Universele bronkodering

Tjalkens, T.J.

Award date:
1983

UNIVERSELE BRONKODERING.

door

Tjalling Tjalkens.

Verslag van het afstudeerwerk
uitgevoerd bij de vakgroep
Informatie- en Communicatietheorie.
T.H. Eindhoven, oktober 1983.

Begeleiders: Prof. dr. ir. J.P.M. Schalkwijk.
Ir. J.A.M. de Brouwer.

Samenvatting.

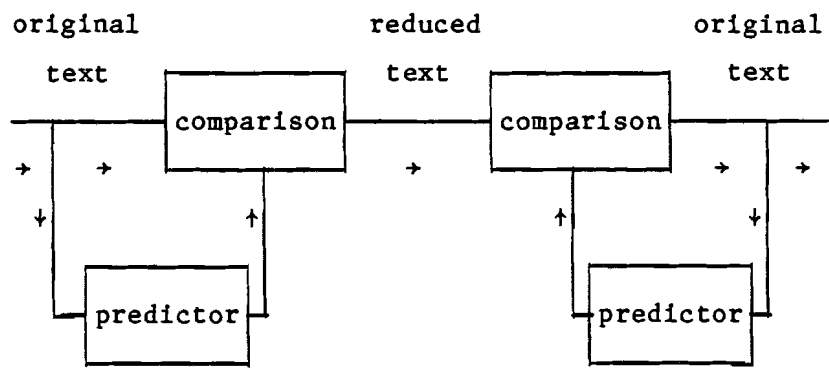
De afstudeeropdracht luidde: het bestuderen van Rissanen's universele data kompressie systeem. Deze opdracht is uitgegroeid tot een algemenere bestudering van de universele data kompressie.
In dit verslag bevinden zich beschrijvingen van Davisson's fundamentele resultaten en Rissanen en Langdon's aanpak van de universele kodering. Ook Tunstall's optimale bron kodes, vergelijkbaar met het Huffman schema, zijn weergegeven. Tevens zijn verschillende beschrijvingen van Ziv en Lempel's en Rissanen's algoritme gegeven. Verder is hier een eerste poging ondernomen de komplexiteit van dit type algoritmes te formuleren en te beperken.

De resultaten van dit werk behelzen, naast de verduidelijking van bovengenoemde beschrijvingen, een vergelijking van Rissanen's algoritme en dat van Ziv-Lempel. Ook is een ander bewijs voor Shannon's bronkoderingsstelling voor geheugenloze bronnen gegeven.

Inhoudsopgave.

1.0 Inleiding.

Datakompressie heeft tot doel het reduceren van de gemiddelde 'bitrate' voor transmissie of opslag van een datastroom. Dit geschiedt door het verwijderen van redundante data en/of overbodige precisie. In het bijzondere geval van foutloze kompressie wordt geëist dat de gereproduceerde datastroom identiek is aan het origineel, d.w.z. er is geen overbodige precisie.
Een mathematische formulering van dit probleem is gegeven door Shannon [1948] en staat nu bekend als de bronkodering. Shannon's aanpak verlangt echter het bekend zijn van de statistische eigenschappen van het datagenererend proces. Daar aan deze eis in de praktijk niet vaak voldaan is, wordt gezocht naar z.g.n. universele kompressie-algoritmen, welke goed werken voor minder precies beschreven bronnen.
Volgens Davisson [1973] is een universele kode een rij van kodes voor een bron met onbekende parameter(s), zodanig dat de rij van kodes kompressiefaktoren oplevert, die naderen tot de haalbare kompressie in het geval dat de parameters wel bekend zijn.
Ook voor dit probleem zijn veel van de fundamenten aangedragen door Shannon [1951]. In dit artikel introduceert hij het volgende kommunikatie systeem, figuur 1.1.



Figuur 1.1.

Vooral in de ideeën van Rissanen en Langdon [1981-a] vinden we veel van dit systeem terug.

De afstudeer opdracht behelsde het bestuderen van een universeel bronkoderings algoritme volgens Rissanen [1983]. Het uiteindelijke resultaat is uitgegroeid tot een algemener overzicht van de universele bronkodering.
Dit overzicht begint met hoofdstuk 2, waarin Davisson's fundamentele definities uit [1973] behandeld worden samen met Rissanen en Langdon's [1981-a] aanpak. Ook wordt hierin een zeer krachtig algoritme volgens Ziv en Lempel [1978] behandeld. Het voorgenoemde algoritme van Rissanen, dat eigenlijk ook in dit overzichts hoofdstuk past, is apart in hoofdstuk 5 gezet, mede omdat na hoofdstuk 4 een beter vergelijk tussen Rissanen's en Ziv en Lempel's algoritme mogelijk is.
In hoofdstuk 3 is één van Davisson's resultaten uitgewerkt. Hieruit worden enkele fundamentele beperkingen van universele kodes voor eindige orde Markov bronnen verduidelijkt.
In hoofdstuk 4 wordt een introduktie gegeven in de beschrijving van bronkodes m.b.v. bomen. De hier behandelde kodes zijn van het segment type, wat wil zeggen dat ze de bronrij in disjunkte delen splitsen en apart koderen. Na de introduktie volgt de beschrijving van een optimale procedure voor het genereren van variabele naar vaste lengte kodes volgens Tunstall [1967] en Verhoeff [1977]. Hierin wordt ook een, minder geslaagde, poging ondernomen om dit principe op bronnen met geheugen toe te passen. Dan wordt het Ziv-Lempel algoritme nogmaals bekeken en wordt de achterliggende modelvorming van dit systeem duidelijk. Deze beschrijving is afkomstig van Langdon [1983]. Ook wordt een beschrijving als 'segment achtige' kode gegeven. Dit laatste is een uitwerking van wat Rissanen in [1983] vermeldt. Het laatste deel van hoofdstuk 4 borduurt voort op de voorgenoemde beschrijving van het Ziv-Lempel algoritme en beschrijft enkele pogingen tot het vinden van universele kodes, voor stationaire bronnen, met begrenste komplexteit. De resultaten van enkele simulaties worden hierin ook beschreven.

2.0 Universele datakompressie.

Er zijn verschillende manieren om het probleem van de universele datakompressie aan te pakken. In dit hoofdstuk worden enkele hiervan behandeld.

2.1 Davisson's aanpak.

Davisson [1973, 1975] modelleert een bron met onbekende of onvolledig bekende parameters als een klasse van bronnen waaruit het toeval één bepaalde bron kiest. Alleen de bronuitkomsten, de datastroom, wordt bekend gemaakt en niet de keuze van de bron. Deze bronnen noemt men 'composite sources', zie b.v. Berger [1971].
De grootte en vorm van deze klasse wordt bepaald door onze, onvolledige, kennis van de mogelijke bronparameters. Enkele voorbeelden hiervan zijn: de klasse van alle stationaire bronnen met een gegeven alfabet (We weten zo goed als niets van de te koderen bron) en de klasse van alle Bernoulli processen. (We weten dat de bronuitkomsten onderling onafhankelijke en gelijk verdeelde stochastische variabelen zijn, echter de bias is onbekend.)

Het model voor deze klasse is het volgende:

> Een bronindex Θ, welke waarden aanneemt uit een verzameling Λ, definieert voor elke $\theta \in \Lambda$ een diskrete tijd bron $X_{-\infty}^{\infty}$ met alfabet A en een statistische beschrijving P_θ zodat voor alle $n \in \mathbb{N}_1$ $p(x_1^n \mid \theta)$ bekend is.

De variabele Θ kan wel of niet beschouwd worden als een stochastische variabele met verdeling $W(\theta)$, afhankelijk van een eventuele voorkennis over de klasse Λ van bronnen.

De hier beschouwde bronnen worden konditioneel stationair en ergodisch verondersteld. Dit wil zeggen dat voor elke $\theta \varepsilon \Lambda$ de door θ geïndiceerde bron stationair ergodisch is. Zou de klasse Λ zelf als een bron beschouwd zijn dan was deze bron natuurlijk niet ergodisch. Het bovenstaande model omzeilt dit probleem op een elegante manier.

Davisson beschouwt alleen blok naar variabel kodering en zodoende komt hij tot de volgende definitie van universele kodering.

Definitie: Universele kodering is elke willekeurige methode van blok naar variabel kodering voor 'composite sources' onder de volgende eisen:
- de kodering is bloksgewijs geheugenloos.
- een of andere prestatiemaat wordt willekeurig dicht benadert als de bloklengte naar oneindig gaat.

De gedachte achter de eerste eis is dat er onzekerheid kan bestaan over de stationairiteit van de bron. Het is goed mogelijk dat de statistieken uit voorgaande blokken onbetrouwbaar zijn.
Daar Davisson alleen foutloze kodering beschouwd zijn de prestatiematen allen functies van de konditionele koderedundantie.

Definitie: De konditionele koderedundantie voor een bron θ en een N blokkode C_N wordt gegeven door:

$$r_\theta(C_N) \triangleq N^{-1}\left[\, \bar{\ell}_\theta(C_N) - H(X_1^N \mid \theta)\,\right] \qquad (2.1)$$

Uit de variabele lengte bronkoderingsstelling, zie b.v. Gallager [1968], volgt dat $r_\theta(C_N) \geq 0$.

2.1.1 De verschillende vormen van universele kodes.

Het koderingsprobleem is dat een kode C_N gekozen moet worden zonder dat θ bekend is. We zullen een rij C_1^∞ universeel noemen als, onafhankelijk van θ, voor deze rij geldt $r_\theta(C_N) \to 0$, $N \to \infty$. Davisson onderscheidt verschillende soorten universele kodes afhankelijk van de manier van konvergentie, b.v. gewogen konvergentie, puntsgewijze konvergentie en uniforme konvergentie.

Definitie: Een koderij heet <u>gewogen universeel</u> als, gegeven een kansmaat W op Λ, $r_\theta(C_N)$ naar nul konvergeert. d.i. als:

$$\lim_{N\to\infty} \int_\Lambda r_\theta(C_N) dW(\theta) = 0 \qquad (2.2)$$

Definitie: Een koderij heet <u>maximin universeel</u> als aan (2.2) is voldaan voor alle mogelijke W op Λ.

Definitie: Een koderij heet <u>zwak minimax universeel</u> of zwak universeel als $r_\theta(C_N) \to 0$ voor $N \to \infty$, puntsgewijs in θ. Oftewel:

$$\lim_{N\to\infty} r_\theta(C_N) = 0, \text{ alle } \theta \in \Lambda. \qquad (2.3)$$

Definitie: Een koderij heet <u>sterk minimax universeel</u> of minimax universeel als $r_\theta(C_N) \to 0$, $N \to \infty$, uniform in θ.

Het verschil is dat met zwak minimax universele kodes niet en met sterk minimax universele kodes wel voor alle θ de redundantie onder elke gewenste grens ε te krijgen is voor alle bloklengtes groter dan een getal $N(\varepsilon)$.
De gegeven definities zijn in oplopende sterkte, d.w.z. een sterkere konvergentie impliceert een zwakkere, maar niet andersom.

2.1.2 Voorwaarden voor universele kodes.

De volgende stellingen geven aan onder welke voorwaarden universele kodes bestaan voor een klasse Λ van bronnen.
We noemen $r_N^*(W)$ de minimale, bereikbare redundantie, gewogen met W en over blokkodes ter lengte N.

$p(x_1^N)$ is de gewogen kansverdelingsfunctie van X_1^N

$$p(x_1^N) \triangleq \int_\Lambda p(x_1^N|\theta)dW(\theta) \tag{2.4}$$

Met de variabele lengte bronkoderingsstelling volgt

$$N^{-1}\left[H(X_1^N)-H(X_1^N|\Theta)\right] \leq r_N^*(W) \leq N^{-1}\left[H(X_1^N)-H(X_1^N|\Theta)\right]+N^{-1} \tag{2.5}$$

En dus kunnen we $r^*(W) \triangleq \lim_{N\to\infty} r_N^*(W)$ (2.6)

schrijven als $r^*(W) = \lim_{N\to\infty} N^{-1} I(X_1^N;\Theta)$ (2.7)

Stelling 2.1: Voor het bestaan van gewogen universele kodes is een nodige en voldoende voorwaarde dat

$$\lim_{N\to\infty} N^{-1} I(X_1^N;\Theta) = 0$$

Voor bronnen met een eindig alfabet wordt $r_N^*(W)$, begrenst als in (2.5), bereikt door Huffman kodering met de met W gewogen kansverdeling, zie (2.4).

Voor konditioneel stationaire, ergodische bronnen is (2.7) te herschrijven als

$$r^*(W) = \lim_{N\to\infty} I(X_N;\Theta|X_1^{N-1}) \tag{2.8}$$

Dit geeft aan dat, indien $r^*(W)=0$, in de limiet, de bronparameters voldoende bepaald zijn uit de bronuitkomsten.

Definiëren we verder

$$\bar{r}_N \triangleq \sup_W r_N^*(W) \tag{2.9}$$

$$\bar{r} \triangleq \lim_{N\to\infty} \bar{r}_N \tag{2.10}$$

waarin het supremum (2.9) gaat over alle mogelijke kansmaten W op Λ, dan volgt met (2.5):

$$\bar{r} = \lim_{N\to\infty} \sup_W N^{-1} I(X_1^N;\Theta) \tag{2.11}$$

Zo volgt

Stelling 2.2: Voor het bestaan van maximin universele kodes is een nodige en voldoende eis dat de kapaciteit van het kanaal tussen Λ en A nul is.
$\bar{r}_N$, zoals in (2.9), wordt voor bronnen met eindig alfabet bereikt door Huffman kodering met de met $\bar{W}$-gewogen kansverdeling. Hierin is $\bar{W}$ de 'least favorable' kansverdeling, indien deze bestaat.

Nu beschouwen we de nodige en voldoende voorwaarde voor het bestaan van minimax universele kodes.

Stelling 2.3: Een nodige en voldoende voorwaarde voor het bestaan van minimax universele kodes voor een klasse Λ is dat er een rij kansfuncties $\{q(x_1^N)\}_{N=1}^{\infty}$ bestaat, zodat

$$\lim_{N\to\infty} N^{-1} H(p:q) = 0, \text{ uniform in } \theta. \qquad (2.12)$$

Hierin is $H(p:q)$ de relatieve entropie van $p(.\,|\,\theta)$ met betrekking tot $q(.)$.

Voor het bestaan van zwak minimax universele kodes behoeft de konvergentie slechts puntsgewijs in θ te zijn.

2.1.3 Davisson's konklusies.

Enkele resultaten van Davisson zijn:

Als Θ een aftelbaar aantal waarden kan aannemen en $H(\Theta)<\infty$, dan bestaan er gewogen universele kodes.

Als Θ slechts een eindig aantal (=M) waarden kan aannemen, dan bestaan er minimax universele kodes. Zie de konstruktie van Davisson [vb. 2 ref 1973, p786]. In 't kort komt het neer op het vormen van M kodeboeken; het zoeken van het kortste kodewoord onder de M alternatieven voor elk datablok, en het verzenden van de kodeboek index en het kortste kodewoord.

Voor de klasse Λ bestaande uit alle stationair ergodische processen met een eindig bron alfabet bestaan zwak minimax universele kodes. Indien de bronnen ook entropie stabiel zijn, zie hieronder, dan bestaan er sterk minimax universele kodes.

Met entropie stabiel wordt bedoeld dat we een rij ε_1^∞ kennen, zodat voor alle bronnen geldt:

$$\frac{H(X_1^k|\theta)}{k} - \lim_{N\to\infty} \frac{H(X_1^N|\theta)}{N} \leq \varepsilon_k \qquad (2.13)$$

Dit wil niets anders zeggen dan dat we voor elke k een afschatting van de redundantie kunnen geven die geldig is voor elke bron in de klasse.

Dit laatste wordt uitgebreider behandeld in de zgn. kodeboek stelling. Deze stelling geeft nodige voorwaarden voor het bestaan van universele kodes en geeft aanwijzingen hoe deze kodes gevormd kunnen worden. Dit in tegenstelling tot de vorige stellingen die alleen de existentie voorwaarden geven.

Uit deze stelling volgt ook het interessante begrip: voldoende statistiek voor Θ. Hiermee wordt bedoeld dat voor het universeel koderen van een bron Λ niet altijd θ precies bekend moet zijn, doch dat een voldoend nauwkeurige benadering voldoet.

Verder bekijkt Davisson de histogram kodering. Histogram kodering betekent dat door middel van het tellen van het voorkomen van subblokken ter lengte k een schatting gemaakt wordt van de kondi-
tionele kansen $p(x_k|x_1^{k-1})$.

Indien de bron konditioneel stationair en ergodisch is en van het Markov type met bekende orde $k<\infty$, dan geldt de kodeboekstelling en bestaan er minimax universele kodes.

In een recent artikel [1983] geeft Davisson een onder- en een bovengrens voor de minimax redundantie van bovengenoemde klasse.

2.2 Het Ziv-Lempel algoritme.

In Ziv en Lempel [1978] wordt een universeel datakompressie algoritme gegeven. In tegenstelling tot Davisson gaan zij in eerste instantie niet uit van een onvolledig bekende bron, doch van een willekeurige rij symbolen. Een z.g.n. individuele rij. Onder de aanname dat het kodeerproces beschreven kan worden als een 'finite state information lossless generalised automata' leiden ze een onder- en een bovengrens af voor de kompressiefaktor van individuele rijen.
Het bewijs van de ondergrens is konstruktief en behelst hun algoritme. Later laten ze zien dat, indien de rij getrokken is uit een ergodische bron met eindige entropie H, de kompressiefaktor van deze rij gelijk is aan de entropie.

De klasse van '<u>finite state information lossless</u>' (FSIL) kodeerprocessen bestaat uit de kodeerders $E := (S,A,B,g,f)$ met:

S een eindige verzameling van toestanden.

A een eindig invoeralfabet.

B een eindige verzameling van uitvoer woorden over een eindig alfabet.

g een volgende toestand functie $g: S\times A\rightarrow S$.

f een uitvoer functie $f: S\times A\rightarrow B$.

Het '<u>information lossless</u>' zijn volgt uit de volgende eis:

Voor alle $z_1 \varepsilon S$ en alle $x_1^n \varepsilon A^n$, $n\geq 1$ wordt x_1^n uniek bepaald uit $(z_1, f^n(z_1,x_1^n), g^n(z_1,x_1^n))$. Hierin wordt met $f^n(z_1,x_1^n)$ resp. $g^n(z_1,x_1^n)$ de rij van n uitvoer woorden resp. de rij van n

toestanden bedoeld, welke ontstaan als in toestand z_1 begonnen wordt met x_1^n als invoer.

Door in B woorden van verschillende lengten toe te staan is elk eindig blok naar variabel schema mogelijk. Indien B ook het lege woord λ bevat, zijn ook variabel naar variabel schema's mogelijk.

Met de kompressiefaktor $\rho(x)$ van een oneindige rij $x \triangleq x_1^\infty$, met een eindig alfabet A, wordt de verhouding tussen de lengte van de kortste representatie van x, realiseerbaar door een FSIL proces, en de lengte van x zelf bedoeld.

2.2.1 Het koderingsalgoritme.

Het Ziv-Lempel algoritme bestaat uit drie stappen. Zie figuur 2.1. In de eerste stap wordt de rij x verdeeld in blokken b_i ter lengte n. Elk blok b wordt in de tweede stap door de 'incremental parser' gesplitst in p+1 segmenten. p is een functie van de bloklengte n en de inhoud van het aangeboden blok b.

Definieer de 'incremental parsing' s_1^{p+1} van x_1^n als:

$$s_1^{p+1} \triangleq x_{n_0+1}^{n_1}, x_{n_1+1}^{n_2}, \ldots, x_{n_p+1}^{n_{p+1}} \qquad (2.14)$$

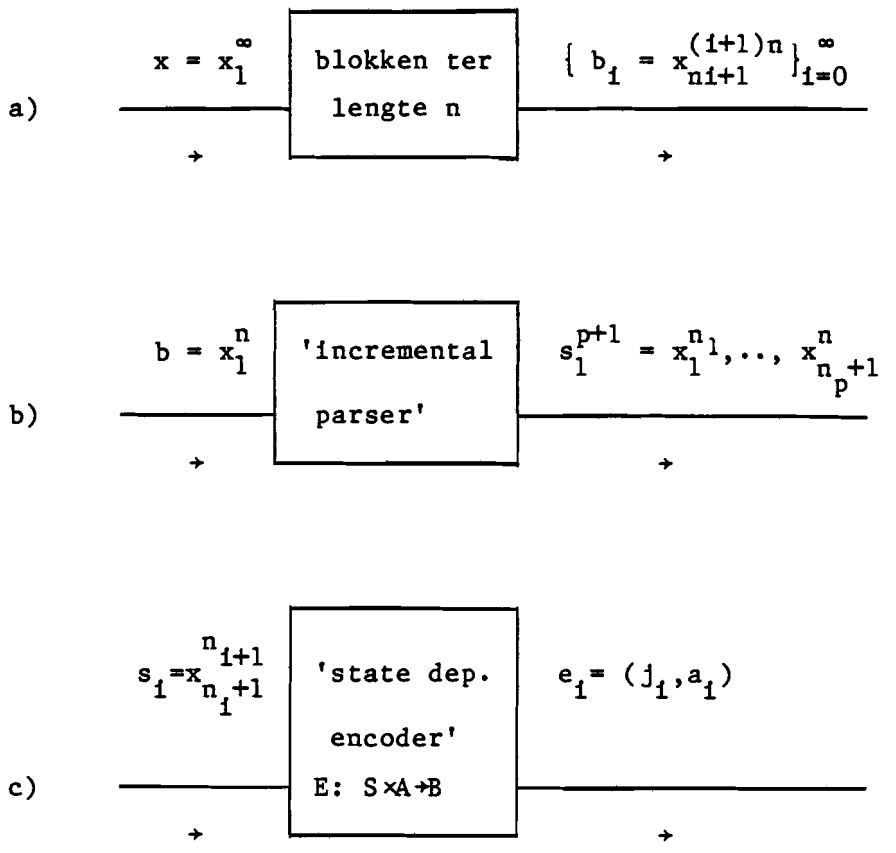
met $0=n_0<n_1<n_2<..<n_p<n_{p+1}=n$

s_1^{p+1} moet voldoen aan de volgende twee eisen:

i) De eerste p segmenten zijn alle verschillend.
Het laatste segment mag gelijk zijn aan een van de voorgaande.

ii) Voor alle segmenten s_i met een lengte $\ell(s_i)>1$ geldt dat de prefix van s_i ter lengte $\ell(s_i)-1$ al voor s_i in de rij voorkomt.

De 'incremental parser' kreëert een nieuw segment zo gauw de prefix van het nog niet ontleedde deel van de invoerrij nog niet in de rij s voorkomt. Eenvoudig is te zien, na enig herschrijven, dat de 'incremental parser' behoort tot de klasse FSIL.



Figuur 2.1. Het Ziv-Lempel algoritme.

a) eerste stap.

b) tweede stap.

c) derde stap.

In de laatste stap vindt de kodering van de segmenten plaats. Uit de eigenschappen van de 'incremental parser' volgt dat elk segment s_i, met lengte $\ell(s_i) \geq 2$, uniek gegeven wordt door de index j_i van de prefix van s_i ter lengte $\ell(s_i)-1$ en de suffix $x_{n_{i+1}}$ van s_i. Deze suffix noteren we als a_i.

Segmenten s_k, met lengte 1, worden gekodeerd als $(0,x_{n_k+1})$.
0 is a.h.w. de index van het lege segment λ, dat we als nulde segment aan elke 'incremental parsing' toe kunnen voegen.
Een voorbeeld van de werking van de 'incremental parser' en de bijbehorende kode:

$n = 16$, $A = \{0,1\}$,
$b = x_1^{16} =$ 00111 10101 10101 1

j	s_j	e_j	kodewoord.
0	λ	-	λ
1	0	(0,0)	0
2	01	(1,1)	11
3	1	(0,1)	001
4	11	(3,1)	111
5	010	(2,0)	0100
6	110	(4,0)	1000
7	10	(3,0)	0110
8	11	(3,1)	0111

Figuur 2.2. 'Parsing' en koderen.

In figuur 2.2 zien we de verwerking van de string b. In de eerste kolom is de segmentindex gegeven; in de tweede het segment en in de derde de bijbehorende kode. In de vierde kolom staat de binaire representatie van de kode e_j. Deze ontstaat op de volgende manier:

> De kode van het j^e segment bestaat uit een index i(j), die een waarde van 0 t/m j-1 kan aannemen. De suffix a(j) kan $|A|$ waarden aannemen. Er zijn dus $j|A|$ verschillende kombinaties te vormen. Deze kunnen worden toegekend aan de getallen 0 t/m $j|A|-1$ als aan elke a uit A een getal $g(a)\varepsilon\{0,1,..,|A|-1\}$ wordt toegekend.

In figuur 2.2, vierde kolom, is gekozen voor de binaire representatie k_j in ceil(log 2j) bits volgens $k_j=2i(j)+g(a(j))$.

De kodestring is eenduidig te dekoderen daar voor elk kodewoord k_i de lengte vooraf bekend is en de afbeeldig van e_i op k_i inverteerbaar is.

Het voorbeeld is karakteristiek voor het algoritme in die zin dat het laat zien dat de kompressiefaktor groter dan een kan zijn. Dit treedt in het algemeen op indien de bloklengte n klein is.

2.2.2 Het resultaat.

Ziv en Lempel bewijzen het volgende:

> Voor elke oneindige rij x is $\rho_E(x,n)$ de door het algoritme gerealiseerde kompressiefaktor bij blokken ter lengte n.
> Nu geldt voor elke $\varepsilon>0$:

$$\rho_E(x,n) \leq \rho(x) + \delta_\varepsilon(x,n) \qquad (2.15)$$

met

$$\lim_{n\to\infty} \delta_{\varepsilon}(x,n) = \varepsilon, \text{ voor alle } x. \qquad (2.16)$$

Zie stelling 4 in Ziv en Lempel [1978], indien x getrokken is uit een ergodische bron met entropie H, geldt dat $\rho(x)=H$. Samen met (2.15) volgt dan dat voor een ergodische bron het Ziv-Lempel algoritme in de limiet optimaal is.

2.3 De aanpak van Rissanen en Langdon.

In navolging van Davisson [1973] splitsen Rissanen en Langdon [1981-a] het probleem van de universele kodering op in een modelvormend- en een koderend deel. Waar Davisson, zie de kodeboekstelling in zijn artikel [1973], deze splitsing als een mogelijk alternatief ziet, is volgens Rissanen en Langdon deze splitsing essentiëel. Zij stellen het probleem als volgt:

Probleem: Gegeven is een eindige doch niet volledig bepaalde verzameling van eindige datarijen. Elk van deze rijen kan door een andere, onvolledig bepaalde bron gegenereerd zijn. Gevraagd is één universeel foutloos kompressiesysteem hiervoor.

Het modelvormend deel splitsen zij verder op in een struktuur, die bestaat uit de mogelijke gebeurtenissen en hun kontekst, en de parameters, welke de waarschijnlijkheden van deze gebeurtenissen zijn. In de struktuur kunnen de redundanties opgenomen worden, die, voor zover ze bekend zijn, in de hele verzameling datarijen voorkomen. De parameters worden dan door de modelvormer voor elke afzonderlijke rij ingevuld. De kodeerder moet met de gegeven parameters een koderij genereren met een lengte dicht bij het modelafhankelijke optimum. Ook moet deze geen restrikties aan de modelvormer opleggen.

Rissanen en Langdon stellen voor om voor de kodeerder een aritmetische kode te gebruiken. Deze voldoet aan de bovengestelde eisen. Deze kodes worden in dit verslag niet verder behandeld.

2.3.1 Het model.

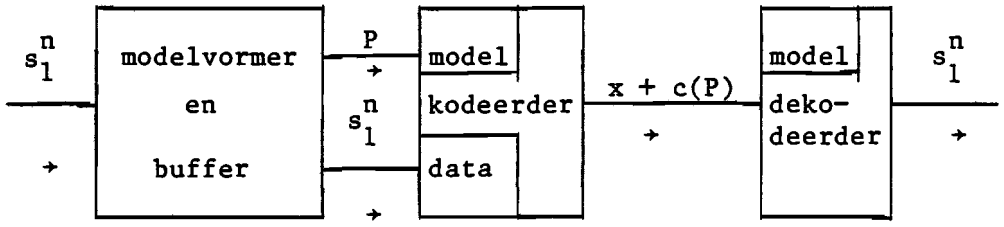
Voor het model staan nog verschillende opties open.

-i Wordt alfabet extensie toegepast of niet?
-ii Is het model stationair of adaptief?

De belangrijkste stelling in hun artikel [1981-a] laat zien dat alfabet extensie, d.w.z. het bij elkaar groeperen van datasegmenten van vaste of variabele lengte, niet zinvol is. In sommige gevallen kan men zelfs beter doen zonder alfabet extensie. Hierbij moet wel worden opgemerkt dat om implementatietechnische redenen alfabet extensie soms wel toegepast kan worden.

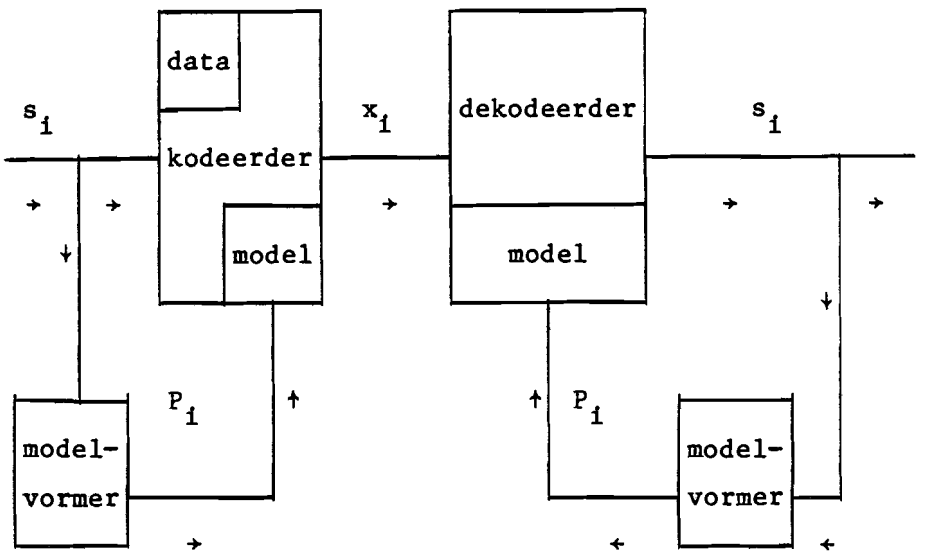
Het verschil tussen een stationair- en een adaptief model komt in het kort neer op het volgende: In een stationair model bepaalt de modelvormer in een eerste stap de parameters uit de volledige datarij. In de tweede stap wordt de gehele rij met behulp van dit eindresultaat gekodeerd. In deze slag moet ook eerst het gebruikte model aan de ontvanger verteld worden. Zie figuur 2.3.

