

Section 5

SOURCES ALPHABÉTIQUES**5.1. Introduction**

Dans les sections 3 et 4, nous avons examiné les processus de lecture/écriture des symboles, sous l'angle statistique ou quantique. Reste maintenant à combiner ces symboles pour former (ou lire) d'abord des mots (sources alphabétiques lexicales) puis des textes (sources alphabétiques préfixées). La procédure employée pour réaliser cela est un calcul. C'est pourquoi notre angle d'étude sera dorénavant essentiellement algorithmique.

La théorie algorithmique de l'information précise la notion d'information à partir de celle de calcul effectif : la complexité d'un message, c'est-à-dire l'information qu'il contient, est la taille minimale de sa description algorithmique (développée initialement par Kolmogorov, Solomonoff et Chaitin, cette théorie est présentée par exemple dans Li et Vitányi 1997). Il s'agit d'une théorie de l'information liée à un message particulier, et complète ainsi la théorie statistique de la communication, élaborée par Shannon, qui est une théorie probabiliste liée à un ensemble de messages. Ces deux théories sont complémentaires.

Soit Σ un ensemble dénombrable de symboles χ_i . Chaque élément est identifié bijectivement par un numéro $i \in \mathbb{N}$ et étiqueté par une chaîne binaire $s_i \in \mathbf{B}^*$ telle que " s_i " $\cong \chi_i$. Une source alphabétique $\mathcal{S} \subseteq \mathbf{B}^*$ est un sous-ensemble de \mathbf{B}^* muni d'un ordre partiel décidable noté \preceq . Le terme "décidable" signifie que :

$$\forall s', s'' \in \mathcal{S}, \text{ il existe un algorithme qui décide si } s' \preceq s'' \text{ ou non.}$$

En termes informationnels, on peut dire que : s' est plus concis que s'' mais s'' est plus précis (parce que redondant) que s' ; la description par s'' du symbole χ_i est exhaustive.

Sans entrer pour l'instant dans le détail du formalisme de la théorie, considérons deux sources alphabétiques binaires \mathcal{S} et \mathcal{W} , et un processus d'encodage Φ tel que :

$$s \in \mathcal{S}, w \in \mathcal{W}, s = \Phi(w).$$

Soit l la fonction de volume usuelle définie par le nombre de caractères.

En toute généralité, on impose au processus Φ les propriétés suivantes [d'après Uspensky 1992] :

- (i) Si $s = \Phi(w')$ et $w'' \succeq w'$, alors $s = \Phi(w'')$

Si une chaîne s possède une description abrégée w' dont il existe une version longue w'' , alors cette description longue représente aussi la chaîne.

(ii) Si $s' = \Phi(w)$ et $s'' \leq s'$, alors $s'' = \Phi(w)$

Si une chaîne de forme longue possède une description, alors cette description représente aussi la chaîne sous sa forme courte équivalente.

(iii) Si $s' = \Phi(w)$ et $s'' = \Phi(w)$, alors $\exists \sigma : \sigma = \Phi(w)$ et $s' \leq \sigma$ et $s'' \leq \sigma$.

Si deux chaînes possèdent la même description, il existe une forme commune à ces deux chaînes, qui les inclut.

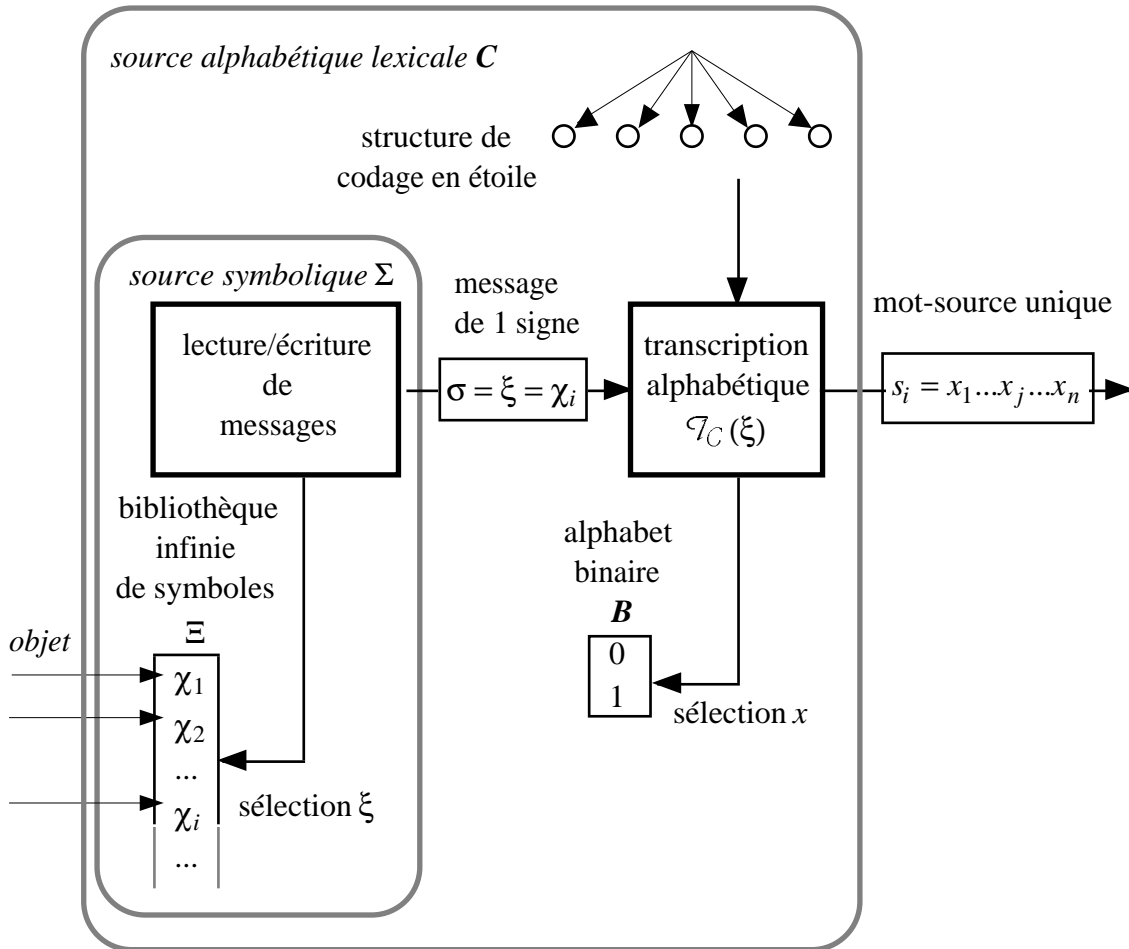
5.1.1. Source alphabétique lexicale

5.1.1.1. Description

Soit une source alphabétique lexicale C constituée : 1°) d'une source symbolique Σ comprenant une bibliothèque infinie (C est "bien formée") dénombrable Ξ de symboles notés $\{\chi_1, \dots, \chi_i, \dots\}$, qui lit ou écrit des messages, de longueur finie m , notés $\sigma = \xi_1 \dots \xi_k \dots \xi_m$, où $\xi_k \in \Xi$. 2°) d'une transcription régulière non déchiffirable \mathcal{T}_C de Ξ dans B^* , où $B = \{\emptyset, 1\}$ est l'alphabet binaire.

La transcription binaire est effectuée selon les règles énoncées au §2.3. Nous appelons "code lexicographique" cette transcription binaire, qui établit une bijection entre l'ensemble des symboles de la bibliothèque et l'ensemble des mots du langage binaire. Puisqu'on ne peut ni ne veut déchiffrer des mots différents au sein d'une même suite de mots-source, on considère une telle suite comme un mot unique, qui est la transcription binaire d'un symbole de la bibliothèque.

On simplifie ici la structure d'une telle source en limitant à $m = 1$ la longueur des messages de la source symbolique contenue dans C (voir schémas 2.2 et 2.3) : un message est la transcription binaire dans l'ordre lexicographique d'un symbole. Il en résulte le schéma simplifié suivant :



• Figure 5.1

Soit $s_i = x_1 \dots x_j \dots x_n$, où $x_j \in \mathbf{B}$, la transcription binaire lexicale du message constitué du symbole χ_i de Ξ lu ou écrit par \mathbf{C} , c'est-à-dire $s_i = \mathcal{T}_{\mathbf{C}}(\chi_i)$. Soit $H(\xi)$ l'entropie de la source symbolique, $l(s_i)$ la longueur d'un mot-source. On rappelle (cf §2.3) qu'un symbole $\chi_i \in \Xi$ est tel que :

- le symbole χ_i est dénoté par l'index $i \in \mathbf{N}^+$.
- le symbole χ_i a une probabilité $p(i)$. Par exemple, $p(i) = 2^{-i}$
- le symbole χ_i est codé dans l'ordre lexicographique par la chaîne s_i de longueur :

$$l(s_i) = \lfloor \log(i+1) \rfloor$$

Convention d'écriture : nous noterons : $\mathbf{C} = \{s_i\}$. En toute rigueur, cette identification est un abus de notation, car une source est un objet qui possède une certaine structure interne (bibliothèque, système de sélection des symboles, système de lecture/écriture des messages). Mais en réduisant la source à une "boîte noire", à telle source \mathbf{C} correspond tel ensemble de

messages binaires (i.e. tel langage dans B^*), et un seul.

5.1.1.2. Propriétés

(i) Une source lexicale est le modèle le plus simple d'ensemble informationnel. C'est une liste qui, bien qu'énumérable dans l'ordre lexicographique, possède un ordre partiel trivial :

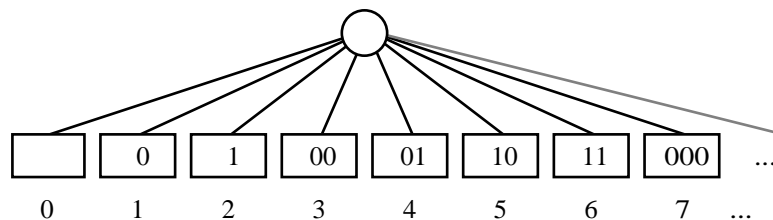
$$s' \leq s'' \Leftrightarrow s' = s''$$

Ce qui signifie que :

- tous les symboles sont équivalents quant à l'information qu'ils portent : cette structure correspond aux entiers naturels, où bien évidemment aucun nombre n'est "plus important" (i.e. : ne contient plus d'information a priori) qu'aucun autre.

- mais ces symboles/nombres entiers ne sont pas transmissibles sous leur forme lexicale, car il est impossible de distinguer deux mots transmis ensembles : dans la chaîne $s's''$ obtenue par la concaténation de s' et de s'' , il n'est pas possible de distinguer s' de s'' (i.e. il n'y a pas de codage déchiffirable).

On peut se représenter un tel ensemble sous la forme d'une structure en étoile, où chaque élément est "sur un pied d'égalité" avec les autres :



• Figure 5.2

C'est la raison pour laquelle l'encodage des symboles dans la figure 5.1 est représenté par cette structure en étoile.

(ii) Pour $s \in C$, on sait que C est muni d'une fonction de volume telle que : $l(s_i) = \lfloor \log(i+1) \rfloor$. Or, sachant que $s = \Phi(w)$, la longueur de l'encodage w de la chaîne s est tel que :

$$l(w) \leq l(s) + 1$$

C'est la longueur de s plus au minimum un bit pour exécuter le programme $\Phi = \text{"PRINT } s\text{"}$ sur une machine spécialement prévue à cet usage.

Inversement, le nombre de programmes w de longueur au plus égale à une certaine quantité n est 2^n . Ces programmes encodent des chaînes s de longueur au plus égale à $n-1$.

Donc le nombre N de chaînes s pour lesquelles $l(w) \leq n$ est :

$$N = \text{Card}(\mathcal{C}) \leq 2^{n-1} < 2^n.$$

Une telle source porte donc en elle-même un renseignement sur sa cardinalité, qui est une information de dénombrement : si un récepteur constate que les mots-code w qu'il reçoit sont de longueur au plus égale à n , il sait que la source comporte au plus 2^{n-1} symboles distincts. La cardinalité est une caractéristique essentielle des sources lexicales. Nous avons vu (§2.4.3) qu'une source lexicale "bien formée" est nécessairement de bibliothèque infinie, ce qui rend la transmission possible si l'on ne transmet *qu'une* chaîne à la fois (et non un texte), car à toute chaîne correspond un symbole. On se trouve ici dans une situation inverse : un encodage étant donné, la connaissance de la longueur maximale des programmes fournit une borne supérieure à la cardinalité de la source qu'ils encodent, mais fait *ipso facto* de celle-ci une source "mal formée". La transmission d'un texte sous la forme d'une chaîne de taille supérieure à n équivaut dans ce cas à une erreur, car à la concaténation de chaînes en une chaîne longue ne correspond aucun symbole de la bibliothèque.

De toutes façons cette connaissance implicite demeure limitée : la quantité $\sum_N 2^{-l(s)}$ est quelconque, peut être supérieure à un, ou infinie. Il n'est donc pas possible de définir une mesure de probabilité à partir de la connaissance de la longueur des chaînes.

Par exemple l'encodage $\mathcal{E} = \{(a,0), (b,10), (c,110), (d,11)\}$, qui n'est pas un code déchiffrable (le message "110110" peut être lu "cda" ou "dac") est tel que :

$$\sum_{i=1}^4 \frac{1}{2^{l(s_i)}} = \frac{9}{8} > 1$$

(v) En conclusion, un tel ensemble informationnel est muni d'une structure en étoile, et composé de symboles discernables et dénombrables. Mais cette possibilité de dénombrement ne suffit pas à le rendre transmissible au sens où il ne permet pas de communiquer un texte constitué de messages successifs, faute de délimiteur ou de procédure de séparation.

5.1.2. Source alphabétique préfixée

5.1.2.1. Description

Soit une source préfixée \mathbf{K} constituée : 1°) d'une source symbolique Σ comprenant une

bibliothèque infinie dénombrable Ξ de symboles notés $\{\chi_1, \dots, \chi_i, \dots\}$, qui lit ou écrit des messages, de longueur finie m , notés $\sigma = \xi_1 \dots \xi_k \dots \xi_m$, où $\xi_k \in \Xi$. 2°) d'une transcription régulière déchiffrable préfixée $\mathcal{T}_{\mathcal{X}}$ de Ξ dans \mathbf{B}^* , où $\mathbf{B} = \{\emptyset, 1\}$ est l'alphabet binaire.

Comment fonctionne une telle source ? On pourrait supposer que chaque symbole χ_i est tout d'abord transcrit sous la forme d'un mot-source s_i binaire dans l'ordre lexicographique, puis encodé sous forme préfixée en un mot-code w_i . Dans cette hypothèse il faudrait voir cette source préfixée comme contenant une source lexicale semblable au modèle qui vient d'être décrit, suivie d'un encodage préfixé. Mais les mots-source n'étant pas séparables, il faudrait alors supposer que l'exécution de l'encodage préfixé est réalisé "mot-à-mot", et que de la concaténation de ces encodages résulte un texte. Tout se passe comme si l'encodage préfixé fonctionnait nécessairement en mode "interprété" et non en mode "compilé".

Cette vue des choses est pour le moins confuse : pourquoi générer du code préfixé donc séparable, à partir d'un ensemble de symboles eux aussi séparables, distinguables par définition, en passant par l'intermédiaire d'un encodage lexicale non déchiffrable ?

