transpuse cu următorul cuvânt - cod \mathbf{c}'_9 obținut de următoarele trei mesaje de informație $\mathbf{m}_4, \mathbf{m}_5, \mathbf{m}_6$).

Secvența de caractere transmisă va începe cu

$$100 \ 110 \ 101 \ 010 \ 001 \ 011 \ 011 \ 000 \ 001 \ 011 \ 010 \ 001 \ \dots$$

Conform Teoremei 20.5, folosind A_2 pentru a detecta $d_2 - 1 = 2$ erori, și apoi A_1 pentru a corecta toate caracterele marcate, se pot corecta în ansamblu toate pachetele de cel mult $s(d_2 - 1) = (d_1 - 1)(d_2 - 1) = 6$ erori consecutive.

De exemplu, să presupunem că primii 6 biți au fost transmiși incorect, deci s-a primit mesajul

$$011 \ 001 \ 101 \ 010 \ 001 \ 011 \ \dots$$

Eliminând efectul transpunerii la adâncimea $s = 3$, se ajunge la cuvintele

$$001000 \quad 100101 \quad 111011$$

(care comparate respectiv cu $\mathbf{c}'_1, \mathbf{c}'_2, \mathbf{c}'_3$ au erori fiecare pe primele două poziții). Deoarece sindromurile sunt nenule, A_2 va detecta erori în toate cele trei cuvinte, deci toți cei 18 biți sunt marcați (îi vom înlocui cu *).

Presupunând că nu mai sunt alte erori, după un procedeu similar se obțin cuvintele $\mathbf{c}'_4, \dots, \mathbf{c}'_8$, fără caractere marcate.

Eliminând acum efectul transpunerii la adâncimea $k_2 = 3$, se obțin cuvintele

$$\mathbf{c}_1 = ***01110, \quad \mathbf{c}_2 = ***00011, \quad \mathbf{c}_3 = **100101.$$

Există o modalitate unică de a ajunge la cuvinte - cod din A_1 prin înlocuirea * cu caractere 0 sau 1.

10.4 Exerciții

10.1 Demonstrați Lema 15.1

10.2 Arătați că dacă un cod ciclic detectează o eroare standard \mathbf{e} , atunci detectează toate erorile standard obținute prin permutări ciclice ale lui \mathbf{e} .

10.3 Verificați că pachetele ciclice standard de erori de lungime 3 din Z_2^{15} au sindromuri diferite pentru codurile din Exemplul 18.7.

10.4 Arătați că $g(X) = 1 + X^2 + X^4 + X^5$ generează un $(15, 10)$ - cod ciclic binar corector de pachete de maxim 2 erori. Este acesta un cod corector de 2 erori independente ?

10.5 Arătați că $g(X) = 1 + X^3 + X^4 + X^5 + X^6$ generează un $(15, 9)$ - cod ciclic binar corector de pachete de maxim 3 erori. Este acesta un cod corector de 3 erori independente ?

10.6 Arătați că $g(X) = 1 + X^4 + X^6 + X^7 + X^8$ generează un $(15, 7)$ - cod ciclic binar corector de 2 erori independente și de pachete de cel mult 4 erori consecutive.

10.7 Fie codul din Exemplul 18.7. Să se decodifice mesajele:

$$\begin{array}{lll} 101101110001000, & 001101100010101, & 100110101010011, \\ 101101000010111, & 000000111110000. & \end{array}$$

10.8 Fie $(15, 10)$ - codul ciclic binar generat de $g(X) = 1 + X^2 + X^4 + X^5$. Ce lungimi au pachetele standard de erori pe care le poate corecta ?

Decodificați mesajele:

$$\begin{array}{llll} 010101000010010, & 011010010010100, & 001101000000100, & 000100010100101, \\ 000000011111001. & & & \end{array}$$

10.9 Fie codul ciclic binar generat de polinomul $g(X) = 1 + X + X^2 + X^3 + X^6$. Codificați mesajele de informație $m_1(X) = 1$, $m_2(X) = X^2$, $m_3(X) = 1 + X$, $m_4(X) = 1 + X^2$, $m_5(X) = X^3$, $m_6(X) = 1$.

Determinați șirul de biți transmiși dacă se folosește o transpunere la adâncimea s unde

$$(a) s = 1; \quad (b) s = 2; \quad (c) s = 3.$$

10.10 Ce secvență de caractere este transmisă dacă se folosește un 0 - cadru de transpunere întârziată ?

10.11 Demonstrați Teorema 20.3

10.12 Folosind codurile definite în Exemplul 18.6, să se codifice următoarele mesaje de informație prin transpunerea încrucișată a lui A_1 cu A_2 :

$$(a) \mathbf{m}_1 = 0110, \mathbf{m}_2 = 1011, \mathbf{m}_3 = 1111, \quad s = 2$$

$$(b) \mathbf{m}_1 = 0110, \mathbf{m}_2 = 1011, \mathbf{m}_3 = 1111, \quad s = 3$$

$$(c) \mathbf{m}_1 = 0010, \mathbf{m}_2 = 1111, \mathbf{m}_3 = 1010, \quad s = 3$$

$$(d) \mathbf{m}_1 = 1000, \mathbf{m}_2 = 0100, \mathbf{m}_3 = 0010, \mathbf{m}_4 = 0001, \mathbf{m}_5 = 0011, \mathbf{m}_6 = 0100 \quad s = 3$$

10.13 Folosind codurile din Exemplul 18.6 au fost obținute prin transpunere încrucișată a lui A_1 cu A_2 la adâncimea $s = 3$, următoarele secvențe binare:

$$(a) 000001001111011000100011100011100011100000000000000000000000 \dots$$

$$(b) 100011001111101010011001111010100110100100011101000100 \dots$$

Aflați mesajele de informație $\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3$.

10.14 Găsiți un rezultat analog Teoremei 20.5 dacă pentru A_2 se folosește s - cadrul de transpunere întârziată în loc de s - transpunere.

Capitolul 11

Alte definiții ale codurilor ciclice

Pentru început vom trece în revistă câteva noțiuni algebrice – fundamentale în definirea unei alte modalități de construcție a codurilor ciclice.

11.1 Elemente primitive în extensii Galois

Definiția 11.1 Fie $(F, +, \cdot)$ un corp finit, în care s-au notat cu 0 și 1 elementele unitate față de cele două operații. Un element $a \in F$ are ordin n ($n \geq 1$) dacă $a^n = 1$ și $a^k \neq 1, \forall k, 0 < k < n$. Vom scrie $\text{ord}(a) = n$.

Propoziția 11.1 Într-un corp finit $(F, +, \cdot)$:

- Orice element nenul are un ordin finit;
- Dacă $n = \text{ord}(a)$ atunci a, a^2, \dots, a^n sunt distincte;
- $a^k = 1$ dacă și numai dacă $\text{ord}(a) | k$ ($\text{ord}(a)$ este divizor al lui k).

Demonstrație: Fie $a \in F, a \neq 0$. Cum F este finit, elementele a, a^2, a^3, \dots nu pot fi toate distincte. Vom alege cel mai mic n pentru care $\exists i a^{n+i} = a^i$. Relația se poate simplifica (lucram într-un corp) cu a^i și se obține $a^n = 1$. Din construcție, a, a^2, \dots, a^n sunt distincte, deci n este ordinul lui a .

Pentru $k = ni$ avem $a^k = (a^n)^i = 1^i = 1$. Reciproc, să presupunem $a^k = 1$. Teorema împărțirii cu rest dă $k = qn + r, 0 \leq r < n$. Avem $1 = a^k = a^{qn+r} = a^r$.

Cum $r < n$ rezultă $r = 0$. q.e.d.

Definiția 11.2 Un element $a \in F$ este primitiv dacă $F^* = F \setminus \{0\} = \{1, a, a^2, a^3, \dots\}$.

Observația 11.1 Dacă F are r elemente, atunci un element este primitiv dacă și numai dacă are ordinul $r - 1$.

Teorema 11.1 Orice corp finit are cel puțin un element primitiv.

Demonstrație: Dacă F este finit ($\text{card}(F) = r$), se poate găsi un element $a \in F$ de ordin n maxim. Evident $n \leq r - 1$. Mai trebuie arătat că $n \geq r - 1$.

Fie $b \in F^*$ de ordin s . Să-l descompunem pe s în factori primi: $s = p^i q^j \dots$ (p, q, \dots numere prime distincte). La rândul lui $n = p^t n'$ unde n' nu este divizibil cu p (iar t poate fi eventual zero).

Să considerăm $s' = s/p^i$ (deci $s = p^i s'$) și fie $c = a^{p^t} b^{s'} \in F$. Vrem să arătăm că c are ordinul $m = p^i n'$. Pentru aceasta, avem:

$$c^m = a^{p^t m} b^{s' m} = a^{p^t n' p^i} b^{p^i s' n'} = (a^n)^{p^i} (b^s)^{n'} = 1^{p^i} 1^{n'} = 1.$$

Mai rămâne de arătat că $c^{m'} = 1 \implies m' \geq m$. Este suficient să verificăm că p^i și n' sunt divizori ai lui m' ; deoarece cele două numere sunt prime între ele, va rezulta că produsul $m = p^i n'$ este de asemenea divizor al lui m' .

Din $1 = (c^{m'})^{n'} = a^{p^t n' m'} b^{s' m' n'} = (a^n)^{m'} b^{s' m' n'}$ rezultă (Propoziția 11.1) că ordinul $s = p^i s'$ al lui b divide $s' m' n'$; cum $(p', n') = 1$, se deduce $p^i | m'$.

În plus, din $1 = (c^{m'})^{p^i} = a^{p^t p^i m'} b^{p^i s' m'} = a^{p^t p^i m'} (b^s)^{m'} = a^{p^t p^i m'}$ rezultă că ordinul $n = p^t n'$ al lui a divide $p^t p^i m'$; deci $n' | p^i m'$, adică $n' | m'$.

Cum n este cel mai mare ordin ale elementelor din F , avem $\text{ord}(e) \leq \text{ord}(a)$, sau $m \leq n$, adică $p^i n' \leq p^t n'$; deci $i \leq t$, adică p^i divide $n = p^t n'$.

În mod similar, toți factorii lui s sunt divizori ai lui n , deci $s | n$.

Am arătat următoarea afirmație:

$$\forall b \in F^* \implies \text{ord}(b) | n$$

Deci b este o rădăcină a polinomului $X^n - 1 = 0$, de unde rezultă că polinomul $X^n - 1 = 0$ are $r - 1$ rădăcini, adică $r - 1 \leq n$. q.e.d.

Fie $(F, +, \cdot)$ un corp finit și $f(X) \in F[X]$ un polinom arbitrar cu coeficienți în F . Spunem că $a \in F$ este rădăcină a lui f dacă $f(a) = 0$.

Se poate demonstra imediat afirmația:

Propoziția 11.2 *Dacă un polinom f are rădăcinile distincte a_1, a_2, \dots, a_n atunci el este divizibil cu polinomul $(X - a_1)(X - a_2) \dots (X - a_n)$.*

Exemplul 11.1 *Polinomul $X^3 + 1$ are o rădăcină 1 în Z_2 . Deci $X^3 + 1$ se divide cu $X + 1$:*

$$X^3 + 1 = (X + 1)(X^2 + X + 1)$$

Aceasta este o factorizare completă; deoarece $X^2 + X + 1$ nu are rădăcini în Z_2 , el nu poate fi descompus în produsul a două polinoame de gradul 1.

Același polinom $X^3 + 1$ are în Z_3 pe 2 ca rădăcină triplă, deci aici putem scrie:

$$X^3 + 1 = (X + 1)^3.$$

Rezultă că factorizarea unui polinom depinde de corpul în care este definit.

Din Teorema 11.1 și Propoziția 20.1 rezultă că dacă $a \in F$ este un element de ordin n , atunci $X^n - 1$ se poate descompune în

$$X^n - 1 = (X - a)(X - a^2) \dots (X - a^n).$$

Definiția 11.3 *Un polinom $f(X) \in F[X]$ de grad n este ireductibil în corpul F dacă nu se poate descompune în produsul a două polinoame din $F[X]$ de grad mai mic decât n . În caz contrar, f este un polinom reductibil în F .*

Observația 11.2

- Orice polinom liniar este ireductibil. Pentru polinoame de grad cel puțin 2, Propoziția 20.1 afirmă că:
Un polinom ireductibil într-un corp nu are rădăcini în acel corp.
Este interesant că reciproca nu este adevărată decât pentru polinoamele de grad 2 și 3. Astfel, polinomul $f(X) = (X^2 + X + 1)^2 \in Z_2[X]$, deși nu are rădăcini în Z_2 , este reductibil.
- În \mathcal{R} singurele polinoame ireductibile sunt de gradul 1 sau 2. Dacă un polinom are grad impar, el are sigur o rădăcină reală, iar dacă este de grad par, atunci are sau cel puțin o rădăcină reală, sau cel puțin două rădăcini complexe de forma $a \pm ib$. În acest ultim caz, el se va divide cu $(x - a)^2 + b^2$, care este un polinom de gradul 2 din $\mathcal{R}[X]$.
- În corpul complex \mathcal{C} , teorema fundamentală a algebrei asigură că orice polinom ireductibil are gradul 1.

Definiția 11.4 Caracteristica unui corp finit F este cel mai mic număr de termeni ai sumei $S = 1 + 1 + \dots + 1$ cu proprietatea $S = 0$.

Propoziția 11.3 Caracteristica unui corp finit este număr prim.

Demonstrație: Să considerăm elementele $S_i = 1 + 1 + \dots + 1$ de câte i termeni fiecare. Deoarece nu pot fi o infinitate de valori distincte, fie p minim cu proprietatea $\exists i, S_i = S_{i+p}$. Rezultă $S_p = S_{i+p} - S_i = 0$. Deci F are caracteristica p .

Presupunem $p = jt$ cu $1 \leq j < p$. Evident $0 = S_{jt} = S_j + S_j + \dots + S_j$ (t termeni). Deoarece $j < p$ avem $S_j \neq 0$. Împărțind relația cu S_j se obține $0 = 1 + 1 + \dots + 1 = S_t$. Deci $t = p$. q.e.d.

Corolarul 11.1 Orice corp de caracteristică q este o extensie a lui Z_q .

Demonstrație: Exercițiu.

Corolarul 11.2 $GF(q^r)$ este un corp de caracteristică q .

Demonstrație: Reamintim că $GF(q^r)$ conține toate polinoamele din $Z_q[X]$ de grad cel mult $r - 1$. În particular, polinoamele de grad 0 (constantele) formează un subcorp izomorf cu Z_q care are caracteristica q . q.e.d.

Propoziția 11.4 Într-un corp de caracteristică q avem $(a + b)^q = a^q + b^q$.

Demonstrație: Se folosește binomul lui Newton, în care $C_q^k = 0, \forall k, 0 < k < q$. q.e.d.

11.2 Polinoame minimale

În cele ce urmează vom restrânge studiul la cazul $F = Z_q$, q număr prim.

Definiția 11.5 Fie β un element dintr-o extensie a lui Z_q . Se numește "polinom minimal" al lui β polinomul normat $g(X) \in Z_q[X]$ de grad minim, cu $g(\beta) = 0$.

Existența acestui polinom este asigurată de teorema 4.1 (Prelegerea 4), pe baza căreia s-au definit extensiile Galois.

Exemplul 11.2 Să considerăm extensia $GF(2^3)$ generată de rădăcina 'al a polinomului $1 + X + X^3$. Deoarece α este primitiv, avem $GF(2^3) = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$. Polinoamele minimale ale fiecărui element sunt date în tabelul:

Element	Polinom minimal
0	X
1	$1 + X$
$\alpha, \alpha^2, \alpha^4$	$1 + X + X^3$
$\alpha^3, \alpha^5, \alpha^6$	$1 + X^2 + X^3$

Propoziția 11.5 Un polinom minimal $g(X) \in Z_q[X]$ este ireductibil peste Z_q .

Demonstrație: Dacă ar exista descompunerea $g(X) = u(X)v(X)$ cu $u(X), v(X) \in Z_q[X]$, atunci vom avea $u(\beta) = 0$ sau $v(\beta) = 0$, ceea ce contrazice definiția polinomului minimal pentru β . q.e.d.

Corolarul 11.3 Dacă $f(X) \in Z_q[X]$ verifică $f(\beta) = 0$ atunci $g(X) | f(X)$.

De aici rezultă că un polinom normat peste Z_q care admite pe β ca rădăcină este polinom minimal al lui β . Acest polinom este unic.

Teorema 11.2 Fie $\beta \in GF(q^k)$. Atunci există un polinom minimal al lui β de grad cel mult k .

Demonstrație: După cum rezultă din definirea extensiei Galois și din Teorema 5.3, $GF(q^k)$ are dimensiunea k . O bază a lui este $\{1\}, \{X\}, \dots, \{X^{k-1}\}$. Deci cele $k + 1$ elemente $1, \beta, \beta^2, \dots, \beta^k \in GF(q^k)$ sunt liniar dependente. Relația lor de dependență conduce la construirea unui polinom de grad maxim k pentru care β este rădăcină. q.e.d.

Teorema 11.3 Elementele nenule din Z_q sunt date de soluțiile ecuației $X^{q-1} - 1 = 0$.

Demonstrație: Elementele din $Z_q \setminus \{0\}$ formează grup multiplicativ. Ordinul fiecărui element divide ordinul grupului, care este $q - 1$ (Teorema 11.1).

Fie $\beta \in Z_q \setminus \{0\}$; subgrupul ciclic generat de β este $1, \beta, \beta^2, \dots, \beta^{t-1}$ unde $b^t = 1$ și $t | q - 1$. Deci $\beta^{q-1} = 1$.

În plus, ecuația $X^{q-1} - 1 = 0$ are $q - 1$ rădăcini. q.e.d.

Teorema 11.4 $X^m - 1 | X^n - 1 \iff m | n$.

Demonstrație: "⇐": Fie $n = md$. Cum $Y - 1$ divide pe $Y^d - 1$, dacă se ia $Y = X^m$, vom avea $X^m - 1 | X^{md} - 1$.

"⇒": Să presupunem că $X^m - 1 | X^n - 1$ și fie $n = md + s$, ($s < m$). Avem $X^n - 1 = X^s(X^{md} - 1) + X^s - 1 = q(X)(X^m - 1) + r(X)$ cu $\text{grad}(r(X)) = s < m$.

Pentru a verifica ipoteza, trebuie ca $s = 0$, deci $n = md$. q.e.d.

Teorema 11.5 Fie $f(X) \in Z_q[X]$ și β o rădăcină a sa (eventual dintr-o extensie a lui Z_q). Atunci și β^q este rădăcină a lui $f(X)$.

Demonstrație: Scriem $f(X) = a_0 + a_1X + \dots + a_nX^n$. Cum Z_q este corp de caracteristică q , este adevărată relația $(a + b)^q = a^q + b^q$ (Propoziția 11.4).

De asemenea (Teorema 20.2) $\forall a \in Z_q \setminus \{0\}$ avem $a^{q-1} - 1 = 0$, deci $a^q = a$.

Atunci se poate scrie

$$(f(X))^q = a_0^q + a_1^q X^q + \dots + a_n^q (X^n)^q = a_0 + a_1 X^q + \dots + a_n X^{nq} = f(X^q).$$

În particular, $f(\beta^q) = (f(\beta))^q = 0$.

q.e.d.

Teorema 11.6 *Orice polinom normat ireductibil peste Z_q de grad m este un factor al polinomului $X^{q^m} - X$.*

Demonstrație: Dacă $g(X) = X$, afirmația este banală.

Să presupunem că $g(X) \neq X$ și fie $\beta \in GF(q^m)$ o rădăcină nenulă a sa, deci $g(\beta) = 0$. Cum $g(X)$ este ireductibil, el va fi chiar polinomul minimal al lui β . Conform Teoremei 20.2, β satisface ecuația $X^{q^m-1} - 1 = 0$, deci (Corolar 11.3) $X^{q^m-1} - 1$ se divide cu polinomul minimal $g(X)$.

q.e.d.

Teorema 11.7 *Orice factor ireductibil $g(X) \in Z_q[X]$ al lui $X^{q^m} - X$ are gradul cel mult m .*

Demonstrație: Fie $g(X) | X^{q^m} - X$, ireductibil peste Z_q , cu $\text{grad}(g(X)) = k$.

Vom considera algebra polinoamelor modulo $g(X)$, în care luăm $\alpha = \{X\}$. Se știe (Teorema 4.1) că $g(\alpha) = 0$. Un element oarecare din această algebră este de forma

$$\beta = a_0 + a_1\alpha + \dots + a_{k-1}\alpha^{k-1}.$$

$$\text{Atunci } \beta^{q^m} = a_0^{q^m} + a_1^{q^m}\alpha^{q^m} + \dots + a_{k-1}^{q^m}(\alpha^{k-1})^{q^m} = a_0 + a_1\alpha^{q^m} + \dots + a_{k-1}(\alpha^{q^m})^{k-1} = a_0 + a_1\alpha + \dots + a_{k-1}\alpha^{k-1} = \beta$$

deoarece $GF(q^m)$ are tot caracteristica q ca și Z_q , iar α – fiind o rădăcină a lui $g(X)$, este rădăcină și a lui $X^{q^m} - X$, deci $\alpha^{q^m} = \alpha$, de unde rezultă $\alpha^{jq^m} = \alpha^j \forall j \geq 0$.

Am obținut în final faptul că β este o soluție a ecuației $X^{q^m} - X = 0$. Sunt posibile q^k asemenea elemente β distincte, iar cum ecuația are q^m rădăcini, rezultă $q^k \leq q^m$, deci $k \leq m$.

q.e.d.

Teorema 11.8 *Fie $\beta \in GF(q^m)$ de polinom minimal $g(X)$, $\text{grad}(g(X)) = k$ și $\text{ord}(\beta) = t$. Atunci*

$$(i) \quad t | q^k - 1;$$

$$(ii) \quad k \text{ este minim cu proprietatea (i).}$$

Demonstrație: Știm (Teorema 11.6) că $g(X)$ este un factor al lui $X^{q^k} - X$; deci $\beta^{q^k} = \beta$, de unde rezultă $t | q^k - 1$.

Să presupunem acum că există $p < k$ cu $t | q^p - 1$. Atunci $\beta^{q^p-1} = 1$, adică β este o rădăcină a ecuației $X^{q^p} - X = 0$, deci $g(X)$ este un factor al polinomului $X^{q^p} - X$. Conform Teoremei 11.7, rezultă $k \leq p$, contradicție.

q.e.d.

Teorema 11.9 *Fie polinomul $g(X) \in Z_q[X]$, $\text{grad}(g(X)) = m$ și $\beta \in GF(q^m)$ o rădăcină a sa. Atunci $\beta, \beta^q, \beta^{q^2}, \dots, \beta^{q^{m-1}}$ sunt toate rădăcinile lui $g(X)$.*

Demonstrație: Deoarece $g(\beta) = 0$, avem – conform Teoremei 20.5 – că $\beta^q, \beta^{q^2}, \dots, \beta^{q^{m-1}}$ sunt și ele rădăcini ale lui $g(X)$. De asemenea, conform Teoremei 11.6, β este rădăcină a lui $X^{q^m} - X$, adică $\beta^{q^m} = \beta$.

Mai rămâne de arătat că aceste rădăcini sunt distincte. Presupunem că există $0 \leq i < j < m$ cu $\beta^{q^i} = \beta^{q^j}$. Avem

$$\beta = \beta^{q^m} = \left(\beta^{q^j}\right)^{q^{m-j}} = \left(\beta^{q^i}\right)^{q^{m-j}} = \beta^{q^{m+i-j}}$$

Deci β este rădăcină a polinomului $X^{q^{m+i-j}} - X$ și – cu Teorema 11.7, $\text{grad}(g(X)) = m \leq m+i-j < m$, contradicție. q.e.d.

Din Teorema 11.9 rezultă că luând extensia $GF(q^m)[X]$, $g(X)$ se poate scrie

$$g(X) = (X - \beta)(X - \beta^q) \dots (X - \beta^{q^{m-1}}).$$

Teorema 11.10 *Fie $g(X)$ un polinom normat ireductibil peste Z_q , $\text{grad}(g(X)) = m$, și $\beta \in GF(q^m)$ o rădăcină a sa. Atunci toate rădăcinile lui $g(X)$ au același ordin.*

Demonstrație: Fie $p = \text{ord}(\beta)$, $p' = \text{ord}(\beta^{q^j})$, ($1 < j < m$) care – conform Teoremei 11.9 – este tot rădăcină a lui $g(X)$. Avem:

$$\left(\beta^{q^j}\right)^p = (\beta^p)^{q^j} = 1 \text{ și deci } p'|p.$$

De asemenea,

$$\beta^{p'} = (\beta^{q^m})^{p'} = \left(\left(\beta^{q^j}\right)^{q^{m-j}}\right)^{p'} = \left(\left(\beta^{q^j}\right)^{p'}\right)^{q^{m-j}} = 1 \implies p|p'.$$

S-a mai folosit Teorema 11.8, conform căreia p divide $q^m - 1$ dar nu divide nici un număr de forma $q^s - 1$ cu $s < m$. Deci $p = p'$. q.e.d.

11.3 Definirea codurilor ciclice prin rădăcini

O altă metodă de definire a unui cod ciclic constă în listarea tuturor rădăcinilor (eventual într-o extensie a lui Z_q) cuvintelor - cod (considerate ca polinoame).

Definiția 11.6 *Fie $\alpha_1, \alpha_2, \dots, \alpha_r$ elemente dintr-o extensie a lui Z_q . Codul ciclic C este format din toate elementele $\{f(X)\}$ din algebra polinoamelor modulo $X^n - 1$, care admit pe α_i , ($1 \leq i \leq r$) ca rădăcini simple.*

De remarcat că în această definiție, n este deocamdată nedeterminat.

Conform Corolarului 11.3, $f(X)$ se va divide cu fiecare polinom minimal $m_i(X)$ corepunzător rădăcinii α_i ($1 \leq i \leq r$). Deci polinomul generator al codului ciclic este

$$g(X) = \text{cmmmc} \{m_1(X), m_2(X), \dots, m_r(X)\}.$$

Propoziția 11.6 $C = (\{g(X)\})$.

Demonstrație: Este lăsată ca exercițiu.

Deoarece polinomul generator $g(X)$ divide pe $X^n - 1$, rezultă că fiecare α_i este rădăcină a lui $X^n - 1$. Deci ordinul $\text{ord}(\alpha_i)$ al fiecărei rădăcinii α_i ($1 \leq i \leq r$) va divide pe n .

O modalitate naturală de definire a lui n este atunci

$$n = \text{cmmmc} \{ \text{ord}(\alpha_1), \text{ord}(\alpha_2), \dots, \text{ord}(\alpha_r) \}.$$

Un caz particular important este acela în care toate rădăcinile $\alpha_1, \alpha_2, \dots, \alpha_r$ sunt puteri ale unui anumit element, adică

$$\alpha_i = \alpha^{u_i} \quad (1 \leq i \leq r).$$

Fie $p = \text{ord}(\alpha)$. Atunci polinomul minimal $m_i(X)$ al lui α_i va avea (Teorema 11.9) toate rădăcinile sale printre elementele $\alpha^{u_i}, \alpha^{qu_i}, \alpha^{q^2u_i}, \dots$

Numărul factorilor lui $g(X)$ și gradul fiecărui polinom minimal $m_i(X)$ se vor determina atunci din ordinul p și din exponenții u_i, qu_i, q^2u_i, \dots ($1 \leq i \leq r$). Într-adevăr, numărul claselor distincte de resturi modulo p din succesiunea u_i, qu_i, \dots va da gradul polinomului minimal $m_i(X)$.

Exemplul 11.3 Fie $\alpha \in GF(2^4)$ primitiv, rădăcină a polinomului $1 + X + X^4$, ireductibil peste Z_2 . Să definim un cod C care să aibă ca rădăcini $\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6$.

Fiind primitiv, $\text{ord}(\alpha) = 2^4 - 1 = 15$ (vezi și Prelegerea 8, Exemplul 8.8).

Fie $m_i(X)$ polinomul minimal al rădăcinii α_i , ($1 \leq i \leq 6$).

Rădăcinile lui $m_1(X)$ sunt $\alpha, \alpha^2, \alpha^4, \alpha^8$ ($\alpha^{16} = \alpha$), deci

$$m_1(X) = m_2(X) = m_4(X) = (X - \alpha)(X - \alpha^2)(X - \alpha^4)(X - \alpha^8).$$

Acest polinom este cunoscut, anume $1 + X + X^4$.

Rădăcinile lui $m_3(X)$ sunt $\alpha^3, \alpha^6, \alpha^{12}, \alpha^{24} = \alpha^9$ ($\alpha^{18} = \alpha^3$), deci $m_3(X)$ este un polinom de gradul 4, egal cu $m_6(X)$.

Notăm $m_3(X) = a_0 + a_1X + a_2X^2 + a_3X^3 + X^4$. Detaliind $m_3(\alpha^3) = 0$, avem

$$a_0 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + a_1 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} + a_2 \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} + a_3 \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

de unde se poate construi sistemul de ecuații

$$a_0 + 1 = 0, \quad a_3 + 1 = 0, \quad a_2 + 1 = 0, \quad a_1 + a_2 + a_3 + 1 = 0,$$

cu soluția $a_0 = a_1 = a_2 = a_3 = 1$, deci $m_3(X) = m_6(X) = 1 + X + X^2 + X^3 + X^4$.

Rădăcinile lui $m_5(X)$ sunt α^5, α^{10} ($\alpha^{20} = \alpha^5$), deci $m_5(X)$ este un polinom de gradul 2: $m_5(X) = b_0 + b_1X + X^2$. Cum $m_5(\alpha^5) = 0$, se obține:

$$b_0 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + b_1 \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

care are soluția $b_0 = b_1 = 1$, deci $m_5(X) = 1 + X + X^2$.

Polinomul generator al codului C este

$$\begin{aligned} g(X) &= \text{cmmmc} \{ m_1(X), m_2(X), m_3(X), m_4(X), m_5(X), m_6(X) \} = \\ &= m_1(X)m_3(X)m_5(X) = (1 + X + X^4)(1 + X + X^2 + X^3 + X^4)(1 + X + X^2) = 1 + X + \\ &X^2 + X^4 + X^5 + X^8 + X^{10}. \end{aligned}$$

De remarcat că era suficient să cerem ca doar $\alpha, \alpha^3, \alpha^5$ să fie rădăcini ale codului.

Fie polinomul $f(X) = a_0 + a_1X + \dots + a_{n-1}X^{n-1} \in Z_q[X]$ și α o rădăcină a sa. Atunci $0 = f(\alpha) = a_0 + a_1\alpha + a_2\alpha^2 + \dots + a_{n-1}\alpha^{n-1}$, sau – altfel scris:

$$(a_0 \ a_1 \ a_2 \ \dots \ a_{n-1}) \begin{pmatrix} \mathbf{1} \\ \alpha \\ \vdots \\ \alpha^{n-1} \end{pmatrix} = 0.$$

Deci cuvântul - cod corespunzător polinomului $f(X)$ este în spațiul nul al matricii $(\mathbf{1} \ \alpha \ \dots \ \alpha^{n-1})$ (1)

Aceasta este și condiția de divizibilitate a polinomului $f(X)$ cu polinomul minimal $m(X)$ al lui α . Mulțimea polinoamelor care o satisfac formează idealul generat de $m(X)$. Dimensiunea acestui ideal este (Prelegerea 9) $n - k$ unde $k = \text{grad}(m(X))$.

Deci spațiul linear generat de matricea (1) are dimensiunea k .

Prin urmare, a cere ca $f(X)$ să admită pe $\alpha_1, \alpha_2, \dots, \alpha_r \in GF(q^m)$ ca rădăcini este echivalent cu a cere ca vectorul corespunzător să aparțină spațiului nul al matricii

$$H_{mr,n} = \begin{pmatrix} \mathbf{1} & \alpha_1 & \alpha_1^2 & \dots & \alpha_1^{n-1} \\ \mathbf{1} & \alpha_2 & \alpha_2^2 & \dots & \alpha_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{1} & \alpha_r & \alpha_r^2 & \dots & \alpha_r^{n-1} \end{pmatrix}$$

Exemplul 11.4 Să revenim la Exemplul 18.5. Polinomul $\{f(X)\}$ aparține codului respectiv dacă și numai dacă vectorul corespunzător aparține spațiului nul al matricii

$$H = \begin{pmatrix} \mathbf{1} & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 & \alpha^7 & \alpha^8 & \alpha^9 & \alpha^{10} & \alpha^{11} & \alpha^{12} & \alpha^{13} & \alpha^{14} \\ \mathbf{1} & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & \mathbf{1} & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & \mathbf{1} & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} \\ \mathbf{1} & \alpha^5 & \alpha^{10} & \mathbf{1} & \alpha^5 & \alpha^{10} & \mathbf{1} & \alpha^5 & \alpha^{10} & \mathbf{1} & \alpha^5 & \alpha^{10} & \mathbf{1} & \alpha^5 & \alpha^{10} \end{pmatrix}$$

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

S-a obținut o matrice 12×15 , deși - conform rezultatelor teoretice, $n = 15$, $n - k = 10$, $k = 5$ și deci matricea de control a codului ar trebui să aibă dimensiunile $(n - k) \times n = 10 \times 15$. Se observă însă că de fapt ultima linie este nulă, iar următoarele două linii sunt identice.

Dei ultimele două linii se pot ignora, ele neaducând nici o informație suplimentară asupra cuvintelor - cod.

11.4 Coduri ciclice ireductibile

Vom mai prezenta și o a treia construcție a codurilor ciclice, realizată de Edwin Berlekamp în 1968.

Definiția 11.7 Fie polinomul $Tr(X) = X + X^q + X^{q^2} + \dots + X^{q^{r-1}} \in GF(q^r)[X]$. Se numește urma lui $\alpha \in GF(q^r)$, expresia $Tr(\alpha)$.

Propoziția 11.7 Tr este o aplicație : $GF(q^r) \rightarrow Z_q$.

Demonstrație: Trebuie arătat că $\forall \alpha \in GF(q^r)$ avem $Tr(\alpha) \in Z_q$. Pentru aceasta este suficient să arătăm că $Tr(\alpha)$ verifică ecuația $X^q - X = 0$. Folosind faptul că $GF(q^r)$ este corp de caracteristică q și că $\forall \alpha \in GF(q^r)$ avem $\alpha^{q^r} = \alpha$, se verifică imediat relația $[Tr(\alpha)]^q = Tr(\alpha)$. q.e.d.

Propoziția 11.8 Tr este aplicație liniară.

Demonstrație: Relația $Tr(\alpha + \beta) = Tr(\alpha) + Tr(\beta)$ este ușor de verificat, deoarece se lucrează în corpuri de caracteristică q . q.e.d.

Propoziția 11.9 Pentru orice $a \in Z_q$ există q^{r-1} valori $\alpha \in GF(q^r)$ cu $Tr(\alpha) = a$.

Demonstrație: Fiecare ecuație $Tr(X) = a$ admite maxim q^{r-1} rădăcini, iar numărul de elemente a posibile este q . Mai trebuie arătat că toate cele q ecuații au rădăcini distincte. Aceasta se poate deduce foarte simplu folosind derivata formală într-un corp de caracteristică q : toate ecuațiile au aceeași derivată: 1. q.e.d.

Teorema 11.11 Polinomul $Tr(X)$ se poate descompune în $GF(q^r)$ în produs de polinoame minimale.

Demonstrație: Demonstrația se bazează pe următorul algoritm:

- (1) Fie polinomul $g(X) = Tr(X)$;
- (2) Se consideră $\alpha \in GF(q^r)$ cu $g(\alpha) = 0$. Deci $g(X)$ este divizibil cu polinomul minimal $m(X)$ al lui α .
- (3) $g(X) := g(X)/m(X)$. Dacă $g(X) = 1$, STOP, altfel se reia pasul (2).

Existența rădăcinilor pentru $g(X)$ (pasul (2)) este asigurată de faptul că în $GF(q^r)$ orice polinom ireductibil este polinom minimal. q.e.d.

Exemplul 11.5 Să considerăm $GF(2^4)$, deci $q = 2$, $r = 4$. Aici se poate verifica descompunerea

$Tr(X) = X + X^2 + X^4 + X^8 = X(1 + X)(1 + X + X^2)(1 + X + X^4)$,
formată din produs de polinoame minimale.

Fie $\alpha \in GF(2^4)$ rădăcină (primitivă) a ecuației $1 + X + X^4 = 0$. Deci toate elementele nenule din $GF(2^4)$ se pot scrie ca puteri ale lui α .

Vom folosi aceasta pentru a lista valorile funcției $Tr : GF(2^4) \rightarrow Z_2$:

X	$Tr(X)$	X	$Tr(X)$	X	$Tr(X)$	X	$Tr(X)$
0	0	α^3	1	α^7	1	α^{11}	1
1	0	α^4	0	α^8	0	α^{12}	1
α	0	α^5	0	α^9	1	α^{13}	1
α^2	0	α^6	1	α^{10}	0	α^{14}	1

După cum se observă, sunt $8 = 2^3$ valori 0 și 8 valori 1.

Putem da acum teorema principală care definește codurile ciclice folosind operatorul Tr :

Teorema 11.12 Fie n număr natural, q număr prim, k ordinul lui q modulo n ($q^k \equiv 1 \pmod{n}$) și $\beta \in GF(q^k)$ element primitiv de ordin n . Atunci

$$C = \{\mathbf{c}_\alpha = (Tr(\alpha), Tr(\alpha\beta), Tr(\alpha\beta^2), \dots, Tr(\alpha\beta^{n-1})) \mid \alpha \in GF(q^k)\}$$

este un (n, k) - cod ciclic peste Z_q .

Demonstrație: Din Propoziția 11.8 rezultă că C este cod liniar.

Se verifică apoi că $\mathbf{c}_{\alpha\beta^{-1}} \in C$ este o permutare ciclică a lui \mathbf{c}_α ; deci C este cod ciclic. Deoarece β este primitiv, înseamnă că polinomul său minimal $m(X) = h_0 + h_1X + \dots + h_kX^k$ are gradul k . Dacă $\mathbf{c}_\alpha = (c_0, c_1, \dots, c_{n-1})$, avem

$$\sum_{i=0}^k c_i h_i = Tr(\alpha m(\beta)) = Tr(\mathbf{0}) = 0,$$

care constituie una din cele $n - k$ ecuații de control pentru C .

Deoarece $m(X)$ este minimal (deci ireductibil), $h(X) = X^k m(X^{-1})$ este polinomul de control pentru C . Cum gradul lui este k , avem un (n, k) - cod ciclic peste Z_q . q.e.d.

De remarcat că această teoremă definește numai codurile ciclice "ireductibile" (care nu conțin subcoduri ciclice netriviiale proprii), deci clasa lor este inclusă strict în clasa codurilor ciclice dată de cele două definiții anterioare (prin ideale și prin rădăcinile polinomului generator).

Exemplul 11.6 Să luăm $n = 15$, $q = 2$, deci $k = 4$. În $GF(2^4)$ vom lua α , rădăcină a polinomului $1 + X + X^4$. Se știe că el este un element primitiv. Aplicația Tr este $Tr(X) = X + X^2 + X^4 + X^8$ iar codul C este format din 16 cuvinte de forma

$$(Tr(\beta), Tr(\beta\alpha), \dots, Tr(\beta\alpha^{n-1})), \forall \beta \in GF(2^4).$$

Pentru $\beta = 0$, $00 \dots 0 \in C$.

Pentru $\beta = 1$, $(Tr(1), Tr(\alpha), \dots, Tr(\alpha^{n-1})) = 000100110101111$

Celelalte 14 cuvinte sunt permutările circulare ale acestuia.

Polinomul de control al codului este $h(X) = X^4 \left(1 + \frac{1}{X} + \frac{1}{X^4}\right) = 1 + X^3 + X^4$, iar polinomul generator

$$g(X) = \frac{X^{16} - X}{X^4 + X^3 + 1} = X + X^4 + X^5 + X^7 + X^9 + X^{10} + X^{11} + X^{12},$$

al cărui vector corespunzător se obține în construcția de sus pentru $\beta = \alpha^2$.

Acesta este un $(15, 4)$ - cod ciclic binar, ireductibil.

De remarcat că pentru $n = 15$ sunt numai două astfel de coduri, celălalt fiind codul dual (rădăcina polinomului $1 + X^3 + X^4$ este de asemenea primitivă).

11.5 Exerciții

11.1 *Demonstrați Corolarul 15.1.*

11.2 *Demonstrați Propoziția 11.6*

11.3 *Construiți următoarele corpuri:*

- (a) $GF(2^2)$;
- (b) $GF(2^3)$ folosind polinomul $1 + X^2 + X^3$;
- (c) $GF(2^4)$ folosind polinomul $1 + X^3 + X^4$;
- (d) $GF(2^5)$ folosind polinomul $1 + X^2 + X^5$.

11.4 *Găsiți o extensie a lui Z_2 în care $X^9 - 1$ să se descompună în factori liniari.*

11.5 *Găsiți toate elementele primitive din $GF(2^3)$ și $GF(2^4)$.*

11.6 *Găsiți toate elementele primitive din $GF(3^2)$ și $GF(5^2)$.*

11.7 *Construiți $GF(2^4)$ ca o extensie a lui $GF(2^2)$.*

11.8 *Fie $\alpha \in GF(2^4)$, rădăcină a polinomului $1 + X + X^4$.*

Găsiți polinoamele minimale ale lui $\beta = \alpha^7$ și $\beta = \alpha^{10}$.

11.9 *Găsiți polinoamele minimale ale tuturor elementelor din:*

- $GF(2^3)$, generat cu $1 + X + X^3$;
- $GF(2^5)$, generat cu $1 + X^2 + X^5$;
- $GF(3^2)$, generat cu $2 + X + X^2$;
- $GF(5^2)$, generat cu $3 + 2X + X^2$.

11.10 *Orice element nenul din $GF(q^m)$ are un ordin prim cu q .*

11.11 *Găsiți polinomul generator al unui cod ciclic binar de lungime 15, de rădăcini $1, \alpha^5, \alpha^7$, $\alpha \in GF(2^4)$ fiind rădăcină a polinomului $1 + X + X^4$.*

Construiți matricea de control a codului.

Arătați că $v(X)$ este polinom - cod dacă și numai dacă ponderea $w(\mathbf{v})$ este pară.

11.12 *Găsiți polinomul generator al unui cod ciclic binar de rădăcini $\alpha^2, \alpha^3, \alpha^6$ unde $\alpha \in GF(2^3)$ este rădăcină a polinomului $1 + X + X^3$.*

11.13 *Să se descompună $Tr(X)$ în produs de polinoame minimale în corpurile:*

- (a) $GF(2^5)$; (b) $GF(3^2)$; (c) $GF(3^3)$; (d) $GF(5^2)$.

11.14 *Folosind Teorema 11.12 să se construiască toate codurile ireductibile pentru $GF(2^3)$, $GF(2^5)$, $GF(3^2)$.*

11.15 *Verificați că polinomul $h(X)$ construit în demonstrația Teoremei 11.12 este polinomul generator al codului dual.*

Capitolul 12

Coduri BCH

Codurile BCH constituie cea mai relevantă clasă de coduri ciclice. Ele au fost definite în mod independent de Bose și Chaudhuri pe de-o parte, Hocquenheim pe de altă parte. Numele de BCH (**B**ose - **C**haudhuri - **H**ocquenheim) a fost dat de Peterson, care s-a ocupat de ele în mod special, construind și algoritmul de decodificare.

12.1 Definirea codurilor BCH

Definiția 12.1 Fie q număr prim, m_0, m, d numere naturale și $\alpha \in GF(q^m)$. Un cod ciclic este cod BCH dacă este definit de rădăcinile polinomului generator

$$\alpha^{m_0}, \alpha^{m_0+1}, \dots, \alpha^{m_0+d-2}.$$

În majoritatea cazurilor studiate se consideră $m_0 = 0$ sau $m_0 = 1$.

Teorema 12.1 Într-un cod BCH lungimea n a cuvintelor - cod este egală cu ordinul e al lui α .

Demonstrație: Conform construcției codurilor ciclice bazată pe rădăcinile polinomului generator (Prelegerea 11), lungimea n a cuvintelor - cod este cel mai mic multiplu comun al ordinelor rădăcinilor. În cazul codurilor BCH avem

$$(\alpha^{m_0})^n = \alpha^{m_0 n} = 1 \text{ și } 1 = (\alpha^{m_0+1})^n = \alpha^{m_0+n} = \alpha^{m_0} \alpha^n,$$

deci $\alpha^n = 1$. Prin urmare $e|n$ și $n \geq e$.

De asemenea, $(\alpha^j)^e = (\alpha^e)^j = 1 \forall j = m_0, m_0 + 1, \dots, m_0 + d - 2$, adică $n \leq e$.

Rezultă $n = e$.

q.e.d.

De remarcat că, dacă α este primitiv, atunci $e = q^m - 1$.

Teorema 12.2 Într-un cod BCH definit de rădăcinile $\alpha^{m_0}, \alpha^{m_0+1}, \dots, \alpha^{m_0+d-2}$ ($\alpha \in GF(q^m)$), distanța minimă a codului este cel puțin d (și deci codul poate corecta orice combinație de $t \leq \left\lfloor \frac{d-1}{2} \right\rfloor$ erori independente).

Demonstrație: Un cod BCH este spațiul nul al matricii

$$H = \begin{pmatrix} 1 & \alpha^{m_0} & (\alpha^{m_0})^2 & \dots & (\alpha^{m_0})^{n-1} \\ 1 & \alpha^{m_0+1} & (\alpha^{m_0+1})^2 & \dots & (\alpha^{m_0+1})^{n-1} \\ & & \vdots & & \\ 1 & \alpha^{m_0+d-2} & (\alpha^{m_0+d-2})^2 & \dots & (\alpha^{m_0+d-2})^{n-1} \end{pmatrix}$$

Să considerăm următorul determinant, obținut prin alegerea arbitrară a $d-1$ coloane din H :

$$\begin{aligned} D &= \begin{vmatrix} (\alpha^{m_0})^{j_1} & (\alpha^{m_0})^{j_2} & \dots & (\alpha^{m_0})^{j_{d-1}} \\ (\alpha^{m_0+1})^{j_1} & (\alpha^{m_0+1})^{j_2} & \dots & (\alpha^{m_0+1})^{j_{d-1}} \\ & & \vdots & \\ (\alpha^{m_0+d-2})^{j_1} & (\alpha^{m_0+d-2})^{j_2} & \dots & (\alpha^{m_0+d-2})^{j_{d-1}} \end{vmatrix} = \\ &= \alpha^{m_0(j_1+j_2+\dots+j_{d-1})} \begin{vmatrix} 1 & 1 & \dots & 1 \\ \alpha^{j_1} & \alpha^{j_2} & \dots & \alpha^{j_{d-1}} \\ & & \vdots & \\ (\alpha^{j_1})^{d-2} & (\alpha^{j_2})^{d-2} & \dots & (\alpha^{j_{d-1}})^{d-2} \end{vmatrix} = \\ &= \alpha^{m_0(j_1+j_2+\dots+j_{d-1})} \prod_{i>k} (\alpha^{j_i} - \alpha^{j_k}) \neq 0 \end{aligned}$$

pentru că $j_i \neq j_k$ dacă $i \neq k$.

Deci orice submulțime formată cu maxim $d-1$ coloane din H este liniar independentă de unde – conform Teoremei 2.4 (Prelegerea 2) – distanța minimă a codului este cel puțin d . q.e.d.

Exemplul 12.1 Dacă se lucrează în binar, puterile pare ale rădăcinilor se pot omite din listă (dacă α^i este rădăcină, atunci și α^{2i} este rădăcină – conform Teoremei 11.9, Prelegerea 11). Atunci un cod BCH binar corector de 3 erori este definit de rădăcinile β, β^3, β^5 , unde β este un element primitiv dintr-o extensie a lui Z_2 . De exemplu, dacă $\beta = \alpha$, rădăcină a polinomului $1 + X + X^4$, se obține codul din Exemplul 11.3, Prelegerea 11.

Exemplul 12.2 Fie $GF(3^2)$ generat de rădăcina α a polinomului ireductibil $g(X) = 2 + X + X^2$. Deoarece $\text{ord}(\alpha) = 8$, orice cod generat va avea lungimea 8. Să construim codul BCH de rădăcini $\alpha^2, \alpha^3, \alpha^4$.

Polinoamele minimale sunt:

$$\begin{aligned} m_2(X) &= 1 + X^2 \text{ cu rădăcini } \alpha^2, \alpha^6; \\ m_3(X) &= 2 + X + X^2 \text{ cu rădăcini } \alpha^3, \alpha \\ m_4(X) &= 1 + X, \text{ numai cu rădăcina } \alpha^4. \end{aligned}$$

Deci polinomul generator este

$$g(X) = (1 + X^2)(2 + X + X^2)(1 + X) = 2 + X^2 + X^3 + 2X^4 + X^5.$$

Codul este un $(8, 3)$ - cod ciclic. Deoarece printre rădăcinile sale este și α , se poate considera $m_0 = 1$, $d = 5$, deci codul poate corecta cel mult două erori independente.

De remarcat că dacă folosim acest cod drept cod liniar, cum ponderea minimă a cuvintelor sale este 4 (de exemplu 02010201 este cuvânt - cod), distanța minimă este 4, deci codul nu poate corecta decât o singură eroare. În general, tratarea codurilor ciclice sub formă de coduri liniare scade de obicei puterea de corecție a erorilor independente.

Numerele m, m_0, d, q constituie parametrii codului BCH. Ideea generală este de a-l mări pe d , pentru a putea corecta cât mai multe combinații de erori. Acest lucru este posibil totdeauna, deoarece m și q sunt alese arbitrar.

Să observăm că pentru $\alpha \in GF(q^m)$ primitiv, toate puterile lui α până la $q^m - 1$ sunt distincte. În cazul binar, avem următorul rezultat:

Teorema 12.3 *Fie m un număr natural oarecare și $\alpha \in GF(2^m)$ primitiv. Atunci există un cod BCH de lungime $n = 2^m - 1$ care corectează orice combinație de $t \leq 2^{m-1} - 1$ erori independente, folosind cel mult mt poziții de control.*

Demonstrație: Vom considera codul binar definit de rădăcinile $\alpha, \alpha^2, \dots, \alpha^{2t}$. Acesta este un cod BCH pentru $q = 2$, $m_0 = 1$, $d = 2t + 1$, $n = 2^m - 1$ și cu distanța minimă cel puțin $2t + 1$. Pentru ca rădăcinile menționate mai sus să fie distincte, trebuie ca $2t \leq n - 1 = 2^m - 1$.

Să observăm că orice putere pară a lui α este rădăcină a unui polinom minimal corespunzător unei puteri impare anterioare a lui α . Deci este suficient să dăm la început doar rădăcinile $\alpha, \alpha^3, \dots, \alpha^{2t-1}$ ale cuvintelor codului binar.

Avem – după definiție: $g(X) = cmmmc\{m_1(X), m_3(X), \dots, m_{2t-1}(X)\}$.

Pe de altă parte, $m_i(X) | X^n - 1 = X^{2^m-1} - 1$ și deci – conform Teoremei 11.7 (Prelegerea 11), $grad(m_i(X)) \leq m$. Din cele două relații rezultă $grad(g(X)) \leq mt$.

Cum $grad(g(X))$ dă numărul de simboluri de control, teorema este demonstrată.q.e.d.

12.2 Algoritmul Peterson de decodificare

Fie codul BCH definit de rădăcinile $\alpha^{m_0}, \alpha^{m_0+1}, \dots, \alpha^{m_0+2t-1}$ ($d = 2t + 1$) cu $\alpha \in GF(q^m)$.

Un asemenea cod corectează orice combinație de maxim t erori independente.

Fie $\{f(X)\}$ un cuvânt - cod transmis și

$$\{r(X)\} = \{f(X) + e(X)\} = \{f(X)\} + \{e(X)\}$$

secvența recepționată, unde $\{e(X)\}$ este vectorul - eroare.

Avem, pentru $j = m_0, m_0 + 1, \dots, m_0 + 2t - 1$:

$$S_j = r(\alpha^j) = f(\alpha^j) + e(\alpha^j) = e(\alpha^j).$$

S_j se numesc *componentele sindromului*.

La decodificare, eroarea-tip \mathbf{e} este determinată complet dacă se știu:

- Valorile componentelor sale $Y_i \in Z_q$;
- Localizările lor $X_i \in GF(q^m)$.

Un cuvânt din $GH(q^m)$ este un vector cu q^m componente; vom nota pozițiile acestora folosind puterile lui α^i : pe poziția i se află coeficientul lui α^{i-1} ($1 \leq i \leq q^m$). Deci α^{i-1} va localiza a i -a componentă dintr-un cuvânt.

Deci, dacă au intervenit t erori, eroarea - tip este complet caracterizată de cunoașterea celor t perechi nenule (Y_i, X_i) ($1 \leq i \leq t$).

Putem conveni (evident) că

$$Y_i = 0 \quad \Longleftrightarrow \quad X_i = 0.$$

De remarcat că X_i nu reprezintă α^i ci un α^j arbitrar, care va trebui aflat.

Deci, a găsi pe $\{e(X)\}$ revine la a determina $2t$ elemente X_i, Y_i ($1 \leq i \leq t$), grupate în t perechi. Vom considera $Y_i = 0 \forall i$ ($t < i \leq q^m - 1$). Avem – conform notației polinomiale

$$S_j = e(\alpha^j) = \sum_{i=1}^t Y_i X_i^j, \quad (m_0 \leq j \leq m_0 + 2t - 1)$$

Atunci

$$(S_j)^q = \left(\sum_{i=1}^t Y_i X_i^j \right)^q = \sum_{i=1}^t Y_i^q X_i^{jq} = \sum_{i=1}^t Y_i X_i^{jq} = S_{jq}.$$

Deci, determinarea polinomului - eroare $\{e(X)\}$ se reduce la aflarea soluțiilor (Y_i, X_i) , ($1 \leq i \leq t$) ale sistemului de ecuații

$$\sum_{i=1}^t Y_i X_i^j = S_j, \quad m_0 \leq j \leq m_0 + 2t - 1, \quad (1)$$

Dacă au intervenit k ($k < t$) erori, se completează cele k perechi nule (Y_i, X_i) cu alte $t - k$ perechi nule $(0, 0)$. De fapt, acest număr k nu este cunoscut a priori; de aceea, se pleacă de la t erori potențiale (maximul pe care codul BCH îl poate corecta în mod cert), caracterizate de t perechi (Y_i, X_i) . Problema care se pune este deci de a-l determina pe k și cele $2k$ necunoscute Y_i, X_i ($1 \leq i \leq k$).

Orice metodă de corectare a erorilor revine la rezolvarea sistemului de ecuații (1), care este un sistem neliniar de $2t$ ecuații cu $2t$ necunoscute, problemă NP - completă. Să încercăm o abordare mai simplă a acestei probleme.

Fie polinomul

$$(X_1 - X)(X_2 - X) \dots (X_t - X) = \sigma_t + \sigma_{t-1}(-X) + \dots + \sigma_1(-X)^{t-1} + (-X)^t.$$

unde σ_j sunt funcțiile simetrice date de relațiile lui Viète.

Dacă în această relație se ia $X = X_i$ ($1 \leq i \leq t$), se ajunge la

$$X_i^t + (-1)\sigma_1 X_i^{t-1} + \dots + (-1)^{t-1} \sigma_{t-1} X_i + (-1)^t \sigma_t = 0, \quad (1 \leq i \leq t).$$

Înmulțim cu $Y_i X_i^j$ și sumăm după i . Se obține:

$$\sum_{i=1}^t Y_i X_i^{j+t} + (-1)\sigma_1 \sum_{i=1}^t Y_i X_i^{j+t-1} + \dots + (-1)^{t-1} \sigma_{t-1} \sum_{i=1}^t Y_i X_i^{j+1} + (-1)^t \sigma_t \sum_{i=1}^t Y_i X_i^j = 0, \\ j = m_0, m_0 + 1, \dots, m_0 + t - 1$$

relație care se poate scrie

$$S_j (-1)^t \sigma_t + S_{j+1} (-1)^{t-1} \sigma_{t-1} + \dots + S_{j+t-1} (-1) \sigma_1 + S_{j+t} = 0, \quad (2) \\ m_0 \leq j \leq m_0 + t - 1$$

Algoritmul lui Peterson de corectare a k ($k \leq t$) erori într-un cod BCH este:

1. Se rezolvă sistemul (2) în care necunoscutele sunt $(-1)^p \sigma_p$, ($1 \leq p \leq t$);
2. Se înlocuiesc valorile aflate în ecuația

$$X^t + (-1)\sigma_1 X^{t-1} + \dots + (-1)^{t-1} \sigma_{t-1} X + (-1)^t \sigma_t = 0$$

și se află cele t rădăcini $X_1, X_2, \dots, X_t \in GF(q^m)$ ale sale;

3. Cu aceste valori introduse în sistemul (1), se determină din primele t ecuații valorile $Y_1, Y_2, \dots, Y_t \in Z_q$;
4. Dacă (Y_i, α^{ji}) ($1 \leq i \leq k$) sunt cele $k \leq t$ valori nenule ale soluției obținute, atunci:
 - Pentru $k = 0$, cuvântul recepționat este $\{r(X)\}$ (fără erori);

- Pentru $0 < k \leq t$, fie $e(X) = \sum_{i=1}^k Y_i X^{ji}$.