Het modelvormen en het koderen gebeurt na elkaar. De rij moet in zijn geheel opgeslagen worden in een buffer voor de tweede slag. Dit is vaak onakseptabel of zelfs onmogelijk. Een standaard oplossing hiervoor is dat de parameters bepaald worden uit een klein aantal representatieve datarijen en dat deze dan vast in het algoritme ingebracht worden. De eerste stap vervalt dan. Gezien de probleemstelling, zie boven, rijst direkt de vraag: is er een kleine deelverzameling representatieve datarijen en zo ja, hoe worden deze bepaald? Veelal wordt hiervoor een ad hoc keuze gedaan. Een beter alternatief is het adaptieve model.



Figuur 2.3. Het stationaire schema.
P : de model parameters.
s : de datarij.
x : de gekodeerde datarij.
c(P) : de gekodeerde parameters.

In een adaptief model worden al koderend de parameters gevormd. Hier geschieden het modelleren en het koderen parallel. Zie ook figuur 2.4 verderop.
De eerste toepassing voor dit adaptief schema is het aanpassen aan stationaire statistieken. Het adaptief schema geeft echter ook de mogelijkheid om het model aan te passen aan niet stationaire bronnen. In [1981-b] stellen Langdon en Rissanen voor een statistische signifikantie test op de opvolgende symbolen te doen. De resultaten in hun artikel [1981-b] laten zien dat deze laatste methode zeer geschikt is. De resultaten tonen ook een slechts minimaal verschil tussen het stationaire twee-slagen algoritme en het stationair adaptieve algoritme.



Figuur 2.4. Het adaptieve schema.
P_i : de parameters na i-1 symbolen.
s_i : het i^e symbool.
x_i : het i^e kodedeel.

2.3.2 Een alternatieve bronstruktuur.

Rissanen en Langdon stellen een enigzins aangepaste bronmodel-struktuur voor. Het enig wezelijke verschil tussen dit model en het traditionele model van het stochastische proces is dat hun model de nadruk legt op het eindig zijn van de datarijen, terwijl het stochastische proces gezien wordt als een nooit stoppend gebeuren.
Hun informatiebron genereert één uitkomst uit een verzameling S^* waarin S het eindige bron- of data-alfabet is. Een funktie P kent aan elke rij uit S^* een reëel getal tussen nul en één toe.

Aan P worden de volgende eisen gesteld:

$$P: S^* \to [0,1]$$

$$P(\lambda) \stackrel{\Delta}{=} 1 \qquad (2.17)$$

$$P(s) = \sum_{x \in S} P(sx), \text{ voor alle } s \in S^*$$

De informatiebron $<S^*,P>$ is geen stochastische variabele; gesommeerd over S^* is $\sum P(s)$ immers groter dan 1.
Toch kan $P(s)$ als de waarschijnlijkheid van s opgevat worden. Hiervoor zijn eerst de volgende begrippen nodig.

Definitie: Een deelverzameling A van S^* heet <u>kompleet</u> als elke rij $x_1^\infty \in S^\infty$ een prefix in A bezit.

Definitie: Een deelverzameling A van S^* heet <u>proper</u> als elke rij $u \in A$ geen andere prefix dan zichzelf in A bezit.

Voor alle deelverzamelingen A van S^*, die zowel kompleet als proper zijn, geldt dat de variabele x met waarden in A en een kans gegeven voor alle $s \in A$ als $Pr\{x=s\} = P(s)$ een korrekt gedefinieerde stochastische variabele is. Dit is eenvoudig te zien.

Laat s een willekeurig vast element uit S^* zijn.
Nu volgt dat voor alle komplete en propere deelverzamelingen A_s van S^* met $s \in A_s$ aan s dezelfde kans $P(s)$ wordt toegekend. Hierin vinden we de rechtvaardiging voor de naam waarschijnlijkheid van $P(s)$.

Stationairiteit en onafhankelijkheid worden aangegeven door de eisen (2.18) resp. (2.19).

$$P(s) = \sum_{x \in S} P(xs), \text{ voor alle } s \in S^* \qquad (2.18)$$

$$P(ss') = P(s)P(s'), \text{ voor alle } s \in S^* \text{ en } s' \in S^* \qquad (2.19)$$

Ook de konditionele waarschijnlijkheden kunnen gegeven worden in termen van de funktie P, n.l.

$$P(x|s) = \frac{P(sx)}{P(s)}, \text{ voor alle } x \in S \text{ en } s \in S^*. \qquad (2.20)$$

Evenzo

$$P(s_1^n) = P(s_1)P(s_2|s_1)P(s_3|s_1^2)\ldots P(s_n|s_1^{n-1}) \quad \text{voor alle } s_1^n \in S^* \qquad (2.21)$$

2.3.3 De kontekst.

De modelvormer moet in feite de funktie P schatten uit de datarij. Zonder extra restrikties aan de vorm van P is dit duidelijk een onmogelijke zaak. De restriktie aan de algemene vorm van P is de modelstruktuur.

Uit formules (2.20) en (2.21) zien we dat P ook bepaald kan worden uit de konditionele waarschijnlijkheden.

Rissanen en Langdon nemen aan dat de konditionele waarschijnlijkheden bepaald worden door een of andere relevante eigenschap van het verleden, d.w.z. de al verwerkte prefix van de datarij. Deze eigenschap noemen zij de kontekst.

Formeel:

f is een struktuurfunktie, welke S^* partitioneert in K equivalentieklassen of konteksten.

$$f: S^* \rightarrow Z \triangleq \{0,1,\ldots,K-1\} \qquad (2.22)$$

Een geschikte subklasse van struktuurfunkties wordt gegeven door de eis dat f een 'finite state machine' (FSM) is. Dit is een, praktisch gezien, reële eis. Zie Rissanen en Langdon [1981-a en 1981-b].
Een FSM is in feite een eindig Markov model met een equivalentie-relatie tussen de verschillende toestanden. Het voordeel van deze scheiding is dat een relatief groot Markov model toch weinig vrije parameters heeft.

De manier waarop f van invloed is op de konditionele waarschijnlijkheden is afhankelijk van de keuze of het model stationair dan wel adaptief is.
In het stationaire geval wordt de konditionele waarschijnlijkheid beperkt tot konditionering op de kontekst, dus:

$$P(x|s) = P(x|f(s)) \qquad (2.23)$$

Indien S kardinaliteit d heeft zijn er K.(d-1) parameters nodig om $<S^*,P>$ te beschrijven.
In het adaptieve geval hangt de konditionele kans behalve van f(s) ook af van de lijst van volgende symbolen. d.w.z. $f(s_1^t)=z$ is een kontekst, dan behoort s_{t+1} tot de lijst van volgende symbolen $s[z]$ van z. Dus:

$$P(x|s) = P(x|f(s),s[f(s)]) \qquad (2.24)$$

In het adaptieve geval zijn K.(d-1) registers nodig voor de opslag van de berekende parameters. Rissanen en Langdon definieren de komplexiteit van zowel het stationaire- als het adaptieve schema als K.(d-1).

Gegeven een struktuurformule f en een rij s worden de kansen $P(x|z)$ in het stationaire geval bepaald volgens:

$$P(x|z) = c(x|z)/c(z) \qquad (2.25)$$

Hierin is $c(x|z)$ het aantal malen dat het symbool x volgt op het optreden van de kontekst z. c(z) staat voor het aantal keren dat kontekst z optreedt. Deze keuze maximaliseert $P(s)$ gegeven de funktie f.

Daar $P(s)$ als een echte kans beschouwd kan worden, kan aan:

$$I(s) \triangleq -\log P(s) \qquad (2.26)$$

de interpretatie optimale kodelengte gegeven worden.
Het maximaliseren van $P(s)$ minimaliseert $I(s)$, wat gewenst is.

Een voorbeeld:

$S = \{0,1\}$
definiëer $f: S^* \rightarrow \{0,1\}$

$$f(s_1^n) \triangleq s_n, \quad f(\lambda) \triangleq 0.$$

Dit is een 1^e orde Markov bron model.

De datarij s is 00011 01010 0101

dan volgt voor c(z): $c(0)=9$, $c(1)=5$.

$$c(x|z): \quad c(0|0)=4, \quad c(1|0)=5$$

$$c(0|1)=4, \quad c(1|1)=1$$

Hieruit volgt $P(0|0) = 4/9$, $P(0|1) = 4/5$,

zodat $I(s) = 9\log 9 - 8\log 4 = 12{,}53$ bit.

De komplexiteit van dit model is 2. Er zijn 2 parameters nodig voor de beschrijving van P.

2.3.4 Besluit.

Behalve de nadruk die op het eindig zijn van de datarijen wordt gelegd, vind ik geen enkele reden om een bron te modelleren als het paar $<S^*,P>$. m.i. kan elk model $<S^*,P>$ even goed door een Markov proces beschreven worden met de, normale, konventie dat een eindige rij s gezien wordt als de verzameling van alle oneindige datarijen x zo, dat s een prefix van elke x is. Zeker daar Rissanen en Langdon de aandacht hoofdzakelijk op 'finite state machines' richten is een bronbeschrijving door een stochastisch proces even geldig.

3.0 Het gedrag van de blok- en de innovatie entropie.

Het koderen van bronuitkomsten kan op twee principiëel verschillende manieren. In de eerste manier, die we segment koderen noemen, splitst de kodeerder de datarij in brokstukken of segmenten. Aan elk van segment wordt een apart kodewoord toegekend.
De andere manier kodeert elk bronsymbool apart, afhankelijk van de vorige symbolen. Dit noemen we kontekst koderen.

We koderen een bron met entropie $H_\infty(U)$ met een segment kode en eisen daarbij dat alle segmenten dezelfde lengte L hebben. Voor deze kode is de blok entropie $H_L(U)$ de ondergrens van de kompressie.
Evenzo is de innovatie entropie $H(U_L|U_1^{L-1})$ de ondergrens indien we koderen met een kontekst kode waarbij de kodering afhangt van de laatste L-1 symbolen.
Het loont de moeite het gedrag van deze entropieën te bekijken.

3.1 De vorm van de entropie funkties.

Uit Gallager [1968] is de volgende stelling bekend:

Voor diskrete stationaire bronnen met $H_1(U)<\infty$ geldt:

- $H(U_L|U_1^{L-1})$ is monotoon niet-stijgend in L,
- $H_L(U)$ is monotoon niet-stijgend in L,
- $H_L(U) \geq H(U_L|U_1^{L-1})$ voor alle L, (3.1)
- $\lim_{L\to\infty} H_L(U) = \lim_{L\to\infty} H(U_L|U_1^{L-1})$

In figuur 3.1 staat het gedrag beschreven van een hypothetische bron met eindig geheugen Λ, d.w.z. de Λ voorgaande symbolen bepalen volledig het gedrag van de bron.

Definitie: Bron U heeft een <u>eindig geheugen</u> Λ als geldt:

$$H(U_L \mid U_1^{L-1}) > H_\infty(U) \text{ voor } L \leq \Lambda,$$
$$H(U_L \mid U_1^{L-1}) = H_\infty(U) \text{ voor } L > \Lambda. \qquad (3.2)$$

Figuur 3.1 vertoont de volgende gedragingen:

Voor $H(U_L \mid U_1^{L-1})$:

Een aanvangsplateau, d.w.z. dat de konditionering op de 1e k symbolen (k=4 in fig 3.1) niets meer vertelt dan geen konditionering. Zie voorbeeld I.

Een tussenplateau, d.w.z. dat de bronentropie gekonditioneert op de k voorgaande symbolen gelijk is aan die gekonditioneert op de k+m voorgaande symbolen. (in fig 3.1: k=8 en m=3). Zie voorbeeld II.

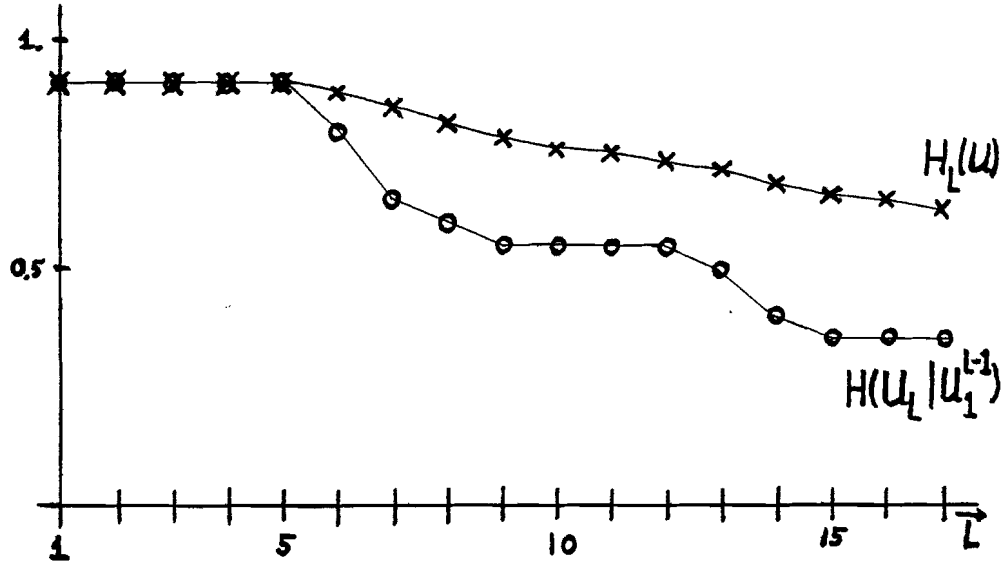
Het eindig geheugen Λ betekent dat $H(U_L \mid U_1^{L-1}) = H_\infty(U)$ voor $L > \Lambda$. (hier $\Lambda = 14$.)

Voor $H_L(U)$:

Een aanvangsplateau identiek aan dat van $H(U_L \mid U_1^{L-1})$.

Na het aanvangsplateau is $H_L(U)$ strikt dalend. Zie verderop voor het bewijs.

Uit het bestaan van de plateaus volgt dat $H(U_L \mid U_1^{L-1})$ en $H_L(U)$ i.h.a. niet convex zijn.



Figuur 3.1.

Het bewijs van het monotoon dalend zijn van $H_L(U)$ na een eindig plateau ter lengte k volgt nu.

Er geldt:

$$H(U_1) > H(U_L|U_1^{L-1}) \text{ als } L>k \tag{3.3}$$

$H_L(U)$ is monotoon dalend als $H_L(U)-H_{L+1}(U)$ strikt positief is.

$$H_L(U)-H_{L+1}(U) = \frac{1}{L+1}\{H_L(U)-H(U_{L+1}|U_1^L)\} \tag{3.4}$$

Met

$$H(U_{L+1}|U_1^L) \leq H(U_L|U_1^{L-1})$$

gaat (3.4) over in:

$$H_L(U) - H_{L+1}(U) \geq \frac{1}{L+1}\{H_L(U)-H(U_L|U_1^{L-1})\} \tag{3.5}$$

Indien L>K geldt (3.3) en dus

$$H_L(U) > H(U_L|U_1^{L-1}),$$

samen met (3.5) geeft dit

$$H_L(U) - H_{L+1}(U) > 0, \qquad (3.6)$$

zodat $H_L(U)$ monotoon dalend is voor alle L>K.

∴

3.2 De plateaus.

De nu volgende twee voorbeelden tonen het bestaan van de plateaus aan.

Voorbeeld I.

Bron I, zie figuur 3.2, toont een aanvangsplateau ter lengte 1 in $H(U_L|U_1^{L-1})$ en $H_L(U)$.
De bron wordt beschreven door twee parameters α en β, beide uit het interval (0,1). Deze bron is stationair ergodisch en kan beschreven worden als een 2^e orde Markov bron.
De stationaire verdeling wordt gegeven door:

$$P_{00} = \frac{\beta^2}{(1-\alpha+\beta)^2}, \quad P_{11} = \frac{(1-\alpha)^2}{(1-\alpha+\beta)^2},$$

$$P_{01} = P_{10} = \frac{(1-\alpha)\beta}{(1-\alpha+\beta)^2}.$$

Na enig rekenen volgt:

$$H(U_1) = h\left(\frac{\beta}{1-\alpha+\beta}\right)$$

$$H(U_2|U_1) = H(U_1)$$

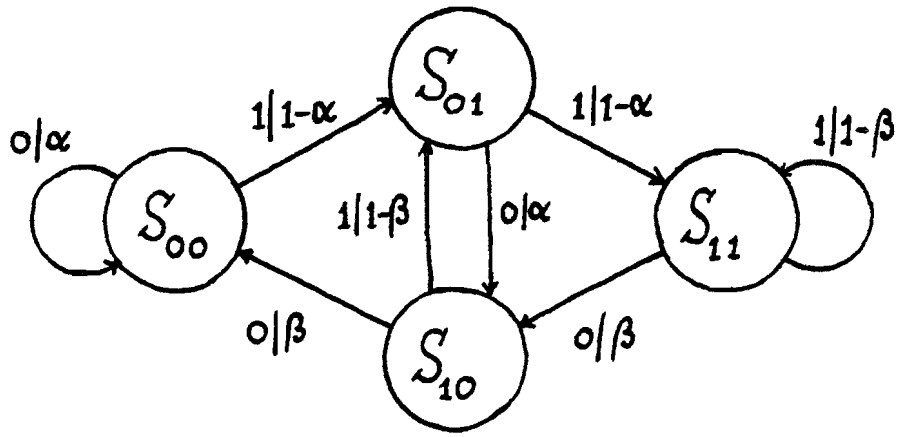
$$H(U_3|U_1U_2) = H_\infty(U) = \frac{1}{1-\alpha+\beta}\{\beta h(\alpha)+\alpha h(\beta)\}$$

h(.) is de binaire entropiefunktie.

We zien dat voor $\alpha \in (0,1)$ en $\beta \in (0,1)$ geldt

$$H(U_1) = H(U_2|U_1) > H(U_3|U_1U_2) = H_\infty(U)$$

De ongelijkheid volgt uit het convex zijn van h(.).



Figuur 3.2. Bron I.

Voorbeeld II.

Bron II is een 3^e orde Markov bron, beschreven door vier parameters $u,x,y,z \in (0,1)$, waar de overgangswaarschijnlijkheden zo gekozen zijn dat deze een tussenplateau vertoont. Zie figuur 3.3.
De stationaire verdeling wordt gegeven door:

$$p_{000} = p_{111} = (1-u)z/N,\ p_{001} = p_{110} = (1-x)z/N,$$

$$p_{010} = p_{101} = (1-x)y/N,\ p_{011} = p_{100} = (1-x)z/N.$$

$$N \triangleq 2\big(z(1-u+1-x) + (1-x)(y+z)\big)$$

Er volgt:

$$H(U_1) = h(\tfrac{1}{2}) = 1 \text{ bit.}$$

$$H(U_2|U_1) = h\left(\frac{2(1-x)(y+z)}{N}\right) \quad \{ = 1 \text{ bit als } (1-u)z=(1-x)y \}$$

$$H(U_3|U_1U_2) = H(U_2|U_1) \text{ als } (1-u)y = (1-x)z$$

$$H(U_4|U_1U_2U_3) = H_\infty(U) = \frac{2}{N} \cdot$$

$$[(1-x)zh(u)+(1-u)zh(x)+(1-x)zh(y)+(1-x)yh(z)].$$

Een getallenvoorbeeld voor deze bron laat zien dat het verschil tussen het plateau en de eindwaarde aanzienlijk kan zijn. Kies:

$$u=\frac{7999}{8000}\ ,\ x=\frac{3997}{4000}\ ,\ y=\frac{3}{4000}\ ,\ z=\frac{1}{8000}\ .$$

dus $(1-u)y=(1-x)z$ en $(1-u)z\neq(1-x)y$

$$H(U_1) = 1 \text{ bit.}$$

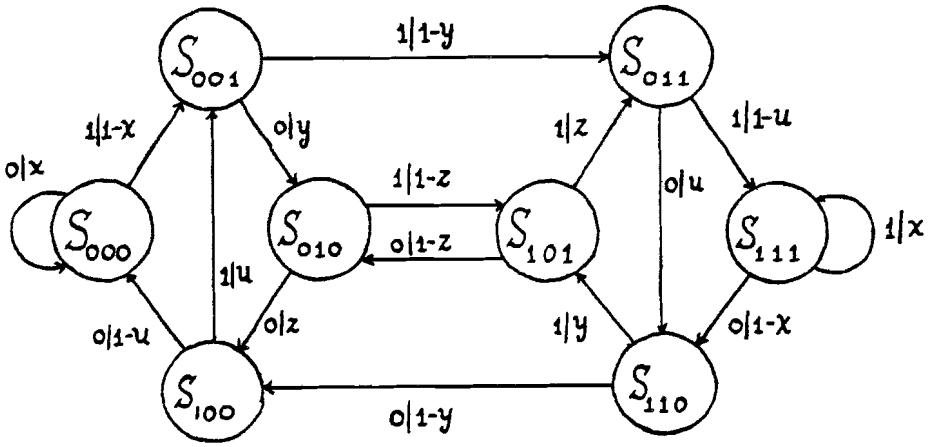
$$H(U_2|U_1) = .592 \text{ bit.}$$

$$H_\infty(U) = .003 \text{ bit.}$$

Voor deze bron heeft de blok entropie het volgende verloop:

$$H_1(U) = 1 \text{ bit.}$$
$$H_2(U) = .796 \text{ bit.}$$
$$H_3(U) = .728 \text{ bit.}$$
$$H_4(U) = .547 \text{ bit.}$$

We zien dat ook na een eventueel aanvangsplateau $H_L(U)$ niet convex behoeft te zijn.



Figuur 3.3. Bron II.

3.3 Konklusie.

De hier beschreven bronnen behoren beide tot de klasse van stationaire ergodische Markov bronnen met eindige orde. Uit Davisson's resultaten [1973] volgt dat er geen sterk minimax universele kodes voor deze klasse bestaan, wel zwak minimax universele kodes.
De bovenstaande voorbeelden laten zien wat hier mee bedoeld wordt. Het verloop van de beide entropieën is zo grillig dat geen algemeen geldende bovengrens voor de reduktiefaktor gegeven kan worden. Dit lukt wel voor de klasse van stationaire ergodische Markov bronnen met bekende eindige orde k. Zie Davisson [1983].

4.0 Bronkodering met segment kodes.

4.1 Komplete en propere segment kodes.

In dit hoofdstuk gaan we kodes beschouwen, die een bronrij opdelen in disjunkte stukken en aan elk stuk een apart kodewoord toekennen. Zo'n kode bestaat uit het drietal (V_B, V_C, f), waarin:

- V_B een eindige verzameling van eindige rijen over het bronalfabet A is,
- V_C een eindige verzameling van eindige rijen over het kodealfabet D is,
- f een inverteerbare funktie van V_B naar V_C is.

Van V_B eisen we dat elke oneindige rij $x \in A^\infty$ op precies één manier op te splitsen is in een rij van elementen uit V_B. Verder eisen we dat V_B en V_C evenveel elementen bevatten.
Van V_C eisen we dat de koderij dekodeerbaar is. Zonder verlies van algemeenheid mogen we eisen dat V_C aan de prefix konditie voldoet. Dit volgt uit een stelling van McMillan, zie Gallager [1968, p49].
f kent aan elk segment $b \in V_B$ een uniek kodewoord $c \in V_C$ toe.
Uit de definitie van een komplete en een propere verzameling, zie hoofdstuk 2.3.2, volgt dat V_B aan de bovenstaande eis voldoet slechts dan als V_B kompleet en proper is.

De verzameling V_C is een propere verzameling, daar aan de prefix konditie voldaan is. Stel dat V_C niet kompleet zou zijn dan kunnen enkele rijen uit V_C ingekort worden zodat V_C alsnog kompleet wordt, zonder dat V_C daarmee niet proper wordt of elementen verliest. Wel zijn enkele kodewoorden in lengte afgenomen en dus is de kode in de zin van de gemiddelde kodewoordlengte niet slechter geworden. We mogen V_C dus beperken tot een komplete en propere verzameling.

4.1.1 Komplete en propere verzamelingen en n-bomen.

Een komplete en propere verzameling V van rijen over een eindig alfabet W van n letters is te beschrijven als de verzameling van bladeren van een komplete n-boom. Met een n-boom bedoelen we dat in de boom elke knoop niet meer dan n opvolgers kan hebben.

Enkele begrippen:

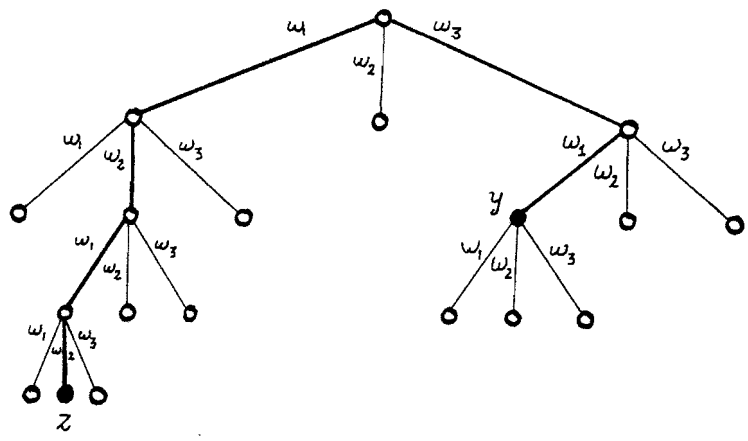
> Een n-boom is kompleet als elke knoop òf nul òf n opvolgers heeft. Een knoop heet een inwendige knoop als het één of meerdere opvolgers heeft, en heet een blad als het geen opvolgers heeft. Een komplete n-boom bevat $\alpha(n-1)+1$ bladeren voor een $\alpha \in \mathbb{N}$.

We zullen aan elk van de n verbindingen naar de opvolgers van een inwendige knoop in een afgesproken volgorde een uniek element uit het alfabet W toekennen, zie figuur 4.1 voor een voorbeeld.
Het is bekend dat elke knoop z in een boom een uniek pad van de wortel naar z heeft. Noteren we achtereenvolgens alle letters die we op dit pad tegen komen, dan krijgen we een eindige rij $w_1^{\ell(z)}$.
Met $\ell(z)$ wordt de lengte van het pad bedoeld. We zullen, gegeven een afspraak over hoe we de takken labellen, geen onderscheid meer maken tussen de knoop z en de rij w.

De afspraak die we de standaard labeling zullen noemen is:

> Laat W gegeven zijn als $\{w_1, w_2, .., w_n\}$ en de komplete n-boom B in een plat vlak ingebed zijn, dan labellen we de takken van links naar rechts met achtereenvolgens w_1, w_2, enz.

De verzameling van bladeren van een komplete n-boom met label alfabet W en standaardlabeling noteren we met L_B.
L_B is een propere en komplete verzameling van rijen over W.



Figuur 4.1. Een komplete 3-boom.
Deze boom is standaard gelabeld.
$z = w_1w_2w_1w_2$
$y = w_3w_1$

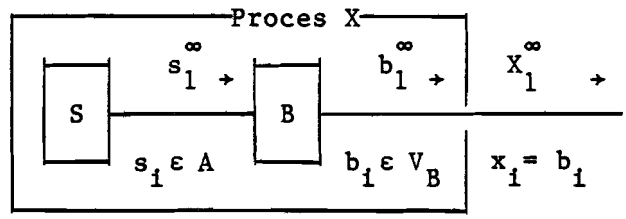
4.1.2 Segment kodes in boombeschrijving.

S is een geheugenloze bron met een alfabet A van n letters, een kansfunktie p(.), en een entropie $H(S)<\infty$. (V_B,V_C,f) is een kode voor deze bron, dus V_B is de verzameling bladeren van een komplete n-boom B.
Indien we een oneindige rij $s_1^\infty \in A^\infty$, gegenereerd door S, opsplitsen in segmenten volgens B, dan kunnen we de kans op een segment $b=s_1^k \in V_B$ schrijven als: $P_B(b) = \prod_{i=1}^{k} p(s_i)$. (4.1)

Daar S geheugenloos is en V_B proper, geldt dat de verschillende segment uitkomsten onafhankelijk zijn. En daar V_B kompleet is geldt:

$$\sum_{b\in V_B} P_B(b) = 1. \qquad (4.2)$$

We beschouwen de bron met segmentvormer B samen als een nieuw stochastisch proces $X \triangleq X_1^{\infty}$. Zie figuur 4.2. De stochastische variabelen X_i zijn onderling onafhankelijk en identiek verdeeld. De uitkomstenverzameling is V_B en de kansfunktie is P_B.



Figuur 4.2.

De entropie van X is gegeven als:

$$H(X) \triangleq - \sum_{b \varepsilon V_B} P_B(b) \log P_B(b) \tag{4.3}$$

De gemiddelde segmentlengte $\bar{\ell}_b$ is gegeven door:

$$\bar{\ell}_b \triangleq \sum_{b \varepsilon V_B} P_B(b) \ell(b) \tag{4.4}$$

We kunnen van het proces X de entropie normeren op de oorspronkelijke bronsymbolen uit A volgens:

$$H(B) \triangleq \frac{H(X)}{\bar{\ell}_b} \tag{4.5}$$

De segmentvormer B stuurt de kodevormer bestaande uit V_C en f. De uitkomsten van deze kodevormer kunnen we zien als een stochastisch proces Y met uitkomsten verzameling V_C en kansfunktie $P_C(c) \triangleq P_B(f^{-1}(c))$ voor alle $c \varepsilon V_C$.

Met C noteren we de komplete n-boom, zodat $V_C = L_C$.

Er geldt:

$$H(Y) = H(X) \tag{4.6}$$

De gemiddelde kodewoordlengte $\bar{\ell}_c$ wordt gegeven door:

$$\bar{\ell}_c \triangleq \sum_{c \in V_C} P_C(c)\, \ell(c) \tag{4.7}$$

De reduktiefaktor ρ welke door deze kode op bron S gerealiseerd wordt is:

$$\rho = \frac{\log \|D\|}{\log \|A\|} \cdot \frac{\bar{\ell}_c}{\bar{\ell}_b} \tag{4.8}$$

In het vervolg nemen we aan dat $\|D\| = \|A\|$, zodat (4.8) over gaat in

$$\rho = \frac{\bar{\ell}_c}{\bar{\ell}_b} \tag{4.9}$$

Daar elke knoop k uit B eenduidig een eindige bronrij definiëert, kunnen we aan elke knoop k een getal q(k) toekennen. Met q(k) de kans op rij k, zoals gedefiniëerd door de bron S. Het toekennen van deze gewichten aan knopen van een boom B noemen we het <u>wegen van B met S</u>.