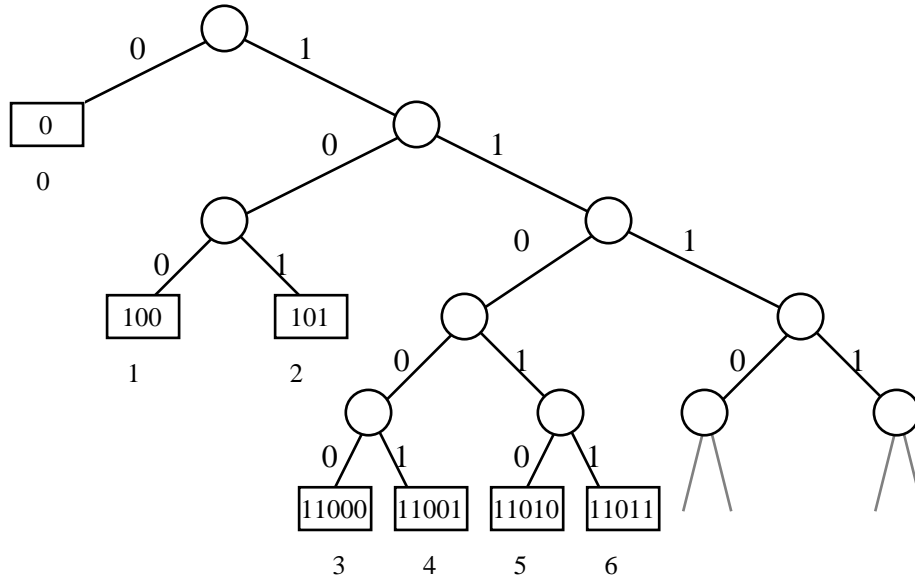
Il semble donc plus naturel et plus simple de considérer une encodage préfixé immédiat des symboles. Par exemple, si ceux-ci sont les entiers naturels, on considère un code préfixe effectué sur \mathbf{N} . Mais bien souvent, les nombres entiers sont implicitement considérés sous leur forme binaire naturelle, donc sous forme lexicale. Nous aurons donc à traiter des codes préfixes sous deux angles différents : 1) indépendamment de tout autre forme de codage : on considère une source préfixée \mathbf{K} qui lit ou écrit des (suites de) mots s_i préfixés codant des symboles χ_i ; 2) relativement à un encodage lexical-lexicographique : on considère une source préfixée \mathbf{K} qui lit ou écrit des mots-code w_i préfixés, codant les mots-source s_i lus ou écrits par une source lexicale \mathbf{C} . Ce dernier cas est un transcodage d'une source lexicale vers une source préfixée (voir §2.4.6). Vu son importance, il fera l'objet d'un traitement particulier. En attendant, dans ce paragraphe, nous nous plaçons selon le premier point de vue.

5.1.2.2. Propriétés

(i) On demande maintenant aux chaînes d'être discernables au sein d'un même ensemble, et distinguables au sein d'un même message. Le code est déchiffrable : des mots communiqués ensembles sont nécessairement délimités. Le modèle d'un tel ensemble informationnel est un arbre binaire complet correspondant à un code préfixe. L'ordre partiel est :

$$s' \leq s'' \Leftrightarrow s' \text{ est un préfixe de } s'' \Leftrightarrow \exists t : [s' t = s'']$$

Convenons d'écrire $\mathbf{K} = \{s_0, s_1, \dots, s_{N-1}\}$ un tel ensemble informationnel descriptible par un arbre tel que celui représenté ci-dessous :



• Figure 5.3 : début de l'arbre binaire du code E1 (cf tableau 5.x)

L'arbre binaire complet représentant un codage préfixe optimal possède N feuilles (i.e. nœuds terminaux – les rectangles dans la figure 5.3), et $N-1$ nœuds internes (cercles dans la figure 5.3). Pour représenter cet encodage, si l'on convient de balayer l'arborescence de haut en bas et de gauche à droite, il faut :

- 1 bit pour spécifier le type de nœud (feuille ou nœud interne), soit $N+N-1$ bits.
- $L_N = \log(N+1)$ bits pour écrire un caractère, en moyenne.

Soit au total une chaîne de $2N - 1 + N \lceil \log N \rceil$ bits pour spécifier l'arbre de codage.

(ii) Théorème 5.1 : inégalité de Kraft

On montre que, pour toute source préfixée, la longueur des mots obéit à la relation suivante, dite *inégalité de Kraft* :

$$\sum_i 2^{-l(s_i)} \leq 1$$

Nous renvoyons le lecteur à l'abondante littérature concernant cette inégalité et sa démonstration. Grossièrement parlant, celle-ci est en quelque sorte une extension de l'inégalité

$$\sum_{i>0} 2^{-i} < 1.$$

(iii) Théorème de codage 5.2

Soit $p(i)$ la probabilité du symbole χ_i auquel correspond le mot-code s_i . Soit $L(s) = \sum_i p(i)l(s_i) = \left\langle \frac{\mathbf{r}}{\mathbf{p}} \middle| \mathbf{l} \right\rangle$ la longueur moyenne des mots. Le *théorème de la voie sans bruit* (1^{er} théorème de Shannon) montre qu'il existe un code irréductible *optimal* pour lequel L est aussi voisine que l'on veut de sa borne inférieure, égale à l'entropie de la source dans le cas d'un alphabet binaire :

$$1 \leq \frac{L(s)}{H(\xi)} \leq 1 + \varepsilon \quad \text{avec} \quad L(s) = \sum_i p(i)l(s_i)$$

Bien que ce théorème soit un théorème d'existence, qui ne dit rien sur la construction effective du code optimal, des méthodes de codage (Fano, Huffman, ...) sont connues, qui permettent d'approcher cette limite. Un exemple en est donné ci-dessous.

Commentaire : dans le cas d'un code irréductible complet, la condition de Kraft est une égalité et la longueur des mots code est telle que :

$$\sum_i 2^{-l(s_i)} = 1$$

Ce qui permet de définir une mesure de probabilité telle que :

$$p(i) = 2^{-l(s_i)}$$

$$\Rightarrow l(s_i) = -\log p(i)$$

$$\Rightarrow p(i).l(s_i) = -p(i).\log p(i)$$

$$\Rightarrow \sum_i p(i)l(s_i) = -\sum_i p(i).\log p(i)$$

$$\Rightarrow L(s) = H(\xi)$$

(iv) En conclusion, un ensemble informationnel \mathcal{S} étant donné,

l'information $\mathcal{F}(\sum_i 2^{-l(s_i)} \leq 1) \Rightarrow$ l'information $\mathcal{F}(\text{Card}(\mathcal{S}) < 2^n)$

La réciproque étant fautive, \mathbf{K} est informationnellement plus riche que \mathbf{C} . Une telle source, en correspondance bijective avec \mathbf{N} , munie d'une structure en arbre, composée d'éléments discernables, dénombrables et déchiffrables, est transmissible au sens où elle permet de communiquer un texte constitué de messages successifs, le délimiteur étant le préfixe ou toute procédure de séparation équivalente. Elle est probabilisable, et énumérable, au sens où nous avons défini ce mot pour la théorie de l'information : on peut non seulement en dénombrer les éléments, mais encore étiqueter ceux-ci selon une certaine arborescence.

Exemple 5.1 : encodage préfixé :

• Soit un canal binaire de bande passante égale à 50 Hz, c'est-à-dire de capacité maximale $C_{max} = 50$ bits/s. Soit une source symbolique binaire S évoluant à la vitesse $V = 100$ pixels/s (chaque pixel étant noir ou blanc). Cette source est par exemple la télécopie d'un texte dactylographié, où il y a beaucoup plus de points blancs que de points noirs. Soient : probabilité d'avoir un pixel blanc = $p(0) = 0,9$; sinon $p(1) = 0,1$.

• L'entropie de S vaut $H_1 = -0,9 \cdot \log(0,9) - 0,1 \cdot \log(0,1) = 0,469$ sh/pixel. Le débit d'information vaut $D = V \cdot H_1 = 46,9$ sh/s $< C_{max}$. D'après le théorème de Shannon, il est possible de coder S pour transmettre les données à travers le canal C .

• On considère la $n^{\text{ème}}$ extension S^n de S : on décrit l'image en blocs de n pixels par des mots de longueur moyenne L_n . La valence de la bibliothèque est $N = 2^n$. La vitesse d'évolution de S^n vaut $v_n = V/n$ blocs/s. Le débit binaire est égal à $D_n = v_n \cdot L_n$ bits/s.

• On cherche un code préfixe K tel que : $D \leq D_n \leq C_{max}$. Soit :

$$V \cdot H_1 \leq V \cdot L_n / n \leq C_{max} \Leftrightarrow 1 \leq L_n / n \cdot H_1 \leq C_{max} / V \cdot H_1$$

La quantité $n \cdot H_1 = H_n$ représente l'entropie moyenne par bloc de S^n . Avec les données numériques précédentes, il vient :

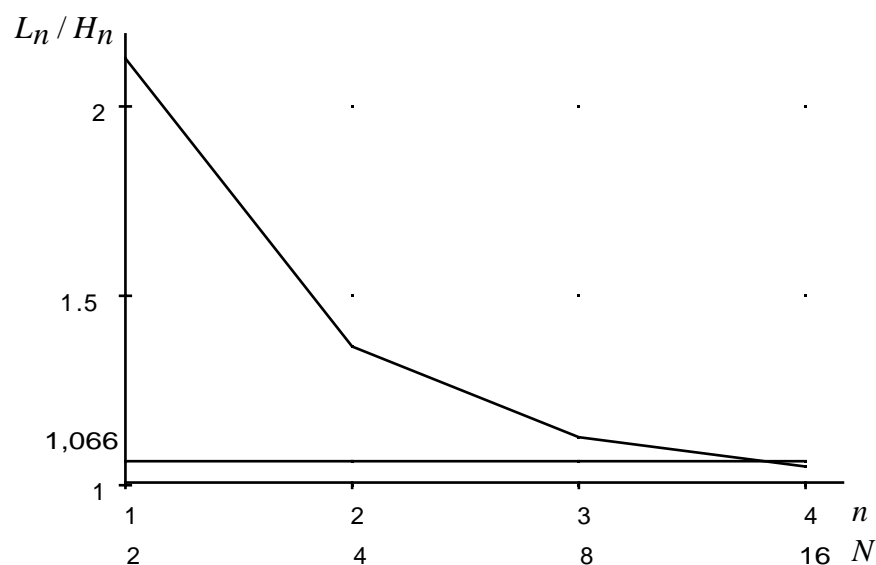
$$1 \leq L_n / H_n \leq 1,066$$

où la quantité L_n / H_n représente l'inverse du rendement du code $\eta = H_n / L_n$. Le tableau 5.1 ci-dessous donne quelques résultats numériques obtenus par la méthode de Huffman. On constate qu'il est nécessaire d'atteindre la 4^{ème} extension de S pour répondre au cahier des charges : la quantité L_n / H_n vaut alors 1,050.

S1	proba	E1	S2	proba	E2	S3	proba	E3	S4	proba	E4
0	0,9	0	00	0,81	0	000	0,729	0	0000	0,6561	0
1	0,1	1	01	0,09	10	001	0,081	100	0001	0,0729	100
			10	0,09	110	010	0,081	101	0010	0,0729	101
			11	0,01	111	011	0,009	11100	0011	0,0081	1111000
						100	0,081	110	0100	0,0729	110
						101	0,009	11101	0101	0,0081	1111001
						110	0,009	11110	0110	0,0081	1111010
						111	0,001	11111	0111	0,0009	11111100
									1000	0,0729	1110
									1001	0,0081	1111011
									1010	0,0081	111110
									1011	0,0009	11111101
									1100	0,0081	1111110
									1101	0,0009	11111110
									1110	0,0009	111111110
									1111	0,0001	111111111
Ln		1			1,290			1,598			1,970
Hn		0,469			0,938			1,407			1,876
Ln/Hn		2,132			1,375			1,136			1,050

• Tableau 5.1 :

- $S^1 = S$; S^2, S^3, S^4 respectivement 2^{ème}, 3^{ème}, 4^{ème} extensions de S .
- E_2, E_3, E_4 : exemples d'encodages préfixés des sources S^2, S^3, S^4 par la méthode de Huffman.
- H_n : entropie de S^n
- L_n : longueur moyenne des mots-code de E_n
- La dernière ligne du tableau indique l'évolution de la quantité L_n/H_n en fonction de n :



- Figure 5.4 : il faut atteindre la 4^{ème} extension de S pour obéir au cahier des charges imposé : $L_n/H_n < 1,066$.

5.1.3. Transcodage d'une source lexicale vers une source préfixée : exemples

Dans l'exemple précédent, le code utilisé dépend de la distribution de probabilité $p(i)$. L'étape suivante consiste à chercher un code qui ne dépende pas d'une distribution particulière. On montre qu'un tel code, dit *universel*, existe, et que ce code est *asymptotiquement optimal* s'il existe une constante c telle que :

$$\lim_{H(\xi) \rightarrow \infty} \frac{L(w)}{H(\xi)} \leq c \quad \text{avec} \quad L(w) = \sum_i p(i)l(w_i)$$

Pour construire un tel code, on considère le codage lexical-lexicographique de la source symbolique Σ . On ajoute (par concaténation) aux chaînes lexicales $s_i = \mathcal{T}_C(\chi_i)$ une information concernant leur longueur, ainsi que d'éventuels caractères de séparation, pour constituer la chaîne préfixée w_i . Ce procédé constitue en réalité un transcodage de la source lexicale C contenant Σ vers une source préfixée K contenant également Σ .

La bibliothèque Ξ de Σ est a priori infinie. Mais dans la pratique, on considèrera dans l'exemple qui suit un sous-ensemble fini de Ξ , de cardinalité $N+1$, et, pour l'étude des propriétés de cette source, on fera tendre N vers des valeurs aussi grandes que l'on veut.

Convention d'écriture : étant donné l'encodage s_i des entiers naturels i selon l'ordre binaire lexicographique (tableau 5.2, 3ème colonne du tableau), pour lequel $l(s_i) = \lfloor \log(i+1) \rfloor$, nous conviendrons, pour alléger l'écriture, et quand il n'y a pas d'ambiguïté, de noter par s à la fois la chaîne s_i et le nombre entier i qu'elle code. Alors la grandeur « $l(s)$ » est un nombre entier qui est «le nombre de bits de la chaîne s_i représentant l'entier i dans le code lexicographique» et l'expression $f(s)$ désigne une «fonction de l'entier i codé par la chaîne s_i ».

Exemple : considérons les codes auto-délimités $w = E(s)$ suivants, avec $s \in \{0,1,2,\dots, N\}$ et $N \geq 1$.

$$w = E_0(s) = "1^s 0" ; E_0(N) = "1^N"$$

$$w = E_1(s) = "1^{l(s)} 0 s"$$

$$w = E_2(s) = "1^{l(l(s))} 0 l(s) s"$$

$$w = E^*(s) = "l^{(\lambda)}(s) 0 l^{(\lambda-1)}(s) 0 \dots 0 l^{(2)}(s) 0 l^{(1)}(s) 0 s 1"$$

	<i>i</i>	<i>s</i>	<i>w</i>		
dec.	binaire	lexique	E1	E2	E*
0	0		0	0	01
1	1	0	100	1000	0001
2	10	1	101	1001	0011
3	11	00	11000	10100	10001
4	100	01	11001	10101	10011
5	101	10	11010	10110	10101
6	110	11	11011	10111	10111
7	111	000	1110000	11000000	10000001
8	1000	001	1110001	11000001	10000011
9	1001	010	1110010	11000010	100000101
10	1010	011	1110011	11000011	100000111
11	1011	100	1110100	11000100	100001001
12	1100	101	1110101	11000101	100001011
13	1101	110	1110110	11000110	100001101
14	1110	111	1110111	11000111	100001111
15	1111	0000	111100000	110010000	1001000001
16	10000	0001	111100001	110010001	1001000011
17	10001	0010	111100010	110010010	1001000101
18	10010	0011	111100011	110010011	1001000111
19	10011	0100	111100100	110010100	1001001001
20	10100	0101	111100101	110010101	1001001011
.../...					
124	1111100	111101	1111110111101	11011111101	101101111011
125	1111101	111110	1111110111110	11011111110	101101111101
126	1111110	111111	1111110111111	11011111111	101101111111
127	1111111	0000000	111111100000000	11100000000000	1000000000000001
128	10000000	0000001	111111100000001	11100000000001	1000000000000011
129	10000001	0000010	111111100000010	11100000000010	1000000000000101
130	10000010	0000011	111111100000011	11100000000011	1000000000000111
etc...					

