CAPITOLUL 3

CODAREA SURSĂ

3.1. CUVINTE DE COD DE LUNGIME FIXĂ

Unele surse de informație generează semnale analogice ce pot fi descrise drept variabile aleatoare de tip continuu. În această clasă intră semnalul vocal generat de un microfon, unda seismică generată de un geofon şi, în general, semnalele generate de diverse traductoare (un traductor fiind un dispozitiv care convertește o mărime neelectrică în una electrică şi invers). Alte surse de informație generează şiruri de simboluri dintr-o mulțime discretă şi finită. De exemplu, o dactilografă bătând la maşina de scris generează un şir de caractere din setul de caractere pentru care sunt prevăzute taste. Acest exemplu pare a nu mai fi de actualitate, căci maşina de scris a fost trimisă la Muzeul de Istorie a Tehnicii prin larga răspândire a calculatoarelor personale. Dar aceeaşi dactilografă, adaptându-se vremurilor, generează de la claviatura unui PC un text ce poate include un set mult mai larg de simboluri, de asemenea însă discret şi finit. Mai mult decât atât, cineva care scrie un text utilizând calculatorul personal generează de fapt un fişier, adică un şir compus numai din două simboluri, 0 şi 1, şir ce poate fi memorat sau transmis la distanță şi pe care o imprimantă îl poate oricând transforma în textul tipărit. De la cursul de „Semnale, circuite şi sisteme“, ştim că, prin eşantionare şi cuantizare, un semnal analogic poate fi transformat într-un semnal digital din care poate fi recuperat cu o distorsiune ce poate fi făcută acceptabil de mică. Prin urmare, indiferent dacă sursa de informație generează un semnal analogic sau unul digital, acest semnal poate fi transformat într-unul echivalent utilizând numai două simboluri, fie ele 0 şi 1. Interesul nostru special pentru această posibilitate,

de a exprima orice informaţie cu numai două simboluri, se explică prin faptul că tranzistorul, care este baza electronicii, funcţionează în regim de comutaţie între două stări, de conducţie şi de blocare, cărora le facem să corespundă cele două simboluri. Remarcabil însă este că această idee a avut-o filosoful englez Francis Bacon (1561–1626) cu circa trei secole şi jumătate înainte de inventarea tranzistorului. El scria: „Un om poate exprima şi comunica intenţiile minţii sale, la orice distanţă prin obiecte... capabile numai de o diferenţă binară".

Recapitulând, orice sursă de informaţie poate fi transformată într-o sursă de simboluri binare. O problemă importantă în memorarea şi în transmiterea digitală a informaţiei este reprezentarea eficientă a datelor generate de sursă. Procesul prin care se realizează această reprezentare se numeşte *codare sursă*. Dispozitivul care efectuează această reprezentare se numeşte *codor sursă*. Dacă variabila aleatoare X este ieşirea unei surse discrete, entropia $H(X)$ a sursei reprezintă cantitatea medie de informaţie emisă de sursă. În acest capitol, vom considera problema codării semnalului de ieşire a sursei, adică, reprezentarea semnalului de ieşire printr-un şir de simboluri binare, numite biţi. O măsură a eficienţei metodei de codare sursă se poate obţine comparând numărul mediu de biţi pe simbol de ieşire dintr-o sursă de entropie $H(X)$.

Un codor sursă eficient trebuie să satisfacă două cerinţe funcţionale:

1. Cuvintele de cod produse de codor trebuie să fie în formă *binară*.
2. Codul sursă să fie *univoc decodabil*, astfel încât şirul sursă originar să poată fi reconstruit perfect din şirul binar codat.

Presupunem că o sursă discretă fără memorie (SDFM) produce un simbol, sau o literă, de ieşire la fiecare T_s secunde. Fiecare simbol este selectat dintr-un alfabet finit de simboluri $x_i, i = 1, 2, \cdots, K$, ce apar cu probabilităţi $P(x_i), i = 1, 2, \cdots, K$. Entropia acestei SDFM în biţi pe simbol sursă este

$$H(X) = -\sum_{i=1}^{K} P(x_i) \log_2 P(x_i) \le \log_2 K \tag{3.1}$$

EXEMPLUL 3.1: La cursul de „Semnale, circuite şi sisteme", se studiază eşantionarea, un proces prin care un semnal analogic este convertit într-un şir corespunzător de eşantioane, care sunt valori ale semnalului analogic prelevate în momente discrete $t_k = kT_e$, unde T_e este perioada de eşantionare, iar k este un număr natural. În cazul foarte important al semnalului vocal, semnalul generat de un microfon este mai întâi filtrat

astfel încât să nu conţină componente de frecvenţă în afara intervalului [300, 3400 Hz] şi este apoi eşantionat cu o frecvenţă $f_e = 1/T_e = 8$ kHz, rezultând astfel un şir de valori indexate după k. Fie D gama dinamică a lui $s(t)$, definită ca diferenţă între valoarea maximă şi valoarea minimă pe care le poate lua $s(t)$: $D = s_{\max}(t) - s_{\min}(t)$. În cazul semnalului vocal, gama dinamică D se împarte în $2^8 = 256$ de nivele. Cuantizarea este procedeul prin care mulţimea infinită a valorilor pe care un eşantion dat de mărime $s(kT_e)$ le poate avea în interiorul gamei dinamice D este redusă la mulţimea discretă şi finită a celor 256 de nivele. În acest caz, alfabetul sursă este alcătuit din cele $K = 256$ de nivele. Fiecărui nivel de eşantionare îi corespunde un cuvânt binar de $\log_2 256 = 8$ biţi.