Dit gebruiken we in het volgende lemma. Dit lemma geeft een alternatieve manier om de gemiddelde segmentlengte van een gewogen boom te bepalen.

Lemma 4.1: Voor een komplete n-boom B gewogen met S geldt:

$$\bar{\ell}_b = \sum_{z \in L_B} q(z)\,\ell(z) = \sum_{y \in B - L_B} q(y) \qquad (4.10)$$

Bewijs:

Laat (4.10) waar zijn voor alle komplete n-bomen met niet meer dan $m = \alpha(n-1)+1$ bladeren, voor bepaalde $\alpha \in \mathbb{N}$.
B^1 is een komplete n-boom met $m+n-1$ bladeren.
z_1 is een blad van B^1 met maximale lengte, dus:

$$z_1 \in L_{B^1} \text{ en voor alle } z \in L_{B^1} : \ell(z_1) \geq \ell(z) \qquad (4.11)$$

De vader y van z_1 bestaat daar B^1 minstens $n+1$ knopen bezit. De n opvolgers van y zijn allen bladeren, daar z de maximale lengte heeft. Noem deze $z_1, z_2, \ldots, z_n$.
Nu geldt:

$$\bar{\ell}_{b^1} = \sum_{z \in L_{B^1} - \{z_1, z_2, \ldots, z_n\}} q(z)\,\ell(z) + q(y)\big(\ell(y)+1\big) \qquad (4.12)$$

Noem B^2 de komplete n-boom die uit B^1 ontstaat als de opvolgers van y weggelaten worden. dus:

$$\bar{\ell}_{b^1} = \sum_{z \in L_{B^2}} q(z)\,\ell(z) + q(y).$$

En voor B^2 geldt de induktie veronderstelling zodat dit over gaat in:

$$\bar{\ell}_{b^1} = \sum_{z \in B^2 - L_{B^2}} q(z) + q(y) = \sum_{z \in B^1 - L_{B^1}} q(z) \qquad (4.13)$$

Hiermee is de induktie stap bewezen.

De basis stap is ook waar daar de unieke n-boom met n bladeren een gemiddelde segmentlengte van één heeft, en slechts één inwendige knoop w, met q(w)=1.

∴

4.1.3 Enkele voorbeelden.

In figuur 4.3 zijn vier voorbeelden van (V_B, V_C, f) kodes gegeven. Zie ook Verhoeff [1977]. Alle voorbeelden hebben betrekking op een geheugenloze binaire bron S met p(0)=0,25 en p(1)=0,75. Dus $H(S) \cong 0{,}811$, $A=\{0,1\}$, $D=\{\alpha,\beta\}$.

Voorbeeld a is een Huffman kode, d.w.z. B^1 is een gebalanceerde 2-boom en C^1 de komplete 2-boom die $\bar{\ell}_c$ minimaliseert.
Een gebalanceerde n-boom is kompleet en heeft alle bladeren op dezelfde afstand van de wortel.

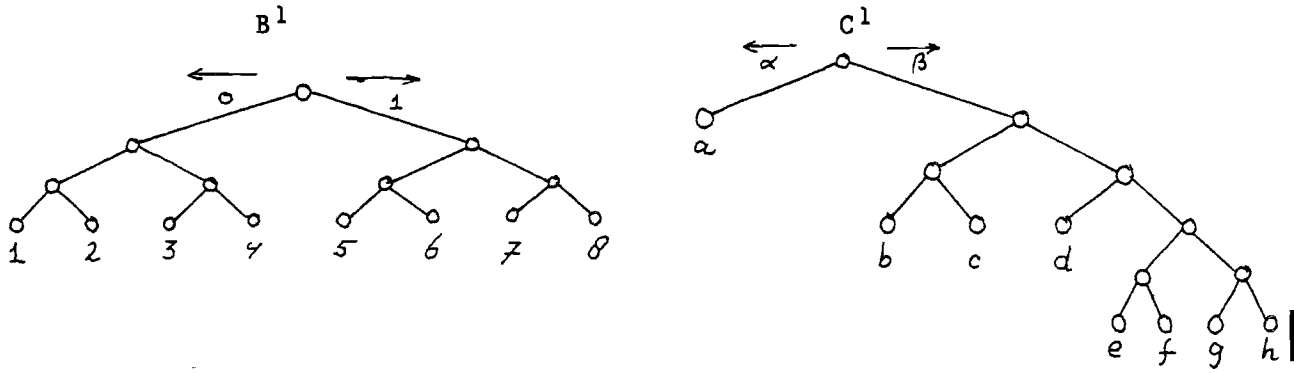
Voorbeeld b is een Tunstall/Verhoeff kode, d.w.z. C^2 is een gebalanceerde 2-boom, en B^2 is zo gekozen dat $\bar{\ell}_b$ maximaal is.

Voorbeeld c is de kode bestaande uit de optimale segmentboom B^3 $(=B^2)$ gevolgd door de kodeboom C^3 verkregen met de Huffman procedure. We zien dat, in dit geval, deze kombinatie niet alleen slechter is dan de beste variabele naar variabele kombinatie (d) maar zelfs slechter dan de Huffman kode (a).

Voorbeeld d is de optimale kombinatie van twee bomen B^4 en C^4 met elk acht bladeren. Deze is gevonden door alle mogelijkheden af te gaan.

Voorbeeld a

Huffman kode

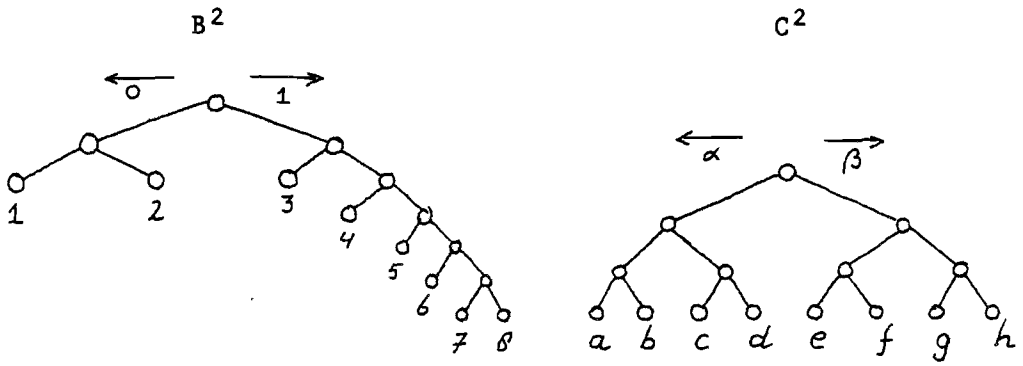


$$f^1: \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ e & f & g & b & h & c & d & a \end{pmatrix}$$

$\rho^1 = 0{,}823$

Voorbeeld b

Tunstall/Verhoeff kode



$$f^2: \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ a & b & c & d & e & f & g & h \end{pmatrix}$$

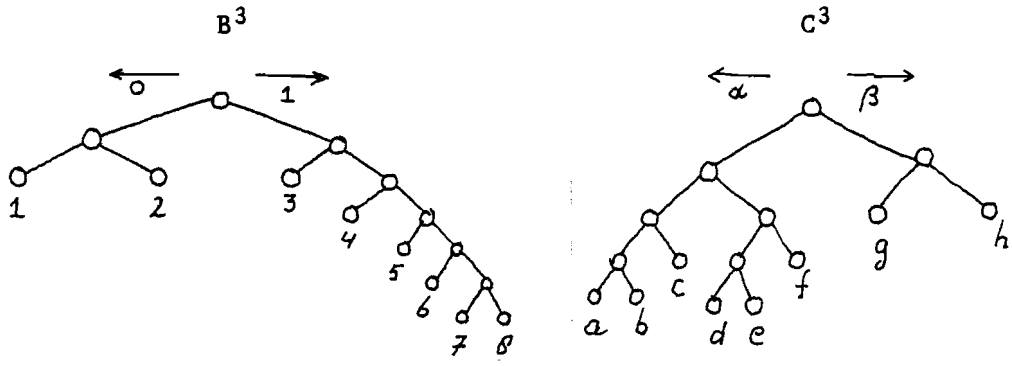
$\rho^2 = 0{,}848$

Figuur 4.3 deel a en b.

Voorbeeld c

De optimale segment boom B^2,

gevolgd door de Huffman procedure.

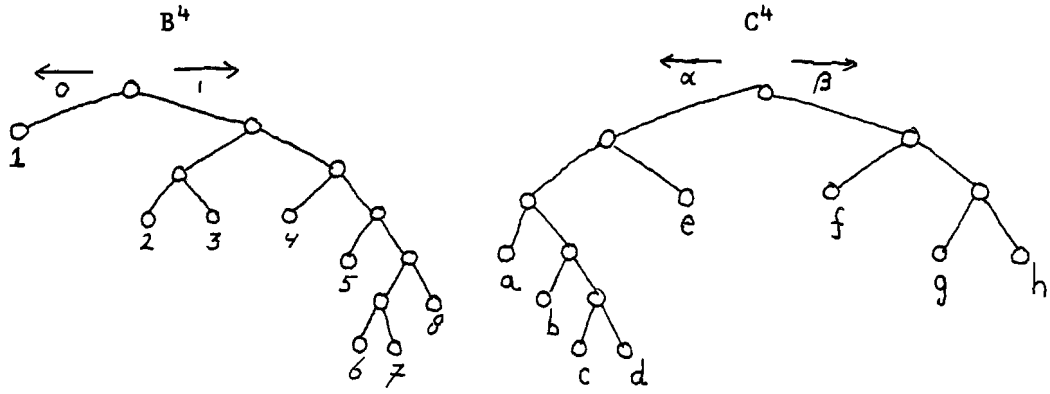


$$f^3: \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ a & g & h & c & d & e & b & f \end{pmatrix}$$

$\rho^3 = 0{,}829$

Voorbeeld d

De optimale acht woorden kode.



$$f^4: \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ f & d & h & g & a & c & b & e \end{pmatrix}$$

$\rho^4 = 0{,}816$

Figuur 4.3. deel c en d.

4.1.4 Shannon's bronkoderings stelling.

Uit Gallager [1968, stelling 3.3.1] volgt:

Voor elke segmentboom B bestaat er een kodeboom C, zodat

$$H(X) \leq \bar{l}_c \leq H(X) + 1 \qquad (4.14)$$

Met (4.5) en (4.9) gaat (4.14) over in:

$$H(B) \leq \rho \leq H(B) + \frac{1}{\bar{l}_b} \qquad (4.15)$$

We zien dat, indien het aantal elementen in V_B toeneemt, zodat ook de gemiddelde segmentlengte groeit, dit kodesysteem $H(B)$ steeds dichter nadert.
Een gevolg van de nu volgende stelling is dat $H(B)$ voor een geheugenloze bron S gelijk is aan $H(S)$, de bronentropie.

Stelling 4.1: Voor een geheugenloze bron S en de stochastische variabele X gedefiniëerd uit S m.b.v. een komplete n-boom B geldt:

$$H(X) = \bar{l}_b \times H(S) \qquad (4.16)$$

Bewijs:

Stel dat (4.16) waar is voor alle stochastische variabelen X beschreven door een willekeurige geheugenloze bron S en een komplete n-boom B met niet meer dan $m = \alpha(n-1)+1$ bladeren.
$\alpha \in \mathbb{N}$.
Laat B^1 weer een komplete n-boom zijn met $m+n-1$ bladeren. Kies een blad z_1 met maximale lengte en noem y de vader van z_1. De opvolgers van y (bladeren) heten $z_1, z_2, .., z_n$.
X^1 ontstaat met S en B^1.

Nu geldt:

$$H(X^1) = -\sum_{z \in L_{B^1}} q(z)\log\, q(z) =$$

$$= -\sum_{z \in L_{B^1} - \{z_1,..,z_n\}} q(z)\log\, q(z) - \sum_{i=1}^{n} q(z_i)\log\, q(z_i) \qquad (4.17)$$

ook geldt:

$$-\sum_{i=1}^{n} q(z_i)\log\, q(z_i) = -q(y)\log\, q(y) + q(y)H(S) \qquad (4.18)$$

Weer ontstaat B^2 uit B^1 door weglaten van z_1, z_2 enz.
Uit (4.17) en (4.18) volgt:

$$H(X^1) = -\sum_{z \in L_{B^2}} q(z)\log\, q(z) + q(y)H(S)$$

$$= (\bar{\ell}_{b^2} + q(y))H(S) = \bar{\ell}_{b^1} H(S) \qquad (4.19)$$

In deze laatste stappen wordt gebruik gemaakt van de induktie veronderstelling en lemma 4.1.

Met (4.19) is de induktie stap bewezen. De basisstap is triviaal.

∴

We zijn nu klaar voor het bewijs van:

Stelling 4.2: (bronkoderings stelling).

Laat $S = (A,P)$ een geheugenloze diskrete bron zijn met een eindig alfabet A. Noteer B^i voor een komplete n-boom met $i(n-1)+1$ bladeren. $i \in \mathbb{N}$.
$\bar{\ell}_{b^i}$ is de gemiddelde segmentlengte van B^i gewogen met S.

Voor elke oneindige rij $\{B^i\}_{i=0}^{\infty}$ met $\bar{\ell}_{B^i} \to \infty$ als $i \to \infty$, bestaat een oneindige rij $\{(C^i, f_i)\}_{i=0}^{\infty}$, zodat ρ_i, de reduktiefaktor van de kode $(L_B, L_C, f)_i$ op S, nadert tot $H(S)$ voor $i \to \infty$.

Het bewijs volgt direkt uit definitie 4.5, stelling 4.1 en formule (4.15).

Zoals deze stelling laat zien is de exakte vorm van de rij $\{B^i\}_{i=0}^{\infty}$ niet van groot belang. De stelling legt de nadruk op de juiste keuze van $\{(C^i, f_i)\}_{i=0}^{\infty}$. De optimale keuze wordt gegeven door Huffman's algoritme. Zie b.v. Gallager [1968].
Natuurlijk is de keuze van de boom B^i wel van belang om, voor elke i, ρ_i te optimaliseren.

Formule (4.15) is sterker dan de overeenkomstige stelling 3.3.2 in Gallager [1968] in die zin dat in Gallager de segmentboom B gebalanceerd is, terwijl (4.15) slechts eist dat B kompleet is. Echter, stelling 4.2 spreekt slechts over geheugenloze bronnen waar Shannon's variabele lengte bronkoderingsstelling over stationaire bronnen gaat. Zie Gallager [1968]. Dat dit ook niet zonder meer mogelijk is wordt aannemelijk gemaakt in hoofdstuk 4.4.2.

Zoals het Huffman algoritme de optimale kodeboom C zoekt bij konstante lengte segmenten, zoekt de Tunstall/Verhoeff procedure de maximale segmentboom bij konstante kodewoord lengte.

Algoritmes voor het efficiënt genereren van optimale variabel naar variabel segment kodes bestaan nog niet.

4.1.5 Het Tunstall/Verhoeff algoritme.

Het Tunstall/Verhoeff algoritme ontwerpt optimale variabel naar vast segment kodes voor geheugenloze diskrete bronnen S. Voor dit type kodes is bij een bepaald aantal kodewoorden de kode optimaal als de gemiddelde segmentlengte maximaal is. In [1967] geeft Tunstall een efficiënt recursief algoritme voor de maximale segmentboom. Verhoeff, in [1977], vindt dit algoritme opnieuw en bewijst ook dat deze kodes, in de limiet voor het aantal kodewoorden naar oneindig, optimaal zijn, d.w.z. de bronentropie bereiken.
Het Tunstall/Verhoeff algoritme speelt een grote rol in de volgende delen van dit verslag.

In het vervolg wordt telkens verondersteld dat de komplete n-boom B gewogen is met een geheugenloze diskrete bron S.

Definitie: De <u>maximale uitbreiding</u> van een komplete n-boom B met k bladeren is de komplete n-boom met k+n-1 bladeren die uit B ontstaat door het uitbreiden met n bladeren van het blad z met het maximale gewicht q(z).

Definitie: De <u>maximale (S,k,n) boom</u> is gedefiniëerd als de komplete n-boom B met k bladeren die $\bar{l}_b$ maximaliseert over de verzameling van alle komplete n-bomen met k bladeren, gewogen met bron S.

Zowel de maximale uitbreiding als de maximale (S,k,n) boom behoeven niet uniek te zijn, doch bestaan altijd.

Het algoritme:

Vorm de segmentboom B met het gewenste aantal bladeren (= m) door het herhaald toepassen van de maximale uitbreiding, beginnende met de unieke maximale (S,n,n) boom.

Voor m moet gelden: $m = \alpha(n-1)+1$, $\alpha \in \mathbb{N}$.
Daar de kodeboom C gebalanceerd moet zijn, moet voor m ook gelden: $m = n^{\ell}$, $\ell \in \mathbb{N}$.
Gelukkig is voor elke ℓ een α te vinden, zodat m aan beide eisen voldoet. n.l. kies $\alpha = \sum_{i=0}^{\ell-1} n^i$.

Kies een willekeurige inverteerbare afbeelding van L_B naar L_C.

De Tunstall/Verhoeff kode is gegeven door (L_B, L_C, f).

Nu volgen de stellingen die de optimaliteit van het algoritme tonen.
Eerst een lemma. Dit lemma geeft een eenvoudige test voor het bepalen of een boom maximaal is.

Lemma 4.2: Als voor een komplete n-boom B voor alle k bladeren $z \in L_B$ en alle inwendige knopen $y \in B-L_B$ geldt: $q(z) \leq q(y)$,
dan is B een maximale (S,k,n) boom. En andersom.

Het bewijs van dit lemma en van de volgende twee stellingen worden in bijlage A behandeld.

Stelling 4.3: Als B een maximale (S,k,n) boom is en B^1 ontstaat uit B door een maximale uitbreiding, dan is B^1 een maximale (S,k+n-1,n) boom.

Uit deze stelling volgt dat het Tunstall/Verhoeff algoritme maximale (S,k,n) segmentbomen genereert.

Stelling 4.4: Noem $(V_B, V_C, f)_M$ de Tunstall/Verhoeff kode met M segmenten voor een geheugenloze diskrete bron S. M is zodanig dat C_M een gebalanceerde boom is.
Nu geldt:

$$\lim_{M\to\infty} \frac{^n\log M}{\bar{\ell}_b} = H(S).$$

Met de limiet $M\to\infty$ wordt bedoeld de limiet $\ell\to\infty$, met $M = n^\ell$ en $\ell \in \mathbb{N}$.

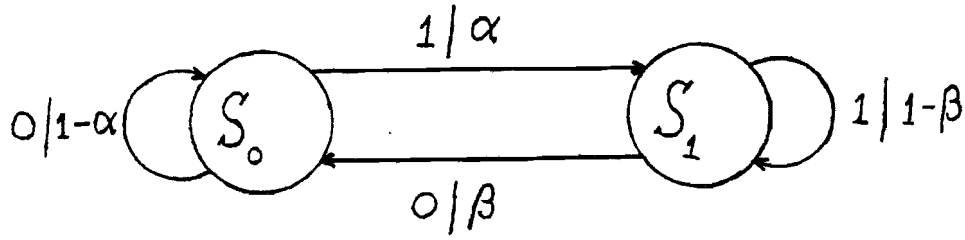
Zowel Tunstall als Verhoeff vergelijken met enkele voorbeelden hun kodes met Huffman's vast naar variabel schema. Bij een gelijk aantal kodewoorden is de kompressie van het Huffman schema meestal beter.
Een mogelijk voordeel van het 'nieuwe' schema is dat de vaste kodewoord lengte de synchronisatie kan vereenvoudigen.

4.2 Het segment koderen van bronnen met geheugen.

Indien de bron algemeen stationair ergodisch is, ondervinden we nieuwe problemen bij het vinden van optimale segment bomen.
Laten we aan de hand van een eenvoudig voorbeeld eens kijken wat die problemen zijn.

4.2.1 Het eerste voorbeeld.

S is een eerste orde binaire Markov bron. Zie figuur 4.4.
De overgangswaarschijnlijkheden zijn $p(1|0) = \alpha$, $p(0|1) = \beta$.



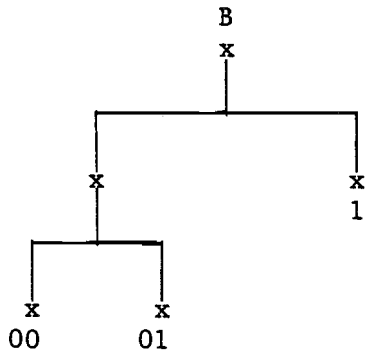
Figuur 4.4.

De stationaire verdeling is:

$$p_0 = \frac{\beta}{\alpha+\beta},\ p_1 = \frac{\alpha}{\alpha+\beta} \tag{4.20}$$

Laten we veronderstellen, dat $\beta > \alpha$ (4.21)
De bron heeft een voorkeur voor nullen.

Het Tunstall/Verhoeff algoritme genereert de komplete 2-boom B met drie bladeren, zoals gegeven in figuur 4.5.



Figuur 4.5.

In de eerste, naïeve, beschouwing berekenen we de segment kansen als:

$$p("00") = p_0 p(0|0)^2 + p_1 p(0|1)p(0|0) = \frac{(1-\alpha)\beta}{\alpha+\beta}$$

$$p("01") = p_0 p(0|0)p(1|0)+p_1 p(0|1)p(1|0) = \frac{\alpha\beta}{\alpha+\beta} \quad (4.22)$$

$$p("1") = p_0 p(1|0) + p_1 p(1|1) = \frac{\alpha}{\alpha+\beta}$$

De gemiddelde segmentlengte is:

$$\bar{\ell}_b = 1 + \frac{\beta}{\alpha+\beta} \quad (4.23)$$

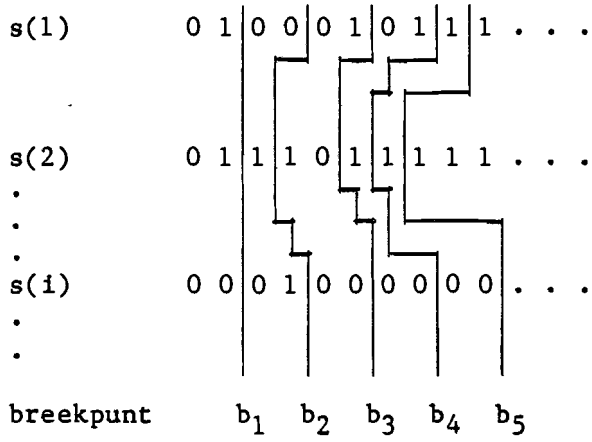
Echter, wat is nu eigenlijk de kans om in toestand 0 aan een nieuw segment te beginnen? We hebben verondersteld dat dit de stationaire kans p_0 is. Maar, we beginnen slechts dan in toestand 0 als het vorige segment 00 was, dus

$$Pr\{\text{start in toestand } 0\} = \frac{(1-\alpha)\beta}{\alpha+\beta} \quad (4.24)$$

(4.24) is niet gelijk aan p_0 volgens (4.20) dus de gehele analyse is fout!

Wat is er nu aan de hand?

De bron is stationair, d.w.z. dat over het ensemble van uitkomsten de kansverdeling invariant is, echter, B vormt de segmenten afhankelijk van de inhoud, dus over het hele ensemble staan de 'breekpunten' niet op dezelfde plaatsen. Zie figuur 4.6. We zijn geïnteresseerd in de kansverdeling direkt na een breekpunt.



Figuur 4.6.
De s(j)'s zijn ensemble uitkomsten.
De b_i's zijn breekpunten.

Laten we die kans gaan berekenen. Daar $\beta > \alpha$, zie (4.21), houdt de Tunstall/Verhoeff boom dezelfde vorm. (Figuur 4.5).

We schrijven:

$$p(\text{in } 0) \triangleq p(\text{"00"})$$

$$p(\text{in } 1) \triangleq p(\text{"01"}) + p(\text{"1"})$$

$$p(\text{"1"}) = p(\text{in } 0)p(1\,|\,0) + p(\text{in } 1)p(1\,|\,1)$$

$$= \frac{\alpha(2-\alpha) - \alpha\beta}{\alpha(2-\alpha) + (1-\alpha)\beta} \qquad (4.25)$$

$$p(\text{"01"}) = p(\text{in } 0)p(0\,|\,0)p(1\,|\,0) + p(\text{in } 1)p(0\,|\,1)p(1\,|\,0)$$

$$= \frac{\alpha\beta}{\alpha(2-\alpha) + (1-\alpha)\beta}$$

$$p(\text{"00"}) = p(\text{in } 0)p(0\,|\,0)^2 + p(\text{in } 1)p(0\,|\,1)p(0\,|\,0)$$

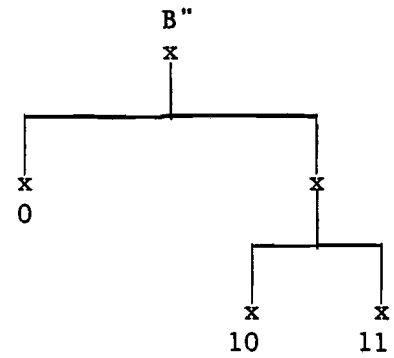
$$= \frac{(1-\alpha)\beta}{\alpha(2-\alpha) + (1-\alpha)\beta}$$

En nu volgt:

$$\bar{\ell}_b' = 1 + \frac{\beta}{\alpha(2-\alpha) + (1-\alpha)\beta} \qquad (4.26)$$

We zullen de, goede, segmentlengte (4.26) vergelijken met (4.23). Er volgt:

$$\begin{aligned} \bar{\ell}_b' &> \bar{\ell}_b \quad \text{als } \alpha+\beta > 1 \\ \bar{\ell}_b' &= \bar{\ell}_b \quad \text{als } \alpha+\beta = 1 \\ \bar{\ell}_b' &< \bar{\ell}_b \quad \text{als } \alpha+\beta < 1 \end{aligned} \qquad (4.27)$$



Figuur 4.7.

Als we hetzelfde uitrekenen voor de andere mogelijke boom B", zie figuur 4.7, dan volgt:

$$\bar{\ell}_b = 1 + \frac{\alpha}{\beta(2-\beta) + \alpha(1-\beta)} \qquad (4.28)$$

Vergelijken we (4.26) met (4.28) dan zien we dat in het beschouwde geval, zie (4.21), B' beter is, geheel volgens verwachting.

De vraag is nu: zal de maximale uitbreiding, waarbij we elke keer alle kansen geheel opnieuw moeten berekenen, van een maximale boom opnieuw een maximale boom opleveren? Vergelijk stelling 4.3.

Het is mij niet gelukt dit, of het tegendeel, te bewijzen.
Zelfs als dit zou lukken dan is de berekening van de maximale uitbreiding zo complex dat het algoritme niet efficiënt zou zijn. Bij k bladeren moet een lineair stelsel van k vergelijkingen opgelost worden, en k is een groot geheel getal.

Een heel ander probleem is, wat te doen als niet aan elk blad eenduidig een brontoestand kan worden toegekend?

Nog een ander 'probleem' komt naar voren bij de volgende beschouwing.

4.2.2 Het tweede voorbeeld.

In Csiszar en Körner [1981, hoofdstuk 4] wordt in een opgave een voorbeeld gegeven van een stationaire bron waarvoor een kodering te vinden is die een betere reduktie realiseert dan de entropie van de bron aangeeft. Een uitwerking van dat probleem volgt hieronder.

X en Y zijn twee diskrete geheugenloze bronnen met hetzelfde alfabet A. We noteren :

$$H(X) = H_1 \tag{4.29a}$$

$$H(Y) = H_2 \tag{4.29b}$$

Er geldt:

$$H_1 \neq H_2 \tag{4.29c}$$

U is een binaire stochastische variabele met waarden uit $\{1,2\}$. U, X en Y zijn onderling onafhankelijk en U's kansfunktie is gegeven als:

$$P\{U=1\} = \alpha$$

$$P\{U=2\} = 1-\alpha, \quad \text{met } \alpha \in (0,1)$$

Met U, X en Y definiëren we een stationaire bron Z volgens:

$$Z = \begin{cases} X \text{ , als } U = 1 \\ Y \text{ , als } U = 2 \end{cases}$$

We zullen nu eerst een expressie afleiden voor $H_\infty(Z)$:

$$H_\infty(Z) = \lim_{n\to\infty} \frac{1}{n} H(Z_1^n) = \lim_{n\to\infty} \frac{1}{n} [H(Z_1^n|U) + I(U;Z_1^n)]$$

En met

$$\frac{1}{n} H(Z_1^n|U) = \alpha H_1 + (1-\alpha)H_2$$

en

$$0 \leq I(U;Z_1^n) \leq \log \|U\| = 1$$

volgt:

$$H_\infty(Z) = \alpha H_1 + (1-\alpha)H_2 = E_U\{H_u\} \qquad (4.30)$$

Voor de bronnen X en Y ontwerpen we Tunstall/Verhoeff kodes met elk M kodewoorden.
We noteren:

$\bar{\ell}_{b1}$: de gemiddelde segmentlengte van de kode voor X.

$\bar{\ell}_{b2}$: de gemiddelde segmentlengte van de kode voor Y.

Gegeven een datarij $Z_1^n=z_1^n$, nemen we de beslissing of z_1^n door X of door Y gegenereerd is. Daar $H_1 \# H_2$ zal, voor n groot genoeg, deze beslissing met willekeurig kleine foutkans genomen kunnen worden. In wat nu volgt stellen we $n=\infty$ en nemen we aan dat we foutloos beslissen.

De reduktiefaktor van dit systeem wordt gegeven door:

$$\rho_M = \frac{\log M}{E_U\{\bar{\ell}_{bu}\}} \tag{4.31}$$

Daar de twee kodes Tunstall/Verhoeff kodes zijn geldt:

$$\bar{\ell}_{bu} = \frac{\log M}{H_u} - \delta_u, \quad \text{met } \lim_{M\to\infty} \delta_u = 0, \quad u = 1,2 \tag{4.32}$$

En dus

$$E_U\{\bar{\ell}_{bu}\} = \log M \left\{\frac{\alpha}{H_1} + \frac{1-\alpha}{H_2} - \frac{\alpha\delta_1+(1-\alpha)\delta_2}{\log M}\right\}$$

Zodat volgt:

$$\rho = \lim_{M\to\infty} \rho_M = \frac{1}{E_U\{\frac{1}{H_u}\}} \tag{4.33}$$

Met het lemma 4.3, zie hieronder, volgt, samen met (4.29c), (4.30) en (4.33)

$$\rho < H_{\infty}(Z) \tag{4.34}$$

Lemma 4.3.

Voor een diskrete stochastische variabele $\underline{x}$ met eindig veel waarden en $p(\underline{x}=x)>0$, geldt voor elke positief reëeltallige funktie $f(\underline{x})$:

$$\frac{1}{E\{f(\underline{x})\}} \leq E\left\{\frac{1}{f(\underline{x})}\right\}$$

Met gelijkheid slechts dan als $f(\underline{x})$ een konstante funktie is.