• Tableau 5.2

Les fonctions de volume de ces codes sont respectivement :

$$E_0 : l(w) = s + 1 \text{ pour } s \neq N ; l(w) = s \text{ si } s = N.$$

$$E_1 : l(w) = 2l(s) + 1 = 2\lfloor \log(s+1) \rfloor + 1$$

$$E_2 : l(w) = 2l(l(s)) + l(s) + 1 = 2\lfloor \log \log(\lfloor \log(s+1) \rfloor + 1) \rfloor + \lfloor \log(s+1) \rfloor + 1$$

$$E^* : l(w) = 1 + \sum_{k=1}^{\lambda} (l^{(k)}(s) + 1) = l(s) + Q_{\lambda}(l(s)) + \lambda + 1 + O(\lambda)$$

La fonction quasilogarithme itéré est définie par :

$$\text{qlog}^{(1)} x = \text{qlog } x \quad ; \quad \text{qlog}^{(k+1)} x = \text{qlog}(\text{qlog}^{(k)} x)$$

La fonction itération du quasilogarithme, notée $\log^* x$, est le nombre d'itérations effectuées sur la fonction quasilogarithme tant que $x > 1$, soit $\text{qlog } x > 0$:

$$\lambda = \log^* x = \max \{ k \geq 1 : \text{qlog}^{(k)} x > 0 \}$$

C'est une fonction à croissance très lente :

$$x = 2 \quad \lambda = \log^* 2 = 1$$

$$x = 4 \quad \lambda = \log^* 4 = 2$$

$$x = 16 \quad \lambda = \log^* 16 = 3$$

$$x = 65536 \quad \lambda = \log^* 65536 = 4$$

$$x = 2^{65536} \quad \lambda = \log^* 2^{65536} = 5$$

Dans la pratique, sachant que l'on estime le nombre d'atomes dans l'univers à 10^{80} environ, on ne rencontrera pas des valeurs de λ supérieures à 5 !

La fonction somme des logarithmes itérés est définie par :

$$Q_\lambda(x) = \text{qlog}^{(1)} x + \text{qlog}^{(2)} x + \dots + \text{qlog}^{(\lambda)} x$$

Dans E^* , l'itération est définie sur des quantités très proches de ces valeurs :

$$l(s) = \lfloor \log(s+1) \rfloor$$

$$l^{(\lambda)}(s) = l(l^{(\lambda-1)}(s)) ; \dots ; l^{(2)}(s) = l(l(s)) ; l^{(1)}(s) = l(s).$$

$\lambda = \log^* l(s)$ est le nombre d'étapes nécessaires pour obtenir $l^{(\lambda)}(s) \cong "1"$

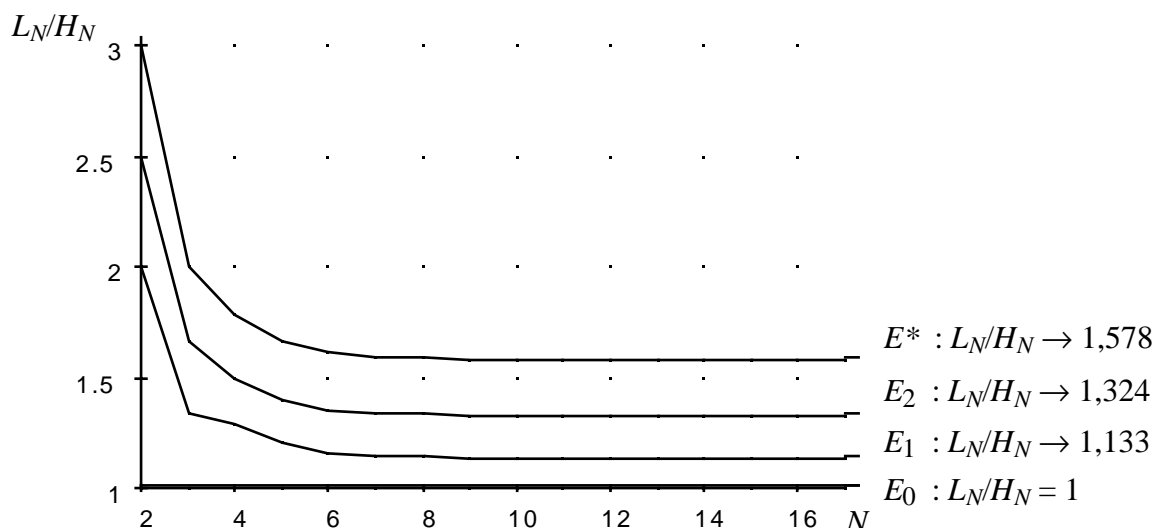
Ces codes sont-ils asymptotiquement optimaux, et si oui, quelles sont leurs "performances" ? On considèrera deux hypothèses : (a) cas d'une probabilité en 2^{-s} , (b) cas d'une distribution de probabilité uniforme.

(a) Soit : $p(s) = 2^{-(s+1)}$ si $s < N$, $p(s) = 2^{-N}$ si $s = N$. Il vient :

$$\frac{L_N}{H_N} = \frac{\sum_s p(s) l(w)}{-\sum_s p(s) \log p(s)} = \frac{\sum_{s=0}^{N-1} 2^{-(s+1)} l(w_s) + 2^{-N} l(w_N)}{\sum_{s=0}^{N-1} 2^{-(s+1)} \cdot (s+1) + 2^{-N} \cdot N}$$

avec : $\sum_{i=0}^N p(s) = 1$ et $\lim_{N \rightarrow \infty} H_N = 2$

Pour le code E_0 on a exactement $L_N = H_N \forall N$: c'est le cas optimal. Pour les autres codes, lorsque la valence N de la source augmente indéfiniment, le rapport L_N/H_N tend rapidement vers une limite incompressible supérieure à 1, tandis que l'entropie H_N reste finie (elle tend vers deux). Par conséquent les codes E_1 , E_2 et E^* ne sont pas optimaux (figure 5 .x).



• Figure 5.5

(b) Soit : $p(s) = 1/(N+1)$, distribution uniforme, donc $H_N = \log N$:

$$\frac{L_N}{H_N} = \frac{\sum_s p(s) l(w)}{-\sum_s p(s) \log p(s)} = \frac{\sum_{s=0}^N \frac{1}{N} l(w_s)}{\log N} = \frac{\sum_{s=0}^N l(w_s)}{N \log N}$$

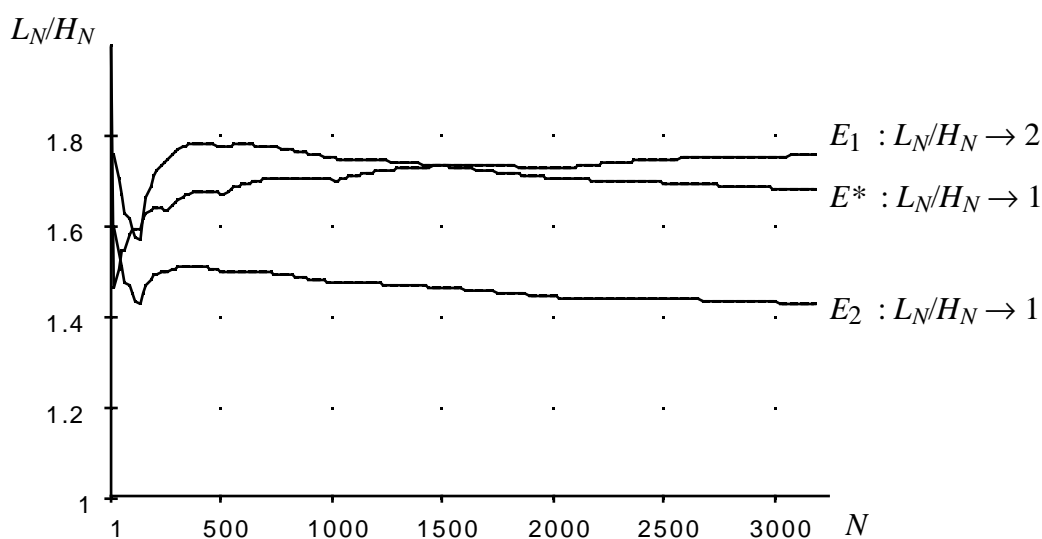
avec : $\sum_{s=0}^N p(s) = 1$ et $\lim_{N \rightarrow \infty} H_N = \infty$

Lorsque N augmente, le code E_0 n'est évidemment pas optimal, puisque la longueur des mots est égale à $N+1$, alors que l'entropie est en $\log N$: le rapport L_N/H_N tend vers l'infini.

Pour le code E_1 , ce rapport augmente également avec N , et tend vers 2 par valeur inférieure

(car $\lim_{x \rightarrow \infty} \int \frac{2 \log x + 1}{x \cdot \log x} dx = 2$). E_1 n'est donc pas optimal.

En revanche, pour les codes E_2 et E^* , ce rapport tend vers 1 par valeur supérieure (car $\log x + \log \log x < 2 \log x$). Selon la définition donnée plus haut, ces codes sont donc asymptotiquement optimaux. On notera toutefois qu'il est nécessaire de considérer de grandes valeurs de N : il faut atteindre $N \approx 1500$ pour que E^* soit plus intéressant que E_1 , et $N > 2^{2048}$ pour que E^* soit plus intéressant que E_2 (figure 5.6)



• Figure 5.6

On montre que le code E^* est proche de la limite théorique, au-delà de laquelle on ne peut plus satisfaire l'inégalité de Kraft : à quelques variantes minimales près, il n'est pas possible d'aller plus loin dans la compression de l'information, pour N assez grand.

5.2. Complexités algorithmiques des sources alphabétiques

5.2.1. Présentation générale et classification

(i) Soient S et W deux sources alphabétiques lexicales de même alphabet B (mais de bibliothèques généralement différentes). Nous précisons la définition 2.27 dans le sens suivant :

Définition 5.1 :

Un mode de description ou code est un ensemble récursivement énumérable $\mathcal{E} \subseteq \mathbf{W} \times \mathbf{S}$, et $\langle w, s \rangle \in \mathcal{E}$ est un encodage de s .

Définition 5.2 : la complexité de s selon \mathcal{E} est : $C_{\mathcal{E}}(s) = \min\{l(w) : \langle w, s \rangle \in \mathcal{E}\}$

Par convention, $C_{\mathcal{E}}(s) = \infty$ si une telle description n'existe pas : dans ce cas, il n'y a aucun symbole de \mathbf{W} correspondant au symbole de \mathbf{S} encodé dans la chaîne binaire s .

Pour utiliser cet encodage (et restituer s connaissant w), on utilise un modèle de calcul qui, à partir d'une entrée $w \in \mathbf{B}^*$, permet de calculer une sortie s , à l'aide d'une fonction récursive partielle Φ_k réalisée par une machine de Turing T_k .

La complexité d'une chaîne $s \in \mathbf{S}$ est alors la longueur minimale du programme w appliqué au calculateur Φ_k qui produit s :

$$C_{\Phi_k}(s) = \min\{l(w) : \Phi_k(w) = s\}$$

Si on connaît Φ_k , un problème est de connaître le plus petit programme w qui produit s , et donc $C_{\Phi_k}(s) = \min(l(w))$. Mais on peut imaginer qu'il est aussi possible de produire s à partir de w à l'aide d'un autre programme Φ_j réalisé par une machine T_j , ce qui conduit à un second problème, qui est de déterminer si un programme est équivalent à un programme donné. On sait que le premier est récursivement énumérable, alors que le second est non calculable. Cette remarque a un double corollaire, à la fois positif et négatif :

Corollaire "négatif" : la complexité d'une chaîne n'est pas calculable dans le cas général.

Corollaire "positif" : la complexité d'une chaîne ne dépend pas de la machine utilisée, à une constante près.

La démonstration de cette dernière affirmation met en jeu les deux théorèmes suivants, sachant que l'on peut énumérer les machines de Turing T_1, T_2, \dots sous la forme Φ_1, Φ_2, \dots telles que Φ_k est la fonction calculée par T_k pour tout k .

(ii) Théorème d'existence 5.3 : il existe une fonction récursive partielle universelle Φ_0 calculée par une machine de Turing universelle U telle que :

$$\Phi_0(\mathcal{E}(kw)) = \Phi_k(w)$$

où $\mathcal{E}(kw)$ désigne un encodage auto-délimité de la $k^{\text{ième}}$ machine de Turing T_k exécutant le programme w . Par exemple $\mathcal{E}(kw) \cong "1^{l(k)}0kw"$: le programme complet comprend le code exécutable k de la machine T_k suivi de la donnée w à laquelle il s'applique ; l'encodage auto-

délimité permet de séparer facilement la partie exécutable des données.

(iii) Théorème d'invariance 5.4 : la fonction Φ_0 est telle que :

$$C_{\Phi_0}(s) \leq C_{\Phi_k}(s) + c_{\Phi_k}$$

La complexité d'une chaîne ne dépend pas de la machine utilisée, à une constante additive près. Une machine universelle U étant donnée, chaque machine de Turing particulière est assimilable à un programme exécutable. Alors la complexité d'un objet est la longueur du plus petit programme qui peut produire cet objet. Dans la suite de ce texte, on notera simplement $C(s) = \min\{l(w) : \Phi(w) = s\}$.

Exemple : divers auteurs ont proposé des exemples simples de machines universelles codées sous la forme d'une constante binaire minimale. Citons notamment R. Penrose 1989 qui propose une description binaire de MTU longue de 5495 bits. Chaitin 1987 utilise une version simplifiée de LISP pour construire en une dizaine de pages de listing un interpréteur exécutant la quinzaine d'instructions d'une machine à registres. Cet auteur en propose une version encore simplifiée dans [Chaitin, 1995/1996]. Li et Vitányi utilisent le lambda-calcul et les opérateurs primitifs S et K définis par les règles de réécriture suivantes :

$$Sxyz = xz(yz)$$

$$Kxy = x$$

pour construire une machine décrite par une chaîne de 425 bits. La notion de machine universelle n'est donc pas une notion purement abstraite : il est possible d'en construire des exemples concrets.

Dans le cas d'une chaîne aléatoire, la longueur du programme qui la code est de l'ordre de grandeur de celle de la chaîne elle-même. C'est le programme $w = \text{"PRINT } s\text{"}$, dont le nombre de bits est au moins égal à $l(s)+1$, avec $w = \text{"}1s\text{"}$. D'où :

(iv) Théorème de la borne supérieure 5.5 : il existe une constante c telle que :

$$C(s) \leq l(s) + c$$

La plupart des chaînes étant incompressibles, $C(s) \approx \log s$ pour presque toutes les chaînes. Précisons ce point :

(v) Posons $l(s) = n$. Étant donné n , il y a 2^n chaînes de longueur n , mais seulement

$\sum_{k=0}^{n-1} 2^k = 2^n - 1$ programmes w décrivant les chaînes s et de longueur plus courte qu'elles,

c'est-à-dire tels que $l(w) \leq n - 1$. Il y a donc au moins $2^n - (2^n - 1) = 1$ chaîne telle que $l(w) \geq$

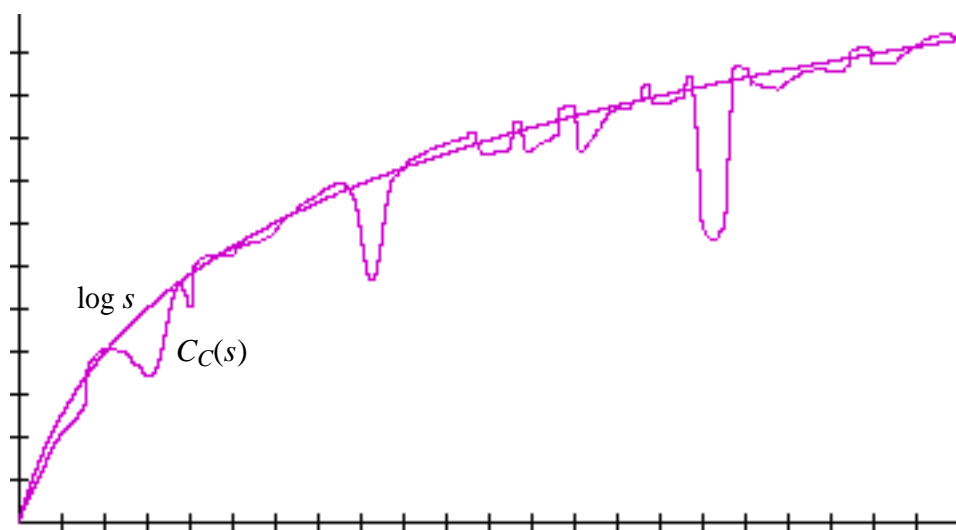
n , donc telle que $C(s) \geq l(s)$. De même il y a au moins $2^n - 2^{n-1} + 1 > \frac{1}{2}2^n$ chaînes telles que

$l(w) \geq n - 1$; au moins $\frac{3}{4}2^n$ chaînes telles que $l(w) \geq n - 2$; etc.