EXEMPLUL 3.2: Cu ajutorul calculatorului personal, putem introduce de la claviatură mult mai multe semne decât permite o maşină de scris obişnuită. Codul ASCII (de la American National Standard Code for Information Interchange) are 8 biţi pe caracter. Şapte biţi sunt utilizaţi pentru a specifica un caracter, iar al optulea este un bit de paritate, care poate fi *pară* sau *impară*, la alegere. Deci, în acest cod, alfabetul sursă are $2^7 = 128$ caractere.

Numărul mediu de biţi pe simbol emişi de sursă este $H(X)$ dată de (3.1), iar viteza cu care debitează sursa în biţi/s este prin definiţie $H(X)/T_s$.

Un *cod sursă* C pentru o variabilă aleatoare X este o aplicaţie de la domeniul de existenţă al lui X la mulţimea şirurilor de lungime finită de simboluri dintr-un alfabet cu B litere. Cel mai adesea, vom considera alfabetul binar $B = \{0, 1\}$.

Notă. Reamintim de la cursul de Analiză matematică ce înţelegem printr-o *aplicaţie*; aceasta este o *funcţie* pentru care domeniul de definiţie şi cel de existenţă nu sunt neapărat mulţimi numerice, ci şi mulţimi abstracte (de exemplu, mulţimi de evenimente).

Fie C_i cuvântul de cod corespunzător simbolului sursă x_i şi să notăm cu $l(x_i)$ lungimea lui C_i. Prin definiţie, *lungimea medie* L a unui cod sursă C pentru o variabilă aleatoare X având funcţia masă de probabilitate $P(x)$ este dată de

$$L = \sum_{i=1}^{K} P(x_i) l(x_i) \tag{3.2}$$

EXEMPLUL 3.3 (*Codul Morse*): Pictorul american Samuel Finley Breese Morse (1791 – 1872) a inventat un cod relativ eficient pentru alfabetul latin utilizat în limba engleză cu numai patru simboluri: un punct, o linie, un spațiu între litere şi un spațiu între cuvinte. În codul Morse, şiruri scurte reprezintă litere frecvente (de exemplu, un singur punct reprezintă *E*), iar şiruri lungi reprezintă litere ce apar rareori (de exemplu, *Q* se reprezintă prin „linie, linie, punct, linie“).

Să considerăm o schemă de codare bloc în care fiecare simbol din alfabetul sursă se reprezintă printr-un vector binar unic cu L componente. Fiindcă alfabetul sursă are K simboluri, iar K poate fi sau nu o putere a lui 2, distingem două cazuri:

1. $K = 2^k$

Avem deci

$$L = \log_2 K = \log_2 2^k = k \tag{3.3}$$

2. K nu este o putere a lui 2. Avem în acest caz $2^k < K < 2^{k+1}$ şi deci

$$L = k + 1 \tag{3.4}$$

unde k este cel mai mare întreg mai mic decât $\log_2 K$.

Definim eficiența codării pentru o sursă discretă fără memorie (SDFM) drept raportul

$$\eta = \frac{H(X)}{L}. \tag{3.5}$$

Se vede că, dacă mărimea alfabetului K este o putere a lui 2 iar literele debitate de sursă sunt egal probabile, $L = H(X)$ şi eficiența este maximă: $\eta = 1$. Dacă, însă, K nu este o putere a lui 2, dar simbolurile sursă sunt egal probabile, L diferă de $H(X)$ cu cel mult 1 bit pe simbol. Pentru $\log_2 K \gg 1$, eficiența acestei scheme de codare este ridicată. Pe de altă parte, dacă alfabetul are un număr K mic de simboluri, se poate mări eficiența unui cod de lungime fixă codând împreună un şir de J simboluri. Codul sursă constă din vectori binari cu N componente, ce permit codarea a 2^N şiruri de J simboluri dintr-un alfabet cu K litere. Condiția pentru N se scrie:

$$N \geq J \log_2 K \tag{3.6}$$

Prin urmare, valoarea întreagă minimă a lui N este

$$N = J \log_2 2^k + 1 = Jk + 1 \tag{3.7}$$

Numărul mediu de biți pe simbol sursă este acum $N/J = k + 1/J$ şi astfel ineficiența s-a redus cu aproximativ un factor de $1/J$ în raport cu schema de codare simbol cu simbol descrisă mai înainte. Făcând *J* suficient de mare, eficiența procedeului de codare, măsurată în raportul $JH(X)/N$, poate fi făcută oricât de apropiată de 1 se doreşte.