Het bewijs van dit lemma staat in bijlage B.

Zouden we in plaats van een Tunstall/Verhoeff kode gebruik hebben gemaakt van Huffman kodes voor het koderen van de bronnen X en Y, dan verkrijgen we het volgende:

$$\rho_M = \frac{E_U\{\bar{\ell}_{c_u}\}}{\log M}$$

Zodat volgt:

$$\rho = \lim_{M\to\infty} \rho_M = E_U\{H_u\} = H_{\infty}(Z)$$

We zien dat voor Huffman kodes $H_{\infty}(Z)$ wel het maximaal haalbare geeft.

4.2.3 Diskussie van het resultaat.

Voor bronnen met geheugen is het niet gelukt om goede Tunstall achtige kodes te maken. Het eerste voorbeeld maakt ons duidelijk dat de kansen aan de knopen van een segmentboom niet alleen van de

bron, maar ook van de boom zelf afhankelijk zijn. Hierdoor is het efficiënt vinden van maximale bomen een nog open probleem.
Het voorbeeld van Csiszar en Körner laat zien dat voor een algemene stationaire bron de entropie niet altijd een zinvol begrip is. Echter, bron Z is wel stationair, doch beslist niet ergodisch. De vraag blijft of voor ergodische bronnen (4.34) ook kan optreden.
In een poging hier een antwoord op te kunnen geven, is in bijlage D een kodeersysteem voor o.a. bekende eindige orde Markov bronnen gegeven. Het systeem haalt echter exakt de entropie en geeft dus geen uitsluitsel. Daar het een konstruktief schema is, is het toch in dit verslag opgenomen.

4.3 Het Ziv-Lempel algoritme opnieuw.

In dit hoofdstuk wordt het Ziv-Lempel algoritme op twee verschillende manieren beschouwd. Zo wordt de werking van het algoritme verduidelijkt. We beschouwen alleen het binaire geval, echter voor elke andere alfabetgrootte gaat een gelijksoortig verhaal op.
Eerst wordt Langdon's beschrijving van het algoritme kort behandeld. Hieruit wordt duidelijk welk model het algoritme van de bron maakt. Daarna volgt een aanpassing van Ziv en Lempel's schema, die op een iets andere beschouwing berust.

4.3.1 Beschrijving met inkomplete bomen.

In [1981-a] stellen Rissanen en Langdon dat alfabet extensie voor bronkodering niet noodzakelijk is. Onder andere bewijzen ze dat voor elk schema met extensie een equivalent en niet komplexer schema zonder extensie bestaat.
Het Ziv-Lempel algoritme, zie hoofdstuk 2.2, past wel alfabet extensie toe. Langdon geeft in [1983] een symboolsgewijze beschrijving van dit algoritme. Zijn beschrijving is een adaptief kontekst koderings algoritme, zie hoofdstuk 2.3. Deze beschrijving laat duidelijk zien hoe het Ziv-Lempel schema een bron model vormt.

4.3.1.1 De inkomplete Ziv-Lempel boom.

De segmentverzameling van het Ziv-Lempel schema definiëert eenduidig een n-boom. Deze boom is i.h.a. niet kompleet en de segmentverzameling bestaat uit alle knopen van de boom.
Een voorbeeld zal de bedoeling duidelijk maken.

Laat de binaire bronrij s beginnen met

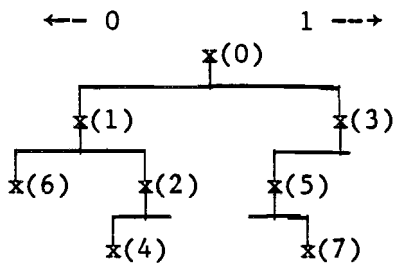
00110 10100 0101. .. (4.35)

De segmentverzameling is dan:

segment : λ, 0, 01, 1, 010, 10, 00, 101

index : 0, 1, 2, 3, 4, 5, 6, 7 (4.36)

De Ziv-Lempel boom is gegeven in figuur 4.8.



Figuur 4.8. De Ziv-Lempel boom.

Van komplete n-bomen zijn de knopen ðf bladeren ðf inwendige knopen. Inkomplete bomen, zoals in figuur 4.8, bezitten nog een derde soort knopen, die semibladeren genoemd worden. b.v. knoop 2 in figuur 4.8.
Een semiblad heeft k opvolgers, met $0 < k < n$.

4.3.1.2 Het kontekst koderen.

We kunnen figuur 4.8 beschouwen als de kontekst voor het koderen van het volgende segment, als op een zinvolle manier aan elke knoop z van de boom een konditionele kans $p(x|z)$, x uit het bronalfabet, wordt toegekend.

Langdon kent de volgende kansen toe aan de knopen van de Ziv-Lempel boom:

Noem het aantal knopen in de subboom met z als wortel c(z). Dus als z een blad is dan c(z) = 1, en $c(\lambda) = \sigma$, het totale aantal verwerkte segmenten.

Definiëer

$$p(z) \stackrel{\Delta}{=} \frac{c(z)}{c(\lambda)} = \frac{c(z)}{\sigma} \tag{4.37}$$

dan

$$p(x|z) = \frac{p(zx)}{p(z)} = \frac{c(zx)}{c(z)} \tag{4.38}$$

Stel dat s, (4.35), voortgezet wordt met 011...., dan kodeert het Ziv-Lempel schema het volgende segment 011 als het paar $(01,1)$, zie hoofdstuk 2.2.1, en gaat over naar een nieuwe kontekst.
In de symboolsgewijze beschrijving wordt elk symbool apart gekodeerd. Daar de segmentverzameling niet proper is moet voor de symboolsgewijze beschrijving een nieuw symbool, de komma, geïntroduceerd worden. Deze komma geeft het einde van een segment aan. De ontvangst van de komma resulteert niet in het genereren van data doch laat de ontvanger over gaan naar een nieuwe kontekst.

Daar "01" een semiblad is, zie figuur 4.8, moet een komma toegevoegd worden. In de symboolsgewijze beschrijving wordt dus verzonden:

"0" : in de kontekst λ,
"1" : in de kontekst "0",
"," : in de kontekst "01",
"1" : in de kontekst "01,", dit is het tweede deel van het kode paar.

De kode voor "01," heeft dus dezelfde funktie als de kode voor de index van "01" in het Ziv-Lempel algoritme.

In het Ziv-Lempel algoritme zijn ceil(log σ) bits nodig voor het koderen van de index. Het nu volgende laat zien dat de symboolsgewijze beschrijving ook log σ bits nodig heeft.

Laat e het segment voorstellen waarvan de index naar de ontvanger verzonden moet worden. Als e een blad van de Ziv-Lempel boom is, dan weten we dat het volgende symbool altijd de komma is. Is e een semiblad dan is de komma een van de mogelijke gebeurtenissen en heeft een kans $p(,|e) = 1/c(e)$.

Of e nu een blad of een semiblad is, in beide gevallen volgt:

$$p(e,) = p(e)p(,|e) = 1/\sigma \qquad (4.39)$$

Zie hoofdstuk 2.3.3 formule (2.26), de ideale kodewoordlengte voor "e," is gegeven als

$$I(e,) = \log \sigma \qquad (4.40)$$

hetgeen gelijk is aan wat het Ziv-Lempel algoritme nodig heeft. (Op de afronding na).

4.3.1.3 Konklusies.

De symboolsgewijze beschrijving van Langdon is een beschrijving van het door het Ziv-Lempel algoritme gegenereerde kontekst model.
In de segmentlijst blijven ook segmenten bestaan die, geheel of gedeeltelijk, niet meer gebruikt zullen worden. Voor elke knoop, of kontekst, z wordt een kans van 1/c(z) gereserveerd voor alle nog niet opgetreden gebeurtenissen. Dit blijft zo, ook al zijn

alle bij z mogelijke gebeurtenissen al opgetreden. Dit laatste is duidelijk zichtbaar in Langdon's beschrijving evenals de, in de limiet verwaarloosbare, invloed hiervan op de reduktiefaktor.

4.3.2 Beschrijving met komplete n-bomen.

Niets let ons om de onbruikbare delen uit de segmentlijst te verwijderen. We zullen het algoritme aanpassen door het volgende segment niet te koderen als een paar (i,a) doch direkt te koderen als de index in de lijst van mogelijke nieuwe segmenten.

Een voorbeeld.

s is zoals gegeven in (4.35).

In de onderstaande figuur 4.9 staan achtereenvolgens de Ziv-Lempel segmentlijst en de mogelijke nieuwe segmentenlijst voor het algoritme losgelaten op rij s.

slag#	Ziv-Lempel lijst	mogelijke segmenten lijst
0	λ	0, 1
1	λ, 0	00, 01, 1
2	λ, 0, 01	00, 010, 011, 1
3	λ, 0, 01, 1	00, 010, 011, 10, 11
4	λ, 0, 01, 1, 010	00, 0100, 0101, 011, 10, 11
5	λ, 0, 01, 1, 010, 10	00, 0100, 0101, 011, 100, 101, 11
6	λ, 0, 01, 1, 010, 10, 00	000, 001, 0100, 0101, 011, 100, 101, 11
7	λ, 0, 01, 1, 010, 10, 00, 101	000, 001, 0100, 0101, 011, 100, 1010, 1011, 11

Figuur 4.9. De twee segmentlijsten.

In de i^e slag van het algoritme wordt gebruik gemaakt van de respektievelijke segmentlijst die achter de $(i-1)^e$ slag staan.
De Ziv-Lempel lijst voor de i^e slag bevat i elementen, de mogelijke segmentenlijst hiervoor bevat er i+1. Echter, aan de index van de Ziv-Lempel lijst wordt nog één bronsymbool toegevoegd hetgeen het mogelijke aantal kodewoorden verdubbelt. Dus het Ziv-Lempel algoritme heeft voor de i^e slag de keuze uit 2i kodewoorden, elk ter lengte ceil(log 2i) en waarvan er i-1 niet gekozen kunnen worden. Het aangepaste algoritme heeft de keuze uit de i+1 mogelijke kodewoorden ter lengte ceil(log (i+1)). Indien de alfabetgrootte toeneemt wordt het verschil minder spektakulair.

4.3.2.1 Het aangepaste algoritme.

In welke vorm kan het aangepaste algoritme het beste gegeven worden?

Observeer dat elke lijst van mogelijke segmenten kompleet en proper is, en dus te beschrijven is als de lexicografisch geordende verzameling van bladeren van een komplete 2-boom.
Ma, zie referentie 6 in Rissanen [1983], legt een eenvoudig verband tussen deze bomen en het Tunstall/Verhoeff algoritme.
We beginnen, in het binaire geval, met de boom B^0 bestaande uit de wortel en twee bladeren. Aan elke knoop van de boom kennen we een gewicht toe. De wortel van de B^0 krijgt gewicht $w(\lambda)=2$, de bladeren elk gewicht 1.

B^0 gebruiken we om het eerste segment af te passen. Van alle bezochte knopen, vanaf de wortel tot en met het segment afhankelijke blad, worden de bijbehorende gewichten één opgehoogd. De nieuwe boom B^1 wordt uit B^0 gevormd door het unieke blad met gewicht 2 uit te breiden met twee nieuwe bladeren, elk met gewicht 1. B^1 gebruiken we voor het vinden van het tweede segment en het vormen van B^2, enz.

De zo gevormde segmenten zijn dezelfde als die door het Ziv-Lempel algoritme gegenereerd worden.
Het is duidelijk dat voor elke boom B^i het gewicht van een knoop z aangeeft hoeveel bladeren de subboom, met z als wortel, bevat.
Het aantal keren dat z (z ongelijk aan λ) een prefix is van de tot dan toe gevonden segmenten wordt gegeven door $w(z)-1$. Het totale aantal verwerkte segmenten is $w(\lambda)-2$.

4.3.2.2 Het bron model.

We kunnen met behulp van deze bomen een informatiebron definiëren voor elk te vormen segment, en wel alsvolgt:

B^{i-1} is de boom, die gebruikt wordt om het i^e segment te vinden. Weer volgens hoofdstuk 4.1.2 kunnen we een stochastische variabele X_i definiëren. Zie figuur 4.2.
De bijbehorende kansfunktie $P_i(X_i)$ bepalen we alsvolgt:

Aan elke knoop z van B^{i-1} kennen we een nieuw getal q(z) toe volgens

$$q(z) \triangleq \frac{w(z)}{w(\lambda)} \tag{4.41}$$

Er geldt:

$$q(z) = q(z0) + q(z1) \text{ voor alle inwendige knopen } z.$$

En dus kunnen we q(z) beschouwen als de door het model gedefiniëerde kans op z.

Ook definiëren we

$$q(x|z) \triangleq \frac{q(zx)}{q(z)} \text{ voor alle } z \in B^{i-1} \text{ en } x \in \{0,1\}, \text{ zo dat } zx \in B^{i-1}. \tag{4.42}$$

Dit is de, model afhankelijke, konditionele kans op x gegeven z.

De kansfunktie $P_i(X_i)$ wordt aldus gegeven door

$$P_i(X_i = z) = q(z) \quad z \in L_{B^{i-1}} \text{ (blad van } B^{i-1}) \tag{4.43}$$

Echter, voor elke boom B^i geldt w(z)=1 voor elk blad z dus

$$P_i(X_i) = \frac{1}{w(\lambda)} \qquad z \in L_{B^{i-1}} \tag{4.44}$$

Daar $w(\lambda)$ gelijk is aan 2 plus het aantal verwerkte segmenten volgt

$$w(\lambda) = (i+1) \tag{4.45}$$

en dus

$$P_i(X_i) = \frac{1}{i+1} \tag{4.46}$$

Daar alle kansen even groot zijn hebben de optimale kodewoorden dezelfde lengte ceil(log (i+1))

Stel dat van rij s m segmenten gevonden zijn dan is de lengte van de koderij c gegeven door (zie (2.26))

$$\ell(c) = \sum_{i=1}^{m} \text{ceil}(\log (i+1)) \tag{4.47}$$

Vergelijk dit met de lengte van de korresponderende koderij c' van het Ziv-Lempel algoritme: (zie hoofdstuk 2.2.1)

$$\ell(c') = \sum_{i=1}^{m} \text{ceil}(\log 2i) = \sum_{i=1}^{m} \text{ceil}(\log i) + m \tag{4.48}$$

Dus $\ell(c')$ is bijna m bits langer, oftewel de aanpassing is bijna één bit per segment beter. Echter, daar de segmenten steeds langer worden is m in de limiet verwaarloosbaar t.o.v. de koderijlengte.

De Ziv-Lempel bomen B^i zijn zo opgezet dat telkens geldt

$$w(z) = 1 \text{ voor alle bladeren } z \text{ van } B^i.$$

$$w(y) > 1 \text{ voor alle inwendige knopen } y \text{ van } B^i.$$

Oftewel, er geldt:

$$q(z) < q(y) \text{ voor elk blad } z \text{ en elke inwendige knoop } y.$$

dus (zie lemma 4.2) met respekt tot de model gedefiniëerde kansen $q(.)$ hebben alle bomen de optimale vorm, d.w.z. ze maximaliseren de gemiddelde segmentlengte. vgl. het Tunstall/Verhoeff schema.

4.4 Universele segmentkodes.

Het Ziv-Lempel algoritme is een voorbeeld van een universele "segment-achtige" kode.
Een nadeel van dit algoritme is dat de komplexiteit van het gevormde model blijft groeien, onafhankelijk van de komplexiteit van de te koderen bron.
In dit hoofdstuk wordt het gedrag bekeken van universele segmentkodes met een begrenste komplexiteit.

4.4.1 Het basis algoritme.

De voorgestelde kodes zijn komplete en propere segmentkodes $(L_B, L_C, f)_K$, met

- K het aantal kodewoorden.
- B_K een komplete n-boom met K bladeren.
- C_K een gebalanceerde n-boom met K bladeren.
- f een afbeelding van L_B naar L_C volgens:
  stel L_B en L_C zijn lexicografisch geordend, dan wordt het i^e element van L_B op het i^e element van L_C afgebeeld.

De kodes zijn van het variabele naar vaste lengte type.
K is een maat voor de begrensde komplexiteit van de kode.

Gegeven K is de enig overgebleven vrijheid de vorm van B_K.

Voor geheugenloze bronnen S is de maximale (S,K,n) boom natuurlijk de beste keuze voor B_K. Echter S is onbekend, dus B_K kan niet à priori de juiste vorm gegeven worden.

We zullen het probleem dan ook op een andere manier benaderen.
Met in het achterhoofd de beschrijving van het Ziv-Lempel algoritme, zoals gegeven in hoofdstuk 4.3.2, kennen we aan B_K gewichten toe, die aangepast worden, afhankelijk van de vorm van het afgepaste segment. Dit gebeurt weer door de gewichten van alle bezochte knopen één op te hogen.

We initialiseren B_K zo dat voor alle inwendige knopen y geldt (met w(z) het gewicht van knoop z)

$$w(y) = \sum_{x \in A} w(yx) \qquad (4.49)$$

A is het bronalfabet.

De bovengenoemde aanpassing van de gewichten verstoort de relatie (4.49) niet, wat eenvoudig is in te zien.
We gaan m.b.v. B_K een model voor de bron opstellen, net zo als in hoofdstuk 4.3.2.

Aan alle knopen z kennen we een getal q(z) toe volgens

$$q(z) \triangleq \frac{w(z)}{w(\lambda)} \qquad (4.50)$$

Uit (4.49) volgt direkt dat q(z) konsistent gedefiniëerd is.
Als de bron geheugenloos is weten we dat q(z) nadert tot p(z), de bron-kans op rij z.

De kansen q(.), of, equivalent, de gewichten w(.), vormen het mechanisme voor het aanpassen van de segmentboom aan de bronstatistieken.

In plaats van de maximale uitbreiding maken we gebruik van het begrip de maximale uitwisseling, gegeven het model bepaald door q(.). Hierbij gebruiken we het begrip bladvader.

Een bladvader is een inwendige knoop van een komplete n-boom met enkel bladeren als opvolgers.

De maximale uitwisseling van een komplete n-boom B is de komplete n-boom B^1 die uit B ontstaat door bij een bladvader y van B de n opvolgers weg te halen, en aan een blad z van B weer n bladeren te plaatsen. Van y eisen we dat w(y) minimaal is over alle bladvaders en van z eisen we dat w(z) maximaal is over alle bladeren van B. De gewichten die aan de nieuwe bladeren toegekend worden voldoen aan:

$$w(z) = \sum_{x \in A} w(zx) \text{ met z het bepaalde oude blad} \qquad (4.51)$$

en

$$\left| w(zx) - w(zx') \right| \leq 1 \text{ voor alle } x,x' \in A \qquad (4.52)$$

Dus het gewicht van z wordt zo gelijk mogelijk over de nieuwe bladeren verdeeld.

Waarom moet (4.52) gelden?

Daar in het model behorende bij B niets bekend is over de kansen op de nieuwe bladeren is er geen reden om het ene blad zwaarder te wegen dan het andere. We gaan er namelijk niet van uit dat de bron geheugenloos is, zodat de andere gewichten w(y) in de boom geen indikatie behoeven te zijn voor de gewichten van de nieuwe bladeren.

Binnen dit model voor B geldt het lemma 4.2 nog steeds, zodat de eis

$$q(z) \leq q(y) \text{ voor alle bladeren z en inwendige knopen y} \qquad (4.53)$$

een test is voor het maximaal zijn van B.

Stelling 4.5 : Het nu volgende algoritme genereert voor elke K ten alle tijden maximale segmentbomen in de zin van de test (4.53).

Algoritme : { basis algoritme }

Stap 1. { initialisatie }
B_K^0 is de gebalanceerde n-boom met K bladeren.
w(z) := 1, voor alle bladeren z van B_K^0.
w(y) wordt konsistent, zie (4.49), voortgezet, zodat o.a. $w(\lambda) = K$.
i := 0. { i is de index. i geeft aan hoeveel segmenten al verwerkt zijn. }

Stap 2. { het vinden en koderen van het nieuwe segment. }
Pas het $(i+1)^e$ segment af m.b.v. B_K^i. Kodeer dit met C en f. { C en f werken samen met alle B_k^i's, en geven als kodewoord de lexicografische index van een blad z van de segmentboom in de lijst van alle bladeren. }
Hoog de gewichten w(y) bij alle bezochte knopen y één op.

Stap 3. { aanpassing voor volgende segment }
Stel : y is de bladvader van B_K^i met minimaal gewicht.
z is het blad van B_K^i met maximaal gewicht.
Als w(y) < w(z)
dan : B_K^{i+1} is de maximale uitwisseling van B_K^i. { de wisselpunten zijn y en z }
anders : B_K^{i+1} is gelijk aan B_K^i.
i := i + 1.

Stap 4. Ga naar Stap 2.

Het bewijs van stelling 4.5.

Stap 1. Voor de bladeren z van B_K^0 geldt w(z) = 1, en voor de inwendige knopen y geldt w(y) > 1. Dus, volgens de test, is B_K^0 maximaal.

Stap 2. Is een transitie stap.

Stap 3. Hier wordt de nieuwe boom gevormd. Te bewijzen is dat deze boom ook maximaal is.

Stel w(y) ≥ w(z), dan wordt er niet gewisseld. Daar y minimaal is over alle bladvaders en dus over alle inwendige knopen, geldt

q(u) ≥ q(v) voor alle bladeren v en inwendige knopen u.

Dus is in dit geval de nieuwe boom ook maximaal.

Stel w(y) < w(z), dan wordt er gewisseld. Dit deel van het bewijs wordt in verschillende stukjes behandeld.

a) De identificatie van het blad z.
z is het blad met maximaal gewicht.
B_K^i was, voor het ophogen, maximaal, dus zijn alle bladeren van de boom niet zwaarder dan de inwendige knopen.
Noem het blad, waarvan het gewicht één is opgehoogd, b.b is de enig mogelijke kandidaat voor de wisseling, dus b ≡ z.

b) y wordt een blad van de nieuwe boom. Voor y moet gelden:

$w(y) \leq w(u)$, voor alle inwendige knopen u van B_K^{i+1}.

z is een bladvader van B_K^{i+1}. Daar w(y) minimaal is over alle bladvaders van B_K^i en $w(y) < w(z)$, geldt het bovenstaande.

c) De opvolgers zx van z zijn bladeren van de nieuwe boom. Er moet gelden:

$w(zx) \leq w(u)$ voor alle inwendige knopen u van de nieuwe boom.

Het gewicht w(zx) is na de uitwisseling niet groter dan $\text{entier}\left(\frac{w(z)}{n}\right)+1$ en met $w(z) \geq 1$ en $n \geq 2$ volgt $w(zx) \leq w(z)$, waarbij zx genomen is uit de nieuwe boom en w(z) de oude, d.i. voor het ophogen, waarde is. En daar het oude gewicht van z niet groter was dan het gewicht van de bladvaders van B_K^i geldt $w(zx) \leq w(u)$ voor alle inwendige knopen u van de nieuwe boom.

d) $w(u) \leq w(v)$ voor alle bladeren u, met $u \neq y$ en $u \neq zx$, van B_K^{i+1}, en alle in-wendige knopen $v \neq z$.

Voor alle inwendige knopen $v \neq z$ geldt dat hun gewichten niet afgenomen zijn. En voor alle bladeren $u \neq y$ of zx

geldt dat w(u) onveranderd is, dus geldt w(u) $\leq$ w(v).
Ook geldt w(u) $\leq$ w(z) daar z een maximaal gewicht had.
Tevens geldt w(v) $\geq$ w(y) daar y een minimaal gewicht bezat.
En met w(zx)$\leq$w(z)$\leq$w(t), met t een willekeurige inwendige knoop van B_K^i geldt w(zx)$\leq$w(v).

Dus is B_K^{i+1} ook maximaal.

∴

4.4.2 De verschillende vormen van het algoritme.

Het bovengenoemde basis algoritme is een adaptief algoritme.
We zullen in dit hoofdstuk drie al dan niet adaptieve aanpassingen van dit algoritme geven.

Laten we aannemen dat de te verzenden rij een bij de zender en de ontvanger bekende eindige lengte van L symbolen heeft.

Elk algoritme wordt gegeven als een kodeer algoritme en een bijpassend dekodeer algoritme.

<u>Kodeer algoritme 1</u>.

Stap 1. Initialiseer B_K^0 volgens Stap 1 van het basis algoritme.

Stap 2. Pas segment i+1 af volgens B_K^i, en voeg als het nodig is nullen toe. { Zie hieronder voor uitleg. }
Zend het bijbehorende kodewoord uit.
Hoog de gewichten van de bezochte knopen één op.

Stap 3. Identiek aan Stap 3 van het basis algoritme.

Stap 4. Als de rij niet volledig verwerkt is dan ga naar Stap 2 anders STOP.

Dekodeer algoritme 1.

Stap 1. Initialisatie volgens Stap 1 van het basis algoritme.

Stap 2. Ontvang het $(i+1)^e$ kodewoord. { Dit kan daar de lengte van elk kodewoord konstant is, n.l. $^n\log K$. }
Bepaal m.b.v. B_K^i, C en f het bijbehorende segment. Haal er de eventueel toegevoegde nullen af. { Zie opmerking beneden }
Hoog de gewichten van de bezochte knopen op.

Stap 3. Identiek aan Stap 3 van het basis algoritme.

Stap 4. Als de rij niet volledig verwerkt is dan ga naar Stap 2 anders STOP.

Opmerking:

Daar de datarij een eindige lengte L heeft kan het laatste deel van de rij te klein zijn. d.w.z. te kort om een segment uit de segment boom te zijn. De zender voegt net zo veel nullen toe tot wel een volledig segment bereikt is. Daar de ontvanger ook de lengte L kent kan deze de toegevoegde nullen er weer vanaf halen.

Het tweede algoritme is een stationair, twee slagen, algoritme.

Kodeer algoritme 2.

Stap 1. Initialisatie. { Zie Stap 1 van het basis algoritme }

Stap 2. Pas segment i+1 af, evt. nullen toevoegen.
Pas de gewichten van de bezochte knopen aan.

Stap 3. Identiek aan Stap 3 van het basis algoritme.

Stap 4. Als de rij niet geheel verwerkt is
dan ga naar Stap 2
anders ga naar Stap 5.

Stap 5. Herinitialiseer de datarij voor de tweede slag.

Stap 6. Zend een beschrijving van de laatst gevormde boom B_K^m over naar de ontvanger.

Stap 7. Pas het $(i+1)^e$ segment af m.b.v. B_K^m, evt. nullen toevoegen.
Zend het bijbehorende kodewoord uit.

Stap 8. Als de rij niet verwerkt is,
dan i := i + 1, ga naar Stap 7,
anders STOP.

Dekodeer algoritme 2.

Stap 1. Ontvang de beschrijving van B_K^m en vorm deze boom.
i := 0.

Stap 2. Ontvang het $(i+1)^e$ kodewoord en bepaal m.b.v. B_K^m, C_K en f het bijbehorende segment. Haal de eventueel toegevoegde nullen eraf.

Stap 3. Als de rij niet geheel ontvangen is,
dan i := i + 1, ga naar Stap 2,
anders STOP.

Het derde en laatste algoritme is een twee slags adaptief algoritme. De bedoeling van de eerste slag is om een betere start boom te krijgen dan in het eerste algoritme, terwijl een toevallig slechte boom B_K^m niet zo'n invloed heeft als in het tweede algoritme.

Kodeer algoritme 3.

Stap 1. Initialiseer B_K^0. i := 0.

Stap 2. Pas segment i+1 af m.b.v. B_K^i, en voeg evt. nullen toe.
Pas de gewichten van de bezochte knopen aan.

Stap 3. Vorm B_K^{i+1} uit B_K^i. i := i + 1.

Stap 4. Als de rij niet verwerkt is,
dan ga naar Stap 2,
anders ga naar Stap 5.

Stap 5. Herinitialiseer de datarij.

Stap 6. Zend een beschrijving van de laatste boom B_K^m over.
i := m.

Stap 7. Pas segment i-m+1 af m.b.v B^i, voeg evt. nullen toe.
Zend het bijbehorende kodewoord uit.
Pas de gewichten aan van de bezochte knopen.

Stap 8. Vorm B_K^{i+1} uit B_K^i. i := i + 1.

Stap 9. Als de rij niet verwerkt is,
dan ga naar Stap 7,
anders STOP.

Dekodeer algoritme 3.

Stap 1. Ontvang de beschrijving van B_K^m en vorm deze. i := m.

Stap 2. Dekodeer kodewoord i-m+1 m.b.v. B_K^i, C_K en f. Haal de evt. toegevoegde nullen eraf.

Stap 3. Vorm B_K^{i+1} uit B_K^i. i := i + 1.

Stap 4. Als de rij niet geheel ontvangen is,
dan ga naar Stap 2,
anders STOP.

In tegenstelling tot de laatste twee heeft het eerste algoritme geen model-overhead kosten, daarintegen moet in het eerste algoritme de boom eerst de juiste vorm nog vinden en dit kost in het begin aan kompressie.

Veronderstellen we een konvergerend gedrag van het basisalgoritme, dan zal voor algoritme 1 de vorm van de boom op den duur stabiel worden en zal het verlies aan reduktie, t.o.v. de stabiele vorm, eindig zijn en dus in de limiet verwaarloosbaar. Zie bijlage C.

Uit de kombinatoriek is bekend dat het aantal komplete binaire bomen met K bladeren gegeven wordt door c_{K-2}, met c_n het n^e Catalan getal.

De Catalan getallen zijn gegeven als:

$$c_n \triangleq (2n+2)\ \binom{2n+1}{n}.$$

Met behulp van Stirlings bovengrens:

$$n! < \sqrt{(2\pi n)}\left(\frac{n}{e}\right)^n \exp\left(\frac{1}{12n}\right).$$

is dan te bewijzen dat een komplete binaire boom met K bladeren minder dan 2K bit informatie bevat.

De bijdrage van de modelkosten in de reduktiefaktor bedraagt dus niet meer dan $\frac{2K}{L}$. Voor grote L, d.w.z. groot t.o.v. K, is deze bijdrage dus verwaarloosbaar klein.

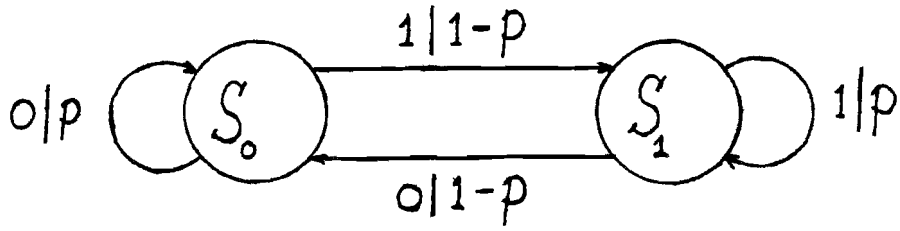
Het derde algoritme gedraagt zich als de kombinatie van één en twee.
We mogen dus, gegeven de veronderstelling dat het basisalgoritme konvergeert, aannemen dat alle drie de algoritmen o.d.d. hetzelfde gedrag vertonen.