Définition 5.3 : une chaîne est c -incompressible si $C(s) \geq l(s) - c$. En généralisant le raisonnement précédent, il y a au moins une fraction égale à $1 - 2^{-c}$ de l'ensemble des 2^n chaînes de longueur n qui sont c -incompressibles. D'où :

Théorème d'incompressibilité 5.6 : il y a au moins $2^n (1 - 2^{-c}) + 1$ chaînes de complexité $C(s) \geq n - c$ (\Leftrightarrow il y a au plus $1/2^c$ chaîne de longueur n ayant une complexité $< n - c$, c'est-à-dire qui peut être compressée de plus de c caractères)

Des points (iv) et (v) on déduit que, presque partout, la complexité des chaînes s est de l'ordre de grandeur de leur longueur, soit $\log s$.



- Figure 5.7 : représentation figurée du graphe de $C_C(s)$, soulignant l'allure irrégulière de cette fonction, proche cependant de $\log s$ dans la plupart des cas. Les minima de $C_C(s)$ correspondent à des chaînes de moindre complexité. Il peut s'agir par exemple de chaînes formées de la répétition d'une sous-chaîne donnée (un "motif"), ou encore de chaînes apparemment aléatoires, mais dont le codage sous forme d'un programme de calcul est relativement court (programme de calcul des décimales de π par exemple).

Nous venons d'examiner le cas de la complexité $C(s)$, qui traite de l'encodage d'une source lexicale vers une autre source lexicale. Nous ne détaillerons pas plus avant les définitions et théorèmes qui fondent de façon similaire les autres sortes d'encodage. Nous allons seulement en donner la classification.

Selon la structure de \mathbf{W} et de \mathbf{S} , autrement dit selon que l'on code un élément-source s de \mathbf{C} ou \mathbf{K} par un élément-mot w de \mathbf{C} ou \mathbf{K} , il y a quatre possibilités pour réaliser un encodage dans $\mathbf{W} \times \mathbf{S}$ [Uspensky, 1992]. Toutes les mesures qui suivent évaluent la complexité d'une chaîne s par la longueur de sa description w . Elles diffèrent seulement par la présence ou l'absence de préfixage sur s ou w . Les propriétés de ces complexités seront détaillées dans les paragraphes qui suivent.

1- *Complexité simple*, qui correspond à la première (au sens historique) théorie algorithmique de l'information : $\mathcal{E} \subseteq \mathbf{C} \times \mathbf{C}$.

$$C_{\mathbf{C}}(s) = C_{\Phi}(s) = \min\{l(w) : \Phi(w) = s\}$$

Notation : au lieu de $C(s)$, nous écrirons dorénavant $C_{\mathbf{C}}(s)$.

- On n'impose ni à w ni à s d'être autodélimités. Cela revient à coder un objet s quelconque pris dans un ensemble isomorphe à \mathbf{N} par une chaîne w appartenant à un langage de même structure : la description d'un nombre est un nombre.

- Ordre de grandeur : $C_{\mathbf{C}}(s) \approx l(s) + O(1) \quad \forall s$, car le plus petit programme produisant s , supposée aléatoire, est `PRINT s`. Un tel programme comprend au moins $l(s)+1$ bits pour être exécutable sur une machine qui serait spécialement conçue pour cette fonction.

- $C_{\mathbf{C}}$ est notée K_A dans [Kolmogorov 1965], BB ou KS dans [Uspensky 1992], KS dans [Dubacq 1998], C dans [Li et Vitányi 1997].

2- *Complexité uniforme* : $\mathcal{E} \subseteq \mathbf{C} \times \mathbf{K}$:

$$C_{\mathbf{K}}(s) = C_{\Phi}(s; l(s)) = \min\{l(w) : \Phi(l(s), w) = s_{1:k} \quad \forall k \leq l(s)\}$$

où $s_{1:k}$ représente les k premiers bits de s .

- On encode un objet autodélimité $s \in \mathbf{K}$ (donc de longueur connue) par un entier $w \in \mathbf{C}$: la description d'un mot est un nombre (i.e. le processus de décodage calcule un mot à partir d'un nombre).

- Ordre de grandeur : $C_{\mathbf{K}}(s) \approx l(s) + O(1) \quad \forall s$.

- Cette complexité est appelée "*uniform complexity*" dans [Loveland 1969], "*decision complexity*" dans [Zvonkin et Levin, 1970]. C_K est notée $K_A(y^n; n)$ dans [Loveland 1969], KR dans [Zvonkin et Levin 1970], BT ou KD dans [Uspensky 1992], $C(x; l(x))$ dans [Li et Vitányi 1997].

3-Complexité préfixée : $\mathcal{E} \subseteq \mathbf{K} \times \mathbf{C}$:

$$K_C(s) = C_\Phi(s) = \min\{l(w) : \Phi(w) = s\} \text{ avec } w \text{ autodélimité.}$$

- On code un objet s quelconque pris dans un ensemble similaire à \mathbf{N} ($s \in \mathbf{C}$) à l'aide d'une chaîne autodélimitée $w \in \mathbf{K}$. La description d'un nombre est un mot (i.e. le processus de décodage calcule un nombre à partir d'un mot).

- Ordre de grandeur (cf § 3-1, exemple 3.2) : une estimation assez précise est : $K_C(s) \approx l(s) + Q_\lambda(l(s)) + O(1) \forall s$, car le plus petit programme qui produit s contient s et le codage de la longueur de s .

- K_C est notée KP dans [Levin 1976], TB ou KP dans [Uspensky 1992], KP dans [Dubacq 1998], K dans [Li et Vitányi 1997].

4-Complexité monotone : $\mathcal{E} \subseteq \mathbf{K} \times \mathbf{K}$:

$$K_K(s) = \min\{l(w) : \Phi(w) = s\} \text{ avec } s \text{ et } w \text{ autodélimités.}$$

- Borne supérieure : $K_K(s) \leq l(s) + O(1) (\forall s)$

- On code un objet s autodélimité par une chaîne w autodélimitée. La description d'un mot est un mot (i.e. le processus de décodage convertit un mot en un autre mot).

- Ordre de grandeur : $K_K(s) \approx l(s) + O(1) \forall s$, car le programme qui recopie une chaîne autodélimitée est lui-même autodélimité (par exemple "PRINT. s .stop" fonctionne comme "PRINT. s .STOP").

- K_K est notée TT dans [Uspensky 1992].

Les relations entre ces différentes entropies sont résumées dans le tableau 5.3 et la figure 5.8 ci-dessous (d'après Uspensky 1992, Li et Vitányi 1997).

On montre qu'il est possible d'ordonner les différentes entropies entre elles :

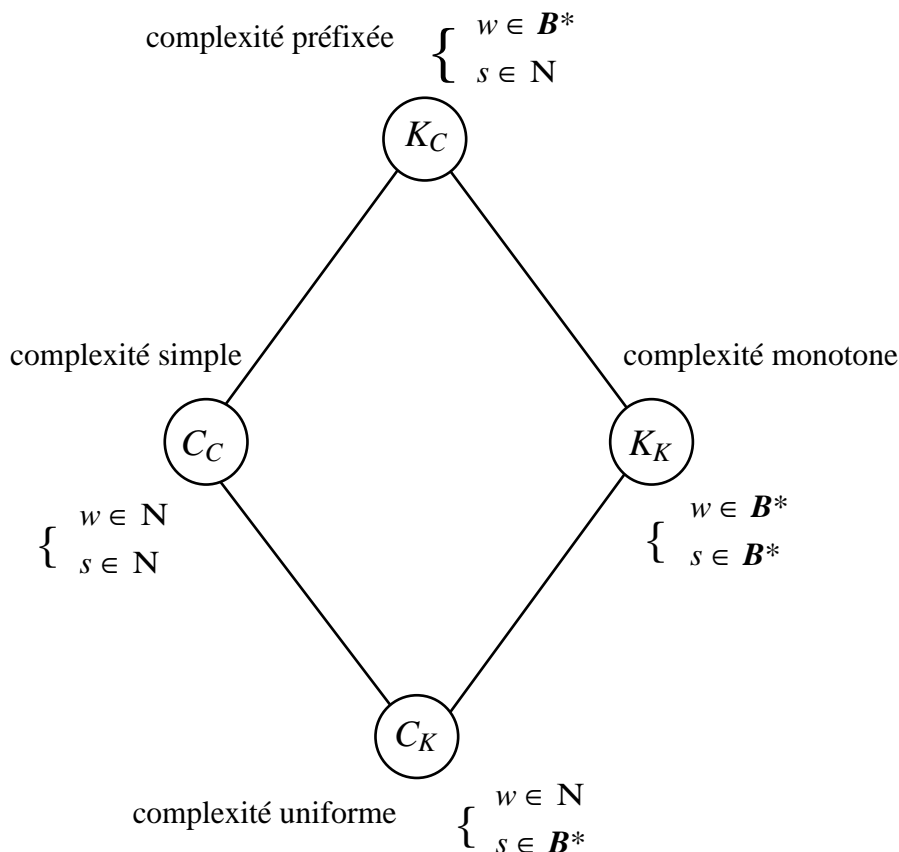
$$K_C > C_C ; K_C > K_K ; C_C > C_K ; K_K > C_K ; \text{ et, bien sûr, } K_C > C_K$$

Mais on ne peut comparer C_C et K_K , car on ne peut affirmer : ni $C_C > K_K$; ni $K_K > C_C$ [Uspensky 1992].

La figure 5.6 illustre cette classification. Dans les paragraphes qui suivent, nous allons reprendre point par point l'ensemble de ces définitions.

	$C_C(s)$	$K_K(s)$	$K_C(s)$	$C_K(s)$
$l(s)$	$O(1)$	$O(1)$	$Q_\lambda(l(s))$	$O(1)$
$C_C(s)$	-	$Q_\lambda(l(s))$	$Q_\lambda(l(s))$	< 0
$K_K(s)$	$\log l(s)$	-	$Q_\lambda(l(s))$	< 0
$K_C(s)$	< 0	< 0	-	< 0
$C_K(s)$	$Q_\lambda(l(s))$	$Q_\lambda(l(s))$	$\log l(s) + Q_\lambda(l(s))$	-

- **Tableau 5.3** : résultats des soustractions (à une constante $O(1)$ près) : ordre de grandeur d'une complexité marquée en colonne moins ordre de grandeur d'une complexité marquée en ligne (ces ordres de grandeur indiquent comment évolue le rapport $L_k/H_k \geq 1$ lorsque celui-ci tend vers 1 par valeurs supérieures, pour $H \rightarrow \infty$).



• Figure 5.8 : classification schématique des quatre complexités.

5.2.2. Complexité simple d'une source lexicale

5.2.2.1. Source unique

Un certain nombre de conséquences dérivent de la définition $C_C(s) = \min\{l(w) : \Phi(w) = s\} = l(s) + O(1) \approx \log s$.

(i) Monotonie

Théorème 5.7 : $C_C(s)$ n'est pas monotone sur les préfixes : la complexité d'une partie peut être supérieure à la complexité du tout !

□ [Li, Vitányi, 1997, p111] Soient deux entiers m et n tels que $m < n$ et $C_C(m) > C_C(n)$: on peut choisir par exemple pour m un nombre aléatoire tel que $C_C(m) \approx \log m$, et choisir $n = 2^{l(m)}$ tel que $C_C(n) \approx \log \log m$. Soient deux chaînes r et s telles que $r \equiv "1^n"$

et $s \equiv "1^m"$. Alors s est un préfixe de r , mais $C_C(s) > C_C(r)$.



Exemple

Le programme v produit la chaîne $r \equiv "1^{425}"$ constituée de quatre-cent-vingt-cinq "1" successifs :

REPETER [425 , "1 "]

En code lexicographique, 425 s'écrit "10101010" :

REPETER ["10101010" , "1 "]

On peut donc comprimer ce programme en écrivant (avec $4 \equiv "01"$) :

REPETER [REPETER ["01" , "10"] , "1 "]

Si on suppose que l'instruction REPETER $[x, y]$ comporte c bits, ce dernier programme occupe $2c + 2 + 2 + 1 = 2c + 5$ bits.

Le programme w produit la chaîne $s \equiv "1^{406}"$:

REPETER [406 , "1 "]

soit (avec $406 \equiv "10010111"$) :

REPETER ["10010111" , "1 "]

La chaîne "10010111" n'est pas compressible de façon immédiate. La longueur totale du programme w reste donc à $c + 8 + 1 = c + 9$ bits.

Supposons que dans la machine utilisée on ait $c = 3$. Alors $l(v) = 11$ et $l(w) = 12$. Donc $l(v) < l(w)$ alors que s est un préfixe de r .

(ii) Non-calculabilité

Théorème 5.8 : $C_C(s)$ n'est pas une fonction récursive partielle.

Étant donné un programme w calculant s (par exemple "PRINT s "), on pourrait énumérer tous les programmes jusqu'à w , en cherchant à savoir s'il existe un programme w' calculant aussi s et tel que $l(w') < l(w)$. Mais cette opération n'est pas calculable, car il n'existe pas de procédure effective pour comparer deux programmes.

(iii) Énumérabilité

Théorème 5.9 : il existe une fonction totale récursive $\Phi(t, s)$ convergeant simplement vers $C_C(s)$ quand t tend vers l'infini, telle que :

$$\forall t \in \mathbf{N}, \forall s \in \mathbf{B}^*, \Phi(t, s) \geq \Phi(t + 1, s) \geq C_C(s)$$

□ [Li, Vitányi, 1997, p121] Pour tout programme w , si l'exécution de U sur t étapes produit s et s'arrête, définissons $\Phi(s,t)$ comme :

$\Phi(s,t)$ = longueur du plus petit programme w qui produit s si un tel programme existe,

$$\Phi(s,t) = l(s) + c \text{ sinon.}$$

$\Phi(s,t)$ est récursive, totale, monotone décroissante, car $\forall s, t' > t \Rightarrow \Phi(s,t') \leq \Phi(s,t)$. Et il est clair que : $C_C(s) = \lim_{t \rightarrow \infty} \Phi(s,t)$.

Donc $C_C(s)$, bien que non calculable, est approximable par le dessus, c'est-à-dire par majoration.

▣

Définitions 5.4 :

Nous résumons ici quelques définitions importantes, données sous plusieurs formes dont l'équivalence fait l'objet de théorèmes dont nous ne traiterons pas :

- une fonction $f: \mathbb{N} \rightarrow \mathbb{R}$ est *énumérable par le bas* ou *énumérable* :

- ssi l'ensemble $\{(x,q) : x \in \mathbb{N}, q \in \mathbb{Q}, q \leq f(x)\}$ est récursivement énumérable

- ssi il existe une fonction $\Phi(x,q)$ récursive monotone croissante en q telle que : $f(x) = \lim_{q \rightarrow \infty} \Phi(x,q)$.

- une fonction $f: \mathbb{N} \rightarrow \mathbb{R}$ est *énumérable par le dessus* ou *co-énumérable* :

- ssi $-f$ est énumérable
- ssi l'ensemble $\{(x,q) : x \in \mathbb{N}, q \in \mathbb{Q}, f(x) \leq q\}$ est récursivement énumérable

- ssi il existe une fonction $\Phi(x,q)$ récursive monotone décroissante en q telle que : $f(x) = \lim_{q \rightarrow \infty} \Phi(x,q)$.

Théorème 5.10 : $C_C(s)$ est co-énumérable

Cela découle immédiatement du résultat précédent (th. 5.9), car $C_C(s) = \lim_{t \rightarrow \infty} \Phi(s,t)$ avec $\Phi(x,t)$ récursive monotone décroissante. Si l'on se donne une borne supérieure b arbitraire, l'ensemble défini par $\{(s,b) : s, b \in \mathbb{N}, C_C(s) \leq b\}$ est récursivement énumérable (en exécutant U , le résultat finira toujours par apparaître) : la fonction $C_C(s)$, bien que non récursive (i.e. non

calculable, car on ne sait pas *quand* le résultat apparaîtra), est co-énumérable.