Metodele de codare descrise mai sus nu introduc distorsiune întrucât codarea simbolurilor sursă sau a blocurilor de simboluri de cuvinte de cod este univocă. Acest tip de codare se numeşte *fără zgomot.*

Să presupunem că încercăm să reducem lungimea cuvântului de cod *L* slăbind condiția ca procesul de codare să fie univoc. În acest scop, selectăm cele mai probabile $2^N - 1$ blocuri de câte *J* simboluri pe care le codăm univoc, urmând ca celelalte $K^J - (2^N - 1)$ blocuri de câte *J* simboluri să se reprezinte prin singurul cuvânt de cod rămas nefolosit. La recepție, acest procedeu are drept rezultat o decodare greşită, ce intervine cu o anumită probabilitate de eroare, de fiecare dată când un bloc de probabilitate scăzută este pus în corespondență cu acel unic cuvânt de cod rămas nefolosit la codarea blocurilor mai probabile. Să notăm cu P_e această probabilitate de eroare. Pe baza acestui procedeu de codare bloc, Shannon a demonstrat în anul 1948 următoarea teoremă de codare sursă.

Teorema codării sursă I

Fie *X* variabila aleatoare reprezentând simbolurile emise de o SDFM având entropie finită $H(X)$. Blocuri de câte *J* simboluri debitate de sursă se codează în cuvinte de cod de lungime *N* dintr-un alfabet binar. Pentru orice $\varepsilon > 0$, probabilitatea P_e a unei decodări eronate a unui bloc poate fi făcută arbitrar de mică dacă

$$L \equiv \frac{N}{J} \geq H(X) + \varepsilon \tag{3.8}$$

iar *J* este suficient de mare. Reciproc, dacă

$$L \leq H(X) - \varepsilon, \tag{3.9}$$

P_e devine arbitrar de apropiată de 1 dacă *J* este suficient de mare.

Din această teoremă, rezultă că numărul mediu de biți pe simbol necesari pentru a coda ieşirea unei SDFM cu o probabilitate de decodare

eronată arbitrar de mică este mărginit inferior de entropia sursei. Pe de altă parte, dacă $L < X(X)$, frecvența unei decodări eronate se apropie de 100% atunci când J creşte arbitrar.

3.2 CUVINTE DE COD DE LUNGIME VARIABILĂ

Dacă simbolurile emise de sursă nu sunt egal probabile, o metodă mai eficientă de codare este de a utiliza cuvinte de cod de lungime variabilă. Codul Morse menționat în **Exemplul 3.3** este cea mai veche ilustrare a acestei idei. Acest tip de codare se numeşte *codare entropică*.

Pentru a vedea ce probleme ridică o codare sursă în care cuvintele de cod au lungime variabilă în funcție de probabilitatea simbolurilor sursă, să considerăm exemplul unei SDFM cu litere de ieşire a_1, a_2, a_3, a_4 şi probabilități corespunzătoare $P(a_1) = \frac{1}{2}$, $P(a_2) = \frac{1}{4}$ şi $P(a_3) = P(a_4) = \frac{1}{8}$. În Tabelul 3.1 se arată trei variante de codare.

Tabelul 3.1

Coduri de lungime variabilă

Literă	$P(a_k)$	Cod I	Cod II	Cod III
a_1	$\frac{1}{2}$	1	0	0
a_2	$\frac{1}{4}$	00	10	01
a_3	$\frac{1}{8}$	01	110	011
a_4	$\frac{1}{8}$	10	111	111

Codul I are un „cusur“ fundamental. Să presupunem că recepționăm şirul $001001\cdots$. Primii doi biți 00 corespund în mod clar simbolului a_2, însă următorii patru biți prezintă ambiguitate, căci nu sunt univoc decodabili: ei pot fi decodați fie ca a_4a_3, fie ca $a_1a_2a_1$. Este posibil ca această ambiguitate să fie rezolvată aşteptând să ajungă la receptor biți

suplimentari, dar o astfel de întârziere de decodare este nedorită. De aceea, vom considera numai coduri care sunt decodabile *instantaneu*, adică, fără nici o întârziere de decodare.

Codul II din Tabelul 3.1 este *univoc decodabil* şi *decodabil instantaneu*. Este convenabil să reprezentăm cuvintele de cod grafic drept *frunze* (noduri terminale) ale unui *arbore*, după cum se arată în figura 3.1.

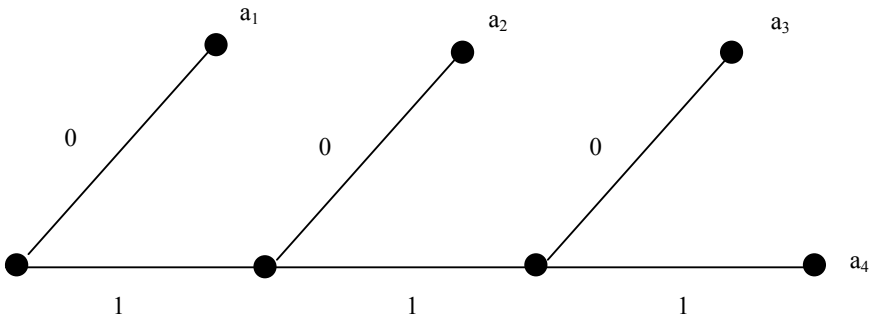

Fig. 3.1. Arbore de cod pentru codul II din Tabelul 3.1.