Het probleem blijft of het basisalgoritme konvergeert voor een bepaalde klasse van bronnen, en als het konvergeert, konvergeert het dan naar de optimale keuze.
Ook voor geheugenloze bronnen is het mij niet gelukt dit te bewijzen. Het probleem ligt in het feit dat ik niet kan bewijzen dat, ondanks de optredende uitwissellingen van knopen, de modelkansen q(.) konvergeren naar de bronkansen p(.).

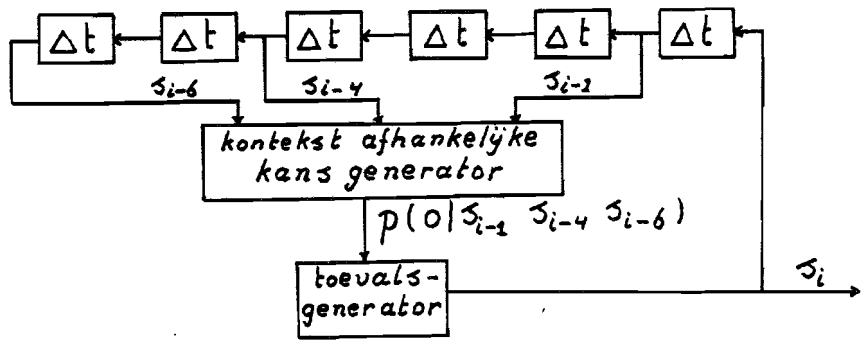
Ik heb enkele simulaties met de drie algoritmen uitgevoerd. De resultaten hiervan, zie het volgende hoofdstuk, geven wel een indikatie dat de algoritmen redelijk werken, doch beslist geen eenduidig antwoord.

4.4.3 Enkele resultaten.

Voor de simulaties heb ik drie verschillende binaire brontypen gebruikt.
Het eerste type is de geheugenloze bron, het tweede is de differentiële bron, zie figuur 4.10, en het laatste is een zesde orde Markov bron, beschreven als een 'finitely generated source', zie Rissanen [1983] en hoofdstuk 5. Zie ook figuur 4.11.



Figuur 4.10.



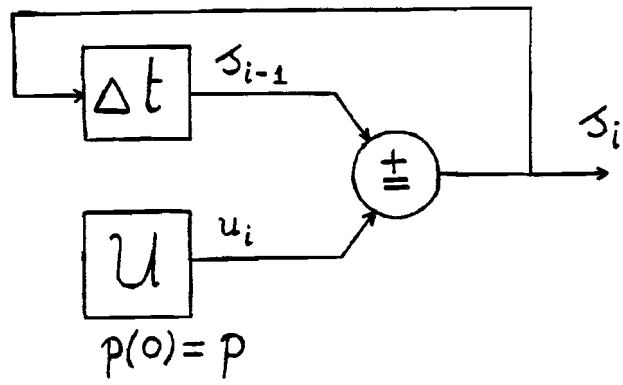
Figuur 4.11.

Dat ik het tweede type een differentiële bron noem wordt duidelijk uit het volgende.
De bron S_a van figuur 4.10 heeft veel weg van een geheugenloze bron. Zie figuur 4.12. Hierin is U een geheugenloze bron met $p(0) = p$, dezelfde als de kans $p(0|0) = p(1|1) = p$ in figuur 4.10.

In het geheugenelement, Δt, wordt de vorige uitkomst s_{i-1} van de totale bron S_b bewaard. De uitkomst s_i van S_b ontstaat volgens:

$$s_i \stackrel{\Delta}{=} u_i \stackrel{+}{=} s_{i-1}, \quad (\stackrel{+}{=} \text{ is de modulo 2 optelling.})$$

Bron S_b begint op 'tijdstip' $i = 1$. De inhoud van Δt op $i=1$, d.i. s_0, moet nog gedefiniëerd worden. Kies voor s_0 een binaire stochastische variabele met kans $p(s_0=0)=\frac{1}{2}$.



Figuur 4.12.

Nu is te bewijzen dat S_a, figuur 4.10, en S_b, figuur 4.12, dezelfde zijn.

Voor S_a geldt:

$$p_a(s_i=0) = \tfrac{1}{2} \qquad i \geq 1.$$

$$p_a(s_{i+1}=0 \,|\, s_i=0) = p \qquad i \geq 1.$$

$$p_a(s_{i+1}=1 \,|\, s_i=1) = p \qquad i \geq 1.$$

$$p_a(s_t \,|\, s_i^{t-1}) = P_a(s_t \,|\, s_{t-1}) \qquad t \geq 2, \; t > i \geq 1$$

Voor S_b geldt:

$$p_b(s_i=0) = p_b(s_{i-1}=0)p(u_i=0) + p_b(s_{i-1}=1)p(u_i=1).$$

Stel voor $i < t$ geldt $p_b(s_i) = \frac{1}{2}$,
dan $p_b(s_t=0) = \frac{1}{2}p + \frac{1}{2}(1-p) = \frac{1}{2}$.
En daar $p_b(s_0=0) = \frac{1}{2}$, per definitie, volgt:

$$p_b(s_i=0) = \tfrac{1}{2} \qquad i \geq 1.$$

$$p_b(s_{i+1}=0 \,|\, s_i=0) = p(u_{i+1}=0) = p \quad i \geq 1.$$

$$p_b(s_{i+1}=1 \,|\, s_i=1) = p(u_{i+1}=0) = p \quad i \geq 1.$$

$$p_b(s_t \,|\, s_i^{t-1}) = p_b(u_t \oplus s_{t-1} \,|\, s_i^{t-1}) = p_b(s_t \,|\, s_{t-1}) \quad t \geq 2, t > i \geq 1.$$

$\therefore$

We konkluderen dat S_a en S_b dezelfde statistische eigenschappen hebben. Indien de ontvanger s_0 kent, dan kan deze de bron S_b dekoderen als de bron U.

Van elk type zijn zes bronrijen van 10^6 bit elk aangemaakt. Hiervoor is gebruik gemaakt van een brongenererend programma, gebaseerd op het 'finitely generated source' model, en de Elias toevalsgenerator, zie Kleinen [1979].
De resultaten staan vermeld in figuur 4.13. In tabel 4.13c staan voor het derde type de konditionele kansen per kontekst vermeld.
De genererende indices van dit type zijn:

$$i_1=1, \; i_2=4, \; i_3=6.$$

Geheugenloze bron.

datarij naam	p(0)	entropie	
		berekend	gemeten
SOURCE90	0,300	0,88129	0,882
SOURCE91	0,200	0,72193	0,723
SOURCE92	0,100	0,46900	0,468
SOURCE93	0,050	0,28640	0,228
SOURCE94	0,010	0,08079	0,0810
SOURCE95	0,001	0,01141	0,0112

Figuur 4.13a.

Differentiële bron.

datarij naam	p	entropie	
		berekend	gemeten [1]
SOURCE30	0,300	0,88129	0,881
SOURCE31	0,200	0,72193	0,723
SOURCE32	0,100	0,46900	0,468
SOURCE33	0,050	0,28640	0,288
SOURCE34	0,010	0,08079	0,0794
SOURCE35	0,001	0,01141	0,0116

Figuur 4.13b.

voetnoot [1] : Dit is de gemeten $H_{\infty}(S)$.

Zesde orde Markov bron.

datarij naam	$p(0 \mid z_1z_4z_6)$, met $z_1z_4z_6$ [3] is								entropie	
	000	001	010	011	100	101	110	111	berekend [1]	gemeten [2]
SOURCE60	0,7961	0,35	0,35	0,4	0,6	0,65	0,65	0,2039	0,88130	0,881
SOURCE61	0,8015	0,35	0,25	0,1	0,9	0,75	0,65	0,1985	0,72191	0,722
SOURCE62	0,92285	0,15	0,08	0,9	0,1	0,92	0,85	0,07715	0,46902	0,468
SOURCE63	0,96637	0,05	0,1	0,1	0,9	0,9	0,95	0,03363	0,28642	0,285
SOURCE64	0,98	0,99454	0,01	0,01	0,99	0,99	0,00546	0,02	0,08078	0,0810
SOURCE65	0,999034	0,01	0,995	0,01	0,99	0,005	0,99	0,000966	0,01141	0,0104

Figuur 4.13c.

voetnoot [1] : De theoretische waarde is m.b.v. een computer programma berekend volgens het 64 toestanden Markov model.

[2] : Dit is de gemeten $H_\infty(S)$.

[3] : $z_1z_4z_6$ is resp. het eerste, het vierde en het zesde symbool voorafgaande aan het huidige.

Van alle achttien bronrijen zijn schattingen van de innovatie entropie gemaakt. Het hiervoor gebruikte algoritme werkt alsvolgt:

$\{$ schat $H(S_L|S_1^{L-1})$ $\}$

Stap 1. tel het aantal voorkomens van alle subrijen z_1^L; noem dit $c(z_1^L)$.

Stap 2. bepaal de schattingen van de konditionele kansen $q(z_L|z_1^{L-1})$ als:

$$q(z_L|z_1^{L-1}) = \frac{c(z_1^L)}{\sum_{z \in A} c(z_1^{L-1},z)} .$$

Stap 3. bepaal schattingen van de kansen $q(z_1^{L-1})$ als:

$$q(z_1^{L-1}) = \sum_{z \in A} c(z_1^{L-1},z) / \sum_{x_1^L \in A^L} c(x_1^L).$$

Stap 4. de entropie schatting wordt gegeven door:

$$H(S_L|z_1^{L-1}) = - \sum q(s_L|z_1^{L-1}) \log q(s_L|z^{L-1})$$

$$H(S_L|S_1^{L-1}) = \sum_{z_1^{L-1} \in A^{L-1}} q(z_1^{L-1}) H(S_L|z_1^{L-1}).$$

De resultaten hiervan zijn grafisch weergegeven in figuur 4.14. We zien dat de zesde orde Markov bronnen een nogal grillig verloop hebben.


1,00
0,96
0,92
0,88
0,84
0,80
0,76
0,72
0,68
0,64
0,60
0,56
0,52
0,48
0,44
0,40
0,36
0,32
0,28
0,24
0,20
0,16
0,12
0,08
0,04
0,00
SOURCE 90
SOURCE 91
SOURCE 92
SOURCE 93
SOURCE 94
SOURCE 95
SOURCE 30
SOURCE 31
SOURCE 32
SOURCE 33
SOURCE 34
SOURCE 35
SOURCE 60
SOURCE 61
SOURCE 62
SOURCE 63
SOURCE 64
SOURCE 65
0 1 2 3 4 5 t
0 1 2 3 4 5 t
0 1 2 3 4 5 6 7 8 9 10 t

Figuur 4.14.

De drie algoritmen zijn uitgeprobeerd op alle achttien bronrijen, met achtereenvolgens 16, 256, 1024 en 8192 bladeren. Ook het aangepaste Ziv-Lempel algoritme, zie hoofdstuk 4.3, is op deze rijen losgelaten. In figuur 4.15 zijn de resultaten opgenomen.
In de grafieken 4.16a t/m f zijn verschillende uitkomsten verduidelijkt. De figuren 4.16a t/m c tonen de werking van het eerste algoritme op respektievelijk de geheugenloze bron, de differentiële bron en de Markov bron. De parameter is de boomgrootte.
De figuren 4.16d t/m f tonen de werking van het derde algoritme, met 1024 bladeren, op de drie bronnen met als parameter de lengte van de verwerkte datarij. (d.i. het getal achter de dubbele punt rechts in de grafiek) Deze laatste figuren geven een indruk van de konvergentie snelheid van het algoritme.
Het derde algoritme is gekozen om de invloed van de modelkosten bij kleine rijlengten goed uit te laten komen.

We zien i.h.a. het volgende gedrag:

De drie algoritmen ontlopen elkaar niet veel en er is geen uitgesproken favoriet.

Voor de geheugenloze bronnen wordt de entropie tamelijk dicht benaderd.

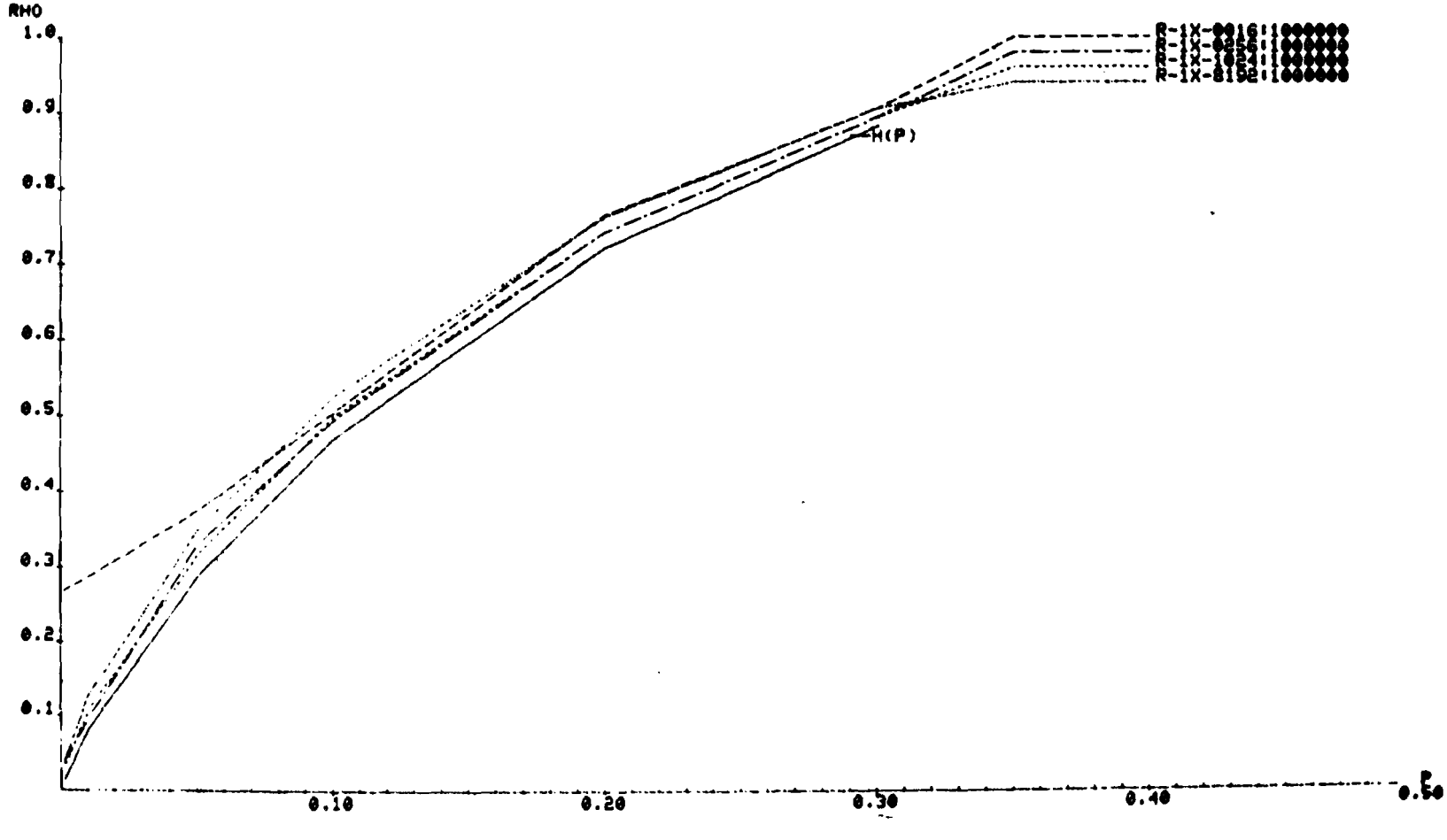
Ook voor bronnen met geheugen vindt kompressie plaats, al wordt hier de entropie minder dicht benaderd.

t.o.v. het Ziv-Lempel algoritme zijn bij een geschikte keuze voor de boomgrootte de nieuwe algoritmen beter en eisen ze minder geheugenruimte.

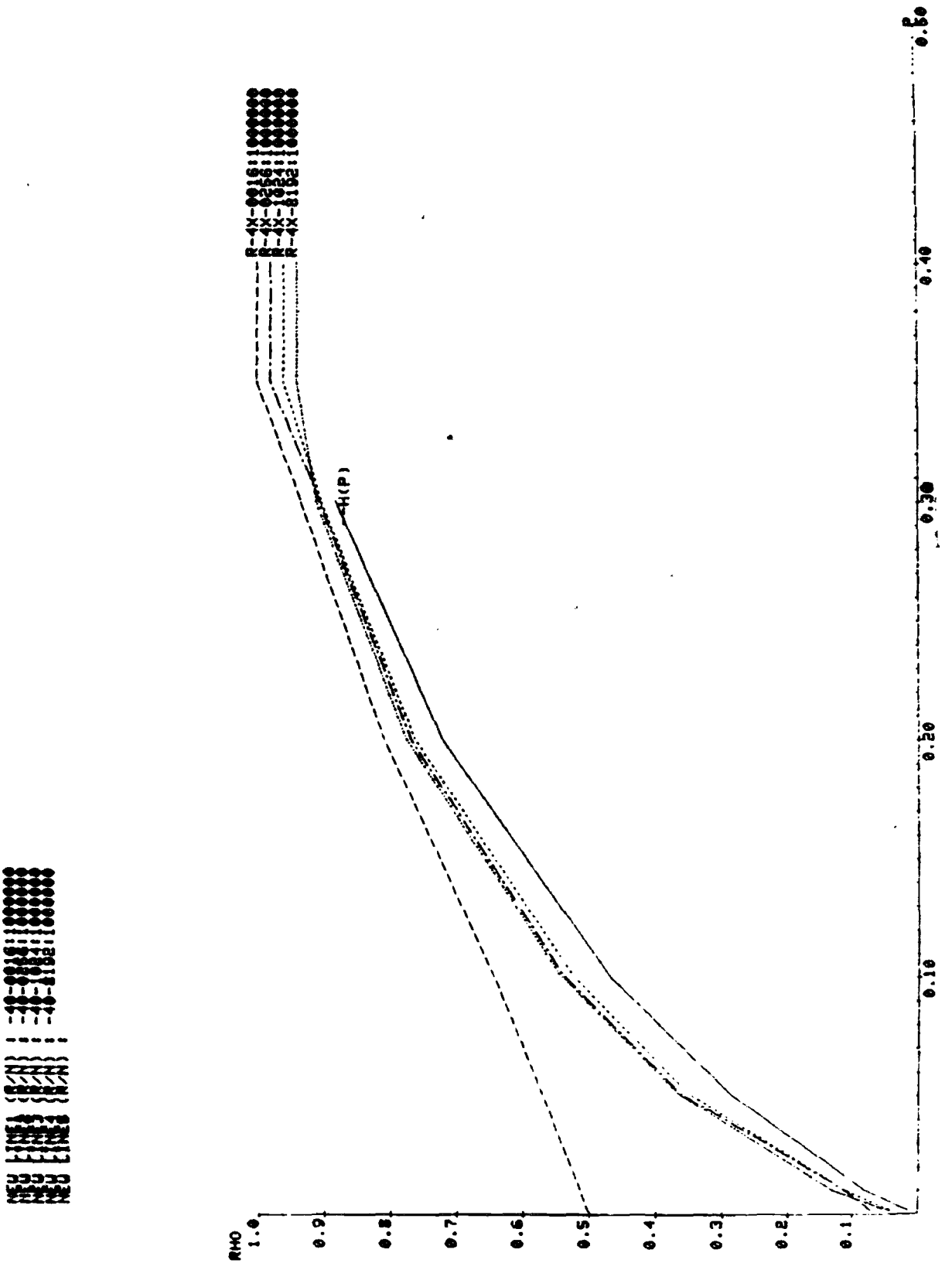
Voor grotere boomgrootte wordt de kompressie weer slechter, daar de boom dan geen tijd heeft gehad de juiste vorm aan te nemen.

datarij naam	kompressie na 1.000.000 bronsymbolen.													
	Algoritme 1				Algoritme 2				Algoritme 3				Ziv-Lempel	
	#bladeren				#bladeren				#bladeren					
	16	256	1024	8192	16	256	1024	8192	16	256	1024	8192	ρ	#blad
SOURCE90	0,9044	0,8931	0,8930	0,9050	0,9042	0,8927	0,8922	0,9079	0,9045	0,8928	0,8924	0,9072	0,9347	62512
SOURCE91	0,7650	0,7422	0,7423	0,7631	0,7649	0,7417	0,7399	0,7572	0,7649	0,7417	0,7397	0,7551	0,7802	52860
SOURCE92	0,5029	0,4929	0,4976	0,5242	0,5029	0,4915	0,4933	0,5111	0,5029	0,4915	0,4934	0,5078	0,5204	36619
SOURCE93	0,3733	0,3285	0,3152	0,3477	0,3733	0,3268	0,3090	0,3373	0,3733	0,3266	0,3085	0,3328	0,3348	24504
SOURCE94	0,2859	0,0942	0,1064	0,1271	0,2859	0,0900	0,1028	0,1285	0,2859	0,0894	0,1021	0,1249	0,1100	9023
SOURCE95	0,2686	0,0378	0,0278	0,0348	0,2686	0,0359	0,0209	0,0386	0,2686	0,0359	0,0197	0,0366	0,0213	2117
SOURCE30	0,9335	0,9055	0,9032	0,9114	0,9335	0,9053	0,9024	0,9340	0,9335	0,9054	0,9053	0,9142	0,9420	62969
SOURCE31	0,8085	0,7689	0,7628	0,7760	0,8085	0,7685	0,7608	0,7722	0,8085	0,7683	0,7609	0,7703	0,7958	53833
SOURCE32	0,6431	0,5372	0,5231	0,5423	0,6431	0,5363	0,5192	0,5311	0,6431	0,5362	0,5189	0,5281	0,5446	38131
SOURCE33	0,5694	0,3594	0,3479	0,3629	0,5695	0,3576	0,3436	0,3531	0,5695	0,3576	0,3434	0,3493	0,3555	25880
SOURCE34	0,5129	0,1110	0,1115	0,1309	0,5129	0,1087	0,1063	0,1332	0,5129	0,1088	0,1054	0,1302	0,1179	9588
SOURCE35	0,5014	0,0683	0,0354	0,0399	0,5014	0,0671	0,0305	0,0446	0,5014	0,0670	0,0276	0,0390	0,0287	2729
SOURCE60	0,9800	0,9472	0,9343	0,9329	0,9800	0,9465	0,9342	0,9382	0,9800	0,9467	0,9382	0,9382	0,9649	64403
SOURCE61	0,9557	0,8511	0,8230	0,8190	0,9736	0,8506	0,8208	0,8150	0,9556	0,8506	0,8137	0,8137	0,8378	56456
SOURCE62	1,0000	0,9072	0,7283	0,6875	1,0000	0,9089	0,7239	0,6741	1,0000	0,9067	0,6712	0,6712	0,6885	47127
SOURCE63	0,8287	0,4187	0,4083	0,4025	0,8334	0,4165	0,4043	0,3903	0,8288	0,4161	0,3873	0,3873	0,3966	28619
SOURCE64	0,9858	0,4458	0,2025	0,1795	0,9784	0,4598	0,1926	0,1642	0,9850	0,4498	0,1601	0,1601	0,1632	12826
SOURCE65	0,5046	0,0701	0,0348	0,0392	0,5012	0,0688	0,0277	0,0404	0,5012	0,0688	0,0378	0,0378	0,0275	2635

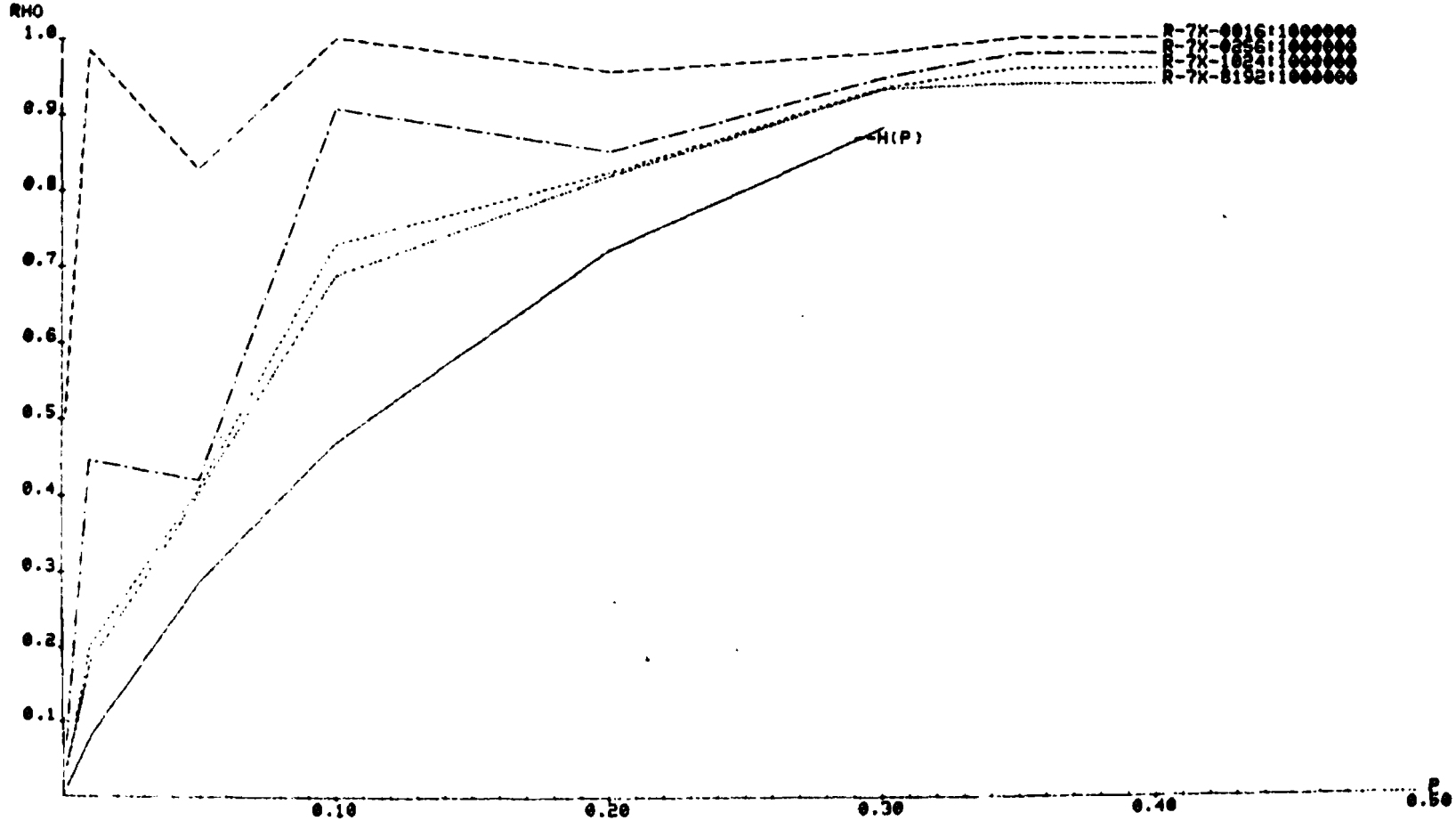
Figuur 4.15



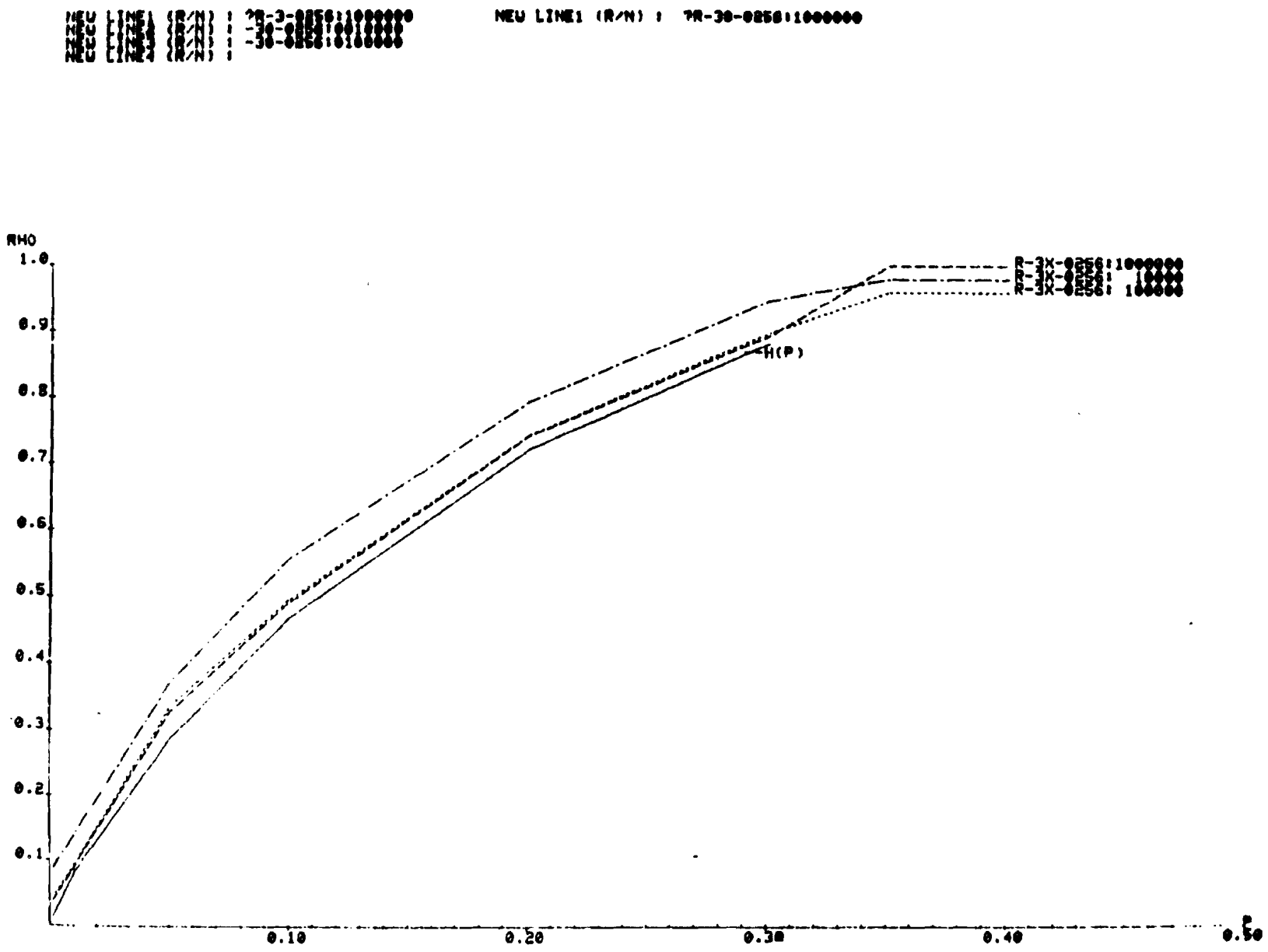
Figuur 4.16a.