(iv) Non-invariance par permutation

Théorème 5.11 : la complexité simple n'est pas invariante selon une permutation récursive [Schnorr, 1977].

5.2.2.2. Sources conjointes

(i) Complexité conditionnelle

Définition 5.5 : soit $E(s,r)$ l'encodage autodélimité de la concaténation de deux chaînes s et r . Par exemple, comme vu plus haut, soit $E(s,r) \equiv "1^{l(s)} 0sr"$. Appliquons le programme $E(s,r)$ sur la machine universelle notée Φ . On définit la complexité conditionnelle par :

$$C_C(s|r) = \min\{l(w) : \Phi(E(s,r)) = s\}$$

d'où l'on déduit immédiatement que :

$$C_C(s) = C_C(s|\epsilon)$$

$$C_C(s|s) = 0$$

La complexité $C_C(s|r)$ a les mêmes propriétés que la complexité $C_C(s)$, et les mêmes défauts... $C_C(s|r)$ n'est pas calculable, et n'est pas monotone sur les préfixes : la complexité de la chaîne st peut être *inférieure* à la complexité de la chaîne s !

Théorème 5.12

$$C_C(s|r) \leq C_C(s) + O(1)$$

□ [Li, Vitányi, 1997, p101] Considérons la machine T calculant la fonction Φ qui, pour tout r, t , produit s à partir de l'entrée $\langle t,r \rangle$, ssi la machine de référence U calcule s pour l'entrée $\langle t,\epsilon \rangle$. Alors $C_{C\Phi}(s|r) = C_C(s)$. Par le théorème 5.2, il existe une constante c telle que $C_C(s|r) \leq C_{C\Phi}(s|r) + c = C_C(s) + c$.

□

Une information supplémentaire sur s ne peut pas augmenter la complexité de s , mais seulement la diminuer.

(ii) Cas particulier

La quantité $C_C(s|l(s))$, ou complexité conditionnelle sur la longueur, représente la longueur du plus petit programme qui produit s , la longueur de s étant connue, ce qui permet d'économiser environ $\log l(s)$ bits :

$$C_C(s|l(s)) \leq C_C(s) + O(1)$$

Cette quantité n'est pas monotone sur les préfixes, notamment dans les cas simples, où la connaissance de la longueur de s peut servir à produire facilement s .

Exemple

Soit une chaîne d'un type particulier appelé n -chaîne : $x \cong "n0^{n-l(n)}"$. La longueur d'une telle chaîne est par définition $l(x) = n$. Donc si l'on connaît n il existe une machine qui, pour tout n , produit x en un nombre fini d'étapes, soit : $C_C(x|l(x)) = c$. Mais n est quelconque, et a dans le cas général une complexité $C_C(n) \approx \log n$, de même que $C_C(l(n)) \approx \log \log n$. Donc l'ordre de grandeur de $C_C(n|l(n)) \approx \log n - \log \log n$ est supérieur à c pour n assez grand. La complexité $C_C(x|l(x))$ n'est donc pas monotone sur le préfixe n de la chaîne x .

Le raisonnement est le même si on remplit une chaîne x' non pas avec des 0, mais par la répétition de " n ", équivalent à un motif répété $n / l(n)$ fois (le dernier motif étant éventuellement tronqué si x ne contient pas un nombre entier de fois cette chaîne " n ") : à une constante près, la complexité de x' est la même que x .

Comme le montre Loveland [Cette importante propriété fut signalée à Loveland par Martin-Löf lors d'une communication privée. Voir Loveland, 1969a], la complexité conditionnelle sur la longueur $C_C(s|l(s))$ est invariante par permutation récursive, à une constante près.

- Soit $\#$ la chaîne obtenue en inversant l'ordre des bits de s (par exemple $s = 0011 \Rightarrow \# = 1100$). On a bien sûr $l(\#) = l(s)$. Alors, pour tout s , il existe une constante c telle que $|C_C(s|l(s)) - C_C(\#|l(s))| < c$. Cette propriété est vraie pour toute permutation récursive de s . Aussi la complexité simple conditionnelle sachant la longueur de s peut être considérée comme étant bornée par la complexité conditionnelle de sa forme la moins complexe plus la complexité de toute permutation effectuée sur cette forme simple. Comme nous l'évoquions en introduction (dans l'exemple du §1.3), l'ordre interne des caractères composant une chaîne *de longueur connue* est en lui-même une information.



(iii) Sous-additivitéDéfinition 5.6

Soit $C_C(r,s) = C_C(\langle r,s \rangle)$. $C_C(r,s)$ est la longueur du plus petit programme tel que U calcule r et s , et les imprime séparément.

Théorème 5.13 : la complexité simple n'est pas sous-additive, car :

$$C_C(r,s) \leq C_C(r) + C_C(s) + 2 \log (\min (C_C(r), C_C(s)))$$

- [Li, Vitányi, 1997, p101] Le principe de la démonstration est très simple : supposons que, sur une machine U , nous ayons $r = \Phi(v)$ et $s = \Phi(w)$, v (resp. w) étant le plus petit programme qui produit r (resp. s). Pour que la machine U divise la chaîne vw pour exécuter séparément chaque programme, il faut coder vw par exemple par " $1^{l(v)}0l(v)vw$ " ou par " $1^{l(w)}0l(w)wv$ ".

Corollaire

L'idée qui consiste à adjoindre au code de la chaîne rs l'information nécessaire pour exécuter séparément le code calculant r et celui calculant s , peut être interprétée un peu différemment en disant que l'information contenue dans rs est égale à l'information contenue dans r , plus l'information contenue dans s sachant r , plus un terme n'excédant pas $\log l(rs)$, soit $\log C_C(r,s)$:

$$C_C(r,s) = C_C(r) + C_C(s|r) + O(\log C_C(r,s))$$

(iv) Information mutuelle

Définition 5.7 : l'information mutuelle algorithmique propre à s contenue dans r est définie par :

$$C_C(r:s) = C_C(s) - C_C(s|r)$$

Comme $C_C(s|r) \leq C_C(s)$ à une constante additive près indépendante de r et s , il vient :

$$C_C(r:s) \geq 0$$

(v) Symétrie

A une constante additive près indépendante de r et s , on a la propriété :

$$C_C(r,s) = C_C(s,r)$$

car il revient au même de coder r puis s pour produire r puis s , ou inversement. Donc :

$$\begin{aligned}
& C_C(r) + C_C(s|r) + O(\log C_C(r,s)) = C_C(s) + C_C(r|s) \\
\Rightarrow & C_C(r) - C_C(r|s) + O(\log C_C(r,s)) = C_C(s) - C_C(s|r) \\
\Rightarrow & |C_C(r:s) - C_C(s:r)| = O(\log C_C(r,s))
\end{aligned}$$

L'information mutuelle algorithmique simple n'est pas symétrique. Ce résultat contre-intuitif, qui vient s'ajouter à la non sous-additivité de C_C , est souvent considéré comme un "défaut" réhivitoire de la théorie. En fait il n'en est rien : si un tel "reproche" était à faire, ce n'est pas à la théorie qu'il faudrait l'adresser, mais à son objet, les sources lexicales. Ce serait en réalité une critique tout aussi dépourvue de sens. Ces propriétés qui paraissent étranges découlent d'une même caractéristique de ces sources, qui sont régulières mais non déchiffrables. Cette limitation inhérente à ce type de source fait qu'une "chaîne" rs est en fait un même et unique objet, insécable, une totalité indivise dès lors que deux objets supposés préalablement distincts ont été associés par concaténation. Or cette opération est implicite si l'on veut comparer deux objets entre eux, ou un objet par rapport à l'autre : sur une machine de Turing, il faut bien écrire chaque chaîne sur le ruban de données, l'une après l'autre. Ce qui, dans le cas de la complexité simple, suppose l'existence d'une information contextuelle explicite supplémentaire concernant la longueur de chaque chaîne, en $O(\log C_C(r,s))$ ou $O(\log C_C(r \text{ ou } s))$ selon le cas. De là découlent les "défauts" de la complexité simple, non-additivité et non-symétrie.

5.2.3. Complexité uniforme d'une source préfixée

5.2.3.1. Source unique

On se donne une chaîne préfixée s , qui contient une information sur sa propre longueur. On code cette chaîne par un programme qui, lui, n'est pas autodélimité.

Exemple préliminaire

Reprenons le cas d'une n -chaîne "à motif" examinée dans l'exemple du §5222-i. Par exemple, $x = "101101101101"$, $l(x) = n = "101" = 12$, $r = n / l(n) = 4$ (le motif est répété quatre fois). Pour imprimer une telle chaîne x , deux méthodes sont possibles :

-soit : calculer le nombre r de répétitions du motif, puis imprimer r fois ce motif. En simplifiant quelque peu (on néglige les problèmes de troncature), il vient :

Lire $n \cong "n"$

Calculer $r = n / l(n)$

$k = 0$

Tant que $k < r$, **faire :**

Imprimer " n "

$k \rightarrow k+1$

-soit : copier "bit à bit" la chaîne " x ". On ne tient pas compte du fait que " x " est la répétition d'un motif ; on utilise seulement le fait que sa longueur est connue, par exemple en lisant son préfixe.

Lire $n \cong "n"$

$i = 0$

Tant que $i < n$, **faire :**

Imprimer " x_i "

$i \rightarrow i+1$

La première de ces méthodes a été utilisée dans l'exemple §5222-i, en appliquant la définition de la complexité simple conditionnelle sachant la longueur à une n -chaîne, entraînant la non-monotonie de cette complexité. La deuxième méthode consiste à s'astreindre à écrire la chaîne bit après bit. On ne peut donc plus "jouer" sur telle ou telle propriété de celle-ci – par exemple sur le fait qu'elle est constituée par la répétition d'un motif – pour comprimer le programme qui la recopie. Cela aboutit à la définition de la complexité uniforme suivante :

Définition 5.8

La complexité uniforme est définie par :

$$C_K(s) = C_{\Phi}(s; l(s)) = \min\{l(w) : \Phi(l(s), w) = s_{1:k} \quad \forall k \leq l(s)\}$$

où $s_{1:k}$ représente les k premiers bits de s .

$$C_K(s) = \infty$$

si un tel programme n'existe pas.

Loveland donne la définition équivalente suivante [Loveland, 1969] :

$$C_K(s) = C_{\Phi}(s; l(s)) = \min l(w)$$

où $w \in \{w \mid \Phi(l(s), w) = s_{1:k}, s_{1:k} \text{ p } s, \forall k \leq l(s)\}$

Pour tout k , la machine Φ produit le préfixe $s_{1:k}$ de s .

La complexité uniforme conserve les principales propriétés de la complexité simple :

-elle est définie sur toute machine universelle, à une constante additive près.

$-C_K(s) \leq l(s) + O(1)$ et $C_K(r|s) \leq C_K(r) + O(1)$

$-C_K(s)$ n'est pas récursive partielle.

-l'allure générale de $C_K(s)$, dont l'ordre de grandeur est $\log s$, est concave, sauf dans les zones où les chaînes sont de faible complexité.

$-C_K(s)$ est co-énumérable.

Mais, par rapport à la complexité simple, $C_K(s)$ possède une propriété supplémentaire essentielle :

Théorème 5.14 :

La complexité uniforme est monotone sur les préfixes.

□ Démonstration évidente, compte tenu de sa définition. $l(s)$ sert seulement à déterminer la longueur de la chaîne, mais ne peut être utilisée pour comprimer le programme.

Corollairement, la complexité uniforme est encadrée par :

$$C_C(s|l(s)) + O(1) \leq C_K(s) \leq C_C(s) + O(1)$$

□

-En revanche, C_K n'est pas invariante selon une permutation récursive [Loveland, 1969a, Schnorr, 1977]. On a vu au §5.2.2.2-ii qu'une forme particulière de complexité, la complexité simple sachant la longueur $C_C(s|l(s))$, permet de borner les variations de la quantité d'information nécessaire pour décrire une chaîne lorsqu'une permutation récursive est appliquée à celle-ci. Cela n'est plus vrai de $C_K(s)$, dont la définition est pourtant voisine de $C_C(s|l(s))$: il n'y a pas de constante qui limite la différence en complexité uniforme entre s et π . Comme nous l'évoquions en introduction (dans l'exemple du §1.3), il faut tenir compte de l'ordre interne des caractères composant une chaîne, et c'est bien ce que mesure la complexité uniforme, qui mesure la longueur du plus petit programme calculant uniformément un chaîne ordonnée.

□ [d'après Loveland, 1969a] Soit une chaîne s^n , dont le dernier bit est un 1, telle que $l(s^n) = n$ et $C_K(s^n) \approx n$. Soit $r^m = 011011100\dots\pi^n$ la concaténation de la succession des nombres binaires jusqu'à π^n . On a donc $s^n < \pi^m$. Alors $C_K(r^m) = c$, une constante indépendante de m , alors que $C_K(\pi^m) \geq C_K(s^n) \approx n \approx \log m - \log \log m$. Sachant que la complexité uniforme est une

fonction monotone non décroissante sur la longueur, on en déduit qu'il est impossible qu'une constante limite la différence entre les complexités uniformes de s^n et s^{n+1} pour tout s .



5.2.3.2. Sources conjointes

Si l'on connaît la longueur d'un des programmes codant deux chaînes, la complexité simple conditionnelle sachant la longueur est sous-additive [Li, Vitányi, 1997, p102] :

$$C_C(r,s|C_C(r)) \leq C_C(r) + C_C(s) + O(1)$$

Dans le cas de la complexité uniforme, et bien qu'ils ne soient pas eux-mêmes autodélimités, des programmes qui calculent r et s encodent des quantités préfixées. L'arrêt d'un programme est assuré lorsque celui-ci arrive à la fin de la chaîne qu'il calcule, il n'y a donc pas besoin d'ajouter une information pour séparer deux programmes successifs, ce qui rend la quantité $C_K(r,s)$ sous-additive :

$$C_K(r,s) \leq C_K(r) + C_K(s) + O(1)$$

En outre, il est clair qu'il est équivalent de calculer r puis s , ou l'inverse. Donc :

$$C_K(r,s) = C_K(s,r) + O(1)$$

5.2.4. Complexité préfixée d'une source lexicale

5.2.4.1. Source unique

C'est la problématique inverse du paragraphe précédent : la chaîne n'est pas autodélimitée, contrairement à son encodage.

(i) Nous renvoyons à la littérature (par exemple [Li, Vitányi, 1997, Dubacq 1998] pour la description des propriétés de base de la complexité préfixée. En résumé :

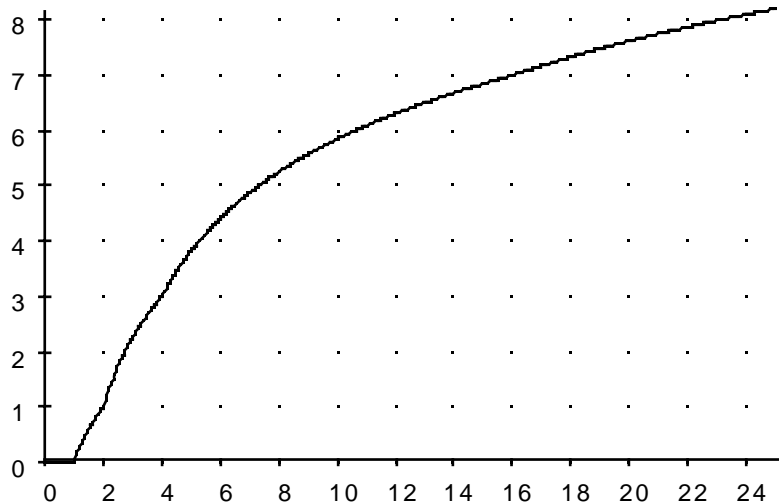
- $K_C(s)$ est définie sur toute machine universelle, à une constante additive près.

- $K_C(s)$ n'est pas récursive partielle.