Se observă că bitul 0 indică sfârşitul unui cuvânt de cod pentru primele trei cuvinte de cod şi că nici un cuvânt de cod nu este prefixul vreunui alt cuvânt de cod. În general, *condiția de prefix* cere ca, pentru un cuvânt de cod C_i de lungime i având componentele $(b_1, b_2, \cdots, b_i)$, să nu existe nici un alt cuvânt de cod de lungime $j < i$ cu componente $(b_1, b_2, \cdots, b_j)$ pentru $1 \le j \le i-1$. Cu alte cuvinte, să nu existe nici un cuvânt de cod de lungime $j < i$ care să fie identic cu primii j biți ai altui cuvânt de cod de lungime $i > j$. Această proprietate face ca un cod să fie decodabil instantaneu.

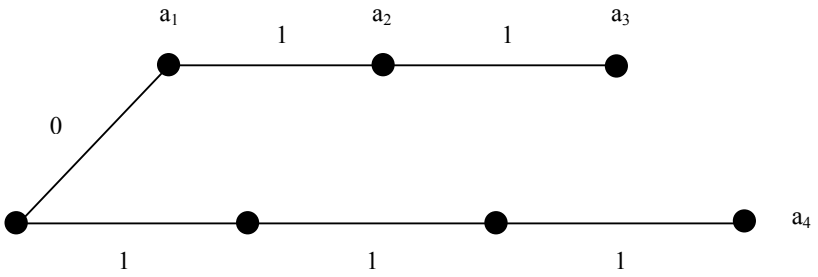

Fig. 3.2. Arbore de cod pentru codul III din Tabelul 3.1.

Codul III dat în Tabelul 3.1 are graful arbore arătat în figura 3.2. Se vede că este decodabil univoc dar nu şi instantaneu. Este clar că acest cod nu satisface condiția de prefix.

Principalul nostru obiectiv este de a imagina o procedură sistematică pentru construcția codurilor de lungime variabilă decodabile univoc care să fie eficiente în sensul că lungimea medie (3.2) în biți pe literă debitată de sursă să fie minimizată. Condițiile pentru existența unui cod ce satisface condiția de prefix sunt date de teorema următoare.

Inegalitatea lui Kraft-McMillan

O condiție necesară şi suficientă pentru existența unui cod binar instantaneu (adică, ce îndeplineşte condiția de prefix) este ca lungimile cuvintelor de cod $l_1 \le l_2 \le \cdots \le l_K$ să satisfacă inegalitatea

$$\sum_{i=1}^{K} 2^{-l_i} \le 1 \tag{3.10}$$

DEMONSTRAȚIE

Fie l mai mare decât oricare dintre lungimile l_i, în biți. Construim un arbore binar complet de ordin l care are 2^l noduri terminale (frunze) şi două noduri de ordin i provenind din fiecare nod de ordin i–1, pentru orice i, $1 \le i \le l$. Ramurile arborelui reprezintă simbolurile binare ale cuvântului de cod. Spre exemplu, cele două ramuri ce ies din nodul rădăcină reprezintă cele două valori posibile ale primului bit. Fiecare cuvânt de cod este reprezentat de o frunză a arborelui. Drumul de la rădăcină la frunză ne dă biții ce compun respectivul cuvânt de cod. Ilustrăm aceasta în figura 3.3 pentru un arbore având 16 noduri terminale şi o sursă care generează cinci litere cu $l_1 = 1, l_2 = 2, l_3 = 3$ şi $l_4 = l_5 = 4$.

Condiția de prefix impusă cuvintelor de cod face ca nici un cuvânt de cod să nu fie strămoşul vreunui alt cuvânt de cod din arbore. De aceea, fiecare cuvânt de cod îşi elimină descendenții ca posibile cuvinte de cod.

Să considerăm toate cele 2^l noduri ale arborelui de la nivelul l. Unele dintre ele sunt cuvinte de cod, altele sunt descendenți ai unor cuvinte de cod, iar celelalte nu sunt nici una, nici alta. Dacă una din primele două ramuri ce ies din rădăcină (adică, i = 1) formează ea singură un cuvânt de cod ($l_1 = 1$), jumătate din ramurile arborelui, adică 2^{l-1}, sunt eliminate.