Cuvântul - cod decodificat este $\{f(X)\} = \{r(X)\} - \{e(X)\}$.

De remarcat că în cazul binar, pasul (3) nu mai este necesar, deoarece

$$Y_i = 1, \forall i \leq k.$$

Pentru corectitudinea algoritmului, trebuie studiate compatibilitățile sistemelor de ecuații (1) și (2).

Teorema 12.4 Sistemul $\sum_{i=1}^t Y_i X_i^j = S_j$ ($m_0 \leq j \leq m_0 + t - 1$) de necunoscute Y_i este compatibil dacă și numai dacă au intervenit t erori.

Demonstrație: Considerat ca sistem în Y , sistemul este liniar; determinantul său principal este

$$\begin{vmatrix} X_1^{m_0} & X_2^{m_0} & \dots & X_t^{m_0} \\ X_1^{m_0+1} & X_2^{m_0+1} & \dots & X_t^{m_0+1} \\ X_1^{m_0+2} & X_2^{m_0+2} & \dots & X_t^{m_0+2} \\ \vdots & \vdots & \ddots & \vdots \\ X_1^{m_0+t-1} & X_2^{m_0+t-1} & \dots & X_t^{m_0+t-1} \end{vmatrix} = X_1^{m_0} X_2^{m_0} \dots X_t^{m_0} \begin{vmatrix} 1 & 1 & \dots & 1 \\ X_1 & X_2 & \dots & X_t \\ X_1^2 & X_2^2 & \dots & X_t^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^{t-1} & X_2^{t-1} & \dots & X_t^{t-1} \end{vmatrix}$$

$$= X_1^{m_0} X_2^{m_0} \dots X_t^{m_0} \prod_{1 \leq j < i \leq t} (X_i - X_j)$$

Acesta este nenul numai dacă au intervenit t erori, pentru că atunci toți X_i sunt distincți și nenuli. q.e.d.

Teorema 12.5 *Matricea*
$$M = \begin{pmatrix} S_{m_0} & S_{m_0+1} & \dots & S_{m_0+t-1} \\ S_{m_0+1} & S_{m_0+2} & \dots & S_{m_0+t} \\ & & \vdots & \\ S_{m_0+t-1} & S_{m_0+t} & \dots & S_{m_0+2t-1} \end{pmatrix}$$

este nesingulară dacă au intervenit t erori și este singulară dacă au intervenit mai puțin de t erori.

Demonstrație: Ținând cont de forma sistemului (1), este posibilă descompunerea matricii M în produs de 3 matrici $M = ABA^T$ unde:

$$A = \begin{pmatrix} 1 & 1 & \dots & 1 \\ X_1 & X_2 & \dots & X_t \\ X_1^2 & X_2^2 & \dots & X_t^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^{t-1} & X_2^{t-1} & \dots & X_t^{t-1} \end{pmatrix} \quad B = \begin{pmatrix} Y_1 X_1^m & 0 & \dots & 0 \\ 0 & Y_2 X_2^m & \dots & 0 \\ & & \vdots & \\ 0 & 0 & \dots & Y_t X_t^m \end{pmatrix}.$$

M este nesingulară dacă cele două matrici din descompunere sunt nesingulare. Matricea A este nesingulară dacă și numai dacă valorile X_i ($1 \leq i \leq t$) sunt distincte și nenule, deci dacă au intervenit t erori. Matricea B este nesingulară dacă toate elementele de pe diagonala sa principală sunt nenule, deci dacă toate perechile (Y_i, X_i) ($1 \leq i \leq t$) sunt nenule – adică au apărut t erori. q.e.d.

Aplicarea algoritmului Peterson se face în felul următor:

1. Folosind polinomul recepționat $\{r(X)\}$, se determină

$$S_j = r(\alpha^j) \quad (m_0 \leq j \leq m_0 + 2t - 1)$$

2. Se studiază compatibilitatea sistemului

$$S_j(-1)^t \sigma_t + S_{j+1}(-1)^{t-1} \sigma_{t-1} + \dots + S_{j+t-1}(-1) \sigma_1 + S_{j+t} = 0, \\ m_0 \leq j \leq m_0 + t - 1.$$

Dacă matricea M a acestui sistem este nesingulară, înseamnă că au apărut t erori. Dacă M este singulară, atunci se observă că $\sigma_t = 0$ (deoarece $\sigma_t = X_1 X_2 \dots X_t$) și – după ce se înlocuiește $\sigma_t = 0$ – se obține un nou sistem liniar format din $t - 1$ ecuații (neglijând prima sau ultima din ecuațiile de mai sus) cu $t - 1$ necunoscute.

Se studiază matricea acestui nou sistem. Dacă ea este nesingulară, înseamnă că au apărut $t - 1$ erori. Dacă și aceasta este singulară, cum $\sigma_{t-1} = 0$, se repetă procedeul.

Primul sistem compatibil (cu matricea nesingulară) care se obține va da numărul k ($k \leq t$) de erori care au intervenit în transmisie, precum și valorile

$$(-1)\sigma_1, (-1)^2\sigma_2, \dots, (-1)^k\sigma_k.$$

3. Se introduc valorile determinate mai sus în polinomul

$$P = X^k + (-1)\sigma_1 X^{k-1} + \dots + (-1)^{k-1} \sigma_{k-1} X + (-1)^k \sigma_k$$

și se află cele k rădăcini $X_1, X_2, \dots, X_k \in GF(q^m)$ ale ecuației $P = 0$ (prin diverse metode, eventual chiar prin simpla verificare a ecuației cu toate elementele din $GF(q^m)$). Aceste rădăcini vor da localizările erorilor.

De remarcat că ordinea de notare a rădăcinilor X_1, X_2, \dots, X_t este neesențială.

4. Se introduc aceste soluții în ecuațiile sistemului

$$\sum_{i=1}^k Y_i X_i^j = S_j, \quad j = m_0, m_0 + 1, \dots, m_0 + k - 1$$

și se determină valorile erorilor Y_1, Y_2, \dots, Y_k .

5. Se calculează astfel polinomul eroare $e(X) = \sum_{i=1}^k Y_i X^{j_i}$, unde $X_i = \alpha^{j_i}$ ($1 \leq i \leq k$).

Acesta se scade din polinomul recepționat și se obține cuvântul - cod transmis:

$$\{r(X)\} - \{e(X)\} = \{r(X) - e(X)\} = \{f(X)\}$$

Exemplul 12.3 Fie codul BCH definit de rădăcinile $\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6$ unde $\alpha \in GF(2^4)$ este element primitiv, rădăcină a polinomului $1 + X + X^4$ (Exemplul 11.3, Prelegerea 11 și Exemplul 20.3).

Deoarece $t = 3$, acest cod este capabil să corecteze maxim 3 erori independente.

Să presupunem că au apărut două erori, pe pozițiile α^3 și α^{10} (adică pe pozițiile 4 și 11) ale cuvântului transmis. Ele sunt reprezentate prin polinomul eroare $e(X) = X^3 + X^{10}$.

Să calculăm sindromul (pentru operațiile folosite, a se vedea Exemplele 8.8 și 8.9, Prelegerea 8):

$$\begin{aligned} S_1 &= r(\alpha) = e(\alpha) = \alpha^3 + \alpha^{10} = \alpha^{12} \\ S_2 &= (S_1)^2 = \alpha^{24} = \alpha^9 \\ S_3 &= r(\alpha^3) = e(\alpha^3) = \alpha^9 + \alpha^{30} = \alpha^9 + \alpha^0 = \alpha^7 \\ S_4 &= (S_2)^2 = \alpha^{18} = \alpha^3 \\ S_5 &= r(\alpha^5) = e(\alpha^5) = \alpha^{15} + \alpha^{50} = \alpha^0 + \alpha^5 = \alpha^{10} \\ S_6 &= (S_3)^2 = \alpha^{14}. \end{aligned}$$

Observăm că într-o situație reală, nu se va ști unde sunt localizate aceste erori și deci - în cazul în care au intervenit două erori pe pozițiile α^3 și α^{10} (care nu se cunosc încă) - din vectorul recepționat se obține direct sindromul

$$S = (S_1, S_2, S_3, S_4, S_5, S_6) = (\alpha^{12}, \alpha^9, \alpha^7, \alpha^3, \alpha^{10}, \alpha^{14}),$$

fără să știm din combinațiile căror localizări ale erorilor provin aceste componente ale sindromului.

Să considerăm sistemul de ecuații

$$\begin{cases} S_1\sigma_3 + S_2\sigma_2 + S_3\sigma_1 = S_4 \\ S_2\sigma_3 + S_3\sigma_2 + S_4\sigma_1 = S_5 \\ S_3\sigma_3 + S_4\sigma_2 + S_5\sigma_1 = S_6 \end{cases} \quad \text{adică} \quad \begin{cases} \alpha^{12}\sigma_3 + \alpha^9\sigma_2 + \alpha^7\sigma_1 = \alpha^3 \\ \alpha^9\sigma_3 + \alpha^7\sigma_2 + \alpha^7\sigma_1 = \alpha^{10} \\ \alpha^7\sigma_3 + \alpha^3\sigma_2 + \alpha^{10}\sigma_1 = \alpha^{14} \end{cases}$$

Înmulțim cele trei ecuații ale sistemului respectiv cu

$$\alpha^{-7} = \alpha^8, \quad \alpha^{-3} = \alpha^{12}, \quad \alpha^{-10} = \alpha^5$$

și obținem:

$$\begin{cases} \alpha^5\sigma_3 + \alpha^2\sigma_2 + \sigma_1 = \alpha^{11} \\ \alpha^6\sigma_3 + \alpha^4\sigma_2 + \sigma_1 = \alpha^7 \\ \alpha^{12}\sigma_3 + \alpha^8\sigma_2 + \sigma_1 = \alpha^4 \end{cases}$$

Adunăm prima ecuație la ultimele două și atunci

$$\begin{cases} \alpha^9\sigma_3 + \alpha^{10}\sigma_2 = \alpha^8 \\ \alpha^{14}\sigma_3 + \alpha^0\sigma_2 = \alpha^{13} \end{cases}$$

Deoarece a doua ecuație se obține din prima prin înmulțire cu α^5 , ele nu sunt independente; deci au apărut mai puțin de 3 erori. Facem $\sigma_3 = 0$ și a doua ecuație va da $\sigma_2 = \alpha^{13}$. Din prima ecuație a sistemului se ajunge la $\alpha^0 + \sigma_1 = \alpha^{11}$, adică $\sigma_1 = \alpha^{12}$.

Deci, în acest caz $k = 2$, $\sigma_1 = \alpha^{12}$, $\sigma_2 = \alpha^{13}$. Se ajunge la ecuația $X^2 + \sigma_1 X + \sigma_2 = X^2 + \alpha^{12} X + \alpha^{13} = 0$, care are ca rădăcini $X_1 = \alpha^3$, $X_2 = \alpha^{10}$.

Am redescoperit astfel cele două localizări ale erorilor. Codul fiind binar, corectarea se face imediat: se modifică (din 1 în 0 sau din 0 în 1) biții de pe pozițiile 4 respectiv 11 din cuvântul recepționat.

Exemplul 12.4 Fie codul BCH definit de rădăcinile $\alpha, \alpha^2, \alpha^3, \alpha^4$ unde $\alpha \in GF(2^4)$ este rădăcina polinomului $1 + X + X^4$. Acest cod are $n = 15$, $m_0 = 1$, $m_0 + 2t - 1 = 4$, deci $t = 2$; codul poate corecta în mod sigur cel mult două erori independente.

Să presupunem că s-a recepționat cuvântul

$$\mathbf{r} = 100100110000100.$$

Să determinăm cuvântul - cod transmis.

Polinomul corespunzător este $r(X) = 1 + X^3 + X^6 + X^7 + X^{12}$, iar componentele sindromului sunt

$$\begin{aligned} S_1 &= r(\alpha) = \alpha^0, & S_2 &= r(\alpha^2) = (S_1)^2 = \alpha^0, \\ S_3 &= r(\alpha^3) = \alpha^4, & S_4 &= r(\alpha^4) = (S_2)^2 = \alpha^0. \end{aligned}$$

Trebuie rezolvat sistemul

$$\begin{cases} S_1\sigma_2 + S_2\sigma_1 = S_3 \\ S_2\sigma_2 + S_3\sigma_1 = S_4 \end{cases} \quad \text{adică} \quad \begin{cases} \alpha^0\sigma_2 + \alpha^0\sigma_1 = \alpha^4 \\ \alpha^0\sigma_2 + \alpha^4\sigma_1 = \alpha^0 \end{cases},$$

de unde se obține $\sigma_1 = \alpha^0$, $\sigma_2 = \alpha$.

Polinomul de localizare a erorilor este $X^2 + \sigma_1 X + \sigma_2 = X^2 + X + \alpha$, cu rădăcinile $X_1 = \alpha^7$, $X_2 = \alpha^9$. Au apărut deci două erori, pe pozițiile 8 și 10. Pentru că suntem în cazul binar, $Y_1 = Y_2 = 1$; vectorul - eroare obținut este $e(X) = X^7 + X^9$, deci cuvântul - cod transmis este

$$\{f(X)\} = \{r(X)\} - \{e(X)\} = \{r(X) + e(X)\} = 1 + X + X^6 + X^9 + X^{12}$$

adică $\mathbf{f} = 100100100100100$.

12.3 Localizarea erorilor la codurile BCH binare

După cum s-a observat, în cazul codurilor BCH binare este suficient să se localizeze eroarea; corectarea se face imediat prin complementarea biților aflați pe pozițiile respective. După determinarea valorilor $\sigma_1, \sigma_2, \dots, \sigma_t$, algoritmul Peterson nu explicitează modul de rezolvare a ecuației de localizare al erorii, care uneori poate fi dificil de prelucrat.

Un procedeu eficient de determinare a pozițiilor erorilor pentru codurile BCH binare a fost realizat de Chien.

Fie polinomul (de localizare a erorii):

$$S(X) = (1 - X_1 X)(1 - X_2 X) \dots (1 - X_t X) = \sigma_0 + \sigma_1 X + \sigma_2 X^2 + \dots + \sigma_t X^t$$

unde

$$\begin{aligned}\sigma_0 &= 1 \\ \sigma_1 &= X_1 + X_2 + \dots + X_t\end{aligned}$$

$$\dots \\ \sigma_t = X_1 X_2 \dots X_t$$

(operațiile fiind efectuate în binar, semnul $-$ este asimilat cu $+$).

Rădăcinile acestui polinom sunt inversele localizărilor erorilor X_i^{-1} ($1 \leq i \leq t$).

Procedeeul lui Chien de corectare a erorilor – pentru codurile BCH binare – este următorul:

Fie $\mathbf{r} = r_0 r_1 \dots r_{n-1}$ cuvântul recepționat; componentele lui se vor decodifica pas - cu - pas (pe măsură ce sunt recepționate).

Să considerăm primul bit primit r_{n-1} și să cercetăm dacă el este corect sau nu. Aceasta este echivalent cu a verifica dacă α^{n-1} este o localizare a erorii, sau dacă $\alpha^{-(n-1)} = \alpha$ este o rădăcină a polinomului $S(X)$ de localizare a erorii.

Dacă $\sigma_1 \alpha + \sigma_2 \alpha^2 + \dots + \sigma_t \alpha^t = \mathbf{1}$ adică $S(\alpha) = \mathbf{0}$, atunci a intervenit o eroare pe poziția α^{n-1} și deci componenta r_{n-1} trebuie corectată.

Dacă $\sigma_1 \alpha + \sigma_2 \alpha^2 + \dots + \sigma_t \alpha^t \neq \mathbf{1}$ adică $S(\alpha) \neq \mathbf{0}$, atunci r_{n-1} s-a recepționat corect.

Procedeeul se repetă: în general componenta recepționată r_{n-s} este corectă dacă $\alpha^{-(n-s)} = \alpha^s$ nu este rădăcină a polinomului $S(X)$, adică

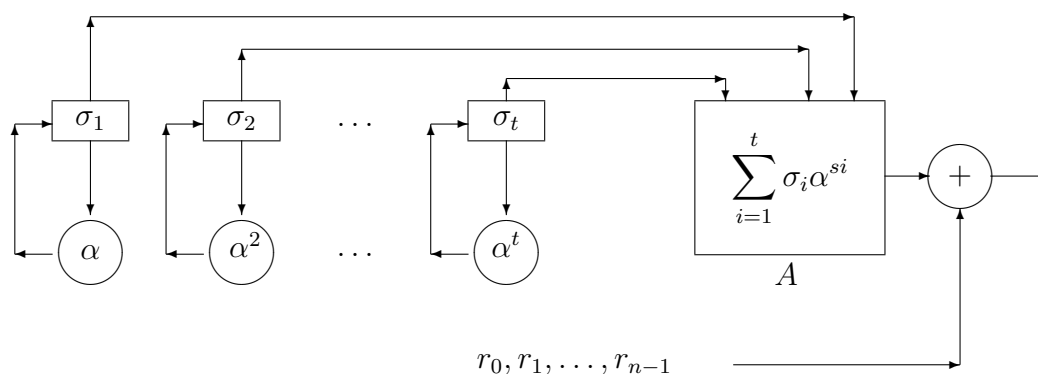
$$\sigma_1 \alpha^s + \sigma_2 \alpha^{2s} + \dots + \sigma_t \alpha^{ts} \neq \mathbf{1}.$$

În caz contrar, r_{n-s} va fi corectată în $r_{n-s} + 1 \pmod{2}$.

Algoritmul lui Chien este deci:

1. $s := 0$;
2. **while** $s \leq n$ **do**
 - (a) $s := s + 1$;
 - (b) Se calculează $P := \sigma_1 \alpha^s + \sigma_2 \alpha^{2s} + \dots + \sigma_t \alpha^{ts}$;
 - (c) Se decodifică $a_{n-s} := r_{n-s} + v$ unde $v = \begin{cases} 0 & \text{dacă } P \neq \mathbf{1} \\ 1 & \text{dacă } P = \mathbf{1} \end{cases}$

Circuitul liniar care implementează acest algoritm este:



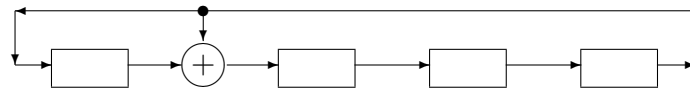
Cele t circuite din stânga realizează la fiecare tact multiplicarea cu α^i .

La momentul inițial blocul A calculează suma $\sigma_1\alpha + \sigma_2\alpha^2 + \dots + \sigma_t\alpha^t$ dând la ieșire valoarea 1 (scalar) dacă această sumă este egală cu $\mathbf{1} = (1, 0, \dots, 0)$, și 0 (scalar) dacă suma anterioară dă orice element din $GF(2^m)$ diferit de $\mathbf{1}$.

În funcție de acest rezultat, componenta r_{n-1} este schimbată, respectiv păstrată.

La momentul următor, blocul A calculează suma $\sigma_1\alpha^2 + \sigma_2\alpha^4 + \dots + \sigma_t\alpha^{2t}$. Dacă această sumă este $\mathbf{1}$, α^2 este rădăcina a polinomului de localizare a erorii $S(X)$, deci $\alpha^{-2} = \alpha^{n-2}$ este o localizare a erorii. În caz contrar, suma este diferită de $\mathbf{1}$. La ieșirea din blocul A se obține scalarul 1 respectiv 0, componenta r_{n-2} fiind modificată respectiv păstrată.

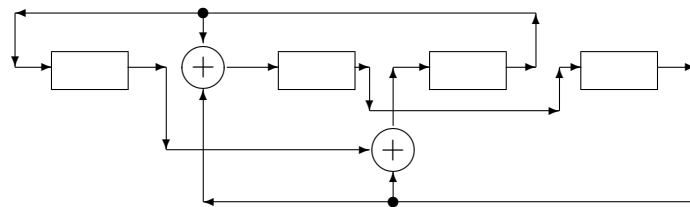
Exemplul 12.5 Fie $\alpha \in GF(2^4)$ primitiv, rădăcina polinomului $1 + X + X^4$. Să presupunem că se știu σ_1 și σ_2 . Circuitul care multiplică cu α este (Prelegerea 8, Figura 8.4) circuitul de împărțire la $m(X) = 1 + X + X^4$:



Circuitul de înmulțire cu α^2 este – conform egalității

$$\alpha^2(a_1 + a_2\alpha + a_3\alpha^2 + a_4\alpha^3) = a_1\alpha^2 + a_2\alpha^3 + a_3\alpha^4 + a_4\alpha^5 = a_3 + (a_3 + a_4)\alpha + (a_1 + a_4)\alpha^2 + a_2\alpha^3$$

următorul (vezi și Exemplul 8.10, Prelegerea 8):



Inițial în elementele de înmagazinare ale celor două circuite se află σ_1 respectiv σ_2 .

Pentru cazul binar vom nota cu

$$\begin{matrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_n \end{matrix} \begin{matrix} \longrightarrow \\ \longrightarrow \\ \longrightarrow \end{matrix} \bigg) \longrightarrow \mathbf{x}_1 \vee \mathbf{x}_2 \vee \dots \vee \mathbf{x}_n$$

poarta OR care dă valoarea (scalar) 0 dacă și numai dacă

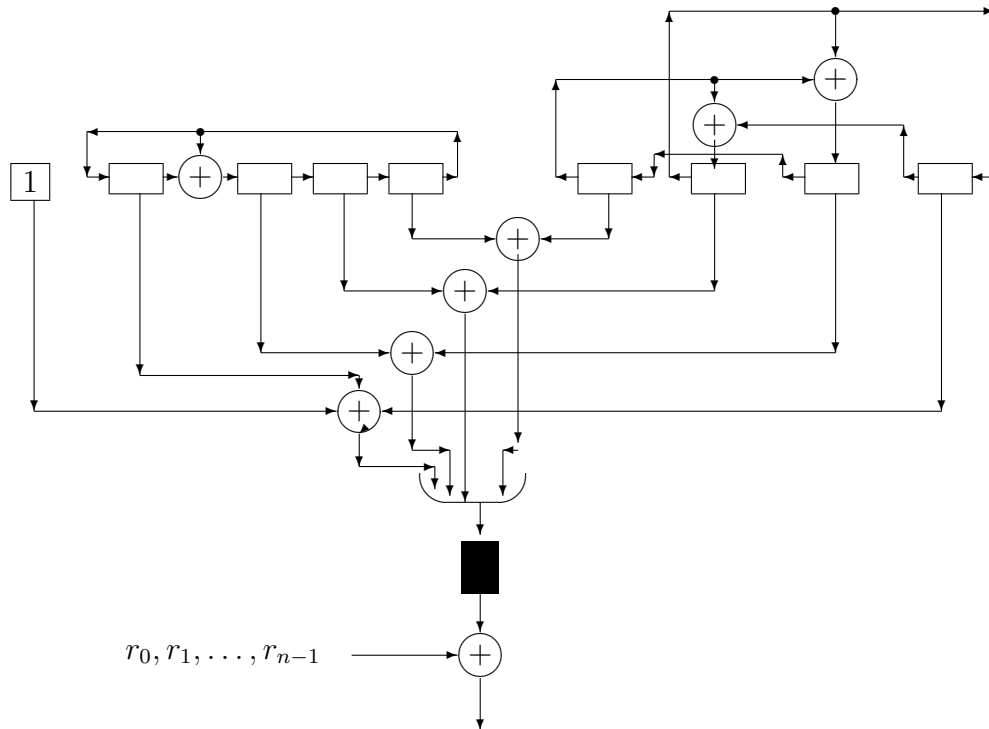
$$\mathbf{x}_1 = \mathbf{x}_2 = \dots = \mathbf{x}_n = \mathbf{0},$$

și cu

$$x \longrightarrow \blacksquare \longrightarrow \bar{x}$$

poarta NOT.

Atunci, circuitul de implementare a procedurii Chien de localizare a erorii este:



În acest circuit, cele patru elemente de înmagazinare din stânga formează circuitul de înmulțire cu α (unde inițial se introduce σ_1), iar cele patru elemente din dreapta formează circuitul de înmulțire cu α^2 , aranjat în ordine inversă (unde σ_2 se introduce inițial de la dreapta spre stânga).

De asemenea, există un element de înmagazinare care conține bitul 1, element pe care îl furnizează la fiecare tact.

Inițial, circuitul funcționează odată la momentul 0 fără a introduce nimic de pe canal (pentru "amorsarea" în elementele de înmagazinare a coeficienților pentru $\sigma_1\alpha$ respectiv $\sigma_2\alpha^2$).

La momentul următor se începe procesul efectiv de decodificare. Dacă r_{n-1} a fost perturbat, atunci α^{n-1} este localizarea erorii, adică α este rădăcina a polinomului de localizare a erorii și avem $\mathbf{1} + \sigma_1\alpha + \sigma_2\alpha^2 = \mathbf{0}$. În acest caz, la intrarea în "sumatorul" OR este vectorul $\mathbf{0} = (0, 0, 0, 0)$, iar la ieșire scalarul 0. După negare se obține scalarul 1, care se adună (pentru corecție) la r_{n-1} .

Dacă r_{n-1} este corect, la intrarea în sumator se obține o valoare nenulă, deci ieșirea va fi 1, care prin negare este 0 și r_{n-1} rămâne nemodificat.

12.4 Exerciții

12.1 Să se verifice valabilitatea descompunerii $M = ABA^T$ din demonstrația Teoremei 20.2

12.2 Fie $\alpha \in GF(2^m)$ ($m > 2$) primitiv. Să se arate că polinoamele minimale $m_1(X)$ și $m_3(X)$ au același grad.

12.3 Construiți un cod BCH binar corector de 2 erori de lungime $n = 11$.

12.4 Construiți un cod BCH ternar de lungime $n = 26$ și $d = 5$.

12.5 Să se determine polinomul generator și cel de control pentru codul BCH binar corector de 3 erori de lungime

$$(a) n = 15 \quad (b) n = 31.$$

Codificați mesajul de informație 111...1.

12.6 Caracterizați codurile BCH care conțin toate elementele extensiei Galois $GF(q^m)$.

12.7 Dacă $g(X) = 1 + X^4 + X^6 + X^7 + X^8$ este polinomul generator al unui cod BCH binar de lungime 15, să se cerceteze dacă următoarele secvențe sunt cuvinte - cod:

$$(a) 011001011000010, \quad (b) 000111010000110, \\ (c) 011100000010001, \quad (d) 111111111111111.$$

12.8 Aflați rădăcinile polinoamelor cu coeficienți în $GF(2^4)$ (α este rădăcina polinomului $1 + X + X^4$):

$$\begin{array}{lll} X^2 + \alpha^4 X + \alpha^{13}, & X^2 + \alpha^7 X + \alpha^2, & X^2 + \alpha^2 X + \alpha^5, \\ X^2 + \alpha^6, & X^2 + \alpha^2 X, & X^2 + X + \alpha^8. \end{array}$$

12.9 Definim codul BCH de polinom generator $g(X) = m_1(X)m_3(X)$ peste $GF(2^4)$ folosind elementul primitiv α , rădăcină a polinomului $1 + X + X^4$. Se recepționează cuvintele \mathbf{r} și se calculează sindromurile \mathbf{rH} . Știind aceste sindromuri, localizați erorile (dacă se poate):

$$\begin{array}{llll} a) 01000101, & b) 11101000, & c) 11001101, & d) 01000000, \\ e) 00000100, & f) 10100100, & g) 00111101, & h) 00000000. \end{array}$$

12.10 Pentru codul BCH definit în exercițiul anterior, decodificați mesajele:

$$\begin{array}{lll} a) 1100000000000000, & b) 000010000100001, & c) 010001010100000, \\ d) 110011100111000, & e) 110011100100000, & f) 111000000000001, \\ g) 1011100000000000, & h) 101010010110001, & i) 010000100000000, \\ j) 010101001011000, & k) 110111011101100, & m) 101110000001000, \\ n) 111001011000000, & p) 000111010000110. \end{array}$$

12.11 În codul BCH binar de lungime $n = 15$ corector de 3 erori (vezi Exemplele 11.1 și 11.8), să se decodifice mesajele:

$$(a) 0010001000000000, \quad (b) 101011001000000.$$

12.12 În codul BCH peste Z_3 de lungime $n = 8$, corector de 2 erori, să se decodifice mesajele

$$\begin{array}{lll} (a) 00100000 & (b) 12121212 & (c) 11211122 \\ (d) 02020202 & (e) 10201020 & (f) 22110011. \end{array}$$

12.13 Fie $\alpha \in GF(2^5)$ primitiv, rădăcină a polinomului $1 + X^2 + X^5$. Se definește un cod BCH binar de lungime $n = 31$ și $d = 5$.

1. Să se construiască polinomul generator al codului: $g(X) = m_1(X)m_3(X)$.

2. Să se decodifice mesajul 1001011011110000110101010111111.

Capitolul 13

Coduri Reed - Solomon

13.1 Definirea codurilor RS

O clasă foarte interesantă de coduri ciclice a fost definită în 1960 de Reed și Solomon. Numite în articolul inițial *coduri polinomiale*, Peterson arată un an mai târziu că ele sunt de fapt un caz particular de coduri BCH.

Construcția dată de Reed - Solomon a fost următoarea:

Fie $m \geq 2$ și $\alpha \in GF(q^m)$ primitiv (deci $ord(\alpha) = n = q^m - 1$).

Dacă $\mathbf{a}_0\mathbf{a}_1 \dots \mathbf{a}_{k-1} \in GF(q^m)^k$ este secvența de informație, se definește polinomul $h(X) = \mathbf{a}_0 + \mathbf{a}_1X + \dots + \mathbf{a}_{k-1}X^{k-1} \in GF(q^m)[X]$. Codul polinomial va fi format de toate cuvintele de forma $h(\mathbf{1})h(\alpha) \dots h(\alpha^{n-k})$.

Teorema 13.1 *Un cod polinomial este un cod BCH peste $GF(q^m)$, definit de rădăcinile $\alpha, \alpha^2, \dots, \alpha^{n-k}$.*

Demonstrație: În algebra polinoamelor modulo $X^n - 1$ ($n = q^m - 1$) să considerăm clasa $\{f(X)\}$ asociată cuvântului - cod $h(\mathbf{1})h(\alpha) \dots h(\alpha^{n-1})$:

$$f(X) = h(\mathbf{1}) + h(\alpha)X + \dots + h(\alpha^{n-1})X^{n-1} = \mathbf{a}_0(1 + X + X^2 + \dots + X^{n-1}) + \mathbf{a}_1(1 + \alpha X + \alpha^2 X^2 + \dots + \alpha^{n-1} X^{n-1}) + \mathbf{a}_2(1 + \alpha^2 X + \alpha^4 X^2 + \dots + (\alpha^2 X)^{n-1}) + \dots + \mathbf{a}_{k-1}(1 + \alpha^{k-1} X + \dots + (\alpha^{k-1} X)^{n-1}).$$

Se știe că toate elementele nenule din $GF(q^m)$ sunt rădăcinile polinomului $X^{q^m-1} - 1 = X^n - 1 = (X - 1)(1 + X + X^2 + \dots + X^{n-1})$, deci toate elementele din $GF(q^m) \setminus \{0, 1\}$ ¹ sunt rădăcinile polinomului $1 + X + X^2 + \dots + X^{n-1}$.

Prin calcul se obține $f(\mathbf{1}) = \mathbf{a}_0$, $f(\alpha^{-1}) = -\mathbf{a}_1$, $f(\alpha^{-2}) = -\mathbf{a}_2, \dots, f(\alpha^{-(k-1)}) = -\mathbf{a}_{k-1}$ și $f(\alpha^{-j}) = \mathbf{0}$ dacă $n - 1 \geq j \geq k$.

Dar cum $\alpha^n = 1$, avem $\alpha^{-j} = \alpha^{n-j}$ și deci $f(\alpha^i) = \mathbf{0}$, ($1 \leq i \leq n - k$). q.e.d.

În continuare vom numi aceste coduri *Reed - Solomon* pentru care vom folosi următoarea definiție:

Definiția 13.1 *Fie $\alpha \in GF(q^m)$ primitiv. Se numește cod Reed - Solomon (RS) codul BCH de polinom generator $g(X) \in GF(q^m)[X]$ definit prin*

$$g(X) = (X - \alpha^{m_0})(X - \alpha^{m_0+1}) \dots (X - \alpha^{m_0+d-2}).$$

¹Reamintim, $\mathbf{1} = (1, 0, \dots, 0)$, $\mathbf{0} = (0, \dots, 0)$.

Deci, diferența față de codurile BCH generale este aceea că un cod RS este definit direct pe extensie, nu pe corpul de bază. Astfel, la un cod BCH cuvintele - cod sunt din Z_q^n , pe când la un cod RS, ele sunt din $GF(q^m)^n$. Cum și $GF(q^m)$ este corp, toate proprietățile codurilor BCH se păstrează și în cazul codurilor RS.

Deoarece au fost studiate (și utilizate) numai codurile Reed - Solomon de caracteristică 2, vom considera în continuare $q = 2$.

Exemplul 13.1 Fie $GF(2^3)$ generat de rădăcina α a polinomului $1 + X + X^3$, și să considerăm codul RS definit de polinomul $g(X) = (\alpha + X)(\alpha^2 + X) = \alpha^3 + \alpha^4 X + X^2$. Este un cod ciclic de lungime $n = 7$, $k = 5$ și $d = 3$. Folosind definiția din Prelegerea 9.3, se poate construi matricea generatoare a codului:

$$G = \begin{pmatrix} \alpha^3 & \alpha^4 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \alpha^3 & \alpha^4 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \alpha^3 & \alpha^4 & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \alpha^3 & \alpha^4 & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \alpha^3 & \alpha^4 & \mathbf{1} \end{pmatrix}$$

Codul are 8^5 cuvinte, obținute prin înmulțirea cu $g(X)$ a tuturor polinoamelor din $GF(2^3)[X]$ de grad cel mult 4 sau - echivalent - din înmulțirea cu matricea G a tuturor cuvintelor din $GF(2^3)^5$.

De exemplu, codificarea mesajului de informație $\mathbf{a} = \mathbf{1}\alpha\mathbf{0}\mathbf{0}\alpha^3$, sau $a(X) = \mathbf{1} + \alpha X + \alpha^3 X^4$ este $a(X)g(X) = \mathbf{1} + \alpha + (\alpha + \alpha^2)X^2 + \alpha X^3 + (1 + \alpha^2)X^4 + X^5 + (1 + \alpha)X^6 = \alpha^3 + \alpha^4 X^2 + \alpha X^3 + \alpha^6 X^4 + X^5 + \alpha^3 X^6$ (respectiv $\alpha^3\mathbf{0}\alpha^4\alpha^6\mathbf{1}\alpha^3$).

O matrice de control foarte simplă se obține folosind (conform Prelegerii 11.3) rădăcinile α și α^2 ale cuvintelor - cod (reamintim, $\alpha^7 = 1$):

$$H = \begin{pmatrix} \mathbf{1} & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \mathbf{1} & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \end{pmatrix}.$$

Teorema 13.2 Distanța minimă a unui cod RS este $d = n - k + 1$.

Demonstrație: Să notăm - temporar - cu d_H distanța minimă a unui cod. Conform Teoremei 3.3 (Prelegerea 3), $d_H \leq n - k + 1$. Pe de altă parte, într-un cod RS, $\text{grad}(g(X)) = n - k = d - 1$ deci $d = n - k + 1$. Din cele două afirmații rezultă $d_H \leq d$. Pe de altă parte, din Teorema 12.2, cum codul RS este un caz particular de cod BCH, rezultă $d_H \geq d$. q.e.d.

Din această teoremă rezultă că un cod RS are *distanța maximă separabilă* (se mai spune că este un cod DMS).

13.2 Coduri RS scurte

Deoarece $\alpha \in GF(2^m)$ este primitiv, toate cuvintele unui cod RS au lungimea $n = 2^m - 1$ și $k = n - d + 1 = 2^m - d$ simboluri de informație. Uneori însă sunt necesare coduri RS ale căror caracteristici să nu fie de această formă. Ele se obțin astfel:

Fie C un (n, k) - cod RS de distanță minimă d și s ($1 \leq s < 2^m - d$) un număr întreg. Se construiește codul RS *scurt* $C(s)$ format din toate cuvintele lui C care au $\mathbf{0}$ pe ultimele s poziții, după care apoi aceste s poziții se ignoră.

Exemplul 13.2 Fie codul C dat de $g(X) = \alpha + X$, unde α este rădăcina polinomului $1 + X + X^2$ și generează $GF(2^2)$. C are deci $n = 3$, $k = 2$, $d = 2$ și 16 cuvinte - cod.

Codul $C(1)$ se obține luând toate cuvintele lui C care au $\mathbf{0}$ pe ultima poziție (acestea sunt $\mathbf{000}, \alpha\mathbf{10}, \alpha^2\alpha\mathbf{0}, \mathbf{1}\alpha\mathbf{0}$), din care se elimină ultimul caracter.

Deci $C(1) = \{\mathbf{00}, \alpha\mathbf{1}, \alpha^2\alpha, \mathbf{1}\alpha\}$.

Atenție: Ultima poziție nu înseamnă ultimul bit. Fiecare element din cuvânt are aici două poziții binare. Astfel, cele patru cuvinte - cod alese din C au reprezentarea binară

$$000000, 011000, 110100, 100100$$

iar codul scurt, reprezentat în binar este $\hat{C}(1) = \{0000, 0110, 1101, 1001\}$.

Din definiție,

$$C(s) = \{\mathbf{b} \in C \mid \text{grad}(b(X)) < n - s\}.$$

Dacă $g(X)$ este polinomul generator al codului Reed - Solomon C , atunci $C(s)$ conține toate polinoamele de forma $b(X)$ unde $b(X) = a(X)g(X)$, $\text{grad}(a(X)) < k - s = 2^m - d - s$ (deoarece $\text{grad}(g(X)) = d - 1 = n - k$).

De aici rezultă că matricea generatoare a unui cod RS scurt $C(s)$ este

$$G = \begin{pmatrix} g(X) \\ Xg(X) \\ \vdots \\ X^{k-s-1}g(X) \end{pmatrix}$$

care se obține din matricea generatoare a codului C , prin eliminarea ultimelor s linii. Numărul de simboluri de informație este egal cu numărul de linii din G , deci

$$k_s = k - s = 2^m - d - s.$$

Să notăm acum cu d respectiv d_s distanțele minime ale celor două coduri. Se observă că $\forall \mathbf{c}_1, \mathbf{c}_2 \in C(s)$, $\mathbf{c}'_1 = \mathbf{c}_1\mathbf{00} \dots \mathbf{0}$, $\mathbf{c}'_2 = \mathbf{c}_2\mathbf{00} \dots \mathbf{0} \in C$ și deci $d(\mathbf{c}_1, \mathbf{c}_2) = d(\mathbf{c}'_1, \mathbf{c}'_2)$, adică $d_s \geq d$.

Pe de-altă parte (Teorema 3.3, Prelegerea 3),

$$d_s \leq n_s - k_s + 1 = 2^m - 1 - s - (2^m - d - s) + 1 = d, \text{ deci } d_s = d.$$

Din cele arătate mai sus putem enunța teorema:

Teorema 13.3 Fie C un (n, k) - cod RS cu distanța minimă d și $C(s)$ codul său scurt, cu parametrii n_s, k_s, d_s . Atunci

$$n_s = 2^m - 1 - s, \quad k_s = 2^m - d - s, \quad d_s = d$$

și $C(s)$ este un cod DMS.

Remarca 13.1 Alte coduri scurte se pot obține eliminând orice combinație fixă de s elemente din cuvintele unui cod RS. Se poate vedea imediat că proprietățile arătate mai sus nu se schimbă.

Exemplul 13.3 Plecând de la codul RS din Exemplul 13.1, codul scurt $C(2)$ va avea ca matrice generatoare matricea G fără ultimele două linii și coloane, iar ca parametri $n_2 = 5$, $k_2 = 3$, $d_2 = 3$. Matricea sa generatoare este

$$G = \begin{pmatrix} \alpha^3 & \alpha^4 & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \alpha^3 & \alpha^4 & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \alpha^3 & \alpha^4 & \mathbf{1} \end{pmatrix}$$

13.3 Decodificarea codurilor RS

Fiind cazuri particulare de coduri BCH, codurile RS se pot decodifica folosind direct algoritmul Peterson (Prelegerea 12). Să reluăm detaliat pe un exemplu aplicarea acestui algoritm.

Exemplul 13.4 Fie $g(X) = (1 + X)(\alpha + X)(\alpha^2 + X)(\alpha^3 + X)$ unde $\alpha \in GF(2^3)$ este rădăcina (primitivă) a polinomului $1 + X + X^3$. Codul are distanța minimă $d = 5$, deci poate corecta maxim 2 erori independente. Mai știm că:

$$\alpha^3 = 1 + \alpha, \quad \alpha^4 = \alpha + \alpha^2, \quad \alpha^5 = 1 + \alpha + \alpha^2, \quad \alpha^6 = 1 + \alpha^2.$$

Să presupunem că s-a recepționat cuvântul $\mathbf{r} = \alpha^6 \alpha^5 \alpha^2 \mathbf{1} \mathbf{0} \alpha^2$. Pentru decodificare se aplică următorii pași:

(1): Se calculează cele patru sindromuri:

$$S_0 = v(\alpha^0) = \alpha^6 + \alpha + \alpha^5 + \alpha^2 + \mathbf{1} + \mathbf{0} + \alpha^2 = \mathbf{1} + \alpha^2 + \alpha + \mathbf{1} + \alpha + \alpha^2 + \alpha^2 + \mathbf{1} + \alpha^2 = \mathbf{1};$$

$$S_1 = v(\alpha) = \alpha^6 + \alpha^2 + \alpha^7 + \alpha^5 + \alpha^4 + \mathbf{0} + \alpha^8 = \alpha^3;$$

$$S_2 = v(\alpha^2) = \alpha^6 + \alpha^3 + \alpha^9 + \alpha^8 + \alpha^8 + \mathbf{0} + \alpha^{14} = \alpha^3;$$

$$S_3 = v(\alpha^3) = \alpha^6 + \alpha^4 + \alpha^{11} + \alpha^{11} \alpha^{12} + \mathbf{0} + \alpha^{20} = \mathbf{1}.$$

(2) Deoarece rangul matricii $\begin{pmatrix} S_0 & S_1 \\ S_1 & S_2 \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \alpha^3 \\ \alpha^3 & \alpha^3 \end{pmatrix}$ este 2, cuvântul \mathbf{r} are două erori; cum distanța minimă este 5, ele pot fi corectate.

(3) Se rezolvă sistemul liniar

$$\begin{cases} S_0 \sigma_2 + S_1 \sigma_1 = S_2 \\ S_1 \sigma_2 + S_2 \sigma_1 = S_3 \end{cases} \text{ adică } \begin{cases} \sigma_2 + \alpha^3 \sigma_1 = \alpha^3 \\ \alpha^3 \sigma_2 + \alpha^3 \sigma_1 = \mathbf{1} \end{cases}$$

Adunând cele două ecuații, se obține $(\mathbf{1} + \alpha^3) \sigma_2 = \mathbf{1} + \alpha^3$, deci $\sigma_2 = \mathbf{1}$.

Înlocuind în prima ecuație, avem $\alpha^3 \sigma_1 = \alpha^3 + \mathbf{1} = \alpha$, de unde, prin înmulțire cu α^4 se ajunge la $\sigma_1 = \alpha^5$.

(4) Polinomul de localizare a erorii este $\sigma(X) = X^2 + \sigma_1 X + \sigma_2 = X^2 + \alpha^5 X + \mathbf{1}$. Prin încercări se obține $\sigma(\alpha) = \sigma(\alpha^6) = \mathbf{0}$, deci cele două erori sunt pe pozițiile $X_1 = \alpha$, $X_2 = \alpha^6$.

(5) Se rezolvă sistemul liniar $\begin{pmatrix} X_1^0 & X_2^0 \\ X_1 & X_2 \end{pmatrix} \cdot \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} = \begin{pmatrix} S_0 \\ S_1 \end{pmatrix}$, adică:

$$\begin{cases} Y_1 + Y_2 = \mathbf{1} \\ \alpha Y_1 + \alpha^6 Y_2 = \alpha^3 \end{cases}$$

Înmulțind a doua ecuație cu α și adunând la prima, se obține $\alpha^6 Y_1 = \alpha^5$, deci $Y_1 = \alpha^6$. După înlocuire în prima ecuație, se determină și $Y_2 = \alpha^2$.

(6) Rezumând, au apărut erorile α^6 pe poziția 1 și α^2 pe poziția 6. Deci eroarea este $\mathbf{e} = \mathbf{0} \alpha^6 \mathbf{0} \mathbf{0} \mathbf{0} \mathbf{0} \alpha^2$, iar cuvântul - cod trimis a fost

$$\mathbf{c} = \mathbf{r} + \mathbf{e} = \alpha^6 \alpha^5 \alpha^5 \alpha^2 \mathbf{1} \mathbf{0} \mathbf{0}.$$

13.4 Altă construcție a codurilor RS

Plecând de la construcția inițială, se poate realiza și o altă definiție pentru codurile RS. Ea se bazează pe o modalitate diferită de reprezentare a cuvintelor din Z_q^n , anume considerând polinoamele drept funcții.

Definiția 13.2 Fie $S \subseteq GF(2^m)$. Două polinoame $p(X), q(X)$ reprezintă aceeași funcție $: S \rightarrow GF(2^m)$ dacă și numai dacă $\forall \beta \in S, p(\beta) = q(\beta)$.

Exemplul 13.5 Fie $S = GF(2^3)$, construit folosind rădăcina primitivă α a polinomului $1 + X + X^3$. Definim funcția $f : S \rightarrow Z_2$ prin

$$\begin{array}{c|cccccccc} X & \mathbf{0} & \mathbf{1} & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \hline f(X) & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{array}$$

Atunci $f(x)$ se poate reprezenta prin

$$\mathbf{v}_f = f(\mathbf{0})f(\mathbf{1})f(\alpha)f(\alpha^2)f(\alpha^3)f(\alpha^4)f(\alpha^5)f(\alpha^6) = 00110100.$$

Exemplul 13.6 Fie $S = GF(2^3)$ și funcția $f : S \rightarrow S$ definită $\mathbf{v}_f = \alpha^4 \mathbf{0} \mathbf{1} \alpha^2 \mathbf{1} \alpha \mathbf{0} \mathbf{0}$.

Considerând f drept polinom, folosim metoda coeficienților nedeterminați pentru a obține reprezentarea

$$f(X) = \alpha^4 + \alpha^2 X + \alpha^3 X^2 + X^3.$$

Mulțimea $A = \{f(X) \in GF(2^m)[X] \mid \text{grad}(f(X)) \leq k - 1\}$ se poate structura evident ca un spațiu liniar de dimensiune k , cu baza $\{\mathbf{1}, X, X^2, \dots, X^{k-1}\}$. Deoarece toate aceste polinoame pot fi considerate funcții definite pe o anumită mulțime $S \subseteq GF(2^m)$, vom numi A spațiul funcțiilor pe S .

De remarcat că în această fază S nu este definit explicit.

Teorema 13.4 Fie mulțimea S , $\text{card}(S) = n$ și $k \leq n$. Mulțimea tuturor funcțiilor $: S \rightarrow GF(2^m)$ reprezentate prin polinoame de grad $\leq k - 1$ formează un spațiu liniar de funcții, cu baza $\{\mathbf{1}, X, \dots, X^{k-1}\}$.

Demonstrație: Evident, orice polinom de grad cel mult $k - 1$ este generat de $\{\mathbf{1}, X, \dots, X^{k-1}\}$. Tot ce trebuie arătat este că fiecare funcție $f : S \rightarrow GF(2^m)$ are o reprezentare unică sub formă de polinom.

Să presupunem (prin absurd) că $p(X)$ și $q(X)$ sunt egale ca funcții pe S . Deci $\forall \alpha \in S, p(\alpha) = q(\alpha)$. Dar atunci $p(X) - q(X)$ este un polinom de grad $< k$ care are n ($n \geq k$) rădăcini, deci $p(X) - q(X) \equiv 0$, adică $p(X) \equiv q(X)$. q.e.d.

Exemplul 13.7 Fie $S = GF(2^3)$ construit folosind rădăcina α a ecuației $1 + X + X^3 = 0$, și să considerăm toate polinoamele din $GF(2^3)[X]$, de grad cel mult doi. O bază pentru acest spațiu de funcții este $\{\mathbf{1}, X, X^2\}$, cu reprezentarea corespunzătoare:

$$\begin{array}{lll} \mathbf{1} & - & \mathbf{11111111} \\ X & - & \mathbf{01}\alpha\alpha^2\alpha^3\alpha^4\alpha^5\alpha^6 \\ X^2 & - & \mathbf{01}\alpha^2\alpha^4\alpha^6\alpha\alpha^3\alpha^5 \end{array}$$

Atunci, orice polinom $f(X) = \mathbf{a}_0 + \mathbf{a}_1X + \mathbf{a}_2X^2 \in GF(2^3)[X]$, considerat ca o funcție, poate fi reprezentat sub formă de vector prin

$$\mathbf{v}_f = (\mathbf{a}_0 \ \mathbf{a}_1 \ \mathbf{a}_2) \begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \mathbf{0} & \mathbf{1} & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \end{pmatrix}$$

Sunt $8^3 = 512$ astfel de polinoame.

Teorema 13.5 Spațiul funcțiilor pe $S \subseteq GF(2^m)$ ($\text{card}(S) = n$) al tuturor polinoamelor din $GF(2^m)[X]$ de grad $\leq k - 1$ formează un (n, k) - cod liniar DMS.

Demonstrație: Alegem o mulțime cu n elemente $S \subseteq GF(2^m)$ și considerăm spațiul liniar al funcțiilor tuturor polinoamelor $p : S \rightarrow GF(2^m)$ de grad cel mult $k - 1$. Lungimea fiecărui cuvânt - cod este n , iar dimensiunea spațiului este k (Teorema 20.1). Mai rămâne de arătat că acest cod este DMS (are distanță maximă separabilă), adică $d = n - k + 1$. Pentru aceasta, să observăm că orice polinom $p(X)$ are maxim $k - 1$ rădăcini distincte; deci reprezentarea \mathbf{v}_p are cel mult $k - 1$ zero-uri, adică $w(\mathbf{v}_p) \geq n - k + 1$. În particular, $d = \min\{w(\mathbf{v}_p)\} \geq n - k + 1$.

Pe de altă parte, conform Teoremei 3.3, $d \leq n - k + 1$, deci $d = n - k + 1$. q.e.d.

Fie n un divizor al lui $2^m - 1$ și $S = \{\alpha \in GF(2^m) | \alpha^n = 1\}$ mulțimea rădăcinilor de ordinul n ale unității din $GF(2^m)$. Evident, S conține toate rădăcinile din $GF(2^m)$ ale ecuației $X^n + 1 = 0$ și - fiind un corp finit - are cel puțin un element primitiv (Teorema 11.1).

Teorema 13.6 Fie $p(X), q(X) \in GF(2^m)[X]$ și $S \subseteq GF(2^m)$ mulțimea rădăcinilor de ordinul n ale unității. Atunci $p(X)$ și $q(X)$ reprezintă aceeași funcție $f : S \rightarrow GF(2^m)$ dacă și numai dacă $p(X) \equiv q(X) \pmod{X^n + 1}$.

Demonstrație: " \Leftarrow ": Fie $q(X) = h(X)(X^n + 1) + p(X)$. Atunci pentru orice $\alpha \in S$ avem $q(\alpha) = h(\alpha)(1 + \alpha^n) + p(\alpha)$. Dar $1 + \alpha^n = 0$, deci $q(\alpha) = p(\alpha)$.

" \Rightarrow ": Fie $\alpha \in S$ primitiv. Deci orice element din S are forma α^i ($0 \leq i < n$). Dacă $q(\alpha^i) = p(\alpha^i)$, atunci α^i ($0 \leq i < n$) sunt rădăcini ale polinomului $p(X) - q(X)$. Putem scrie atunci

$$p(X) - q(X) = h(X) \prod_{i=0}^{n-1} (1 + \alpha^i) = h(X)(1 + X^n).$$

q.e.d.

Teorema 13.7 Fie S mulțimea rădăcinilor de ordinul n ale unității din $GF(2^m)$. Atunci spațiul funcțiilor tuturor polinoamelor din $GF(2^m)[X]$ de grad $\leq k - 1$ pe S formează un (n, k) - cod ciclic peste $GF(2^m)$.

Demonstrație: Să notăm cu C acest spațiu și fie S ca în ipoteză, generat de α primitiv. Atunci, dacă $p(X) \in GF(2^m)[X]$ este un polinom de grad cel mult $k - 1$, avem $\mathbf{v}_p = p(\mathbf{1})p(\alpha) \dots p(\alpha^{n-1}) \in C$.

Fie $q(X) = p(\alpha X)$. Cum $\text{grad}(q(X)) \leq k - 1$, rezultă $\mathbf{v}_q \in C$. Dar $q(\mathbf{1})q(\alpha) \dots q(\alpha^{n-1}) = p(\alpha)p(\alpha^2) \dots p(\alpha^{n-1})p(\mathbf{1}) \in C$, care ce reprezintă - conform definiției - un cod ciclic.

q.e.d.

Exemplul 13.8 Să considerăm $S = GF(2^3)$ spațiul rădăcinilor de ordin 7 ale unității, generat de α , rădăcina ecuației $1 + X + X^3 = 0$. Fie de exemplu $p(X) = \alpha^4 + \alpha^2 X + \alpha^3 X^2 + X^3 = (\mathbf{1} + X)(\alpha^5 + X)(\alpha^6 + X)$, care are reprezentarea $\mathbf{01}\alpha^2\mathbf{1}\alpha\mathbf{00}$.

Permutarea ei ciclică este $\mathbf{1}\alpha^2\mathbf{1}\alpha\mathbf{000}$, care corespunde funcției $p(\alpha X) = \alpha^4 + \alpha^3 X + \alpha^5 X^2 + \alpha^3 X^3 = (\alpha^4 + X)(\alpha^5 + X)(\alpha^6 + X)\alpha^3$.

Definiția 13.3 Fie polinomul $V(X) = \mathbf{V}_0 + \mathbf{V}_1 X + \dots + \mathbf{V}_{n-1} X^{n-1} \in GF(2^m)[X]$ și $\alpha \in GF(2^m)$ o rădăcină primitivă de ordinul n a unității. Spunem că $v(X) = \mathbf{v}_0 + \mathbf{v}_1 X + \dots + \mathbf{v}_{n-1} X^{n-1}$ este "transformata" lui $V(X)$ dacă

$$V(\alpha^j) = \sum_{i=0}^{n-1} \mathbf{V}_i \alpha^{ji} = \mathbf{v}_j, \quad j = 0, 1, \dots, n-1.$$

Altfel spus, dacă se notează cu $M_{n,n} = (\alpha^{ij})_{1 \leq i, j \leq n}$, atunci

$$(\mathbf{V}_0 \ \mathbf{V}_1 \ \dots \ \mathbf{V}_{n-1})M = (\mathbf{v}_0 \ \mathbf{v}_1 \ \dots \ \mathbf{v}_{n-1}).$$

Matricea M este numită *transformata Fourier finită*. Ea este nesingulară, deci se poate scrie $(\mathbf{V}_0 \ \mathbf{V}_1 \ \dots \ \mathbf{V}_{n-1}) = (\mathbf{v}_0 \ \mathbf{v}_1 \ \dots \ \mathbf{v}_{n-1})M^{-1}$ sau

$$\mathbf{V}_i = \sum_{j=1}^{n-1} \mathbf{v}_j \alpha^{-ij} = v(\alpha^{-i})$$

Vom demonstra aceasta formulă în mod direct, fără a folosi inversa matricii M :

Teorema 13.8 Fie $\alpha \in GF(2^m)$ o rădăcină primitivă de ordin n a unității. Dacă $\mathbf{v}_j = V(\alpha^j)$ pentru $V(X) = \mathbf{V}_0 + \mathbf{V}_1 X + \dots + \mathbf{V}_{n-1} X^{n-1} \in GF(2^m)[X]$, atunci $\mathbf{V}_i = v(\alpha^{-i})$ unde $v(X) = \mathbf{v}_0 + \mathbf{v}_1 X + \dots + \mathbf{v}_{n-1} X^{n-1}$.

Demonstrație: Deoarece α este primitiv, vom avea

$$\sum_{j=0}^{n-1} \alpha^{(k-i)j} = \begin{cases} n \pmod{2} & \text{pentru } k = i \\ 0 & \text{pentru } k \neq i \end{cases}$$

Atunci $v(\alpha^{-i}) = \sum_{j=0}^{n-1} \mathbf{v}_j \alpha^{-ij} = \sum_{j=0}^{n-1} \left(\sum_{k=0}^{n-1} \mathbf{V}_k \alpha^{kj} \right) \alpha^{-ij} = \sum_{k=0}^{n-1} \mathbf{V}_k \left(\sum_{j=0}^{n-1} \alpha^{(k-i)j} \right) = \mathbf{V}_i$.

q.e.d.

Deci, fiind dat un vector de valori $(\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_{n-1})$, se pot regăsi coeficienții polinomului $V(X) = \mathbf{V}_0 + \mathbf{V}_1 X + \dots + \mathbf{V}_{n-1} X^{n-1}$, lucru esențial în definirea unui algoritm de decodificare.