- $K_C(s) \leq l(s) + Q_\lambda(l(s)) + O(1) \leq l(s) + 2 \log l(s) + O(1)$

-l'allure générale de $K_C(s)$, dont l'ordre de grandeur est $Q_\lambda(s)$, est concave, sauf dans les zones où les chaînes sont de faible complexité.

- $K_C(s)$ est co-énumérable.



• Figure 5.9 : la fonction $Q_\lambda(s)$

Une conséquence majeure de la définition de $K_C(s)$ est que l'ensemble des chaînes est probabilisable :

(ii) On peut définir une mesure universelle $\pi(s)$ sur \mathbb{N} à partir d'une machine de Turing universelle préfixée U : on considère tous les programmes préfixés w qui produisent la chaîne s et s'arrêtent :

Définition 5.9 :

$$\pi(s) = \sum_{\Phi(w)=s} 2^{-l(w)}$$

Cette mesure est une probabilité dans le cas des codes instantanés complets, pour lesquels la condition de Kraft est une égalité.

Soit w^* le programme préfixé minimal qui produit s . La complexité algorithmique est définie par :

$$K_C(s) = l(w^*)$$

On a vu que, statistiquement, la longueur moyenne des mots d'un code préfixe optimal tend vers l'entropie de la source, soit en moyenne : $L(w) \approx H(s)$. Pour un symbole s_i codé par un mot w_i , cela correspond à : $l(w_i) \approx -\log p(s_i)$. Dans le meilleur des cas, l'égalité est réalisée, et la quantité $-\log p(s_i)$ représente la longueur minimale d'un mot codant l'état s_i .

Nous admettons le théorème de codage algorithmique de l'information, prouvé par exemple dans [Li et Vitányi, 1997], qui énonce une relation équivalente :

Théorème 5.15 :

$$l(w^*) = K_C(s) = -\log \pi(s) + O(1)$$

Ces résultats s'étendent aux probabilités conditionnelles (voir ci-dessous) :

$$K_C(s|r) = -\log \pi(s|r) + O(1)$$

Les théories de l'information appliquées respectivement aux processus communicants et à la description algorithmique autodélimitée des objets apparaissent donc comme ayant un schéma très similaire : pour une source dénombrable composée de symboles distinguables, il existe un code irréductible déchiffable où les mots-code sont d'autant plus longs que les symboles qu'ils codent sont plus improbables. Dans les deux théories la notion d'information est fondamentalement la même et repose sur le contenu d'improbabilité, de nouveauté, d'inattendu des objets.

(iii) Théorème 5.16 : La chaîne $\langle s, K_C(s) \rangle$ contient la même information que w^* , programme minimal qui calcule s :

$$K_C(s) = K_C(\langle s, K_C(s) \rangle) + O(1) = K_C(s, K_C(s)) + O(1)$$

- (\leq) Si on connaît s et la longueur du plus petit programme qui calcule s , il est possible d'expliciter celui-ci en exécutant les $2^{K_C(s)}$ programmes autodélimités de longueur $K_C(s)$. Comme la plupart de ces programmes bouclent ou ne fonctionnent pas, il faut les tester pas à pas (1^{ère} passe : on exécute le 1^{er} pas du programme 1 ; 2^{ème} passe : pas 2 du programme 1, puis pas 1 du programme 2 ; 3^{ème} passe : pas 3 du programme 1, pas 2 du programme 2, pas 1 du programme 3, ..., $k^{\text{ème}}$ passe : pas k du programme 1, ..., pas i du programme $k-i+1$, ..., pas 1 du programme k , etc). On compare le résultat que chaque programme produit, s'il en produit, avec la chaîne s . Lorsque celle-ci apparaît, on connaît alors le plus petit programme qui produit s .
- (\geq) La réciproque est triviale : à partir de w^* , on accède facilement à $\langle s, K_C(s) \rangle$ par $s = \Phi(w^*)$ et $K_C(s) = l(w^*)$.



La définition de l'information que l'on obtient ainsi est plus subtile qu'il n'y paraît. Quel est exactement l'information portée par le symbole numéroté par la chaîne s ? Est-ce $K_C(s)$? En partie seulement : la fonction $K_C(s)$ étant non récursive, la connaissance de s ne suffit pas pour calculer $K_C(s)$. De s , on ne peut donc déduire son contenu d'information $K_C(s)$, ce qui est assez paradoxal ! En fait, il faut associer s et $K_C(s)$ pour exprimer explicitement cette information, sous l'une des formes équivalentes $\langle s, K_C(s) \rangle$ ou w^* . Examinons ce point.

Définition 5.10 : Gács [Gács, 1974] définit "la complexité de la complexité" par la quantité $K_C(K_C(s)|s)$.

De l'évaluation $K_C(s) \leq l(s) + 2 \log l(s) + O(1)$ on déduit :

$$K_C(K_C(s)|s) \leq \log l(s) + 2 \log \log l(s) + O(1)$$

Théorème 5.17 : on montre [Gács, 1974 ; Li et Vitányi 1997] que :

$$\exists s : K_C(K_C(s)|s) \geq \log l(s) - \log \log l(s) + O(1)$$

Normalement, la complexité conditionnelle devrait être telle que, par définition, $K_C(s, r) = K_C(s) + K_C(r|s)$. En appliquant cette définition avec $r = K_C(s)$ à la complexité de la complexité, on devrait avoir : $K_C(s, K_C(s)) - K_C(s) = K_C(K_C(s)|s)$, soit encore, d'après la relation $K_C(s) = K_C(s, K_C(s)) + O(1)$ précédente : $K_C(K_C(s)|s) = O(1)$. Or cela est faux, puisque certaines chaînes sont telles que $K_C(K_C(s)|s) \geq \log l(s) - \log \log l(s) + O(1)$.

On en déduit que, pour certaines chaînes s , la connaissance de s n'est pas d'un grand secours pour calculer $K_C(s)$. C'est le cas notamment de chaînes qui ne sont pas aléatoires, mais dont on ne connaît pas le programme de calcul le plus court. Il existe par conséquent une information non négligeable, incompressible, contenue dans $K_C(s)$, en plus du contenu s lui-même.

Remarque :

On montre de même que :

$$\exists s : C_C(C_C(s)|s) \geq \log l(s) - \log \log l(s) + O(1)$$

$$C_C(s, C_C(s)) = C_C(s) + O(1)$$

Ces relations conduisent pour la complexité C_C à un résultat identique. Mais la non-additivité de celle-ci est vraie pour presque toutes les chaînes, aléatoires notamment : elle ressort de la définition même de C_C . Alors que la non-additivité de K_C est due à certaines chaînes dont la complexité $K_C(K_C(s)|s)$ peut être très grande.

On verra plus loin les importantes conséquences de cette discussion, dont la conclusion, en première approximation, est la suivante : l'écriture préfixée d'une chaîne s , c'est-à-dire la mise en forme de cette chaîne effectuée de telle façon que son encodage soit distinguable, donc communicable, porte une double information, explicite concernant le contenu d'information propre à la chaîne, implicite concernant le moyen de l'obtenir (i.e. par un décodage lors de la

lecture). Il est clair que les conséquences de cette observation dans le domaines des processus communicants sont fondamentales.

5.2.4.2. Sources conjointes

(i) Théorème 5.18

$$K_C(s) \leq K_C(s,r) + O(1)$$

De la concaténation des encodages de s et de r (i.e. des programmes de calcul, qui sont autodélimités) on déduit facilement l'encodage de s puis on calcule s lui-même. La constante $O(1)$ est la longueur de l'instruction de séparation. La complexité K_C est donc monotone sur les préfixes.

(ii) Théorème 5.19 (commutativité) :

$$K_C(s,r) = K_C(r,s)$$

L'encodage du couple (s,r) est obtenu après concaténation des encodages préfixés de s et r . Il est facile de construire une nouvelle chaîne en inversant leur ordre, à l'aide d'une instruction de longueur $O(1)$.

(iii) Théorème 5.20 (sous-additivité) :

$$K_C(s,r) \leq K_C(s) + K_C(r) + O(1)$$

- Les programmes v et w codant respectivement r et s étant délimités peuvent être concaténés sans confusion. Toutefois, la relation $K_C(s,r) = K_C(s) + K_C(r|s)$ étant fautive dans ce cas où $r = K_C(s)$, on déduit que la complexité K_C n'est pas additive, mais seulement sous-additive.

▣

De même :

$$K_C(s,r) \leq K_C(s) + K_C(r|s) + O(1)$$

(iv) Théorème 5.21 (relation fondamentale sur l'information conditionnelle)

$$K_C(s,r) = K_C(s) + K_C(r|\langle s, K_C(s) \rangle) + O(1)$$

ou : $K_C(s,r) = K_C(s) + K_C(r|w^*) + O(1)$

- *Preuve.* On se limitera ici à en donner l'idée générale [voir Li et Vitányi, 1997] : on se donne w^* (plus petit programme autodélimité qui calcule s) et v^* (plus petit programme

autodélimité qui calcule r étant donné w^* , c'est-à-dire s et $K_C(s)$, soit $\langle s, K_C(s) \rangle$. Alors à partir de la concaténation w^*v^* on peut successivement lire w^* , puis calculer s en exécutant w^* et calculer $K_C(s) = l(w^*)$, et enfin calculer r par l'exécution de v^* sur les données s et $K_C(s)$.

Remarque : calculer r directement à partir de la donnée s (et non pas à partir de son encodage w^*) par un programme τ^* supposerait la lecture de la chaîne concaténée $s\tau^*$, lecture rendue impossible par le fait que la chaîne s , contrairement à son encodage w^* , n'est pas autodélimitée. Autre solution : calculer r à partir de la donnée s par le programme v^* en calculant d'abord l'encodage w^* de s puis en produisant la chaîne, utilisable, w^*v^* ; mais le calcul de w^* à partir de s est impossible !



On voit ici apparaître une différence essentielle avec la théorie statistique de l'information, où $H(x,y) = H(x) + H(y|x)$: l'information contenue dans le couple (x,y) est égale à l'information contenue dans x plus l'information contenue dans y sachant x . Ici, il ne suffit pas de connaître s et r pour calculer (s,r) , car, d'une manière générale, étant donné une chaîne on ne sait pas calculer le programme minimal qui produit cette chaîne. La notion de complexité algorithmique, liée à la notion de calculabilité, introduit une dissymétrie fondamentale dans le flux informationnel. Si la théorie algorithmique de l'information produisait un résultat analogue à la théorie statistique (donc tel que $K_C(s,r) = K_C(s) + K_C(r|s)$), cela voudrait dire que la description minimum (i.e. de longueur minimum) d'un couple d'objets (s,r) serait égale à la description minimum de s plus la description minimum de r sachant s . Or cela est faux : l'information contenue dans (s,r) est égale à l'information contenue dans s plus l'information contenue dans r sachant non pas s , mais w^* , le programme minimum qui permet de calculer s . Le contenu d'information r est identifié par référence non pas à s , mais à l'encodage de s . Cela introduit en quelque sorte une notion "opératoire" d'effectivité dans la théorie de l'information : connaître une information, c'est non seulement en connaître le contenu, mais encore en connaître l'origine, la structure interne, le code, afin d'être capable de la produire, ne serait-ce que par un moyen aussi trivial que "PRINT s ". Car un tel programme apporte une information implicite supplémentaire par rapport à s tout seul : il signifie que, étant de longueur minimale, s est aléatoire et n'a donc pas de structure interne, de forme cachée. A contrario, si l'encodage w^* comprime en quelques lignes de programmation une chaîne s qui serait beaucoup plus longue, la connaissance de ce programme fait de la source connaissant r une source "intelligente", qui décrit r en référence non pas à une longue série d'items énumérés les uns à la

suite des autres, mais à une courte synthèse qui les résume tous.

(v) Information mutuelle : l'asymétrie de l'information mutuelle au sens de Kolmogorov découle de la propriété de sous-additivité :

$$\underline{\text{Définition 5.11}} : K_C(s:r) = K_C(s) - K_C(s/r)$$

Il vient :

$$|K_C(s:r) - K_C(r:s)| \geq \log l(s) - \log \log l(s) + O(1)$$

Ce résultat constitue une différence majeure avec la théorie shannonienne de l'information, mais est en accord avec l'analyse ternaire de la communication telle que nous l'avons présentée en section 1. Cela tient au fait que, étant donné deux chaînes transmises s et r , dont l'une, par exemple s , est considérée comme étant un contenu d'information alors que l'autre, r , est considérée comme une référence, c'est-à-dire une donnée associée à s , alors ce programme r doit lui-même pouvoir être interprété à l'aide d'un autre programme, ce qui suppose, comme on vient de le voir, une information supplémentaire "invisible" contenue dans la complexité de r .

5.2.4.3. Complexité préfixée explicite d'une source lexicale

(i) Cette information "invisible" doit en fait être exprimée explicitement dans le message de référence pour que celui-ci soit opérationnel, c'est-à-dire effectif. Il est donc nécessaire de préciser la définition de la complexité préfixée de s connaissant r , par :

Définition 5.12 : complexité préfixée "explicite" conditionnelle :

$$k_c(s|r) = K_C(s | \langle r, K_C(r) \rangle)$$

Avec :

$$k_c(s) = K_C(s) \quad (\text{en remplaçant } r \text{ par } \varepsilon \text{ dans la définition ci-dessus})$$

$$k_c(s,r) = K_C(s,r)$$

(ii) De cette définition particulière initialement proposée par Chaitin, on déduit un certain nombre de relations qui, cette fois, confèrent à cette quantité d'information la propriété d'additivité et à l'information mutuelle celle de symétrie :

$$\begin{aligned} k_c(s:r) &= K_C(s) - K_C(s | \langle r, K_C(r) \rangle) &&= K_C(r) - K_C(r | \langle s, K_C(s) \rangle) \\ &= K_C(s) - K_C(s | v^*) &&= K_C(r) - K_C(r | w^*) \\ &= k_c(s) - k_c(s|r) &&= k_c(r) - k_c(r|s) \end{aligned}$$

Nous donnons, sans démonstration, la liste des relations remarquables propres à la quantité k_c , en rappelant les relations correspondantes propres à K_C et H .

K_C	k_c	H
<i>(les relations sur $K_C(s)$ et $k_c(s)$ sont données à $O(1)$ près)</i>		
<i>monotonie</i>		
$K_C(s) \leq K_C(s,r)$	$k_c(s) \leq k_c(s,r)$	$H(s) \leq H(s,r)$
$K_C(s r) \leq K_C(s)$	$k_c(s r) \leq k_c(s)$	$H(s r) \leq H(s)$
<i>sous-additivité</i>		
$K_C(s,r) \leq K_C(s) + K_C(r)$	$k_c(s,r) \leq k_c(s) + k_c(r)$	$H(s,r) \leq H(s) + H(r)$
<i>complexité de la complexité</i>		
$K_C(s, K_C(s)) = K_C(s)$	$k_c(s, k_c(s)) = k_c(s)$	<i>n'existe pas pour H</i>
$K_C(K_C(s) s) \geq \log l(s) - \log \log l(s)$	$k_c(k_c(s) s) = 0$	"
<i>additivité</i>		
$K_C(s,r) = K_C(s) + K_C(r s, K_C(s))$		
$K_C(s,r) \leq K_C(s) + K_C(r s)$	$k_c(s,r) = k_c(s) + k_c(r s)$	$H(s,r) = H(s) + H(r s)$
$K_C(s s) = 0$	$k_c(s s) = 0$	$H(s s) = 0$
<i>symétrie</i>		
$K_C(s,r) = K_C(r,s)$	$k_c(s,r) = k_c(r,s)$	$H(s,r) = H(r,s)$
	$k_c(s:r) \geq 0$	$H(s:r) \geq 0$
	$k_c(s:s) = k_c(s)$	$H(s:s) = H(s)$
$ K_C(s:r) - K_C(r:s) $	$k_c(s:r) = k_c(r:s)$	$H(s:r) = H(r:s)$
$\geq \log l(s) - \log \log l(s)$	$= k_c(s) + k_c(r) - k_c(s,r)$	$= H(s) + H(r) - H(s,r)$

• Tableau 5.4 : propriétés de la complexité préfixée explicite

(iii) La fonction de complexité préfixée explicite conditionnelle n'est pas énumérable. Ce résultat fondamental est l'objet du théorème suivant :

Théorème 5.22 : la fonction $k_c(s|r)$ n'est pas co-énumérable.