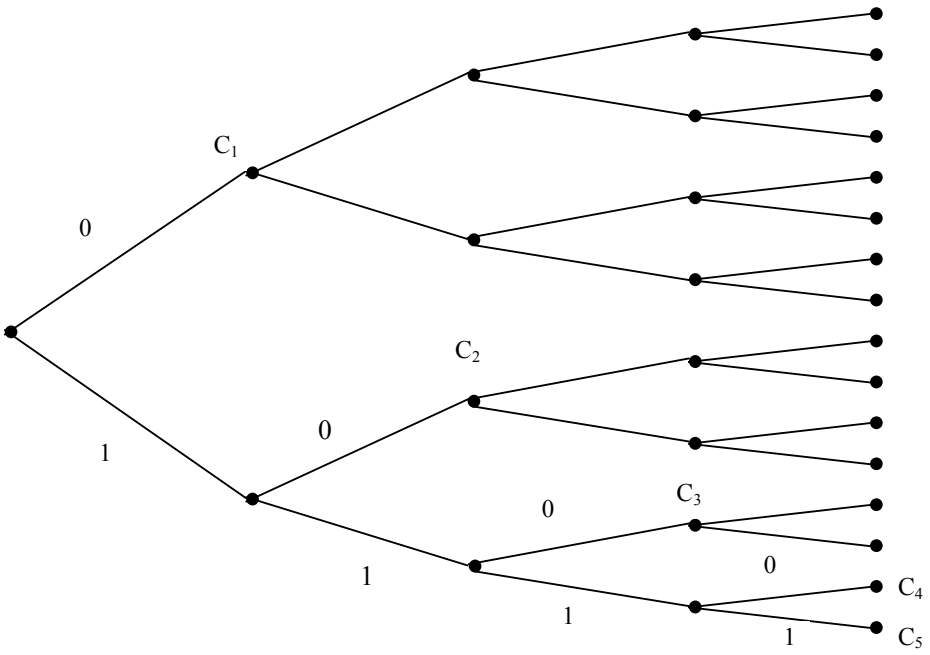

Fig. 3.3. Construcția unui cod arbore binar cuprins într-un arbore complet.

Conform aceluiaşi raționament, un cuvânt de cod de la nivelul l_i are 2^{l-l_i} descendenți la nivelul maxim l. Aceste mulțimi de descendenți trebuie să fie disjuncte. De asemenea, numărul total al nodurilor din aceste mulțimi trebuie să fie mai mic sau egal cu 2^l. Prin urmare, sumând peste toate cuvintele de cod, obținem

$$\sum_{i=1}^{K} 2^{l-l_i} \le 2^l \tag{3.11}$$

din care, prin împărțire la 2^l, rezultă inegalitatea lui Kraft-McMillan (3.10).

Reciproc, dacă se dă orice mulțime de lungimi ale cuvintelor de cod $l_1, l_2, \cdots, l_K$ ce satisfac inegalitatea lui Kraft-McMillan, putem întotdeauna construi un arbore asemenea celui din figura 3.3. Etichetăm primul nod de adâncime l_1 drept cuvântul de cod C_1 şi eliminăm descendenții săi din arbore. Apoi etichetăm primul nod rămas de adâncime l_2 drept cuvântul de

cod C_2, etc. Procedând în acest mod, construim un cod prefix cu lungimile specificate $l_1, l_2, \cdots, l_K$.

Teorema codării sursă II

Fie X variabila aleatoare reprezentând simbolurile $x_i, 1 \le i \le K,$ emise de o SDFM cu entropie finită $H(X)$ cu probabilităţile corespunzătoare de apariţie $p_i = P(x_i), 1 \le i \le K.$ Este posibil să construim un cod ce satisface condiţia de prefix şi are o lungime medie L ce satisface inegalităţile

$$H(X) \le L < H(X) + 1 \tag{3.12}$$

DEMONSTRAŢIE

Vom aproxima distribuţia $p_i, 1 \le i \le K$ cu distribuţia

$$r_i = \frac{2^{-l_i}}{c}, \quad 1 \le i \le K \tag{3.13}$$

unde

$$c = \sum_{i=1}^{K} 2^{-l_i} \tag{3.14}$$

Datorită inegalităţii lui Kraft-McMillan, $c \le 1$. Scriem diferenţa dintre lungime medie şi entropie astfel:

$$\begin{aligned} L - H(X) &= \sum_{i=1}^{K} p_i l_i - \sum_{i=1}^{K} p_i \log_2 \frac{1}{p_i} \\ &= \sum_{i=1}^{K} p_i \log_2 p_i - \sum_{i=1}^{K} p_i \log_2 2^{-l_i} \\ &= \sum_{i=1}^{K} p_i \log_2 \frac{p_i}{r_i} - \log_2 c \\ &= D(p \,\|\, r) + \log_2 \frac{1}{c} \\ &\ge 0 \end{aligned} \tag{3.15}$$

din cauză că entropia relativă este nenegativă iar $c \le 1$. Rezultă că $L \ge H(X)$ cu egalitate dacă şi numai dacă $p_i = 2^{-l_i}$, adică, dacă şi numai dacă $-\log_2 p_i$ este un întreg pentru toţi i.

Alegerea lungimilor $l_i = \log_2 \frac{1}{p_i}$ ne dă deci $L = H(X)$. Dar fiindcă $\log_2 \frac{1}{p_i}$ s-ar putea să nu fie un întreg, îl rotunjim prin majorare pentru a obține lungimi întregi ale cuvintelor de cod:

$$l_i = \left\lceil \log_2 \left(\frac{1}{p_i} \right) \right\rceil, \tag{3.16}$$

unde $\lceil x \rceil$ este cel mai mic întreg $\geq x$. Aceste lungimi satisfac inegalitatea lui Kraft-McMillan deoarece

$$\sum_{i=1}^{K} 2^{-\left\lceil \log_2 \frac{1}{p_i} \right\rceil} \leq \sum_{i=1}^{K} 2^{-\log_2 \frac{1}{p_i}} = \sum_{i=1}^{K} p_i = 1. \tag{3.17}$$

Această alegere a lungimilor cuvintelor de cod satisface

$$\log_2 \frac{1}{p_i} \leq l_i < \log_2 \frac{1}{p_i} + 1. \tag{3.18}$$

Înmulțind cu p_i şi sumând peste *i*, obținem (3.12).