Teorema 13.9 Fie $S \subseteq GF(2^m)$ mulțimea rădăcinilor de ordinul n ale unității din $GF(2^m)$, generat de rădăcina primitivă α . Spațiul liniar al tuturor funcțiilor polinomiale de grad $< n - d + 1$ pe S este un cod ciclic DMS cu polinomul generator

$$g(X) = (\alpha + X)(\alpha^2 + X) \dots (\alpha^{d-1} + X).$$

Demonstrație: Funcția polinomială $V(X)$ a cărei transformată corespunde lui $v(X) =$

$a(X)g(X)$ este $V(X) = \sum_{i=0}^{n-1} v(\alpha^{n-i})X^i$. Deoarece $g(\alpha^p) = 0$, ($1 \leq p \leq d-1$), vom

avea $v(\alpha^{n-i}) = 0$ pentru $i = n - d + 1, n - d + 2, \dots, n - 1$. Rezultă că pentru orice i , ($n - d + 1 \leq i \leq n - 1$) coeficientul lui X^i din $V(X)$ este zero, deci $V(X)$ are gradul $< n - d + 1$. q.e.d.

Dacă se ia $n = 2^m - 1$, se ajunge la o altă modalitate de construcție a codurilor RS, folosind o altă matrice generatoare (deci o altă dispunere a biților de informație).

Exemplul 13.9 Fie α o rădăcină a polinomului $1 + X + X^3$, cu care se construiește $GF(2^3)$. Considerăm codul RS de polinom generator $g(X) = (1 + X)(\alpha + X)(\alpha^2 + X)(\alpha^3 + X) = \alpha^6 + \alpha^5 X + \alpha^5 X^2 + \alpha^2 X^3 + X^4$ (care corespunde reprezentării $\alpha^6 \alpha^5 \alpha^5 \alpha^2 \mathbf{100}$).

Transformata lui $g(X)$ este polinomul $G(X) = \sum_{i=0}^6 g(\alpha^{7-i})X^i$.

Pentru că $g(\alpha^0)g(\alpha) \dots g(\alpha^6) = \mathbf{00001}\alpha\alpha^4$, avem

$$G(X) = g(\mathbf{1}) + g(\alpha^{7-1})X + g(\alpha^{7-2})X^2 + g(\alpha^{7-3})X^3 = \alpha^4 X + \alpha X^2 + X^3.$$

Cum $d - 1 = 3$, avem $d = 4$ deci $gr(G(X)) < n - d + 1 = 4$.

Se verifică ușor că $G(X)$ este o funcție polinomială cu reprezentarea

$$G(\alpha^0)G(\alpha) \dots G(\alpha^6) = \alpha^6 \alpha^5 \alpha^5 \alpha^2 \mathbf{100}.$$

Putem gândi acest cod RS ca spațiul tuturor polinoamelor de grade 1, 2 și 3, adică

$$\{Xp(X) \mid p(X) \in GF(2^3)[X], grad(p(X)) < 3\}.$$

Baza pentru acest spațiu liniar este $\{X, X^2, X^3\}$, iar matricea generatoare

$$\begin{pmatrix} \mathbf{1} & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \mathbf{1} & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \\ \mathbf{1} & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha & \alpha^4 \end{pmatrix}$$

Deci $G(X) = \alpha^4 X + \alpha X^2 + X^3$ va avea reprezentarea

$$(\alpha^4 \alpha \mathbf{1}) \begin{pmatrix} \mathbf{1} & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \mathbf{1} & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \\ \mathbf{1} & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha & \alpha^4 \end{pmatrix} = (\alpha^6 \alpha^5 \alpha^5 \alpha^2 \mathbf{100}).$$

13.4.1 Algoritmul de decodificare Berlekamp - Massey

Fie $g(X)$ polinomul generator al unui cod RS, $c(X) = a(X)g(X)$ un polinom - cod transmis, și $v(X) = c(X) + e(X)$ polinomul recepționat. Notăm $V(X), C(X)$ și $E(X)$ transformatele lui $v(X), c(X)$ respectiv $e(X)$. Cum transformata Fourier finită este o aplicație liniară, se verifică imediat $V(X) = C(X) + E(X)$.

Deoarece $g(X)$ are $d - 1$ rădăcini consecutive α^i , ($m_0 \leq i \leq m_0 + d - 2$), sindromul polinomial $s(X)$ va avea $s(\alpha^{n-i}) = e(\alpha^{n-i}) = E_i$ pentru $i = n - m_0 - d + 1, \dots, n - m_0$. Deci sindromul va putea determina $d - 1$ coeficienți ai transformatei $E(X)$.

Pentru ceilalți coeficienți va trebui să reluăm polinomul de localizare a erorilor $\sigma(X)$ din algoritmul Peterson de decodificare a codurilor BCH.

$\sigma(X)$ are proprietatea că $\sigma(\alpha^k) = 0 \iff e_k \neq 0$.

Deoarece $E(\alpha^k) = e_k$, vom avea $\sigma(\alpha^k)E(\alpha^k) = 0 \forall k$, deci - lucrând în algebra polinoamelor modulo $1 + X^n$:

$$\{\sigma(X)\}\{E(X)\} = 0$$

și

$$\{\sigma(X)\}\{E(X)\} = \left\{ \left(\sum_{i=0}^t \sigma_i X^i \right) \left(\sum_{j=0}^{n-1} \mathbf{E}_j X^j \right) \right\},$$

unde $t = [(d-1)/2]$. Calculând în ambele relații coeficientul lui X^{t+j} , se obține:

$$\sigma_t \mathbf{E}_j + \sigma_{t-1} \mathbf{E}_{j+1} + \dots + \sigma_0 \mathbf{E}_{j+t} = 0.$$

Pentru că știm deja $d-1$ valori consecutive ale lui \mathbf{E}_j , putem determina recursiv coeficienții σ_i , pe care îi folosim la generarea tuturor valorilor \mathbf{E}_j .

În [7] se definește o altă formă a acestui algoritm.

Exemplul 13.10 Să reluăm codul RS definit în Exemplul 14.4, în care considerăm că s-a recepționat cuvântul $\mathbf{v} = \alpha^6 \alpha^5 \alpha^2 \mathbf{10} \alpha^2$.

Deoarece $d = 5$, avem $t \leq 2$ și

$$\mathbf{E}_0 = v(\alpha^0) = \mathbf{1}, \mathbf{E}_6 = v(\alpha) = \alpha^3, \mathbf{E}_5 = v(\alpha^2) = \alpha^3, \mathbf{E}_4 = v(\alpha^3) = \mathbf{1}.$$

Polinomul de localizare a erorii este $\sigma(X) = X^2 + \sigma_1 X + \sigma_0$.

Determinăm σ_0, σ_1 ca în Exemplul 14.4 și găsim $\sigma_0 = \mathbf{1}$, $\sigma_1 = \alpha^5$. Se poate determina astfel relația de recurență:

$$\mathbf{E}_k = \alpha^5 \mathbf{E}_{k+1} + \mathbf{E}_{k+2}.$$

Deoarece $(\mathbf{E}_0, \mathbf{E}_6, \mathbf{E}_5, \mathbf{E}_4) = (\mathbf{1}, \alpha^3, \alpha^3, \mathbf{1})$, vom avea:

$$\mathbf{E}_3 = \alpha^5 \mathbf{E}_4 + \mathbf{E}_5 = \alpha^5 + \alpha^3 = \alpha^2,$$

$$\mathbf{E}_2 = \alpha^5 \mathbf{E}_3 + \mathbf{E}_4 = \alpha^5 \cdot \alpha^2 + \mathbf{1} = \mathbf{0},$$

$$\mathbf{E}_1 = \alpha^5 \mathbf{E}_2 + \mathbf{E}_3 = \mathbf{0} + \alpha^2 = \alpha^2.$$

Am determinat transformata lui $e(X)$, adică $E(X) = \mathbf{1} + \alpha^2 X + \alpha^2 X^3 + X^4 + \alpha^3 X^5 + \alpha^3 X^6$. De aici rezultă $\mathbf{e} = (E(\alpha^0), E(\alpha), E(\alpha^2), \dots, E(\alpha^6)) = \mathbf{0} \alpha^6 \mathbf{0000} \alpha^2$, deci cuvântul decodificat este

$$\mathbf{c} = \mathbf{v} + \mathbf{e} = \alpha^6 \alpha^5 \alpha^5 \alpha^2 \mathbf{100}.$$

Pe de-altă parte, dacă s-ar fi folosit matricea generatoare din Exemplul 17.3, nu ar mai fi fost necesar să se determine vectorul eroare \mathbf{e} , ci doar să se calculeze toate valorile lui $v(\alpha^k)$, ($0 \leq k \leq 6$), din care se obține direct mesajul transmis; mai precis, se află transformata lui $v(X)$, care se adună la transformata lui $e(X)$:

$$\mathbf{v}_0 \mathbf{v}_1 \dots \mathbf{v}_6 = v(\alpha^0) v(\alpha^6) v(\alpha^5) \dots v(\alpha) = \mathbf{1} \alpha \alpha^6 \mathbf{1} \alpha^3 \alpha^3, \quad \mathbf{E}_0 \mathbf{E}_1 \dots \mathbf{E}_6 = \mathbf{1} \alpha^2 \mathbf{0} \alpha^2 \mathbf{1} \alpha^3 \alpha^3,$$

și deci

$$\mathbf{C}_0 \mathbf{C}_1 \dots \mathbf{C}_6 = \mathbf{V}_0 \mathbf{V}_1 \dots \mathbf{V}_6 + \mathbf{E}_0 \mathbf{E}_1 \dots \mathbf{E}_6 = \mathbf{0} \alpha^4 \alpha \mathbf{1000}.$$

S-a obținut $C(X) = \alpha^4 X + \alpha X^2 + X^3$. Deci biții de informație sunt $(\alpha^4, \alpha, \mathbf{1})$.

De aici se obține forma cuvântului-cod transmis: $\mathbf{c} = \alpha^6 \alpha^5 \alpha^5 \alpha^2 \mathbf{100}$.

13.5 Ștergeri

O ștergere este o eroare detectată, pentru corectarea ei trebuind determinată doar valoarea erorii. Numărul locației de ștergere este poziția elementului perturbat. Acest număr este în general cunoscut (din citirea fizică a mesajului primit sau din structura codului).

Astfel, fie C un cod RS peste $GF(2^m)$ și să considerăm codul \hat{C} format din reprezentarea binară a cuvintelor codului C (fiecare element din $GF(2^m)$ se scrie ca un cuvânt binar de lungime m). Se definește și codul \hat{C}' obținut prin adăugarea câte unui bit de paritate la fiecare simbol din componența cuvintelor codului \hat{C} .

Exemplul 13.11 Să considerăm codul RS din Exemplul 20.3. Pentru a forma \hat{C}' , se înlocuiește fiecare element $\mathbf{0}, \mathbf{1}, \alpha, \alpha^2$ respectiv cu 000, 101, 011, 110 (al treilea bit este bitul de paritate). Astfel, pentru $\alpha \mathbf{10} \in A$ avem $011101000 \in \hat{C}'$.

Deoarece fiecare element dintr-un cuvânt - cod $\mathbf{c} \in C$ este reprezentat printr-un cuvânt binar de lungime $m + 1$, cuvântul - cod corespunzător $\hat{\mathbf{c}}' \in \hat{C}'$ va avea lungimea $(2^m - 1)(m + 1)$. Se observă că fiecare element nenul din \mathbf{c} este înlocuit cu un cuvânt de pondere pară. Deci, $\hat{\mathbf{c}}'$ va fi format din $2^m - 1$ cuvinte binare de lungimi $m + 1$, fiecare de pondere pară.

În acest fel, dacă în cuvântul recepționat $\hat{\mathbf{v}}'$ există un grup de pondere impară, știm că a apărut o eroare printre cei $m + 1$ biți corespunzători. Considerat din nou ca un cuvânt cu elemente din $GF(2^m)$, știm că acest cuvânt are un element perturbat; nu cunoaștem valoarea erorii, dar știm poziția sa. Avem deci o "ștergere".

Exemplul 13.12 *Reluând construcția din exemplul anterior, să presupunem că s-a primit 011100000. Deci, au apărut erori în al doilea grup de 3 biți (acesta are pondere impară), așa că α^1 este un numărul locației de ștergere. Știind acest număr, putem înlocui al doilea element cu $\mathbf{0}$ (ușurează calculul sindromului), și vom încerca decodificarea lui $\mathbf{v} = \alpha\mathbf{00}$ în cel mai apropiat cuvânt - cod din A , știind că poziția erorii este $X_1 = \alpha$.*

Teorema 13.10 *Fie C un cod RS peste $GF(2^m)$ de distanța minimă d , și \mathbf{v} un cuvânt recepționat care are s ștergeri și t erori care nu sunt ștergeri. Atunci \mathbf{v} poate fi decodificat corect dacă $2t + s \leq d - 1$.*

Demonstrație: De remarcat că în Teorema 3.7 (pentru coduri liniare), o condiție mai slabă ($t + s \leq d - 1$) asigură corectarea celor t erori și detectarea (doar) a s ștersături. Această teoremă realizează corectarea completă a cuvântului.

Fie B mulțimea pozițiilor ștersăturilor și A mulțimea pozițiilor erorilor; deci $A \setminus B$ este mulțimea pozițiilor erorilor care nu sunt ștersături. Definim $\sigma_B(X) = \prod_{i \in B} (\alpha^i + X)$ ca fiind polinomul de localizare a ștersăturilor. Atunci polinomul de localizare a erorilor se poate scrie

$$\sigma_A(X) = \sigma_B(X)\sigma_{A \setminus B}(X).$$

Aflarea pozițiilor erorilor revine la determinarea rădăcinilor polinomului $\sigma_{A \setminus B}(X)$. Pentru aceasta se poate folosi algoritmul Peterson de decodificare a codurilor BCH (Prelegerea 12), cu câteva schimbări, corespunzătoare sindromurilor "modificate".

Astfel, fie

$$\sigma_B(X) = B_0 + B_1X + \dots + B_{s-1}X^{s-1} + X^s \text{ și } \sigma_{A \setminus B}(X) = A_0 + A_1X + \dots + A_{t-1}X^{t-1} + X^t.$$

Similar algoritmului Peterson, vom înmulți ambii membri ai egalității $\sigma_B(X)\sigma_{A \setminus B}(X) = \sigma_A(X)$ cu $Y_i X_i^j$ ($m_0 \leq j \leq m_0 + d - 2$) (reamintim, X_i corespund locațiilor erorilor, iar Y_i sunt valorile acestor erori).

Apoi se înlocuiește $X = X_i$ și se face suma după $i = 1, \dots, t + s$. Se obține

$$S_j' A_0 + S_{j+1}' A_1 + \dots + S_{j+t-1}' A_{t-1} + S_{j+t}' = 0 \quad (m_0 \leq j \leq m_0 + d - 2) \quad (1)$$

unde S_k' sunt sindromurile modificate, de forma

$$S_k' = B_0 S_k + B_1 S_{k+1} + \dots + B_{s-1} S_{k+s-1} + S_{k+s}. \quad (2)$$

Pentru că se știu valorile lui S_j pentru $j = m_0, \dots, m_0 + d - 2$ iar B_0, B_1, \dots, B_{s-1} sunt cunoscute, se pot determina S_j' pentru $m_0 \leq j \leq d - 1 - s$. Deoarece $2t + s \leq d - 1$, deci $2t \leq d - 1 - s$, se poate rezolva sistemul (1) scris matricial

$$\begin{pmatrix} S_{m_0}' & S_{m_0+1}' & \cdots & S_{m_0+t-1}' \\ S_{m_0+1}' & S_{m_0+2}' & \cdots & S_{m_0+t}' \\ & & \ddots & \\ S_{m_0+t-1}' & S_{m_0+t}' & \cdots & S_{m_0+2t-1}' \end{pmatrix} \begin{pmatrix} A_0 \\ A_1 \\ \vdots \\ A_{t-1} \end{pmatrix} = \begin{pmatrix} S_{m_0+t}' \\ S_{m_0+t+1}' \\ \vdots \\ S_{m_0+2t} \end{pmatrix} \quad (3)$$

cu t necunoscute A_0, A_1, \dots, A_{t-1} .

După determinarea polinomului de localizare a erorilor, algoritmul Peterson se poate aplica în continuare nemodificat. q.e.d.

Fie C un cod RS peste $GF(2^m)$, de polinom generator $g(X) = (\alpha^{m_0} + X) \dots (\alpha^{m_0+d-2} + X)$. Se transmite cuvântul - cod \mathbf{c} și se recepționează \mathbf{v} cu s ștergeri localizate în elementele mulțimii $B = \{X_1, \dots, X_s\}$. Fie $\sigma_B(X) = B_0 + B_1X + \dots + B_{s-1}X^{s-1} + X^s$ polinomul de localizare al ștergerilor.

Polinomul de localizare a erorilor $\sigma_A(X) = \sigma_{A \setminus B}(X)\sigma_B(X)$ poate fi determinat aflând $\sigma_{A \setminus B}(X) = A_0 + A_1X + \dots + A_{t-1}X^{t-1} + X^t$ astfel:

- (a) Se calculează $S_j = v(\alpha^j)$ pentru $j = m_0, m_0 + 1, \dots, m_0 + d - 2$;
 (b) Se determină S_j' ($m_0 \leq j \leq m_0 + d - 2 - s$) cu relațiile (2);
 (c) Se rezolvă sistemul liniar (3) de necunoscute A_i .

Exemplul 13.13 Fie C codul RS peste $GF(2^4)$ construit de rădăcina α a ecuației $1 + X + X^4 = 0$, definit cu polinomul generator $g(X) = (1 + X)(\alpha + X) \dots (\alpha^4 + X)$ și \hat{C}' codul binar asociat în care sunt codificate mesajele. S-a primit mesajul

$$\hat{\mathbf{v}}' = 11101 \ 11001 \ 00101 \ 00110 \ 10010 \ 0 \dots 0,$$

deci $\mathbf{v} = \alpha^{10}\mathbf{0}\alpha^2\mathbf{0}\alpha^6\alpha^{14}\mathbf{0}\dots\mathbf{0}$.

Singura ștergere localizată este α^1 , deci $\sigma_B(X) = \alpha + X$.

Trebuie să determinăm polinomul de localizare a erorilor.

Din $v(X) = \alpha^{10} + \alpha^2X^2 + \alpha^6X^4 + \alpha^{14}X^5$ calculăm sindromurile

$$S_0 = \alpha^5, \quad S_1 = \mathbf{0}, \quad S_2 = \alpha^3, \quad S_3 = \alpha^4, \quad S_4 = \alpha^3.$$

Deoarece $B_0 = \alpha$ și $s = 1$, relația de recurență pentru sindromurile modificate este $S_j' = B_0S_j + S_{j+1} = \alpha S_j + S_{j+1}$ ($0 \leq j \leq 3$).

Se determină astfel $S_0' = \alpha^6$, $S_1' = \alpha^3$, $S_2' = \mathbf{0}$, $S_3' = \alpha^{11}$.

Deoarece $d = 5$ avem $t = 2$, deci sistemul (3) este

$$\begin{pmatrix} S_0' & S_1' \\ S_1' & S_2' \end{pmatrix} \begin{pmatrix} A_0 \\ A_1 \end{pmatrix} = \begin{pmatrix} S_2' \\ S_3' \end{pmatrix} \quad \text{sau} \quad \begin{pmatrix} \alpha^6 & \alpha^3 \\ \alpha^3 & \mathbf{0} \end{pmatrix} \begin{pmatrix} A_0 \\ A_1 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \alpha^{11} \end{pmatrix}$$

cu soluția $A_0 = \alpha^8$, $A_1 = \alpha^{11}$.

S-a obținut polinomul $\sigma_{A \setminus B}(X) = \alpha^8 + \alpha^{11}X + X^2$. Deci,

$$\sigma_A(X) = \sigma_B(X)\sigma_{A \setminus B}(X) = (\alpha + X)(\alpha^8 + \alpha^{11}X + X^2) = (\alpha + X)(\alpha^3 + X)(\alpha^5 + X).$$

Rezultă că erorile sunt localizate pe pozițiile $X_1 = \alpha$, $X_2 = \alpha^3$, $X_3 = \alpha^5$.

Mai departe se lucrează pentru determinarea valorilor erorilor, folosind algoritmul Peterson și sindromurile originale. Trebuie rezolvat sistemul:

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \alpha & \alpha^3 & \alpha^5 \\ \alpha^2 & \alpha^6 & \alpha^{10} \end{pmatrix} \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = \begin{pmatrix} \alpha^5 \\ \mathbf{0} \\ \alpha^3 \end{pmatrix}$$

cu soluția $Y_1 = \alpha^{12}$, $Y_2 = \mathbf{1}$, $Y_3 = \alpha^3$.

Deci $v(X)$ se decodifică în $c(X) = v(X) + e(X) = (\alpha^{10} + \alpha^2X^2 + \alpha^6X^4 + \alpha^{14}X^5) + (\alpha^{12}X + X^3 + \alpha^3X^5)$, adică $\mathbf{c} = \alpha^{10}\alpha^{12}\alpha^2\mathbf{1}\alpha^6\mathbf{1}\mathbf{0}\dots\mathbf{0}$, sau - în \hat{C}' :

$$11101 \ 11110 \ 00101 \ 10001 \ 00110 \ 10001 \ 0 \dots 0.$$

13.6 Exerciții

13.1 Construiți matricea generatoare și listați toate cuvintele - cod ale codului RS din Exemplul 20.3. Scrieți și forma binară a acestor cuvinte.

13.2 Fie codul definit de $g(X) = (1 + X)(\alpha + X)(\alpha^2 + X)(\alpha^3 + X)$, bazat pe extensia Galois $GF(2^3)$ generată de α , rădăcină a polinomului $1 + X + X^3$.

1. Determinați n, k, d și numărul de cuvinte - cod.
2. Construiți o matrice generatoare și o matrice de control a codului.
3. Codificați mesajele de informație $10\alpha^2$, 111 , $\alpha^2\alpha^4\alpha^6$.

13.3 Construiți coduri RS cu următorii parametri:

- (a) $m = 2, d = 3, m_0 = 2$;
- (b) $m = 3, d = 3, m_0 = 2$;
- (c) $m = 3, d = 5, m_0 = 0$;
- (d) $m = 4, d = 5, m_0 = 0$;
- (e) $m = 5, d = 7, m_0 = 0$.

13.4 Pentru fiecare cod determinat la exercițiul anterior determinați n, k și numărul de cuvinte - cod.

13.5 Arătați că un $(2^m - 1, k)$ - cod RS are 2^{mk} cuvinte - cod.

13.6 Fie codul RS de polinom generator $g(X) = (1 + X)(\alpha + X)(\alpha^2 + X)(\alpha^3 + X)(\alpha^4 + X)(\alpha^5 + X) \in GF(2^4)[X]$ unde α este rădăcina polinomului $1 + X + X^4$. Decodificați mesajele:

- (a) $0\alpha^3\alpha\alpha^5\alpha^3\alpha^2\alpha^6\alpha^{10}\alpha 000000$
- (b) $1\alpha^4\alpha^2\alpha 0010\alpha\alpha^5\alpha^3\alpha^2 0\alpha^{10}\alpha$
- (c) $\alpha 0\alpha^7 0\alpha^{12}\alpha^3\alpha^3 10000000$

13.7 Fie codul RS generat de $g(X) = (\alpha + X)(\alpha^2 + X)(\alpha^3 + X)(\alpha^4 + X)$ unde $\alpha \in GF(2^4)$ este rădăcina polinomului $1 + X + X^4$. Decodificați mesajele:

- (a) $001\alpha^8 00\alpha^5 00000000$
- (b) $0\alpha^{10} 0\alpha^6\alpha^{13} 0\alpha^8\alpha^{11}\alpha^3\alpha^5 000000$
- (c) $\alpha^4 0100\alpha^2\alpha^5\alpha^{12}\alpha^{14} 000000$

13.8 Plecând de la codul RS din exercițiul precedent, se formează codul scurt $C(4)$ de lungime $n = 11$ (și $k = 7$). Decodificați mesajele:

- (a) $001\alpha^8 00\alpha^5 0000$
- (b) $0\alpha^{10} 0\alpha^6\alpha^{13} 0\alpha^8\alpha^{11}\alpha^3\alpha^5 0$
- (c) $\alpha^4 0100\alpha^2\alpha^5\alpha^{12}\alpha^{14} 00$

13.9 Arătați că transformata Fourier finită M este inversabilă.

13.10 Fiind dat $GF(2^3)$ construit cu polinomul $1 + X + X^3$, găsiți matricea generatoare a unui cod DMS de lungime 7, pentru spațiul funcțiilor definit pe

$S = GF(2^3) \setminus \{0\}$, cu baza:

(a) $\{X, X^2, X^3\}$; (b) $\{1, X, X^2, X^3, X^4\}$; (c) $\{X, X^2, X^3\}$.

Arătați că toate aceste coduri sunt ciclice. Determinați polinomul generator pentru fiecare cod.

13.11 Fiind dat $GF(2^3)$ construit cu $1 + X + X^3$, determinați pentru fiecare $G(X)$ reprezentarea \mathbf{v}_g :

(a) $G(X) = X + \alpha X^3$, (b) $G(X) = 1 + X^2 + X^4$.

13.12 În aceleași condiții din exercițiul precedent, determinați $G(X)$ știind valorile lui \mathbf{v}_g :

(a) $\mathbf{v}_g = \alpha^3 \alpha \alpha^4 \mathbf{0} \alpha^6 \alpha^5 \alpha^2$; (b) $\mathbf{v}_g = \alpha^4 \alpha^2 \alpha \alpha^3 \mathbf{0} \alpha^6 \mathbf{1}$.

13.13 Folosind codul definit în Exemplul 18.7, decodificați mesajele:

(a) 11101 11110 11010 00111 11110 10100 10110 10100 0 ... 0

(b) 11000 00000 01010 11111 11011 00000 10001 00101 0 ... 0

(c) 00000 10000 10000 10000 00101 10100 11101 0 ... 0.

Capitolul 14

Alte clase de coduri ciclice binare

14.1 Coduri Hamming ciclice

Să considerăm spațiul nul al matricii

$$H = (\mathbf{1} \ \alpha \ \alpha^2 \ \dots \ \alpha^{n-1}),$$

unde $\alpha \in GF(2^p)$ este primitiv, deci $n = \text{ord}(\alpha) = 2^p - 1$.

Cum α poate fi considerat ca un vector coloană cu p componente, H este o matrice $p \times (2^p - 1)$ în care toate coloanele sunt distincte. Acesta este după cum se știe (Prelegerea 3) un cod Hamming $(2^p - 1, 2^p - p - 1)$ - construit însă ca un cod ciclic.

Exemplul 14.1 Fie $\alpha \in GF(2^3)$, α rădăcină a polinomului ireductibil $1 + X + X^3$. Am văzut că $\text{ord}(\alpha) = 7$, deci α este primitiv. Construim matricea

$$H = (\mathbf{1} \ \alpha \ \alpha^2 \ \alpha^3 \ \alpha^4 \ \alpha^5 \ \alpha^6) = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Ea este matricea de control pentru un $(7, 4)$ - cod Hamming binar.

Deci codul studiat în Exemplul 3.1 (Prelegerea 3) poate fi tratat și ca un cod ciclic de polinom generator $h(X) = 1 + X + X^3$.

Un cod Hamming binar ciclic se definește cerând ca orice polinom - cod $\{f(X)\}$ să admită ca rădăcină un element primitiv $\alpha \in GF(2^p)$. Generatorul unui asemenea cod este chiar polinomul minimal al lui α : $g(X) = m(X)$.

Deoarece și α^2 este rădăcină a polinoamelor - cod (Teorema 7.9), codul Hamming binar ciclic este un caz particular de cod BCH cu $d = 3$, capabil să corecteze o eroare.

Codificarea se realizează folosind un circuit liniar cu $p = n - k$ elemente de înmagazinare (vezi paragraful 5.4.2, Prelegerea 5), corespunzător polinomului minimal $m(X)$. Calculul sindromului cuvântului recepționat $\{r(X)\}$ - adică $\mathbf{r}H^T = r(\alpha)$ se realizează folosind același circuit de împărțire cu $m(X)$. Acest sindrom - în cazul în care este nenul, deci a intervenit o eroare simplă - este una din coloanele matricii H , adică una din valorile $\mathbf{1}, \alpha, \alpha^2, \dots, \alpha^{n-1}$.

Pentru a afla care componentă a fost afectată de eroare, se poate folosi același circuit în felul următor: neglijăm intrarea și lăsăm circuitul să funcționeze, având inițial în elementele de înmagazinare sindromul $r(\alpha)$, care este de forma

$$\alpha^j \quad (0 < j \leq 2^p - 2).$$

La fiecare tact, circuitul înmulțește conținutul elementelor de înmagazinare cu α ; deci, după $n - j = 2^p - 1 - j$ momente, în elementele de înmagazinare se obține

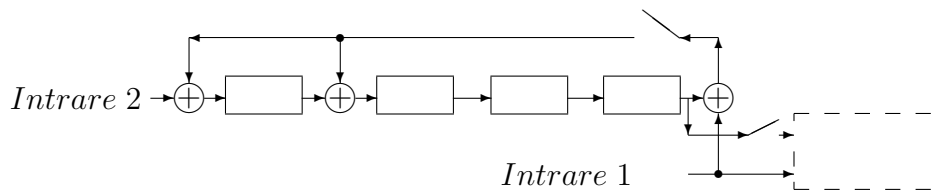
$$\alpha^{j+2^p-1-j} = \alpha^{2^p-1} = \alpha^n = \alpha^0 = \mathbf{1} = (1, 0, \dots, 0).$$

Cum componentele cuvântului recepționat sunt scrise în ordinea descrescătoare a puterilor (prima componentă este coeficientul termenului de rang maxim), înseamnă că va trebui să corectăm componenta $n - j$ din cuvânt, înlocuind 0 cu 1 sau 1 cu 0.

Exemplul 14.2 Fie $\alpha \in GF(2^4)$, rădăcină a polinomului $1 + X + X^4$. Codul Hamming binar (15,11) obținut folosind ca rădăcină pe α - deci de polinom generator $g(X) = 1 + X + X^4$, are matricea de control

$$H = (\mathbf{1} \ \alpha \ \dots \ \alpha^{14}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Circuitul care realizează (în funcție de necesități) codificarea, calculul sindromului și corectarea erorilor simple este (vezi și paragraful 5.4.2):



Codificarea se obține introducând la Intrarea 1 mesajul de informație de lungime k , mesaj considerat ca un polinom de gradul $n - 1$ cu primii coeficienți (corespunzători termenilor de rang mare) elementele mesajului informațional și cu ultimii $n - k$ coeficienți nuli.

La momentul inițial în elementele de înmagazinare se află numai 0, circuitul feedback este închis, iar circuitul spre canal este deschis. De asemenea, Intrarea 2 este ignorată.

Simultan cu transmiterea pe canal, elementele de informație se introduc și în circuitul liniar. După k momente, aici se află pozițiile de control.

În acest moment se decuplează feedback-ul, se cuplează legătura spre canal și se ignoră cele două intrări. Timp de $n - k$ tacti, elementele de control înmagazinate sunt trimise pe canal, imediat după mesajul de informație.

Pentru calculul sindromului: se neglijează canalul, se închide feedback-ul, iar în elementele de înmagazinare se găsește inițial $(0, 0, \dots, 0)$. Cuvântul recepționat este introdus prin Intrarea 2. După n tacti, aici se va găsi $r(\alpha)$.

Pentru corectarea erorii: se neglijează cele două intrări, se închide feedback-ul și se lasă să funcționeze circuitul până când în elementele de înmagazinare se obține vectorul $(1, 0, \dots, 0)$. Numărul de tacti scurși dă indicele componentei din mesajul recepționat, care trebuie corectată prin complementare.

Să considerăm spațiul nul al matricii

$$H = \begin{pmatrix} \mathbf{1} & \alpha & \alpha^2 & \dots & \alpha^{2^p-2} \\ 1 & 1 & 1 & \dots & 1 \end{pmatrix},$$

unde $\alpha \in GF(2^p)$ este primitiv. H are $p + 1$ linii și $n = 2^p - 1$ coloane (pe prima coloană, primele p poziții formează vectorul $\mathbf{1} = \alpha^0$, iar pe a $p + 1$ -a poziție se află scalarul $1 \in Z_2$). Codul obținut este codul Hamming binar extins, capabil să corecteze o eroare și să detecteze două erori (vezi și Algoritm B , Prelegerea 3).

Un asemenea cod se poate defini cerând ca orice cuvânt - cod $\{f(X)\}$ să aibă ca rădăcini α și $\mathbf{1}$. Generatorul unui astfel de cod ciclic este $g(X) = (1 + X)m(X)$, unde $m(X)$ este polinomul minimal al lui α .

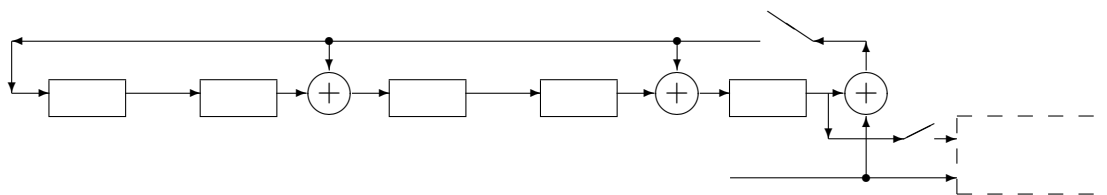
Codificarea în acest caz se face – ca mai sus – folosind circuitul cu $n - k + 1 = p + 1$ elemente de înmagazinare. Pentru calculul sindromului cuvântului recepționat $\{r(X)\}$ se folosesc însă practic două circuite: unul de împărțire la $m(X)$ (pentru calculul lui $r(\alpha)$) și altul de împărțire la $1 + X$ (pentru $r(\mathbf{1})$). Discuția este următoarea:

- Dacă $r(\alpha) = r(\mathbf{1}) = \mathbf{0}$, atunci nu a intervenit nici o eroare (sau avem o eroare nedetectabilă);
- Dacă $r(\alpha) \neq \mathbf{0}$, $r(\mathbf{1}) \neq \mathbf{0}$, avem o eroare simplă care se corectează cu circuitul de împărțire cu $m(X)$, după procedeul descris în Exemplul 20.3;
- Dacă $r(\alpha) \neq \mathbf{0}$, $r(\mathbf{1}) = \mathbf{0}$, atunci au fost detectate două erori.

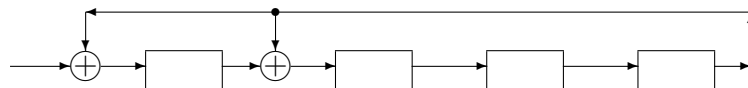
Exemplul 14.3 Să extindem codul Hamming binar din Exemplul 20.3. Vom avea polinomul generator

$$g(X) = (1 + X)(1 + X + X^4) = 1 + X^2 + X^4 + X^5$$

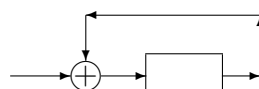
Circuitul liniar care codifică mesajele de informație (de lungime $k = 10$) este:



Sindromul se calculează cu ajutorul circuitului liniar de împărțire cu polinomul $m(X) = 1 + X + X^4$:



iar $r(\mathbf{1})$ se obține din circuitul de împărțire la $1 + X$:



Circuitul liniar construit pentru $m(X)$ efectuează și corectarea erorilor simple.

14.2 Coduri Hamming ciclice nebinare

Am studiat codurile Hamming atât sub forma liniară, cât și ca un caz particular de coduri Reed - Muller sau coduri ciclice. În această ultimă situație, ele se pot implementa cu ajutorul circuitelor liniare și – fapt important – admit o variantă de generalizare naturală la cazul nebinar.

Definiția 14.1 Fie $\alpha \in GF(q^p)$ un element primitiv. Se definește codul Hamming ciclic C astfel:

$$\{f(X)\} \in C \iff f(X) \text{ are rădăcina } \beta = \alpha^{q-1}.$$

sau – echivalent – C are matricea de control $H = (\mathbf{1} \ \beta \ \beta^2 \ \dots \ \beta^{n-1})$ unde $n = \text{ord}(\beta)$.

În cazul $q = 2$ se obține definiția codului Hamming binar.

Din

$$\beta^n = \mathbf{1} = (\alpha)^{q^p-1} = (\alpha^{q-1})^{\frac{q^p-1}{q-1}} \text{ rezultă } n = \frac{q^p-1}{q-1}.$$

Se observă imediat că H are un număr maxim de coloane distincte și nenule.

Propoziția 14.1 $(n, q-1) = 1$ dacă și numai dacă $(p, q-1) = 1$.

Demonstrație: Din $q-1 \mid q^j-1$ ($1 \leq j \leq p$) rezultă $q^j = (q-1)s_j + 1$. Deci

$$n = \frac{q^p-1}{q-1} = q^{p-1} + q^{p-2} + \dots + q + 1 = (q-1)(s_{p-1} + \dots + s_1) + p$$

și atunci

$$(p, q-1) = 1 \iff (n, q-1) = 1. \quad \text{q.e.d.}$$

Teorema 14.1 Spațiul nul al matricii $H = (\mathbf{1} \ \beta \ \beta^2 \ \dots \ \beta^{n-1})$ unde $\beta = \alpha^{q-1}$,

$n = \frac{q^p-1}{q-1}$ și $\alpha \in GF(q^p)$ primitiv, are distanța minimă $d \geq 3$ dacă și numai dacă $(n, q-1) = 1$.

Demonstrație: " \Leftarrow ": Să presupunem că există două coloane liniar dependente în matricea H . Deci $\beta^i = a\beta^j$ cu $a \in Z_q$, adică $\beta^{i-j} = a$.

Elementele nenule din Z_q sunt rădăcinile ecuației $X^{q-1} - 1 = 0$ (Teorema 7.3). Mai mult, ele sunt chiar primele $q-1$ puteri ale lui α^n . Într-adevăr, pentru orice $s = 0, 1, \dots, q-2$ avem

$$(\alpha^{ns})^{q-1} = (\alpha^{q^p-1})^s = \mathbf{1}^s = \mathbf{1}.$$

Deci pentru orice $a \in Z_q \setminus \{0\}$ există s ($0 \leq s \leq q-1$) cu $a = \alpha^{ns}$. Se poate scrie atunci $\beta^{i-j} = \alpha^{ns}$, adică $\alpha^{(i-j)(q-1)} = \alpha^{ns} \Rightarrow (i-j)(q-1) = ns$.

Dar $s < q-1$ iar $(n, q-1) = 1$, deci $i = j$, de unde rezultă $a = 1$, $\beta^i = \beta^j$, imposibil pentru că toate coloanele lui H sunt distincte. Rezultă că orice două coloane din H sunt liniar independente, adică (Teorema 2.4) $d \geq 3$.

" \Rightarrow ": Dacă $q-1$ și n nu sunt prime între ele, atunci ecuația anterioară admite o soluție cu $i \neq j$; deci există două coloane distincte liniar dependente β^i și β^j în matricea H , ceea ce contrazice faptul că $d \geq 3$. q.e.d.

Pentru operația de codificare se folosește circuitul de împărțire cu polinomul minimal $m(X)$ al lui β (care aici este chiar polinomul generator al codului Hamming ciclic C). Același circuit va calcula sindromul $r(\beta)$ al polinomului recepționat $\{r(X)\}$.

În cazul în care a intervenit o eroare, $r(\beta)$ este de forma $b\beta^j$ (eroarea b a apărut pe poziția β^j). Funcționarea secvențială a circuitului liniar (ignorând intrarea) revine la înmulțirea succesivă cu β . Deci, după $n - j$ tacti, în elementele de înmagazinare se obține vectorul $(b \ 0 \ \dots \ 0)$. Atunci, din componenta $n - j$ a cuvântului recepționat se va scădea eroarea b . De observat că – deoarece coloanele lui H sunt nenule și distincte – primul tact în care se obține un vector cu toate componentele nule înafară de prima, este $n - j$.

Exemplul 14.4 Să considerăm $p = q = 3$ (astfel, $(m, q - 1) = 1$). Folosind rădăcina α a polinomului primitiv $1 + 2X + X^3$ vom genera extensia $GF(3^3)$. Fie $\beta = \alpha^{q-1} = \alpha^2$. Deoarece $\alpha^{3^3-1} = \alpha^{26} = 1$, avem $\text{ord}(\beta) = 13$, deci $n = 13$. Pentru a construi polinomul minimal $m(X)$ al lui β , să observăm (Secțiunea 7.3) că el are ca rădăcini α^2, α^6 și α^{18} ($\alpha^{54} = \alpha^2$). Deci $m(X) = a + bX + cX^2 + X^3$. Punând condiția $m(\alpha^2) = a + b\alpha^2 + c\alpha^4 + \alpha^6 = \mathbf{0}$, obținem ecuația matricială:

$$a \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + c \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

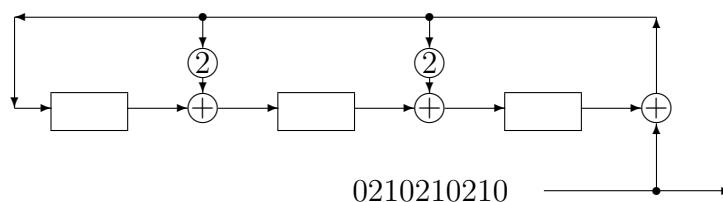
cu soluția $a = 2, b = c = 1$. Deci $m(X) = 2 + X + X^2 + X^3$.

Codul Hamming ciclic de polinom generator $g(X) = m(X) = 2 + X + X^2 + X^3$ are matricea de control

$$H = (\mathbf{1} \ \alpha^2 \ \alpha^4 \ \dots \ \alpha^{24}) = \begin{pmatrix} 1 & 0 & 0 & 1 & 2 & 0 & 2 & 0 & 1 & 1 & 1 & 2 & 1 \\ 0 & 0 & 2 & 1 & 0 & 1 & 0 & 2 & 2 & 2 & 1 & 2 & 1 \\ 0 & 1 & 1 & 1 & 2 & 1 & 1 & 0 & 0 & 1 & 2 & 0 & 2 \end{pmatrix}.$$

Am definit astfel $(13, 10)$ - codul Hamming ternar, evocat în Exemplul 3.5 relativ la Problema Pronosportului.

Să considerăm mesajul de informație 0120120120. Pentru operația de codificare se va folosi circuitul (pentru detalii, a se revedea secțiunea 5.4.2):



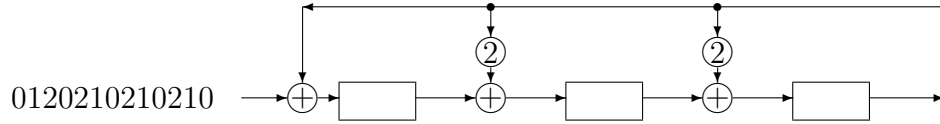
(s-a ținut cont că se lucrează în Z_3 , deci $-1 = 2, -2 = 1$). În primii 10 tacti de funcționare, biții de informație sunt trimiși pe canal și – simultan – intră în circuitul liniar, a cărui funcționare este (pe prima coloană se află informația, iar pe celelalte trei – conținuturile elementelor de înmagazinare):

0	0	0	0		2	2	1	2
1	1	2	2		0	2	0	2
2	1	0	1		1	0	2	0
0	1	0	2		2	2	1	0
1	0	1	0		0	0	2	1

În elementele de înmagazinare au rămas caracterele de control 021. Ele sunt trimise prin canal la tacti 11 – 13 de la dreapta spre stânga, cu semn schimbat: în ordine $-1 =$

2, $-2 = 1$, $-0 = 0$. Deci cuvântul - cod transmis a fost $\mathbf{a} = 0120120120210$. Se verifică imediat că $\mathbf{a}H^T = \mathbf{0}$.

La recepție, primele 10 caractere sunt cele de informație. Pentru a verifica dacă au apărut sau nu erori, se folosește același circuit, dar cu altă intrare:



Funcționarea acestui circuit liniar timp de 13 tacti, este dată de tabelul:

	0	0	0		0	1	2	0
0	0	0	0		1	1	1	2
1	1	0	0		2	1	2	2
2	2	1	0		0	2	2	0
0	0	2	1		2	2	2	2
1	2	2	1		1	0	0	0
2	0	1	1		0	0	0	0

Sindromul $a(\alpha^2)$ obținut este 000, deci nu a apărut nici o eroare.

Dacă s-ar fi recepționat însă cuvântul $\mathbf{a}' = 0120120100210$, după funcționarea circuitului timp de 13 tacti s-ar fi ajuns în elementele de înmagazinare la 220. Cum acesta este nenul, circuitul se lasă să funcționeze în continuare, ignorând intrarea, până se ajunge la un sindrom de forma $b00$ cu $b \neq 0$. Mai exact, funcționarea circuitului va fi (pe prima coloană se numără tactii):

0	2	2	0		5	1	0	1
1	0	2	2		6	1	0	2
2	2	1	0		7	2	2	1
3	0	2	1		8	1	1	1
4	1	2	1		9	1	0	0

Deci, după 9 tacti s-a ajuns la 100. Corecția se face astfel: din componenta a 9 - a a cuvântului recepționat se scade $b = 1$, obținându-se $0 - 1 = 2 \pmod{3}$.

Observația 14.1 Definiția codurilor Hamming ciclice nebinare nu este echivalentă cu cea a codurilor Hamming nebinare dată în Secțiunea 3.1.1. De exemplu, codul (dat în Exemplitul 3.4) cu matricea de control

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

este un $(4,2)$ - cod Hamming ternar. El conține 9 cuvinte

$$\{(0000, 1012, 2021, 0111, 1120, 2102, 0222, 1201, 2210)\},$$

care - evident - nu formează un cod ciclic.

Deci între cele două definiții ale codurilor Hamming nebinare (liniare și ciclice) există doar o relație de incluziune strictă (orice cod Hamming ciclic este un cod Hamming liniar).

14.3 Coduri Goppa

Această clasă de coduri a fost introdusă de Goppa în 1970. Deși în general sunt coduri neciclice, ele constituie o generalizare a codurilor BCH și formează o punte între teoria codurilor și criptografie. Din acest motive, prezentarea lor este deosebit de utilă.

Fie q un număr prim și $g(X) \in Z_q[X]$. Dacă $a \in Z_q$ este un element cu proprietatea $g(a) \neq 0$, vom nota

$$\frac{1}{X-a} = -\frac{g(X) - g(a)}{X-a}g(a)^{-1}.$$

Propoziția 14.2 În algebra polinoamelor din $Z_q[X]$ modulo $g(X)$, polinomul $\frac{1}{X-a}$ este inversul lui $X-a$.

Demonstrație: Faptul că $\frac{1}{X-a}$ este polinom se verifică imediat. Pe de altă parte,

$$(X-a)\frac{1}{X-a} = [g(a) - g(X)]g(a)^{-1} = 1 - g(a)^{-1}g(X) \equiv 1 \pmod{g(X)}. \quad \text{q.e.d.}$$

Definiția 14.2 Un cod Goppa de lungime n peste Z_q este determinat prin:

1. Un polinom $g(X) \in Z_q[X]$ de grad t ($t \geq 2$) definit peste o extensie $GF(q^m)$ a lui Z_q ;
2. $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n \in GF(q^m)$ cu $g(\mathbf{a}_i) \neq 0$ ($1 \leq i \leq n$).

Codul conține toate cuvintele $\mathbf{v} = v_1v_2 \dots v_n \in Z_q^n$ pentru care $\sum_{i=1}^n \frac{v_i}{X - \mathbf{a}_i} = \mathbf{0}$.

Teorema 14.2 Codul Goppa conține numai cuvintele $\mathbf{v} = v_1v_2 \dots v_n$ pentru care

$$\sum_{i=1}^n v_i g(\mathbf{a}_i)^{-1} \mathbf{a}_i^s = \mathbf{0} \quad (0 \leq s \leq t-1)$$

unde $t = \text{grad}(g(X))$.

Demonstrație: Să explicităm polinomul $\frac{1}{X-a}$. Dacă $g(X) = g_0 + g_1X + \dots + g_tX^t$ ($g_t \neq 0$), atunci pentru orice $a \in GF(q^m)$ cu $g(a) \neq 0$, are loc descompunerea

$$\frac{g(X) - g(a)}{X-a} = g_tX^{t-1} + (ag_t + g_{t-1})X^{t-2} + (a^2g_t + ag_{t-1} + g_{t-2})X^{t-3} + \dots + (a^{t-1}g_t + a^{t-2}g_{t-1} + \dots + g_1).$$

Deci coeficientul lui X^{p-1} ($1 \leq p \leq t$) în polinomul $\frac{1}{X-a}$ este $-g(a)^{-1} \sum_{j=p}^t a^{t-j} g_j$.

Ecuția caracteristică a codurilor Goppa $\sum_{i=1}^n \frac{v_i}{X - \mathbf{a}_i} = \mathbf{0}$ implică faptul că toți coeficienții lui X^{p-1} ($1 \leq p \leq t$) sunt nuli. Deci

$$\sum_{i=1}^n v_i g(\mathbf{a}_i)^{-1} \sum_{j=p}^t \mathbf{a}_i^{t-j} g_j = \mathbf{0}, \quad (1 \leq p \leq t).$$

Cazul $p = t$ conduce la $g_t \sum_{i=1}^n v_i g(\mathbf{a}_i)^{-1} = \mathbf{0}$ și – deoarece $g_t \neq 0$ – se obține

$$\sum_{i=1}^n v_i g(\mathbf{a}_i)^{-1} = \mathbf{0}.$$

Din cazul $p = t - 1$ se obține $g_t \sum_{i=1}^n v_i g(\mathbf{a}_i)^{-1} \mathbf{a}_i + g_{t-1} \sum_{i=1}^n v_i g(\mathbf{a}_i)^{-1} = \mathbf{0}$. Cum a doua sumă este zero, iar $g_t \neq 0$, va rezulta $\sum_{i=1}^n v_i g(\mathbf{a}_i)^{-1} \mathbf{a}_i = \mathbf{0}$ ș.a.m.d.

Am obținut echivalența

$$\sum_{i=1}^n \frac{v_i}{X - a_i} = \mathbf{0} \iff \sum_{i=1}^n v_i g(\mathbf{a}_i)^{-1} \mathbf{a}_i^s = \mathbf{0} \quad (0 \leq s \leq t - 1)$$

q.e.d.

Corolarul 14.1 Codul Goppa determinat de un polinom $g(X)$ de gradul t are matricea de control

$$H = \begin{pmatrix} \frac{1}{g(a_1)} & \frac{1}{g(a_2)} & \frac{1}{g(a_3)} & \cdots & \frac{1}{g(a_n)} \\ \frac{a_1}{g(a_1)} & \frac{a_2}{g(a_2)} & \frac{a_3}{g(a_3)} & \cdots & \frac{a_n}{g(a_n)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{a_1^{t-1}}{g(a_1)} & \frac{a_2^{t-1}}{g(a_2)} & \frac{a_3^{t-1}}{g(a_3)} & \cdots & \frac{a_n^{t-1}}{g(a_n)} \end{pmatrix}.$$

Demonstrație: Este lăsată ca exercițiu.

Exemplul 14.5 Polinomul $1 + X + X^3 \in \mathbb{Z}_2[X]$ este ireductibil. Fie α o rădăcină a sa, cu ajutorul căreia se generează extensia $GF(2^3)$. Dacă alegem $g(X) = 1 + X + X^2$ și drept elemente \mathbf{a}_i cele 8 componente ale lui $GF(2^3)$, vom obține un cod Goppa de lungime 8, cu matricea de control

$$H = \begin{pmatrix} \frac{1}{g(0)} & \frac{1}{g(1)} & \frac{1}{g(\alpha)} & \cdots & \frac{1}{g(\alpha^6)} \\ \mathbf{0} & \frac{1}{g(1)} & \frac{\alpha}{g(\alpha)} & \cdots & \frac{\alpha^6}{g(\alpha^6)} \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \mathbf{1} & \alpha^2 & \alpha^4 & \alpha^2 & \alpha & \alpha & \alpha^4 \\ \mathbf{0} & \mathbf{1} & \alpha^3 & \alpha^6 & \alpha^5 & \alpha^5 & \alpha^6 & \alpha^3 \end{pmatrix} =$$

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

Acest cod este format din numai patru cuvinte:

$$\{00000000, \quad 00111111, \quad 11001011, \quad 11110100\}.$$

El are distanța minimă 5; s-a obținut deci un $(8, 2)$ - cod binar corector de două erori. De remarcat că acesta nu este un cod ciclic.

Observația 14.2 Să considerăm un cod BCH definit prin rădăcinile

$$\alpha^{m_0}, \alpha^{m_0+1}, \dots, \alpha^{m_0+d-2}$$

unde $\alpha \in GF(q^m)$ are ordinul n . Acest cod se poate scrie ca un cod Goppa astfel: se ia polinomul $g(X) = X^{m_0+d-2}$ și elementele $\alpha^0, \alpha^{-1}, \dots, \alpha^{-(n-1)}$. Vom avea pentru orice $i = 0, 1, \dots, d-2$ și $p = 0, 1, \dots, n-1$:

$$\frac{(\alpha^{-p})^i}{(\alpha^{-p})^{m_0+d-2}} = (\alpha^p)^{m_0+d-2-i} = (\alpha^{m_0+d-2-i})^p$$

Pe baza aceasta, codul Goppa rezultat are matricea de control

$$H = \begin{pmatrix} \mathbf{1} & \alpha^{m_0+d-2} & (\alpha^{m_0+d-2})^2 & \dots & (\alpha^{m_0+d-2})^{n-1} \\ \mathbf{1} & \alpha^{m_0+d-3} & (\alpha^{m_0+d-3})^2 & \dots & (\alpha^{m_0+d-3})^{n-1} \\ & & \vdots & & \\ \mathbf{1} & \alpha^{m_0} & (\alpha^{m_0})^2 & \dots & (\alpha^{m_0})^{n-1} \end{pmatrix}$$

care coincide cu matricea de control a codului BCH considerat, în care s-a efectuat o inversare a liniilor.

Se poate arăta (Van Lint) că singurele coduri Goppa ciclice sunt codurile BCH.

Propoziția 14.3 Codul Goppa binar de lungime n , determinat de un polinom de grad t peste $GF(2^m)$ are $k \geq n - mt$ simboluri de informație și distanța minimă $\geq t + 1$.

Demonstrație: Matricea de control a codului Goppa are t linii în $GF(2^m)$, deci mt linii scrise în binar. Rezultă că numărul $n - k$ de simboluri de control este cel mult mt : $n - k \leq mt$.

Pe de-altă parte, se arată ușor că orice combinație liniară de t coloane din H este liniar independentă. q.e.d.

Definiția 14.3 Fie $g(X)$ un polinom ireductibil peste $GF(q^m)$. Codul Goppa determinat de $g(X)$ și de toate elementele lui $GF(q^m)$ se numește ireductibil.

Ideea de "ireductibil" este similară codurilor ciclice ireductibile: un cod Goppa ireductibil nu are subcoduri Goppa proprii.

Codul prezentat în Exemplul 17.3 este un cod Goppa ireductibil. Despre codurile Goppa ireductibile se poate prezenta următorul rezultat:

Teorema 14.3 Un cod Goppa binar ireductibil, determinat de un polinom de gradul t poate corecta cel puțin t erori independente.

Pentru demonstrație va trebui să facem o construcție suplimentară.

Definiția 14.4 Derivata formală a polinomului $a(X) = a_0 + a_1X + \dots + a_nX^n$ este definită prin $a'(X) = a_1 + 2a_2X + 3a_3X^2 + \dots + na_nX^{n-1}$.

Lema 14.1 Pentru derivata formală sunt adevărate egalitățile:

1. $(a(X) + b(X))' = a'(X) + b'(X)$;

$$2. (a(X)b(X))' = a'(X)b(X) + a(X)b'(X)$$

Demonstrație: Exercițiu.

Lema 14.2 Într-un corp de caracteristică 2 derivata formală a unui polinom este un pătrat perfect.

Demonstrație: Deoarece pentru n par, $(X^n)' = nX^{n-1} = 0$, derivata formală a unui polinom va avea numai puteri pare. Cum fiecare coeficient se poate scrie ca un pătrat perfect (se lucrează într-un corp de caracteristică 2), avem $f'(X) = f_0^2 + f_2^2 X^2 + f_4^2 X^4 + \dots$ și, dacă se ia $g(X) = f_0 + f_2 X + f_4 X^2 + \dots$, avem $f'(X) = g^2(X)$. q.e.d.

Să revenim acum la *Demonstrația Teoremei 20.2*: Fie $g(X) \in Z_2[X]$ un polinom ireductibil de grad t ($t \geq 2$) peste $GF(2^m) = \{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\}$ ($n = 2^m$), α o rădăcină a lui g ($g(\alpha) = 0$) și $\mathbf{v} = v_1 v_2 \dots v_n$ un cuvânt al codului Goppa, $w(\mathbf{v}) = s$. Notăm $v_{i_1}, v_{i_2}, \dots, v_{i_s}$ biții egali cu 1 din \mathbf{v} .

Deoarece \mathbf{v} este cuvânt - cod, avem $\sum_{i=1}^n \frac{v_i}{X - \mathbf{a}_i} = \sum_{k=1}^s \frac{1}{X - \mathbf{a}_{i_k}} = \mathbf{0}$.

Polinomul $f(X) = (X - \mathbf{a}_{i_1})(X - \mathbf{a}_{i_2}) \dots (X - \mathbf{a}_{i_s})$ nu este divizibil cu $g(X)$ (deoarece $g(X)$ este ireductibil de grad t). Rezultă că, în algebra claselor de resturi $Z_2[X]/g(X)$, $f(\alpha) \neq 0$ (Teorema 4.1) deci admite invers. În consecință, există un polinom $r(X)$ cu $\{f(\alpha)\}\{r(\alpha)\} = 1$.