□ On peut démontrer ce résultat par l'absurde [Chaitin, 1987, Li et Vitányi, 1997] :

Soit $k_c(s|r) = k_c(s,r) - k_c(r) = K_C(s,r) - K_C(r)$

Si $k_c(s|r)$ est co-énumérable, alors, r étant donné, $f(s) = 2^{-k_c(s|r)}$ est énumérable.

$$\Rightarrow \sum_s 2^{-k_c(s|r)} \leq 1$$

Donc pour un r donné (avec $l(r) \geq 1$), l'ensemble des $k_c(s|r)$ est l'ensemble des longueurs d'un code préfixé.

Or la complexité K_C est aussi une fonction énumérable. Donc pour un r donné (avec $l(r) \geq 1$), l'ensemble des $K_C(s|r)$ est aussi l'ensemble des longueurs d'un code préfixé. Par définition de la complexité K_C , c'est même l'ensemble des longueurs les plus courtes possibles. D'après le théorème d'invariance,

$$K_C(s|r) \leq k_c(s|r) + c$$

où c est une constante qui dépend de la machine utilisée et de la forme particulière que prend le programme de calcul (k_c), mais qui ne dépend pas de s .

Si l'on prend comme cas particulier $s = K_C(r)$, il vient :

$$K_C(K_C(r)|r) \leq k_c(K_C(r)|r) + c$$

Or par définition $k_c(K_C(r)|r) = K_C(K_C(r)|\langle r, K_C(r) \rangle)$. Donc $k_c(K_C(r)|r) = O(1)$, car de la chaîne $\langle r, K_C(r) \rangle$ on déduit immédiatement $K_C(r)$.

Mais par ailleurs on a vu que :

$$\exists r : K_C(K_C(r)|r) \geq \log l(r) - \log \log l(r) + O(1)$$

Il existe donc des r pour lesquels la quantité $K_C(K_C(r)|r)$ est largement supérieure à $k_c(K_C(r)|r)$, c'est-à-dire à une constante.

▣

Etant donné deux sources s et r associées dans un processus informationnel commun, comme par exemple un processus communicant, il faut, pour que le calcul ou la communication soient effectifs et que l'information soit réellement exploitable, que r (respectivement s) soit concaténée (par une méthode de préfixage) avec l'indication de sa complexité $K_C(r)$, c'est-à-dire de la longueur du plus petit programme qui permet de calculer r . Chacun de ces contenus d'information s , r , $K_C(s)$, $K_C(r)$ fait partie d'un ensemble énumérable ou co-énumérable, mais leur association – l'association d'une référence à un contenu – conduit à définir une fonction de complexité $k_c(s|r)$ qui, elle, n'est pas co-énumérable.

Corollaires :

(i) L'information préfixée explicite mutuelle n'est pas énumérable.

□ $K_C(s|\langle r, K_C(r) \rangle)$ n'est pas co-énumérable. Donc $\langle K_C(s|\langle r, K_C(r) \rangle) \rangle$ n'est pas énumérable.

Donc $k_C(s:r) = K_C(s) - K_C(s|<r, K_C(r)>)$ n'est pas énumérable.

▣

(ii) $k_c(s|r)$ n'étant pas énumérable, le procédé d'énumération mentionné au §5241-iii n'est plus utilisable.

Dans le cas de la quantité $K_C(s|r)$, on pouvait déterminer avec ce procédé le plus petit programme u^* de longueur $K_C(s|r)$ produisant s sachant r à partir de la donnée de la chaîne $\langle s, K_C(s|r) \rangle$, de la même façon que l'on retrouve le plus petit programme w^* produisant s à partir de la donnée de la chaîne $\langle s, K_C(s) \rangle$. Trouver directement u^* à partir de s sachant r était impossible car ce processus est non récursif. Mais du moins pouvait-on trouver u^* par énumération. Il s'agissait en quelque sorte de trouver une fonction particulière dans l'ensemble dénombrable des fonctions de 2^{K_C} dans \mathbb{N} .

Dans le cas présent, la situation est plus grave. Appelons u^{**} le plus petit programme qui calcule s à partir de la donnée $\langle s, k_C(s|r) \rangle = \langle s, K_C(s|v^*) \rangle = \langle s, K_C(s|<r, K_C(r)>) \rangle$, expression où apparaît deux fois la quantité $K_C =$ "longueur du plus petit programme", qui équivaut à rechercher une fonction particulière dans l'ensemble non dénombrable des fonctions de \mathbb{N} dans \mathbb{N} . Non seulement on ne peut trouver directement u^{**} à partir de s sachant r pour cause de non récursivité, mais encore on ne peut plus retrouver u^{**} par énumération pour cause de non dénombrabilité.

On se trouve ainsi face à une sorte de "mur de l'information", une impossibilité absolue et définitive d'éclairer complètement un processus informationnel associant symétriquement deux objets r et s .

5.2.5. Complexité monotone d'une source préfixée

Il existe plusieurs définitions de la complexité monotone, conduisant à des complexités légèrement différentes les unes des autres [Levin 1973, Schnorr 1977, Shen' 1984, Uspensky 1992]. Donnons la définition de Schnorr, reprise dans [Li et Vitányi, 1997, p 312] :

Définition 5.13 :

Une fonction récursive partielle $\Phi : \mathbf{B}^* \times \mathbb{N} \rightarrow \mathbf{B}^*$ est appelée interpréteur monotone ssi :

- $\forall (w,n) \in \text{domaine de } \Phi, l(\Phi(w,n)) = n$

- $\forall (w,n), (wv, n+k) \in \text{domaine de } \Phi, \Phi(w,n) \text{ est un préfixe de } \Phi(wv, n+k)$

Φ est exécutée par exemple par une machine de Turing qui posséderait deux bandes d'entrée (pour w et n). à lecture seule, une bande de travail et une bande de sortie à écriture seule. On en déduit une définition de la complexité monotone par rapport à Φ :

Définition 5.14 :

$$K_{K\Phi}(s) = \min \{l(w) : \Phi(w,l(s)) = s\}$$

$K_{K\Phi}(s) = \infty$ si un tel programme n'existe pas.

On montre que le théorème d'invariance est vrai pour cette forme de complexité. Donc, à une constante près, et pour tout Φ , on sait qu'il existe un interpréteur monotone de référence Φ_0 qui permet de poser $K_K(s) = K_{K\Phi_0}(s) = K_{K\Phi}(s) + O(1)$. Les propriétés "habituelles" de la complexité en découlent :

$$-K_K(s) \leq l(s) + O(1) \text{ et } K_K(r|s) \leq K_K(r) + O(1)$$

- $K_K(s)$ n'est pas récursive partielle.

-l'allure générale de $K_K(s)$, dont l'ordre de grandeur est $\log s$, est concave, sauf dans les zones où les chaînes sont de faible complexité.

- $K_K(s)$ est co-énumérable.

-le théorème de codage algorithmique de l'information s'applique :

$$K_K(s) = -\log \pi(s) + O(1)$$

$\pi(s)$ est une mesure algorithmique de la probabilité de la chaîne s calculée à partir de la longueur du plus petit programme autodélimité w^* qui calcule s .

5.3. Extension au continu des complexités préfixées

Considérons des chaînes dont les codes sont des programmes autodélimités. Plus le programme est long, plus la chaîne est improbable : $\pi(s) = 2^{-l(w^*)}$. Le préfixage des programmes permet d'attribuer à la quantité $\pi(s)$ la nature d'une probabilité car $\sum \pi(s) = 1$ (plus exactement ≤ 1 , mais on supposera le codage optimal). On peut étendre ce raisonnement

à un ensemble continu en conservant la condition d'une sommation égale à 1 :

$$\sum \pi(s) = 1 \rightarrow \int p(s) ds = 1$$

L'examen des propriétés de la complexité préfixée K_C montre que cette extension ne présente pas de difficulté particulière. Les chaînes s étant quelconques (lexicales) n'introduisent pas de contrainte supplémentaire sur la définition de leur probabilité, qui dépend uniquement de leur codage (préfixé).

En revanche l'extension au continu de la fonction de complexité monotone, que nous allons examiner maintenant, pose problème.

Schématiquement, rappelons tout d'abord que :

(a) structure en étoile d'un ensemble de chaînes : complexité d'une chaîne $s =$ longueur l d'une autre chaîne, nommée "programme" \Rightarrow probabilité *a priori* $p(s)$ d'une chaîne s définie à partir de la cardinalité $d(\mathbf{P}) = 2^l$ de l'ensemble \mathbf{P} des programmes de longueur l . Soit $p(s) = 2^{-l}$.

(b) structure en arbre d'un ensemble de chaînes préfixées : complexité d'une chaîne $s =$ longueur l d'une autre chaîne, autodélimitée \Rightarrow probabilité $\pi(s)$ d'une chaîne s définie par l'inégalité de Kraft, soit $\pi(s) = 2^{-l}$.

Mais l'extension au continu de ce schéma conduit à des conclusions incompatibles, car :

(a) si l'on étend au continu la cardinalité de \mathbf{P} , on ne peut plus définir de probabilité (seulement une densité de probabilité), car, stricto sensu, $p(s) = 0$.

(b) alors que l'extension $\sum 2^{-l} = 1 \rightarrow \int p(s) ds = 1$ rend possible la définition d'un ensemble aléatoire.

Dans le cas de la complexité préfixée, la probabilité d'une chaîne est fonction seulement de son code. On ne peut définir de probabilité sur l'ensemble des chaînes elles-mêmes, car cette mesure est nulle.

La situation de la complexité monotone est plus compliquée : chaîne et code étant tous deux préfixés, les probabilités associées coïncident-elles ? L'une, celle du code, est une probabilité algorithmique telle que, par le théorème de codage, $\pi(s) = 2^{-l(w^*)}$. L'autre est une probabilité a priori $\Pi(s)$ définie sur un ensemble continu, mais préfixé (une telle définition est donc possible). Il faut donc maintenant, si l'on considère un espace continu, différencier la complexité monotone K_K en :

K_{K_1} , ou "complexité a priori", fondée sur une semi-mesure de probabilité définie sur un ensemble continu :

$$K_{K_1}(s) = -\log(\Pi(s))$$

K_{K_0} , ou "complexité monotone", fondée sur une définition algorithmique :

$$K_{K_0}(s) = \min\{l(p) : \phi(p) = s\}$$

qui entraîne à son tour la définition d'une probabilité algorithmique $\pi(s)$ par l'intermédiaire de l'extension au continu du théorème de codage :

$$K_{K_0}(s) = -\log(\pi(s))$$

K_{K_1} est notée KA dans Levin 1973, KA ou ΣT dans Uspensky 1992, KM dans Li et Vitányi 1997.

K_{K_0} est notée km dans Levin 1973, KM ou TT dans Uspensky 1992, Km dans Li et Vitányi 1997.

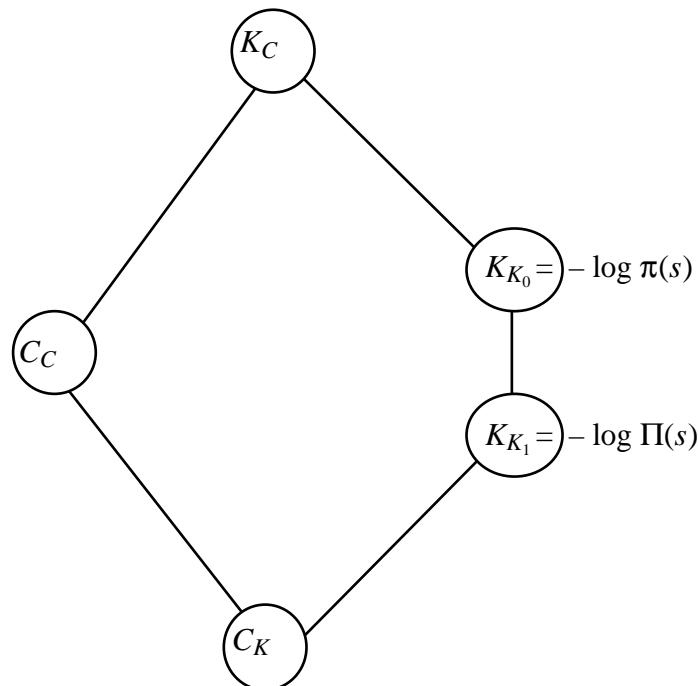
Or on montre [Gács, 1983] que $K_{K_0}(s)$ et $K_{K_1}(s)$ diffèrent d'un ordre de grandeur égal à l'inverse de la fonction d'Ackermann, qui est primitive récursive (contrairement à la fonction d'Ackermann elle-même, qui est calculable mais pas primitive récursive).

Le tableau 5.3 du § 521 se complète donc de la façon suivante (tableau 5.5) :

	$C_C(s)$	$K_{K_1}(s)$	$K_{K_0}(s)$	$K_C(s)$	$C_K(s)$
$l(s)$	O(1)	O(1)	O(1)	$Q_\lambda(l(s))$	O(1)
$C_C(s)$	-	$Q_\lambda(l(s))$	$Q_\lambda(l(s))$	$Q_\lambda(l(s))$	< 0
$K_{K_1}(s)$	$\log l(s)$	-	$A^{-1}(l(s))$	$Q_\lambda(l(s))$	< 0
$K_{K_0}(s)$	$\log l(s)$	< 0	-	$Q_\lambda(l(s))$	< 0
$K_C(s)$	< 0	< 0		-	< 0
$C_K(s)$	$Q_\lambda(l(s))$	$Q_\lambda(l(s))$	$Q_\lambda(l(s))$	$\log l(s) + Q_\lambda(l(s))$	-

- Tableau 5.5 : résultats des soustractions (à une constante O(1) près) : ordre de grandeur d'une entropie marquée en colonne moins ordre de grandeur d'une entropie marquée en ligne pour les ensembles continus.

Le schéma de la figure 5.8 s'en trouve modifié ainsi (figure 5.10) :



- Figure 5.10 : lorsqu'on étend le calcul de l'entropie algorithmique à des mesures continues, le théorème de codage ne s'applique plus à la complexité monotone : il existe une différence entre la complexité algorithmique et le logarithme de la mesure de probabilité. Cette différence est égale à l'inverse de la fonction d'Ackermann calculée sur la longueur des chaînes.

L'extension au continu de la complexité monotone K_K est donc impossible : coder un message écrit dans un certain code, puis un message lui-même codé décrivant ce code, etc, est un processus dont la complexité devient arbitrairement difficile, puisque la croissance de la fonction d'Ackermann est supérieure à celle de toute autre fonction calculable.

Exemple

On peut grossièrement illustrer cette "explosion combinatoire" de la façon suivante.

Chaque mot code est, on l'a vu, nécessairement préfixé. La "profondeur" de codage est égale au facteur k dans l'opération $l_k(s)$. Cette profondeur doit faire l'objet d'un accord préalable entre un émetteur et un récepteur. Or, une chaîne étant donnée, le nombre d'itérations de cette opération est égale à l'inverse de la fonction d'Ackermann $A(3, n)$. Aussi, s'il est possible de coder une chaîne préfixée à l'ordre k en une autre chaîne préfixée à l'ordre h , le nombre de possibilités de transcodage augmente comme $A(l(s))$. En terme de possibilités a priori évaluées par le récepteur (ie. le traducteur), un ensemble de chaînes étant

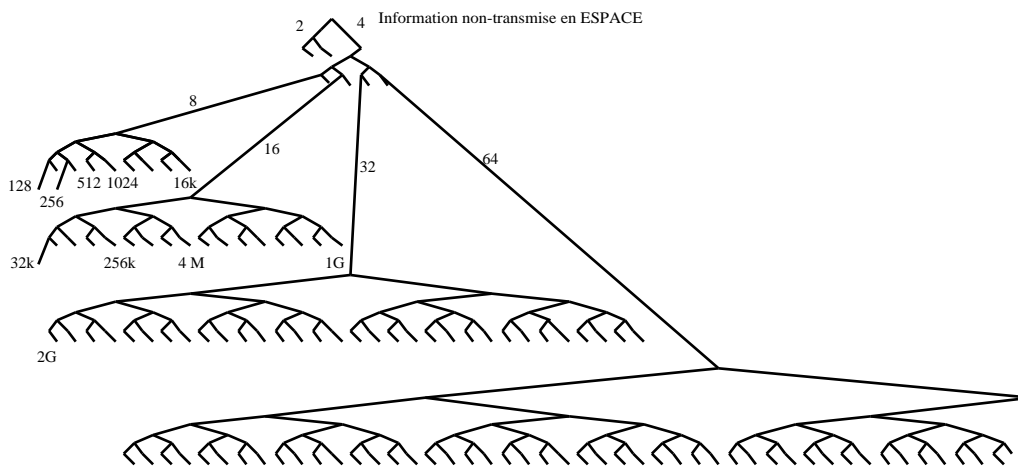
données, celles-ci ont une croissance qui, bien que dénombrable, excède en croissance n'importe qu'elle fonction primitive récursive (figure 5.11).