3.3. CODURI HUFFMAN

În anul 1952, Huffman a conceput un algoritm de codare cu lungime variabilă, optim în sensul că numărul mediu de biți necesari pentru a reprezenta simbolurile sursă este minim astfel încât cuvintele de cod să satisfacă acea condiție de prefix descrisă mai sus care permite ca şirul recepționat să fie decodabil univoc şi instantaneu. Codarea Huffman este cel mai uşor de înțeles prin câteva exemple. Fie o sursă discretă fără memorie (SDFM) cu probabilitățile literelor $p_i = P(x_i), i = 1, 2, \cdots, K$.

EXEMPLUL 3.4: Considerăm o SDFM ce debitează şapte simboluri posibile $x_1, x_2, \cdots, x_7$ cu probabilitățile ilustrate în figura 3.4.

Se ordonează simbolurile sursă în ordine descrescătoare a probabilităților: $P(x_1) > P(x_2) > \cdots > P(x_7)$. Se începe procesul de codare cu cele două

simboluri care sunt cel mai puțin probabile x_6 şi x_7. Cele două simboluri sunt frunzele unui arbore; ramurile terminale respective se întâlnesc într-un nod în care simbolurile x_6 şi x_7 se combină într-un simbol, să spunem, x_6'. Următorul pas este de a uni cele două simboluri mai puțin probabile din mulțimea $x_1, x_2, x_3, x_4, x_5, x_6'$. Acestea sunt x_5 şi x_6', care au împreună probabilitatea de 0,05. Atribuim ramurii de la x_5 un 0, iar ramurii de la x_6' un 1. Această procedură continuă până când epuizăm mulțimea literelor sursă. Rezultatul este un arbore de cod ale cărui ramuri conțin cuvintele de cod dorite. Cuvintele de cod se obțin începând de la nodul cel mai din dreapta şi mergând spre stânga. Cuvintele de cod sunt listate în Tabelul 3.2.

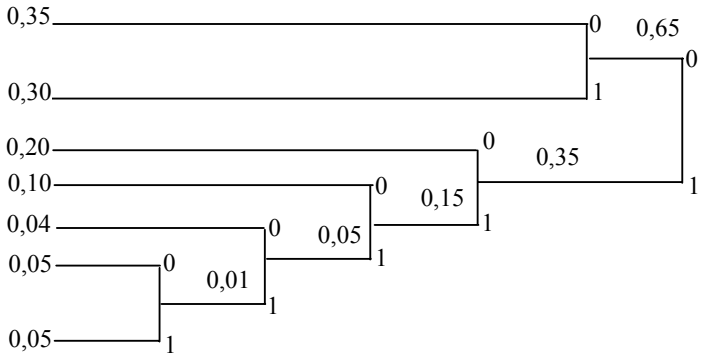

Fig. 3.4 . Exemplu de codare sursă pentru o SDFM cu lungime variabilă a cuvintelor.

Tabelul 3.2

Codul Huffman corespunzător figurii 3.4

Literă	Probabilitate	Autoinformație	Cod
x_1	0,35	1,5146	00
x_2	0,30	1,7370	01
x_3	0,20	2,3219	10
x_4	0,10	3,3219	110
x_5	0,04	4,6439	1110
x_6	0,005	7,6439	11110
x_7	0,005	7,6439	11111

Pentru acest cod, $H(X) = 2{,}11 \quad L = 2{,}21.$

Acest cod nu este în mod necesar unic. Spre exemplu, la penultimul pas din procedura de codare, avem egalitate între x_1 şi x_3', căci aceste simboluri sunt egal probabile. În acest punct, facem alegerea de a împerechea x_1 cu x_2. O alternativă este de a împerechea x_2 cu x_3'. Codul ce rezultă dacă facem această alegere este ilustrat în figura 3.5. Numărul mediu de biţi pe simbol sursă pentru acest cod este tot 2,21. Codurile sunt, deci, la fel de eficiente. În al doilea rând, atribuirea unui 0 ramurii superioare şi a unui 1 ramurii inferioare (mai puţin probabile) este arbitrară. N-avem decât să inversăm atribuirea lui 0 şi a lui 1 pentru a obţine un cod eficient care îndeplineşte condiţia de prefix.

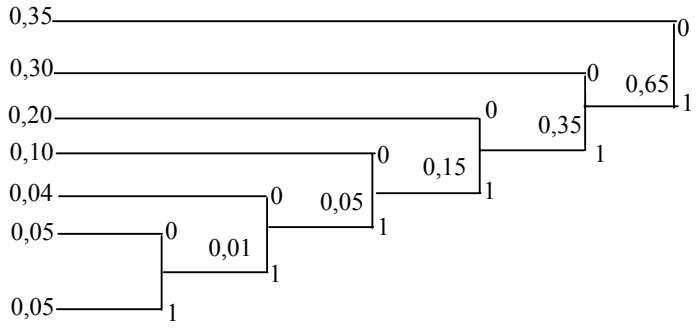

Fig. 3.5. Un alt cod, echivalent, pentru Exemplul 3.4.