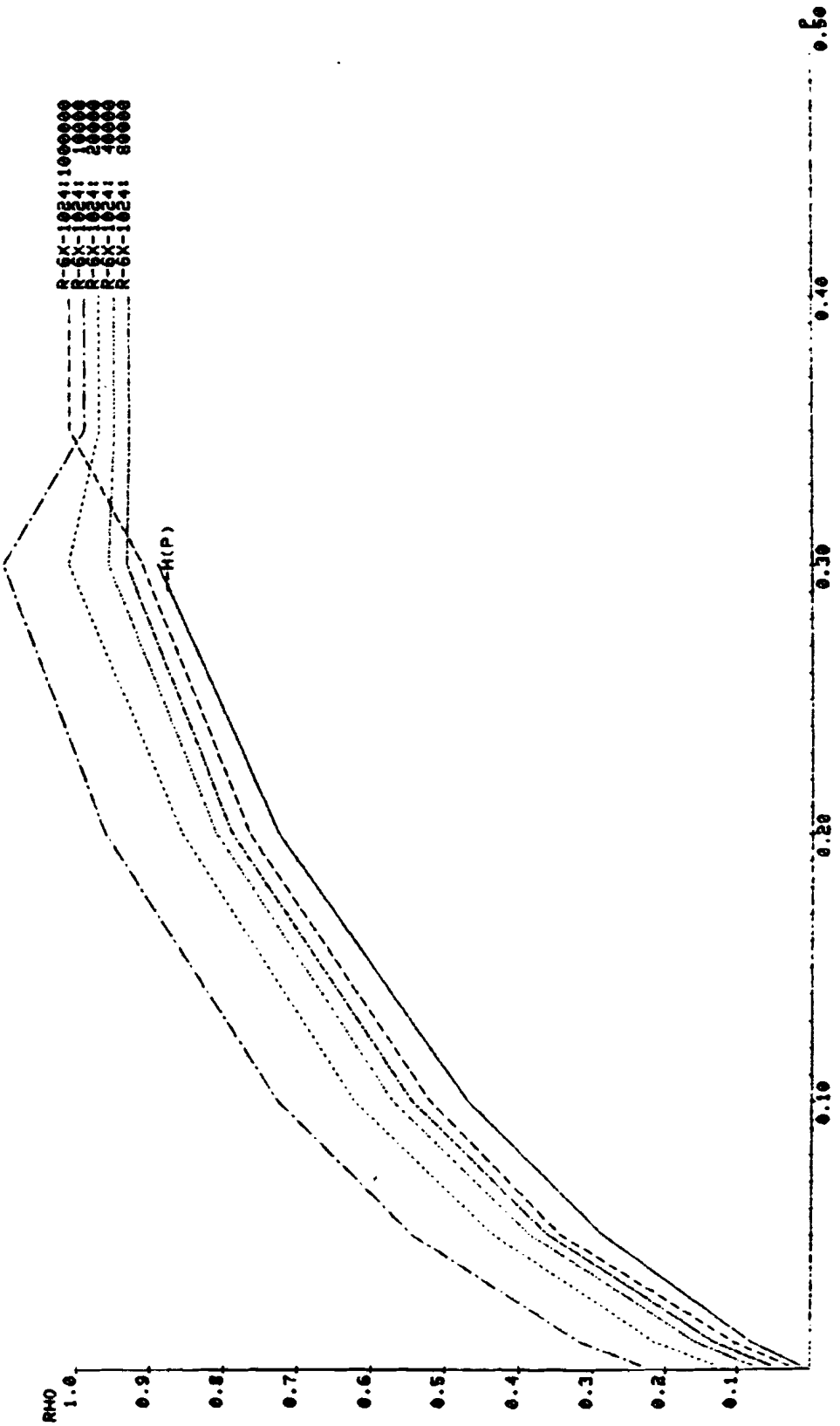
Figuur 4.16b.



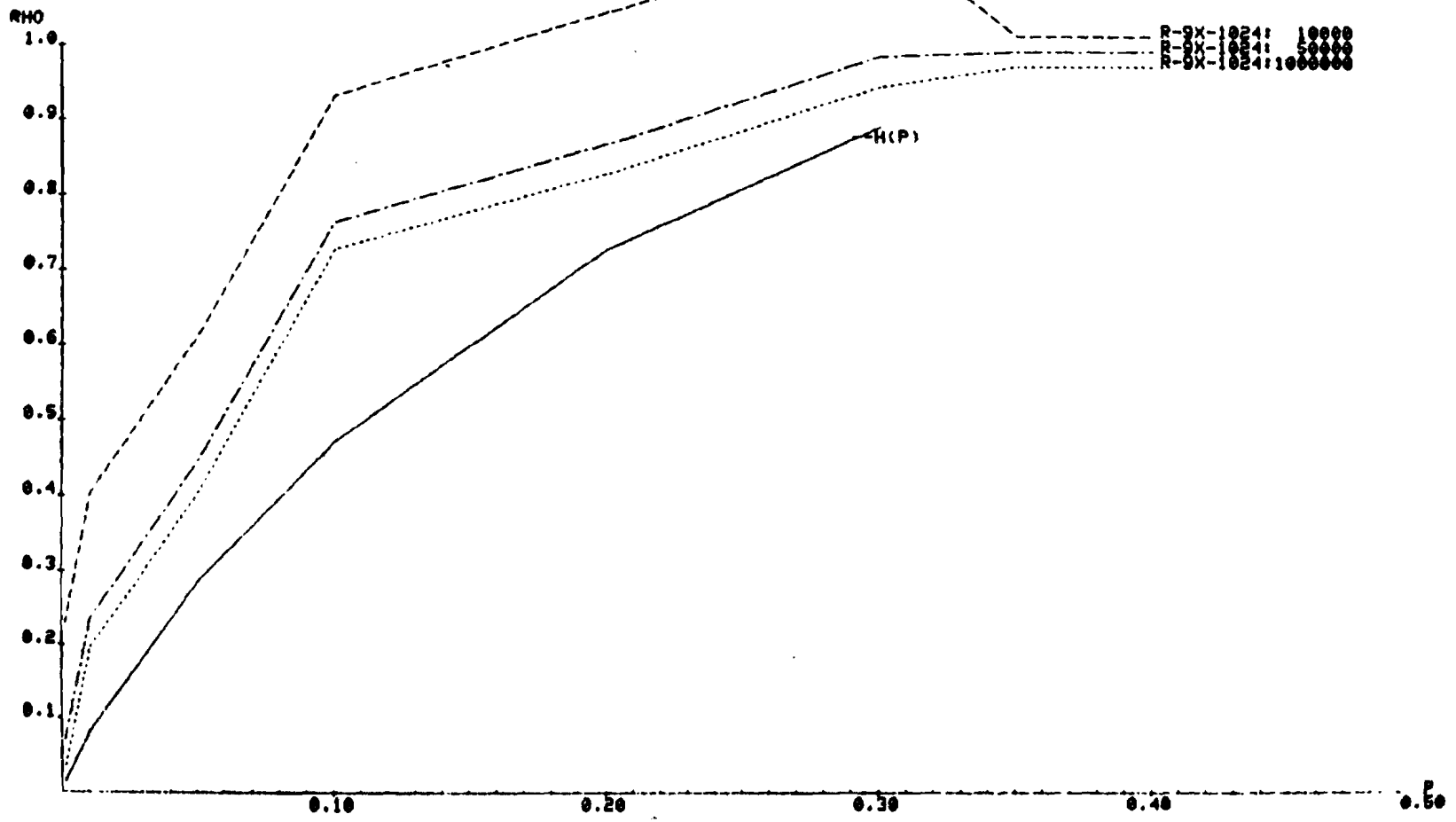
Figuur 4.16c.


NEW LINE1 (R/N) : 2R-3-0256:1000000
NEW LINE1 (R/N) : 2R-30-0256:1000000
RHO
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
H(P)
0.10
0.20
0.30
0.40
0.50

Figuur 4.16d.



Figuur 4.16e.



Figuur 4.16f.

Van de geheugenloze bronnen, SOURCE90 t/m SOURCE95, zijn ook de Tunstall/Verhoeff bomen bepaald voor de vier boomgrootten. In figuur 4.17 zijn de resultaten voor het eerste en het derde algoritme getabelleerd.
Ons valt op dat voor kleine boomgrootten de Tunstall boom vorm snel gevonden wordt en dat voor grotere bomen deze redelijk wordt benaderd. Het derde algoritme is in dit opzicht duidelijk beter dan het eerste.
In figuur 4.18a wordt een uitgewerkt voorbeeld met tussenresultaten gegeven van het eerste algoritme bij 256 bladeren op de geheugenloze bron met p(0) = 0,3. In dit geval wordt de Tunstall vorm bereikt.
In de eerste kolom van deze tabel staat de lengte van de verwerkte datarij vermeld; in de tweede de dan behaalde 'over all' kompressie; in de derde het aantal verwerkte segmenten; in de vierde het aantal aanpassingen van de boomvorm bij de laatste ±10000 bronsymbolen; in de vijfde de gemiddelde segmentlengte van de huidige boomvorm volgens het werkelijke bron model en in de zesde kolom de procentuele afwijking van de vijfde kolom t.o.v. de optimale waarde.
Het tweede voorbeeld in figuur 4.18b laat zien dat niet altijd de optimale vorm bereikt wordt. In deze tabel staat de werking van het derde algoritme met 1024 bladeren op dezelfde bron vermeld.
We mogen konkluderen dat deze algoritmen inderdaad in staat zijn bronnen met geheugen te komprimeren. Er is echter geen uitsluitsel over de kwaliteit, d.w.z de redundantie, konvergentiesnelheid of komplexiteit, van de algoritmen.
Het oscillerend gedrag van sommige simulaties kan waarschijnlijk voor een deel verklaard worden uit het feit dat de knoopkansen afhankelijk zijn van de boomvorm, zie hoofdstuk 4.2.

bronrij naam	Tunstall/Verhoeff segmentlengtes $\bar{\ell}_s$ en de afwijking van het 1e en 3e algoritme t.o.v. $\bar{\ell}_s$ in procenten.											
	#bladeren = 16			#bladeren = 256			#bladeren = 1024			#bladeren = 8192		
	$\bar{\ell}_s$	Δ (%)		$\bar{\ell}_s$	Δ (%)		$\bar{\ell}_s$	Δ (%)		$\bar{\ell}_s$	Δ (%)	
		prg#1	prg#3		prog#1	prog#3		prog#1	prog#3		prog#1	prog#3
SOURCE90	4,426	0,00	0,00	8,970	0,00	0,00	11,239	-0,0405	-0,0212	14,645	-0,511	-0,261
SOURCE91	5,239	0,00	0,00	10,812	-0,0111	-0,0075	13,583	-0,0806	-0,0493	17,738	-1,01	-0,466
SOURCE92	7,941	0,00	0,00	16,269	-0,0153	-0,0083	20,355	-0,223	-0,121	26,841	-2,42	-1,17
SOURCE93	10,734	0,00	0,00	24,660	-0,244	-0,160	32,807	-0,492	-0,238	42,604	-4,77	-2,46
SOURCE94	13,994	0,00	0,00	92,289	-2,95	-1,74	103,610	-3,87	-3,00	134,880	-14,1	-8,44
SOURCE95	14,895	0,00	0,00	225,177	0,00	0,00	640,611	-17,7	-7,28	*[1]	-	-

Figuur 4.17.

Voetnoot [1] : bij deze bron p(0) = 0,001 en #bladeren = 8192 was het programma niet in staat $\bar{\ell}_s$ uit te rekenen.

```
OK, SEG #PROG1
GO
GEEF FILENUMMER        SOURCE :  90
GEEF FILENUMMER        REPORT :   2
# LEAVES =  256
TUNSTALL TREE? (J/N) : J
KANS OP EEN 0 IS : 0.3000
GEMIDDELDE LENGTE TUNSTALL BOOM =     8.970

  SRCLEN    COMPR.   #SEGMNTS   ADAPT   SEG-LEN   PROC-DIF

   10002  0.92701      1159      223    8.842   -1.42
   20009  0.91679      2293       77    8.913  -0.634
   30008  0.91282      3424       45    8.916  -0.601
   40000  0.90740      4537       28    8.937  -0.363
   50006  0.90565      5661       37    8.950  -0.225
   60007  0.90309      6774       17    8.956  -0.150
   70003  0.90179      7891       12    8.958  -0.135
   80001  0.90059      9006       12    8.960  -0.103
   90007  0.89886     10113        7    8.962 -0.836E-01
  100002  0.89814     11227        5    8.962 -0.898E-01
  110008  0.89826     12352       12    8.964 -0.690E-01
  120003  0.89804     13471       15    8.965 -0.496E-01
  130003  0.89776     14589       10    8.963 -0.794E-01
  140005  0.89791     15714       11    8.964 -0.600E-01
  150007  0.89788     16836       10    8.964 -0.600E-01
  160003  0.89813     17963        8    8.963 -0.703E-01
  170002  0.89754     19073        6    8.964 -0.600E-01
  180002  0.89719     20187        5    8.965 -0.496E-01
  190000  0.89684     21300        2    8.967 -0.331E-01
  200004  0.89606     22402        2    8.967 -0.331E-01
  210002  0.89599     23520        5    8.968 -0.240E-01
  220003  0.89595     24639        3    8.968 -0.240E-01
  230008  0.89586     25757        6    8.967 -0.331E-01
  240003  0.89562     26869        4    8.966 -0.435E-01
  250003  0.89548     27984        7    8.968 -0.241E-01
  260009  0.89606     29123        2    8.966 -0.435E-01
  270008  0.89580     30234        6    8.968 -0.228E-01
  280002  0.89596     31359        4    8.968 -0.228E-01
  290000  0.89578     32472        4    8.968 -0.194E-01
  300004  0.89585     33595        5    8.968 -0.228E-01
  310001  0.89574     34710        4    8.968 -0.228E-01
  320004  0.89566     35827        3    8.968 -0.228E-01
  330002  0.89536     36934        6    8.969 -0.104E-01
  340004  0.89542     38056        1    8.969 -0.104E-01
  350007  0.89534     39172        1    8.969 -0.104E-01
  360011  0.89511     40281        2    8.969 -0.104E-01
  370009  0.89505     41397        8    8.969 -0.104E-01
  380007  0.89472     42500        4    8.969 -0.104E-01
  390003  0.89448     43606        5    8.969 -0.104E-01
  400003  0.89441     44721        1    8.970   0.000
  410002  0.89440     45838        2    8.970   0.000
  420007  0.89446     46960        3    8.970   0.000
  430003  0.89443     48076        5    8.969 -0.104E-01
  440005  0.89444     49195        1    8.969 -0.104E-01
  450005  0.89443     50312        4    8.969 -0.104E-01
  460011  0.89447     51433        6    8.969 -0.104E-01
  470008  0.89435     52544        7    8.969 -0.104E-01
  480004  0.89439     53664        4    8.969 -0.104E-01
  490007  0.89445     54786        3    8.969 -0.104E-01
  500007  0.89437     55899        4    8.969 -0.104E-01
  510003  0.89428     57011        2    8.969 -0.104E-01
  520009  0.89426     58128        1    8.969 -0.104E-01
  530007  0.89421     59242        2    8.969 -0.104E-01
  540005  0.89433     60368        4    8.970   0.000
  550009  0.89417     61475        4    8.970   0.000
  560001  0.89404     62583        2    8.970   0.000
  570005  0.89394     63694        2    8.970   0.000
  580004  0.89370     64794        5    8.970   0.000
  590007  0.89366     65908        0    8.970   0.000
  600010  0.89356     67018        2    8.970   0.000
  610000  0.89342     68123        3    8.970   0.000
  620000  0.89350     69246        3    8.970   0.000
  630004  0.89357     70369        2    8.970   0.000
  640002  0.89342     71474        2    8.970   0.000
  650000  0.89345     72593        3    8.970   0.000
  660000  0.89336     73702        3    8.970   0.000
  670003  0.89327     74812        0    8.970   0.000
  680003  0.89325     75927        5    8.970   0.000
  690009  0.89321     77040       10    8.970   0.000
  700000  0.89322     78157        7    8.970   0.000
  710002  0.89315     79267        2    8.970   0.000
  720006  0.89316     80385        2    8.970   0.000
  730011  0.89309     81496        1    8.970   0.000
  740003  0.89308     82610        5    8.970   0.000
  750002  0.89309     83727        1    8.970   0.000
  760002  0.89312     84847        0    8.970   0.000
  770000  0.89312     85963        0    8.970   0.000
  780003  0.89319     87086        0    8.970   0.000
  790004  0.89329     88213        3    8.970   0.000
  800001  0.89316     89316        3    8.970   0.000
  810007  0.89309     90426        3    8.970   0.000
  820001  0.89317     91550        1    8.970   0.000
  830007  0.89318     92668        2    8.970   0.000
  840004  0.89324     93791        3    8.970   0.000
  850006  0.89321     94904        1    8.970   0.000
  860006  0.89310     96009        2    8.970   0.000
  870005  0.89310     97125        0    8.970   0.000
  880005  0.89307     98238        1    8.970   0.000
  890003  0.89307     99354        3    8.970   0.000
  900000  0.89301    100464        1    8.970   0.000
  910005  0.89301    101580        2    8.970   0.000
  920011  0.89299    102695        4    8.970   0.000
  930005  0.89301    103813        2    8.970   0.000
  940001  0.89302    104930        2    8.970   0.000
  950000  0.89301    106045        5    8.970   0.000
  960001  0.89303    107164        1    8.970   0.000
  970001  0.89315    108294        6    8.970   0.000
  980006  0.89311    109407        5    8.970   0.000
  990005  0.89303    110513        1    8.970   0.000
  999993  0.89305    111631        0    8.970   0.000

**** STOP
```

Figuur 4.18a.

```
OK, SEG #PROG3
GO
GEEF FILENUMMER        SOURCE : 90
GEEF FILENUMMER        REPORT : 13
# LEAVES = 1024.
TUNSTALL TREE? (J/N) : J
KANS OP EEN 0 IS : 0.3000
GEMIDDELDE LENGTE TUNSTALL BOOM =    11.239

  SRCLEN    COMPR.   #SEGMNTS  ADAPT  SEG-LEN  PROC-DIF

       0  0.00000      89300      0   11.234 -0.405E-01
   10008  1.09992      90196      6   11.234 -0.402E-01
   20001  0.99785      91091      7   11.234 -0.402E-01
   30005  0.96577      91993     10   11.234 -0.443E-01
   40006  0.94431      92873      5   11.235 -0.339E-01
   50011  0.93395      93766      4   11.234 -0.368E-01
   60007  0.92619      94653      6   11.234 -0.376E-01
   70003  0.92107      95543      8   11.234 -0.431E-01
   80010  0.91711      96433      5   11.234 -0.413E-01
   90003  0.91350      97317      7   11.233 -0.462E-01
  100007  0.91092      98205      7   11.234 -0.411E-01
  110002  0.90915      99096      6   11.234 -0.428E-01
  120001  0.90764      99987      4   11.234 -0.428E-01
  130003  0.90658     100881      5   11.234 -0.363E-01
  140000  0.90606     101780      2   11.234 -0.381E-01
  150010  0.90539     102677      6   11.234 -0.435E-01
  160010  0.90499     103576      5   11.234 -0.361E-01
  170002  0.90374     104459      3   11.234 -0.398E-01
  180013  0.90292     105349      7   11.234 -0.378E-01
  190000  0.90220     106237     10   11.234 -0.381E-01
  200001  0.90114     107118      5   11.234 -0.381E-01
  210002  0.90065     108009      5   11.235 -0.344E-01
  220003  0.90025     108901      3   11.234 -0.378E-01
  230013  0.89990     109794      2   11.234 -0.361E-01
  240001  0.89928     110678      3   11.234 -0.361E-01
  250000  0.89879     111565      3   11.234 -0.398E-01
  260007  0.89912     112473      3   11.235 -0.344E-01
  270012  0.89873     113362      8   11.235 -0.307E-01
  280002  0.89881     114262      6   11.235 -0.307E-01
  290006  0.89828     115146      1   11.235 -0.307E-01
  300003  0.89815     116040      5   11.235 -0.344E-01
  310007  0.89791     116931      6   11.234 -0.383E-01
  320003  0.89764     117820      8   11.235 -0.346E-01
  330003  0.89723     118704      2   11.234 -0.364E-01
  340000  0.89735     119605      3   11.235 -0.346E-01
  350002  0.89716     120496      4   11.234 -0.366E-01
  360000  0.89683     121381      4   11.234 -0.384E-01
  370005  0.89663     122271      4   11.235 -0.346E-01
  380004  0.89614     123149      4   11.235 -0.329E-01
  390001  0.89584     124033      3   11.235 -0.329E-01
  400000  0.89574     124925      8   11.235 -0.309E-01
  410005  0.89572     125820      5   11.235 -0.309E-01
  420009  0.89562     126712      7   11.235 -0.349E-01
  430007  0.89563     127608      6   11.235 -0.329E-01
  440000  0.89563     128503      8   11.235 -0.331E-01
  450005  0.89559     129397      3   11.235 -0.329E-01
  460009  0.89559     130293      3   11.235 -0.312E-01
  470000  0.89544     131181      3   11.235 -0.292E-01
  480001  0.89545     132077      3   11.235 -0.309E-01
  490002  0.89544     132972      3   11.235 -0.312E-01
  500008  0.89528     133860      6   11.235 -0.309E-01
  510002  0.89515     134748      2   11.235 -0.292E-01
  520010  0.89506     135639      3   11.235 -0.309E-01
  530002  0.89494     136527      2   11.235 -0.289E-01
  540010  0.89496     137424      2   11.236 -0.270E-01
  550008  0.89473     138306      4   11.235 -0.272E-01
  560001  0.89457     139191      6   11.236 -0.252E-01
  570007  0.89442     140078      1   11.236 -0.269E-01
  580002  0.89415     140956      6   11.236 -0.252E-01
  590008  0.89410     141848      4   11.236 -0.232E-01
  600005  0.89396     142733      3   11.236 -0.249E-01
  610007  0.89369     143611      3   11.236 -0.252E-01
  620006  0.89378     144510      5   11.236 -0.232E-01
  630000  0.89379     145404      2   11.236 -0.249E-01
  640003  0.89365     146289      1   11.236 -0.232E-01
  650004  0.89364     147182      3   11.236 -0.215E-01
  660002  0.89349     148066      6   11.236 -0.267E-01
  670007  0.89342     148955      5   11.236 -0.197E-01
  680000  0.89338     149845      4   11.236 -0.197E-01
  690010  0.89329     150733      3   11.235 -0.272E-01
  700000  0.89323     151621      7   11.235 -0.287E-01
  710009  0.89311     152507      2   11.236 -0.235E-01
  720008  0.89311     153400      2   11.236 -0.235E-01
  730006  0.89304     154288      2   11.236 -0.197E-01
  740008  0.89299     155177      7   11.237 -0.178E-01
  750000  0.89294     156066      4   11.236 -0.197E-01
  760000  0.89298     156962      2   11.236 -0.197E-01
  770005  0.89294     157852      4   11.236 -0.195E-01
  780007  0.89303     158752      6   11.236 -0.212E-01
  790000  0.89312     159652      3   11.236 -0.212E-01
  800008  0.89295     160532      3   11.236 -0.212E-01
  810003  0.89288     161419      4   11.236 -0.232E-01
  820001  0.89290     162313      2   11.236 -0.212E-01
  830003  0.89292     163208      5   11.236 -0.232E-01
  840011  0.89294     164103      2   11.236 -0.232E-01
  850000  0.89283     164986      1   11.236 -0.215E-01
  860002  0.89275     165872      2   11.236 -0.195E-01
  870004  0.89274     166764      0   11.236 -0.195E-01
  880009  0.89268     167652      2   11.237 -0.178E-01
  890003  0.89265     168541      2   11.237 -0.158E-01
  900002  0.89255     169425      1   11.237 -0.158E-01
  910003  0.89256     170318      4   11.237 -0.178E-01
  920013  0.89252     171208      3   11.237 -0.178E-01
  930000  0.89252     172100      2   11.237 -0.178E-01
  940007  0.89249     172990      3   11.237 -0.178E-01
  950004  0.89244     173877      5   11.237 -0.178E-01
  960002  0.89246     174772      7   11.236 -0.197E-01
  970009  0.89255     175673      2   11.236 -0.195E-01
  980003  0.89249     176559      5   11.236 -0.212E-01
  990003  0.89239     177442      4   11.236 -0.232E-01
  999994  0.89240     178335      4   11.236 -0.212E-01

**** STOP
```

Figuur 4.18b.

5.0 Rissanen's universele kodeersysteem.

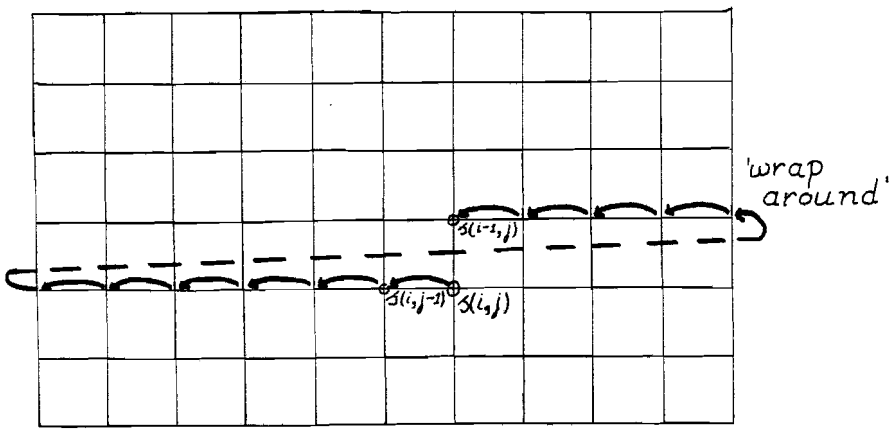
In [1983] geeft Rissanen een nieuw universeel data reduktie systeem, gebaseerd op de ideeën van Rissanen en Langdon [1981-a], zie hoofdstuk 2.3. De daarin voorgestelde splitsing van het systeem in een modelvormend en een koderend deel wordt door Rissanen overgenomen.

Daar, zoals gesteld in Rissanen en Langdon [1981-a], voor het koderend deel veel goede methoden beschikbaar zijn, o.a. arithmetische kodering, beperkt Rissanen zich in [1983] tot het beschrijven van de modelvormer.

Rissanen noemt het Ziv-Lempel algoritme als een voorbeeld van een goede universele kode en heeft hierop het volgende komentaar.

We beschouwen een twee-dimensionaal begrensd vlak van binaire stochastische variabelen s(i,j).
Een goed model neemt voor elk punt s(i,j) de afhankelijkheden van de omliggende punten mee, zowel in horizontale als in vertikale richting.
Een 'segment-achtige' kode neemt slechts de afhankelijkheden binnen een segment mee. We zullen dus grote, twee-dimensionale segmenten moeten gebruiken.
Voor het Ziv-Lempel algoritme proberen we te werken met twee-dimensionale segmenten ter hoogte k. Om dan afhankelijkheden verder weg dan k lijnen hoog mee te kunnen nemen gebruiken we een 'wrap around' techniek. d.w.z. als het segment verder groeit dan de kantlijn dan wordt het aan de andere kant op de volgende k lijnen voortgezet. Zie figuur 5.1.

Stel dat het vlak 2000 symbolen breed is, dan zou de Ziv-Lempel boom, zie hoofdstuk 4.3, 2000 symbolen diep moeten groeien om bij k=1 de variabele s(i,j) af te kunnen laten hangen van de direkt bovenliggende variabele.



Figuur 5.1.

Een eindig dokument bezit niet zoveel symbolen dat de boom deze diepte zal bereiken.

Indien we k groot kiezen, b.v. k=10, om voldoende vertikale afhankelijkheden mee te kunnen nemen, dan heeft elke knoop in de boom 2^{10} opvolgers. Weer zouden we door de datarij heen zijn voor we voldoende afhankelijkheid in de horizontale richting mee kunnen nemen.

Deze problemen ontstaan daar in 'segment-achtige' kodes de segmenten disjunkt moeten zijn, en tevens groot om voldoende afhankelijkheid mee te nemen. Hierdoor zijn er te weinig segmenten, d.w.z. tellingen, om tot een redelijke schatting van de bron te komen.
Rissanen's universele kode is dan ook van het kontekst type. Zie hoofdstuk 2.3.

5.1 Enkele struktuur funkties.

We zullen hier wat dieper op de kontekst kodes ingaan.

Voor de eenvoud nemen we aan dat het bronalfabet binair is. Dit is geen wezenlijke beperking, zoals al duidelijk mag zijn.

Zie hoofdstuk 2.3, de kontekst is een equivalentieklasse van eindige datarijen.

De kontekst van een te koderen symbool s_t, uit een datarij s, is de equivalentieklasse $f(s_1^{t-1})$ waartoe de, al verwerkte, prefix van de datarij behoort.

Een realistische subklasse van struktuurfunkties zijn de funkties die dezelfde waarde toekennen aan alle rijen s met dezelfde k laatste, d.i. meest recente, symbolen. Dus als:

$$f(s_1^{\ell})=f(r_1^{m}),\ m\geq k,\ \ell\geq k,\ \text{dan geldt } s_{\ell-k+1}^{\ell} = r_{m-k+1}^{m}$$

m.a.w. de struktuurfunktie f wordt gegeven door:

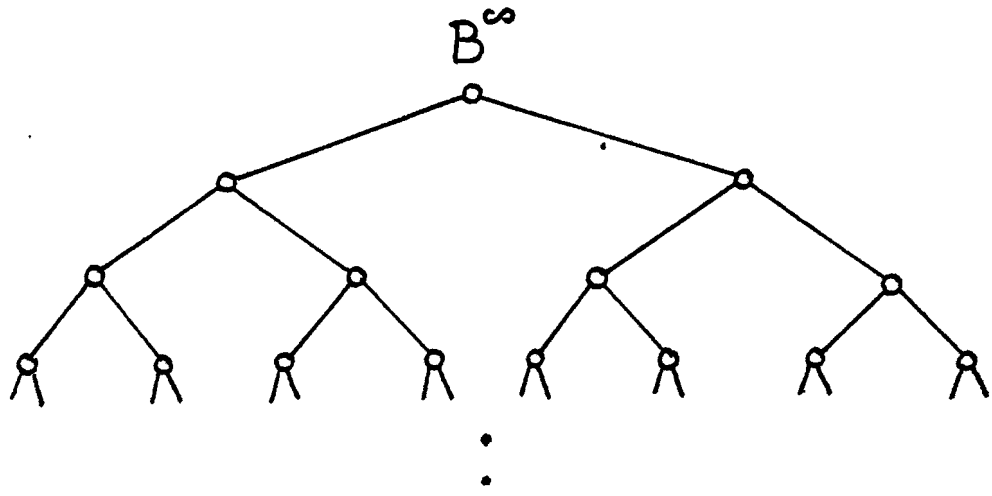
$$f : S^* \rightarrow \{0, 1, \ldots, 2^k-1\}.$$

$$f(s_1^t) = \sum_{i=0}^{k-1} s_{t-i}2^i. \qquad (5.1)$$

Laten we afspreken dat datarijen als het ware vooraf gegaan worden door zoveel nullen als nodig zijn, Dus $s_{-\infty}^{0} \triangleq \underline{0}$, dan kan de aanname $m\geq k$, $\ell\geq k$, in (5.1) vervallen.