Să calculăm derivata formală $f'(X) = \sum_{p=1}^s \frac{f(X)}{X - \mathbf{a}_{i_p}}$. Făcând $X = \alpha$ și înmulțind relația cu $r(\alpha)$, obținem

$$\{f'(\alpha)\}\{r(\alpha)\} = \sum_{p=1}^s \frac{\{f(\alpha)\}\{r(\alpha)\}}{\alpha - \mathbf{a}_{i_p}} = \sum_{p=1}^s \frac{1}{\alpha - \mathbf{a}_{i_p}} = 0 \pmod{g(X)}.$$

Aceasta înseamnă că $f'(X)r(X)$ este divizibil cu $g(X)$. Cum $g(X)$ este ireductibil și nu divide $r(X)$ (deoarece $r(\alpha) \neq 0$), rezultă $g(X)|f'(X)$.

Deoarece suntem într-un corp de caracteristică 2, conform Lemei 15.1 derivata formală este un pătrat perfect: $f'(X) = h^2(X)$. Din faptul că $g(X)$ este divizor ireductibil al lui $h^2(X)$ rezultă $g(X)|h(X)$.

Deci $g^2(X)|h^2(X) = f'(X)$.

Se știe că $\text{grad}(f'(X)) = s - 1$ și $s \geq 1$; deci $\text{grad}(f'(X)) \geq \text{grad}(h^2(X))$, adică $s - 1 \geq 2t$.

Cum ponderea oricărui cuvânt - cod este $\geq 2t + 1$, rezultă că distanța minimă a codului este cel puțin $2t + 1$, deci codul corectează sigur t erori independente. q.e.d.

Exemplul 14.6 Să considerăm $GF(2^3)$ generat de o rădăcină a lui $g(X) = 1 + X + X^3$. Deoarece $g(X)$ are 3 rădăcini în $GF(2^3)$, el nu va avea nici o rădăcină în corpul $GF(2^5)$ ($GF(2^5)$ nu este o extensie a lui $GF(2^3)$), deci aici $g(X)$ este ireductibil.

Codul Goppa ireductibil corespunzător are $n = 32$, $k \geq 32 - 5 \cdot 3 = 17$, $d \geq 7$; deci este un $(32, 17)$ - cod corector de 3 erori.

Decodificarea codurilor Goppa se face în manieră similară decodificării codurilor BCH. Astfel, fie C un cod Goppa definit de $g(X) \in Z_q[X]$ ($\text{grad}(g(X)) = t$), și $\mathbf{a}_1, \dots, \mathbf{a}_n \in GF(q^m)$; considerăm $\mathbf{v} = v_0v_1 \dots v_{n-1}$ un cuvânt - cod și $\mathbf{r} = r_0r_1 \dots r_{n-1}$ cuvântul recepționat după transmiterea lui \mathbf{v} .

Vom nota cu $\mathbf{e} = \mathbf{r} - \mathbf{v} = e_0e_1 \dots e_{n-1}$ secvența - eroare.

Pentru $t/2 \leq w(\mathbf{e}) \leq t$ se poate folosi un algoritm din cei uzuali. În practică însă $t = \text{grad}(g(X))$ este mare și există algoritmi eficienți de decodificare pentru cazul $w(\mathbf{e}) < t/2$. Un astfel de algoritm vom construi mai jos.

Fie $M = \{i \mid e_i \neq 0\}$, $\text{card}(M) = j < t/2$; considerăm sindromul polinomial

$$\{s(X)\} = \sum_{i=0}^{n-1} \frac{e_i}{X - \mathbf{a}_i} \pmod{g(X)}.$$

De remarcat că $\{s(X)\}$ se poate determina prin calcul la primirea cuvântului \mathbf{r} . Se definesc polinoamele $\sigma(X)$ (de localizare a erorii) și $\omega(X)$ prin

$$\sigma(X) = \prod_{i \in M} (X - \mathbf{a}_i), \quad \omega(X) = \sum_{i \in M} e_i \prod_{k \in M \setminus \{i\}} (X - \mathbf{a}_k).$$

Din definiție rezultă că $\sigma(X)$ și $\omega(X)$ sunt prime între ele. În plus, $\text{grad}(\sigma(X)) = j$, $\text{grad}(\omega(X)) < j$. Între ele există relația

$$\{s(X)\}\{\sigma(X)\} = \sum_{i=0}^{n-1} \frac{e_i}{X - \mathbf{a}_i} \prod_{i \in M} (X - \mathbf{a}_i) \equiv \{\omega(X)\} \pmod{g(X)}. \quad (1)$$

$\sigma(X)$ și $\omega(X)$ sunt polinoamele normate de grad minim care verifică relațiile de sus. Într-adevăr, să presupunem că există $\sigma_1(X)$ un polinom nenul normat de grad $< j$ și $\omega_1(X)$ polinomul corespunzător, astfel ca

$$\{s(X)\}\{\sigma_1(X)\} \equiv \{\omega_1(X)\} \pmod{g(X)}$$

Fie $d(X) = \text{cmmdc}(s(X), g(X))$. Din relațiile

$s(X)\sigma(X) = a(X)g(X) + \omega(X)$, $s(X)\sigma_1(X) = b(X)g(X) + \omega_1(X)$,
rezultă că $d(X)$ va divide și polinoamele $\omega(X)$, $\omega_1(X)$. Deci,

$$\begin{aligned} \left\{ \frac{s(X)}{d(X)} \right\} \{\sigma(X)\} - \left\{ \frac{\omega(X)}{d(X)} \right\} &\equiv 0, \\ \left\{ \frac{s(X)}{d(X)} \right\} \{\sigma_1(X)\} - \left\{ \frac{\omega_1(X)}{d(X)} \right\} &\equiv 0 \quad \left(\text{mod } \frac{g(X)}{d(X)} \right) \end{aligned}$$

sau

$$\left\{ \frac{s(X)}{d(X)} \right\} \frac{\{\omega_1(X)\sigma(X) - \omega(X)\sigma_1(X)\}}{\{d(X)\}} \equiv 0 \pmod{g(X)}.$$

Deci $\{\omega_1(X)\sigma(X) - \omega(X)\sigma_1(X)\} \equiv 0 \pmod{g(X)}$. Deoarece polinomul din membrul stâng are grad mai mic decât t , rezultă că membrul stâng este 0. Apoi, cum $\text{cmmdc}(\sigma(X), \omega(X)) = 1$, deducem $\sigma(X) \mid \sigma_1(X) - \text{ceea ce contrazice faptul că } \text{grad}(\sigma_1(X)) < \text{grad}(\sigma(X))$ și $\{\sigma_1(X)\} \neq 0$.

Rezumând, procedeul de decodificare rapidă a mesajelor la codurile Goppa are loc astfel:

1. Din mesajul recepționat \mathbf{r} se calculează sindromul $s(X)$;
 2. Se determină polinomul normat $\sigma(X)$ de grad minim $< t/2$ care verifică relația (1) (un algoritm eficient a fost dat de Berlekamp).
Din rezolvarea ecuației $\sigma(X) = 0$ se află locațiile erorilor;
 3. Se determină e_i ($i \in M$) din relațiile $\omega(a_i) = e_i \prod_{j \in M \setminus \{i\}} (\mathbf{a}_i - \mathbf{a}_j)$.
- Pe baza acestor valori se construiește secvența - eroare \mathbf{e} ;
4. Cuvântul decodificat este $\mathbf{v} = \mathbf{r} - \mathbf{e}$.

14.4 Exerciții

14.1 Să se construiască matricea generatoare a (13, 10) - codului Hamming ternar din Exemplul 14.4.

14.2 În codul din Exemplul 14.4 să se codifice mesajele de informație:

- (a) 1122002211 (b) 0000111122 (c) 1020012210
 (d) 2200211101 (e) 2222222222 (f) 0000000000

14.3 Să se construiască toate codurile Hamming peste Z_3 și Z_5 .

14.4 Demonstrați Corolarul 15.1

14.5 Demonstrați Lema 14.1

14.6 Pentru $g(X) = X^{2t}$ arătați că $\frac{1}{X-a} = \sum_{j=1}^{2t} a^{-j} X^{j-1}$.

14.7 Găsiți distanța minimă a codului Goppa binar determinat de $g(X) = 1 + X^2$ și de elementele $\mathbf{0}, \alpha, \alpha^2 \in GF(2^2)$.

14.8 Polinomul generator este $g(X) = 1 + X^2$ iar parametrii sunt cele 15 elemente nenule ale extensiei $GF(2^4)$, generată de rădăcina polinomului $1 + X + X^4$. Analizați codul Goppa binar astfel construit.

14.9 Găsiți o matrice de control a (16, 8) - codului Goppa binar ireductibil, dat de $g(X) = \alpha^3 + X + X^2$, unde $\alpha \in GF(2^4)$ este primitiv.

14.10 Să se construiască codul Goppa ireductibil peste $GF(2^4)$, cu $g(X) = 1 + X + X^3$. Să se determine matricea de control și matricea generatoare.

Care este distanța minimă a codului ?

14.11 Folosind codul construit anterior, să se decodifice (folosind algoritmi de coduri liniare) cuvintele:

0110110010011110; 0111011001000000; 1011110101100100;
 1111111111111111; 1011011111001011; 1010010110000010.

14.12 Pentru același cod Goppa binar ireductibil, să se decodifice cuvintele

0101100101111001; 1100101100110100; 0001100010010110

folosind algoritmul rapid specific.

14.13 Verificați că polinomul $1 + X + X^2$ este ireductibil în $GF(2^5)$. Determinați parametrii și matricea de control a codului Goppa ireductibil corespunzător.

Capitolul 15

Coduri Preparata

În 1968 Preparata a introdus o clasă de coduri neliniare corectoare de două erori, care s-au dovedit foarte interesante prin proprietățile și prin legăturile pe care le au cu diverse alte coduri deja studiate. Ele sunt o combinație între codurile Reed - Muller și codurile BCH corectoare de două erori. Definiția pe care o vom folosi aparține lui Baker și permite o abordare mai simplă a algoritmilor de codificare/decodificare.

15.1 Definirea codurilor Preparata extinse

Să revenim la modalitatea de marcare a pozițiilor cuvintelor de lungime 2^r folosind elementele lui $GF(2^r)$, așa cum s-a definit în algoritmul de decodificare Peterson pentru codurile BCH. Deci:

Fie $r \geq 2$ un număr întreg, $\alpha \in GF(2^r)$ primitiv, și $U \subseteq GF(2^r)$; se notează $\chi(U)$ cuvântul de lungime 2^r definit prin

$$\begin{aligned} &1 \text{ pe poziția } i \text{ dacă } \alpha^i \in U \text{ (} 0 \leq i \leq 2^r - 2 \text{),} \\ &1 \text{ pe poziția } 2^r - 1 \text{ dacă } \mathbf{0} \in U, \\ &0 \text{ în rest.} \end{aligned}$$

Exemplul 15.1 Fie $\alpha \in GF(2^3)$ primitiv. Atunci

$$\begin{aligned} \chi(\{\mathbf{0}\}) &= 00000001 \\ \chi(\{\alpha^2, \alpha^5, \alpha^6\}) &= 00100110 \\ \chi(\emptyset) &= 00000000. \end{aligned}$$

Se definesc în mod natural operațiile:

$$\begin{aligned} U + \beta &= \{u + \beta | u \in U\}, & \beta U &= \{\beta u | u \in U\}, \\ U \Delta V &= \{u | u \in U \cup V, u \notin U \cap V\}, & U, V &\subset GF(2^r), \beta \in GF(2^r). \end{aligned}$$

Se verifică ușor relațiile

$$\chi(U) + \chi(V) = \chi(U \Delta V), \quad w(\chi(U)) = \text{card}(U). \quad (1)$$

Exemplul 15.2 Fie $GF(2^3)$ construit folosind rădăcina (primitivă) α a polinomului $1 + X + X^3$, și mulțimile $U = \{\alpha^2, \alpha^5, \alpha^6\}$, $V = \{\alpha^5, \mathbf{0}\}$. Atunci

$$\begin{aligned} U + \alpha^2 &= \{\alpha^2 + \alpha^2, \alpha^5 + \alpha^2, \alpha^6 + \alpha^2\} = \{\mathbf{0}, \alpha^3, \alpha^0\}, \\ \alpha^2 U &= \{\alpha^2 \alpha^2, \alpha^2 \alpha^5, \alpha^2 \alpha^6\} = \{\alpha^4, \alpha^0, \alpha\}, \\ \chi(U) + \chi(V) &= 00100110 + 00000101 = 00100011 = \chi(\{\alpha^2, \alpha^6, \mathbf{0}\}) = \chi(U \Delta V). \end{aligned}$$

Definiția 15.1 Fie $r > 2$ un număr impar. Codul extins Preparata $P(r)$ este mulțimea cuvintelor de forma $\chi(U)\chi(V)$ unde $U, V \subseteq GF(2^r)$ verifică condițiile

1. $\text{card}(U), \text{card}(V)$ sunt numere pare;

$$2. \sum_{u \in U} u = \sum_{v \in V} v;$$

$$3. \sum_{u \in U} u^3 + \left(\sum_{u \in U} u \right)^3 = \sum_{v \in V} v^3.$$

Pentru ușurință vom nota cuvintele - cod cu $[\chi(U), \chi(V)]$ (vezi și codul Golay).

Exemplul 15.3 În ipotezele din Exemplul 19.5, să considerăm $U = \{\alpha, \alpha^2, \alpha^5, \mathbf{0}\}$, $V = \{\alpha^0, \alpha, \alpha^2, \alpha^3, \alpha^6, \mathbf{0}\}$. Prima condiție din definiție este verificată. Pentru a doua condiție avem:

$$\sum_{u \in U} u = \alpha + \alpha^2 + \alpha^5 + \mathbf{0} = 010 + 001 + 111 + 000 = 100 = \alpha^0,$$

$$\sum_{v \in V} v = \alpha^0 + \alpha + \alpha^2 + \alpha^3 + \alpha^6 + \mathbf{0} = 100 + 010 + 001 + 110 + 101 + 000 = 100 = \alpha^0.$$

A treia condiție este și ea verificată deoarece

$$\sum_{u \in U} u^3 = \alpha^3 + \alpha^6 + \alpha + \mathbf{0} = 110 + 101 + 010 + 000 = 001 = \alpha^2,$$

$$\sum_{v \in V} v^3 = \alpha^0 + \alpha^3 + \alpha^6 + \alpha^2 + \alpha^4 + \mathbf{0} = 100 + 110 + 101 + 001 + 011 + 000 = 101 = \alpha^6$$

$$\text{și } \alpha^2 + (\alpha^0)^3 = \alpha^6.$$

Deci $[\chi(U), \chi(V)] = 01100101 11110011$ este cuvânt - cod în $P(3)$.

Observații:

- Deoarece $\text{card}(\chi(U)) = \text{card}(\chi(V)) = 2^r$, $P(r)$ este un cod de lungime 2^{r+1} .
- Faptul că $\mathbf{0}$ este sau nu un element al mulțimilor U sau V nu afectează cu nimic condițiile 2 și 3 din Definiția 15.1. Acest element se introduce în cele două mulțimi numai pentru a asigura prima condiție din definiție. Rezultă că biții de pe poziția $2^r - 1$ din $\chi(U)$ și $\chi(V)$ sunt biți de paritate (ceea ce justifică termenul de cod Preparata extins).
- Exponentul '3' din Definiția 15.1 nu este esențial. Dacă el se înlocuiește cu $s = 2^t + 1$ astfel ca aplicațiile $f, g : GF(2^r) \rightarrow GF(2^r)$, $f(x) = x^s$, $g(x) = x^{s-2}$ să fie bijective, toate proprietățile se păstrează.

Aceste coduri se numesc *Coduri Preparata generalizate*.

Lema 15.1 Fie $[\chi(U), \chi(V)], [\chi(A), \chi(B)] \in P(r)$ și $\beta = \sum_{u \in U} u$.

Atunci $[\chi(U \Delta A + \beta), \chi(V \Delta B)] \in P(r)$.

Demonstrație: Vom arăta că noul cuvânt construit verifică condițiile din Definiția 15.1.

1. Deoarece $card(U), card(V), card(A), card(B)$ sunt pare, se arată ușor că:
 $card(V \Delta B) = card(V) + card(B) - 2 \cdot card(V \cap B)$,
 $card(U \Delta A + \beta) = card(U \Delta A) = card(U) + card(A) - 2 \cdot card(U \cap A)$.
2. Să observăm întâi că dacă $I, J \subseteq GF(2^r)$, avem $\sum_{x \in I \Delta J} x = \sum_{x \in I} x + \sum_{x \in J} x$ deoarece orice element $\alpha^i \in I \cap J$ apare de două ori în membrul drept și niciodată în membrul stâng, iar $\alpha^i + \alpha^i = 0$. Deci:

$$\sum_{x \in U \Delta A + \beta} x = \sum_{y \in U \Delta A} (y + \beta) = \sum_{y \in U \Delta A} y + \beta \cdot card(U \Delta A) = \sum_{y \in U} y + \sum_{y \in A} y + \mathbf{0} \text{ (deoarece } card(U \Delta A) \text{ este par)} = \sum_{y \in V} y + \sum_{y \in B} y = \sum_{y \in V \Delta B} y.$$

$$\begin{aligned} 3. \quad \sum_{x \in U \Delta A + \beta} x^3 + \left(\sum_{x \in U \Delta A + \beta} x \right)^3 &= \sum_{y \in U \Delta A} (y + \beta)^3 + \left(\sum_{y \in V \Delta B} y \right)^3 = \\ &= \sum_{y \in U} (y + \beta)^3 + \sum_{y \in A} (y + \beta)^3 + \left(\sum_{y \in V} y + \sum_{y \in B} y \right)^3 = \sum_{y \in U} y^3 + \beta \sum_{y \in U} y^2 + \beta^2 \sum_{y \in U} y + \\ &\beta^3 card(U) + \sum_{y \in A} y^3 + \beta \sum_{y \in A} y^2 + \beta^2 \sum_{y \in A} y + \beta^3 card(A) + \\ &\left(\sum_{y \in V} y \right)^3 + \left(\sum_{y \in V} y \right)^2 \left(\sum_{y \in B} y \right) + \left(\sum_{y \in V} y \right) \left(\sum_{y \in B} y \right)^2 + \left(\sum_{y \in B} y \right)^3. \end{aligned}$$

Dar $\beta = \sum_{y \in U} y = \sum_{y \in V} y$, $\left(\sum_{y \in V} y \right)^2 = \sum_{y \in V} y^2$, iar $card(U), card(V)$ sunt pare. Deci expresia de sus se reduce la

$$\sum_{y \in V} y^3 + \sum_{y \in B} y^3 = \sum_{y \in V \Delta B} y^3.$$

Definiția 15.2 Un cod C este invariant la distanță dacă pentru orice $c_1, c_2 \in C$, are loc relația

$$\forall i \ 0 \leq i \leq n, \quad card(\{\alpha | d(\alpha, c_1) = i\}) = card(\{\alpha | d(\alpha, c_2) = i\}).$$

Exemple simple de coduri invariante la distanță sunt codurile liniare.

Ca o consecință imediată, pentru un cod invariant la distanță care conține cuvântul nul, distanța minimă este ponderea minimă a unui cuvânt - cod nenul.

Vom arăta în continuare că un cod Preparata are această proprietate.

Lema 15.2 $P(r)$ este invariant la distanță.

Demonstrație: Fie $[\chi(U), \chi(V)], [\chi(A), \chi(B)] \in P(r)$ arbitrare aflate la distanța i unul de altul. Conform Lemei 15.1, $[\chi(U \Delta U + \beta), \chi(V \Delta V)], [\chi(U \Delta A + \beta), \chi(V \Delta B)]$ sunt de asemenea cuvinte - cod și se verifică ușor că ele se află la distanța i . Deoarece $U \Delta U = \emptyset$ rezultă $[\chi(U \Delta U + \beta), \chi(V \Delta V)] = \mathbf{0}$, deci $[\chi(U \Delta A + \beta), \chi(V \Delta B)]$ are pondere i . \square

Lema 15.3 Fie $[\chi(U), \chi(V)] \in P(r)$. Atunci $P(r)$ mai conține următoarele cuvinte:

- (i). $[\chi(V), \chi(U)]$;
- (ii). $[\chi(U + \beta), \chi(V + \beta)]$ pentru orice $\beta \in GF(2^r)$;
- (iii). $[\chi(\beta U), \chi(\beta V)]$ pentru orice $\beta \in GF(2^r)$, $\beta \neq \mathbf{0}$.

Demonstrație: Se rezolvă similar demonstrației de la Lema 15.1. □

Exemplul 15.4 Conform Exemplului 17.3, $[\chi(U), \chi(V)] \in P(3)$, unde

$$U = \{\alpha, \alpha^2, \alpha^5, \mathbf{0}\}, V = \{\alpha^0, \alpha, \alpha^2, \alpha^3, \alpha^6, \mathbf{0}\}.$$

Conform Lemei 15.3 în care se ia $\beta = \alpha^3$, vom mai găsi drept cuvinte - cod și pe

$$\begin{aligned} [\chi(V), \chi(U)] &= 11110011 \ 01100101, \\ [\chi(U + \beta), \chi(V + \beta)] &= [\chi(\{\alpha^0, \alpha^5, \alpha^2, \alpha^3\}), \chi(\{\alpha, \alpha^0, \alpha^5, \mathbf{0}, \alpha^4, \alpha^3\})] = \\ &= 10110100 \ 11011101, \\ [\chi(\beta U), \chi(\beta V)] &= [\chi(\{\alpha^4, \alpha^5, \alpha, \mathbf{0}\}), \chi(\{\alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^2, \mathbf{0}\})] = \\ &= 01001101 \ 00111111. \end{aligned}$$

Putem folosi Lema 15.3 pentru a simplifica problema determinării distanței mini-me a lui $P(r)$. Pentru aceasta însă, este nevoie de un rezultat suplimentar (care justifică condiția ca r să fie impar).

Lema 15.4 Fie $\alpha \in GF(2^r)$ primitiv. Atunci α^3 este primitiv dacă r este impar și nu este primitiv dacă r este par.

Demonstrație: Se știe că α^i este primitiv dacă și numai dacă $(i, 2^r - 1) = 1$. Avem $2^r - 1 = (3 - 1)^r - 1 = \mathcal{M}3 + (-1)^r - 1$; deci pentru r impar $2^r - 1 = \mathcal{M}3 - 2 = \mathcal{M}3 + 1$, ceea ce înseamnă că α^3 este primitiv. Similar, pentru r par, $2^r - 1 = \mathcal{M}3$, deci α^3 nu este primitiv. □

Corolarul 15.1 Dacă $r > 2$ este un număr impar, atunci pentru orice $x \in GF(2^r) \setminus \{\mathbf{0}\}$ există y unic (numit rădăcina cubică a lui x) astfel ca $y^3 = x$.

Teorema 15.1 $P(r)$ are distanța minimă $t = 6$.

Demonstrație: Deoarece $P(r)$ este invariant la distanță, el va conține un cuvânt - cod $[\chi(U), \chi(V)]$ de pondere t . Deci

$$t = w(\chi(U)) + w(\chi(V)) = \text{card}(U) + \text{card}(V).$$

Rezultă că t este număr par; mai trebuie verificat că $t \neq 2$, $t \neq 4$ și există un cuvânt - cod de pondere 6.

Să presupunem $t = 2$; atunci $\text{card}(U) = 2$, $\text{card}(V) = 0$ (Lema 15.3 asigură că totul se poate reduce la acest caz). Folosind tot Lema 15.3, (ii), putem presupune că $U = \{\mathbf{0}, \mathbf{x}\}$ cu $\mathbf{x} \neq \mathbf{0}$. Dar atunci $\sum_{u \in U} u = \mathbf{0} + \mathbf{x} = \mathbf{x}$, ceea ce duce la o contradicție a definiției codurilor Preparata, pentru că $V = \emptyset$.

Să presupunem $t = 4$. Cu Lema 15.3, (i) aceasta înseamnă $\text{card}(U) = 4$, $\text{card}(V) = 0$ sau $\text{card}(U) = \text{card}(V) = 2$. În primul caz avem $U = \{\mathbf{0}, \mathbf{x}, \mathbf{y}, \mathbf{z}\}$ cu $\mathbf{x}, \mathbf{y}, \mathbf{z} \neq \mathbf{0}$ și distincte. Ultima condiție a definiției codurilor Preparata este:

$$\mathbf{0}^3 + \mathbf{x}^3 + \mathbf{y}^3 + \mathbf{z}^3 + (\mathbf{0} + \mathbf{x} + \mathbf{y} + \mathbf{z})^3 = \mathbf{0} \text{ sau } (\mathbf{x} + \mathbf{y})(\mathbf{x} + \mathbf{z})(\mathbf{y} + \mathbf{z}) = \mathbf{0}$$

ceea ce este imposibil, cele trei numere fiind distincte și nenule.

În a doua variantă, se poate considera $U = \{\mathbf{0}, \mathbf{x}\}$, $V = \{\mathbf{y}, \mathbf{z}\}$, toate cele patru elemente fiind distincte. Din definiție rezultă

$$\mathbf{0}^3 + \mathbf{x}^3 + (\mathbf{0} + \mathbf{x})^3 = \mathbf{y}^3 + \mathbf{z}^3 \text{ sau } \mathbf{y}^3 + \mathbf{z}^3 = \mathbf{0}.$$

Folosind acum Corolarul 15.1, din $\mathbf{y}^3 = \mathbf{z}^3$ rezultă $\mathbf{y} = \mathbf{z}$, contradicție.

Să construim acum un cuvânt - cod de pondere $t = 6$. Fie $\mathbf{x}, \mathbf{y}, \mathbf{z} \in GF(2^r)$ elemente distincte nenule. Există atunci (Corolarul 15.1) un element $\mathbf{v} \in GF(2^r)$ unic cu $\mathbf{v}^3 = \mathbf{x}^3 + \mathbf{y}^3 + \mathbf{z}^3$. Dacă $\mathbf{v} = \mathbf{x}$, atunci $\mathbf{v}^3 = \mathbf{x}^3$ deci $\mathbf{y}^3 = \mathbf{z}^3$, ceea ce duce la contradicția $\mathbf{y} = \mathbf{z}$. Deci \mathbf{v} este distinct de $\mathbf{x}, \mathbf{y}, \mathbf{z}$.

Definim $\mathbf{u} = \mathbf{v} + \mathbf{x} + \mathbf{y} + \mathbf{z}$. Se observă că $\mathbf{u} \neq \mathbf{0}$ (altfel

$$\mathbf{v}^3 + (\mathbf{x} + \mathbf{y} + \mathbf{z})^3 = (\mathbf{x} + \mathbf{y})(\mathbf{x} + \mathbf{z})(\mathbf{y} + \mathbf{z}) \neq \mathbf{0}$$

deci - cu Corolarul 15.1 - $\mathbf{v} \neq \mathbf{x} + \mathbf{y} + \mathbf{z}$). Fie acum $U = \{\mathbf{0}, \mathbf{u}\}$, $V = \{\mathbf{v}, \mathbf{x}, \mathbf{y}, \mathbf{z}\}$. Din construcție, cum toate elementele sunt distincte, $[\chi(U), \chi(V)]$ este un cuvânt de pondere 6 și se verifică ușor că este cuvânt - cod. \square

Exemplul 15.5 Să considerăm $GF(2^3)$ construit anterior. Fie elementele

$\mathbf{x} = \alpha$, $\mathbf{y} = \alpha^3$, $\mathbf{z} = \alpha^5$. Definim

$$\mathbf{v}^3 = \mathbf{x}^3 + \mathbf{y}^3 + \mathbf{z}^3 = \alpha^3 + \alpha^9 + \alpha^{15} = 110 + 001 + 100 = \alpha^4 = (\alpha^6)^3$$

(deoarece $\alpha^7 = 1$), deci se poate lua $\mathbf{v} = \alpha^6$. Mai departe,

$$\mathbf{u} = \mathbf{v} + \mathbf{x} + \mathbf{y} + \mathbf{z} = \alpha^6 + \alpha + \alpha^3 + \alpha^5 = \alpha^4. \text{ Acum, } U = \{\mathbf{0}, \mathbf{u}\} = \{\mathbf{0}, \alpha^4\},$$

$V = \{\mathbf{v}, \mathbf{x}, \mathbf{y}, \mathbf{z}\} = \{\alpha^6, \alpha, \alpha^3, \alpha^5\}$ și se obține $[\chi(U), \chi(V)] = 00001001 \ 01010110$ care este un cuvânt cod din $P(3)$ de pondere 6.

Teorema 15.2 Codul Preparata nu este cod liniar.

Demonstrație: După cum se știe, $[\chi(U), \chi(V)] + [\chi(A), \chi(B)] = [\chi(U\Delta A), \chi(V\Delta B)]$. Din demonstrația Teoremei 20.5 am văzut cum se pot construi cuvintele - cod $[\chi(U), \chi(V)]$, $[\chi(A), \chi(B)] \in P(r)$ cu $U = \{\mathbf{0}, \mathbf{u}_1\}$, $V = \{\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_1, \mathbf{v}_1\}$,

$A = \{\mathbf{0}, \mathbf{u}_2\}$, $B = \{\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}_2, \mathbf{v}_2\}$. Conform Lemei 15.1,

$\mathbf{c} = [\chi(U\Delta A + \mathbf{u}_1), \chi(V\Delta B)] \in P(r)$. Facem următoarele observații:

$$\text{card}(U\Delta A + \mathbf{u}_1) \leq 2;$$

$$d(\mathbf{c}, [\chi(U\Delta A), \chi(V\Delta B)]) \leq 2 \cdot \text{card}(U\Delta A + \mathbf{u}_1) \leq 4;$$

$$P(r) \text{ are distanța minimă } 6.$$

Din ele rezultă că $[\chi(U), \chi(V)] + [\chi(A), \chi(B)] \notin P(r)$.

Deci $P(r)$ nu este cod liniar. \square

Ca o consecință a acestei teoreme, nu se poate da nici o dimensiune pentru $P(r)$, deci se pare că nu se poate determina numărul de cuvinte - cod. Vom reuși totuși acest lucru ca o consecință a schemei de codificare.

15.2 Codificarea codurilor Preparata extinse

Fie $\alpha \in GF(2^r)$ primitiv și să considerăm codul BCH de polinom generator $g(X) = m_1(X)m_3(X)$ unde $m_i(X)$ este polinomul minimal al lui α^i ($i = 1, 3$). După cum se știe, $\text{grad}(m_1(X)) = \text{grad}(m_3(X)) = r$, deci $\text{grad}(g(X)) = 2r$. Codul BCH astfel definit poate corecta maxim 2 erori și are matricea de control (transpusă):

$$H = \begin{pmatrix} \alpha^0 & \alpha^0 \\ \alpha^1 & \alpha^3 \\ \alpha^2 & \alpha^6 \\ \vdots & \vdots \\ \alpha^{2^r-2} & \alpha^{3(2^r-2)} \end{pmatrix}$$

Deoarece $g(X)$ generează un cod ciclic, rezultă că orice submatrice a lui H formată din $2r$ linii consecutive este nesingulară, deci inversabilă. Se definește matricea A ca submatrice a lui H formată din ultimele $2r$ linii, și H' matricea rămasă prin ștergerea lui A .

Exemplul 15.6 În extensia $GF(2^3)$ generată de rădăcina α a polinomului $1 + X + X^3$ vom avea

$$A = \begin{pmatrix} \alpha & \alpha^3 \\ \alpha^2 & \alpha^6 \\ \alpha^3 & \alpha^2 \\ \alpha^4 & \alpha^5 \\ \alpha^5 & \alpha \\ \alpha^6 & \alpha^4 \end{pmatrix} = \begin{pmatrix} 010 & 110 \\ 001 & 101 \\ 110 & 001 \\ 011 & 111 \\ 111 & 010 \\ 101 & 011 \end{pmatrix} \quad \text{cu} \quad A^{-1} = \begin{pmatrix} 001 & 011 \\ 111 & 010 \\ 011 & 101 \\ 110 & 100 \\ 101 & 110 \\ 111 & 001 \end{pmatrix}.$$

Pentru $GF(2^5)$ generată de rădăcina α a polinomului $1 + X^2 + X^5$ avem

$$A = \begin{pmatrix} \alpha^{21} & \alpha^{63} \\ \alpha^{22} & \alpha^{66} \\ \vdots & \vdots \\ \alpha^{30} & \alpha^{90} \end{pmatrix} = \begin{pmatrix} 00011 & 01000 \\ 10101 & 00001 \\ 11110 & 00101 \\ 01111 & 10001 \\ 10011 & 00111 \\ 11101 & 11011 \\ 11010 & 01100 \\ 01101 & 10101 \\ 10010 & 10011 \\ 01001 & 01101 \end{pmatrix} \quad \text{cu} \quad A^{-1} = \begin{pmatrix} 00111 & 00010 \\ 00011 & 10001 \\ 10011 & 00011 \\ 11011 & 01010 \\ 01101 & 10101 \\ 10101 & 11001 \\ 00110 & 11111 \\ 11001 & 01110 \\ 11000 & 00111 \\ 10001 & 10100 \end{pmatrix}$$

Fie $\mathbf{a} = [\mathbf{a}_L, \mathbf{a}_R]$ un cuvânt binar oarecare de lungime $2^{r+1} - 2r - 2$, unde $|\mathbf{a}_L| = 2^r - 1$, $|\mathbf{a}_R| = 2^r - 2r - 1$. În notație polinomială putem scrie

$$\mathbf{a}_L H = [a_L(\alpha), a_L(\alpha^3)], \quad \mathbf{a}_R H' = [a_R(\alpha), a_R(\alpha^3)].$$

Se mai definește

$$\mathbf{v}_R = [a_L(\alpha) + a_R(\alpha), a_L(\alpha^3) + (a_L(\alpha))^3 + a_R(\alpha^3)] A^{-1}. \quad (2)$$

Teorema 15.3 Fie r un număr impar. Pentru orice cuvânt binar \mathbf{a} de lungime $2^{r+1} - 2r - 2$, dacă $\chi(U) = [\mathbf{a}_L, p_L]$, $\chi(V) = [\mathbf{a}_R, \mathbf{v}_R, p_R]$ unde p_L, p_R sunt biții de control pentru \mathbf{a}_L respectiv $[\mathbf{a}_R, \mathbf{v}_R]$, atunci $[\chi(U), \chi(V)] \in P(r)$.

Demonstrație: $[\mathbf{a}_R, \mathbf{v}_R] H = \mathbf{a}_R H' + \mathbf{v}_R A = [a_R(\alpha), a_R(\alpha^3)] + [a_L(\alpha) + a_R(\alpha), a_L(\alpha^3) + (a_L(\alpha))^3 + a_R(\alpha^3)] = [a_L(\alpha), a_L(\alpha^3) + (a_L(\alpha))^3]$.

Dar $[\mathbf{a}_R, \mathbf{v}_R] = \left[\sum_{v \in V} v, \sum_{v \in V} v^3 \right]$, $a_L(\alpha) = \sum_{u \in U} u$, $a_L(\alpha^3) + (a_L(\alpha))^3 = \sum_{u \in U} u^3 + \left(\sum_{u \in U} u \right)^3$, ceea ce verifică condițiile din Definiția 15.1.

Deci $[\chi(U), \chi(V)]$ este cuvânt - cod în $P(r)$. \square

Corolarul 15.2 $P(r)$ are $2^{2^{r+1}-2r-2}$ cuvinte - cod.

Demonstrație: În Teorema 20.3 există $2^{2^{r+1}-2r-2}$ alegeri posibile pentru \mathbf{a} , toate conducând la cuvinte - cod distincte. \square

Putem da acum un algoritm de codificare a mesajelor de informație în codurile Preparata.

Fie $\mathbf{a}_L, \mathbf{a}_R$ cuvinte de lungimi $2^r - 1$ respectiv $2^r - 2r - 1$.
 Fie \mathbf{v}_R definit de (2).
 Se construiesc p_L, p_R conform Teoremei 20.3.
 Atunci mesajul de informație $\mathbf{a} = [\mathbf{a}_L, \mathbf{a}_R]$ se codifică
 în $[\mathbf{a}_L, \mathbf{p}_L, \mathbf{a}_R, \mathbf{v}_R, \mathbf{p}_R]$.

Exemplul 15.7 Să considerăm $r = 3$, $\mathbf{a}_L = 0110010$, $\mathbf{a}_R = 1$. Deci

$$a_L(\alpha) = \alpha + \alpha^2 + \alpha^5 = \alpha^0, \quad a_R(\alpha) = \alpha^0,$$

$$a_L(\alpha^3) = \alpha^3 + \alpha^6 + \alpha^{15} = \alpha^2, \quad a_R(\alpha^3) = \alpha^0.$$

Din (2) avem $\mathbf{v}_R = [\alpha^0 + \alpha^0, \alpha^2 + \alpha^0 + \alpha^0]A^{-1} = [000 \ 001]A^{-1} = 111001$ unde A^{-1} este matricea din Exemplul 18.5. Atunci vom codifica $\mathbf{a} = [0110010 \ 1]$ în $\mathbf{c} = [\mathbf{a}_L, p_L, \mathbf{a}_R, \mathbf{v}_R, p_R] = [01100010 \ 1 \ 1 \ 111001 \ 1]$.

Deci $\mathbf{c} = [\chi(U), \chi(V)]$ unde $\chi(U) = 01100101$, $\chi(V) = 11110011$, care coincide cu secvența - cod construită în Exemplul 17.3.

15.3 Decodificarea codurilor Preparata extinse

Cum $P(r)$ are distanța minimă 6, el poate corecta maxim 2 erori.

Fie \mathbf{b} un cuvânt recepționat; îl scriem $\mathbf{b} = [\mathbf{b}_L, p_L, \mathbf{b}_R, p_R]$ unde $\mathbf{b}_L, \mathbf{b}_R$ sunt ambele de lungime $2^r - 1$ iar p_L, p_R sunt biți de control a parității. Se calculează

$$\mathbf{b}_L H = [b_L(\alpha), b_L(\alpha^3)], \quad \mathbf{b}_R H = [b_R(\alpha), b_R(\alpha^3)].$$

Apar mai multe cazuri, în funcție de pozițiile unde apar erori.

1. Dacă erorile apar pe pozițiile de control, atunci

$$b_L(\alpha) = b_R(\alpha), \quad b_L(\alpha^3) + (b_L(\alpha))^3 = b_R(\alpha^3)$$

(conform Definiției 15.1), ceea ce se verifică ușor.

2. Dacă există o eroare pe poziția i în \mathbf{b}_R și cel mult o eroare pe pozițiile de control atunci

$$b_L(\alpha) = b_R(\alpha) + \alpha^i, \quad b_L(\alpha^3) + (b_L(\alpha))^3 = b_R(\alpha^3) + \alpha^{3i}, \text{ deci}$$

$$(b_L(\alpha) + b_R(\alpha))^3 = b_L(\alpha^3) + (b_L(\alpha))^3 + b_R(\alpha^3).$$

Dacă ultima relație este verificată, atunci se scrie $\alpha^i = b_L(\alpha) + b_R(\alpha)$ și se schimbă bitul i din \mathbf{b}_R (plus cel mult un bit de control).

3. Dacă există o eroare pe poziția i din \mathbf{b}_L și cel mult o eroare pe pozițiile de control, similar cu cazul anterior (lucru posibil pe baza Lemei 15.3) se verifică relația

$$(b_R(\alpha) + b_L(\alpha))^3 = b_R(\alpha^3) + (b_R(\alpha))^3 + b_L(\alpha^3);$$

în caz afirmativ, se calculează $\alpha^i = b_R(\alpha) + b_L(\alpha)$ și se schimbă bitul i din \mathbf{b}_L (plus cel mult un bit de control).

4. Dacă apar două erori în \mathbf{b}_R pe pozițiile i și j , atunci:

$$b_L(\alpha) = b_R(\alpha) + \alpha^i + \alpha^j, \quad b_L(\alpha^3) + (b_L(\alpha))^3 = b_R(\alpha^3) + \alpha^{3i} + \alpha^{3j},$$

deci se determină $\alpha^i + \alpha^j$ și $\alpha^{3i} + \alpha^{3j}$. Valorile i, j sunt determinate cu un algoritm similar celui de la codurile BCH.

5. Dacă apar două erori în \mathbf{b}_L , invocând din nou Lema 15.3 putem aplica analiza de la cazul anterior.

6. Dacă apar două erori: una pe poziția i în \mathbf{b}_L și una pe poziția j în \mathbf{b}_R , atunci

$$b_L(\alpha) + \alpha^i = b_R(\alpha) + \alpha^j, \quad b_L(\alpha^3) + \alpha^{3i} + (b_L(\alpha) + \alpha^i)^3 = b_R(\alpha^3) + \alpha^{3j}.$$

Acest sistem de necunoscute α^i și α^j se rezolvă astfel: din prima ecuație se scoate $\alpha^j = b_L(\alpha) + \alpha^i + b_R(\alpha)$ și se înlocuiește în a doua ecuație, ceea ce dă

$$b_L(\alpha^3) + \alpha^{3i} + (b_L(\alpha) + \alpha^i)^3 = b_R(\alpha^3) + (b_L(\alpha) + \alpha^i)^3 + (b_L(\alpha) + \alpha^i)^2 b_R(\alpha) + (b_L(\alpha) + \alpha^i) b_R(\alpha)^2 + b_R(\alpha)^3.$$

După simplificări se ajunge la

$$\alpha^{3i} + \alpha^{2i} b_R(\alpha) + \alpha^i (b_R(\alpha))^2 + (b_R(\alpha))^3 = b_L(\alpha^3) + b_R(\alpha^3) + b_L(\alpha)^2 b_R(\alpha) + b_L(\alpha) b_R(\alpha)^2,$$

sau

$$(\alpha^i + b_R(\alpha))^3 = (b_L(\alpha^3) + b_R(\alpha^3)) + (b_L(\alpha) + b_R(\alpha))^3 + b_L(\alpha)^3 + b_R(\alpha)^3.$$

Notând membrul drept al acestei expresii cu Δ , obținem soluția

$$\alpha^i = b_R(\alpha) + \Delta^{1/3}, \quad \alpha^j = b_L(\alpha) + \Delta^{1/3}.$$

Deci locațiile erorilor se pot determina în toate situațiile posibile. Biții de paritate pentru fiecare jumătate de cuvânt dau posibilitatea de a decide ce caz se aplică lui \mathbf{b} . Putem da acum algoritmul de decodificare pentru codurile Preparata $P(r)$.

Fie $\mathbf{b} = [\mathbf{b}_L, p_L, \mathbf{b}_R, p_R]$ cuvântul recepționat.

0. Se calculează $L_1 = b_L(\alpha)$, $L_3 = b_L(\alpha^3)$, $R_1 = b_R(\alpha)$, $R_3 = b_R(\alpha^3)$;
1. Dacă $L_1 + R_1 = 0$ și $L_3 + L_1^3 + R_3 = 0$, atunci singurele erori posibile au apărut pe pozițiile de control.
2. Dacă $(L_1 + R_1)^3 + L_3 + L_1^3 + R_3 = 0$, calculăm $\alpha^i = L_1 + R_1$. Se corectează poziția i din \mathbf{b}_R și cel mult un bit de control al parității; se cere retransmisia dacă ambii biți de paritate trebuie modificați.
3. Dacă $(L_1 + R_1)^3 + R_3 + R_1^3 + L_3 = 0$, calculăm $\alpha^i = L_1 + R_1$. Se corectează poziția i din \mathbf{b}_L și cel mult un bit de control al parității; se cere retransmisia dacă ambii biți de paritate trebuie modificați.
4. Dacă ambele jumătăți ale lui \mathbf{b} sunt de pondere pară și

$$X^2 + (L_1 + R_1)X + \frac{L_3 + L_1^3 + R_3 + (L_1 + R_1)^3}{L_1 + R_1} = (X + \alpha^i)(X + \alpha^j),$$
 se corectează pozițiile i și j din \mathbf{b}_L .
5. Dacă ambele jumătăți ale lui \mathbf{b} sunt de pondere pară și

$$X^2 + (L_1 + R_1)X + \frac{R_3 + R_1^3 + R_3 + (L_1 + R_1)^3}{L_1 + R_1} = (X + \alpha^i)(X + \alpha^j),$$
 se corectează pozițiile i și j din \mathbf{b}_R .
6. Dacă ambele jumătăți ale lui \mathbf{b} sunt de pondere impară, se calculează

$$\alpha^i = R_1 + (L_1^3 + R_1^3 + (L_1 + R_1)^3 + L_3 + R_3)^{1/3}$$

$$\alpha^j = L_1 + (L_1^3 + R_1^3 + (L_1 + R_1)^3 + L_3 + R_3)^{1/3}$$
 Se corectează poziția i din \mathbf{b}_L și poziția j din \mathbf{b}_R .
7. Dacă nu a apărut nici una din situațiile anterioare, se cere retransmisia cuvântului.

Exemplul 15.8 Să decodificăm următoarele cuvinte primite, despre care se presupune că au fost codificate folosind $P(3)$, unde $GF(2^3)$ s-a generat cu rădăcina α a polinomului $1 + X + X^3$:

I: $\mathbf{b} = 10010011 \ 11100111$

$$(0) \ [L_1, L_3] = \mathbf{b}_L H = 111 \ 110, \quad [R_1, R_3] = \mathbf{b}_R H = 101 \ 110$$

$$(1) \ L_1 + R_1 = 111 + 101 = \alpha \neq 0$$

$$(2) \ (L_1 + R_1)^3 + L_3 + L_1^3 + R_3 = \alpha^3 + \alpha^3 + \alpha^{15} + \alpha^3 = \alpha^0 \neq 0$$

$$(3) \ (L_1 + R_1)^3 + R_3 + R_1^3 + L_3 = \alpha^3 + \alpha^3 + \alpha^{18} + \alpha^3 = \alpha^6 \neq 0$$

$$(4) \ X^2 + \alpha X + \frac{\alpha^3 + \alpha^{15} + \alpha^3 + \alpha^3}{\alpha} = X^2 + \alpha X + \alpha^6 = (X + \alpha^2)(X + \alpha^4)$$

Deci \mathbf{b} se decodifică în 10010011 11001111.

II: $\mathbf{b} = 10100100 \ 10001001$

$$(0) \ [L_1, L_3] = \mathbf{b}_L H = [010, 011], \quad [R_1, R_3] = \mathbf{b}_R H = [111, 011]$$

$$(1) \ L_1 + R_1 = 010 + 111 = \alpha^6 \neq 0$$

$$(2) \ (L_1 + R_1)^3 + L_3 + L_1^3 + R_3 = \alpha^{18} + \alpha^4 + \alpha^3 + \alpha^4 = \alpha^6 \neq 0$$

$$(3) \ (L_1 + R_1)^3 + R_3 + R_1^3 + L_3 = \alpha^{18} + \alpha^4 + \alpha^{15} + \alpha^4 = \alpha^2 \neq 0$$

Ambele jumătăți ale lui \mathbf{b} au pondere impară, deci

$$(6) \ \alpha^i = \alpha^5 + (\alpha^3 + \alpha^{15} + \alpha^{18} + \alpha^4 + \alpha^4)^{1/3} = \alpha^5 + (\alpha^5)^{1/3} = \alpha^5 + (\alpha^{12})^{1/3} = \alpha^5 + \alpha^4 = \alpha^0.$$

S-a obținut $i = 0$. Se poate determina apoi imediat

$$\alpha^j = \alpha + \alpha^4 = \alpha^2, \text{ deci } j = 2.$$

Cuvântul \mathbf{b} se decodifică în 00100100 10101001.

III: $\mathbf{b} = 10001000 \ 11101001$

$$(0) \ [l_1, L_3] = \mathbf{b}_L H = [111, 011], \quad [R_1, R_3] = \mathbf{b}_R H = [100, 000]$$

$$(1) \ L_1 + R_1 = 111 + 100 = \alpha^4 \neq 0$$

$$(2) \ (L_1 + R_1)^3 + L_3 + L_1^3 + R_3 = \alpha^{12} + \alpha^4 + \alpha^{15} + \mathbf{0} = \alpha^3 \neq 0$$

$$(3) \ (L_1 + R_1)^3 + R_3 + R_1^3 + L_3 = \alpha^{12} + \mathbf{0} + \alpha^0 + \alpha^4 = \mathbf{0}$$

Calculăm $\alpha^i = L_1 + R_1 = \alpha^4$, deci $i = 4$. Dar modificarea bitului 4 în \mathbf{b}_L cere ca ambii biți de control să fie modificați; deci se cere retransmisia cuvântului.

Pentru codul Preparata obținut prin relaxarea lui $P(r)$, se poate aplica aceeași strategie de decodificare folosită la codul Golay; cum acesta are $d = 5$, poate corecta de asemenea cel mult 2 erori independente.

15.4 Coduri înrudite cu codul Preparata

15.4.1 Codul NHordstrom - Robinson

Să plecăm de la codul Golay binar extins C_{24} ale cărui cuvinte le scriem sub forma $\mathbf{a} = [\mathbf{a}_1, \mathbf{a}_2]$ unde $|\mathbf{a}_1| = 8$, $|\mathbf{a}_2| = 16$. Considerăm submulțimea $GNR \subset C_{24}$ ale cărei elemente verifică condițiile:

1. Dacă $\mathbf{a} \in GNR$ atunci $w(\mathbf{a}_1) \in \{0, 2\}$.
2. Dacă $\mathbf{a}, \mathbf{b} \in GNR$ atunci $d(\mathbf{a}_1, \mathbf{b}_1) \leq 2$.

Evident, GNR va avea 256 cuvinte. Pe baza lui se construiește codul

$$NR = \{\mathbf{a}_2 | \mathbf{a} = [\mathbf{a}_1, \mathbf{a}_2] \in GNR\}.$$

Acesta este un cod neliniar de lungime $n = 16$ și distanță minimă $d = 6$. Numit *codul NHordstrom - Robinson*, el coincide cu codul Preparata $P(3)$.

Un alt cod derivat din acesta este codul *Nadler*, construit astfel: se păstrează din NR doar cuvintele - cod care coincid pe ultimele două poziții. Apoi la toate aceste cuvinte se șterg ultimele trei caractere (o dublă scurtare urmată de o relaxare). Codul astfel obținut este unic, are $n = 13$, $d = 5$ și 64 cuvinte - cod.

15.4.2 Codurile Kerdock

Definiția 15.3 Fie $r \geq 3$, r impar și $U, V \subset GF(2^r)$, $\text{card}(U), \text{card}(V)$ pare.

Un cod Kerdock $K(r)$ este format din toate cuvintele de forma $[U, V]$ care verifică condițiile:

$$1. \ \forall s \ (1 \leq s \leq 2^r - 2) \ (1 \leq w_2(s) < r - 2) \Rightarrow \sum_{u \in U} u^s = \sum_{v \in V} v^s = \mathbf{0};$$

$$2. \ \forall s \ (\leq s \leq 2^r - 2) \ (w_2(s) = r - 2) \Rightarrow \sum_{u \in U} u^s = \sum_{v \in V} v^s = \left(\sum_{u \in U} u^{-1} + \sum_{v \in V} v^{-1} \right)^{-s}$$

unde $w_2(s)$ reprezintă ponderea reprezentării în binar a lui s .

Se consideră prin convenție $\forall s, \mathbf{0}^{-s} = \mathbf{0}$.

Codul Kerdok $K(r)$ (definit în 1972) este de fapt un subcod al codului Reed - Muller $\mathcal{RM}(2, r+1)$; el este format din perechi de translatări de cuvinte din $\mathcal{RM}(1, r)$ selectate în așa fel ca să maximizeze distanța minimă dintre două translatări.

Este interesant că dintre toate codurile cu aceeași distanță minimă, codul Kerdok are numărul maxim de cuvinte - cod.

Nu vom detalia demonstrații referitoare la proprietățile acestor coduri. Pentru informații suplimentare se pot folosi [9] și [4], cu mențiunea că justificările pleacă de la o modalitate diferită de definire a codurilor $K(r)$ și $P(r)$. Rezultatele sunt însă deosebit de interesante. Astfel:

Teorema 15.4 $K(r)$ este invariant la distanță.

Lema 15.5 $K(3) = P(3)$.

Teorema 15.5 Codurile $P(r)$ și $K(r)$ sunt duale.

Ca o consecință imediată, $P(3)$ este un cod auto-dual.

15.5 Exerciții

15.1 Demonstrați relațiile (1).

15.2 Demonstrați Lema 15.3.

15.3 Demonstrați Corolarul 15.1.

15.4 Aplicați Lema 15.3 cuvântului - cod definit în Exemplul 18.7, folosind
 $\beta = \alpha^0, \quad \beta = \alpha, \quad \beta = \alpha^6$

15.5 De ce $[\chi(\beta U), \chi(\beta V)]$ nu este cuvânt - cod pentru $\beta = \mathbf{0}$?

15.6 Arătați că cele trei cuvinte obținute în Exemplul 18.7 satisfac condițiile definiției codurilor Preparata.

15.7 Fie $GF(2^3)$ generat de rădăcina α a polinomului $1 + X + X^3 = 0$. Folosind următoarele cuvinte $\mathbf{x}, \mathbf{y}, \mathbf{z} \in GF(2^3)$ construiți cuvinte - cod de pondere 6 din $P(3)$:

$$\begin{aligned} \mathbf{x} = \alpha \quad \mathbf{y} = \alpha^2, \quad \mathbf{z} = \alpha^3, \\ \mathbf{x} = \alpha \quad \mathbf{y} = \alpha^4, \quad \mathbf{z} = \alpha^6, \\ \mathbf{x} = \alpha^0 \quad \mathbf{y} = \alpha^3, \quad \mathbf{z} = \alpha^6, \end{aligned}$$

15.8 Fie $GF(2^3)$ construit cu $1 + X + X^3$, iar A^{-1} din Exemplul 18.5. Codificați următoarele mesaje folosind $P(3)$:

$$\begin{aligned} \mathbf{a}_L = 1010100, \quad \mathbf{a}_R = 1; \\ \mathbf{a}_L = 1010100, \quad \mathbf{a}_R = 0; \\ \mathbf{a}_L = 1111111, \quad \mathbf{a}_R = 1; \\ \mathbf{a}_L = 1111111, \quad \mathbf{a}_R = 0; \\ \mathbf{a}_L = 0000000, \quad \mathbf{a}_R = 1; \end{aligned}$$

15.9 Fie $GF(2^5)$ construit cu $1 + X^2 + X^5$, iar A^{-1} din Exemplul 18.5. Codificați următoarele mesaje folosind $P(5)$:

$$\mathbf{a}_L = 10100\dots 0, \quad \mathbf{a}_R = 000001000100\dots 0;$$

$$\mathbf{a}_L = 10100\dots 0, \quad \mathbf{a}_R = 00\dots 0;$$

$$\mathbf{a}_L = 10100\dots 0, \quad \mathbf{a}_R = 111100\dots 0;$$

$$\mathbf{a}_L = 0000\dots 0, \quad \mathbf{a}_R = 100\dots 0;$$

Care este lungimea lui \mathbf{a}_L ? dar a lui \mathbf{a}_R ?

15.10 În aceleași condiții din Exemplul 18.2, să se decodifice cuvintele:

10000001 11101000 00011010 01000010 00100101 10100100

01010110 00011110 11101000 10001001 10011001 01010101

01000111 11001000 10101101 11010000 11101110 01010101

10111011 01101010 01011101 11101101 10011100 10100100

01101101 10011000 10101010 10111011 10100101 00010001

15.11 Decodificați cuvintele primite, care au fost codificate cu $P(5)$, unde $GF(2^5)$ a fost construit folosind polinomul $1 + X^2 + X^5$.

1100011000100000000000000000000001000010 00011110000000000000000000000000110010000

101000000010000000000000000000000000 00000100010000000010001010101011100

15.12 Fie \mathbf{b} un cuvânt recepționat în care $w(\mathbf{b}_L)$ este impar iar $w(\mathbf{b}_R)$ este par. Poate fi decodificat \mathbf{b} la pasul 2 al algoritmului într-un cuvânt - cod situat la distanță cel mult 2 ?

Capitolul 16

Coduri convoluționale

În cazul codurilor prezentate până acum, la codificarea unui mesaj de informație se obține un cuvânt - cod de lungime n , ale cărui caractere nu depind decât de cele k elemente de informație curente. Avem de-a face cu un proces de codificare fără memorie, iar aceste coduri sunt numite și *coduri - bloc*.

P. Elias a introdus în 1965 o clasă de coduri, remarcabile prin siguranța sporită pe care o asigură în transmiterea informației: **codurile convoluționale**. În cazul lor, codificarea unui mesaj de lungime k se face ținând cont și de codificările mesajelor de informație anterioare.

De remarcat că NASA și Agenția Spațială Europeană folosesc o combinație între aceste coduri și codurile Reed - Solomon. Fiecare mesaj de informație este codificat inițial cu un cod *RS*, apoi cu un cod convoluțional.

16.1 Coduri liniare și coduri convoluționale

După cum am văzut, un (n, k) - cod liniar codifică un mesaj de informație $\mathbf{u} \in Z_q^k$ într-un cuvânt - cod $\mathbf{v} = \mathbf{u}G$ de lungime n ($\mathbf{v} \in Z_q^n$). Un cod liniar poate fi privit însă și ca o aplicație care transformă mesaje sursă (de orice lungime $i \cdot k$, $i = 1, 2, \dots$) în mesaje - cod de lungime $i \cdot n$.