Tabelul 3.3

Codul Huffman corespunzător figurii 3.5.

Literă	Cod
x_1	0
x_2	10
x_3	110
x_4	1110
x_5	11110
x_6	111110
x_7	111111

Lungimea medie a acestei variante de cod este tot $L = 2{,}21$.

EXEMPLUL 3.5: O sursă discretă fără memorie (SDFM) are un alfabet de ieşire cu opt litere ale căror probabilităţi sunt date în figura 3.6. Entropia acestei surse este $H(X) = 2{,}63$ biţi pe simbol. Codul Huffman ilustrat în figura 3.6 are o lungime medie de $L = 2{,}70$ biţi pe simbol, ceea ce înseamnă o eficienţă de 0,97.

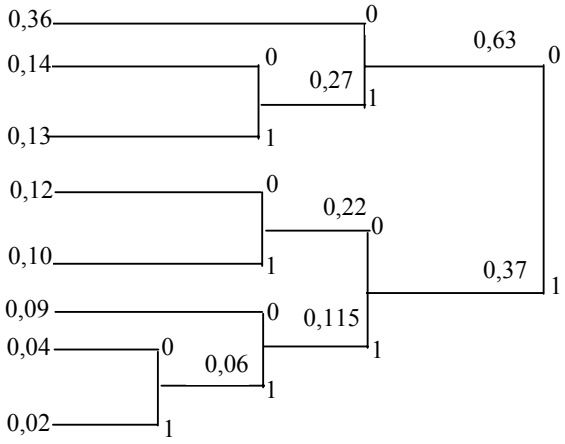

Fig. 3.6. Codul Huffman pentru Exemplul 3.5.

Rezultă codul Huffman dat în Tabelul 3.4.

Tabelul 3.4

Codul Huffman corespunzător figurii 3.6

Literă	Cod
x_1	00
x_2	010
x_3	011
x_4	100
x_5	101
x_6	110
x_7	1110
x_8	1111

Algoritmul Huffman descris în Exemplele 3.4 şi 3.5, codând simbol cu simbol, generează un cod prefix având o lungime medie L ce satisface inegalitatea (3.12). O procedură mai eficientă este de a coda blocuri de J simboluri considerate ca „supersimboluri“, ceea ce face ca lungimea medie L_J să satisfacă inegalitatea

$$JH(X) \le L_J < JX(X) + 1. \tag{3.19}$$

Împărțind la J, obținem

$$H(X) \le \frac{L_J}{J} < H(X) + \frac{1}{J} \tag{3.20}$$

Numărul mediu de biți pe simbol sursă este

$$L = \frac{L_J}{J} \tag{3.21}$$

Luând J suficient de mare, se vede că putem face ca lungimea medie L să fie oricât de apropiată de entropia sursei $H(X)$.

EXEMPLUL 3.6: O SDFM are alfabetul de ieşire compus din trei litere x_1, x_2 şi x_3 pe care le generează cu probabilitățile date în Tabelul 3.5. Entropia sursei este $H(X) = 1{,}518$. Codul Huffman dat în acest tabel necesită $L = 1{,}55$ biți pe simbol, ceea ce reprezintă o eficiență de 97,9%.

Tabelul 3.5

Cod Huffman pentru Exemplul 3.6

Literă	Probabilitate	Autoinformație	Cod
x_1	0,45	1,156	1
x_2	0,35	1,520	00
x_3	0,20	2,330	01

Să codăm acum perechi de litere $(J = 2)$. Entropia variabilei aleatoare care ia ca valori perechi de litere este $2H(X) = 3{,}036$ biți pe perechi de simboluri. Codul Huffman necesită 3,0675 biți pe pereche de simboluri. Eficiența codării creşte deci la $2H(X)/L_2 = 0{,}990$, adică, la 99%.

Tabelul 3.6

Cod Huffman pentru codarea perechilor de litere

Literă	Probabilitate	Autoinformație	Cod
x_1x_1	0,2025	2,312	10
x_1x_2	0,1575	2,675	001
x_2x_1	0,1575	2,675	010
x_2x_2	0,1225	3,039	011
x_1x_3	0,09	3,486	111
x_3x_1	0,09	3,486	0000
x_2x_3	0,07	3,850	0001
x_3x_2	0,07	3,850	1100
x_3x_3	0,04	4,660	1101

3.4. ALGORITMUL DE CODARE LEMPEL-ZIV

Deşi algoritmul de codare Huffman este optim în sensul că este satisfăcută condiția de prefix iar lungimea medie a blocurilor este minimă, el are neajunsul că necesită cunoaşterea distribuției de probabilitate a alfabetului sursei. În practică, statisticile surselor nu se cunosc întotdeauna *a priori*. Acesta este şi cazul când transmitem un mesaj prin Internet, generând un lung şir de litere din alfabetul latin şi de alte semne alfanumerice.