De verschillende equivalentieklassen zijn nu in feite niets anders dan de verschillende toestanden van een k^e orde binaire Markov keten.

We kunnen deze konteksten ook beschrijven m.b.v. komplete 2-bomen. Noem B^{∞} de komplete 2-boom met oneindige diepte, zie figuur 5.2.



Figuur 5.2.

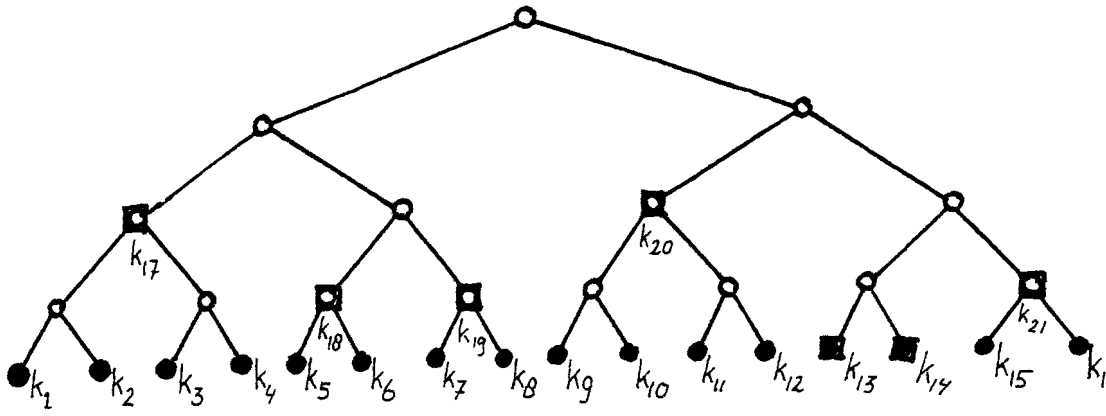
Als s_1^{t-1} de voorgeschiedenis is van s_t, dan zijn s_{t-1}, s_{t-2}^{t-1}, s_{t-3}^{t-1} ect. mogelijke konteksten. Deze konteksten vinden we terug in de boom als de knopen op het pad, gegeven door s_{t-1}, s_{t-2}, s_{t-3}, etc. Dus de rij s_1^{t-1} wordt achterstevoren op B^{∞} afgepast en aangevuld, door oneindig veel nullen. In figuur 5.2 is dit aangegeven voor $s_{t-1}s_{t-2}s_{t-3}\ldots = 010\ldots$

De struktuurfunktie f wordt in B^{∞} gegeven door een aantal gemerkte knopen. Specifieker, als f gegeven is als in (5.1) dan zijn alle 2^k knopen, op diepte k in de boom, gemarkeerd.
Zien we de gemarkeerde knopen als de bladeren van een kontekstboom B, dan valt op dat B gebalanceerd is.

Een eerste uitbreiding van de struktuurfunkties f volgens (5.1) wordt gegeven door te eisen dat de kontekstboom B een willekeurige komplete 2-boom moet zijn. Dat B kompleet moet zijn volgt uit de behoefte om voor elk verleden een kontekst te kunnen vinden.

Wat hebben we nu eigenlijk gedaan?

Van de komplete struktuurboom B is b een blad op de maximale diepte. Noem deze diepte k. De gebalanceerde boom B^1 met diepte k definiëert een k^e orde Markov bron, waarvan echter een aantal toestanden dezelfde konditionele kans op een nul hebben. De struktuurfunktie zal verschillende van deze toestanden afbeelden op dezelfde klasse. Zie figuur 5.3.



Figuur 5.3,

⊡ is een blad van B,
◙ is een blad van B en B^1,
• is een blad van B^1

De konteksten k_1, k_2, k_3 en k_4 uit B^1 zijn eigenlijk equivalent en worden via B afgebeeld op k_{17}, evenzo k_5 en k_6 op k_{18}; k_7 en k_8 op k_{19}; k_9, k_{10}, k_{11} en k_{12} op k_{20} en k_{15} en k_{16} op k_{21}.

We zijn nu klaar voor de introduktie van de door Rissanen gebruikte struktuurfunkties.

We noteren een permutatie π van de natuurlijke getallen als

$$i \to \pi_i, \quad i \text{ en } \pi_i \text{ zijn natuurlijke getallen.} \qquad (5.2)$$

Hiermee bedoelen we dat het getal i vervangen wordt door het getal π_i.

Met de permutatie π definiëren we voor elke rij $s_1^{t-1} \in S^*$ een nieuwe rij $\sigma(s_1^{t-1}) \in S^*$ volgens

$$\sigma(s_1^{t-1}) = s_{t-\pi_1} s_{t-\pi_2} \cdots s_{t-\pi_{t-1}} \qquad (5.3)$$

b.v. als π de identieke permutatie is, dus $\pi_i = i$, dan is $\sigma(s_1^{t-1})$ precies de rij s_1^{t-1} achterstevoren genoteerd.
$\sigma(.)$ noemen we de <u>sorteerfunktie</u>.

We zullen de volgende konventie aanhouden:

$$s_j \stackrel{\Delta}{=} 0, \text{ als } j \leq 0 \qquad (5.4)$$

zodat (5.3) voor elke s en elke π gedefiniëerd is.

$\sigma(s_1^{t-1})$ zullen we ook noteren als de rij z_1^{t-1}, dus

$$z_i \stackrel{\Delta}{=} s_{t-\pi_i}, \quad 1 \leq i < t. \qquad (5.5)$$

z_1^{t-1} noemen we de <u>gesorteerde rij</u> van s_1^{t-1}, of ook wel het <u>gesorteerde verleden</u> van s_t.

Nu de sorteerfunktie geïntroduceerd is kunnen we (5.1) op een nettere manier weergeven, dus als f de struktuurfunktie is die alle eindige rijen met dezelfde laatste k symbolen op dezelfde klasse afbeeldt, dan is f te schrijven als:

$$f : S^* \to S^k$$
$$f(s_1^{t-1}) = z_1^k \qquad (5.6)$$

Hierin nemen we impliciet de identieke permutatie aan, dus

$$z_1^k = s_{t-1}s_{t-2} \cdots s_{t-k}.$$

De boom B^∞, figuur 5.2, herinterpreteren we zo, dat voor elke s_1^{t-1} de gesorteerde rij z_1^{t-1} op B^∞ kan worden afgepast.

Laat B een willekeurige eindige komplete sub-boom van B^∞ zijn, die de wortel met B^∞ gemeen heeft, dan kunnen we de struktuurfunktie f, met impliciete sorteerfunktie $\sigma(.)$, definiëren als:

$$f : S^* \to L_B$$
$$f(s_1^{t-1}) = z_1^p, \text{ met } z_1^p \in L_B \text{ (dus p is i.h.a. geen konstante.)} \qquad (5.7)$$

5.2 De 'finitely generated sources'.

Rissanen definiëert een klasse van 'finitely generated sources' (FGS). Hij laat dan zien dat zijn algoritme goed werkt voor deze FGS.
Voor het behandelen van het algoritme zullen we eerst kijken wat Rissanen met de klasse FGS bedoelt.

Laat een rij van n natuurlijke getallen gegeven zijn als
$i_1, i_2, \ldots, i_n$ met $0 < i_1 < i_2 < \ldots < i_n$.

Laat k_1^∞ de oneindige rij van gehele getallen zijn, dus $k_i = i$.

Uit k_1^∞ vormen we ℓ_1^∞ door de getallen i_1^n te laten volgen door de rij k_1^∞, waaruit de getallen $i_1, i_2, \ldots, i_n$ weggelaten zijn.

Nu definiëren de indices i een permutatie π^0 van $\mathbb{N}_1$ volgens:

$$i \to \pi_i^0 \triangleq \ell_i \tag{5.8}$$

Een voorbeeld.

$n = 3,\ i_1 = 2,\ i_2 = 5,\ i_3 = 6.$

$k_1^\infty = 1, 2, 3, \ldots$

$\ell_1^\infty = 2, 5, 6, 1, 3, 4, 7, 8, \ldots$

Definiëer nu een struktuurfunktie f^0, met permutatie π^0 en dus sorteerfunktie $\sigma^0(.)$, als:

$$\begin{aligned} &f^0 : S^* \to L_{B^0} \\ &f(s_1^{t-1}) = s_{t-i_1} s_{t-i_2} \cdots s_{t-i_n} \end{aligned} \tag{5.9}$$

Hierin geldt weer de afspraak, dat $s_i = 0$ voor $i \leq 0$.

Met B^0 wordt de bij π^0 behorende gebalanceerde binaire boom ter diepte n bedoeld. Deze boom is een kontekst boom.

Een stationair ergodisch proces (S,P) heet '<u>finitely generated</u>' (FG) als geldt:

$$P(su) = P(s)P(u \mid f^0(s)) \text{ voor alle } s \in S^*,\ u \in S \tag{5.10}$$

Van de rij i_1^n, die we de <u>genererende indices</u> zullen noemen, eisen we dat ze uniek is, zodat deze rij bepaald kan worden.

De entropie van een FG bron S met bijbehorende kontekstboom B^0 wordt gegeven door:

$$H^0(S \mid B^0) = \sum_{z \in L_{B^0}} P(z) H(S \mid Z=z) = H_\infty(S) \tag{5.11}$$

Een FG bron is eigenlijk een Markovbron met eindige orde i_n.
De 2^{i_n} toestanden van de bron zijn onder te verdelen in 2^n equivalentieklassen.

Een voorbeeld.

S is een binaire twee-dimensionale bron, d.w.z. dat S een eindige rij s = s(1,1) s(1,2) ... s(1,L) s(2,1) s(2,2) ... s(2,L) ... s(M,1) ... s(M,L) genereert. L is de pagina breedte en M is het aantal regels op een pagina. b.v. M = 4000, L = 2000, dus s bevat 8.10^6 symbolen.

Voor S geldt, dat de waarde van s(i,j) alleen afhangt van s(i-1,j) en s(i,j-1), het symbool direkt boven resp. direkt naast s(i,j). Aan de randen van het, eindige, veld gelden, in praktische gevallen, natuurlijk andere relaties dan in het midden. De bron S heeft hier geen last van.
S is een L^e orde Markov bron, dus met $2^{2000} \approx 10^{660}$ toestanden. Duidelijk is dat dit te veel toestanden zijn om met 'segment achtige' universele kodes een goede kompressie te kunnen bereiken.
Zelfs het afschatten van de 2^{2000} verschillende kansen is een onmogelijke zaak daar slechts c.a. 8.10^6 tellingen beschikbaar zijn.
Maken we gebruik van het feit dat S een FG bron is met genererende indices $i_1 = 1$, $i_2 = 2000$, dan behoeven we slechts 4 kansen te schatten, wat goed mogelijk is voor dit probleem.
Ook het aantal te schatten parameters is in de 'ongesorteerde' versie veel te groot voor praktische toepassingen. Er zijn momenteel nog geen 2^{2000} geheugenelementen geproduceerd!

Stel dat we niet exakt de genererende indices van S kennen. Een intelligente gok is te veronderstellen dat de omliggende symbolen s(i,j-1) s(i-1,j+1) s(i-1,j) s(i-1,j-1) redelijk voldoen. Dan behoeven we nog slechts 16 kansen te schatten. Een hanteerbaar aantal.

5.3 Rissanen's modelvormer.

Rissanen ontwerpt in [1983] een modelvormer voor de klasse van FGS. In principe komt dit neer op:

- Het vinden van de genererende indices i_1^n, en dus impliciet ook de 'orde' van de FGS.
- Het vinden van goede schattingen voor de konditionele kansen behorende bij de gevonden konteksten.

Een extra komplikatie ontstaat doordat de twee taken tegelijk moeten gebeuren.

In de beschrijving met komplete binaire bomen gaat dit over in:

- Het vinden van de juiste vorm van de boom.
- Het vinden van schattingen voor de konditionele kansen.

In het laatste deel van het vorige hoofdstuk kozen we een bepaalde sorteerfunktie $\sigma(.)$ volgens wat we een intelligente gok noemden. We komen daar nu even op terug.

Stel dat we behalve over de funktie S ook over een komplete binaire boom T beschikken. Elke knoop in deze boom definiëert een mogelijke kontekst.
Met deze boom kunnen we voor een bepaalde rij s een schatting maken voor de verschillende konditionele kansen, door het voorkomen van konteksten te tellen.

Hoe gaat dit nu precies in zijn werk?

Aan elke knoop x van T worden twee tellingen $c_0(x)$ en $c_1(x)$ toegevoegd. Voor elk element s_t in de rij wordt de rij $z_1^{t-1} = \sigma(s_1^{t-1})$, voor zover nodig aangevuld met nullen, afgepast langs de boom.

Van elke bezochte knoop x wordt de telling $c_0(x)$ (indien $s_t=0$) of $c_1(x)$ (indien $s_t=1$) één opgehoogd. Dus

$$c(x) \triangleq c_0(x) + c_1(x) \tag{5.12}$$

geeft aan hoe vaak de kontekst x in de rij voorkomt.

De bron is ergodisch dus zullen de frakties $\frac{c_0(x)}{c(x)}$ en $\frac{c_1(x)}{c(x)}$ steeds betere schattingen zijn voor $p(0\,|\,x)$ en $p(1\,|\,x)$.
Evenzo is $\frac{c(x)}{c(\lambda)}$ een schatting voor $p(x)$.

We noteren:

$$\begin{aligned} f(u\,|\,x) &\triangleq \frac{c_u(x)}{c(x)}, \quad u = 0,1 \\ f(x) &\triangleq \frac{c(x)}{c(\lambda)}, \end{aligned} \tag{5.13}$$

voor alle $x \in T$.

Als B nu een komplete sub-boom van T is, en B en T hebben de wortel gemeen, dan heeft dit model een konditionele entropie gegeven door:

$$H(S\,|\,B) = \sum_{x \in L_B} f(x) H(S\,|\,x) \tag{5.14}$$

met

$$H(S\,|\,x) = -\sum_{s \in S} f(s\,|\,x) \log f(s\,|\,x) \tag{5.15}$$

Het probleem is dat we niet over een boom B kunnen beschikken, daar we van de bron de genererende indices niet kennen, m.a.w. we kunnen de diepte m van B niet bepalen.

We moeten dus een voorziening treffen om de kontekst boom zo groot te laten groeien als dat nodig is. Echter, de boom moet het liefst niet veel groter worden dan nodig is, en ook zouden we graag willen dat m niet veel groter is dan het aantal indices.

Rissanen definiëert een algoritme CONTEXT.
Dit algoritme genereert een rij van steedt groter wordende komplete binaire bomen T(t), met per knoop twee tellingen, waarvoor voor alle knopen x geldt:

$$\begin{aligned} & c_u(x0) + c_u(x1) = c_u(x), \quad u = 0,1 \\ & c(x0) + c(x1) = c(x), \end{aligned} \tag{5.16}$$

indien al deze tellingen groter dan nul zijn.

Deze tellingen uit het algoritme CONTEXT zijn niet gelijk aan de voorgenoemden, zie formule (5.12). Het verschil is dat enkele voorkomens van bepaalde konteksten x in het begin van de rij niet meegeteld worden. Dit is echter een eindig aantal en dus geldt ook voor Rissanen's tellingen dat de frakties $\frac{c_u(x)}{c(x)}$ en $\frac{c(x)}{c(\lambda)}$ steeds betere schattingen voor de bronkansen zijn. Dus, zoals in (5.13), noteren we:

$$\begin{aligned} & f_t(u|x) \triangleq \frac{c_u(x)}{c(x)}, \quad u = 0,1 \\ & f_t(x) \quad \triangleq \frac{c(x)}{c(\lambda)} \end{aligned} \tag{5.17}$$

Het algoritme CONTEXT genereert steeds groter wordende bomen. Uit de boom T(t-1) moet nog een aantal knopen gekozen worden die de werkelijke kontekst van het volgende symbool s_t vormen.

Rissanen gebruikt hiervoor een 'intelligent' algoritme dat zelf de knopen kiest.
Hij introduceert kosten voor elke geselekteerde kontekst. Een kontekst x wordt slechts dan geaccepteerd in de verzameling van geselekteerde konteksten als de opbrengst, d.i. de daling van de entropie, meer is dan de kosten.

Zoals we al hebben gezien zijn de geselekteerde konteksten te beschrijven als de bladeren van een komplete sub-boom B(t) van T(t), waarbij B(t) en T(t) de wortel gemeen hebben.
Stel dat x0 en x1 twee geselekteerde konteksten zijn. Wat is de toename in de konditionele entropie als x0 en x1 vervangen worden door x? d.w.z. we krimpen B(t) bij de bladvader x in. Noem deze ingekrompen boom $B^1(t)$, dan is de toename $\Delta(t,x)$ te schrijven als:

$$\Delta(t,x) \stackrel{\Delta}{=} H\left(S \middle| B^1(t-1)\right) - H\left(S \middle| B(t-1)\right) \qquad (5.18)$$

$$= f_{t-1}(x) H\left(S \middle| x\right) - f_{t-1}(x0) H\left(S \middle| x0\right) - f_{t-1}(x1) H\left(S \middle| x1\right)$$

Hierin zijn de entropieën gegeven door (5.14) en (5.15).

Uit de stelling van Gibbs volgt:

$$\Delta(t,x) \geqslant 0 \text{ , voor alle t en x} \qquad (5.19)$$
en alle bronnen S en bomen B(t).

Rissanen's selektie regel luidt:

Selekteer langs het pad z_1^{t-1} ($= \sigma(s_1^{t-1})$) van de boom T(t-1) als kontekst de knoop x die het diepst ligt en voldoet aan:

$$\Delta(t,x) > \frac{\log t}{t} \qquad (5.20a)$$

$$\ell(x) \leqslant \beta \log t \qquad (5.20b)$$

$$\min \{ c(x0), c(x1) \} \geqslant 2\alpha t/\sqrt{\log t} \qquad (5.20c)$$

α en β zijn positieve getallen, hun doel is om het zoeken in de boom te begrenzen. Dit is alleen zinvol voor eindige datarijen, voor oneindig lange rijen voldoet elke positieve waarde. De grenzen (5.20b) en (5.20c) vereenvoudigen het bewijs van Rissanen's hoofdstelling.

Als op een bepaald moment de ratio's $f_t(x)$ en $f_t(u|x)$ gestabiliseerd zijn en daardoor B(t) de konstante vorm B aanneemt, dan volgt eenvoudig dat de totale kosten C_T, met

$$C_T \triangleq H(S|B) + (\|L_B\| \log t)/t \qquad (5.21)$$

lokaal geminimaliseerd is.
Hiermee bedoelen we dat C_T toeneemt zowel als we de boom B bij één willekeurig blad uitbreiden of bij één willekeurige bladvader inkrimpen.

Rissanen geeft de volgende argumenten om de kosten per kontekst knoop als (log t)/t te zien.

Als eerste noemt hij dat deze vorm het algoritme het gewenste gedrag laten vertonen. Hiermee bedoelt hij dat met deze kosten voor de klasse FGS de entropie gehaald wordt o.d.d. Dit is Rissanen's hoofdstelling, zie verderop.

Het andere argument is gebaseerd op de observatie dat als het een twee slags algoritme zou zijn waar in de eerste slag, met een datarij ter lengte t, alleen de optimale boom B gezocht wordt. Dan zijn alle tellingen in log t bits representeerbaar. Dus zouden er $\|L_B\| \log t$ bits nodig zijn voor het overzenden van deze tellingen. (Opmerking: eigenlijk moet hier een faktor twee bij daar twee tellingen per kontekst verzonden moeten worden.)
In [1983] zegt Rissanen, zonder daar het bewijs te leveren, dat dezelfde kosten gelden in het adaptieve geval.

5.4 Rissanen's hoofdstellling.

We spreken nu nog een notatie af.
Gegeven een rij genererende indices i_1^n en een willekeurige sorteerfunktie σ, die dus niet van de indices afhangt. We noemen B_σ de gebalanceerde binaire boom ter diepte m, als m voldoet aan de volgende eis.

> m is het kleinste positieve, gehele getal, zodat z_1^m, het gesorteerde verleden ter diepte m, alle relevante symbolen s_{t-i_j}, $1 \leq j \leq n$, bevat.

Ons valt het volgende op:

- $m \geq n$, want z_1^m moet minstens n symbolen bevatten.
- als $m=n$ dan zet $\sigma(s)$ dezelfde n symbolen vooraan als de door i_1^n gedefiniëerde permutatie π^0. Natuurlijk behoeven die elementen niet in dezelfde volgorde te staan.

Noteer met $x^*(t)$ de kontekst die op tijdstip t geselekteerd wordt door de regels (5.20).

Rissanen's hoofdstelling luidt nu alsvolgt:

$$\Pr\{x^*(t) \in L_{B_\sigma}\} \to 1, \text{ als } t \to \infty \qquad (5.22)$$

$$-\frac{1}{t} \log P(s_1^t) \to H^0(S|B^0) \text{ als } t \to \infty \qquad (5.23)$$

In (5.23) wordt met $H^0(S|B^0)$ de entropie bedoeld volgens (5.11). $P(s_1^t)$ wordt gegeven door:

$$P(s_1^t) = \prod_{i=1}^{t} f_{i-1}(s_i|x^*(i)) \qquad (5.24)$$

Dit is dus de door het model aan s_1^t toegekende kans.

5.5 Resultaten en problemen.

Rissanen's algoritme zorgt door het introduceren van kosten dat de kontekst boom het snelst groeit op de belangrijkste punten.
Het gebruiken van een sorteerfunktie σ is een zeer belangrijke beslissing. Zoals een voorbeeld al liet zien is hiermee het komprimeren van bepaalde bronnen mogelijk gemaakt. Kontekst kodeer systemen werden reeds gebruikt in Rissanen en Langdon [1981-a].

Rissanen's hoofdstelling is in het licht van het volgende niet eens zo spektakulair.

Weer behandelen we de klasse FGS.
In plaats van kosten te introduceren, en de boom knoop voor knoop te laten groeien, gaan we veel naïever en grover te werk. We maken de kontekst boom B(t) steeds gebalanceerd. Dus we beslissen op verschillende momenten, door de gehele boom T(t) tot een bepaalde diepte d in beschouwing te nemen, hoe diep de boom B(t) moet zijn. Noem deze diepte m(t).
Op den duur zullen de schattingen $f_t(x)$ en $f_t(u|x)$ zich stabiliseren en is $H(S|B(i))$ konstant. Het is eenvoudig te zien dat dan de boom B(i) ook niet meer verandert. Deze boom is dezelfde als B_σ.
Dit 'algoritme' is beslist kwalitatief slechter, d.w.z. veel komplexer zonder een betere reduktie op te leveren, dan Rissanen's algoritme. Echter, het maakt duidelijk dat voor de klasse FGS, Rissanen's algoritme in feite via schattingen van $H(S_L|S_1^{L-1})$ de eindige geheugendiepte, d.i. de orde, van de Markov bron zoekt.

Daar deze orde van te voren niet bekend is, kan, zie Davisson [1973] en hoofdstuk 3, op geen enkel eindig tijdstip de, foutloze, beslissing genomen worden dat de orde bekend is.
Hoewel Rissanen beweert dat de bomen T(t) gegenereerd door CONTEXT niet veel, b.v. slechts twee keer, groter behoeven te zijn dan de kontekst boom B(t) op dat moment, volgt uit de mogelijke plateau vorming dat de kans dat er FG bronnen bestaan met plateaus groter dan die lengte niet nul is, m.a.w. we kunnen T(t) niet begrenzen of we hebben een kleine, maar positieve, kans dat we B_σ nooit zullen vinden.

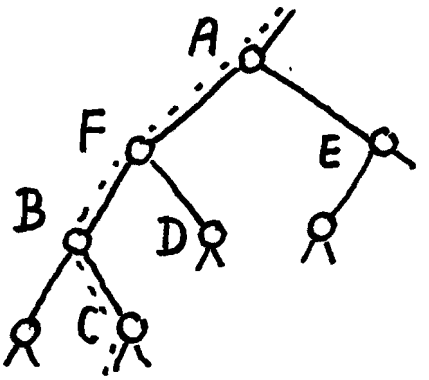
In de preprint van Rissanen's [1983] artikel geeft hij aan hoe het mogelijk zou zijn om vanuit de boom B_σ de genererende indices te vinden en daarmee de grootte van de noodzakelijke kontekst boom te verkleinen door σ aan te passen.
Het probleem hierbij is m.i. dat B_σ niet bekend kan zijn op een eindig tijdstip. We kunnen pas beslissen als de eindige rij verwerkt is, wat wel zin heeft voor een twee slags algoritme, maar geen zin heeft voor een een slags adaptief algoritme. De andere mogelijkheid is 'on the fly' de sorteerfunktie aan te passen, echter, de kriteria die Rissanen hiervoor aangeeft, en vermoedelijk elk mogelijk kriterium, houden de mogelijkheid in dat het geheel alleen maar slechter gemaakt zal worden.

Rissanen's oplossing in [1983] is om enkele verschillende σ's te proberen en de beste te gebruiken. We baseren ons dan weer op de intelligentie van de gebruiker en niet op de (pseudo) intelligentie van het kompressie systeem.

Tot slot wil ik opmerken dat in Rissanen's artikel enkele foutjes zijn geslopen.

Het eerste foutje betreft het selektie kriterium (5.20).
Zie figuur 5.4, waar een gedeelte van T(t-1) getekend is voor een moment t. Op dit moment is s_t het nieuwe symbool en het verleden

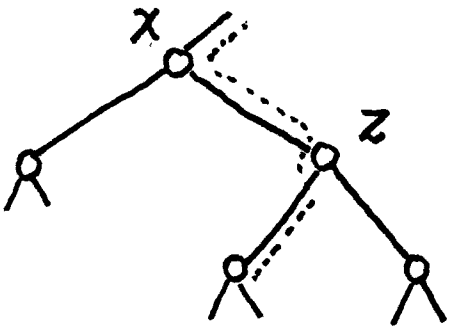
$\sigma(s_1^{t-1})$ loopt o.a. langs de knopen A, B en C. A was een blad van B(t-1), dus de laatste keer dat een pad langs A liep werd deze gekozen.



Figuur 5.4.

Stel dat nu B geselekteerd moet worden. Mijns inziens is dit mogelijk. De selektie van B zal, volgens Rissanen, A's selektie ongedaan maken. De nieuwe kontekst boom B(t) is nu niet kompleet meer, n.l. de paden over E en D hebben geen geselekteerde knoop meer.
De oplossing die ik hiervoor wil aangeven komt na de behandeling van het volgende probleem aan de beurt.

Het tweede foutje is in het bewijs van de hoofdstelling geslopen. Zie de bijlage in Rissanen [1983].
Het bewijs van (5.22) wordt opgesplitst in twee delen.
In het eerste deel bewijst Rissanen dat op den duur alle geselekteerde konteksten minstens zo groot zijn als de knoop $z \in L_{B_\sigma}$ op hetzelfde pad. Echter, bezie de situatie in figuur 5.5. z is een element van L_{B_σ}. x is een bladvader van B_σ.
Op tijdstip t loopt het pad van $\sigma(s_1^{t-1})$ langs x en z.
Rissanen bewijst dat $\Delta(t,x)$ op den duur groter dan nul moet zijn en dus, besluit hij, is x een propere prefix van $x^*(t)$. Echter, zijn selektie regels rechtvaardigen niet meer dan de konklusie dat x een prefix van $x^*(t)$ is, dus hoeft $x^*(t)$ niet minstens zo groot als z te zijn.



Figuur 5.5.

De oplossing hiervoor, en tevens een deel van de oplossing van het eerste probleem, is om de selektie zodanig aan te passen dat als een knoop x aan de selektie regels voldoet, dan niet x maar x0 en x1 geselekteerd worden.

Ook voor het tweede deel van het bewijs van (5.22) is deze aanpassing nodig. Hier wordt bewezen dat $\Delta(t,z) \to 0$ als $t \to \infty$ voor z dieper dan B_σ.

Een laatste opmerking over dit bewijs.
Rissanen beweert dat er een getal M bestaat zodat voor alle $t > M$ geldt $\Delta(t,x) > (\log t)/t$ met x de knoop uit figuur 5.5.
Dit helpt ons niet daar ook dit getal M onbekend is. Of B_σ bereikt is, is in een eindige tijd niet beslisbaar, maar zelfs of langs een enkel pad een deel van B_S gevonden is, is niet te bepalen.

De komplete oplossing voor het eerste probleem bestaat uit het vernieuwde selektie kriterium plus de afspraak dat de boom 'geleidelijk' geselekteerd wordt. Dit is het snelst duidelijk te maken door te zeggen dat in figuur 5.4 ook de knopen D en E geselekteerd moeten worden.

Het belang van Rissanen's algoritme is niet zo zeer het feit dat het FG bronnen optimaal kan koderen, doch dat het voor veel, ook niet FG bronnen, efficiënt kan werken en een goed gebruik maakt van de beschikbare geheugenruimte.

6.0 Slot.

In dit hoofdstuk worden enkele vergelijkingen gemaakt tussen kontekst- en segment kodes.

Als eerste valt op dat de kontekst kodes behoren tot de blokkodes. Hiermee wordt bedoelt dat de datarij in stukken van een vaste lengte worden gehakt, onafhankelijk van de inhoud van het stuk.
Dit hoeft voor segment kodes niet te gelden.
Daar in een blokkode de bloklengten data onafhankelijk zijn is voor deze kodes, o.a Rissanen's arithmetische kodes en Huffman's vaste naar variabele lengte kodes, een stationaire beschrijving te vinden als de bron zelf stationair is.
Voor de data afhankelijke variabele lengte segment kodes is deze beschrijving minder makkelijk te vinden. Zie hoofdstuk 4.2.