Exemplul 16.1 *Codul generat de matricea*

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

definește o funcție astfel:

$$a_0a_1a_2a_3a_4a_5 \dots \longmapsto a_0a_1a_2x_0a_3a_4a_5x_1 \dots$$

unde

$$x_0 = a_0 + a_1 + a_2, \quad x_1 = a_3 + a_4 + a_5, \dots$$

O astfel de funcție poate fi exprimată foarte simplu prin polinoame. Astfel, fie $a(X) = a_0 + a_1X + a_2X^2 + \dots$ polinomul reprezentând mesajul sursă, și $u(X) = u_0 + u_1X + u_2X^2 + \dots$ mesajul codificat. Atunci un cod liniar este o aplicație C definită $C(a(X)) = u(X)$. O astfel de aplicație are următoarele proprietăți:

- C este liniară:

$$C(a(X) + a'(X)) = C(a(X)) + C(a'(X));$$

$$C(ta(X)) = tC(a(X)).$$

- C este invariantă în timp: întârzierea mesajului sursă cu k tacti are ca efect decalarea răspunsului cu n tacti:

$$C(X^k a(X)) = X^n C(a(X)).$$

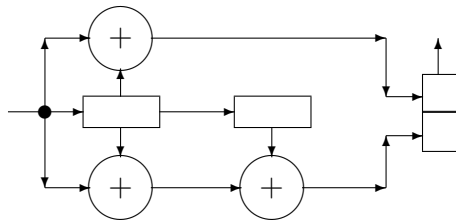
- C nu are memorie: codificarea unui mesaj sursă de lungime k nu depinde de mesajele sursă precedente.

Aceste trei proprietăți caracterizează complet codurile liniare. Dacă se renunță la ultima proprietate, se obține o nouă clasă de coduri, numite *coduri convoluționale*.

Definiția 16.1 Fie q un număr prim. Prin (n, k) - cod convoluțional se înțelege o aplicație liniară $C : Z_q[X] \rightarrow Z_q[X]$ cu proprietatea (de invarianță de timp)

$$C(X^k a(X)) = X^n C(a(X)).$$

Exemplul 16.2 Să considerăm circuitul liniar:



El reprezintă un $(2, 1)$ - cod convoluțional binar. Este liniar (datorită circuitului), invariant în timp (o întârziere de 1 tact la intrare provoacă o întârziere de 2 tacti la ieșire).

De exemplu, pentru intrarea 1 se obține 11 la primul tact, 11 la al doilea tact și 01 la al treilea (după care urmează 0000...).

$$C(1) = 111101 = 1 + X + X^2 + X^3 + X^5.$$

Datorită invarianței în timp, avem

$$C(X) = C(01) = 00111101, \quad C(X^2) = C(001) = 0000111101, \dots$$

(s-a ținut cont de dualitatea de notare polinom - cuvânt).

Datorită liniarității, răspunsul $C(1)$ caracterizează complet codul. De exemplu, mesajul de intrare 101 se codifică prin

$$C(101) = C(1+X^2) = C(1)+C(X^2) = C(1)+C(001) = 11110100\dots+0000111101\dots = 111101101 = 1 + X + X^2 + X^3 + X^4 + X^6 + X^7 + X^9.$$

16.2 Coduri $(n, 1)$ - convoluționale

În cazul particular $k = 1$, codul este complet determinat de ieșirea pentru mesajul de intrare 1. Notăm

$$C(1) = g_0(X)$$

polinomul generator al acestei subclase de coduri. Din invarianța în timp rezultă

$$C(X^i) = X^{ni} g_0(X), \quad \forall i \geq 1.$$

Deci, pentru orice polinom $a(X) = a_0 + a_1X + \dots + a_pX^p$ avem

$$C(a(X)) = a_0C(1) + a_1C(X) + a_2C(X^2) + \dots + a_pC(X^p) = a_0g_0(X) + a_1X^n g_0(X) + a_2X^{2n}g_0(X) + \dots + a_pX^{np}g_0(X) = a(X^n)g_0(X).$$

Aceasta conduce la ideea că un $(n, 1)$ - cod convoluțional este generat de polinomul $g_0(X)$ conform regulii

$$a(X) \mapsto a(X^n)g_0(X).$$

Invers, orice polinom $g_0(X) \in Z_q[X]$ definește un $(n, 1)$ - cod convoluțional.

Observație: Ideea de polinom generator este similară celei de la codurile ciclice, unde codificarea mesajului de informație $a(X)$ revenea la înmulțirea cu polinomul generator. La codurile convoluționale nu există însă restricție referitor la gradul lui $a(X)$ și – în plus – la înmulțire se folosește $a(X^n)$.

Există o modalitate simplă de construcție a unui circuit linear, generator al unui astfel de cod. Vom descrie aceasta pentru cazul binar.

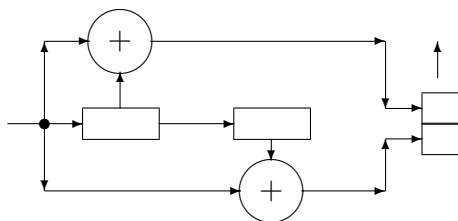
Polinomul $g_0(X)$ se descompune în sume de n termeni consecutivi, fiecare sumă descriind modul de conectare al elementelor de înmagazinare. Astfel, fie

$$g_0(X) = (b_0 + b_1X + \dots + b_{n-1}X^{n-1}) + (b_nX^n + b_{n+1}X^{n+1} + \dots + b_{2n-1}X^{2n-1}) + \dots + (b_{mn}X^{mn} + b_{mn+1}X^{mn+1} + \dots + b_{mn+n-1}X^{mn+n-1})$$

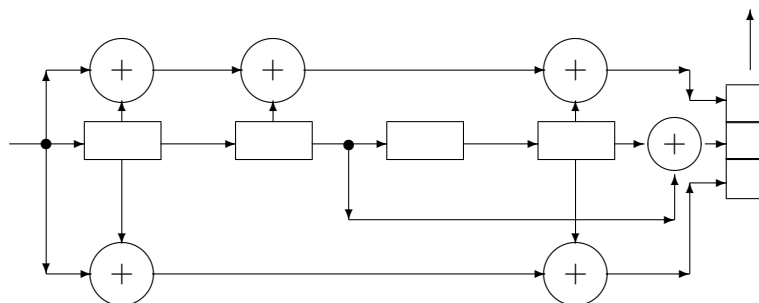
Numărul de $m+1$ paranteze indică faptul că sunt necesare m elemente de înmagazinare (prima paranteză se referă la intrare). A i -a secvență de termeni descrie ieșirea celui de-al i -lea element de înmagazinare ($b_p = 1$ semnifică o legătură directă cu al p -lea element al ieșirii).

În plus, elementele de înmagazinare sunt legate între ele secvențial.

Exemplul 16.3 Să considerăm $(2, 1)$ - codul convoluțional de polinom generator $g_0(X) = 1 + X + X^2 + X^5 = (1 + X) + (X^2 + 0) + (0 + X^5)$. Prima paranteză, $1 + X$, arată că ambii biți de ieșire sunt legați de intrare, $X^2 + 0$ indică faptul că primul element de înmagazinare este legat direct doar de primul bit de ieșire, $0 + X^5$ arată că al doilea element de înmagazinare este legat numai de al doilea bit de ieșire. În acest fel se obține codificatorul descris de circuitul linear:



Exemplul 16.4 Fie $(3, 1)$ - codul convoluțional (deci cu 3 biți de ieșire) generat de polinomul $g_0(X) = 1 + X^2 + X^3 + X^5 + X^6 + X^7 + X^{12} + X^{13} + X^{14}$. Pentru a construi circuitul linear, rescriem acest polinom astfel: $g_0(X) = (1 + 0 + X^2) + (X^3 + 0 + X^5) + (X^6 + X^7 + 0) + (X^{12} + X^{13} + X^{14})$. Se obține



Similar codurilor ciclice, plecând de la polinomul generator, se poate construi matricea generatoare a codurilor $(n, 1)$ - convoluționale. Aceasta este

$$G = \begin{pmatrix} g_0(X) \\ X^n g_0(X) \\ X^{2n} g_0(X) \\ \vdots \end{pmatrix}$$

Numărul liniilor (și deci și al coloanelor) lui G nu este fixat, el depinzând de lungimea mesajului sursă. Astfel, pentru un mesaj de lungime i , matricea G are i linii și $1 + \text{grad}(g_0(X)) + (i - 1) \cdot n$ coloane.

Exemplul 16.5 Fie $(3, 1)$ - codul convoluțional binar de polinom generator $g_0(X) = 1 + X + X^3 + X^4 + X^5 + X^8$. Pentru un mesaj sursă de lungime 3, matricea G va avea dimensiunea 3'times15:

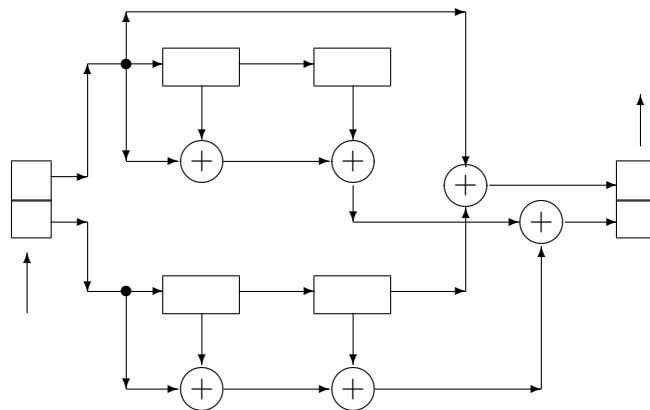
$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & & & & & & & \\ & & & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & & & & \\ & & & & & & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & \end{pmatrix}$$

(spațiile libere sunt completate cu 0). Astfel, codificarea mesajului 101 este $101 \cdot G = 110001000110001$.

16.3 Coduri (n, k) - convoluționale

Să dezvoltăm o descriere a codurilor (n, k) - convoluționale, similară celei date anterior pentru cazul $k = 1$. Vom începe cu un exemplu:

Exemplul 16.6 Fie circuitele liniare a două $(2, 1)$ - coduri convoluționale, care au ieșirea comună:



La fiecare tact intrarea este formată din doi biți care formează intrările în cele două circuite liniare. O întârziere cu doi tacti la intrare generează o întârziere de doi tacti la ieșire; deci, acesta este un $(2, 2)$ - cod convoluțional.

Pentru intrarea 10 ieșirea este 110101 (funcționează doar circuitul superior), iar pentru 01 ieșirea este 010111 (numai pe baza circuitului inferior). Formal,

$$C(1) = 1 + X + X^3 + X^5 = g_0(X), \quad C(X) = X + X^3 + X^4 + X^5 = g_1(X).$$

Folosind invarianța în timp se pot determina și alte ieșiri:

$$C(X^{2i}) = X^{2i} g_0(X), \quad C(X^{2i+1}) = X^{2i} g_1(X) \quad \forall i \geq 1.$$

Definiția 16.2 Pentru un (n, k) - cod convoluțional, polinoamele

$$g_i(X) = C(X^i), \quad 0 \leq i \leq k-1$$

se numesc polinoame generatoare.

Teorema 16.1 Un (n, k) - cod convoluțional de polinoame generatoare $g_i(X)$, $(0 \leq i \leq k-1)$ este determinat de funcția de codificare

$$a(X) \mapsto \sum_{i=0}^{k-1} a^{(i)}(X^n)g_i(X) \quad (1)$$

unde pentru secvența $\mathbf{a} = a_0a_1a_2\dots$ s-a notat $\mathbf{a}^{(i)} = a_i a_{i+k} a_{i+2k} \dots$ $(0 \leq i \leq k-1)$.

Reciproc, fiind date polinoamele $g_i(X)$ $i = 0, 1, \dots, k-1$ și un număr n , formula (1) determină un (n, k) - cod convoluțional.

Demonstrație: Vom începe prin a arăta ultima afirmație. Aplicația

$$a(X) \mapsto a^{(i)}(X^n)g_i(X)$$

este liniară (ca o compunere de transformări liniare) și invariantă în timp (deci definește un (n, k) - cod convoluțional). Ultima aserțiune rezultă din observația că pentru orice secvență \mathbf{a} ,

$$(X^k a(X))^{(i)} \mapsto X^n a^{(i)}(X).$$

Deci ieșirea la $X^k a(X)$ prin această aplicație este $X^n \sum_{i=0}^{k-1} a^{(i)}(X^n)g_i(X)$.

Să arătăm acum că un (n, k) - cod convoluțional definit de aplicația C este identic cu cel dat de $C'(a(X)) = \sum_{i=0}^{k-1} a^{(i)}(X^n)g_i(X)$ unde $g_i(X) = C(X^i)$ sunt polinoamele generatoare.

Deoarece C și C' sunt ambele liniare, este suficient să arătăm că $C(X^r) = C'(X^r)$ pentru orice $r \geq 0$; atunci vom avea

$$C(a_0 + a_1X + a_2X^2 + \dots) = \sum_{r \geq 0} a_r C(X^r) = \sum_{r \geq 0} a_r C'(X^r) = C'(a_0 + a_1X + \dots).$$

Deoarece ambele aplicații sunt (n, k) invariante în timp, ne putem restrânge la $0 \leq r \leq k-1$. Pentru $\mathbf{a} = 00\dots 0100\dots$ (cu 1 pe poziția r), este evident că $\mathbf{a}^{(r)} = 100\dots$ și $\mathbf{a}^{(i)} = 000\dots$ pentru $i \neq r$. Deci $C'(X^r) = g_r(X) = C(X^r)$. \square

Analog codurilor $(n, 1)$ - convoluționale, și în acest caz se poate defini matricea generatoare, care este o reprezentare matricială a relației (1). Liniile matricii sunt $g_0(X), g_1(X), \dots, g_{k-1}(X)$, X considerate ca polinoame de grad infinit. Pentru mesajul $\mathbf{a} = a_0a_1\dots a_r$ codificarea este $\mathbf{a}G = a_0g_0(X) + \dots + a_{k-1}g_{k-1}(X) + X^n[a_kg_0(X) + \dots + a_{2k-1}g_{k-1}(X)] + X^{2n}[a_{2k}g_0(X) + \dots + a_{3k-1}g_{k-1}(X)] + \dots = (a_0 + a_kX^n + a_{2k}X^{2n} + \dots)g_0(X) + \dots + (a_{k-1} + a_{2k-1}X^n + \dots)g_{k-1}(X) = \sum_{i=0}^{k-1} a^{(i)}(X^n)g_i(X)$.

Exemplul 16.7 Matricea generatoare a $(2, 2)$ - codului convoluțional descris în Exemplul 18.6 este

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

(locurile goale sunt completate cu 0).

Pentru intrarea 101 ieșirea este

$$(101) \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} = (11100001), \text{ deci}$$

$$C(1 + X^3) = 1 + X + X^3 + X^8.$$

S-a prezentat anterior modul de construcție al unui circuit liniar pentru $(n, 1)$ - coduri convoluționale, plecând de la polinomul generator $g(X)$. În mod similar se determină circuitul liniar de codificare pentru un (n, k) - cod convoluțional:

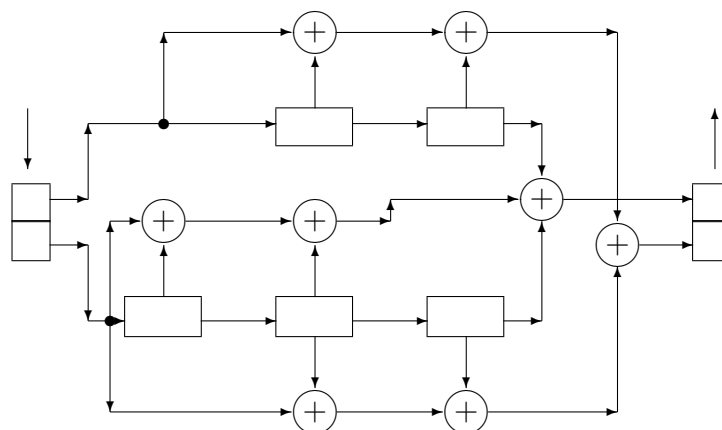
1. Pentru $i = 0, 1, \dots, k - 1$ se construiește circuitul liniar al $(n, 1)$ - codului dat de $a(X) \mapsto a(X^n)g_i(X)$;
2. Se folosește un buffer comun de ieșire (de n simboluri), în care ajung rezultatele însumate ale celor k circuite;
3. Secvența de intrare este segmentată în grupe de câte k simboluri, fiecare din ele constituind intrarea în câte un circuit liniar.

Exemplul 16.8 Să construim circuitul liniar pentru $(2, 2)$ - codul definit de polinoamele generatoare

$$g_0(X) = X + X^3 + X^4 + X^5, \quad g_1(X) = 1 + X + X^2 + X^4 + X^5 + X^6 + X^7.$$

Scriem $g_0(X) = (0 + X) + (0 + X^3) + (X^4 + X^5)$ deci circuitul său liniar are 2 elemente de înmagazinare. Pentru $g_1(X) = (1 + X) + (X^2 + 0) + (X^4 + X^5) + (X^6 + X^7)$ sunt necesare trei elemente de înmagazinare.

Reprezentarea sa grafică este:



Definiția 16.3 $U_n(n, k)$ - cod convoluțional are memorie N dacă

$$\forall i (0 \leq i \leq k-1), \quad \text{grad}(g_i(X)) \leq Nn,$$

iar N este minim cu această proprietate. Numărul Nn se numește lungimea restrânsă a codului.

După cum s-a văzut, un cod convoluțional de memorie N poate fi implementat cu o combinație de $N-1$ elemente de înmagazinare. N este numărul de simboluri de ieșire care pot fi modificate la schimbarea unui singur caracter de intrare.

16.4 Coduri convoluționale sistematice

Un (n, k) - cod convoluțional sistematic are proprietatea că la fiecare ieșire de n caractere, pe primele k poziții se găsesc simbolurile de informație. Vom da o construcție directă a acestor coduri, plecând tot de la similitudinea cu codurile - bloc.

Fie n, k, N ($k < n$) numere naturale nenule. Considerăm sistemul de $k(n-k)$ secvențe cu N componente peste Z_q :

$$g(i, j) = g_0(i, j)g_1(i, j) \dots g_{N-1}(i, j), \quad 1 \leq i \leq k, \quad 1 \leq j \leq n-k.$$

Aceste secvențe se numesc *subgeneratori*.

Considerăm matricea

$$Q_0 = \begin{pmatrix} g_\infty(1) \\ g_\infty(2) \\ \vdots \\ g_\infty(k) \end{pmatrix} = (I_k P_0 O_k P_1 O_k P_2 O_k \dots P_{N-2} O_k P_{N-1} O_k O_k \dots)$$

unde I_k este matricea unitate, O_k este matricea nulă (ambele de ordin k),

$$P_t = \begin{pmatrix} g_t(1, 1) & g_t(1, 2) & \dots & g_t(1, n-k) \\ g_t(2, 1) & g_t(2, 2) & \dots & g_t(2, n-k) \\ \vdots & \vdots & \ddots & \vdots \\ g_t(k, 1) & g_t(k, 2) & \dots & g_t(k, n-k) \end{pmatrix}, \quad 0 \leq t \leq N-1$$

iar

$$g_\infty(i) = 0 \dots 010 \dots 0g_0(i, 1) \dots g_0(i, n-k)0 \dots 0g_1(i, 1) \dots g_1(i, n-k)0 \dots 0 \\ 0 \dots 0g_{N-1}(i, 1) \dots g_{N-1}(i, n-k)0 \dots \quad (1 \leq i \leq k).$$

Cuvintele obținute din primele Nn componente ale lui $g_\infty(i)$ ($1 \leq i \leq k$):

$$g(i) = 0 \dots 010 \dots 0g_0(i, 1) \dots g_0(i, n-k)0 \dots 0g_{N-1}(i, 1) \dots g_{N-1}(i, n-k)$$

se numesc *generatori*.

Fie l un număr natural. Se definește operatorul

$$D^l g_\infty(i) = \underbrace{0 \dots 0}_{ln} g_\infty(i)$$

care translatează $g_\infty(i)$ cu ln poziții spre dreapta.

În sfârșit, fie

$$G_\infty = \begin{pmatrix} Q_0 \\ DQ_0 \\ D^2Q_0 \\ \vdots \end{pmatrix} = \begin{pmatrix} g_\infty(1) \\ \vdots \\ g_\infty(k) \\ Dg_\infty(1) \\ \vdots \\ Dg_\infty(k) \\ D^2g_\infty(1) \\ \vdots \end{pmatrix} = \begin{pmatrix} I_k P_0 & O_k P_1 & O_k P_2 & \dots & O_k P_{N-1} & \dots \\ & I_k P_0 & O_k P_1 & \dots & O_k P_{N-2} & O_k P_{N-1} & \dots \\ & & I_k P_0 & \dots & O_k P_{N-3} & O_k P_{N-2} & O_k P_{N-1} \\ & & & \vdots & & & \end{pmatrix}$$

matricea generatoare a unui (n, k) - cod convoluțional sistematic.

Observație: Evident, un (n, k) - cod convoluțional sistematic este un (n, k) - cod convoluțional. Invers, fiind dat un (n, k) - cod convoluțional, el se poate transforma în unul sistematic prin adunarea la fiecare polinom generator $g_i(X) = \sum_{p=0}^{mn+n-1} b_{i,p} X^i$ a

polinomului $g'_i(X) = \sum_{p=0}^m \sum_{j=0}^{k-1} (k - b_{i,np+j}) X^{np+j} + b_i X^i \quad (0 \leq i \leq k-1)$.

Deși are - teoretic - un număr infinit de linii și coloane, fiind în formă canonică, pentru această reprezentare se poate construi imediat matricea de control:

$$H_\infty = \begin{pmatrix} -P_0^T & I_{n-k} & & & & \dots \\ -P_1^T & O_{n-k} & -P_0^T & I_{n-k} & & \dots \\ -P_2^T & O_{n-k} & -P_1^T & O_{n-k} & -P_0^T & I_{n-k} & \dots \\ & & & \vdots & & & \\ -P_{N-1}^T & O_{n-k} & -P_{N-2}^T & O_{n-k} & -P_{N-3}^T & O_{n-k} & \dots \\ & & -P_{N-1}^T & O_{n-k} & -P_{N-2}^T & O_{n-k} & \dots \\ & & & & -P_{N-1}^T & O_{n-k} & \dots \\ & & & & & \vdots & \end{pmatrix}$$

a cărei transpusă este

$$H_\infty^T = \begin{pmatrix} -P_0 & -P_1 & -P_2 & \dots & -P_{N-1} & \dots \\ I_{n-k} & O_{n-k} & O_{n-k} & \dots & O_{n-k} & \dots \\ & -P_0 & -P_1 & \dots & -P_{N-2} & -P_{N-1} & \dots \\ & I_{n-k} & O_{n-k} & \dots & O_{n-k} & O_{n-k} & \dots \\ & & & \vdots & & & \end{pmatrix}$$

Se verifică imediat egalitatea $G_\infty H_\infty^T = O_\infty$.

Să considerăm un mesaj de informație \mathbf{a} , sub forma unei succesiuni (teoretic) infinite. El va fi descompus inițial în blocuri de lungime k :

$$\mathbf{a} = \underbrace{a_0(1) \dots a_0(k)}_{\mathbf{a}_0} \underbrace{a_1(1) \dots a_1(k)}_{\mathbf{a}_1} \dots \underbrace{a_p(1) \dots a_p(k)}_{\mathbf{a}_p} \dots$$

Mesajul codificat se obține similar codurilor liniare, prin înmulțirea lui \mathbf{a} cu matricea generatoare: $\mathbf{c} = \mathbf{a}G_\infty$; textul rezultat va fi descompus în blocuri de lungime n :

$$\mathbf{c} = \mathbf{a}G_\infty = \underbrace{c_0(1) \dots c_0(n)}_{\mathbf{c}_0} \underbrace{c_1(1) \dots c_1(n)}_{\mathbf{c}_1} \dots \underbrace{c_p(1) \dots c_p(n)}_{\mathbf{c}_p} \dots$$

Ținând cont de forma canonică a matricii G_∞ , rezultă următoarele relații:

$$\begin{aligned} c_p(i) &= a_p(i), \quad (1 \leq i \leq k) \\ c_p(k+j) &= \sum_{i=1}^k \sum_{t=0}^{N-1} a_{p-t}(i) g_t(i, j), \quad (1 \leq j \leq n-k). \end{aligned} \quad (2)$$

Aceeași formulă de codificare se poate scrie și în alt mod:

$$\begin{aligned} \mathbf{c} = \mathbf{a}G_\infty &= \sum_{i=1}^k a_0(i)g_\infty(i) + \sum_{i=1}^k a_1(i)Dg_\infty(i) + \sum_{i=1}^k a_2(i)D^2g_\infty(i) + \dots, \text{ deci} \\ \mathbf{c} &= \sum_{i=1}^k \sum_{t=0}^{\infty} a_t(i)D^t g_\infty(i) \end{aligned} \quad (3)$$

Exemplul 16.9 Să considerăm un $(2, 1)$ - cod convoluțional binar cu $N = 4$ și subgenerator $g(1, 1) = 1101$. Generatorul va fi atunci $g(1) = 110100001$ iar matricea

$$G_\infty = \begin{pmatrix} 11 & 01 & 00 & 01 & & \dots \\ & 11 & 01 & 00 & 01 & \dots \\ & & 11 & 01 & 00 & 01 & \dots \\ & & & & & & \ddots \end{pmatrix}$$

Dacă dorim să se transmită secvența de informație $\mathbf{a} = 10011\dots$, ea va fi codificată în $\mathbf{c} = \mathbf{a}G_\infty = 1101001010\dots$

Exemplul 16.10 Fie $(3, 2)$ - codul convoluțional binar cu $N = 3$ și subgeneratori $g(1, 1) = 101$, $g(2, 1) = 110$. Calculând generatorii, se obține

$$g(1) = 101000001, \quad g(2) = 011001000 \text{ și deci}$$

$$G_\infty = \begin{pmatrix} 101 & 000 & 001 & & \dots \\ 011 & 001 & 000 & & \dots \\ & 101 & 000 & 001 & \dots \\ & 011 & 001 & 000 & \dots \\ & & 101 & 000 & 001 & \dots \\ & & 011 & 001 & 000 & \dots \\ & & & & & & \ddots \end{pmatrix}$$

Fie $\mathbf{a} = 110010\dots$ o secvență de informație. Rezultatul codificării ei este $\mathbf{c} = \mathbf{a}G_\infty = 110001100\dots$

16.5 Arborescența codurilor convoluționale

Cuvintele - cod dintr-un cod convoluțional sistematic se pot reprezenta folosind un graf arborescent infinit, cu noduri situate la o distanță de n poziții și cu q^k descendenți din fiecare nod. Codificarea unui mesaj revine la parcurgerea unui drum în acest arbore.

Pentru simplificare, să considerăm un $(n, 1)$ - cod convoluțional sistematic binar (generalizarea este imediată) de lungime restrânsă N . El este complet caracterizat prin $n - 1$ subgeneratori

$$g(1, j) = g_0(1, j)g_1(1, j) \dots g_{N-1}(1, j), \quad (1 \leq j \leq n - 1).$$

Codul are un singur generator, anume

$$g(1) = 1g_0(1, 1)g_0(1, 2) \dots g_0(1, n - 1)0g_1(1, 1)g_1(1, 2) \dots g_1(1, n - 1)0 \dots \\ \dots 0g_{N-1}(1, 1) \dots g_{N-1}(1, n - 1)$$

Vom folosi notațiile

$$\mathbf{g}_0 = 1g_0(1, 1) \dots g_0(1, n - 1), \\ \mathbf{g}_p = 0g_p(1, 1) \dots g_p(1, n - 1), \quad (1 \leq p \leq N - 1).$$

Atunci $g(1) = \mathbf{g}_0\mathbf{g}_1 \dots \mathbf{g}_{N-1}$, iar \mathbf{g}_p este a p -a ramificație a lui $g(1)$. Matricea generatoare a codului va fi - cu aceste notații:

$$G_\infty = \begin{pmatrix} g_\infty(1) \\ Dg_\infty(1) \\ D^2g_\infty(1) \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{g}_0 & \mathbf{g}_1 & \mathbf{g}_2 & \dots & \mathbf{g}_{N-1} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{g}_0 & \mathbf{g}_1 & \dots & \mathbf{g}_{N-2} & \mathbf{g}_{N-1} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{g}_0 & \dots & \mathbf{g}_{N-3} & \mathbf{g}_{N-2} & \mathbf{g}_{N-1} & \dots \\ & & & & & & & \vdots \end{pmatrix}$$

unde $\mathbf{0}$ este un bloc de n zerouri.

Fie $\mathbf{a} = a_0a_1a_2 \dots$ secvența de informație și $\mathbf{c} = \mathbf{c}_0\mathbf{c}_1\mathbf{c}_2 \dots$ cuvântul - cod corespunzător, unde $\mathbf{c}_p = c_p(1)c_p(2) \dots c_p(n)$. Are loc egalitatea

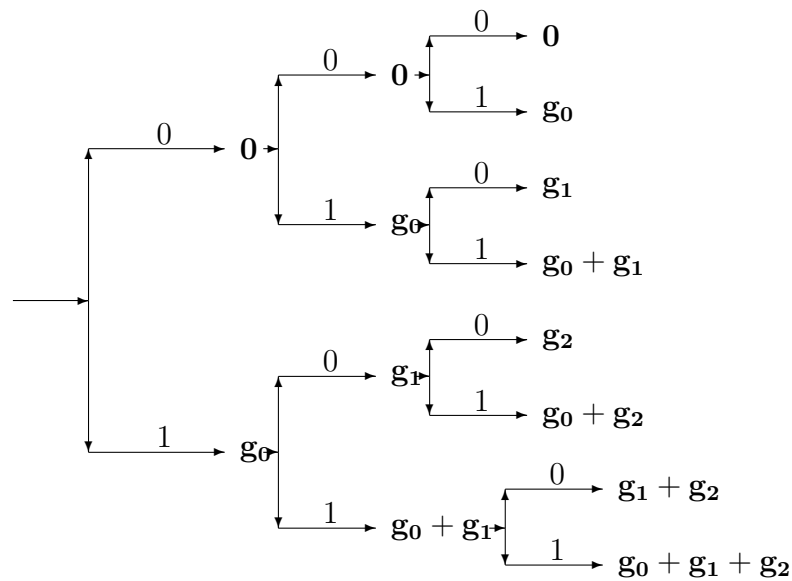
$$\mathbf{c} = \mathbf{c}_0\mathbf{c}_1\mathbf{c}_2 \dots = \mathbf{a}G_\infty = a_0g_\infty(1) + a_1Dg_\infty(1) + a_2D^2g_\infty(1) + \dots \quad (4)$$

De aici rezultă că $\mathbf{c}_0 = \mathbf{0}$ dacă $a_0 = 0$ și $\mathbf{c}_0 = \mathbf{g}_0$ dacă $a_0 = 1$. Cu alte cuvinte, codul poate fi partiționat în două submulțimi de mărime egală: o submulțime S_0 conține toate secvențele care corespund lui $a_0 = 0$ (deci toate cuvintele - cod din S_0 au același prefix $\mathbf{0}$); cealaltă submulțime S_1 conține toate cuvintele - cod corespunzătoare lui $a_0 = 1$ (deci toate cuvintele - cod din S_1 au prefixul \mathbf{g}_0).

La rândul lui, S_0 se împarte în două submulțimi egale S_{00} și S_{01} . S_{00} corespunde secvențelor - cod pentru care $a_0 = a_1 = 0$ (deci toate cuvintele - cod din S_{00} încep cu $\mathbf{00}$). S_{01} conține secvențele cu $a_0 = 0$, $a_1 = 1$, deci cu prefixul $\mathbf{0g}_0$. Similar, S_1 se partiționează în S_{10} și S_{11} , S_{10} fiind pentru $a_0 = 1$, $a_1 = 0$ (deci cuvintele - cod din S_{10} au prefixul $\mathbf{g}_0\mathbf{g}_1$); secvențele din S_{11} încep cu prefixul $\mathbf{g}_0(\mathbf{g}_0 + \mathbf{g}_1)$.

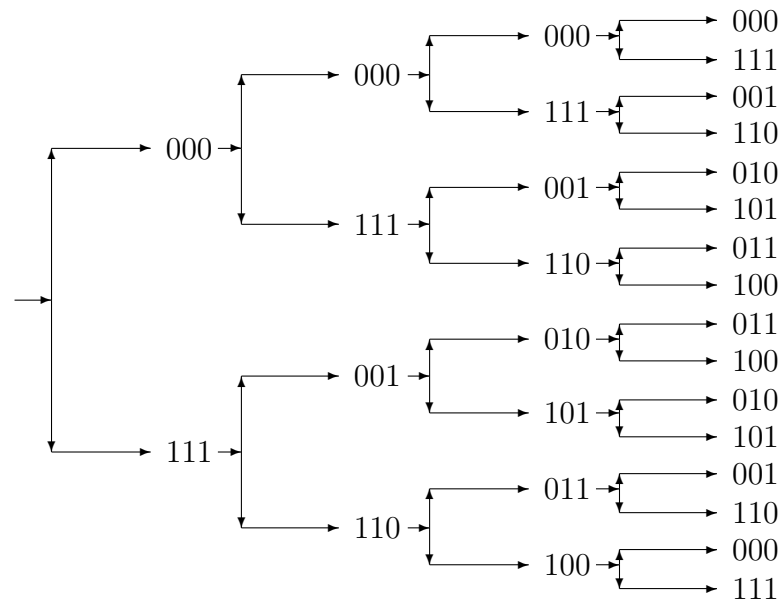
Acest procedeu continuă indefinit.

Deci cuvintele - cod pot fi aranjate într-un arbore cu noduri de n caractere și câte două ramificații din fiecare nod. O ramificație este marcată de un cod bloc de n caractere, corespunzătoare unui caracter de informație particular, iar întreaga secvență corespunde unui drum prin arbore.



Operația de codificare poate fi privită atunci ca un proces în care este trasat un drum particular în arbore, conform instrucțiunilor date de mesajul de informație.

Exemplul 16.11 Să considerăm $(3, 1)$ - codul convoluțional binar cu $N = 4$, generat de $g(1, 1) = 1011$, $g(1, 2) = 1101$. Generatorul codului este $g(1) = 111\ 001\ 010\ 011$, iar arborele de codificare



Dacă vom considera de exemplu mesajul de informație $\mathbf{a} = 1001\dots$, secvența - cod generată va fi $\mathbf{c} = 111\ 001\ 010\ 100\dots$

Această construcție se poate extinde la un (n, k) - cod convoluțional sistematic binar. Deoarece la momentul 0 există 2^k secvențe posibile de lungime k , toate cuvintele - cod se pot partiționa în 2^k submulțimi $S_0, S_1, \dots, S_{2^k-1}$. Toate secvențele din S_i încep cu același bloc, corespunzător aceluiași mesaj de informație. Fiecare S_i se împarte la rândul lui în 2^k submulțimi, după tipul celui de-al doilea bloc de la tactul 1. Procesul continuă în mod similar până la epuizarea mesajului de informație.

16.6 Exerciții

16.1 Construiți un codificator pentru $(2,1)$ - codul convoluțional binar de polinom generator $g_0(X) = 1 + X + X^3 + X^4 + X^6 + X^7 + X^9$. Cât este N ? Codificați mesajele 110, 0110, 1010.

16.2 Construiți o matrice generatoare pentru codul definit anterior.

16.3 Construiți un codificator pentru $(2,1)$ - codul convoluțional ternar de polinom generator $g_0(X) = 1 + 2X + 2X^3$.

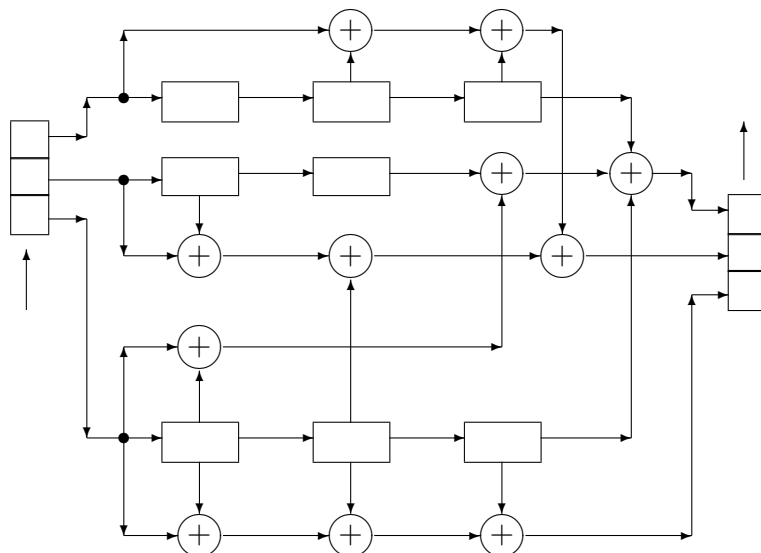
Să se determine o matrice generatoare.

Să se codifice mesajele 102, 200, 0120.

16.4 Construiți un codificator pentru un $(3,2)$ - cod convoluțional binar cu $N = 2$.

16.5 Codificatorul din Exemplul 16.8 are 5 elemente de înmagazinare. Construiți un codificator pentru același cod, care utilizează doar 4 elemente de înmagazinare.

16.6 Construiți polinoamele generatoare ale codului convoluțional dat mai jos:



16.7 Construiți o matrice generatoare pentru codul din exercițiul anterior. Codificați mesajul 11001.

16.8 Construiți un codificator pentru $(4,3)$ - codul binar, de polinoame generatoare $g_0(X) = 1 + X + X^3 + X^5 + X^6$, $g_1(X) = X + X^2 + X^3 + X^4$, $g_2(X) = 1 + X^2 + X^3 + X^5 + X^6$.

16.9 Să se construiască matricea generatoare și de control pentru $(3,2)$ - codul convoluțional sistematic ternar de subgeneratori $g(1,1) = 1200$, $g(2,1) = 2011$.

Să se codifice mesajele 11220, 1012, 2222.

16.10 Aceeași problemă pentru $(4,2)$ - codul binar cu $g(1,1) = 001$, $g(1,2) = 110$, $g(2,1) = 101$, $g(2,2) = 111$.

Să se codifice mesajele 1100, 011, 001100.

Să se construiască reprezentarea arborescentă.

16.11 Să se demonstreze formulele (2).

Capitolul 17

Decodificarea codurilor convoluționale

17.1 Capacitatea de corectare a erorilor

Algoritmii de decodificare pentru codurile convoluționale folosesc aceeași idee de la codurile liniare: cuvântul primit este decodificat în cel mai apropiat cuvânt - cod, în sensul distanței Hamming.

Ideea este de a nu decodifica deodată tot cuvântul primit, ci pas cu pas, la fiecare tact fiind decodificate n caractere (conținutul unui buffer de ieșire) în k simboluri (mărimea bufferului de intrare). Distanța Hamming (notată cu d_H) pentru un (n, k) - cod convoluțional se definește

$$d_H(a(X), b(X)) = d(a_0a_1 \dots a_{n-1}, b_0b_1 \dots b_{n-1}).$$

Observație: În cele ce urmează vom continua să identificăm secvențele de $p+1$ caractere $\mathbf{a} = a_0a_1 \dots a_p$ cu polinoamele de gradul p $a(X) = a_0 + a_1X + \dots + a_pX^p$. Se va folosi adesea chiar notația (neambiguă) $a(X) = \mathbf{a}$.

Fie deci $a(X)$ mesajul de informație; el se codifică în $v(X) = C(a(X))$. Să presupunem că se recepționează $u(X)$. La primul tact se determină cuvântul - cod $w(X)$ cu $d_H(u(X), w(X))$ minim (cel mai apropiat cuvânt - cod), iar $u_0u_1 \dots u_{n-1}$ se transformă în $w_0w_1 \dots w_{n-1}$. Dacă decodificarea este corectă, atunci s-au determinat primele n caractere și se pot găsi imediat caracterele de informație $a_0a_1 \dots a_{k-1}$ (conținutul primei intrări). Se definește apoi

$$u(X) := u(X) - C(a_0 + a_1X + \dots + a_{k-1}X^{k-1})$$

(care aduce 0 pe primele n caractere ale cuvântului recepționat), se ignoră primele n caractere și procedeul continuă inductiv.

Motivația constă în faptul că acum nu se lucrează cu $v(X) = C(a(X))$ ci cu $v(X) - C(a_0 + a_1X + \dots + a_{k-1}X^{k-1}) = C(a(X) - a_0 - a_1X - \dots - a_{k-1}X^{k-1}) = C(a_kX^k + a_{k+1}X^{k+1} + \dots) = C(X^k(a_k + a_{k+1}X + a_{k+2}X^2 + \dots)) = X^kC(a_k + a_{k+1}X + a_{k+2}X^2 + \dots)$.

Exemplul 17.1 Să considerăm $(2, 1)$ - codul convoluțional binar de polinom generator $g_0(X) = 1 + X + X^3$. Cuvintele lui cod sunt

$$\begin{array}{rcl}
& \mathbf{0} & 00000000 \dots \\
& g_0(X) & 11010000 \dots \\
& X^2 g_0(X) & 00110100 \dots \\
(1 + X^2)g_0(X) & & 11100100 \dots \\
\dots & & \dots
\end{array}$$

Dacă se primește - de exemplu - $\mathbf{u} = 11110001$, vom determina cel mai apropiat cuvânt - cod, care este $(1 + X^2 + X^4)g_0(X)$, adică 11101001. Primul grup de $n = 2$ biți este 11, deci primul bit de informație ($k = 1$) este 1. Fie acum

$$\mathbf{u} := \mathbf{u} - C(1) = 11110001 - 11010000 = 00100001.$$

Se ignoră primele două simboluri din noul \mathbf{u} și se determină cel mai apropiat cuvânt - cod de 100001; acesta este 000...0; deci al doilea grup cadru este 00 - adică al doilea simbol de informație este 0.

Refacem $\mathbf{u} := \mathbf{u} - X^2 C(0) = 00100001$. Se șterg primele patru simboluri din \mathbf{u} și se caută cel mai apropiat cuvânt - cod de 0001. Acesta este din nou 00... Deci \mathbf{u} se corectează în

$$C(1 + 0X + 0X^2) = 11000000.$$

Definiția 17.1 Pentru un (n, k) - cod convoluțional se definește distanța liberă prin

$$d_{lib} = \min\{d_H(C(a(X)), C(a'(X))) \mid a_0 a_1 \dots a_{k-1} \neq a_0' a_1' a_{k-1}'\}.$$

Similar observației de la codurile liniare, d_{lib} este cea mai mică pondere a unui cuvânt - cod $C(a'(X))$ cu proprietatea $a_0' a_1' \dots a_{k-1}' \neq \mathbf{0}$. (s-a notat $a'(X) = a(X) - a'(X)$ unde $a(X), a'(X)$ sunt două mesaje de informație arbitrare distincte).

Exemplul 17.2 Reluând codul construit în Exemplul 20.3, acesta are $d_{lib} = 3$, deoarece $g_0(X)$ are ponderea 3 și orice cuvânt $C(a_0 + a_1 X + \dots)$ cu $a_0 \neq 0$ are ponderea minim 3.

Propoziția 17.1 Un cod convoluțional corectează t erori dacă și numai dacă $d_{lib} > 2t$.

Demonstrație: Să presupunem $d_{lib} > 2t$. La recepția unui cuvânt $v(X)$ cu cel mult t erori, există un cuvânt - cod $u(X) = C(a(X))$ aflat la o distanță Hamming minimă de acesta: $d_H(u(X), v(X)) = s$. Dacă $u'(X) = C(a'(X))$ este cuvântul - cod trimis prin canal (recepționat drept $v(X)$), din ipoteză $d_H(v(X), u'(X)) \leq t$. Deci, cu inegalitatea triunghiului, $d(u(X), u'(X)) \leq s + t$. Cum s este cea mai mică distanță Hamming, avem $s \leq t$, deci $d(C(a(X)), C(a'(X))) \leq s + t \leq 2t < d_{lib}$. Din definiția distanței libere rezultă că primele k simboluri din $a(X)$ au drept cel mai apropiat grup de k simboluri generat de cod grupul primelor k simboluri din $a'(X)$, ceea ce permite decodificarea și corectarea erorilor.

Pentru următoarele grupuri, procedeul decurge analog.

Invers, să presupunem $d_{lib} \leq 2t$ și fie $u(X) = C(a(X))$, $u'(X) = C(a'(X))$ cu $d_H(u(X), u'(X)) = d_{lib}$ și $u_0 \dots u_{k-1} \neq u_0' u_{k-1}'$. Fie $i_1, i_2, \dots, i_{d_{lib}}$ toți indicii i pentru care $u_i \neq u_i'$. Construim următorul cuvânt $v(X)$:

$$v_i = \begin{cases} u_i & \text{dacă } u_i = u_i' \text{ sau } i = i_{2s+1} \\ u_i' & \text{dacă } i = i_{2s} \end{cases}$$

Avem $d_H(u(X), v(X)) \leq d_H(u'(X), v(X)) \leq t$. Presupunem că a fost trimis $u'(X)$ și recepționat $v(X)$. Atunci eroarea, care a modificat cel mult t simboluri - poate conduce la o decodificare incorectă, deoarece $u(X)$ este cuvântul cod cel mai apropiat.

Deci, în această ipoteză, codul nu va corecta t erori. \square

Din analiza de sus rezultă că un parametru important pentru un cod convoluțional este distanța liberă, care specifică numărul maxim de erori care pot fi corectate. Din nefericire ([1]), nu se cunoaște nici o metodă analitică de construcție de coduri convoluționale bune, în sensul unei maximizări a d_{lib} pentru n și k date. Folosind calculatorul, au fost găsite însă o serie de astfel de coduri bune. Listăm câteva din ele pentru cazul binar și $k = 1$:

n	d_{lib}	$g_0(X)$
2	5	$110111 = 1 + X + X^3 + X^4 + X^5$
2	6	11110111
2	7	1101011011
2	8	110111011011
2	10	11011111001011
2	10	111011110111
3	8	111011111
3	10	111011101111
3	12	111011101011111
4	10	111101111111
4	13	1111011110011111

17.2 Decodifocarea codurilor sistematice

Păstrând similitudinea cu codurile liniare, să folosim la codurile sistematice ideea de decodificare bazată pe calculul sindromului.

Fie un (n, k) - cod convoluțional sistematic definit prin matricea generatoare G_∞ și cea de control H_∞ , construite pe baza subgeneratorilor de lungime N $g(i, j)$, $1 \leq i \leq k$, $1 \leq j \leq n - k$ (a se vedea prelegerea anterioară). Din relațiile $G_\infty H_\infty^T = \mathbf{0}$ și $\mathbf{c} = \mathbf{a}G_\infty$ se obține $\mathbf{c}H_\infty^T = \mathbf{0}$.

Să presupunem că s-a recepționat secvența (infinită) $\mathbf{r} = \mathbf{c} + \mathbf{e}$ unde \mathbf{e} este o secvență eroare. Ca și la codurile liniare, sindromul va fi

$$\mathbf{s} = \mathbf{r}H_\infty^T = \mathbf{c}H_\infty^T + \mathbf{e}H_\infty^T = \mathbf{e}H_\infty^T.$$

Secvența recepționată se poate structura în blocuri de n caractere:

$$\mathbf{r} = \underbrace{r_0(1) \dots r_0(n)}_{\mathbf{r}_0} \underbrace{r_1(1) \dots r_1(n)}_{\mathbf{r}_1} \dots \underbrace{r_p(1) \dots r_p(n)}_{\mathbf{r}_p} \dots$$

Similar se segmentează sindromul în blocuri de lungime $n - k$:

$$\mathbf{s} = \underbrace{s_0(1) \dots s_0(n - k)}_{\mathbf{s}_0} \underbrace{s_1(1) \dots s_1(n - k)}_{\mathbf{s}_1} \dots \underbrace{s_p(1) \dots s_p(n - k)}_{\mathbf{s}_p} \dots$$

Vom avea

$$s_p(j) = r_p(k + j) - \sum_{i=1}^k r_p(i)g_0(i, j) - \sum_{i=1}^k r_{p-1}(i)g_1(i, j) - \dots - \sum_{i=1}^k r_{p-N+1}(i)g_{N-1}(i, j),$$

$$1 \leq j \leq n - k.$$

Această relație se poate scrie și $s_p(j) = r_p(k + j) - A_p(j)$, unde

$A_p(j) = \sum_{t=0}^{N-1} \sum_{i=1}^k r_{p-t}(i)g_t(i, j)$ este rezultatul codificării secvenței $r_p(1) \dots r_p(k)$ folosind matricea G_∞ , în funcție de $r_{p-1}(1), \dots, r_{p-1}(k), \dots, r_{p-N+1}(1), \dots, r_{p-N+1}(k)$.

Din aceste relații, se poate deduce o formulă pentru calculul elementelor sindromului, în funcție de caracterele secvenței - eroare:

$$s_p(j) = e_p(k+j) - \sum_{i=1}^k e_p(i)g_0(i,j) - \sum_{i=1}^k e_{p-1}(i)g_1(i,j) - \dots - \sum_{i=1}^k e_{p-N+1}(i)g_{N-1}(i,j),$$

$$1 \leq j \leq n-k,$$

$$\text{unde } \mathbf{e} = \underbrace{e_0(1) \dots e_0(n)}_{\mathbf{e}_0} \underbrace{e_1(1) \dots e_1(n)}_{\mathbf{e}_1} \dots \underbrace{e_p(1) \dots e_p(n)}_{\mathbf{e}_p} \dots$$

este secvența de eroare - tip, segmentată în blocuri de lungime n . Deci, pentru orice $j = 1, 2, \dots, n-k$, putem scrie:

$$s_0(j) = e_0(k+j) - \sum_{i=1}^k e_0(i)g_0(i,j),$$

$$s_1(j) = e_1(k+j) - \sum_{i=1}^k e_1(i)g_0(i,j) - \sum_{i=1}^k e_0(i)g_1(i,j),$$

$$\vdots$$

$$s_{N-1}(j) = e_{N-1}(k+j) - \sum_{p=0}^{N-1} \sum_{i=1}^k e_p(i)g_{N-p-1}(i,j),$$

$$s_N(j) = e_N(j) - \sum_{p=0}^{N-1} \sum_{i=1}^k e_{p+1}(i)g_{N-p-1}(i,j),$$

$$\vdots$$

$$s_{N+m}(j) = e_{N+m}(k+j) - \sum_{p=1}^{N-1} \sum_{i=1}^k e_{p+m-1}(i)g_{N-p-1}(i,j)$$

$$\vdots$$

De remarcat că secvența \mathbf{e}_0 afectează numai primele N componente ale sindromului: $\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_{N-1}$, deoarece apare numai în primele $N(n-k)$ ecuații din sistemul de sus.

Pentru corectarea erorilor, se determină inițial secvența \mathbf{e}_0 ; după aceasta, sindromul se recalculază scăzând din el componentele lui \mathbf{e}_0 . Această operație se numește *rearanjarea sindromului*. În paralel, \mathbf{e}_0 se folosește la corectarea primelor n caractere recepționate; operația se realizează identic cu cea de la codurile liniare.

În continuare, se va folosi un nou sindrom, anume:

$$s_0'(j) = 0$$

$$s_m'(j) = e_m(k+j) - \sum_{p=0}^{m-1} \sum_{i=1}^k e_{m-p}(i)g_p(i,j), \quad 1 \leq m \leq N-1, \quad 1 \leq j \leq n-k$$

$$s_{N+m}'(j) = s_{N+m}(j), \quad \forall m \geq 0.$$

De remarcat că secvența \mathbf{e}_1 figurează numai în expresiile lui $\mathbf{s}_1'', \mathbf{s}_2'', \dots, \mathbf{s}_N''$ unde $\mathbf{s}_i'' = s_i''(1) \dots s_i''(n-k)$, $1 \leq i \leq N$. Deci aceste componente apar în $N(n-k)$ ecuații. După determinarea lui \mathbf{e}_1 se corectează blocul \mathbf{r}_1 din secvența recepționată, apoi se recalculază sindromul, ș.a.m.d.

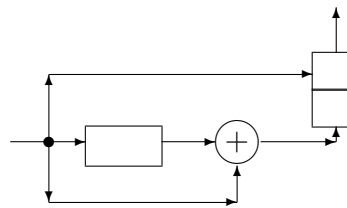
17.3 Algoritmul Viterbi

Cel mai cunoscut algoritm de decodificare pentru codurile convoluționale aparține lui Viterbi și a fost folosit pe scară largă în comunicațiile spațiale. Astfel - ca să dăm numai un exemplu, stația Voyager în misiunea sa din 1974 spre Marte, Saturn și Jupiter, a folosit pentru transmisii un $(2, 1)$ - cod convoluțional binar de polinom generator $g_0(X) = 1 + X + X^2 + X^5 + X^6 + X^7 + X^8 + X^{10} + X^{11} + X^{12}$.

În descrierea algoritmului Viterbi vom folosi o reprezentare în *rețea* a codurilor convoluționale, care reia sub o formă finită reprezentarea arborescentă. Din nou, ne vom mărgini la construcții pentru $(n, 1)$ - coduri convoluționale binare, extensia la cazul general fiind imediată.

O *diagramă rețea* este un graf orientat infinit, care are ca noduri stările unui circuit liniar la fiecare moment (tact). Un nod marcat care corespunde unei stări S la momentul i , este legat direct cu alte două noduri, corespunzătoare stărilor circuitului la momentul $i+1$: S_0 (pentru intrarea 0) respectiv starea S_1 (pentru intrarea 1). Grafic, arcul $S \rightarrow S_0$ va fi totdeauna trasat sub arcul $S \rightarrow S_1$. Fiecare arc este marcat cu secvența de ieșire a circuitului pentru intrarea corespunzătoare.

Exemplul 17.3 $(2, 1)$ - codul convoluțional definit de circuitul liniar

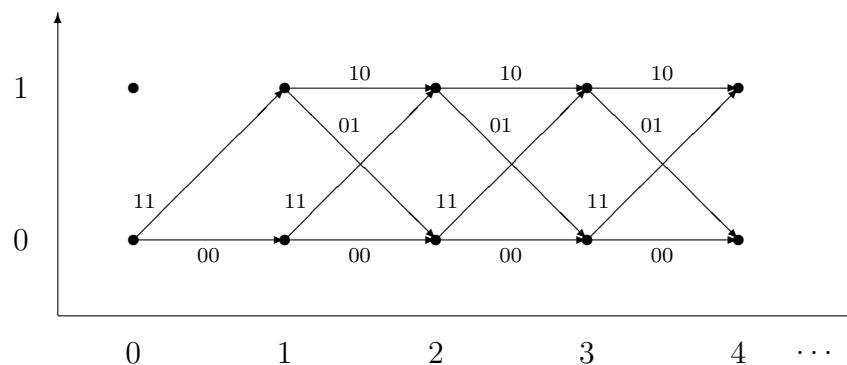


are două stări posibile (după conținutul elementului de înmagazinare): 0 și 1.

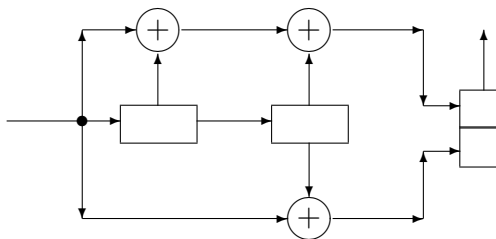
Pentru intrarea i ($i = 0, 1$) ieșirea este secvența

i	dacă starea este 0,
$i(1 - i)$	dacă starea este 1.

Diagrama rețea a codului este:



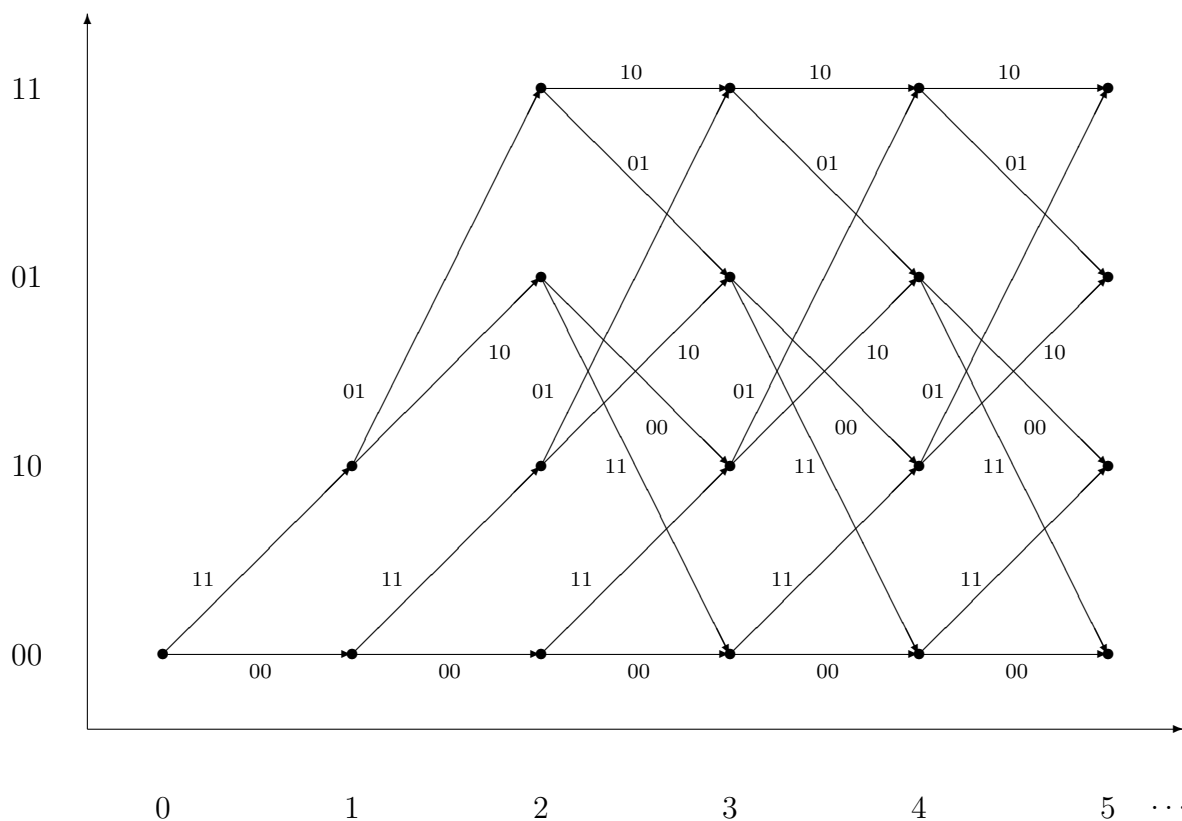
Exemplul 17.4 $(2, 1)$ - codul convoluțional (de memorie $N = 2$) reprezentat prin circuitul liniar



are patru stări: 00, 01, 10 și 11 (după conținuturile elementelor de înmagazinare).