Algoritmul de codare Lempel-Ziv este independent de statisticile sursei. El utilizează cuvinte de cod de lungime fixă pentru a reprezenta un număr variabil de simboluri sursă. În practică, se utilizează blocuri fixe având lungimea de 12 biți, ceea ce permite alcătuirea unui dicționar, numit şi *carte de cod*, cu 4096 de elemente. Ideea de bază este de a împărți datele emise de sursă în segmente care sunt cele mai scurte subşiruri neîntâlnite încă. Se presupune că simbolurile binare 0 şi 1 sunt deja memorate în această ordine în cartea de cod. În scop ilustrativ, vom considera blocuri mult mai scurte, de numai 5 biți.

EXEMPLUL 3.7: O sursă generează şirul binar 1010110100100111010 1000011001110101100011011 0...
Înainte de începerea codării, scriem

Subşiruri codate: 0, 1

Date de segmentat: 1010110100100111010100001100111010110001101 10...

Procesul de codare începe de la stânga. Cu simbolurile 0 şi 1 deja memorate, *cel mai scurt subşir* întâlnit pentru prima oară este 10, astfel încât avem

Subşiruri codate: 0, 1, 10

Date de segmentat:1011010010011101010000110011101011000110110... Cel de al doilea subşir cel mai scurt care n-a mai apărut este 101; scriem deci

Subşiruri codate: 0, 1, 10, 101

Date de segmentat:1010010011101010000110011101011000110110... Următorul subşir cel mai scurt care n-a mai apărut este 1010; deci, scriem

Subşiruri codate: 0, 1, 10, 101, 1010

Date de segmentat:0100111010100001100111010110001101 10...

Continuăm în modul descris mai sus până când şirul de date va fi fost segmentat complet. Rezultă *cartea de cod* dată în Tabelul 3.7.

Prima coloană din Tabelul 3.7 indică poziţiile numerice ale subşirurilor din cartea de cod. Ultimul simbol din fiecare subşir din cartea de cod este un *simbol de inovaţie*, denumit astfel întrucât anexarea sa la un subşir particular îl deosebeşte de toate subşirurile memorate anterior în cartea de cod. De aceea, ultimul bit din cuvântul de cod reprezintă simbolul de inovaţie pentru respectivul subşir. Restul biţilor reprezintă în binar „pointerul“ către *subşirul rădăcină* ce corespunde respectivului subşir cu excepţia simbolului de inovaţie. Primul subşir din trenul de date, 10, este compus prin concatenarea celui de al doilea element din cartea de cod, numerotat 1 şi având valoarea binară 1, cu simbolul de inovaţie 0; de aceea, el este reprezentat prin numărul 10. Cel de al doilea subşir din trenul de date, 101, constă din elementul numerotat 2 şi din simbolul de inovaţie 1, aşa încât este reprezentat prin numărul 21. Celelalte subşiruri se codează similar. De exemplu, subşirul numerotat 9 este compus prin concatenarea subşirului 6 cu simbolul de inovaţie 0, astfel încât se scrie 60.

Tabelul 3.7

Carte de cod pentru algoritmul Lempel-Ziv

Poziție numerică	Subșir	Reprezentare numerică	Cuvânt de cod
0	0		
1	1		
2	10	10	00010
3	101	21	00101
4	1010	30	00110
5	01	01	00001
6	00	00	00000
7	11	11	00011
8	10101	41	01001
9	000	60	01100
10	011	51	01011
11	001	61	01101
12	110	70	01110
13	1011	31	00111
14	0001	91	10011
15	10110	130	11010

Decodarea este la fel de simplă. Se utilizează pointerul pentru a identifica subșirul rădăcină și se adaugă apoi simbolul de inovație. Fie, spre exemplu, cuvântul de cod 11010 din poziția 15. Ultimul bit, 0, este bitul de inovație. Restul de biți, 1101, indică subșirul rădăcină 1011 de la poziția 13. Prin urmare, cuvântul de cod 11010 este decodat drept 10110, ceea ce este corect.

Se observă că algoritmul a codat 45 biți sursă în 14 cuvinte de cod de câte 5 biți fiecare, cu un total de 70 de biți. S-ar spune că nu s-a realizat nici o compresie a datelor, ba dimpotrivă. Această ineficiență se explică prin faptul că am considerat un șir foarte scurt. Dacă se mărește lungimea șirului de date, eficiența crește. În cazul unui text obișnuit în limba engleză, codarea Huffman realizează o compactare de aproximativ 43%, în vreme ce algoritmul Lempel-Ziv asigură o compactare de aproximativ 55%.

Desigur, codorul și decodorul trebuie să lucreze în sincronism, astfel încât, după completarea elementelor din *cartea de cod*, a cărei capacitate este mare dar finită, algoritmul să o ia de la început pentru un nou șir de date.

Să nu scăpăm însă din vedere că între codor și decodor este o distanță fizică, iar transmisia informației între ele se face pe un canal de comunicație. Modul în care se efectează această transmisie se prezintă în capitolul următor.