Voor geheugenloze bronnen S is in hoofdstuk 4.1.4, stelling 4.2, bewezen dat de entropie $H(S)$ en de optimale reduktie dezelfde zijn voor alle vormen van koderen.
In een volgend hoofdstuk werd duidelijk gemaakt dat deze situatie verandert als we bronnen met geheugen beschouwen. Voor blokkodes zijn de entropie $H_{\infty}(s)$ en de optimale reduktiefaktor waarschijnlijk wel hetzelfde, echter, voor variabele lengte segment kodes blijkt uit een voorbeeld dat de reduktie faktor beter kan zijn dan de entropie, hoewel dit een zeer gekunsteld voorbeeld is.

Zoals Davisson's resultaat in [1973] en hoofdstuk 3 laten zien, zal het moeilijk, zo niet onmogelijk, zijn om een eenvoudige universele kode te maken voor de klasse van alle eindige orde Markov bronnen.
Een beter kwaliteits kriterium i.p.v. de redundantie t.o.v. $H_{\infty}(S)$ kan de redundantie t.o.v. $H_L(S)$ dan wel $H(S_L | S_1^{L-1})$ zijn voor nader te specificeren L.
Hiermee wordt bedoeld dat goede kodes zeer snel (d.w.z. voor korte rijen) de blok- of innovatie entropie naderen voor steeds grotere L.

Een keuze tussen het universele algoritme van Ziv en Lempel en dat van Rissanen is niet eenvoudig te maken.
Rissanen's algoritme springt, meestal maar niet altijd, efficiënter om met de beschikbare geheugenruimte. Ook verwacht ik dat Rissanen's algoritme sneller komprimeert dan dat van Ziv en Lempel. Het is wel zo dat Rissanen's algoritme, voor de goede werking, enige voorkennis over de bron nodig heeft.

Als laatste wil ik hier alle studenten en medewerkers van de vakgroep bedanken voor hun vriendelijke belangstelling en hulp.
Vooral Frans Willems wil ik bedanken voor de hulp bij het Tunstall algoritme en lemma 4.2, dat van hem afkomstig is.
Prof. J.P.M. Schalkwijk en ir. J.A.M. de Brouwer wil ik verder bedanken voor de mogelijkheid om hier te kunnen afstuderen en hun begeleiding hierin.

Referenties.

Berger, T. [1971] Rate Distortion Theory.
Englewood Cliffs, N.Y., Prentice-Hall, 1971.

Csiszar, I. and J. Körner [1981] Information Theory. Coding Theorems for Discrete Memoryless Systems.
Akadémiai kiado, Budapest, 1981

Davisson, L.D. [1973] "Universal Noiseless Coding."
IEEE Trans. Inform. Theory, vol IT-19, no 6, nov 1973.

- [1975] "Universal Source Coding."
Includes "Advances in Source Coding" by T. Berger,
CISM. no 166, Springer Verlag, 1975.

- [1983] "Minimax Noiseless Universal Coding for Markov Sources."
IEEE Trans. Inform. Theory, vol IT-30, no 2, march 1983.

Gallager, R.G. [1968] Information Theory and Reliable Communication."
New York, N.Y., John Wiley and Sons Inc, 1968.

Kleinen, J.H.M. [1979] Het Genereren van Pseudo-Random Data Sequences m.b.v. het Elias Algoritme en Fibonacci Series.
Afstudeerverslag, vakgroep EI,
T.H. Eindhoven, 1979.

Langdon Jr, G.G. and J. Rissanen [1981-b] "Compression of Black-White Images with Arithmetic Coding." IEEE Trans. Comm., vol COM-29, no 6, june 1981.

Langdon Jr, G.G. [1983] "A Note on the Ziv-Lempel Model for Compressing Individual Sequences." IEEE Trans. Inform. Theory, vol IT-29, no 2, march 1983.

Rissanen, J. and G.G. Langdon Jr [1981-a] "Universal Modeling and Coding." IEEE Trans. Inform. Theory, vol IT-27, no 1, jan 1981.

Rissanen, J. [1983] "A Universal Data Compression System." IEEE Trans. Inform. Theory, vol IT-29, no 5, sept 1983.

Shannon, C.E. [1948] "A Mathematical Theory of Communication." B.S.T.J. 27, Parts I & II, july & oct 1948.

- [1951] "Prediction and Entropy of Printed English" B.S;T.J., jan 1951.

Tunstall, B.P. [1967] Synthesis of Noiseless Compression Codes. Ph. D. Thesis, Georgia Inst. Techn., 1967

Verhoeff, J. [1977] "A New Data Compression Technique." Annals. Syst. Res., no 6, 1977.

Ziv, J. and A. Lempel [1978] "Compression of Individual Sequences via Variable Rate Coding." IEEE Trans. Inform. Theory, vol IT-24, no 5, sept 1978.

Lijst van gebruikte symbolen en notaties.

A	de verzameling van bronletters. het bron alfabet.
B	een boom of de verzameling knopen van een boom.
B_k	een boom met k bladeren.
B^i	de i^e boom in een rij van bomen.
ceil(x)	het kleinste gehele getal $y \geq x$.
D	de verzameling van kodeletters. het kode alfabet.
L_B	de verzameling van bladeren van een boom B.
$\ell(s)$	de lengte van een rij s. d.i. het aantal symbolen in s.
$\bar{\ell}_b$	de gemiddelde segment lengte van een boom B gegeven een bepaalde bron S.
λ	de lege rij.
$\mathbb{N}$	de verzameling der natuurlijke getallen $\{0, 1, ..$
$\mathbb{N}_m$	de verzameling $\{ m, m+1, \ldots \}$
s	een al dan niet eindige rij symbolen.
s_i	het i^e element uit een rij s.
s_i^j	de deelrij van een rij s gevormd door de symbolen s_i, $s_{1+1}, \cdots, s_j$

V_B de verzameling van bron segmenten.

V_C de verzameling van kode woorden.

V^n het n-voudig carthesisch produkt van de verzameling V.

V^∞ de verzameling van alle oneindig lange rijen over de letter verzameling V.

V^* de verzameling van alle eindige lengte rijen over de letter verzameling V.

$\therefore$ het einde van een bewijs.

Bijlage A.

In deze bijlage wordt het bewijs geleverd van de drie stellingen uit hoofdstuk 4.

Lemma 4.2: Als voor een komplete n-boom B, gewogen met S, voor alle k bladeren z en alle inwendige knopen y geldt:

$$q(z) \leq q(y),$$

dan is B een maximale (S,k,n) boom. En omgekeerd.

Bewijs.

Eerste stap. { als $q(z) \leq q(y)$ dan maximaal. }.

Stel voor een komplete n-boom B, gewogen met S, geldt:
voor alle $z \in L_B$ en alle $y \in B-L_B$: $q(z) \leq q(y)$. (A.1)

B heeft een gemiddelde segmentlengte $\bar{\ell}_b$.
Noem de optimale komplete n-boom B^0 met gemiddelde segmentlengte ℓ^0.

Stel dat geldt:

$$\bar{\ell}_b < \ell^0 \qquad (A.2)$$

dit is equivalent met (zie lemma 4.1)

$$\sum_{z \in B-L_B} q(z) < \sum_{z \in B^0-L_{B^0}} q(z) \qquad (A.3)$$

Noem B^1 de verzameling van inwendige knopen die B en B^0 gemeen hebben,

B^2 de verzameling inwendige knopen van B, die geen inwendige knopen van B^0 zijn,

B^3 de verzameling inwendige knopen van B^0, die geen inwendige knopen van B zijn.

B en B^0 hebben evenveel bladeren, dus ook evenveel inwendige knopen, dus B^2 en B^3 bevatten ook evenveel knopen.

Nu gaat (A.3) over in

$$\sum_{z \in B^2} q(z) < \sum_{z \in B^3} q(z) \qquad (A.4)$$

Uit (A.4) volgt dat er zeker een knoop a ε B^2 en een knoop b ε B^3 bestaan met

$$q(a) < q(b) \qquad (A.5)$$

Uit de definitie van B^3 volgt dat ergens op het pad tussen de wortel en knoop b een knoop c ligt, die een inwendige knoop van B^0 is én een blad van B. Zie figuur A-1. Daar c niet verder van de wortel ligt dan b op hetzelfde pad, geldt:

$$q(c) \geq q(b) \qquad (A.6)$$

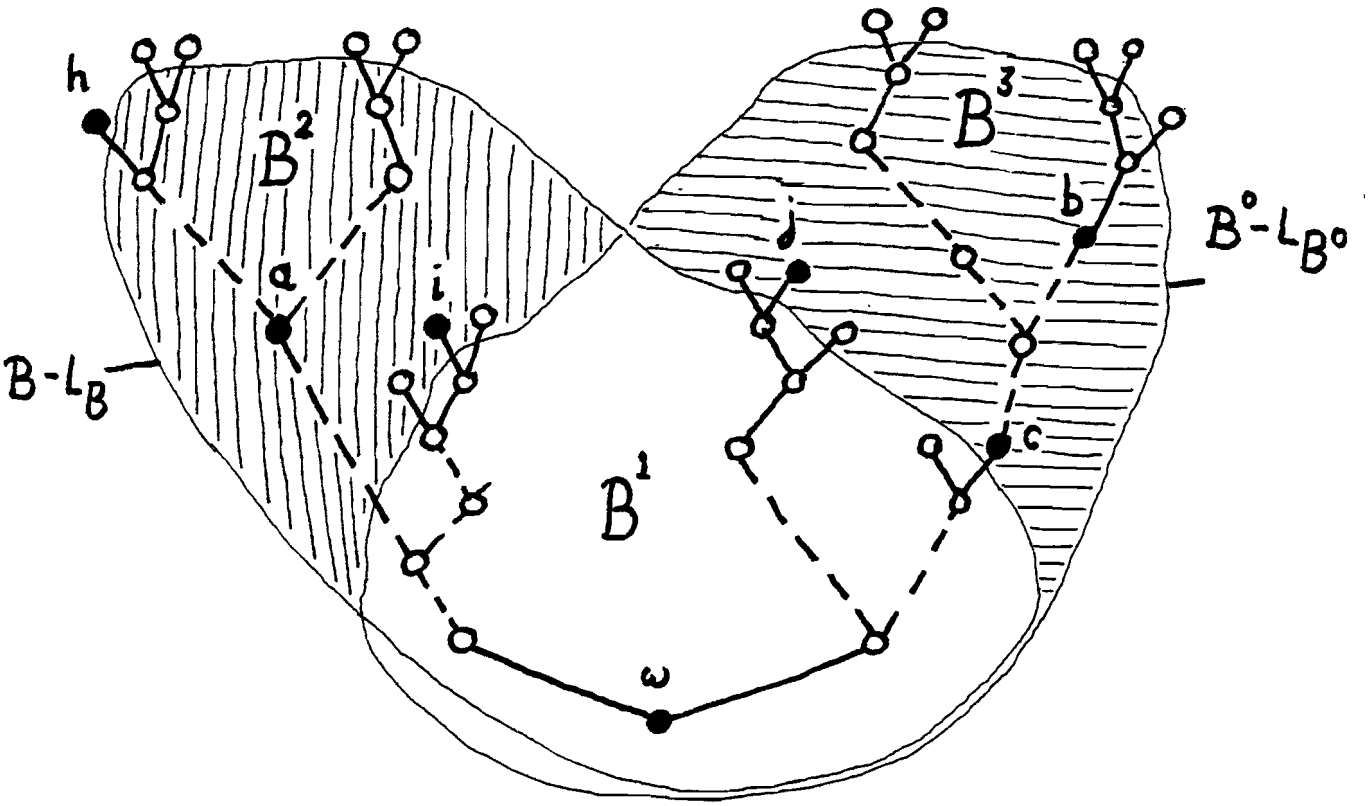
en met (A.5) volgt:

$$q(a) < q(c) \qquad (A.7)$$

Nu geldt daar a een inwendige knoop van B is, en c een blad

van B, dat (A.7) in tegenspraak is met (A.1) en dus dat de veronderstelling (A.2) inkorrekt is.

Hiermee is de eerste stap bewezen.



Figuur A.1.

knoop w is de wortel van B en B^0.
knoop h is een blad van B,
en bestaat niet in B^0.
knoop i is een inwendige knoop van B,
en een blad van B^0.
knoop j is een blad van B,
en een inwendige knoop van B^0.
knoop k bestaat niet in B,
en is een blad van B^0.

knopen a, b en c, zie tekst.

Tweede stap. { b maximaal dan $q(z) \lessdot q(y)$ }.

Stel B is een maximale (S,k,n) boom.
Stel er bestaat een inwendige knoop a en een blad b van B met:

$$q(a) < q(b) \tag{A.8}$$

In de sub-boom met a als wortel bestaat een <u>bladvader</u> c, d.w.z. een inwendige knoop met enkel bladeren als opvolgers. Zie figuur A.2. Ook geldt:

$$q(c) \lessdot q(a) \tag{A.9}$$

Dus met (A.8) en (A.9):

$$q(c) < q(b) \tag{A.10}$$

Laat B^1 uit B ontstaan door de n bladeren onder c weg te halen en deze toe te voegen aan knoop b. (b was een blad en wordt een bladvader). Nu geldt:

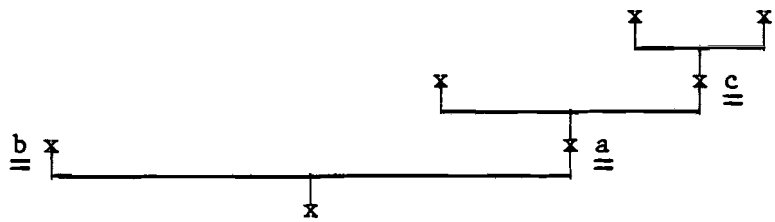
$$\bar{\ell}_{b^1} - \bar{\ell}_b = \sum_{z \in L_{B^1}} q(z) - \sum_{z \in L_B} q(z) = q(b) - q(c) > 0 \tag{A.11}$$

Het positief zijn volgt uit (A.10).

(A.11) is in tegenspraak met het maximaal zijn van B, dus kan (A.8) niet gelden.

Hiermee is ook de tweede stap bewezen.

$\therefore$



Figuur A.2.

Stelling 4.3: Als B een maximale (S,k,n) boom is en B^1 ontstaat uit B door een maximale uitbreiding, dan is B^1 een maximale (S,k+n-1,n) boom.

Bewijs.

Laat a het blad uit B zijn met het grootste gewicht, dus B^1 ontstaat door n bladeren aan a toe te voegen.

B is maximaal, dus (lemma 4.2):

> Voor alle $z \in L_B$ en voor alle $y \in B-L_B$ geldt $q(y) \geq q(z)$. Voor alle bladeren in B^1, behalve de opvolgers van a, en alle inwendige knopen van B^1, behalve a, is de situatie onveranderd.

a is het zwaarste blad van B, oftewel:

$$q(a) \geq q(z), \text{ voor alle } z \in L_B \qquad (A.12)$$

Natuurlijk geldt:

$$q(a) \geq q(x), \text{ voor alle opvolgers } x \text{ van } a. \qquad (A.13)$$

Voor alle $y \in B-L_B$ geldt:

$$q(y) \geq q(a),$$

dus met (A.13)

$$q(y) \geq q(x), \text{ voor alle inwendige knopen } y \text{ van } B \text{ en alle opvolgers } x \text{ van } a \qquad (A.14)$$

Uit (A.12), (A.13) en (A.14) volgt:

$$q(z) \leq q(y), \text{ voor alle } z \in L_{B^1} \text{ en alle } y \in B^1-L_{B^1}.$$

En m.b.v. lemma 4.2 volgt dan dat B^1 een maximale $(S,k+n-1,n)$ boom is.

∴

Voor het bewijs van stelling 4.4 maken we gebruik van de volgende hulpstelling.

Lemma A.1: Stel B is een maximale (S,M,n) boom, p_0 is de kleinste onder de bron kansen $p(a)$, met a een bronletter, dan geldt:

$$\text{voor alle } z \in L_B : q(z) \leq \frac{1}{p_0{}^M} \qquad (A.16)$$

We nemen aan dat p_0 niet nul is.

Bewijs.

Kies a een willekeurig blad uit B_M.
b de vader van a. (d.w.z. a is een opvolger van b)
c een willekeurig blad uit B_M.

Nu geldt: (vader-zoon relatie)

$$q(a) \geq p_0 q(b) \tag{A.17}$$

En (lemma 4.2) $q(b) \geq q(c)$, zodat

$$q(a) \geq p_0 q(c) \tag{A.18}$$

Tevens (zie formule 4.2)

$$\sum_{u \in L_{B_M}} q(u) = 1 \tag{A.19}$$

Uit (A.18) en (A.19) volgt:

$$1 \geq \sum_{u \in L_{B_M}} p_0 q(c) = p_0 M q(c) \tag{A.20}$$

(A.20) geldt voor een willekeurig blad c, dus geldt (A.16).

∴

Stelling 4.4: Noem $(V_B, V_C, f)_M$ de Tunstall/Verhoeff kode met M segmenten voor een geheugenloze bron S.
M is zo gekozen dat C_M gebalanceerd is.
Nu geldt:

$$\lim_{M \to \infty} \frac{n \log M}{\bar{\ell}_b} = H(S) \tag{A.21}$$

Bewijs.

S is de bron (A,P), $(V_B, V_C, f)_M$ een Tunstall/Verhoeff kode hiervoor. p_0 is gegeven zoals in lemma A.1.

Zie hoofdstuk 4.1:

X_M is een stochastische variabele met uitkomstenverzameling V_{B_M}, de verzameling bladeren van een maximale (S,M,n) boom B_M. De kansfunktie van X_M noteren we als P_{B_M}.

Uit lemma 4.1 volgt:

$$H(X_M) = \bar{l}_{b_M} H(S) \tag{A.22}$$

We kunnen $H(X_M)$ alsvolgt insluiten.
Er zijn M bladeren, dus

$$H(X_M) \leq {}^n\log M \tag{A.23}$$

Uit lemma A.1 volgt:

$$H(X_M) \geq {}^n\log p_0 M \tag{A.24}$$

Uit (A.22), (A.23) en (A.24) volgt:

$$\lim_{M\to\infty} \frac{{}^n\log M}{\bar{l}_{b_M}} = \lim_{M\to\infty} \frac{H(X_M)}{\bar{l}_{b_M}} = H(S).$$

Hetgeen te bewijzen was.

Met slechts een kleine modifikatie in lemma A.1 en stelling 4.4 is te bewijzen dat (A.21) ook geldt indien van de n bronletters er slechts een aantal k, $1 \leq k \leq n$, een kans ongelijk aan nul hebben.

Bijlage B.

Lemma 4.3.

Voor een diskrete stochastische variabele $\underline{x}$ met eindig veel waarden en $p(\underline{x}=x)>0$, geldt voor elke positief reëeltallige funktie $f(\underline{x})$:

$$\frac{1}{E\{f(\underline{x})\}} \leq E\left\{\frac{1}{f(\underline{x})}\right\} \tag{B.1}$$

Met gelijkheid slechts dan als $f(\underline{x})$ een konstante funktie is.

Bewijs.

Stel $\underline{x}$ kan waarden aannemen uit $\{x_1, x_2, \ldots, x_n\}$, met $Pr\{\underline{x}=x_i\} = p(x_i)$.

Dus:

$$E\{f(\underline{x})\}E\left\{\frac{1}{f(\underline{x})}\right\} = \sum_{i=1}^{n} p^2(x_i) + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} p(x_i)p(x_j)\frac{f(x_i)}{f(x_j)} \tag{B.2}$$

We weten dat de dubbele som in (B.2) minimaal is als $f(x_i) = c$, voor alle i, dus:

$$E\{f(\underline{x})\}E\left\{\frac{1}{f(\underline{x})}\right\} \geq \sum_{i=1}^{n} p^2(x_i) + \sum_{i=1}^{n} \sum_{j=1}^{i-1} 2p(x_i)p(x_j) \tag{B.3}$$

$$= 1.$$

En daar $f(\underline{x})$ positief is, geldt (B.1).

$\therefore$

Bijlage C.

We willen bewijzen dat, als de segment boom konvergeert naar een eindvorm B^0, bij het basis algoritme van hoofdstuk 4.4, dat dan de totale kompressie faktor konvergeert naar de waarde ρ^0. Hierin is ρ^0 de kompressie die met B^0 op de datarij gehaald wordt.
We spreken steeds over een bepaalde datarij s_1^∞, daar zowel ρ^0 als B^0 van deze rij afhangen.

We noteren:

x : de lengte van het verwerkte stuk van de datarij.

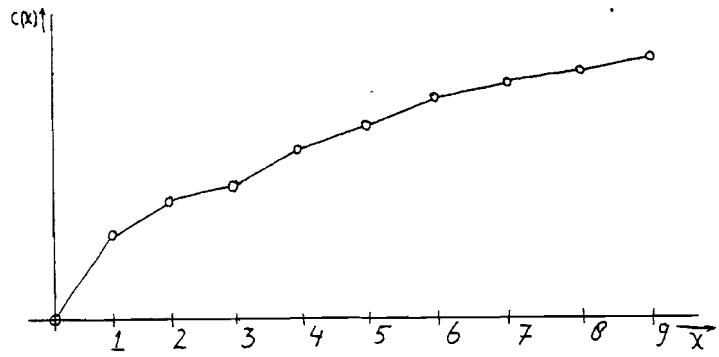
c(x) : de lengte van de koderij behorende bij het verwerkte stuk van de datarij ter lengte x.

c(x), en x zelf, zijn alleen gedefiniëert indien x en c(x) gehele getallen zijn. We zullen c(x) en x uitbreiden naar de reële getallen door c(x) ook voor niet geheeltallige x te definiëren.

Noem $x^1 \triangleq \text{entier}(x)$,

dan $c(x) \triangleq c(x^1) + (x-x^1)(c(x^1+1)-c(x^1))$ (C.1)

dus c(x) wordt stuksgewijs lineair en kontinu voortgezet, zie figuur C.1.



Figuur C.1.

We definiëren een funktie $\gamma(x)$ volgens:

$$\gamma : \mathbb{R} \to \mathbb{R}$$

$$\gamma(x) = \lim_{\delta \to 0} \frac{c(x+\delta) - c(x)}{\delta} \tag{C.2}$$

De interpretatie van $\gamma(x)$ is dat $\gamma(x)$ de momentane reduktie faktor is op het moment dat er x symbolen verwerkt zijn.

Daar we aannemen dat de boom naar een eindvorm konvergeert geldt:

$$\lim_{x \to \infty} \gamma(x) = \rho^0 \tag{C.3}$$

De totale kompressie wordt gegeven door:

$$\rho(x) \triangleq \frac{c(x)}{x} \tag{C.4}$$

We willen bewijzen dat, onder de aanname van (C.3), geldt:

$$\lim_{x \to \infty} \rho(x) = \rho^0 \tag{C.5}$$

Bewijs.

Met (C.2) en (C.4) volgt:

$$\rho(x) = \frac{1}{x} \int_0^x \gamma(z)dz \tag{C.6}$$

En dus

$$\lim_{x \to \infty} \rho(x) = \lim_{x \to \infty} \frac{1}{x} \int_0^x \gamma(z)dz \tag{C.7}$$

Laat nu y(x) een funktie zijn die voldoet aan:

$$0 \leq y(x) < x, \text{ voor } x \text{ groot genoeg.} \tag{C.8a}$$

$$y(x) \to \infty \text{ als } x \to \infty \tag{C.8b}$$

$$\lim_{x\to\infty} \frac{y(x)}{x} = 0 \tag{C.8c}$$

b.v. kies $y(x) = \ln(x)$.

Nu geldt:

$$\lim_{x\to\infty} \frac{\int_{y(x)}^{x} \gamma(z)dz}{x-y(x)} = \rho^0 \tag{C.9}$$

want, we kunnen de integraal insluiten volgens:

$$\gamma^+ \triangleq \max\{\gamma(z): y(x) \leq z \leq x\}, \text{ dus } \lim_{x\to\infty} \gamma^+ = \rho^0.$$

$$\gamma^- \triangleq \min\{\gamma(z): y(x) \leq z \leq x\}, \text{ dus } \lim_{x\to\infty} \gamma^- = \rho^0.$$

$$\text{en } (x-y(x))\gamma^- \leq \int_{y(x)}^{x} \gamma(z)dz \leq (x-y(x))\gamma^+.$$

zodat (C.9) inderdaad geldt.

(C.7) is te herschrijven als

$$\lim \rho(x) = \lim \frac{\int_0^{y(x)} \gamma(z)dz}{x} +$$

$$+ \lim_{x\to\infty} \frac{x-y(x)}{x} \cdot \frac{\int_{y(x)}^{x} \gamma(z)dz}{x-y(x)} \tag{C.10}$$

Met (C.8c) en (C.9) volgt dat

$$\lim_{x\to\infty} \frac{x-y(x)}{x} \cdot \frac{\int_{y(x)}^{x} \gamma(z)dz}{x-y(x)} = \rho^0 \qquad (C.11)$$

De eerste limiet aan de rechterzijde van (C.10) is nul volgens

$$\lim_{x\to\infty} \frac{\int_{0}^{y(x)} \gamma(z)dz}{x} = \lim_{x\to\infty} \frac{\xi(x)y(x)}{x} \qquad (C.12)$$

met $0 \leq \xi(x) < \infty$.

Samen met (C.8c) volgt dan

$$\lim_{x\to\infty} \rho(x) = \rho^0$$

$\therefore$

Bijlage D.

In deze bijlage wordt een konstruktief schema voor het koderen van o.a. bekende eindige orde Markov bronnen geschetst.

Laat S een ergodische 'unifilar' Markov bron zijn met $r<\infty$ toestanden. T is de toestandenverzameling en $p(t_i)$, $t_i \in T$, de limietverdeling van de toestanden.
De entropie van deze bron is gegeven door

$$H_\infty(S) = \sum_{t \in T} p(t) H(S \mid t) \qquad (D.1)$$

We gaan nu het volgende kodeerssyteem voor S opzetten.
We nemen aan dat we T_0, de begintoestand van de bron, kennen. m.b.v. T_0 en de datarij s_1^∞ kan de toestandenrij T_0^∞ bepaald worden. s_1^∞ wordt alsvolgt over r rijen $v_{(i)}$, $1 \leq i \leq r$, verdeeld:

> In de volgorde j = 1, 2, 3, ...
> wordt s_j rechts aan het rijtje $v_{(T_{j-1})}$ geplakt, en alle rijtjes $v_{(i)}$ beginnen als de lege rij λ.

In plaats van één rij met geheugen hebben we nu r geheugenloze rijen, met bijbehorende entropieën $H(S \mid t_i)$.
Stel dat de ontvanger deze rijen plus de waarde van T_0 bezit, dan vormt hij de rij s_1^∞ volgens:

> In de volgorde j = 1, 2, 3, ...
> wordt s_j links afgenomen van het rijtje $v_{(T_{j-1})}$.
> Daar de ontvanger T_0 kent, kan hij s_1 bepalen en daarmee T_1 en dus s_2, etc.

Dit schema heeft last van een mogelijk oneindig lange dekodeervertraging.

De rijtjes $v_{(i)}$ koderen we m.b.v. het Tunstall/Verhoeff algoritme en elk kodewoord voorzien we van een label i, zodat de ontvanger de kodewoorden kan scheiden in de verschillende toestanden. Hiervoor hebben we ceil$({}^{n}\log i)$ symbolen nodig. Dit label neemt voor steeds langere kodewoorden een steeds geringere en in de limiet verwaarloosbare fraktie in beslag. Daar alle nu volgende limieten over eindige verzamelingen van eindige getallen gaan, kunnen we de invloed van het label op de reduktie, in de limiet, al direkt verwaarlozen. We nemen dit niet verder in de beschouwing mee.

Voor elke rij s_1^{∞} van de ergodische bron realiseert dit schema een totale reduktie gelijk aan:

$$\rho_{M|T_0} \triangleq \frac{{}^{n}\log M}{\sum_{j=1}^{r} \Pr\{\text{segment uit } v_{(j)}\}\bar{\ell}_{b_j}} \qquad (D.2)$$

Hierin is M het aantal kodewoorden; we nemen n.l. ook aan, dat alle r segment kodes $(V_B, V_C, f)_j$ evenveel segmenten hebben.
$\bar{\ell}_{b_j}$ is de gemiddelde segmentlengte van kode j. (Bij log M behoort eigenlijk dat restje index bijgeteld te worden).

Daar de bron ergodisch is, geldt:

$$\Pr\{\text{segment uit } v_{(j)}\} = \frac{p(t_j)\frac{1}{\bar{\ell}_{b_j}}}{\sum_{k=1}^{r} p(t_k)\frac{1}{\bar{\ell}_{b_k}}} \qquad (D.3)$$

Er geldt:

$$\rho_M\big|_{T_0} = \frac{\dfrac{{}^n\log M}{\sum_{j=1}^{r} p(t_j)}}{\dfrac{\sum_{j=1}^{r} p(t_j)\dfrac{1}{\bar{\ell}_{b_j}}}{\sum_{k=1}^{r} p(t_j)\dfrac{1}{\bar{\ell}_{b_k}}}} = \sum_{j=1}^{r} p(t_j)\frac{{}^n\log M}{\bar{\ell}_{b_j}} \qquad (D.4)$$

En in de limiet:

$$\rho_{T_0} \triangleq \lim_{M\to\infty} \rho_M\big|_{T_0} = \lim_{M\to\infty} \sum_{j=1}^{r} p(t_j)\frac{{}^n\log M}{\bar{\ell}_{b_j}} =$$

$$= \sum_{j=1}^{r} p(t_j) \lim_{M\to\infty} \frac{{}^n\log M}{\bar{\ell}_{b_j}} = \sum_{j=1}^{r} p(t_j) H\big(S\big|t_j\big)$$

$$= H_\infty(S) \qquad (D.5)$$

Dit volgt uit resp. (D.4), een standaard limiet stelling, stelling 4.4 en (D.1).

In het bovenstaande schema is het cruciaal dat T_0 bekend is bij de kodeerder. De kodeerder kan, zonder dat dit ρ_{T_0} beïnvloedt, T_0 aan de dekodeerder bekend maken. Hiermee is echter niet bewezen dat de reduktiefaktor in de limiet ook niet zou toenemen als we ook de kodeerder T_0 zouden ontzeggen.
Echter, stel dat S een k^e orde Markov bron is, $k < \infty$. Na k bronsymbolen weet de kodeerder in welke toestand de bron zich bevindt. Deze zendt nu de eerste k symbolen over, evt. ongekodeerd, en zowel de kodeerder als de dekodeerder kennen nu T_k. De reduktiefaktor zal hier in de limiet niet onder lijden.