Din starea $S = 00$ se ajunge la $S_0 = 00$ dacă intrarea este 0, sau în $S_1 = 10$ dacă intrarea este 1. Ieșirile corespunzătoare sunt 00 respectiv 11.

Analog, dacă $S = 10$, atunci $S_0 = 01$ (intrare 0 ieșire 10) și $S_1 = 11$ (intrare 1 ieșire 01), etc. Se ajunge la diagrama rețea următoare:



Orice drum prin rețea, care pleacă din momentul 0 conduce - prin citirea marcajelor arcelor - la un cuvânt - cod. De exemplu, pentru intrarea 100110, codul generat este $C(100110) = 111000011001$ (reamintim, spre dreapta pleacă totdeauna două arce, iar arcul pentru intrarea 0 este situat sub arcul pentru intrarea 1).

Invers, orice cuvânt - cod reprezintă marcajul unui drum în rețea.

Să construim acum algoritmul de decodificare Viterbi.

La recepția unui cuvânt $\mathbf{v} = \mathbf{v}_0\mathbf{v}_1\mathbf{v}_2 \dots$ vom căuta construcția unui drum în rețea, cât mai apropiat de \mathbf{v} .

Pentru fiecare moment i și stare S vom lista drumurile *active* prin rețea până la starea S la momentul i . Un drum este *activ* dacă *discrepanța* sa este minimă (prin *discrepanță* se înțelege distanța Hamming dintre cuvântul generat de drumul prin arbore și cuvântul de aceeași lungime primit la intrare). Vom adnota fiecare stare la momentul i cu discrepanța drumului său activ. Pentru calculul ei se poate folosi o formulă recursivă definită astfel:

Fie $d(S \rightarrow S'')$ distanța Hamming dintre marcajul arcului $S \rightarrow S''$ și secvența de n caractere primită, iar $di(S)$ discrepanța stării S . Atunci

$$di(S) = \min_{S \rightarrow S''} \{di(S) + d(S \rightarrow S'')\} \quad (1)$$

Algoritmul va avea o durată de funcționare finită numită *fereastră de decodificare*, pe care o notăm cu b (deci va funcționa b tacti).

Algoritmul Viterbi:

Intrare: Secvența $\mathbf{v}_0\mathbf{v}_1 \dots$, $\mathbf{v}_p = v_{p,0}v_{p,1} \dots v_{p,n-1}$;

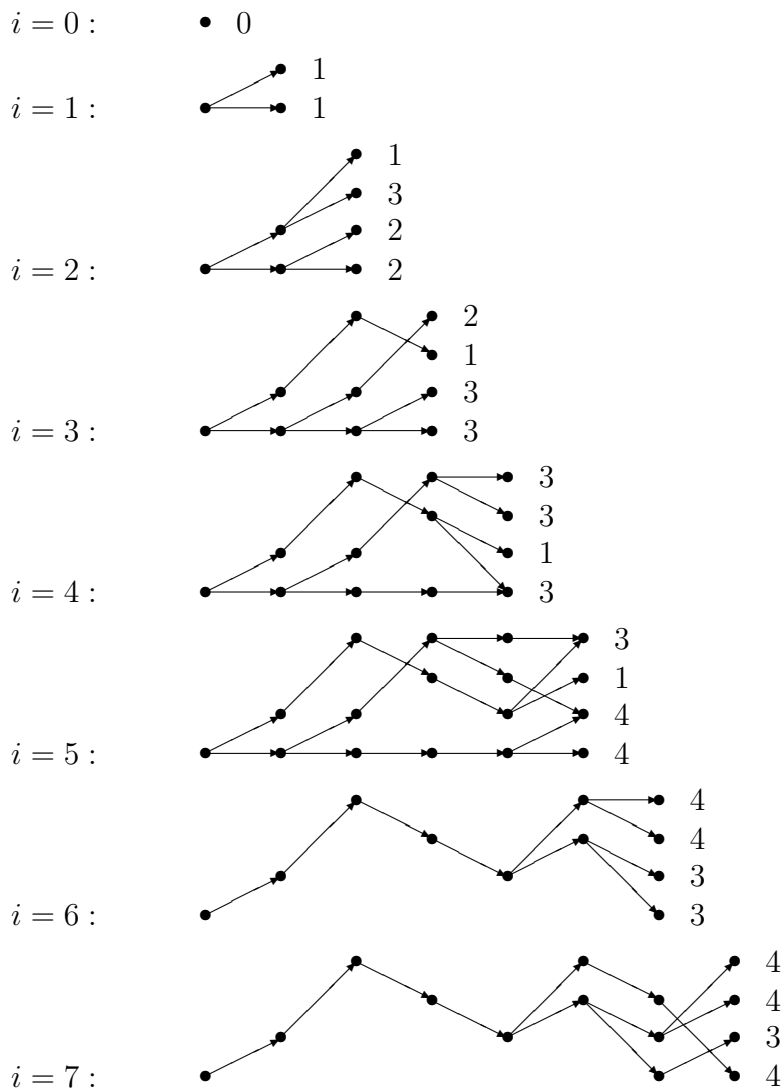
Algoritm:

1. Fie $p := 0$ și S_0 starea inițială a rețelei (starea cu toate elementele de înmagazinare 0);
2. Se marchează $di(S_0) = 0$;
3. **for** $i := 1$ **to** b **do**
 Pentru toate stările S accesibile la momentul $p+i$ se determină $di(S)$ folosind formula (1), și se listează toate drumurile active care duc la S .
4. Dacă toate drumurile active la momentul $p + b$ încep cu același arc $S_0 \rightarrow S_1$, secvența de intrare \mathbf{v}_p se corectează în marcajul \mathbf{u}_p al acestui prim arc (și se decodifică în primul caracter al lui S_1). Altfel, eroarea nu este corectabilă, STOP.
5. $p := p + 1$, $S_0 := S_1$, salt la pasul 3.

Exemplul 17.5 Să reluăm codul descris în Exemplul 18.7, împreună cu diagrama sa de rețea. Presupunem că s-a primit secvența $\mathbf{v} = 10010100101100000 \dots$. Deci

$\mathbf{v}_0 = 10$, $\mathbf{v}_1 = 01$, $\mathbf{v}_2 = 01$, $\mathbf{v}_3 = 00$, $\mathbf{v}_4 = 10$, $\mathbf{v}_5 = 11$, $\mathbf{v}_6 = 00 \dots$

Vom lucra cu o fereastră de decodificare $b = 7$. Detaliem grafic numai procedura de decodificare a primului bloc (cazul $p = 0$).



Se ajunge la concluzia că toate arcele active au același început: arcu marcat cu 11. Deci primul bloc $\mathbf{v}_0 = 10$ se decodifică în 11, după care se ia ca nod inițial $S_0 = 10$ și procesul se repetă.

De remarcat că pentru o fereastră de decodificare $b = 5$, eroarea nu s-ar fi putut corecta (primul arc nu este unic).

În această prezentare a algoritmului Viterbi, apare o problemă: cât de mare este fereastra b ? Cât de departe mergem prin diagrama rețea până să fim siguri că totuși este posibilă decodificarea și corectarea erorilor?

Pentru aceasta să refacem sub o formă finită reprezentarea diagramelor - rețea.

Din definiția dată în Prelegerea 16 se observă că dacă $g_0(X), g_1(X), \dots, g_{n-1}(X)$ sunt polinoamele generatoare și $a(X)$ un mesaj de informație, codificarea poate fi scrisă ca o secvență de n componente (infinite)

$$c(X) = (a(X)g_0(X), a(X)g_1(X), \dots, a(X)g_{n-1}(X))$$

ale căror caractere se transmit în paralel, în ordinea crescătoare a puterilor lui X .

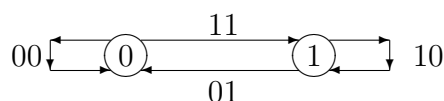
Exemplul 17.6 Fie $(2,1)$ -codul generat de $g_0(X) = 1 + X + X^3$ și $g_1(X) = 1 + X$. Mesajul $a(X) = 1 + X^2$ este codificat în

$c(X) = ((1+X^2)(1+X+X^3), (1+X^2)(1+X)) = (1+X+X^2+X^5, 1+X+X^2+X^3)$.
Deci cuvântul cod este $\mathbf{c} = 11\ 11\ 11\ 01\ 00\ 10\ 00\ \dots$

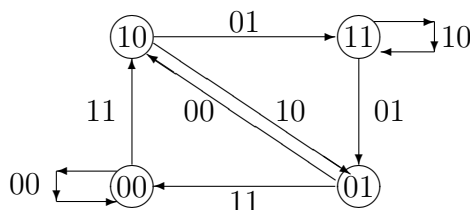
Vom construi un translator (*diagramă de translatare*) asociat unui $(n, 1)$ - cod convoluțional astfel:

Stările translatorului sunt N - tupluri binare, fiecare stare reprezentând o situație posibilă a celor N elemente de înmagazinare. O stare $s_1 \dots s_{N-1}s_N$ este legată direct prin arc marcat $x_1 \dots x_n$ de starea $is_1 \dots s_{N-1}$, ($i = 0, 1$) dacă în circuitul liniar corespunzător codului, intrarea i conduce la modificarea conținuturilor elementelor de înmagazinare, din (s_1, \dots, s_N) în (i, s_1, \dots, s_{N-1}) și are ca efect ieșirea $x_1 \dots x_n$.

Exemplul 17.7 *Rețeaua codului definit în Exemplul 17.3 poate fi reprezentată sub formă de diagramă de translatare astfel:*



Pentru codul din Exemplul 18.7, reprezentarea finită este:



De remarcat că pe fiecare arc este suficient să marcăm doar secvența de ieșire; caracterul de intrare este reprezentat de primul caracter al stării în care intră arcul respectiv.

Definiția 17.2 *Se definește lungimea unui drum ca fiind numărul de arce care formează acel drum.*

Ponderea unui drum este suma ponderilor arcelor care formează drumul.

Reamintim, *ponderea unui cuvânt* este numărul de caractere nenule din cuvântul respectiv.

Conform Definiției 17.1, distanța liberă a unui cod este ponderea nenulă minimă a unui cuvânt cod - deci a unei secvențe care marchează un drum prin diagrama de translatare.

Propoziția 17.2 *Distanța liberă a unui $(n, 1)$ - cod convoluțional este egală cu ponderea nenulă minimă a unui ciclu care trece prin starea inițială $00 \dots 0$ a diagramei de translatare a codului.*

Demonstrație: Este imediată.

Fie $S_0 = 00 \dots 0$ starea inițială. Pentru orice t , $1 \leq t \leq \left\lceil \frac{d_{lib} - 1}{2} \right\rceil$, se definește $d(t)$ ca fiind cel mai mic număr întreg pozitiv p cu proprietatea că orice drum $S_0 \rightarrow S_1 \rightarrow \dots \rightarrow S_{p-1}$, ($S_0 \neq S_1$) are ponderea mai mare de $2t$. Pentru determinarea algoritmică a lui $d(t)$ se poate adapta imediat un algoritm similar din teoria grafurilor.

Deoarece codul este liniar, valoarea lui $d(t)$ se păstrează pentru orice stare a diagramei de translatare.

Acum, în condițiile că pot apare maxim t erori, algoritmul Viterbi va funcționa corect pentru o fereastră $b = d(t)$ (datorită Propoziției 17.1).

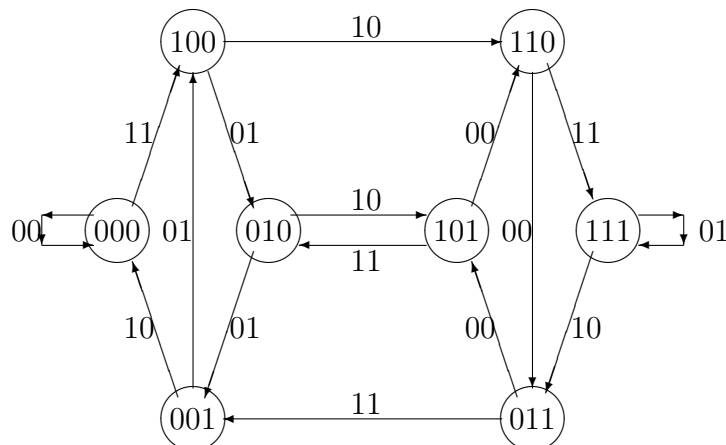
Reluarea algoritmului Viterbi la fiecare pas cu fereastra maximă $d(t)$ este însă destul de costisitoare ca timp; de aceea, practic, se folosește o fereastră mobilă $b \leq d(t)$ și algoritmul trece la faza de decodificare atunci când toate drumurile active selectate au același prim arc. Dacă s-a ajuns la $b = t(b)$ și nu s-a selectat un prim arc comun, algoritmul eșuează.

17.4 Coduri convoluționale catastrofice

Dacă este definit neglijent, un cod convoluțional poate conduce la următoarea situație, complet nefericită: un mesaj în care a fost perturbat un număr mic de caractere generează prin decodificare o infinitate de erori.

Pentru a studia acest caz, considerăm din nou numai codurile $(n, 1)$ - convoluționale binare. Pentru a vedea în ce constă problema, să luăm următorul exemplu:

Exemplul 17.8 Fie $(2, 1)$ - codul convoluțional binar, cu polinoamele generatoare $g_0(X) = 1 + X^3$, $g_1(X) = 1 + X + X^2$. Diagrama sa de translatare este



Să presupunem că s-a transmis mesajul de informație $\mathbf{a} = 00\dots$, căruia îi corespunde cuvântul - cod nul $\mathbf{c} = 0000\dots$. La recepție s-a primit un cuvânt cu primele trei simboluri greșite: $\mathbf{v} = 111000\dots$. Decodificarea sa nu ridică nici o problemă, acesta fiind de asemenea cuvânt - cod, care marchează drumul prin stările $000 - 100 - 110 - 011 - 101 - 110 - \dots$

Deci, în ipoteza generală din Teoria Codurilor (cea a decodificării cele mai probabile), nu au apărut erori, și mesajul decodificat este $\mathbf{a}'' = 110110110\dots$, care diferă de \mathbf{a} într-o infinitate de poziții.

În acest mod, modificarea doar a primelor trei caractere a dus la situația în care aceste erori se propagă într-o secvență infinită.

Se observă din acest exemplu că tot necazul provine din faptul că diagrama conține două cicluri distincte cu ieșirea 0: $000 - 000$ și $110 - 011 - 101 - 110$.

Definiția 17.3 Un cod convoluțional se numește catastrofic dacă diagrama sa de translatare conține două cicluri distincte de pondere zero.

Observație: Orice cod convoluțional conține un ciclu de pondere zero, anume bucla care pleacă și intră în starea inițială. Reamintim, ponderea unui drum este suma ponderilor arcelor sale.

Teorema 17.1 *Un $(n, 1)$ - cod convoluțional este catastrofic dacă și numai dacă $\text{cmmdc}(g_0(X), \dots, g_{n-1}) = 1$.*

Demonstrație: Teorema rămâne valabilă dacă o demonstrăm pentru cazul $n = 2$. Fie deci un $(2, 1)$ - cod convoluțional, de polinoame generatoare $g_0(X), g_1(X)$. Pentru a marca un ciclu, un cuvânt de intrare trebuie să fie de forma $b(X) + X^r a(X) \cdot (1 + X^p + X^{2p} + \dots)$, cu $\text{grad}(a(X)) < p$. Pentru a elimina din discuție bucla din starea S_0 , vom considera $a(X) \neq 0$. Să presupunem că în diagrama de translatore a codului există un ciclu de lungime p și pondere 0. Atunci $a(X) \cdot g_0(X)$ și $a(X) \cdot g_1(X)$ trebuie să se dividă cu $1 + X^p$, deci $a(X) \cdot g'(X)$ (unde $g'(X) = \text{cmmdc}(g_0(X), g_1(X))$), se divide cu $1 + X^p$.

Dacă $g'(X) = 1$, se ajunge la o contradicție (deoarece $a(X)$ are grad mai mic decât p și nu este polinomul identic nul). Deci $\text{grad}(g'(X)) \geq 1$. În acest caz, toate mesajele de informație $b(X) + a(X) [g'(X)]^t (1 + X^p + X^{2p} + \dots)$, ($t \geq 0$) se codifică identic, ceea ce va conduce la imposibilitatea decodificării.

Reciproca se demonstrează similar. □

17.5 Exerciții

17.1 *Determinați d_{ib} pentru fiecare din codurile următoare:*

- (a) $g_0(X) = 1 + X^2, \quad g_1(X) = 1 + X + X^2;$
- (b) $g_0(X) = 1 + X + X^2 + X^3, \quad g_1(X) = 1 + X^2 + X^3;$
- (c) $g_0(X) = 1 + X^3 + X^4, \quad g_1(X) = 1 + X + X^2 + X^4.$

17.2 *Folosind $(2, 2)$ - codul convoluțional definit în Exemplul 18.7, și fereastra de decodificare $b = 7$, decodificați (cât este posibil) secvențele*

$$\mathbf{v} = 01000001000 \dots$$

$$\mathbf{v} = 11000110010010001110010 \dots$$

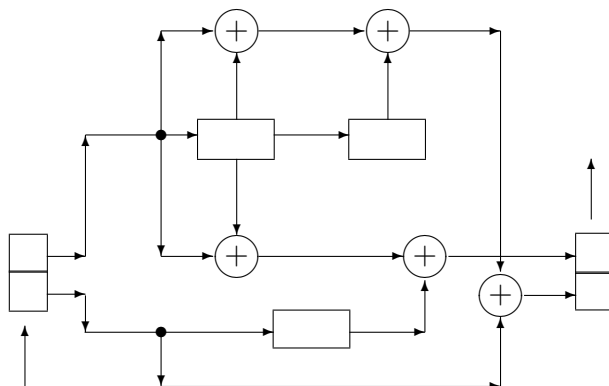
17.3 *Trasați diagrama rețea a $(2, 1)$ - codului convoluțional generat de $g_0(X) = 1 + X^2 + X^3 + X^4$. Folosiți algoritmul Viterbi pentru decodificarea secvenței $\mathbf{v} = 1000001000001000 \dots$*

17.4 *Fie $(2, 1)$ - codul generat de $g_0(X) = 1 + X + X^2 + X^3, \quad g_1(X) = 1 + X^2 + X^3$. Decodificați primele 4 caractere de informație ale cuvântului recepționat $\mathbf{c} = 11000000 \dots$ pentru*

$$b = 2, \quad b = 3, \quad b = 4.$$

17.5 *Generalizați conceptul de diagramă rețea la (n, k) - coduri.*

Trasați rețeaua $(2, 2)$ - codului definit de circuitul liniar:



17.6 Generalizați noțiunea de diagramă de translatăre pentru (n, k) coduri binare.
 Construiți diagrama de translatăre a codului definit în exercițiul anterior.

17.7 Demonstrați Propoziția 17.2,

17.8 Pentru codurile definite în Exercițiul 17.1, determinați diagramele de translatăre și $d(t)$ pentru $1 \leq t \leq \left\lfloor \frac{d_{lib} - 1}{2} \right\rfloor$.

17.9 Ce se întâmplă dacă se încearcă să se determine $d(t)$ pentru $t > \left\lfloor \frac{d_{lib} - 1}{2} \right\rfloor$?

17.10 Construiți un algoritm pentru determinarea lui $d(t)$ în cazul unui $(n, 1)$ - cod convoluțional ($1 \leq t \leq \left\lfloor \frac{d_{lib} - 1}{2} \right\rfloor$ dat).

17.11 Pentru fiecare din codurile următoare, decideți dacă sunt catastrofice sau nu. În caz afirmativ, determinați ciclurile de pondere 0.

- (a) $g_0(X) = 1 + X, \quad g_1(X) = 1 + X + X^2 + X^3;$
- (b) $g_0(X) = 1 + X + X^4, \quad g_1(X) = 1 + X^2 + X^4;$
- (c) $g_0(X) = 1 + X + X^2, \quad g_1(X) = 1 + X + X^3 + X^4.$

Capitolul 18

Coduri - tablou

18.1 Definirea codurilor tablou

În Capitolul 3 au fost definite codurile - produs. Deși construcția prezentată era cunoscută din anii '60, dezvoltarea acestei clase de coduri a fost explozivă abia după trei decenii, odată cu introducerea pe piață a telefoniei mobile. Acum s-a arătat că orice cod - bloc poate fi generat finit recursiv, folosind o variantă de coduri - produs.

Să considerăm o clasă particulară de coduri - produs binare, în care $n_1 = k_1 + 1$, $n_2 = k_2 + 1$ (deci singurul simbol de control este bitul de control al parității). Acest cod este generat de produsul Kronecker a două matrici de forma

$$G_{p,p+1} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 1 \\ 0 & 1 & 0 & \dots & 0 & 1 \\ & & & \vdots & & \\ 0 & 0 & 0 & \dots & 1 & 1 \end{pmatrix} \quad (1)$$

unde $p = k_1, k_2$.

Un astfel de $(n_1 \cdot n_2, k_1 \cdot k_2)$ - cod are distanță $d = d_1 \cdot d_2 = 2 \cdot 2 = 4$, deci corectează o eroare și detectează două erori. Chiar dacă în scrierea liniară a cuvintelor - cod ultimul simbol de control se poate elimina (el este de fapt suma modulo 2 a biților de control de pe ultima linie și coloană) distanța se reduce la 3, deci capacitatea de corectare a unei erori rămâne.

Exemplul 18.1 Fie un $(3, 2)$ - cod binar sistematic, generat de matricea $G = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$.

Codul produs $(3, 2)(3, 2) = (9, 4)$ va avea ca matrice generatoare

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

Dacă $\mathbf{a} = 1001$ este un mesaj de informație, el se codifică în $\mathbf{x} = \mathbf{a}G' = 101011110$, sau - folosind forma matricială:

$$\mathbf{x} = \begin{matrix} 1 & 0 & \underline{1} \\ 0 & 1 & \underline{1} \\ \underline{1} & \underline{1} & \underline{0} \end{matrix} = 10\underline{1011110}$$

caracterele de control fiind subliniate, iar cuvântul - cod este transmis linie cu linie.

18.2 Structura de rețea a codurilor tablou

În cadrul codurilor convoluționale s-a definit o structură de rețea pentru facilitarea codificării, decodificării, detectării și corectării de erori (Prelegerea 17). Această structură se poate extinde cu mici modificări și la codurile liniare.

Fie C un (n, k) - cod liniar care se poate reprezenta printr-un cod - tablou. Lui i se asociază un graf orientat cu arce marcate. Vârfurile grafului sunt partiționate în coloane $V_0, V_1, \dots, V_{N_c-1}$ ($N_c \leq n + 1$). O coloană V_i conține multe stări la care se poate ajunge la momentul i . Fiecare arc U pleacă dintr-o stare $q_j \in V_i$ și ajunge la o stare $q_m \in V_{i+1}$ ($i \geq 0$); în acest caz, q_m este *succesorul* lui q_j , iar q_j este *predecesorul* lui q_m . Arcul este marcat cu o pereche $\delta(U)/l(U)$, unde $\delta(U) \in Z_q^*$ este un mesaj de informație, iar $l(U) \in Z_q^*$ este mesajul - cod corespunzător (de obicei ele se reduc la câte un singur caracter).

Proprietăți:

- (a) $V_0 = \{q_0\}$, $V_{N_c-1} = \{q_{N_c-1}\}$; deci, orice rețea pleacă dintr-o stare unică q_0 (*rădăcina*) și ajunge de asemenea într-o stare unică q_{N_c-1} (*țintă*);
- (b) Orice stare este accesibilă din rădăcină prin cel puțin un drum;
- (c) Din orice stare se poate ajunge la țintă prin cel puțin un drum.

Numim *drum prin rețea* un drum de la rădăcină la țintă. Printr-o rețea se pot cataloga toate cuvintele unui cod liniar reprezentabil sub formă de cod - tablou. Orice cuvânt - cod corespunde unui drum unic prin rețea. Deci o astfel de rețea va conține q^k drumuri distincte. Fiecare astfel de drum are două marcaje:

- (i) concatenarea marcajelor de informație $\delta(U)$ ale fiecărui arc U al drumului; această secvență formează mesajul de informație care se codifică.
- (ii) concatenarea marcajelor $l(U)$ ale fiecărui arc U al drumului; ele formează cuvântul care codifică secvența de informație.

În acest fel, operația de codificare sau decodificare (fără corectare de erori) se reduce pentru fiecare mesaj de informație (cod) la determinarea aceluși drum prin rețea, marcat de mesajul respectiv.

Să considerăm numai $(n_1 \cdot n_2, k_1 \cdot k_2)$ - coduri - tablou binare (generalizarea la cazul nebinar se face fără probleme deosebite). Pentru aceste coduri se pot construi evident mai multe structuri de rețea; vom considera aici numai rețelele cu un număr minim de stări pe fiecare coloană. Construcția unei astfel de rețele va necesita

$$N_s = 2^{\min\{k_1, k_2\}}$$

stări (reamintim, aici $k_i = n_i - 1$, $i = 1, 2$). Procedura de generare ([8]) a rețelei este:

1. Fie $n_1 = \min\{n_1, n_2\}$ (pentru cazul invers se procedează similar);
2. Se determină numărul N_s de stări și numărul N_c de coloane prin relațiile

$$N_s = 2^{n_1-1}, \quad N_c = n_2 + 1;$$

3. Pentru fiecare coloană V_p ($0 \leq p \leq N_c - 1$) se notează elementele (stările) sale prin $(A)_p = (a_1 a_2 \dots a_{k_1})_p$, unde $a_i \in Z_2$;
4. Rădăcina reței este $(00 \dots 0)_0$, iar scopul este $(00 \dots 0)_{N_c-1}$;
5. Pentru orice pereche de stări $(A, B) \in V_p \text{ times } V_{p+1}$ ($0 \leq p < N_c - 2$), se marchează arcul corespunzător cu $\delta_p(A, B)/l_p(A, B)$ definite $\delta_p(A, B) = (A)_p + (B)_{p+1}$, $l_p(A, B) = \delta_p(A, B)c_1$, unde c_1 este bitul de paritate al secvenței ei $\delta_p(A, B)$ iar suma se realizează în Z_2 .
6. Arcele care leagă stările $A \in V_{N_c-2}$ de țintă au $\delta_{N_c-2}(A, B) = \epsilon$, $l_{N_c-2}(A, B) = (A)_{N_c-2}c_1$ unde c_1 este bitul de paritate al secvenței ei $(A)_{N_c-2}$ care definește starea A .

Există $2^{k_1 \cdot k_2}$ drumuri distincte prin rețea, fiecare corespunzând unui cuvânt - cod unic. Procedura poate fi extinsă ușor la codurile produs cu mai multe simboluri de control.

Exemplul 18.2 Să construim rețeaua asociată codului tablou $(3, 2)(3, 2)$ (deci $n_1 = n_2 = 3$, $k_1 = k_2 = 2$, $d = 4$). Ea va avea $N_s = 2^{n_1-1} = 2^{3-1} = 4$ stări distincte și $N_c = n_2 + 1 = 3 + 1 = 4$ coloane.

Fiecare stare va fi de forma $(a_1 a_2)_p$ cu $p = 0, 1, 2, 3$, $a_1, a_2 \in \{0, 1\}$.

$(00)_0$ și $(00)_3$ vor fi rădăcina respectiv ținta reței.

Marcajele arcelor sunt de forma $\delta_p(A, B)/l_p(A, B)$, obținute prin toate combinațiile posibile ale lui $(a_1 a_2)_p$ și $(a_1 a_2)_{p+1}$.

Astfel, pentru $p = 0$ avem:

$$\begin{aligned} \delta_0(00, 00) &= (00)_0 + (00)_1 = 00 & \delta_0(00, 01) &= (00)_0 + (01)_1 = 01 \\ \delta_0(00, 01) &= (00)_0 + (10)_1 = 10 & \delta_0(00, 11) &= (00)_0 + (11)_1 = 11. \end{aligned}$$

iar fiecare arc va fi marcat cu o pereche de forma

$$\delta_0/l_0 = a_1 a_2 / a_1 a_2 c_1$$

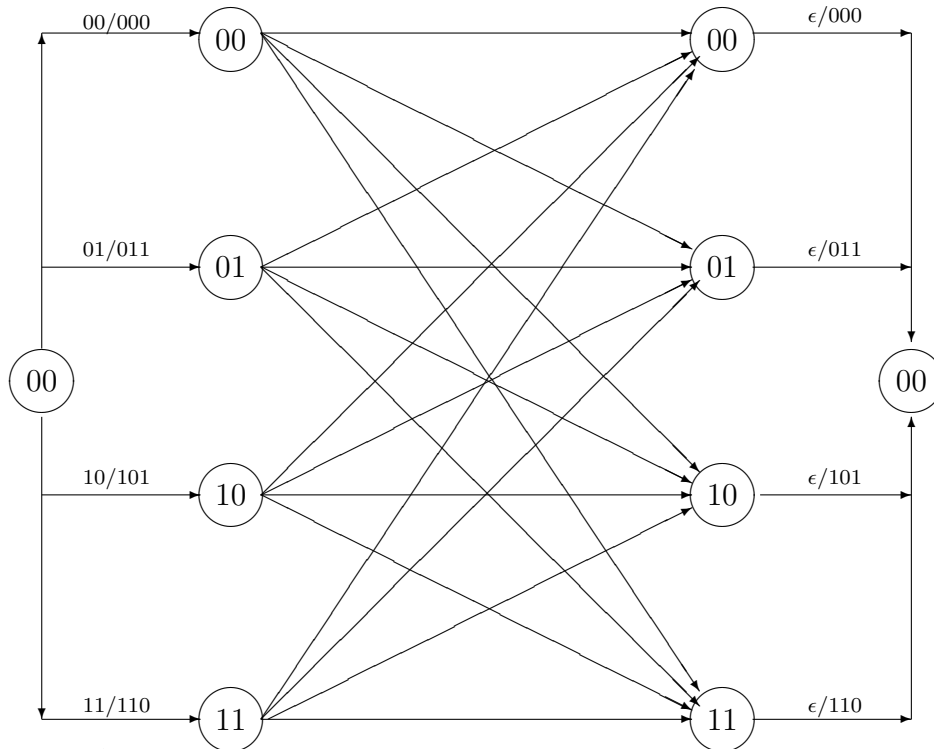
unde $c_1 = a_1 + a_2$.

Pe nivelul $p = 1$, arcele sunt marcate prin perechi din tabelul următor:

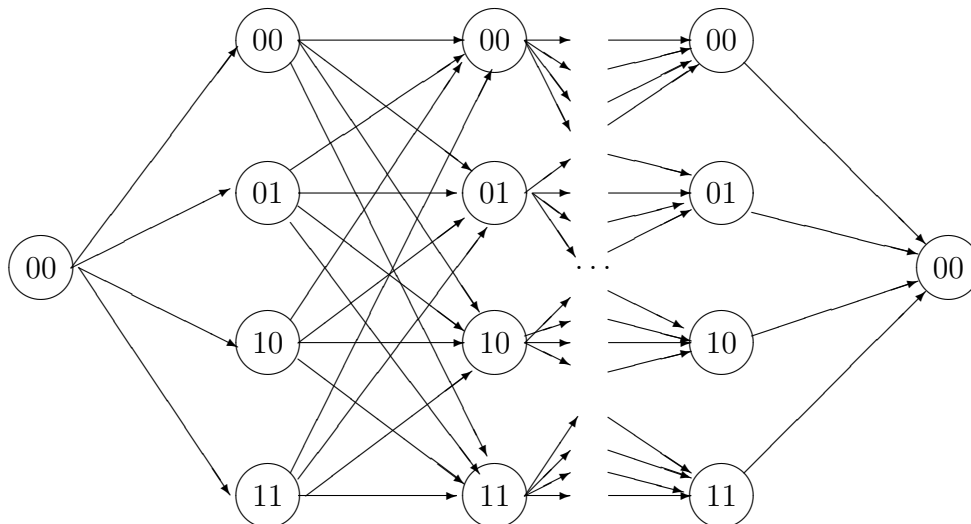
$\delta_1(A, B)$	00	01	10	11
00	00/000	01/011	10/101	11/110
01	01/011	00/000	11/110	10/101
10	10/101	11/110	00/000	01/011
11	11/110	10/101	01/011	00/000

La ultimul nivel, pentru $p = N_c - 2 = 2$ vom avea $\delta_2(A, 00) = \epsilon$ și $l_2(00, 00) = 000$, $l_2(01, 00) = 011$, $l_2(10, 00) = 101$, $l_2(11, 00) = 110$ (pe ultimul arc sunt numai simboluri de control).

Rețeaua se poate reprezenta grafic astfel:



Exemplul 18.3 În mod similar se construiesc codurile tablou obținute prin produsul a două $(3,2)(n_2, k_2)$ - coduri bloc. Toate au același număr de stări $N_s = 2^2 = 4$, doar adâncimea (numărul de coloane) variind în funcție de n_2 ($N_c = n_2 + 1$). Reprezentarea grafică a rețelei este



marcajul nodurilor fiind realizat similar cu cel din Exemplul 18.2.

18.3 Decodificarea codurilor tablou

Operația de codificare se realizează foarte simplu: fiecare mesaj de informație de $k_1 \cdot k_2$ caractere, se scrie ca o secvență de $k_2 + 1$ grupuri, primele k_2 grupuri având câte k_1 simboluri, iar ultimul ϵ . Această secvență trasează prin rețea un drum unic

$$(00 \dots 0) - A_1 - \dots - A_{N_c-2} - (00 \dots 0)$$

de la rădăcină la țintă. Ceea ce se obține prin concatenarea ieșirilor

$$l_0(00 \dots 0, A_1)l_1(A_1, A_2) \dots l_{N_c-2}(A_{N_c-2}, 00 \dots 0)$$

va fi un cuvânt - cod al $(n_1 \cdot n_2, k_1 \cdot k_2)$ - codului - tablou reprezentat de rețea.

Pentru decodificare se poate folosi o variantă a algoritmului Viterbi, cu mici modificări, necesare trecerii de la codurile convoluționale la coduri bloc.

1. Cuvântul recepționat este împărțit în n_2 subcuvinte, fiecare având n_1 valori digitizate (vezi Exemplul 18.4) ale alfabetului - cod.
2. La fiecare nivel p , pentru fiecare arc (A_p, B_{p+1}) se calculează *metrica arcului*, care este distanța euclidiană dintre al $p+1$ -lea subcuvânt recepționat și $l_p(A_p, B_{p+1})$. Distanța euclidiană dintre $\mathbf{x} = x_1x_2 \dots x_n$ și $\mathbf{y} = y_1y_2 \dots y_n$ este

$$d_E(\mathbf{x}, \mathbf{y}) = (x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_n - y_n)^2.$$

3. Pentru fiecare nod B al rețelei se determină *metrica nodului B* , astfel:
 - (a) Metrica nodului rădăcină este 0;
 - (b) Pentru un nod $B_{p+1} \in V_{p+1}$ ($p \geq 0$), metrica lui B este cea mai mică valoare a sumei dintre metrica unui nod $A_p \in V_p$ și $d_E((A)_p, (B)_{p+1})$, minimum fiind luat după toate nodurile din V_p .
4. Arcul selectat în calculul metricii nodului se introduce într-un *drum posibil*, iar toate celelalte arce de pe nivelul respectiv sunt eliminate.
5. După ce s-a calculat metrica nodului țintă, se consideră drumul posibil corespunzător și decodificarea se face prin concatenarea sub-mesajelor de informație care marchează arcele acestui drum.

Exemplul 18.4 Să reluăm codul - tablou definit în Exemplul 18.2. Presupunem că s-a transmis cuvântul cod $\mathbf{a} = 011101110$ și s-a recepționat mesajul digitizat

$$\mathbf{b} = 0.2 \ 0.4 \ 0.4 \ 0.3 \ 0.1 \ 0.9 \ 1.0 \ 0.9 \ 0.6.$$

Presupunând că orice simbol digitizat din intervalul $[0.0, 0.5)$ reprezintă caracterul binar 0, iar $[0.5, 1.0]$ reprezintă 1, putem considera că s-a primit secvența $\mathbf{b} = 000001111$ deci au apărut 4 erori. Într-o decodificare obișnuită a codurilor liniare, deoarece $d = 4$, erorile nu pot fi corectate. Folosind structura de rețea însă, acest lucru este realizabil. Să detaliem pașii decodificării cuvântului \mathbf{b} .

- Metrica nodului rădăcină $(00)_0$ este 0.
- Prima subsecvență de $n_1 = 3$ caractere primite este 0.2 0.4 0.4.
- Se calculează metricile celor patru arce care pleacă din rădăcină:

$$d_E((00)_0, (00)_1) = (0.2 - 0.0)^2 + (0.4 - 0.0)^2 + (0.4 - 0.0)^2 = 0.36;$$

$$d_E((00)_0, (01)_1) = (0.2 - 0.0)^2 + (0.4 - 1.0)^2 + (0.4 - 1.0)^2 = 0.76;$$

$$d_E((00)_0, (10)_1) = (0.2 - 1.0)^2 + (0.4 - 0.0)^2 + (0.4 - 1.0)^2 = 1.16;$$

$$d_E((00)_0, (11)_1) = (0.2 - 1.0)^2 + (0.4 - 1.0)^2 + (0.4 - 0.0)^2 = 1.16.$$

- Metricile celor patru noduri de pe nivelul 1 sunt chiar aceste valori.

– Următorul submesaj primit este 0.3 0.1 0.9. Pentru nodurile de pe nivelul 2, valorile calculate ale metricilor (metrica nodului predecesor plus metrica arcului) sunt:

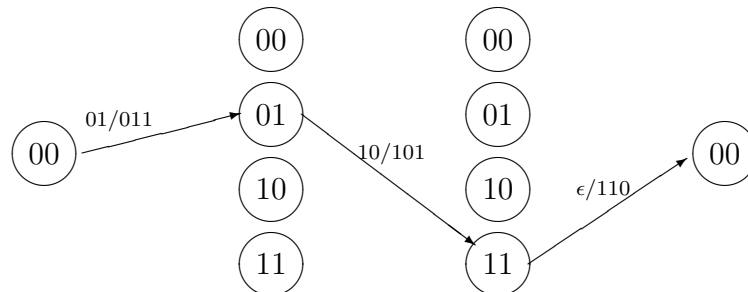
	$(00)_2$	$(01)_2$	$(10)_2$	$(11)_2$
$(00)_1$	1.27	1.27	0.87	2.42
$(01)_1$	1.67	1.67	2.87	1.37
$(10)_1$	1.67	3.27	2.07	2.07
$(11)_1$	3.27	1.67	2.07	2.07

– Minimul pe fiecare coloană ale acestor valori (scris îngroșat) reprezintă metrica nodului respectiv. Deci nodurile $(00)_2$ și $(01)_2$ au metrica 1.27, $(10)_2$ are metrica 0.87, iar $(11)_2$ are metrica 1.37.

– Ultima subsecvență de trei caractere digitizate este 1.0 0.9 0.6. Valorile calculate ale metricilor (metrica nodului predecesor plus metrica arcului) drumurilor care ajung în nodul țintă sunt:

	$(00)_3$
$(00)_2$	3.44
$(01)_2$	2.44
$(10)_2$	1.84
$(11)_2$	1.74

Deci metrica nodului țintă este 1.74. Refăcând acum traiectoria în sens invers, se obține drumul posibil $(00)_0 - (01)_1 - (11)_2 - (00)_3$, reprezentat în figura



Prin concatenarea marcajelor arcelor de pe acest drum, rezultă cuvântul - cod 011101110 (care coincide cu mesajul transmis **a**) și decodificarea lui - mesajul de informație 0110.

18.4 Coduri tablou generalizate (GAC)

Definiția codurilor - tablou se poate extinde pentru a cuprinde și alte clase de coduri liniare (Hamming, Reed - Muller) sau ciclice (*BCH*, Reed - Solomon); ceea ce se obține este un *cod - tablou generalizat* (*GAC* - Generalized Array Codes).

Un *GAC* este o sumă binară de două sau trei coduri - tablou, completate eventual cu linii nule. Procedura de construcție a unui astfel de (n, k) - cod C este:

A. Fie C_1 un cod - produs prin înmulțirea Kronecker G_{A_1} times G_{A_2} , unde:

(i) G_{A_1} este o matrice k_1 times $(k_1 + 1)$ de tipul (1);

(ii) $G_{A_2} = (I_{k_2} | J_{k_2, n_2 - k_2})$ unde I_{k_2} este matricea unitate iar J este o matrice cu toate elementele 1;

(iii) $n = (k_1 + 1) \cdot n_2$, $k_A = k_1 \cdot k_2$.

B. Fie C_2 un cod - produs prin înmulțirea Kronecker $G_{B_1}' \text{times} G_{B_2}$, unde:

(i) G_{B_1} este o matrice $1' \text{times} n_{B_1}$ cu toate elementele 1;

(ii) $G_{B_2} = (G_3|G_4)$ este o matrice $k_B' \text{times} n_{B_2}$ unde G_3 este o matrice cu coloane 0 sau 1, iar G_4 este o matrice de tipul (1) cu k_B linii. Coloanele lui G_{B_2} pot fi eventual permutate.

(iii) $n = n_{B_1} \cdot n_{B_2}$.

C. Fie C_3 un cod - produs prin înmulțirea Kronecker $G_{C_1}' \text{times} G_{C_2}$, unde:

(i) $G_{C_1} = (00 \dots 01)$ are dimensiunea $1' \text{times} n_{C_1}$;

(ii) $G_{C_2} = (11 \dots 1)$ are dimensiunea $1' \text{times} n_{C_2}$;

(iii) $n = n_{C_1} \cdot n_{C_2}$.

D. Codul C este $C_1 + C_2$ sau $C_1 + C_2 + C_3$ unde suma se efectuează în Z_2 ; el are lungimea n , $k = k_A + k_B$ respectiv $k = k_A + k_B + 1$ simboluri de informație și este generat de matricea

$$\begin{pmatrix} G_{A_1}' \text{times} G_{A_2} \\ G_{B_1}' \text{times} G_{B_2} \end{pmatrix} \quad \text{sau} \quad \begin{pmatrix} G_{A_1}' \text{times} G_{A_2} \\ G_{B_1}' \text{times} G_{B_2} \\ G_{C_1}' \text{times} G_{C_2} \end{pmatrix}$$

În exemplele următoare vom prezenta sub formă de coduri GAC unele din cele mai cunoscute coduri - bloc (liniare sau ciclice).

Exemplul 18.5 Să considerăm

$$G_{A_1} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad G_{A_2} = (1 \ 1)$$

$$G_{B_1} = (1 \ 1 \ 1 \ 1), \quad G_{B_2} = (0 \ 1).$$

Dimensiunile codului sunt $n = 4 \cdot 2 = 8$, $k = 3 + 1 = 4$. Matricea generatoare va fi

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

Aceasta corespunde unui cod Reed - Muller $\mathcal{RM}(1, 3)$. Dacă se ignoră ultima coloană (de control), se obține un $(7, 4)$ - cod Hamming binar.

Aranjarea sub formă de tablou a cuvintelor - cod din cele două coduri - produs componente este următoarea:

Fie $\mathbf{a} = a_1 a_2 a_3 a_4$ un mesaj de informație. Atunci cuvântul - cod \mathbf{x} este dat prin:

$$\mathbf{x}_1 = \begin{pmatrix} a_1 & p_1 \\ a_2 & p_2 \\ a_3 & p_3 \\ p_4 & p_5 \end{pmatrix}, \quad \mathbf{x}_2 = \begin{pmatrix} 0 & a_4 \\ 0 & a_4 \\ 0 & a_4 \\ 0 & a_4 \end{pmatrix},$$

$$\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2 = \begin{pmatrix} a_1 & p_1 + a_4 \\ a_2 & p_2 + a_4 \\ a_3 & p_3 + a_4 \\ p_4 & p_4 + a_4 \end{pmatrix} = (a_1, a_4 + p_1, a_2, p_2 + a_4, a_3, p_3 + a_4, p_4, p_4 + a_4),$$

unde $p_i = a_i$ ($1 \leq i \leq 3$) și $p_4 = a_1 + a_2 + a_3$ sunt simboluri de control.

Exemplul 18.6 Codul $\mathcal{RM}(1, 4)$ poate fi generat cu un cod GAC astfel:

$$G_{A_1} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad G_{A_2} = (1 \ 1 \ 1 \ 1)$$

$$G_{B_1} = (1 \ 1 \ 1 \ 1) \quad G_{B_2} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

Deci $n = 4 \cdot 4$, $k = 3 + 2 = 5$ și

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

Eliminând ultimul simbol de control (ultima coloană din matricea G) se ajunge la un $(15, 5)$ - cod binar cu $d = 7$, care este un cod BCH.

Fie $\mathbf{a} = a_1 a_2 a_3 a_4 a_5$ un mesaj de informație. Reprezentarea tabelară a cuvintelor - cod este:

$$\begin{array}{cccc} a_1 & p_1 & p_1 & p_1 \\ a_2 & p_2 & p_2 & p_2 \\ a_3 & p_3 & p_3 & p_3 \\ p_4 & p_4 & p_4 & p_4 \end{array} \quad \mathbf{x}_2 = \begin{array}{cccc} 0 & a_4 & a_5 & a_4 + a_5 \\ 0 & a_4 & a_5 & a_4 + a_5 \\ 0 & a_4 & a_5 & a_4 + a_5 \\ 0 & a_4 & a_5 & a_4 + a_5 \end{array},$$

$$\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2 = \begin{array}{cccc} a_1 & p_1 + a_4 & p_1 + a_5 & p_1 + a_4 + a_5 \\ a_2 & p_2 + a_4 & p_2 + a_5 & p_2 + a_4 + a_5 \\ a_3 & p_3 + a_4 & p_3 + a_5 & p_3 + a_4 + a_5 \\ p_4 & p_4 + a_4 & p_4 + a_5 & p_4 + a_4 + a_5 \end{array},$$

unde $p_i = a_i$ ($1 \leq i \leq 3$), $p_4 = a_1 + a_2 + a_3$ sunt simboluri de control.

Exemplul 18.7 Să plecăm de la următoarele matrici de bază:

$$G_{A_1} = G_{A_2} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

$$G_{B_1} = G_{C_2} = (1 \ 1 \ 1 \ 1), \quad G_{B_2} = G_{C_1} = (0 \ 0 \ 0 \ 1).$$

Se obține un cod $C = C_1 + C_2 + C_3$ cu $n = 4 \cdot 4 = 16$, $k = 3 \cdot 3 + 1 + 1 = 11$ și matrice generatoare

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

El are $d = 4$ și este echivalent cu un cod $\mathcal{RM}(2, 4)$. Dacă se elimină ultimul simbol de control, se obține un $(15, 11)$ - cod Hamming binar.

Fie $\mathbf{a} = a_1 a_2 \dots a_{11}$ un mesaj de informație. Cuvântul - cod \mathbf{x} este format din:

$$\mathbf{x}_1 = \begin{pmatrix} a_1 & a_2 & a_3 & p_1 \\ a_4 & a_5 & a_6 & p_2 \\ a_7 & a_8 & a_9 & p_3 \\ p_4 & p_5 & p_6 & p_7 \end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix} 0 & 0 & 0 & a_{10} \\ 0 & 0 & 0 & a_{10} \\ 0 & 0 & 0 & a_{10} \\ 0 & 0 & 0 & a_{10} \end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ a_{11} & a_{11} & a_{11} & a_{11} \end{pmatrix}$$

$$\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3 = \begin{pmatrix} a_1 & a_2 & a_3 & p_1 + a_{10} \\ a_4 & a_5 & a_6 & p_2 + a_{10} \\ a_7 & a_8 & a_9 & p_3 + a_{10} \\ p_4 + a_{11} & p_5 + a_{11} & p_6 + a_{11} & p_7 + a_{10} + a_{11} \end{pmatrix}$$

Exemplul 18.8 Pentru (24, 12) - codul Golay binar extins se poate da următoarea reprezentare GAC:

$$G_{A_1} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \quad G_{A_2} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$G_{B_1} = (1 \ 1 \ 1) \quad G_{B_2} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$G_{C_1} = (0 \ 0 \ 1) \quad G_{C_2} = (1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1).$$

Fie $\mathbf{a} = a_1 a_2 \dots a_{12}$ un mesaj de informație.

Pentru obținerea cuvântului - cod, avem:

$$\mathbf{x}_1 = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & p_1 & p_2 & p_3 & p_4 \\ a_5 & a_6 & a_7 & a_8 & p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} & p_{13} & p_{14} & p_{15} & p_{16} \end{pmatrix}$$

$$\mathbf{x}_2 = \begin{pmatrix} c_1 & a_9 & c_2 & a_{10} & c_3 & c_4 & a_{11} & c_5 \\ c_1 & a_9 & c_2 & a_{10} & c_3 & c_4 & a_{11} & c_5 \\ c_1 & a_9 & c_2 & a_{10} & c_3 & c_4 & a_{11} & c_5 \end{pmatrix}$$

$$\mathbf{x}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{12} & a_{12} & a_{12} & a_{12} & a_{12} & a_{12} & a_{12} & a_{12} \end{pmatrix}$$

deci

$$\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3 = \begin{pmatrix} a_1 + c_1 & a_2 + a_9 & \dots & a_{11} + p_3 & c_5 + p_4 \\ a_5 + c_1 & a_6 + a_9 & \dots & a_{11} + p_7 & c_5 + p_8 \\ a_{12} + p_9 + c_1 & a_{12} + p_{10} + a_9 & \dots & a_{12} + a_{11} + p_{15} & a_{12} + p_{16} + c_5 \end{pmatrix}$$

De remarcat maniera mult mai simplă de codificare, ea reducându-se la adunări de valori binare (pentru simbolurile de control) și adunări de matrici binare.

18.5 Decodificarea codurilor GAC

Pentru decodificare se poate folosi tot algoritmul lui Viterbi aplicat unei structuri de rețea sub care se pot reprezenta aceste coduri. Structura de rețea se construiește similar cu cea a codurilor - tablou, cu unele mici modificări datorate definiției cuvintelor - cod, ca sume de două sau trei alte secvențe.

Vom considera reprezentarea cuvintelor unui cod GAC sub formă de tablou cu n_1 linii și n_2 coloane. Fie k_i numărul de simboluri noi de informație care pot apărea pe linia i a

cuvintelor - cod și $t = \max_{1 \leq i \leq n_1} \{k_i\}$.

1. Numărul de coloane (adâncimea reț ei) este $N_c = n_1 + 1$, iar numărul de stări distincte este $N_s = 2^t$.
2. Fiecare stare de pe coloana p se notează $(a_1 a_2 \dots a_t)_p$ unde $a_i \in Z_2$ ($1 \leq i \leq t$).
3. Rădăcina reț ei este $(00 \dots 0)_0$ iar ținta $(00 \dots 0)_{n_1}$.
4. Fiecare arc (A_p, B_{p+1}) de pe nivelul p este marcat cu o pereche $\delta_p(A, B)/l_p(A, B)$ unde $\delta_p(A, B)$ este secvenț a de simboluri noi de informaț ie de pe linia p (eventual ϵ), iar $l_p(A, B)$ este linia p a cuvântului - cod.
5. Dacă A_p și B_{p+1} sunt legate printr-un arc și $|\delta_p(A, B)| = s$, atunci $B_{p+1} = A_p + \delta_p(A, B)0^{t-s}$.
6. Dacă construcț ia codului GAC a folosit și codul C_3 , atunci fiecare nod de pe coloana n_2 este legat de țintă prin două arce, marcajul celui de-al doilea arc fiind complementara marcajului primului arc (pe acest nivel $\delta_{n_2}(A, 00 \dots 0) = \epsilon$).

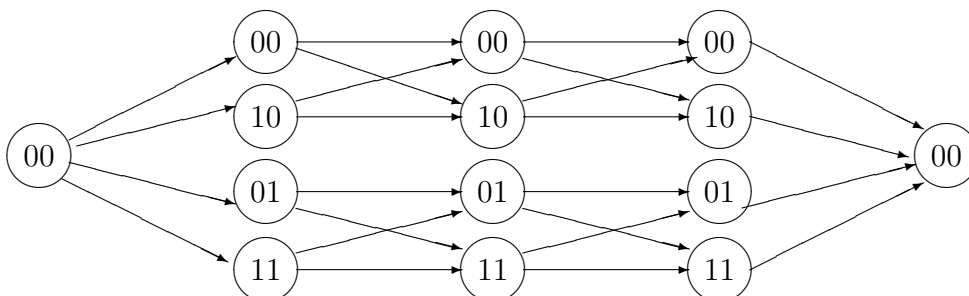
În această construcț ie de reț ea sunt $\prod_{p=1}^{n_1} 2^{k_i}$ drumuri distincte, fiecare din ele core-spunzând unui cuvânt - cod.

Operaț iile de codificare/decodificare (inclusiv aplicarea algoritmului Viterbi a-daptat) sunt identice cu cele definite la codurile - tablou.

Exemplul 18.9 *Reluăm codul definit în Exemplul 18.5. Pentru el se poate construi o reț ea în felul următor:*

Sunt necesare $N_c = n_1 + 1 = 4 + 1 = 5$ coloane. Pentru numărul de stări, să studiem forma de tablou a cuvintelor cod:

$$\mathbf{x} = \begin{matrix} a_1 & p_1 + a_4 \\ a_2 & p_2 + a_4 \\ a_3 & p_3 + a_4 \\ p_4 & p_4 + a_4 \end{matrix}$$



În această reprezentare, pe prima linie sunt două simboluri de informaț ie (a_1 și a_4), pe a doua și a treia câte unul (a_2 respectiv a_3 , a_4 nefiind simbol nou), iar pe a patra linie, nici

unul. Deci numărul de stări al reței este $N_s = 2^2 = 4$, fiecare stare fiind reprezentată printr-o secvență de $t = 2$ caractere binare (00, 01, 10, 11).

Funcția δ este definită prin

$$\delta_0(00, B) = a_1 a_4, \quad \delta_1(A, B) = a_2, \quad \delta_2(A, B) = a_3, \quad \delta_3(A, 00) = \epsilon.$$

Nodurile și arcele sunt marcate cu valorile date de tabelele următoare:

$\delta_0(A, B)/l_0(A, B)$	00	10	01	11
00	00/00	10/11	01/01	11/10
$\delta_1(A, B)/l_1(A, B)$	00	10	01	11
00	0/00	1/11	-	-
10	1/11	0/00	-	-
01	-	-	0/01	1/10
11	-	-	1/10	0/01
$\delta_2(A, B)/l_2(A, B)$	00	10	01	11
00	0/00	1/11	-	-
10	1/11	0/00	-	-
01	-	-	0/01	1/10
11	-	-	1/10	0/01

$\delta_3(A, B)/l_3(A, B)$	00	<i>pentru codul $\mathcal{RM}(1, 3)$,</i>
00	$\epsilon/00$	
10	$\epsilon/11$	
01	$\epsilon/01$	
11	$\epsilon/10$	

$\delta_3(A, B)/l_3(A, B)$	00	<i>pentru (7, 4) - codul Hamming</i>
00	$\epsilon/0$	
10	$\epsilon/1$	
01	$\epsilon/0$	
11	$\epsilon/1$	

Exemplul 18.10 Reprezentarea prin rețea a codurilor $\mathcal{RM}(1, 4)$ și $(15, 5)$ – BCH construite sub formă de GAC în Exemplul 18.6 se realizează astfel:

Adâncimea reței (numărul de coloane) este $N_c = 4 + 1 = 5$. Numărul maxim de simboluri de informație noi se află pe prima linie a cuvintelor - cod (a_1, a_4, a_5) ; deci rețeaua are $N_s = 2^3 = 8$ stări distincte.

Funcția δ este definită

$$\delta_0(000, B) = a_1 a_4 a_5, \quad \delta_1(A, B) = a_2, \quad \delta_2(A, B) = a_3, \quad \delta_3(A, 000) = \epsilon.$$

iar funcția l :

$$l_0(000, B) = a_1(a_1 + a_4)(a_1 + a_5)(a_1 + a_4 + a_5)$$

$$l_1(A, B) = a_2(a_2 + a_4)(a_2 + a_5)(a_2 + a_4 + a_5)$$

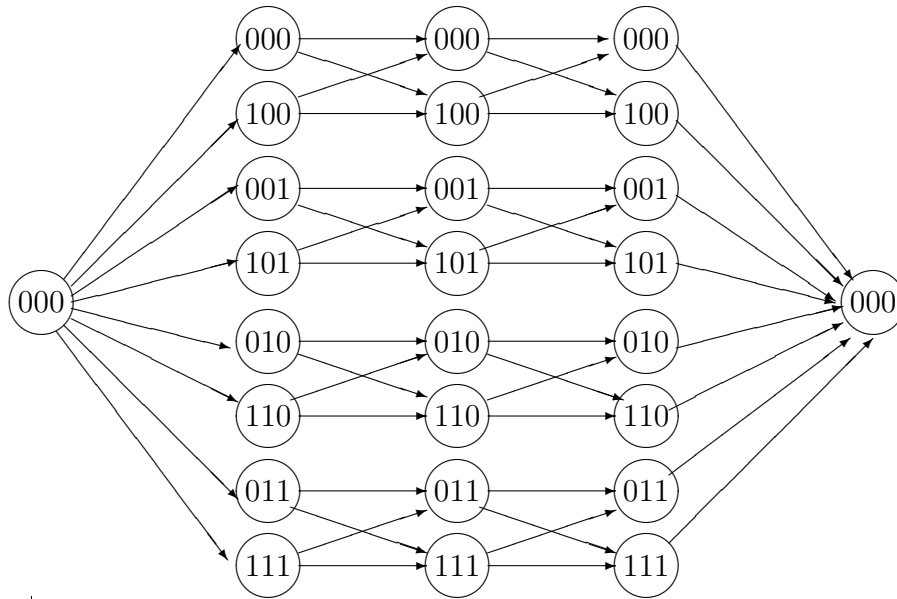
$$l_2(A, B) = a_3(a_3 + a_4)(a_3 + a_5)(a_3 + a_4 + a_5)$$

$$l_3(A, 000) = (a_1 + a_2 + a_3)(a_1 + a_2 + a_3 + a_4)(a_1 + a_2 + a_3 + a_5)(a_1 + a_2 + a_3 + a_4 + a_5)$$

pentru codul $\mathcal{RM}(1, 4)$. Pentru $(15, 5)$ - codul BCH se elimină ultimul bit de control.

Nodurile și arcele sunt etichetate conform tabelelor:

δ_0/l_0	000	100	001	101	010	110	011	111
000	000/0000	100/1111	001/0011	101/1100	010/0101	110/1010	011/0110	111/1001

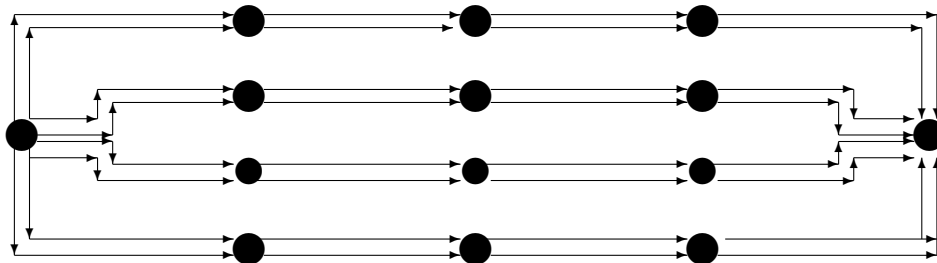


δ_i/l_i	000	100	001	101	010	110	011	111
000	0/0000	1/1111	—	—	—	—	—	—
100	1/1111	0/0000	—	—	—	—	—	—
001	—	—	0/0011	1/1100	—	—	—	—
101	—	—	1/1100	0/0011	—	—	—	—
010	—	—	—	—	0/0101	1/1010	—	—
110	—	—	—	—	1/1010	0/0101	—	—
011	—	—	—	—	—	—	0/0110	1/1001
111	—	—	—	—	—	—	1/1001	0/0110

$(i=1,2)$

δ_3/l_3	0000		δ_3/l_3	0000
000	$\epsilon/0000$		000	$\epsilon/000$
100	$\epsilon/1111$		100	$\epsilon/111$
001	$\epsilon/0011$		001	$\epsilon/001$
101	$\epsilon/1100$	<i>respectiv</i>	101	$\epsilon/110$
010	$\epsilon/0101$		010	$\epsilon/010$
110	$\epsilon/1010$		110	$\epsilon/101$
011	$\epsilon/0110$		011	$\epsilon/011$
111	$\epsilon/1001$		111	$\epsilon/100$

De remarcat că această rețea se poate reprezenta folosind numai 4 stări în loc de 8:



Cele două arce care leagă două noduri sunt marcate cu etichete complementare. Exceptie fac arcele de pe primul nivel unde în definiția funcției δ_0 este complementat numai primul bit (a_1); ceilalți doi biți i de informație a_4a_5 sunt identici pentru fiecare pereche de arce și formează marcajele nodurilor succesive de pe nivelul 1: 00, 01, 10, 11.

Capitolul 19

Alte reprezentări de coduri - bloc

19.1 RLL - coduri

Sistemele de comunicare actuale (telefonice, de înregistrare etc), datorită vitezei mari de lucru, diferă substanțial de canalele clasice (binar simetrice). Pentru ele se folosesc filtre de bandă capabile să elimine fenomenul de interferență care apare. Numite *canale cu răspuns parțial*, ele corectează aceste interferențe folosind algoritmi de tip Viterbi, incorporați prin construcție. Reușita decodificării corecte a cuvintelor recepționate este asigurată de supra-codificări ale mesajului sursă. În prezent sunt folosite trei supra-coduri care asigură transmiterea fără interferențe (în anumite limite) a mesajului: acestea sunt codurile cu lungime limitată și codurile balansate pentru codurile-bloc, turbo-codurile pentru codurile convoluționale.

Definiția 19.1 *Un cod cu lungime limitată (RLL - Run Length Limited Codes), sau (d^0, k^0) - cod, este un cod - bloc cu proprietatea că orice două simboluri 1 consecutive sunt separate prin p zerouri, unde $d^0 \leq p \leq k^0$.*

Parametrul d^0 este folosit pentru controlul interferenței dintre simboluri (fenomen care apare la viteze de transmisie mari, apropiate de capacitatea de saturație a canalului); k^0 impune numărul maxim de zerouri care pot apărea, pentru păstrarea unei rate de informație cât mai convenabile.

Fie C un (n, k) - cod bloc care poate fi reprezentat sub formă de cod tablou $(n_1 \cdot n_2, k_1 \cdot k_2)$, și $\mathbf{c}_{1,n}$ un cuvânt binar de comutație. Codul $C + \mathbf{c} = \{\mathbf{a} + \mathbf{c} | \mathbf{a} \in A\}$ va fi tot un (n, k) - cod, cu restricții RLL.

Dificultatea acestei construcții rezidă în modul de alegere a vectorului \mathbf{c} . De aceea, vom detalia procedura de codificare pentru obținerea de coduri RLL cu $d^0 = 0$:

1. Se definește codul - tablou $(n_1 \cdot n_2, k_1 \cdot k_2)$ care trebuie transformat.
2. Se definește cuvântul de comutație $\mathbf{c} = \mathbf{c}_1 \mathbf{c}_2 \dots \mathbf{c}_{n_2}$, unde $\mathbf{c}_i = c_{i_1} c_{i_2} \dots c_{i_{n_1}}$ ($1 \leq i \leq n_2$) este secvența binară de comutație pentru linia i .
3. Pentru linia i (care este un cuvânt - cod cu un singur simbol de control), cuvântul de comutație verifică următoarea condiție:
Dacă n_1 este par, \mathbf{c}_i are un număr impar de 1,
altfel, \mathbf{c}_i are un număr par de 1.

Exemplul 19.1 Fie $(16, 9)$ - codul tablou, având cuvintele - cod de forma

$$\mathbf{a} = \begin{matrix} a_1 & a_2 & a_3 & p_1 \\ a_4 & a_5 & a_6 & p_2 \\ a_7 & a_8 & a_9 & p_3 \\ p_4 & p_5 & p_6 & p_7 \end{matrix}$$

(este codul C_1 construit în Exemplul 18.7). Deoarece $n_1 = 4$, cuvântul de comutație al fiecărei linii conține un număr impar de 1. Vor exista opt variante posibile pentru \mathbf{c}_i :

$$0001 \quad 0010 \quad 0100 \quad 1000 \quad 1110 \quad 1101 \quad 1011 \quad 0111,$$

deci se pot defini $N_0 = 8^4$ cuvinte de comutație distincte \mathbf{c} .

Dacă se ia de exemplu $\mathbf{c} = 0100110110001110$, codul RLL definit pe baza lui va avea cuvintele - cod

$$\mathbf{a} = \begin{matrix} a_1 & \overline{a_2} & a_3 & p_1 \\ \overline{a_4} & \overline{a_5} & a_6 & \overline{p_2} \\ \overline{a_7} & a_8 & a_9 & p_3 \\ \overline{p_4} & \overline{p_5} & \overline{p_6} & p_7 \end{matrix}$$

unde s-a notat $\overline{x} = 1 - x$.

Tehnica descrisă mai sus se poate extinde la coduri GAC, conducând la coduri cu proprietăți RLL optime ([8]). Generalizarea este următoarea:

1. Dacă n_1 este par, \mathbf{c}_i ($1 \leq i \leq n_2$) conține zero sau un număr impar de 1;
2. Dacă n_2 este impar, \mathbf{c}_i ($1 \leq i \leq n_2$) conține un număr par de 1;
3. Dacă $n = n_1 \cdot n_2$ este par, cuvântul de comutație \mathbf{c} va conține un număr par de 1;
4. \mathbf{c}_i nu este cuvânt - cod al codului care formează liniile; dacă aceasta nu se poate evita, atunci \mathbf{c}_{i+1} se alege de asemenea dintre cuvintele aceluiași cod.

Exemplul 19.2 Fie un $(8, 4)$ - cod binar care codifică fiecare mesaj de informație (a_1, a_2, a_3, a_4) în

$$\mathbf{a} = \begin{matrix} a_1 & a_1 + a_4 \\ a_2 & a_2 + a_4 \\ a_3 & a_3 + a_4 \\ p_1 & p_1 + a_4 \end{matrix}$$

Deoarece n și n_1 sunt pare, alegem \mathbf{c}_i ($1 \leq i \leq 4$) cu număr impar de 1, astfel încât numărul total de 1 din \mathbf{c} să fie par. Putem lua de exemplu

$$\mathbf{c}_1 = 10, \quad \mathbf{c}_2 = 00, \quad \mathbf{c}_3 = 01, \quad \mathbf{c}_4 = 00$$

Deci secvența generală de comutație este $\mathbf{c} = 10000100$ și cuvintele - cod cu restricția RLL sunt

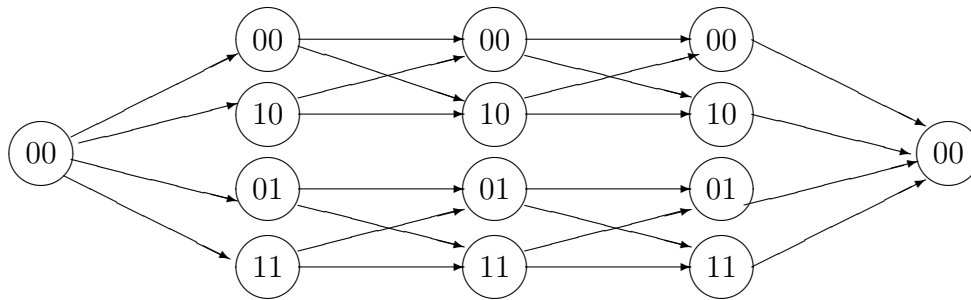
$$\mathbf{a} = \begin{matrix} \overline{a_1} & a_1 + a_4 \\ a_2 & a_2 + a_4 \\ a_3 & a_3 + a_4 \\ p_1 & p_1 + a_4 \end{matrix}$$

Studiind toate cele 16 cuvinte - cod, se găsesc trei cuvinte cod care încep cu trei caractere 1 și trei cuvinte - cod care se termină cu trei caractere 1. Deci $k^0 = 6$ și se poate demonstra că aceasta este valoarea minimă care se poate obține pentru k^0 în cazul $(8, 4)$ - codurilor.

Dacă se relaxează codul prin eliminarea ultimului simbol de control, se ajunge la un $(7, 4)$ - cod Hamming binar, cu $k^0 = 7$.

Pentru decodificarea codurilor *RLL* se poate folosi structura de rețea definită în cazul codurilor *GAC*, cu o singură modificare: la marcajul arcelor, se înlocuiește $l_p(A, B) := l(A, B) + c_p$.

Exemplul 19.3 Să reluăm $(8, 4)$ - codul *GAC* de mai sus, a cărui rețea este descrisă în Exemplul 18.5. Folosind secvența de comutație $\mathbf{c} = 00010001$, se obține un cod *RLL* cu aceeași structură de rețea:



iar marcajele arcelor:

δ_0/l_0	00	10	01	11
00	00/00	10/11	01/01	11/10
δ_1/l_1	00	10	01	11
00	0/01	1/10	—	—
10	1/10	0/01	—	—
01	—	—	0/01	1/11
11	—	—	1/11	0/00
δ_2/l_2	00	10	01	11
00	0/00	1/11	—	—
10	1/11	0/00	—	—
01	—	—	0/01	1/10
11	—	—	1/10	0/01
δ_3/l_3	00			
00	$\epsilon/01$			
10	$\epsilon/10$			
01	$\epsilon/00$			
11	$\epsilon/11$			

(pentru $(7, 4)$ - codul Hamming binar se ignoră ultimul bit din l_3).

De remarcat că față de Exemplul 18.5 s-a modificat numai marcajul nodurilor de pe nivelele 1 și 3).

19.2 Coduri balansate

Codurile balansate reprezintă o clasă specială de coduri *RLL* în care fiecare cuvânt - cod are un număr egal de 0 și 1. Avantajele pe care le oferă această condiție (siguranță a la transmiterea de date la viteze mari, capacitatea de auto-control al tactilor, posibilități de detectare și corectare a erorilor comparabile cu codurile - bloc) o recomandă ca o măsură eficientă de codificare suplimentară.

Algoritm B:

Fie $\mathbf{a} = a_1 a_2 \dots a_k$ un mesaj de informație. Definim următorul cuvânt - cod tablou balansat asociat lui \mathbf{a} :

1. Se alege n_1 astfel ca $2n_1 - 2 \leq k$ și se notează $n = 4 \cdot n_1$.
2. Se construiește un cuvânt - cod tablou cu primele $2n_1 - 2$ caractere de informație:

$$\mathbf{c} = \begin{array}{cccccc} a_1 & a_3 & a_5 & \dots & a_{2n_1-3} & p_1 \\ \overline{a_1} & \overline{a_3} & \overline{a_5} & \dots & \overline{a_{2n_1-3}} & \overline{p_1} \\ a_2 & a_4 & a_6 & \dots & a_{2n_1-2} & p_2 \\ \overline{a_2} & \overline{a_4} & \overline{a_6} & \dots & \overline{a_{2n_1-2}} & \overline{p_2} \end{array}$$

Cuvintele de această formă, citite pe coloană, formează un $(n, 2n_1 - 2)$ - cod tablou balansat C , având $d = 4$. C este folosit pentru definirea caracterelor de control p_1 și p_2 .

3. Se definesc funcțiile $f_1, f_2 : Z_2 \rightarrow Z_2^4$ prin

$$f_1(x, y) = (x, \overline{x}, y, \overline{y})$$

$$f_2(x, y) = (x \wedge y, x \wedge \overline{y}, \overline{x} \wedge y, \overline{x} \wedge \overline{y})$$

unde \wedge este operatorul boolean obișnuit. Evident, coloanele codului C sunt codificate folosind f_1 .

4. Se construiește cuvântul - cod \mathbf{u} folosind ultimele $k - 2n_1 + 2$ caractere de informație $\mathbf{x} = a_{2n_1-1} a_{2n_1} \dots a_k$ astfel:

1. Dacă $\mathbf{x} = 0 \dots 00$, atunci $\mathbf{u} = f_1(a_1, a_2) f_1(a_3, a_4) \dots f(p_1, p_2)$;

2. Dacă $\mathbf{x} = 0 \dots 01$, atunci $\mathbf{u} = f_1(a_1, a_2) f_1(a_3, a_4) \dots \overline{f_1(p_1, p_2)}$;

3. În rest, $\mathbf{u} = g(a_1, a_2) g(a_3, a_4) \dots g(p_1, p_2)$, unde

(i) g este f_2 sau $\overline{f_2}$; fiecare din ele apare de un număr egal de ori (câte $n_1/2$ apariții);

(ii) Dacă se notează f_2 cu 0 și $\overline{f_2}$ cu 1, lui \mathbf{u} i se asociază o secvență α de n_1 caractere binare, cu proprietatea că pentru primele $2n_1 - 2$ caractere de informație fixate, aplicația $\mathbf{x} \mapsto \alpha$ este izotonă față de relația de ordine lexicografică.

5. Cuvântul - cod final se obține prin scrierea pe fiecare linie a unei valori $f_i(x, y)$ din \mathbf{u} . El aparține astfel unui cod GAC A de dimensiuni n_1 times 4.

Sunt multe modalități de transformare a unui cod - bloc binar într-un cod balansat; de exemplu dublarea fiecărui cuvânt - cod \mathbf{a} cu o secvență identică (\mathbf{aa}) sau cu caractere complementate ($\mathbf{a\overline{a}}$) conduc la coduri balansate. Construcția unui cod balansat realizată prin Algoritmul **B** asigură posibilitatea construcției unei rețele de decodificare în maniera celei din Prelegerea 18.

În construcția dată de Algoritmul **B** mai trebuie rezolvată problema alegerii lui n_1 . Aceasta se realizează folosind următoarea teoremă:

Teorema 19.1 Fiind dat k , n_1 este de forma $n_1 = 2s$ unde s este cel mai mic număr pozitiv care verifică inegalitatea

$$2^{k+2-4s} \leq 2 + C_{2s}^s. \quad (1)$$

Demonstrație: Condiția ca n_1 să fie par rezultă din cerința ca în \mathbf{u} jumătate din cele n_1 funcții g să fie f_2 și jumătate să fie $\overline{f_2}$. Fie deci $n_1 = 2s$.

Din condiția $2n_1 - 2 \leq k$ rezultă $k + 2 - 4s \leq 0$.

Deoarece \mathbf{x} este format din $k+2-4s$ caractere binare, sunt 2^{k+2-4s} secvențe \mathbf{x} posibile. Din acestea, $2^{k+2-4s} - 2$ sunt construite folosind numai funcția f_2 , deci trebuie să existe pentru ele cel puțin în tot atâtea secvențe α . Deoarece numărul secvențelor binare α este C_{2s}^s , rezultă inegalitatea din enunț.

Condiția de minim asigură unicitatea codificării. \square

De remarcat că prin această metodă, operația de balansare nu păstrează proprietatea de cod liniar sau ciclic.

Exemplul 19.4 Să construim un cod GAC balansat care să codifice $k = 3$ simboluri de informație. Folosind inegalitatea (1), se obține $s = 1$, deci $n_1 = 2$, $n = 8$.

Codul cadru, care definește simbolurile de control, este un $(8, 2)$ cod bloc având cuvintele - cod de forma (reamintim, reprezentarea desfășurată se face pe coloane):

$$\mathbf{c} = \begin{array}{cc} a_1 & p_1 \\ \overline{a_1} & \overline{p_1} \\ a_2 & p_2 \\ \overline{a_2} & \overline{p_2} \end{array} = a_1 \overline{a_1} a_2 \overline{a_2} p_1 \overline{p_1} p_2 \overline{p_2}$$

unde $p_1 = a_1$, $p_2 = a_2$. Operația de codificare este reprezentată de tabloul:

000 → 01010101	001 → 01011010
010 → 01100110	011 → 01101001
100 → 10011001	101 → 10010110
110 → 10101010	111 → 10100101

Exemplul 19.5 Pentru $k = 4$ simboluri de informație se obține de asemenea $s = 1$ și același $(8, 2)$ - cod cadru, cu două simboluri de control. Cele două caractere de informație rămase, $a_3 a_4$ asigură codificarea în felul următor:

$$\begin{array}{ll} \alpha = 00 & \implies \mathbf{u} = f(a_1, a_2) f(p_1, p_2) \\ \alpha = 10 & \implies \mathbf{u} = f(a_1, a_2) \overline{f(p_1, p_2)} \\ \alpha = 01 & \implies \mathbf{u} = \overline{f_2(a_1, a_2)} f_2(p_1, p_2) \\ \alpha = 11 & \implies \mathbf{u} = \overline{f_2(a_1, a_2)} \overline{f_2(p_1, p_2)} \end{array}$$

Se obține în acest fel un $(8, 4)$ - cod balansat. Din studiul celor 16 cuvinte - cod rezultă că el are $d = 2$.

Exemplul 19.6 Următorul cod balansat a fost studiat de Blaum ([3]), odată cu introducerea metodei de balansare prezentată mai sus. Din $k = 9$, singura valoare posibilă este $s = 2$, deci $n_1 = 4$, $n = 16$. Cuvintele - cod cadru au forma

$$\mathbf{c} = \begin{array}{cccc} a_1 & a_3 & a_5 & p_1 \\ \overline{a_1} & \overline{a_3} & \overline{a_5} & \overline{p_1} \\ a_2 & a_4 & a_6 & p_2 \\ \overline{a_2} & \overline{a_4} & \overline{a_6} & \overline{p_2} \end{array}$$

unde $p_1 = a_1 + a_3 + a_5$, $p_2 = a_2 + a_4 + a_6$.

Procedura de codificare este dată de tabelul următor:

$a_7a_8a_9$	$u_1u_2u_3u_4$	$u_5u_6u_7u_8$	$u_9u_{10}u_{11}u_{12}$	$u_{13}u_{14}u_{15}u_{16}$
000	$f_1(a_1, a_2)$	$f_1(a_3, a_4)$	$f_1(a_5, a_6)$	$f_1(p_1, p_2)$
001	$f_1(a_1, a_2)$	$f_1(a_3, a_4)$	$f_1(a_5, a_6)$	$f_1(p_1, p_2)$
010	$f_2(a_1, a_2)$	$f_2(a_3, a_4)$	$f_2(a_5, a_6)$	$f_2(p_1, p_2)$
011	$f_2(a_1, a_2)$	$f_2(a_3, a_4)$	$f_2(a_5, a_6)$	$f_2(p_1, p_2)$
100	$f_2(a_1, a_2)$	$f_2(a_3, a_4)$	$f_2(a_5, a_6)$	$f_2(p_1, p_2)$
101	$f_2(a_1, a_2)$	$f_2(a_3, a_4)$	$f_2(a_5, a_6)$	$f_2(p_1, p_2)$
110	$f_2(a_1, a_2)$	$f_2(a_3, a_4)$	$f_2(a_5, a_6)$	$f_2(p_1, p_2)$
111	$f_2(a_1, a_2)$	$f_2(a_3, a_4)$	$f_2(a_5, a_6)$	$f_2(p_1, p_2)$

S-a obținut astfel un $(16, 9)$ - cod balansat cu $d = 4$.

Să codificăm mesajul de informație $\mathbf{a} = 001101010$. Pentru determinarea biților de control se construiește cuvântul

$$\mathbf{c} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Deoarece $a_7 = 0$, $a_8 = 1$, $a_9 = 0$, codul va fi generat folosind a treia linie din tabelul de sus:

$$\mathbf{u} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} = 0001\ 1000\ 1101\ 1011.$$

Alte exemple de mesaje de informație și codificările lor:

000000000	0101010101010101
111111001	1010101010100101
000000101	1110000100011110
111111011	1000011110000111

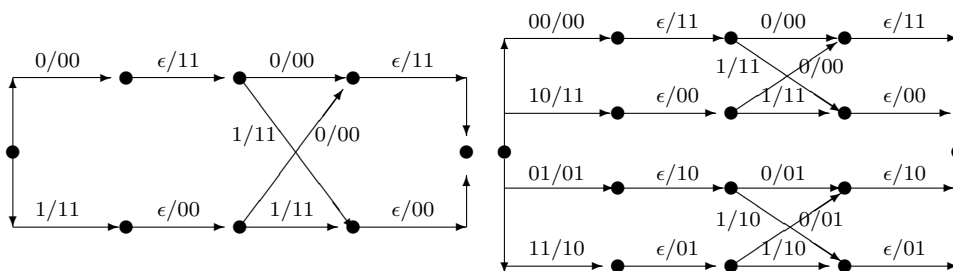
Pentru realizarea unei structuri de rețea a codurilor balansate, dificultatea constă în faptul că aceste coduri nu sunt liniare, deci algoritmi utilizați până acum nu funcționează. Se va utiliza următorul algoritm, specific codurilor balansate definite în această secțiune:

1. Pe baza algoritmilor cunoscuți se construiește structura de rețea R_b a codului - bloc cadru C_b ; fie k_0 numărul ei de coloane (egal cu numărul de coloane din reprezentarea în tablou a cuvintelor - codului C_b).
2. Se copiază rețeaua R_b de $2^p - 1$ ori, unde $p = k - 2n_1 + 2$ este dat de pasul 3 al algoritmului **B**; fiecare copie corespunde unei generări posibile a unui cuvânt - cod balansat.
Toate subrețelele au o singură rădăcină și o singură țintă.
3. În fiecare din cele 2^p subrețele:
 - (a) Arcul $((00 \dots 0)_0, B_1)$ de pe nivelul 0 se marchează cu δ_0/l_0 unde $\delta_0(00 \dots 0, B) = a_{2n_1-1} \dots a_k$, iar $l_0(00 \dots 0, B) = g_1(a_1, a_2)$.
 - (b) Arcul spre țintă are $\delta_{k_0}(A, 00 \dots 0) = \epsilon$ și $l_{k_0}(A, 00 \dots 0) = g(p_1, p_2)$. (notațiile sunt cele din algoritmul **B**).
4. În fiecare subrețea se definește valoarea δ_j/l_j ($1 \leq j < k_0$) a arcului (A_j, B_{j+1}) astfel: $\delta_j(A, B)$ este valoarea nemodificată din R_b , iar $l_j(A, B) = g(a_{2j+1}, a_{2j+2})$.

Exemplul 19.7 Să reluăm $(8, 3)$ - codul balansat din Exemplul 19.4. După ce se construiește rețeaua $(8, 2)$ - codului cadru, deoarece există doar un bit de informație suplimentară avem $p = 1$ deci rețeaua se mai copiază de $2^p - 1 = 1$ ori.

Sunt două maniere de abordare:

A. Dacă transmiterea se face pe linii, apare o construcție conformă cu cea a rețelelor definite pentru codurile GAC. Rețeaua corespunzătoare codului cadru este prezentată în (a), iar cea pentru codul balansat, în (b):



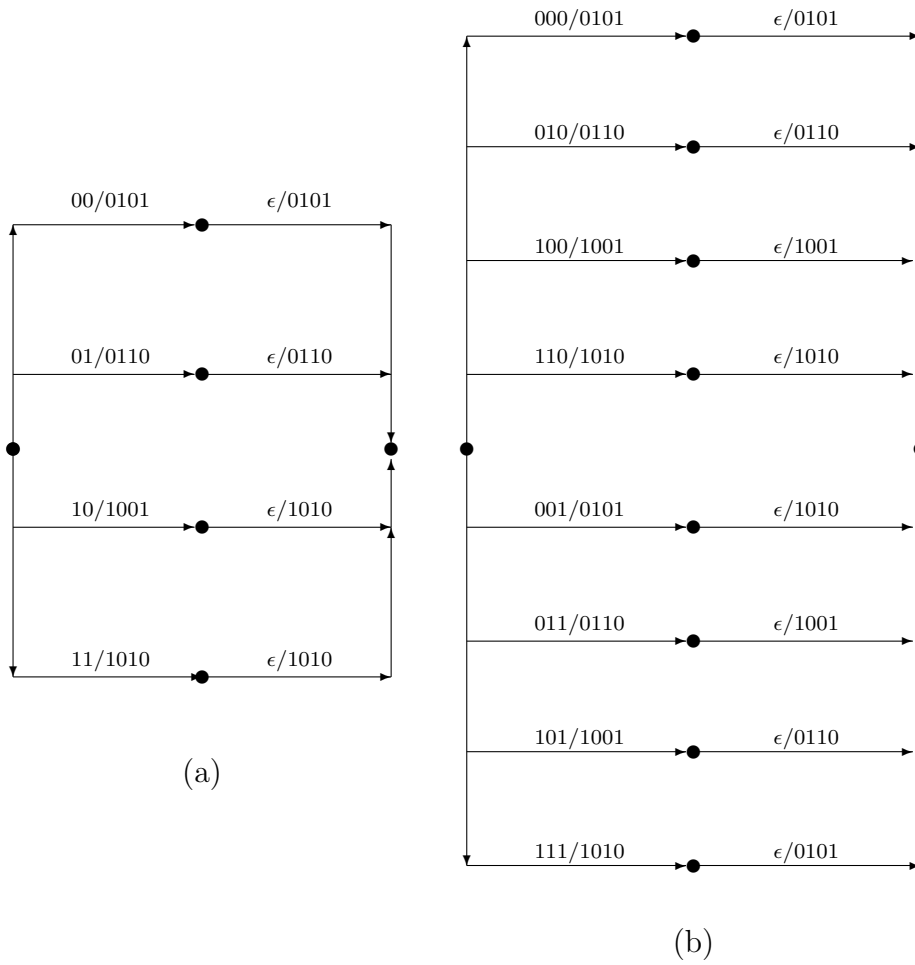
(a)

(b)

B. Dacă transmiterea datelor se face pe coloane (specific codurilor balansate), rețeaua codului cadru este cea prezentată în Figura 19.1, poziția (a); din ea se obține rețeaua (b) a codului balansat.

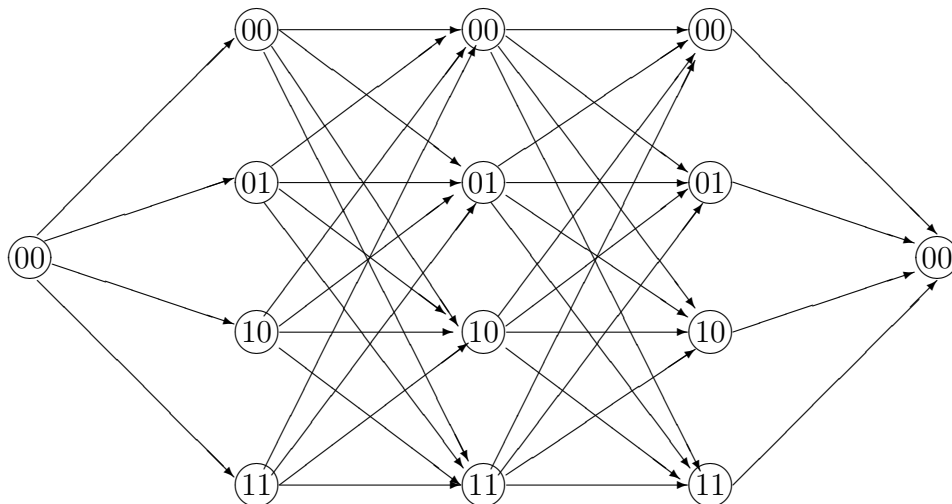
Exemplul 19.8 Să construim rețeaua de (de)codificare pentru $(16, 9)$ - codul balansat definit în Exemplul 19.6. Conform procedurii, sunt necesare 8 subrețele, câte una pentru

Figura 19.1:



fiecare linie din tabelul de codificare. Deoarece cuvintele - cod sunt citite pe coloane, fiecare subreț ea are câte 5 coloane și 4 stări.

Pentru $a_7 = a_8 = a_9 = 0$ (prima linie din tabel) subreț ea ea construită plecând de la (16, 6) - codul cadru din Exemplul 19.6 are structura:



unde marcajul arcelor este dat de tabela:

p	δ_p/l_p	00	01	10	11
0	00	00/0101	01/0110	10/1001	11/1010
1	00	00/0101	01/0110	10/1001	11/1010
	01	01/0110	00/0101	11/1010	10/1001
	10	10/1001	11/1010	00/0101	01/0110
	11	11/1010	10/1001	00/0101	00/0101
2	00	00/0101	01/0110	10/1001	11/1010
	01	01/0110	00/0101	11/1010	10/1001
	10	10/1001	11/1010	00/0101	01/0110
	11	11/1010	10/1001	01/0110	00/0101
3	00	$\epsilon/0101$	—	—	—
	01	$\epsilon/0110$	—	—	—
	10	$\epsilon/1001$	—	—	—
	11	$\epsilon/1010$	—	—	—

Pentru $a_7 = a_8 = 0$, $a_9 = 1$ (a doua linie din tabel) se construiește aceeași subreț ea, în care valorile funcției l_3 din marcajul arcelor ultimului nivel se completează. Prin compunerea acestor două subrețe ele se obțin (16,7) - codul balansat cu $d = 4$.

Celelalte 6 subrețe ele se definesc similar. Rețeaua finală a (16,9) - codului balansat cu $d = 4$ obținută din compunerea lor va avea 32 stări și 5 nivele.

19.3 Exerciții

19.1 Construiți un $(3,2)(3,2)$ - RLL cod tablou și calculați parametrii săi (d^0, k^0) .

19.2 Construiți un $(12,8)$ - cod GAC cu restricții RLL și calculați pentru el (d^0, k^0) .

19.3 Construiți un GAC cod balansat pentru $k = 10$.

19.4 Aceeași problemă pentru $k = 13$.

19.5 Definiți un vector de modificare care transformă codul tablou $(3,2)(3,2)$ într-un cod RLL. Construiți și rețeaua acestui cod.

19.6 Aceeași problemă pentru $(12,8)$ - codul GAC cu $d = 3$.

19.7 Construiți un $(12,5)$ cod balansat cu $d = 4$. Trasați și rețeaua acestui cod.

Capitolul 20

Alte rezultate din teoria codurilor

20.1 Coduri aritmetice

Construcțiile oferite de teoria codurilor pot fi utilizate și în alte domenii decât în cele clasice, de transmitere și recepție corectă a mesajelor. O aplicație legată de operațiile aritmetice pe calculator este prezentată de van Lindt în [11].

Definiția 20.1 Fie $r \geq 2$ un număr întreg fixat. r -ponderea aritmetică unui număr întreg x este definită prin

$$w_{(r)}(x) = \begin{cases} 0 & \text{dacă } x = 0 \\ t & \text{altfel} \end{cases}$$
$$\text{unde } t = \min \left\{ p \left| \sum_{i=1}^p a_i \cdot r^{n(i)} = x, |a_i| < r, n(i) \geq 0 \right. \right\}.$$

În practică se folosesc des cazurile $r = 2, 8, 10, 16$.

Exemplul 20.1

$$w_{(10)}(5) = 1,$$

$$w_{(10)}(199) = 2 \text{ (deoarece } 199 = 200 - 1 = 2 \cdot 10^2 - 1 \cdot 10^0),$$

$$w_{(2)}(-9) = 2,$$

$$w_{(2)}(246) = 3 \text{ (deoarece } 246 = 256 - 8 - 2 = 2^8 - 2^3 - 2^1).$$

Propoziția 20.1

1. $w_{(r)}(-x) = w_{(r)}(x)$;
2. $w_{(r)}(x + y) \leq w_{(r)}(x) + w_{(r)}(y)$.

Demonstrație: Este lăsată ca exercițiu. □

În cele ce urmează vom considera r ca o valoare fixată. Se numește *distanță aritmetică* valoarea

$$d(x, y) = w_{(r)}(x - y).$$

Propoziția 20.2

1. d este o distanță;
2. d este invariantă la translatare.
3. $d(x, y) \leq d_H(x, y)$ unde d_H este distanța Hamming a două secvențe numerice reprezentate în baza r .

Demonstrație: (1) Faptul că d este o distanță se verifică imediat din Definiția 20.1 și Propoziția 20.1.

(2) Deoarece $d(x+z, y+z) = w_{(r)}(x+z-y-z) = w_{(r)}(x-y) = d(x, y)$ rezultă că distanța aritmetică este invariantă la translatore (proprietate pe care distanța Hamming nu o are).

(3) Este lăsat ca exercițiu. Precizăm că cele două numere reprezentate în baza r pot fi aduse la un număr egal de cifre completând eventual numărul mai scurt cu 0-uri (nesemnificative) în față. \square

Definiția 20.2 Fie a, b numere întregi pozitive. Se numește AN - cod mulțimea finită $C_{a,b} = \{a \cdot n \mid 0 \leq n < b\}$.

Ideea folosirii AN - codurilor în aritmetica calculatorului este următoarea: să presupunem că trebuie calculată suma $n_1 + n_2$ (n_1, n_2 pozitive și mici comparativ cu b); fie S suma lor. Cele două numere se codifică în $a \cdot n_1, a \cdot n_2 \in C_{a,b}$. Dacă $S \equiv 0 \pmod{a}$ atunci suma a fost efectuată corect și ea este $S \text{ div } a$. Dacă nu, înseamnă că au apărut erori de calcul și se ia ca rezultat acel număr n_3 cu proprietatea că $d(S, an_3)$ este minimă.

Pentru a corecta orice combinație de maxim t erori este necesar ca C să aibă distanța minimă $\geq 2t + 1$, deci orice număr din C are ponderea minim $2t + 1$.

Teorema 20.1 Fie a un număr întreg fixat și $s = \min\{w_{(r)}(a \cdot n) \mid n \neq 0\}$. Atunci $s \leq 2$.

Demonstrație: Cazurile $a = 0$ și $a = 1$ sunt banale. Pentru $a < 0$ se va lucra cu $-a$. Deci rămâne de studiat numai cazul $a > 1$. Vom folosi Teorema lui Fermat:

$$\text{Dacă } (a, r) = 1 \text{ atunci } r^{\phi(a)} \equiv 1 \pmod{a}.$$

(reamintim, $\phi(a)$ este simbolul Euler).

Deci $a \mid r^{\phi(a)} - 1$. Cum $w_{(r)}(r^{\phi(a)} - 1) = 2$, afirmația este demonstrată pentru cazul când a și r sunt prime între ele.

Să presupunem acum că $(a, r) = d > 1$. Vom avea $a = a_1 d$, $r = r_1 d$ cu $(a_1, r_1) = 1$. Atunci (notând $n = \phi(a_1)$), putem scrie $r^n - 1 = d^n r_1^n - 1 = d^n (r_1^n - 1) + (d^n - 1)$. Se știe că $a_1 \mid r_1^n - 1$. Dacă $(a_1, d) = 1$ atunci $a_1 \mid d^n - 1$, deci $a_1 \mid r^n - 1$, sau $a \mid d \cdot r^n - d$ și cum $d < r$, teorema este demonstrată.

Dacă $(a_1, d) > 1$, raționamentul se reia și el se va termina după un număr finit de pași (deoarece $a_1 < a$, $d < r$). \square

20.1.1 AN - coduri ciclice

Rezultatul menționat în Teorema 20.1 conduce la o dificultate de alegere. O aritmetică a calculatorului eficientă recomandă alegerea unei valori mari pentru b . Pe de altă parte Teorema 20.1 arată că în acest caz riscul de a avea r - ponderea aritmetică minimă cel mult 2 (deci de a nu putea corecta nici o eroare) este mare.

Problema se elimină dacă vom considera codurile AN *modulare*. Fie a, b numere prime și $m = a \cdot b$. Definim $C_{a,b}$ ca subgrup al lui Z_m . Aceasta conduce la o altă definiție pentru distanța dintre două numere. Pentru determinarea ei, să luăm elementele lui Z_m ca vârfuri într-un graf Γ_m ; $a, b \in Z_m$ sunt legate printr-un arc dacă și numai dacă

$$\exists c, j \ (0 < c < r, j \geq 0) \quad a - b \equiv \pm c \cdot r^j \pmod{m}.$$

Definiția 20.3 Distanța modulară $d_m(x, y)$ dintre două numere $a, b \in Z_m$ este lungimea drumului minim dintre a și b în graful Γ_m .

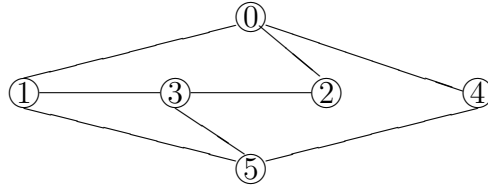
Ponderea modulară a lui $a \in Z_m$ este $w_m(a) = d_m(a, 0)$.

Propoziția 20.3 Pentru $a \in Z_m$

$$w_m(a) = \min\{w_{(r)}(x) \mid x \in \mathcal{Z}, y \equiv x \pmod{m}\}.$$

Demonstrație: Rezultă imediat din Definiția 20.3. □

Exemplul 20.2 Să considerăm $r = 2, a = 2, b = 3$ deci $m = 6$. Graful Γ_6 va fi



deci tabelul distanțelor între elementele lui Z_6 este:

d	0	1	2	3	4	5
0	0	1	1	2	1	2
1	1	0	2	1	2	1
2	1	2	0	1	2	2
3	2	1	1	0	2	1
4	1	2	2	2	0	1
5	2	1	2	1	1	0

De remarcat că alegerea lui m trebuie făcută cu grijă. De exemplu, dacă se ia $r = 3, m = 35$ vom avea $d_{35}(0, 12) = 1$ deoarece $12 = 3^{11} \pmod{35}$. Practic însă, lucrând cu numere din Z_{35} , nu vom putea corecta erori pe poziția corespunzătoare lui 3^{11} .

De aceea, aritmetica pe calculator consideră doar situația $m = r^n - 1$ ($n \geq 2$) care elimină astfel de situații. Orice număr întreg nenul x admite o reprezentare unică

$$x \equiv \sum_{i=0}^{n-1} c_i \cdot r^i \pmod{r^n - 1},$$

cu $0 \leq c_i < r$ nu toți nuli.

Deci Z_{r^n-1} poate fi interpretat ca fiind mulțimea $GF(r^n) \setminus \{0\}$ cuvintelor nenule de lungime n peste alfabetul $\{0, 1, \dots, r - 1\}$.

Pentru $a \cdot b = m = r^n - 1$ distanța modulară devine distanța obișnuită în Z_m și $C_{a,b}$ are un comportament similar unui cod liniar.

Definiția 20.4 Un AN - cod ciclic de lungime n și bază r este un subgrup multiplicativ $C \subseteq Z_{r^n-1}$.

Evident, un astfel de subgrup este ideal principal în inelul Z_{r^n-1} , deci există $a, b \in \mathcal{Z}$ astfel încât $a \cdot b = r^n - 1$ și $C = C_{a,b}$, adică:

$$C = \{a \cdot k \mid k \in \mathcal{Z}, 0 \leq k < b\}.$$

Dacă $x \in C_{a,b}$, atunci $r \cdot x \pmod{r^n - 1}$ este tot în $C_{a,b}$ deoarece acesta este grup multiplicativ; pe de-altă parte noul număr este o permutare ciclică a lui x (ambele fiind scrise în baza r). Numărul b poate fi asimilat polinomului de control al unui cod ciclic.

Exemplul 20.3 Fie $r = 2$, $n = 11$. Atunci $m = 2^{11} - 1 = 2047$. Să considerăm $a = 23$, $b = 89$. Se obține AN - codul ciclic format din 89 multipli ai lui 23 (până la 2047). Există 22 modalități de a semnaliza o eroare, fiecare din ele corespunzând unui număr de forma $\pm 2^j$ ($0 \leq j \leq 11$). Acestea sunt exact toate numerele din $Z_{23} \setminus \{0\}$. Deci orice număr întreg din intervalul $[0, 2047]$ are distanța modulară 0 sau 1 de exact un cuvânt - cod. Acest AN - cod ciclic este perfect și poate fi considerat o generalizare a codurilor Hamming.

În [11] se face afirmația că nu există AN - coduri perfecte corectoare de o eroare pentru $r = 10$ sau $r = 2^k$, ($k > 1$).

20.1.2 AN - coduri corectoare de mai multe erori

Pentru a construi AN - coduri capabile să corecteze erori multiple, va trebui definită o modalitate mai simplă de determinare a ponderii aritmetice sau modulare a numerelor întregi.

Conform Definiției 20.1, orice număr întreg x se poate scrie sub forma

$$x = \sum_{i=1}^{w(r)} a_i \cdot r^{n(i)}$$

cu a_i , $n(i)$ numere întregi, $|a_i| < r$, $n(i) \geq 0$ ($0 \leq i \leq w(r)$). Această reprezentare are defectul că nu este unică. De exemplu, pentru $r = 10$, numărul 99 se poate reprezenta în două moduri diferite:

$$99 = 9 \cdot 10 + 9 \cdot 10^0, \quad 99 = 1 \cdot 10^2 - 1 \cdot 10^0.$$

Se poate obține – prin impunerea de restricții asupra coeficienților – o reprezenta-re unică a numerelor întregi.

Definiția 20.5 Fie $b, c \in \mathcal{Z}$, $|b| < r$, $|c| < r$. Perechea (b, c) se numește admisibilă dacă este adevărată una din relațiile:

- (1) $b \cdot c = 0$;
- (2) $b \cdot c > 0$ și $|b + c| < r$;
- (3) $b \cdot c < 0$ și $|b| > |c|$.

De remarcat că ambele perechi (b, c) , (c, b) sunt admisibile numai în cazurile (1) sau (2). Cazul (3) nu permite comutativitatea relației de admisibilitate.

Exemplul 20.4 Pentru $r = 2$ este posibil numai cazul (1). Deci o reprezentare $x = \sum_{i=0}^{\infty} c_i \cdot 2^i$ în care toate perechile (c_{i+1}, c_i) sunt admisibile, nu are doi coeficienți consecutivi nenuli.

Definiția 20.6 O reprezentare $x = \sum_{i=0}^{\infty} c_i r^i$ cu $c_i \in \mathcal{Z}$, $|c_i| < r$ și $\exists n_x$ cu $c_i = 0 \forall i > n_x$ se numește NAF (non-adjacent form) dacă pentru orice $i \geq 0$, perechea (c_{i+1}, c_i) este admisibilă.

Exemplul 20.5 Pentru $r = 10$ putem scrie:

$$96 = -4 \cdot 10^0 + 0 \cdot 10^1 + 1 \cdot 10^2 \text{ (cazul (1))},$$

$$11 = 1 \cdot 10^0 + 1 \cdot 10^1 \text{ (cazul (2))},$$

$$38 = -2 \cdot 10^0 + 4 \cdot 10^1 \text{ (cazul (3))}.$$

De remarcat că reprezentarea lui $96 = 6 \cdot 10^0 + 9 \cdot 10^1$ nu este în forma NAF deoarece perechea $(9, 6)$ nu este admisibilă. La fel pentru reprezentările celorlaltor numere.

Teorema 20.2 Orice număr întreg x are o reprezentare NAF unică în baza r . Dacă aceasta este

$$x = \sum_{i=0}^{\infty} c_i \cdot r^i,$$

atunci $w_{(r)}(x) = \text{card}(\{i | i \geq 0, c_i \neq 0\})$.

Demonstrație: Fie $\sum_{i=0}^{\infty} b_i \cdot r^i$, ($|b_i| < r$) o reprezentare a lui x în baza r , și i cel mai mic număr cu proprietatea că perechea (b_{i+1}, b_i) nu este admisibilă. Putem presupune că $b_i > 0$ (altfel se va lucra cu $-x$). Vom înlocui b_i cu $b'_i = b_i - r$ și b_{i+1} cu $b'_{i+1} = b_{i+1} + 1$ (dacă $b_{i+1} + 1 = r$, atunci $b'_{i+1} = 0$ și facem deplasarea obișnuită de la adunare).

Dacă $b_{i+1} > 0$, atunci avem sau $b'_{i+1} = 0$, sau $b'_i \cdot b'_{i+1} < 0$ și $b'_{i+1} = b_{i+1} + 1 > r - b_i = |b'_i|$ (deoarece perechea (b_{i+1}, b_i) nu era admisibilă).

Dacă $b_{i+1} < 0$, atunci sau $b'_{i+1} = 0$, sau $b'_i \cdot b'_{i+1} > 0$ și $|b'_i + b'_{i+1}| = r - b_i - b_{i+1} < r$ deoarece $-b_{i+1} \leq b_i$ (perechea (b_{i+1}, b_i) nu era admisibilă).

Deci, (b'_{i+1}, b'_i) este admisibilă, și se verifică similar dacă (b'_i, b_{i-1}) este admisibilă. Procedeeul continuă până se ajunge la $i = 0$.

Să arătăm acum că reprezentarea NAF este unică. Presupunem că există $x \in \mathcal{Z}$ cu două astfel de reprezentări:

$$x = \sum_{i=0}^{\infty} c_i \cdot r^i = \sum_{i=0}^{\infty} c'_i \cdot r^i.$$

Considerăm – fără a micșora generalitatea – că $c_0 \neq c'_0$ și $c_0 > 0$; deci $c'_0 = c_0 - r$. Atunci pentru c'_1 sunt posibile trei valori: $c_1 + 1$, $c_1 + 1 \pm r$. Dacă $c'_1 = c_1 + 1 - r$, atunci $c_1 \geq 0$, deci $c_0 + c_1 \leq r - 1$. Deoarece $c'_0 \cdot c'_1 > 0$, avem $-c'_0 - c'_1 < r$, de unde $r - c_0 + r - c_1 - 1 < r$, deci $c_0 + c_1 > r - 1$, contradicție. Celelalte două cazuri se tratează similar. \square

Reprezentarea NAF a unui număr x se poate afla direct și pe baza teoremei:

Teorema 20.3 Fie $x \in \mathcal{Z}, x \geq 0$. Considerăm reprezentările în baza r a numerelor

$$(r+1) \cdot x = \sum_{i=0}^{\infty} a_i \cdot r^i, \quad x = \sum_{i=0}^{\infty} b_i \cdot r^i,$$

unde $a_i, b_i \in \{0, 1, \dots, r-1\}$. Atunci reprezentarea NAF pentru x este

$$x = \sum_{i=0}^{\infty} (a_{i+1} - b_{i+1}) \cdot r^i.$$

Demonstrație: Știm că pentru două numere naturale nenule a, r , $[a/r]$ reprezintă câtul împărțirii celor două numere, iar $a - [a/r] \cdot r$ - restul.

Din reprezentarea din enunț, rezultă că fiecare coeficient a_i se determină prin adunarea coeficienților corespunzători ai numerelor x și $r \cdot x$, scrise în baza r . Să definim secvența numerică α_i , $i \geq 0$ astfel:

$$\alpha_0 = 0, \quad \alpha_i = \left\lfloor \frac{\alpha_{i-1} + b_{i-1} + b_i}{r} \right\rfloor.$$

Atunci, conform observației de la începutul demonstrației, $a_i = \alpha_{i-1} + b_{i-1} + b_i - \alpha_i \cdot r$. Dacă notăm $c_i := a_i - b_i$, avem $c_i = \alpha_{i-1} + b_{i-1} - \alpha_i \cdot r$.

Mai rămâne de verificat faptul că (c_{i+1}, c_i) este o pereche admisibilă. Relația $|c_{i+1} + c_i| < r$ rezultă imediat din definiția lui α_i . Să presupunem $c_i > 0$, $c_{i+1} < 0$; atunci $\alpha_i = 0$. Vom avea $c_i = \alpha_{i-1} + b_{i-1}$, $c_{i+1} = b_i - r$ și condiția $|c_{i+1}| > |c_i|$ este echivalentă cu $\alpha_{i-1} + b_{i-1} + b_i < r$, adică $\alpha_i = 0$. Celălalt caz se arată analog. \square

Exemplul 20.6 Pentru a găsi reprezentarea NAF a numărului 98 în baza $r = 10$, avem

$$\begin{aligned} 98 &= 8 \cdot 10^0 + 9 \cdot 10^1 + 0 \cdot 10^2 + 0 \cdot 10^3 + \dots \\ 980 + 98 = 1078 &= 8 \cdot 10^0 + 7 \cdot 10^1 + 0 \cdot 10^2 + 1 \cdot 10^3 \\ \text{Deci, } 98 &= (7 - 9) \cdot 10^0 + (0 - 0) \cdot 10^1 + (1 - 0) \cdot 10^2 = -2 + 1 \cdot 10^2. \end{aligned}$$

Similar situației din paragraful anterior, să considerăm acum cazul reprezentării modulare. Vom lua deci $m = r^n - 1$, ($n \geq 2$).

Definiția 20.7 O reprezentare

$$x \equiv \sum_{i=0}^{n-1} c_i \cdot r^i \pmod{m}$$

cu $c_i \in \mathbb{Z}$, $|c_i| < r$ se numește CNAF (cyclic NAF) pentru x dacă $\forall i$ ($0 \leq i \leq n - 1$), (c_{i+1}, c_i) este admisibilă (se consideră $c_n = c_0$).

Din Teoremele 20.2 și 20.3 rezultă un rezultat similar pentru reprezentările CNAF:

Teorema 20.4 Orice număr întreg x admite o reprezentare CNAF modulo m . Această reprezentare este unică, exceptând cazul

$$(r + 1) \cdot x \equiv 0 \not\equiv x \pmod{m},$$

când sunt posibile două reprezentări.

$$\text{Dacă } x \equiv \sum_{i=0}^{n-1} c_i \cdot r^i \pmod{m}, \text{ atunci}$$

$$w_m(x) = \text{card}(\{i | 0 \leq i < n, c_i \neq 0\}).$$

Demonstrație: Construcția este identică cu cea din demonstrația Teoremei 20.3. Unicitatea se arată similar cu cea din demonstrația Teoremei 20.2. Singura excepție este cazul $b_{i+1} \equiv b_i \pmod{m}$. În acest caz sunt posibile două reprezentări:

$$x \equiv b_0 + b_1 \cdot r + \dots + b_{n-1} \cdot r^{n-1} \quad \text{și} \quad x \equiv -b_1 - b_2 \cdot r - \dots - b_0 \cdot r^{n-1},$$

ambele modulo m . \square

20.1.3 Coduri Mandelbaum - Barrows

O clasă de AN coduri a fost definită de Mandelbaum și Barrows, generalizată ulterior de van Lindt ([11]).

Inițial este necesar un rezultat referitor la ponderea modulară în AN coduri ciclice, a cărui demonstrare se află în [11], pag. 127 – 128:

Teorema 20.5 Fie $C \subset \mathcal{Z}/(r^n - 1)$ un AN cod ciclic cu generator a și $b = (r^n - 1)/a = \text{card}(C)$, cu proprietatea că $\forall x \in C$ are o reprezentare CNAF unică. Atunci

$$\sum_{x \in C} w_m(x) = n \left(\left[\frac{r \cdot b}{r + 1} \right] - \left[\frac{b}{r + 1} \right] \right).$$

Teorema 20.6 Fie b un număr prim care nu divide r , cu proprietatea că grupul multiplicativ Z_b este generat de r și -1 . Fie n un număr întreg pozitiv astfel ca $r^n \equiv 1 \pmod{b}$ și $a = (r^n - 1)/b$. Atunci codul $C \subset \mathcal{Z}/(r^n - 1)$ generat de a este un cod echidistant cu distanța

$$\frac{n}{b - 1} \left(\left[\frac{r \cdot b}{r + 1} \right] - \left[\frac{b}{r + 1} \right] \right).$$

Demonstrație: Fie $x \in C \setminus \{0\}$. Atunci $x = a \cdot n \pmod{r^n - 1}$ cu $n \not\equiv 0 \pmod{b}$. Din ipoteză rezultă că există j întreg cu $n \pm r^j \pmod{b}$. Deci $w_m(x) = w_m(\pm r^j \cdot a) = w_m(a)$. Aceasta arată că C este echidistant. Valoarea distanței rezultă din Teorema 20.5. \square

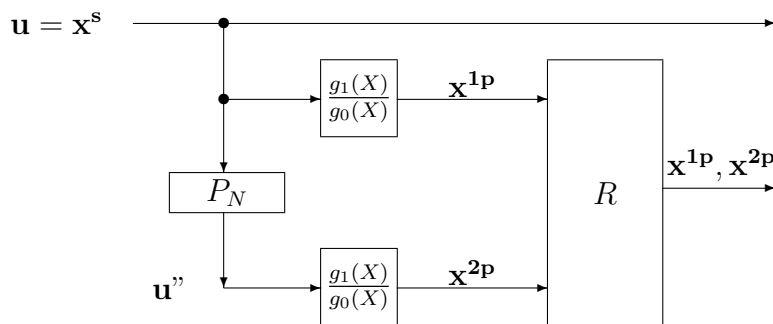
20.2 Turbo - coduri

În ultimii ani, viteza tot mai mare de transmisie a datelor - aproape de capacitatea maximă a canalelor de comunicație - a condus la modalități noi de codificare. Turbo codurile sunt prezentate prima dată în 1993 de Berrou, Glavier și Thitimajshima și combină sub formă de rețea (minim) două coduri convoluționale. Modalitatea de decodificare este total deosebită de algoritmii cunoscuți până acum (se folosesc capacitățile statistice de performanță ale canalelor de transmisie). Ele asigură o rată de corectare a erorilor mult mai ridicată decât la codurile clasice; de aceea turbo - codurile asigură transmisiile de pe stațiile lansate după anul 1993, precum și canalele de satelit.

20.2.1 Structura unui turbo - cod

Un turbo - cod standard este reprezentat de Figura 20.1:

Figura 20.1: Turbo - codificator standard



El folosește două (2, 2) - coduri convoluționale (care - fără a micșora generalitatea - au fost considerate identice), separate printr-un bloc P_N de permutare de N caractere (N fiind o constantă fixată, dependentă de structura canalului de transmisie). Mesajul

de informație este spart în blocuri $u \in Z_q$ de lungime N ; dacă se ignoră mecanismul de relaxare R , rata codicatorului este $1/3$ (N simboluri de informație se transformă în cuvinte - cod de lungime $3N$). Cele 3 ieșiri ($\mathbf{x}^s, \mathbf{x}^{1p}, \mathbf{x}^{2p}$) sunt apoi transmise pe coloană, ca un cod GAC .

Codicatorul

Pentru codul convoluțional, matricea generatoare poate fi considerată (într-o variantă simplificată, bazată pe structura de rețea a mesajelor) $G = (g_0(X) \ g_1(X))$. Codicatorul unui turbo - cod va folosi ca matrice generatoare o formă echivalentă recursivă:

$$G_R^T = \left(1 \quad \frac{g_1(X)}{g_0(X)} \right).$$

Din acest motiv, un codicator convoluțional pentru turbo - coduri este numit *Codicator Sistematic Recursiv* (RSE).

Un mesaj de informație $u(X)$, este codificat de codul convoluțional în $u(X)G = (u(X)g_0(X) \ u(X)g_1(X))$. RSE va realiza aceeași ieșire pentru mesajul $u'(X) = u(X)g_0(X)$ (se verifică imediat relația $u(X)g_0(X)G_R = u(X)G$). Vom numi totuși cuvânt - cod perechea de polinoame $u(X)G$ (deși se mai efectuează o operație de înmulțire pentru obținerea mesajului $u'(X)$).

Se observă că pentru un RSE, cuvântul cod are pondere finită dacă și numai dacă mesajul de intrare se divide cu $g_0(X)$.

Corolarul 20.1 *Un mesaj sursă \mathbf{u}'' cu $w(\mathbf{u}'') = 1$ se codifică într-un cuvânt - cod de pondere infinită.*

Demonstrație: Evident, deoarece $u'(X) = X^p$ nu se divide cu $g_0(X)$. □

Corolarul 20.2 *Pentru orice polinom netrivial $g_0(X) \in Z_q[X]$ există o infinitate de mesaje sursă de pondere 2 care se codifică în cuvinte - cod de pondere finită.*

Demonstrație: Pentru $g_0(X) \in Z_q[X]$, $g_0(X) \neq X^p$, există un n minim cu proprietatea $g_0(X) | X^n - 1$ (n este lungimea secvenței pseudo-aleatoare generată de $g_0(X)$ - a se vedea Relații de recurență liniară, Prelegerea 8).

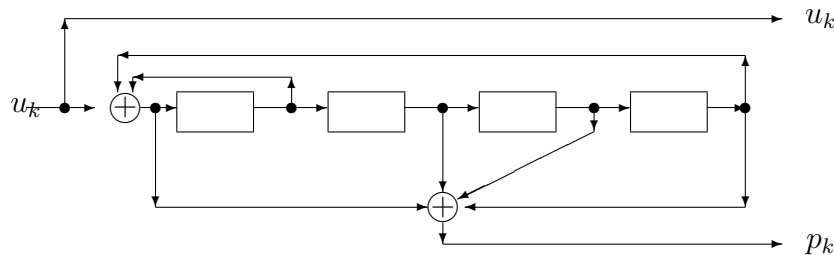
Orice secvență \mathbf{u}'' de forma $u''(X) = aX^i(X^n - 1)$, $a \in Z_q \setminus \{0\}$ are pondere 2 și este divizibilă cu $g_0(X)$, deci codificarea prin RSE va genera un polinom cu un număr finit de termeni. □

Exemplul 20.7 *Fie $g_0(X) = 1 + X + X^4$, $g_1(X) = 1 + X^2 + X^3 + X^4$ polinoame din $Z_2[X]$. În mod uzual se folosește notația în octal; deci, cum $g_0 = 11001_2 = 31_8$, $g_1 = 10111_2 = 27_8$, vom avea $(g_0 \ g_1) = (31, 27)$.*

Matricea generatoare este

$$G_R = \left(1 \quad \frac{1 + X^2 + X^3 + X^4}{1 + X + X^4} \right),$$

iar un circuit liniar care realizează acest RSE are forma:



(*s-a notat cu u_k simbolul de informație curent, iar cu p_k simbolul de control corespunzător*).

Cum $g_0(X)$ este primitiv, lungimea secvențelor este $2^4 - 1 = 15$. De exemplu, mesajul sursă $u(X) = 1 + X^{15}$ se codifică în $(1 + X^{15}, 1 + X + X^2 + X^3 + X^5 + X^7 + X^8 + X^{11}) = (1000000000000001, 1111010110010000)$.

$u(X) = X^7(1 + X^{15})$ va genera același cuvânt - cod, cu o întârziere de șapte tați.

Permutatorul

P_N este un bloc de permutare. Cele N caractere care constituie intrarea în primul codificator RSE sunt rearanjate înainte de a intra în al doilea codificator. Din considerente practice este preferabil ca permutarea folosită să nu păstreze nici o ordine anterioară a simbolurilor (deși acest lucru este uneori dificil, mai ales pentru cuvinte de pondere mică). De asemenea, N trebuie să fie suficient de mare (în practică se folosește $N \geq 1000$). Aceste două cerințe – uzuale în criptografie – sunt necesare în obținerea de performanțe ridicate la decodificare.

Mecanismul de relaxare

Dacă pentru transmiterea de imagini din spațiu sunt folosite coduri cu rate mici de informație (fiecărui bit îi corespund cel puțin 3 caractere cod), în alte situații (comunicări prin satelit de exemplu) sunt preferabile rate mari (cel puțin $1/2$). Rolul mecanismului de relaxare (vezi Capitolul 2) este de a reduce periodic anumite caractere pentru a scurta lungimea cuvintelor - cod. De obicei se elimină biți de control; astfel, pentru a obține o rată de informație $1/2$ se pot elimina toți biții de control pari de la începutul codului și toți biții de control impari de la sfârșit.

20.2.2 Decodificarea turbo - codurilor

Din construcție rezultă că un turbo codificator este liniar, deoarece toate componentele sale sunt liniare. Codificările RSE sunt implementate prin circuite liniare, permutatorul este liniar deoarece poate fi modelat printr-o matrice de permutare. La fel, mecanismul de relaxare nu afectează liniaritatea, deoarece din toate cuvintele - cod se șterg simbolurile de pe aceleași poziții. Importanța liniarității constă în faptul că se poate lua ca referință cuvântul - cod nul. De asemenea, toate construcțiile le facem pentru cazul binar, cu simbolurile ± 1 . Decodificatorul va lucra după principiul obișnuit al decodificării cele mai probabile.

Din păcate, utilizarea algoritmului Viterbi nu este posibilă din cauza operației de permutare folosită în decodificare. Totuși, pentru subsecvențe stricte, un astfel de algoritm poate da rezultate.

Primul algoritm de decodificare pentru turbo - coduri a fost propus de Berrou în 1993 ([12]), bazat pe ideile din [2]. Numit $BCJR$, el folosește o decodificare caracter-cu-caracter

(spre deosebire de Viterbi care decodifica pe secvențe bloc de câte n caractere).

Vom folosi următoarele notații:

- E_i - notarea codicatorului *RSE* i ($1 \leq i \leq 2$);
- D_i - notarea decodicatorului i ($1 \leq i \leq 2$);
- m - capacitatea de memorie (buffer) a codicatorului;
- S - mulțimea celor 2^m stări ale codicatorului;
- $\mathbf{x}^s = x_1^s x_2^s \dots x_N^s = u_1 u_2 \dots u_N$ secvența de informație care se codifică;
- $\mathbf{x}^p = x_1^p x_2^p \dots x_N^p$ cuvântul de control generat de codicator;
- $y_k = y_k^s y_k^p$ o recepție (posibil perturbată) a lui $x_k^s x_k^p$;
- $\mathbf{y}_a^b = y_a y_{a+1} \dots y_b$;
- $\mathbf{y}_1^N = y_1 y_2 \dots y_N$ cuvântul recepționat.

Algoritmul *BCJR* (inițial și modificat)

O primă versiune a algoritmului se bazează pe decodificarea cea mai probabilă *aposteriori* (*MAP* - maximul *aposteriori*). Se realizează decodificarea

$$u_k = \begin{cases} +1 & \text{dacă } P(u_k = +1|\mathbf{y}) > P(u_k = -1|\mathbf{y}) \\ -1 & \text{altfel.} \end{cases}$$

Formal, $\hat{u}_k = \text{sign}[L(u_k)]$, unde $L(u_k)$ este logaritmul raportului probabilităților de potrivire *aposteriori*, definit prin relația

$$L(u_k) = \log \left(\frac{P(u_k = +1|\mathbf{y})}{P(u_k = -1|\mathbf{y})} \right).$$

Dacă se ține cont de faptul că se codifică în rețea, aceasta se scrie:

$$L(u_k) = \log \left(\frac{\sum_{S^+} p(s_{k-1} = s', s_k = s, \mathbf{y})/p(\mathbf{y})}{\sum_{S^-} p(s_{k-1} = s', s_k = s, \mathbf{y})/p(\mathbf{y})} \right) \quad (1)$$

unde

$s_k \in S$ este starea codicatorului la momentul k ,

S^+ este mulțimea perechilor (ordonate) (s', s) corespunzătoare tuturor stărilor de tranziție $(s_{k-1} = s') \rightarrow (s_k = s)$ generate de $u_k = +1$,

S^- este definită similar pentru $u_k = -1$.

Este posibil să simplificăm cu $p(\mathbf{y})$ în (1); deci este necesară doar o formulă pentru calculul lui $p(s', s, \mathbf{y})$. În [2], este construită o variantă sub forma

$$p(s'', s, \mathbf{y}) = \alpha_{k-1}(s'') \cdot \gamma_k(s'', s) \cdot \beta_k(s) \quad (2)$$

unde:

- $\gamma_k(s'', s) = p(s_k = s, y_k | s_{k-1} = s'')$;

- $\alpha_k(s) = p(s_k = s, y_1^k)$ este calculat recursiv cu formula

$$\alpha_k(s) = \sum_{s'' \in S} \alpha_{k-1}(s'') \cdot \gamma_k(s'', s)$$

cu condițiile inițiale $\alpha_0(0) = 1$, $\alpha_0(s \neq 0) = 0$
(codificatorul pleacă din starea 0).

- $\beta_k(s) = p(y_{k+1}^N | s_k = s)$ are formula recursivă de calcul

$$\beta_{k-1}(s'') = \sum_{s \in S} \beta_k(s) \cdot \gamma_k(s'', s)$$

și condițiile $\beta_N(0) = 1$, $\beta_N(s \neq 0) = 0$

(după N biți de intrare codificatorul trebuie să ajungă la starea 0; restricția se realizează alegând corespunzător ultimii m biți, numiți *biți de încheiere*).

Aplicarea acestei variante de decodificare la turbo - coduri (reamintim, algoritmul *BCJR* inițial a fost definit în 1974) are un neajuns: simplificarea cu $p(\mathbf{y})$ conduce la algoritmi numerici instabili. De aceea, Berrou ([12]) face o modificare a algoritmului, în felul următor:

Se definesc probabilitățile (modificate)

$$\tilde{\alpha}_k(s) = \alpha_k(s)/p(y_1^k), \quad \tilde{\beta}_k(s) = \beta_k(s)/p(y_{k+1}^N | y_1^k).$$

Prin împărțirea relației (2) cu $p(\mathbf{y})/p(y_k) = p(y_1^{k-1}) \cdot p(y_{k+1}^N | y_1^k)$ se ajunge la

$$p(s', s | \mathbf{y}) \cdot p(y_k) = \tilde{\alpha}_{k-1}(s') \cdot \gamma_k(s', s) \cdot \tilde{\beta}_k(s).$$

Pentru că $p(y_1^k) = \sum_{s \in S} \alpha_k(s)$, valorile $\tilde{\alpha}_k(s)$ se pot determina din $\alpha_k(s)$ pe baza formulei

$$\tilde{\alpha}_k(s) = \frac{\alpha_k(s)}{\sum_{s \in S} \alpha_k(s)},$$

sau – folosind definiția recursivă a lui $\alpha_k(s)$:

$$\tilde{\alpha}_k(s) = \frac{\sum_{s' \in S} \alpha_{k-1}(s') \cdot \gamma_k(s', s)}{\sum_{s \in S} \sum_{s' \in S} \alpha_{k-1}(s') \cdot \gamma_k(s', s)} = \frac{\sum_{s' \in S} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k(s', s)}{\sum_{s \in S} \sum_{s' \in S} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k(s', s)}$$

ultimul rezultat fiind obținut prin împărțirea numărătorului și numitorului cu $p(y_1^{k-1})$.

Definirea recursivă a lui $\tilde{\beta}_k(s)$ se obține plecând de la

$$p(y_k^N | y_1^{k-1}) = p(y_1^k) \cdot \frac{p(y_{k+1}^N | y_1^k)}{p(y_1^{k-1})} = \sum_{s \in S} \sum_{s' \in S} \alpha_{k-1}(s') \cdot \gamma_k(s', s) \cdot \frac{p(y_{k+1}^N | y_1^k)}{p(y_1^{k-1})} = \sum_{s \in S} \sum_{s' \in S} \tilde{\alpha}_{k-1}(s') \cdot$$

$$\gamma_k(s', s) \cdot p(y_{k+1}^N | y_1^k)$$

și folosind definiția recursivă a lui $\beta_{k-1}(s)$, se ajunge la relația

$$\tilde{\beta}_{k-1}(s') = \frac{\sum_{s \in S} \tilde{\beta}_k(s) \cdot \gamma_k(s', s)}{\sum_{s \in S} \sum_{s' \in S} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k(s', s)}.$$

În final, algoritmul *BCJR* modificat va folosi valoarea $L(u_k)$ dată de relația

$$L(u_k) = \log \left(\frac{\sum_{S^+} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k(s', s) \cdot \tilde{\beta}_k(s)}{\sum_{S^-} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k(s', s) \cdot \tilde{\beta}_k(s)} \right). \quad (3)$$

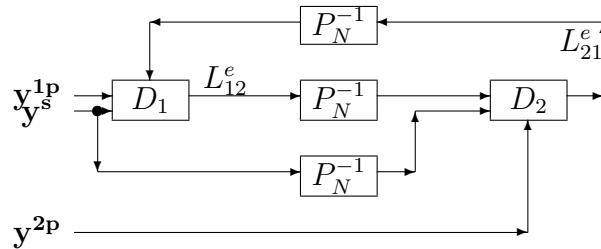
Condițiile la limită pentru $\tilde{\alpha}_k(s)$ și $\tilde{\beta}_k(s)$ sunt cele de la $\alpha_k(s)$ respectiv $\beta_k(s)$.

O altă versiune a algoritmului folosește informația *apriori*. Pentru acesta avem

$$L(u_k) = \log \left(\frac{P(\mathbf{y}|u_k = +1)}{P(\mathbf{y}|u_k = -1)} \right) + \log \left(\frac{P(u_k = +1)}{P(u_k = -1)} \right).$$

Deoarece în mod normal $P(u_k = +1) = P(u_k = -1)$, al doilea termen al sumei este zero în decodificatoarele uzuale. Pentru un turbo - decodor care lucrează recursiv, D_1 primește informație suplimentară de la D_2 , care servește ca informație *apriori*. Similar, D_2 primește de la D_1 informație suplimentară, ș.a.m.d. Ideea constă în faptul că D_2 poate da lui D_1 informații despre u_k la care acesta nu are acces (de exemplu caracterele de control generate de E_2); același lucru îl realizează D_1 pentru D_2 .

Un circuit de decodificare bazat pe algoritmul *BCJR* este:



S-a notat cu P_N^{-1} inversa matricii de permutare P_N din circuitul de codificare. L_{12}^e este informația suplimentară transmisă de la D_1 la D_2 , iar L_{21}^e este cea transmisă de la D_2 la D_1 . Deciziile finale de decodificare pot veni atât de la D_1 cât și de la D_2 .

Mai rămâne de văzut cum se poate obține această informație suplimentară care circulă între cei doi decodificatori recursivi.

Definiția lui $\gamma_k(s', s)$ se poate rescrie

$$\gamma_k(s', s) = P(s|s')p(y_k|s', s) = P(u_k) \cdot p(y_k|u_k)$$

unde evenimentul u_k corespunde tranziției $s's$. Dacă se notează:

$$L^e(u_k) = \log \left(\frac{P(u_k = +1)}{P(u_k = -1)} \right),$$

$$P_+ = P(u_k = +1), \quad P_- = P(u_k = -1), \text{ avem}$$

$$\left(\frac{\sqrt{P_-/P_+}}{1 + P_-/P_+} \right) \cdot \sqrt{P_+/P_-} = P_+ \text{ dacă } u_k = +1,$$

$$\left(\frac{\sqrt{P_-/P_+}}{1 + P_-/P_+} \right) \cdot \sqrt{P_-/P_+} = P_- \text{ dacă } u_k = -1.$$

În această situație se obține

$$P(u_k) = \left(\frac{\exp[-L^e(u_k)/2]}{1 + \exp[-L^e(u_k)]} \right) \cdot \exp[u_k \cdot L^e(u_k)/2] = A_k \cdot \exp[u_k \cdot L^e(u_k)/2], \text{ și (reamintim,}$$

$$y_k = y_k^s y_k^p, \quad x_k = x_k^s x_k^p = u_k x_k^p)$$

$$p(y_k|u_k) \propto \exp \left[-\frac{(y_k^s - u_k)^2}{2\sigma^2} - \frac{(y_k^p - x_k^p)^2}{2\sigma^2} \right] =$$

$$\begin{aligned}
&= \exp \left[-\frac{(y_k^s)^2 + u_k^2 + (y_k^p)^2 + (x_k^p)^2}{2\sigma^2} \right] \cdot \exp \left[\frac{u_k \cdot y_k^s + x_k^p \cdot y_k^p}{\sigma^2} \right] = \\
&= B_k \cdot \exp \left[\frac{y_k^s \cdot u_k + y_k^p \cdot x_k^p}{\sigma^2} \right]
\end{aligned}$$

deci

$$\gamma_k(s', s) \propto A_k \cdot B_k \cdot \exp[u_k \cdot L^e(u_k)/2] \cdot \exp \left[\frac{u_k \cdot y_k^s + x_k^p \cdot y_k^p}{\sigma^2} \right] \quad (4)$$

Deoarece $\gamma_k(s', s)$ apare în (3) la numărător (unde $u_k = +1$) și numitor (unde $u_k = -1$), factorul $A_k \cdot B_k$ se va reduce fiind independent de u_k . De asemenea, particularitățile de canal la transmisia lui ± 1 dau relația $\sigma^2 = N_0/(2E_c)$ unde E_c este energia de canal per bit. Din (4) se obține

$$\begin{aligned}
\gamma_k(s', s) &\sim \exp[u_k \cdot (L^e(u_k) + L_c \cdot y_k^s)/2 + L_c \cdot y_k^p \cdot x_k^p/2] = \\
&= \exp[u_k \cdot (L^e(u_k) + L_c \cdot y_k^s)/2] \cdot \gamma_k^e(s', s),
\end{aligned}$$

$$\text{unde } L_c = \frac{4E_c}{N_0} \text{ și } \gamma_k^e(s', s) = \exp[L_c \cdot y_k^p \cdot x_k^p/2].$$

Combinând această relație cu (3) se ajunge la

$$\begin{aligned}
L(u_k) &= \log \left(\frac{\sum_{s^+} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k^e(s', s) \cdot \tilde{\beta}_k(s) \cdot C_k}{\sum_{s^-} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k^e(s', s) \cdot \tilde{\beta}_k(s) \cdot C_k} \right) = \\
&= L_c y_k^s + L^e(u_k) + \log \left(\frac{\sum_{s^+} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k^e(s', s) \cdot \tilde{\beta}_k(s)}{\sum_{s^-} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k^e(s', s) \cdot \tilde{\beta}_k(s)} \right) \quad (5)
\end{aligned}$$

$$\text{unde s-a notat } C_k = \exp[u_k \cdot (L^e(u_k) + L_c \cdot y_k^s)/2].$$

A doua egalitate rezultă deoarece $C_k(u_k = +1)$ și $C_k(u_k = -1)$ pot fi scoase ca factor. Primul termen al sumei (5) este numit *valoare de canal*, al doilea reprezintă informația apriori despre u_k (furnizată de un decodicator anterior), iar al treilea termen conține informația suplimentară care se trimite la decodicatorul următor. Deci – de exemplu – la orice iterație, D_1 calculează

$$L_1(u_k) = L_c \cdot y_k^s + L_{21}^e(u_k) + L_{12}^e(u_k)$$

unde $L_{21}^e(u_k)$ este informația suplimentară venită de la D_2 și $L_{12}^e(u_k)$ este al treilea termen din (5), folosit ca o informație suplimentară trecută de la D_1 spre D_2 .

Algoritmul *BCJR* se poate formaliza, în unele ipoteze implicite. Astfel:

- se presupune că cei doi codificatori lucrează corect, adică ultimii m biți din mesajul de informație de lungime N se codifică astfel ca la sfârșit E_1 să ajungă la starea zero.

- decodicatorii dețin toată informația referitoare la rețeaua de codificare; astfel, ei au tabele complete cu simbolurile de informație și de control pentru toate tranzițiile de stări $s's$, matricile de permutare și inversele lor.

Algoritmul *BCJR*:

1. (Inițializare):

$$\mathbf{D}_1 : \tilde{\alpha}_0^{(1)}(s) := \begin{cases} 1 & \text{pentru } s = 0 \\ 0 & \text{pentru } s \neq 0 \end{cases} \quad \tilde{\beta}_N^{(1)}(s) := \begin{cases} 1 & \text{pentru } s = 0 \\ 0 & \text{pentru } s \neq 0 \end{cases}$$

$$L_{21}^e(u_k) := 0, \quad k = 1, 2, \dots, N$$

$$\mathbf{D}_2 : \tilde{\alpha}_0^{(2)}(s) := \begin{cases} 1 & \text{pentru } s = 0 \\ 0 & \text{pentru } s \neq 0 \end{cases}, \quad \tilde{\beta}_N^{(2)}(s) := \tilde{\alpha}_N^{(2)}(s), \quad \forall s.$$

$L_{12}^e(u_k)$ se determină din D_1 după prima trecere, deci nu se inițializează.

2. (A n -a iterație):

\mathbf{D}_1 : pentru $k = 1, 2, \dots, N$

- Se determină $y_k := y_k^s y_k^{1p}$ unde y_k^{1p} sunt biții de control recepționați pentru E_1 (posibil perturbați de canal);
- Se determină $\gamma_k(s', s)$ pentru toate tranzițiile posibile $s's$;
- Se determină $\tilde{\alpha}_k^{(1)}(s), \forall s$.
 pentru $k = N, N - 1, \dots, 2$ se determină $\tilde{\beta}_{k-1}^{(1)}(s), \forall s$;
 pentru $k = 1, 2, \dots, N$ se determină $L_{12}^e(u_k)$ cu valorile de probabilități asociate lui D_1 .

\mathbf{D}_2 : pentru $k = 1, 2, \dots, N$

- Se determină $y_k := y_{P_N[k]}^s y_k^{2p}$;
- Se determină $\gamma_k(s', s)$ pentru toate tranzițiile posibile $s's$;
- Se determină $\tilde{\alpha}_k^{(2)}(s), \forall s$;
 pentru $k = N, N - 1, \dots, 2$ se determină $\tilde{\beta}_{k-1}^{(2)}(s), \forall s$;
 pentru $k = 1, 2, \dots, N$ se determină $L_{21}^e(u_k)$ cu valorile de probabilități asociate lui D_2 ;

3. (După ultima iterație):
 Pentru $k = 1, 2, \dots, N$

- Se determină $L_1(u_k) := L_c \cdot y_k^s + L_{21}^e(u_{P_N^{-1}[k]}) + L_{12}^e(u_k)$;
- Dacă $L_1(u_k) > 0$ atunci $u_k := +1$ altfel $u_k := -1$;

4. Stop.

20.3 Exerciții

20.1 Demonstrați Propoziția 20.1.

20.2 Demonstrați Propoziția 20.2, (3).

20.3 Calculați $w_{(2)}$, $w_{(10)}$ și $w_{(16)}$ pentru numerele 100, 32412, 999, 1024.

20.4 Fie $x \in \mathcal{Z}$. Un cod Booth este o reprezentare $x = \sum_{i=0}^{\infty} c_i 3^i$ unde $c_i \in \{-1, 0, 1\}$.

1. Să se reprezinte în codul Booth numerele 23, 455, 81, -6493;
2. Să se arate că pentru orice număr întreg, codul Booth este unic.

20.5 Determinați AN - codurile ciclice din Z_{2^3-1} și Z_{3^3-1} .

Stabiliți valorile a și b pentru fiecare din ele.

20.6 Generalizați Exemplul 20.3. Găsiți un AN - cod ciclic perfect corector de o eroare pentru $r = 3$.

20.7 Scrieți în forma NAF pentru $r = 2$, $r = 10$ și $r = 7$ numerele
 -15 , 32075 , 5665 , -992 .

20.8 Completați demonstrația Teoremei 20.2, verificând unicitatea re-reprezentării NAF în cazurile $c'_1 = c_1 + 1$ și $c'_1 = c_1 + 1 + r$.

20.9 În definiția reprezentării NAF a numărului întreg x (cu completarea $n_0 = -1$), să se arate că

$$n_x \leq k \iff |x| < \frac{r^{k+2}}{r+1}.$$

20.10 Considerăm reprezentarea ternară modulo $3^6 - 1$. Să se determine forma CNAF pentru numărul 455.

20.11 Determinați cuvintele - cod din codul Mandelbaum - Barrows cu $b = 11$, $r = 3$, $n = 5$.

20.12 Fie $g_0(X) = 1 + X + X^3$, $g_1(X) = 1 + X^2 + X^4 + X^5$ din $Z_2[X]$. Construiți circuitul liniar pentru codificatorul RSE.

Codificați mesajele de informație $1 + X^2 + X^3$, $1 + X^7$, $X + X^4 + X^5$.

20.13 Aceeași problemă pentru polinoamele $g_0(X) = 1 + X^3 + X^4$, $g_1(X) = X + X^3 + X^6$.

20.14 Să se construiască un turbo-codificator folosind codificatoarele RSE din Exemplul 20.7, $N = 3$, matricea de comutație $P_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ și fără mecanism de relaxare.

Să se codifice mesajul de informație 100 011 101.

Bibliografie

- [1] J. Adamek - *Foundations of Coding*, Wiley - Interscience, 1991;
- [2] L. Bahl, J. Cocke, F. Jelinek, J. Raviv - *Optimal decoding of linear codes for minimizing symbol error rate*, IEEE Inf. Theory, pp. 284-287, Martie 1974;
- [3] M. Blaum - *A (16, 9, 6, 5, 4) error correcting dc-free block code*, IEEE Transactions on Information Theory, vol. 34, 1988, pp. 38-141;
- [4] C. Carlet - *Codes de Reed - Muller; Codes de Kerdok et de Preparata* (teză de doctorat), PARIS VI, 1990;
- [5] G. Cullmann - *Coduri detectoare și corectoare de erori*, Editura Tehnică, 1972;
- [6] S. Guiașu - *Teoria Codurilor*, Tipografia Universității București, 1976;
- [7] D.G. Hoffman, D.A. Leonard, C.C. Lindner, K.T. Phelps, C.A. Rodger, J.R. Wall - *Coding Theory; The Essentials*, Marcel Dekker, Inc, 1991;
- [8] B. Honary, G. Markarian - *Trellis Decoding of Block Codes, A Practical Approach*, Kluwer Academic Publ., 1997;
- [9] A. M. Kerdok - *A class of low-rate non linear codes*, Information and control, 20 (1972), 182-187;
- [10] J.H. van Lindt - *Coding Theory*, Springer Verlag, 1971;
- [11] J.H. van Lindt - *Introduction to Coding Theory*, Springer Verlag, 1982;
- [12] W.E.Ryan - *A Turbo Code Tutorial*, IEEE Trans. comm. pp. 1261-1271, Oct. 1996;

Cuprins

Cuprins

Prefață

Cuprins

1. Codificare și decodificare

- (a) Codificare
- (b) Exemple de coduri - bloc importante
- (c) Construcția codurilor instantanee
- (d) Coduri Huffman

2. Coduri liniare

- (a) Matrice generatoare
- (b) Matrice de control
- (c) Sindrom
- (d) Pondere, distanță Hamming
- (e) Detectare și corectare de erori
- (f) Capacități de detectare și corectare de erori
- (g) Modificări ale codurilor liniare
- (h) Detectarea și corectarea simultană a erorilor
- (i) Probabilitatea nedetectării erorilor
- (j) Identitatea MacWilliams

3. Clase de coduri liniare

- (a) Coduri Hamming
- (b) Codul Golay
- (c) Unicitatea codurilor perfecte binare
- (d) Coduri Reed - Muller
- (e) Decodificarea codurilor Reed - Muller
 - i. Decodificarea majoritară
 - ii. Algoritm geometric de decodificare majoritară
 - iii. Algoritm algebric de decodificare majoritară
 - iv. Decodificarea rapidă a codurilor $\mathcal{RM}(1, m)$
- (f) Codurile MacDonald
- (g) Coduri derivate din matricile Hadamard

- (h) Coduri produs
 - (i) Coduri optimal maximale
4. Circuite liniare și extensii Galois
- (a) Circuite liniare pentru operații elementare
 - (b) Extensii Galois
5. Coduri ciclice
- (a) Relații de recurență liniară
 - (b) Definierea codurilor ciclice
 - (c) Generarea codurilor ciclice ca ideale
 - (d) Generarea codurilor ciclice folosind circuite liniare
6. Decodificarea codurilor ciclice
- (a) Detectarea și corectarea erorilor
 - (b) Detectarea pachetelor de erori
 - (c) Corectarea pachetelor de erori
 - (d) Transpunere
7. Alte definiții ale codurilor ciclice
- (a) Elemente primitive în extensii Galois
 - (b) Polinoame minimale
 - (c) Definierea codurilor ciclice prin rădăcini
 - (d) Altă definiție a codurilor ciclice
8. Coduri BCH
- (a) Definierea codurilor BCH
 - (b) Algoritmul Peterson de decodificare
 - (c) Localizarea erorii la codurile BCH binare
9. Coduri Reed - Solomon
- (a) Definierea codurilor RS
 - (b) Coduri RS scurte
 - (c) Decodificarea codurilor RS
 - (d) Altă construcție a codurilor RS
 - (e) Ștergeri
 - (f) Aplicații la codificarea mesajelor pe CD-uri

10. Alte clase de coduri ciclice
 - (a) Coduri Hamming ciclice
 - (b) Coduri Hamming ciclice nebinare
 - (c) Coduri Goppa
 - (d) Sistemul de criptare McEliece
11. Coduri Preparata
 - (a) Definirea codurilor Preparata extinse
 - (b) Codificarea codurilor Preparata extinse
 - (c) Decodificarea codurilor Preparata extinse
 - (d) Coduri înrudite cu codul Preparata
12. Coduri convoluționale
 - (a) Coduri liniare și coduri convoluționale
 - (b) Codificarea codurilor convoluționale
 - (c) Decodificarea codurilor convoluționale
 - (d) Algoritmul Viterbi
13. Coduri rețea
 - (a) Structura de rețea a codurilor bloc
 - (b) Tehnici de codificare și decodificare în rețea a codurilor
 - (c) Aplicații la telefonie mobilă a codurilor rețea
14. Alte clase de coduri
 - (a) Coduri aritmetice
 - (b) Turbo - coduri

2

Coduri detectoare/corectoare de erori. Criptarea informației

1. Prezentare teoretică

În cadrul acestei lucrări de laborator se vor prezenta algoritmi CRC și Reed-Solomon folosiți la detectarea și corectarea erorilor care pot apărea într-o transmisie de date. Algoritmii RSA și IDEA prezentați sunt uzual folosiți pentru criptarea informației și se bazează pe chei publice. Implementările hardware ale altor algoritmi de criptare, care se bazează pe metode tradiționale (de exemplu algoritmul de criptare DES), pot fi studiate la <http://www.csit-sun.pub.ro/resources>.

Sume de control

Scopul unei tehnici de detecție a erorilor este acela de a pune la dispoziția receptorului unui mesaj, transmis printr-un canal cu zgomote (posibil de introducere de erori), o metodă de a determina dacă mesajul a fost corupt sau nu. Pentru a face posibil acest lucru, emițătorul construiește o valoare numită sumă de control care este o funcție de mesaj și o anexează acestuia. Receptorul poate să folosească aceeași funcție pentru a calcula suma de control pentru mesajul primit, iar apoi să o compare cu suma de control anexată (concatenată mesajului) pentru a vedea dacă mesajul a fost receptat corect.

Exemplu Să se aleagă o funcție care are ca rezultat (sumă de control) suma octeților din mesaj modulo 256:

$$f(x) = \sum(\text{octeti mesaj}) \bmod 256 \quad (1)$$

Considerând toate valorile în zecimal, se obține:

<i>mesaj</i>	:	7 24 3
<i>mesaj cu suma de control</i>	:	7 24 3 34
<i>mesaj după transmisie</i>	:	7 28 3 38

Al doilea octet al mesajului a suferit o modificare în timpul transmisiei, de la 24 la 28. Cu toate acestea, receptorul poate determina prezența unei erori comparând suma de control transmisă (34) cu cea calculată ($38 = 7 + 28 + 3$).

Dacă însăși suma de control este coruptă, un mesaj transmis corect poate fi (incorect) interpretat drept unul eronat. Acesta nu este însă un eșec periculos. Un eșec periculos are loc atunci când atât mesajul cât și suma de control se modifică astfel încât rezultă într-o transmisie consistentă intern (interpretată ca neavând erori).

Din păcate, această posibilitate nu poate fi evitată și cel mai bun lucru care se poate realiza este de a minimiza probabilitatea ei de apariție prin creșterea cantității de informație din suma de control (de exemplu, lărgind dimensiunea ei la doi octeți în loc de unul).

Coduri CRC

Ideea de bază pentru algoritmi CRC este de a trata mesajul drept un număr reprezentat în binar, de a-l împărți la un alt număr binar fixat și de a considera restul drept sumă de control. La primirea mesajului, receptorul poate efectua aceeași împărțire și poate compara restul cu suma de control primită (restul transmis).

Exemplu Considerând că mesajul care trebuie transmis este alcătuit din 2 octeți (6, 23), el este reprezentat în baza 16 ca numărul 0617 și în baza 2 ca 0000_0110_0001_0111. Se presupune folosirea unei sume de control de 1 octet și a unui împărțitor constant 1001. Atunci suma de control va fi restul împărțirii $0000_0110_0001_0111 : 1001 = \dots 0000010101101$, rest 0010. Mesajul transmis de fapt va fi: 06172, unde 0617 este mesajul inițial (informația utilă), iar 2 este suma de control (restul).

Aritmetica binară fără transport

Toate calculele executate în cadrul algoritmilor CRC sunt realizate în binar, fără transport. Deseori se folosește denumirea de aritmetică polinomială, dar în continuare se va folosi denumirea de aritmetică CRC deoarece la implementarea cu polinoame s-a renunțat.

Adunarea a două numere în aritmetica CRC, așa cum se poate observa în figura 1, este asemănătoare cu adunarea binară obișnuită, însă nu există transport. Aceasta înseamnă că fiecare pereche de biți corespondenți determină bitul corespondent din rezultat, fără nici o referință la alt bit din altă poziție (așa cum se poate observa din exemplul prezentat în figura 2 a).

Definiția operației de scădere este identică cu operația de adunare și poate fi observată în figura 1, iar un exemplu este prezentat în figura 2 b).

Se poate concluziona că atât adunarea cât și scăderea în aritmetica CRC sunt echivalente cu operația SAU EXCLUSIV (XOR), iar operația XOR este propria sa inversă. Acest fapt reduce operațiile primului nivel de putere (adunare, scădere) la una singură, care este propria sa inversă (o proprietate foarte convenabilă a acestei aritmetici).

a	b	a + b	a	b	a - b
0	0	0	0	0	0
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	1	1	0

Figura 1: Definiția operațiilor de adunare/scădere.

Pe baza adunării, se poate defini și înmulțirea, care se realizează natural, fiind suma dintre primul număr deplasat corespunzător și cel de-al doilea număr (se folosește adunarea CRC). Un exemplu pentru această operație este prezentat în figura 2 c).

Pentru realizarea operației de împărțire, este nevoie să se cunoască când *un număr este cuprins în altul*. De aceea, se va considera următoarea definiție: *X este mai mare decât sau egal cu Y dacă poziția celui mai semnificativ bit 1 al lui X este mai mare sau aceeași cu poziția celui mai semnificativ bit 1 al lui Y*. Un exemplu complet este prezentat în figura 2 d).

Transmisia - recepția datelor folosind CRC

Așa cum s-a arătat până acum, calculul CRC este de fapt o simplă împărțire. Pentru realizarea unui calcul CRC este nevoie de un divizor, denumit în limbaj matematic *polinom generator*. Lungimea polinomului uzuală este de 16 sau 32 de biți, CRC-16, CRC-32, și aceste dimensiuni sunt folosite în calculatoarele digitale moderne. *Lungimea unui polinom - W-* este de fapt poziția celui mai semnificativ bit 1 (lungimea polinomului 10011 este 4).

La transmițător, înainte de calculul CRC, se adaugă *W* biți cu valoarea 0 la sfârșitul mesajului care va fi împărțit folosind aritmetica CRC la polinom, astfel încât toți biții mesajului să participe la calculul CRC. Un exemplu este prezentat în figura 2 d). Împărțirea produce un cât, care nu va fi ignorat și un rest, care este suma de control

calculată (CRC-ul). În mod uzual CRC-ul este apoi adăugat mesajului, iar rezultatul este trimis către receptor, în acest caz se transmite 11010110111110.

a.)	b.)	c.)	d.)
$\begin{array}{r} 10011011 + \\ 11001010 \\ \hline 01010001 \end{array}$	$\begin{array}{r} 10011011 - \\ 11001010 \\ \hline 01010001 \end{array}$	$\begin{array}{r} 1101 \times \\ 1011 \\ \hline 1101 \\ 1101 \\ 0000 \\ \hline 1101 \\ 1111111 \end{array}$	$11010110110000 : 10011 = 1100001010, \text{ REST } 1110$ <hr/> $\begin{array}{r} 10011 \\ 10011 \\ \hline 10011 \\ 00001 \\ \hline 00000 \\ 00010 \\ 00000 \\ \hline 00101 \\ 00000 \\ \hline 01011 \\ 00000 \\ \hline 10110 \\ 10011 \\ \hline 01010 \\ 00000 \\ \hline 10100 \\ 10011 \\ \hline 01110 \\ 00000 \\ \hline 1110 \end{array}$

Figura 2: Exemplificarea operațiilor binare fără transport

Receptorul calculează suma de control pentru întreg mesajul primit (fără adăugare de zerouri) și compară restul cu 0. Realizarea acestei operații este motivată de faptul că mesajul transmis T este multiplu de polinomul folosit drept divizor.

Implementarea directă

CRC-ul se poate calcula utilizând noțiunile teoretice prezentate până acum. Algoritmul, implementarea Verilog precum și rezultatele simulării pot fi observate în figura 3.

Implementarea bazată pe tabelă

Acest algoritm este o variantă îmbunătățită a algoritmului anterior, el fiind foarte eficient deoarece implică doar o deplasare, o operație SAU, o operație SAU EXCLUSIV și un acces la memorie pentru fiecare octet. Algoritmul precum și rezultatele implementării sale în Verilog sunt prezentate în figura 4.

```

module crc_simple(message, polynomial, message_crc);
  /* Parametrii: */
  parameter crc_width = 4; /* lungimea CRC-ului; */
  parameter msg_width = 10; /* lungimea mesajului; */
  /*Intrari, iesiri si resurse fizice interne*/
  /* -> mesajul initial: */
  input [msg_width - 1 : 0] message;
  wire [msg_width - 1 : 0] message;
  /* -> polinomul generator: */
  input [crc_width : 0] polynomial;
  wire [crc_width : 0] polynomial;
  /* -> mesajul cu CRC-ul adaugat: */
  output [msg_width + crc_width - 1 : 0] message_crc;
  reg [msg_width + crc_width - 1 : 0] message_crc;
  reg [crc_width - 1 : 0] crc; /* the CRC register; */
  reg msb_crc; /*cel mai semnificativ bit al registrului CRC; */
  /* resurse logice: */
  integer count; /* numarator; */
  /* Calculeaza CRC-ul si atasare: */
  always @(message | polynomial) begin
    /* Initializare: */
    message_crc = message;
    message_crc = message_crc << crc_width;
    crc = 'b0;
    /* Calculeaza CRC-ul: */
    for (count = msg_width + crc_width - 1; count >= 0; count = count - 1)
      begin
        msb_crc = crc[crc_width - 1];
        crc = crc << 1;
        crc[0] = message_crc[count];
        if (msb_crc == 1) begin
          crc = crc ^ polynomial;
        end
      end
    /* Adauga CRC-ul mesajului: */
    message_crc[crc_width - 1 : 0] = crc;
  end
endmodule

```

Algorithm:

1. încarcă registrul cu biți 0
2. adaugă la sfârșitul mesajului W biți 0.
3. cât timp [mesajul mai are biți] deplasează registrul stânga cu un bit, introducând următorul bit din mesaj în poziția 0
4. dacă (un bit 1 a fost scos din registru) registru = registru XOR polinom
5. registrul conține restul

Name	Value	St...	12.48 ns
message	1101011011	1101011011	
polynomial	10011	10011	
message_crc	1101011011...	110101101110	

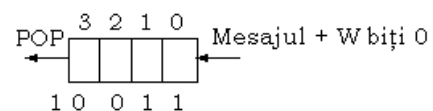
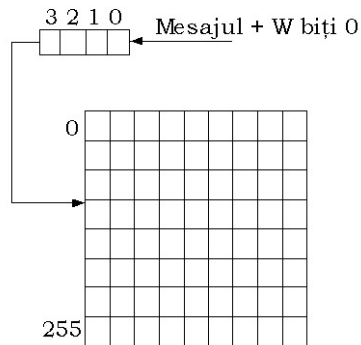


Figura 3: Implementare directă

Algoritm

1. cât timp (mesajul cu zerouri nu este epuizat)
2. octet \leftarrow cel_mai_semnicativ_octet(Registru)
3. Registru \leftarrow (Registru \ll 8) | următorul_octet
4. Registru \leftarrow Registru XOR Tabel[octet]



Name	Value	Sti...
message		C366F955
rc crc		A2BA
message_crc		C366F955A2BA
memory		
memory(0)		0000
memory(1)		1021
memory(2)		2042
memory(3)		3063
memory(4)		4084
memory(5)		50A5
memory(6)		60C6
memory(7)		70E7
memory(8)		8108
memory(9)		9129
memory(10)		A14A
memory(11)		B16B

Figura 4: Implementarea bazată pe tabelă

Coduri Reed-Solomon

Codurile Reed-Solomon (RS) sunt coduri corectoare de erori în bloc inventate în 1960 de Irving Reed și Gustave Solomon. Aceste coduri au început să fie utilizate începând cu 1990, atunci când progresele tehnologice au făcut posibilă trimiterea datelor în cantități mari și la viteze ridicate. Actualmente aceste coduri sunt utilizate într-o gamă largă de echipamente electronice cum sunt:

- dispozitivele pentru stocarea datelor (CD, DVD, hard-disk);
- telefoanele mobile;
- echipamentele folosite în comunicațiile prin satelit;
- televiziunea digitală;
- modemurile de mare viteză (ADSL, xDSL).

Realizarea unei transmisii folosind codurile RS presupune ca, codificatorul RS să preia un bloc de date și să adauge o informație suplimentară caracteristică. Una dintre caracteristicile importante ale codului RS constă în faptul că acest cod va codifica grupuri de simboluri de date.

Decodificatorul RS procesează fiecare bloc și încearcă să corecteze erorile apărute și să recupereze datele trimise original.

Un cod RS este specificat ca $RS(n, k)$ cu simboluri de s biți. Această descriere semnifică faptul că, codificatorul preia k simboluri de paritate astfel încât să rezulte un cuvânt de cod de n simboluri. Sunt $n - k$ simboluri de paritate, de câte s biți fiecare. Un decodificator RS poate corecta până la t simboluri ce conțin erori, cu $2t = n - k$.

Un cod RS este obținut împărțind mesajul original în blocuri de lungime fixă. Fiecare bloc este apoi împărțit în simboluri de m biți. Fiecare simbol are lungime fixă (între 3 și 8 biți). Natura liniară a acestui cod asigură faptul că fiecare cuvânt de m biți este valid pentru codificare astfel încât se pot transmite date binare sau text.

Exemplu Un cod des folosit este $RS(255, 233)$ cu simboluri de 8 biți. Fiecare cuvânt de cod conține 255 de simboluri din care 233 sunt de date și 22 sunt de paritate. Pentru acest cod se pot stabili următoarele relații: $n = 255$, $k = 233$, $s = 8$, $t = 16$.

Codurile RS pot fi scurtate dacă la codificator se fac anumiți biți zero, nu se transmit dar sunt adăugați la decodificator. Spre exemplu, codul $RS(255, 233)$ poate fi scurtat la $(200, 168)$. Operațiile realizate de codificator sunt următoarele:

- se preia un bloc de 168 de biți de date;
- se adaugă virtual 55 de biți de zero creând astfel un cod $(255, 233)$;
- se transmit doar 168 biți de date și 32 biți de paritate.

Un decodificator RS poate corecta un număr de t erori și până la $2t$ ștersături. La decodificarea unui cuvânt RS pot apărea următoarele variante:

- dacă $2s + r < 2t$ atunci codul original transmis poate fi corectat în întregime;
- decodificatorul indică faptul că nu poate reface codul original;
- decodificatorul va genera un cuvânt decodat cu erori și nu va fi semnalat acest lucru.

Arhitectura decodului poate fi urmărită în figura 5.

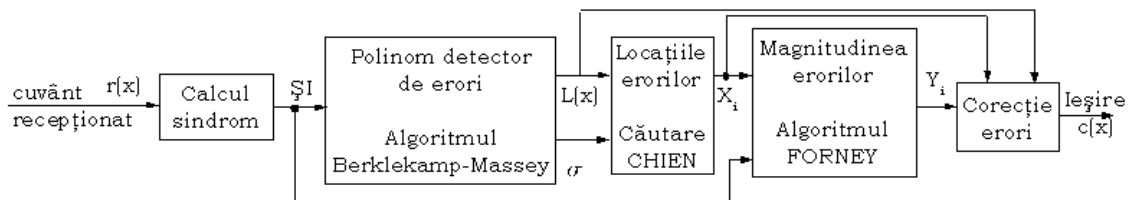


Figura 5: Arhitectura unui decodificator RS.

Algoritmul de criptare RSA

Algoritmul RSA este un sistem criptografic ce utilizează chei publice și a fost creat de un grup de cercetători de la MIT (Massachusetts Institute of Technology) cu scopul de a asigura securitatea datelor schimbate prin intermediul Internet-ului.

Metodele tradiționale de criptare (spre exemplu algoritmul DES - implementările hardware și JAVA precum și simulările acestor implementări pot fi vizualizate la <http://www.csit-sun.pub.ro>) folosesc un număr de $\frac{n \cdot (n-1)}{2}$ chei, în timp ce algoritmi bazați pe chei publice utilizează un număr de cel mult n chei publice.

O altă deosebire constă în faptul că în sistemele tradiționale de criptare, cheia de criptare trebuie ținută secretă deoarece ea trebuie utilizată în cadrul procesului de decriptare. În cazul criptării cu chei publice, cheia de criptare/decriptare nu mai este trimisă receptorului, deci canalul de comunicație dintre transmițător și receptor poate să nu fie securizat.

Utilizarea algoritmului RSA implică crearea a două chei de către transmițător: *una publică* și *una privată*. Cheia publică este trimisă oricărui destinatar la care trebuie trimis mesajul criptat. Cheia privată sau secretă este utilizată pentru decriptarea mesajului criptat cu ajutorul cheii publice.

Modalitatea de realizare a unei comunicații criptate cu ajutorul algoritmului RSA este prezentată în figura 6.

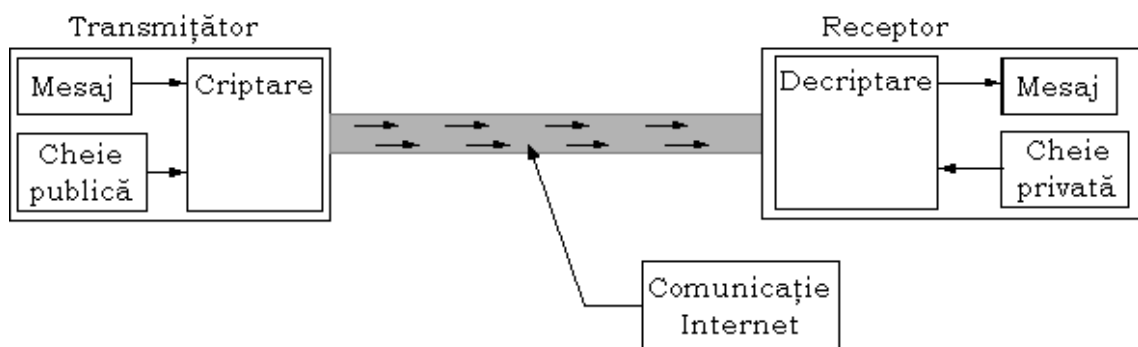


Figura 6: Arhitectura unui decodor RS.

Transmisia folosind algoritmul RSA necesită parcurgerea a două etape importante:

1. Generarea cheilor - se generează două chei una publică și una privată. Pentru aceasta trebuie parcurși următorii pași:

1. se aleg două numere prime p și q cu aceeași magnitudine (lungime) și se generează numărul $n = p \cdot q$;
2. se determină $\Phi = (p - 1) \cdot (q - 1)$;
3. se alege e ca fiind un număr prim în raport cu Φ , deci cel mai mare divizor comun (notat $\text{gcd}(e, \Phi)$) al celor două numere trebuie să fie 1. În implementările practice valoarea lui e este aleasă ca fiind un număr prim Fermat (3, 5, 17, 65537,...);
4. se determină valoarea d care reprezintă inversiunea modulară a lui e și Φ :

$$d = \text{rest}\left(e - \frac{1}{\Phi}\right)$$

Cheia publică este alcătuită din perechea (n, e) , cât timp cheia privată este formată din perechea (n, d) . Implementarea hardware a celui mai mare divizor comun se realizează cu ajutorul algoritmului lui Euclid.

```

Algoritm EuclidExtins(a, b)
  if b = 0 then
    return (a, 1, 0)
  else
    (d', x', y') = EuclidExtins(b, rest( $\frac{a}{b}$ ))
  return (d', y', x' -  $\frac{a}{b} \cdot y'$ )

```

2. Transmisia informației - În cadrul acestei etape, atât transmițătorul cât și receptorul trebuie să execute câteva operații distincte. Transmițătorul realizează următoarele operații:
 - a. obține cheia publică (n, e) de la receptor;
 - b. convertește mesajul într-o mulțime de întregi pozitivi;
 - c. calculează textul criptat conform relației: $c = m^e \text{ mod } n$;
 - d. transmite mesajul c la receptor.

Receptorul realizează următoarele operații:

- a. utilizează cheia privată (n, d) pentru a calcula $m = c^d \text{ mod } n$;
- b. extrage textul din colecția de numere întregi m .

Algoritmul de criptarea IDEA

IDEA este un algoritm bazat pe chei publice care criptează blocuri de câte 64 de biți folosind o cheie de criptare de lungime 128 de biți. Criptarea și decriptarea presupun utilizarea aceluiași algoritm. Implementarea acestui algoritm impune utilizarea a trei operații: *XOR*, *adunarea modulo 65536* și *înmulțirea modulo 65537* care operează pe sub-blocuri de dimensiune 16 biți.

Funcționarea algoritmului constă în parcurgerea a opt pași. Blocul de date de dimensiune 64 de biți este împărțit în 4 părți X_0 , X_1 , X_2 și X_3 , fiecare parte având dimensiunea de 16 biți. În fiecare pas, între cele 4 sub-blocuri se realizează o operație *XOR*, de adunare sau de înmulțire, împreună cu 6 subchei de dimensiune 16 biți fiecare.

Între pașii 2 și 3, sub-blocurile sunt interschimbate, iar în final cele 4 sub-blocuri sunt combinate împreună cu 4 subchei pentru a forma ieșirea. În cadrul fiecărui pas al algoritmului se execută următoarea succesiune de operații:

- se înmulțește X_0 cu prima subcheie;
- se adună X_1 la a doua subcheie;
- se adună X_2 la a treia subcheie;
- se înmulțește X_3 cu a patra subcheie;
- *XOR* între rezultatele pașilor 1 și 3;
- *XOR* între rezultatele pașilor 2 și 4;
- se înmulțește rezultatul pasului 5 cu subcheia numărul 5;
- se adună rezultatele obținute în cadrul pașilor 6 și 7;
- se înmulțește rezultatul de la pasul 8 cu subcheia numărul 6;
- se adună rezultatele obținute la pașii 7 și 9;
- *XOR* între rezultatele pașilor 1 și 9;
- *XOR* între rezultatele pașilor 3 și 9;
- *XOR* între rezultatele pașilor 2 și 10;

- XOR între rezultatele pașilor 4 și 10.

Cele patru rezultate sunt sub-blocurile obținute în urma pașilor 11, 12, 13 și 14. Se inter-schimbă cele două sub-blocuri din mijloc și astfel se obține intrarea pentru următorul pas. Excepție face ultimul pas în care nu se mai execută interschimbarea celor două sub-blocuri din mijloc. După pasul opt se execută următoarea secvență de operații pentru a determina rezultatul final:

- se înmulțește X_0 cu prima subcheie;
- se adună X_7 la a doua subcheie;
- se adună X_2 la a treia subcheie;
- se înmulțește X_3 cu a patra subcheie.

În final cele patru sub-blocuri se vor concatena pentru a forma blocul criptat de lungime 64 de biți.

Algoritmul utilizează 52 de subchei: 6 subchei pentru fiecare pas și 4 subchei pentru pasul final. Generarea subcheilor pornește de la cheia de lungime 128 de biți care se împarte în opt subchei. Acestea reprezintă primele opt subchei utilizate în algoritm. La pasul următor cheia este deplasată la stânga 25 de poziții și apoi împărțită în opt părți. Acest proces de generare a subcheilor este continuat până se generează toate cele 52 de subchei necesare funcționării algoritmului.

Schema generală a algoritmului de criptare *IDEA* este prezentată în figura 7.

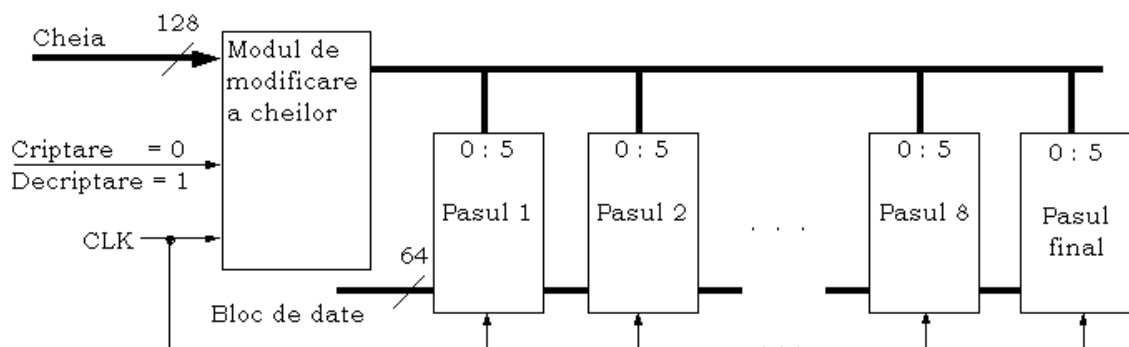


Figura 7: Schema generală a algoritmului de criptare cu chei publice IDEA.

2. Desfășurarea lucrării

Se va proiecta în Verilog utilizând Xilinx WebPACK ISE 10.1 și se va simula un circuit, care implementează algoritmul IDEA. Se va folosi schema generală prezentată în figura 7.

3. Probleme propuse

1. Să se proiecteze în Verilog utilizând Xilinx WebPACK ISE 10.1 și să se simuleze un circuit, care implementează algoritmul CRC bazat pe tabelă.
2. Să se proiecteze în Verilog utilizând Xilinx WebPACK ISE 10.1 și să se simuleze un circuit, care implementează algoritmul de criptare RSA.

Indicații

- Este bine să se calculeze o tabelă de conversie pentru fiecare dintre cele 256 valori de intrare posibile. Pentru a cripta mesajul va fi necesar doar accesul la o memorie locală care memorează tabela determinată. La decriptare se va utiliza același artificiu.
- Pentru a putea implementa în hardware expresia $m^e \bmod n$ se va utiliza următorul algoritm:

```
res = m;
for (i = 2; i <= e; i = i + 1) begin
    res = res * m;
    if (res > m) begin
        res = res % n;
    end
end
end
cypher(m, n, e) = res;
```