Or une définition de cette fonction, donnée par Odifreddi 1989, est :

$$A_0(x) = x+1 ; A_{n+1}(x) = A_n^{(x)}(x) ; A_\omega(x) = A_x(x)$$

$$\text{où : } A_n^{(0)}(x) = x ; A_n^{(k+1)}(x) = A_n(A_n^{(k)}(x))$$

Cette version de la fonction d'Ackermann peut être traduite en terme de cardinaux transfinis, ce qui permet ici de préciser l'origine exacte du non dénombrable dans la théorie de l'échantillonnage, lorsqu'un codage devient arbitrairement difficile.



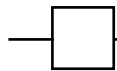
- *Figure 5.11* : le décodage du code E* s'effectue selon un arbre binaire dont la structure résulte d'une opération inverse du codage. Celui-ci est fondé sur une succession d'opérateurs logarithme binaire. Donc inversement le décodage fait intervenir un ensemble de choix binaires formant des sous-ensembles de cardinalité $2, 2^2, 2^{2^2}, 2^n \exp 2, 2^n \exp 2^n \exp 2$ etc

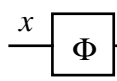
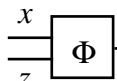
5.4. Résumé

5.4.1. Structure heuristique de la théorie algorithmique

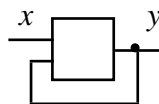
Comme dans les sections précédentes, nous allons recenser les différents calculs (ou, dans le cas présent, absence de calcul s'il s'agit de quantités non récursives) permettant d'évaluer les quantités d'information mutuelle $K(A:B)$. Nous nous limiterons au cas des complexités préfixées standard et explicite. Les constantes additives en $O(1)$ ne sont pas représentées.

Dans ce schéma, quelques symboles nouveaux apparaissent :


 : processus de calcul (cas général ; un exemple : l'énumération du §5241, iii)

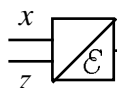
 ou  : interpréteur (machine de Turing) exécutant récursivement

$y = \Phi(x)$ ou $y = \Phi(x, z)$.

 : ce schéma symbolise un processus non récursif : il faut connaître le

résultat, ou tout au moins posséder certaines informations le concernant, par exemple sa longueur, pour pouvoir le calculer par une procédure telle que $y = \Phi(x, y)$ ou $\Phi(x, f(y))$.

 : l'opérateur logarithmique prend ici, par rapport aux schémas des sections précédentes, une double signification : c'est aussi l'opérateur $l(s)$ qui calcule la longueur d'une chaîne.

 : convertisseur effectuant la concaténation de deux chaînes en préfixant

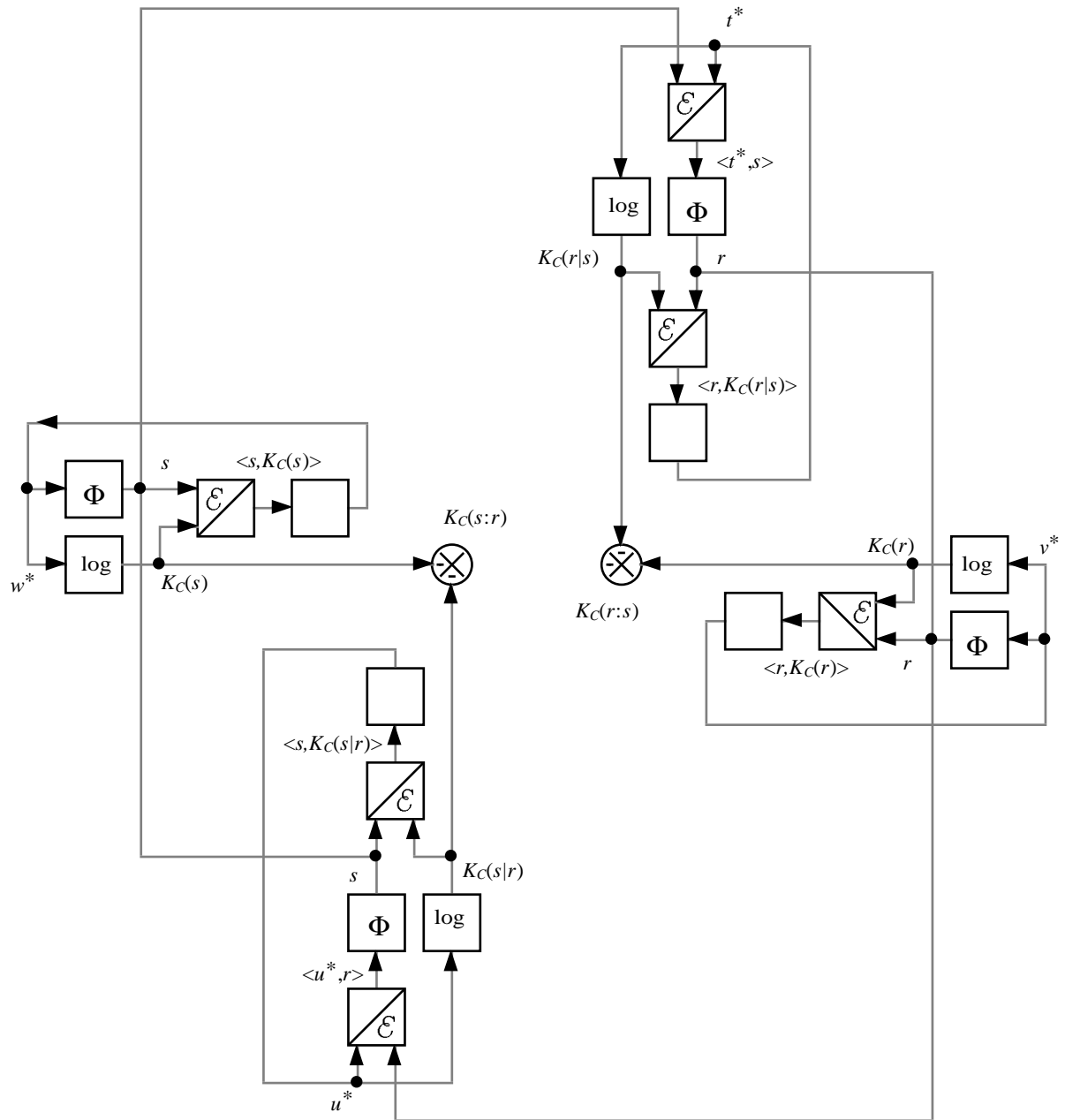
l'une d'entre elles : $y = l(x)xz$ ou $l(z)zx$.

Muni de ce symbolisme, le schéma ci-dessous (figure 5.12) reprend l'ensemble des opérations qui conduisent à $K_C(s:r)$ et $K_C(r:s)$. La complexité préfixée mutuelle n'étant pas symétrique, on aboutit à deux quantités d'information distinctes qui diffèrent numériquement et ne sont pas commensurables, ne provenant pas des mêmes calculs.

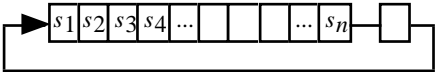
Pour mettre en évidence l'information réellement partagée par r et s , il faut faire intervenir la complexité préfixée explicite (schéma suivant, figure 5.13). Mais alors, d'une part ce qui est mis en commun ou communiqué ce n'est plus la chaîne elle-même (r ou s) mais sa description (resp. v^* ou w^*), d'autre part des quantités continues apparaissent (figurées en traits pleins), ce qui entraîne que le calcul par énumération de certains programmes (u^{**} et t^{**}) n'est plus possible.

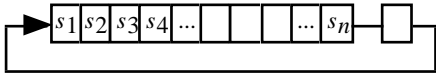
5.4.2. Résumé des propriétés des complexités algorithmiques

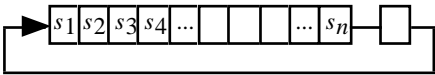
Nous résumons dans les pages suivantes quelques propriétés des entropies ou complexités algorithmiques C_C , C_K , K_C , k_c , K_K (discrète), en rapport avec les propriétés déjà énoncées de H , H_d et S . Comme précédemment, une propriété marqué en grisé est fausse pour

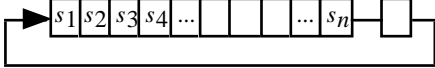


le type entropique considéré (tableaux 5.6 à 5.10).

<i>Tableau 5.6</i>	Complexité simple	(co-énumérable)
<i>Minimum</i>	$C_C(\epsilon) = 0$ $C_C(s s) = 0$	La complexité d'une chaîne vide ou d'une chaîne déjà connue est nulle.
<i>Majoration</i>	$C_C(s) \leq l(s) + O(1) = \log s + O(1)$	La complexité d'une chaîne est au plus égale à sa longueur, à une constante près.
<i>Concavité</i>	$C_C(s) \approx \log s$ presque partout	L'allure générale de la fonction $C_C(s)$ est concave, car presque toutes les chaînes sont aléatoires, donc de complexité $\approx \log s$. Mais il existe des exceptions : ce sont les chaînes de complexité faible.
<i>Monotonie</i>	$C_C(s,r) \not\leq C_C(s)$	La complexité globale d'une chaîne peut être $<$ à la complexité d'une ou plusieurs de ses parties.
<i>Additivité</i>	$C_C(s r) \leq C_C(s)$ $C_C(r,s) = C_C(r) + C_C(s r) + O(\log C_C(r,s))$	La connaissance d'une information sur A diminue l'incertitude sur B. $C_C(s)$ n'est pas additive.
<i>Sous-additivité</i>	$C_C(r,s) \leq C_C(r) + C_C(s) + 2\log(\min(C_C(r), C_C(s)))$	$C_C(s)$ n'est pas sous-additive : il existe par ex. des chaînes t.q. $C_C(r,s) > C_C(r) + C_C(s) + \log n$.
<i>Sous-additivité "forte"</i>		
<i>Signe</i>	$C_C(s) \geq 0$ $C_C(r,s) \geq 0$ $C_C(s r) \geq 0$ $C_C(r:s) \geq 0$	La complexité d'une chaîne est positive ou nulle
<i>Symétrie</i>	$ C_C(r:s) - C_C(s:r) = O(\log C_C(r,s))$	L'information contenue dans A à propos de B n'est pas la même que l'information contenue dans B à propos de A
<i>Changement de coordonnées</i>		La complexité d'une chaîne n'est pas invariante dans un décalage circulaire des caractères.
<i>Aléatoireité</i>	Non	Pas de mesure de probabilité. Pas de tests d'aléatoireité.

<i>Tableau 5.7</i>	Complexité uniforme	(co-énumérable)
<i>Minimum</i>	$C_K(\epsilon) = 0$ $C_K(s s) = 0$	La complexité d'une chaîne vide ou d'une chaîne déjà connue est nulle.
<i>Majoration</i>	$C_K(s) \leq l(s) + O(1) = \log s + O(1)$	La complexité d'une chaîne est au plus égale à sa longueur, à une constante près.
<i>Concavité</i>	$C_K(s) \approx \log s$ presque partout	L'allure générale de la fonction $C_K(s)$ est concave, car presque toutes les chaînes sont aléatoires, donc de complexité $\approx \log s$. Mais il existe des exceptions : ce sont les chaînes de complexité faible.
<i>Monotonie</i>	$C_K(s,r) \geq C_K(s)$	La complexité globale d'une chaîne est $>$ à la complexité d'une ou plusieurs de ses parties.
<i>Additivité</i>	$C_K(s r) \leq C_K(s)$	La connaissance d'une information sur A diminue l'incertitude sur B.
<i>Sous-additivité</i>		$C_K(s)$ n'est pas sous-additive.
<i>Sous-additivité "forte"</i>		
<i>Signe</i>	$C_K(s) \geq 0$ $C_K(r,s) \geq 0$ $C_K(s r) \geq 0$ $C_K(r:s) \geq 0$	La complexité d'une chaîne est positive ou nulle
<i>Symétrie</i>	$ C_K(r:s) - C_K(s:r) = O(\log C_K(r,s))$	L'information contenue dans A à propos de B n'est pas la même que l'information contenue dans B à propos de A
<i>Changement de coordonnées</i>		La complexité d'une chaîne n'est pas invariante dans un décalage circulaire des caractères.
<i>Aléatoirité</i>	Non	Pas de mesure de probabilité. Pas de tests d'aléatoirité.

<u>Tableau 5.8</u>	Complexité préfixée	(co-énumérable)
<i>Minimum</i>	$K_C(\epsilon) = 0$ $K_C(s s) = 0$	La complexité d'une chaîne vide ou d'une chaîne déjà connue est nulle.
<i>Majoration</i>	$K_C(s) \leq l(s) + Q_\lambda(l(s)) + O(1)$	La complexité d'une chaîne est au plus égale à sa longueur, augmentée du préfixe codant celle-ci, à une constante près.
<i>Concavité</i>	$K_C(s) \approx Q_\lambda(s)$ presque partout	L'allure générale de $K_C(s)$ est concave, à l'exception des zones de faible complexité.
<i>Monotonie</i>	$K_C(s,r) \geq K_C(s) + O(1)$	La complexité préfixée du tout est supérieure à la complexité des parties
<i>Additivité</i>	$K_C(s r) \leq K_C(s)$ $K_C(s,r) = K_C(s) + K_C(r s, K_C(s))$ $\leq K_C(s) + K_C(r s)$	La connaissance d'une information sur A diminue l'incertitude sur B. Mais la complexité préfixée n'est pas additive.
<i>Sous-additivité</i>	$K_C(s,r) \leq K_C(s) + K_C(r) + O(1)$	$K_C(s)$ est sous-additive.
<i>Sous-additivité "forte"</i>		
<i>Signe</i>	$K_C(s) \geq 0$ $K_C(r,s) \geq 0$ $K_C(s r) \geq 0$ $K_C(r:s) \geq 0$	La complexité d'une chaîne est positive ou nulle
<i>Symétrie</i>	$ K_C(s:r) - K_C(r:s) $ $\geq \log l(s) - \log \log l(s) + O(1)$	L'information contenue dans A à propos de B n'est pas la même que l'information contenue dans B à propos de A
<i>Changement de coordonnées</i>		La complexité d'une chaîne n'est pas invariante dans un décalage circulaire des caractères.
<i>Aléatoireté</i>	$\pi(s) = O\left(2^{-K_C(s)}\right)$	La source étant préfixée est probabilisable.

<i>Tableau 5.9</i>	Complexité préfixée explicite	(non-énumérable)
<i>Minimum</i>	$k_C(\epsilon) = 0$ $k_C(s s) = 0$	La complexité d'une chaîne vide ou d'une chaîne déjà connue est nulle.
<i>Majoration</i>	$k_C(s) \leq l(s) + Q_\lambda(l(s)) + O(1)$	La complexité d'une chaîne est au plus égale à sa longueur, augmentée du préfixe codant celle-ci, à une constante près.
<i>Concavité</i>	$k_C(s) \approx Q_\lambda(s)$ presque partout	L'allure générale de $k_C(s)$ est concave, à l'exception des zones de faible complexité.
<i>Monotonie</i>	$k_C(s,r) \geq k_C(s) + O(1)$	La complexité du tout est supérieure à la complexité des parties
<i>Additivité</i>	$k_C(s r) \leq k_C(s)$ $k_C(s,r) = k_C(s) + k_C(r s)$	La complexité préfixée explicite est additive.
<i>Sous-additivité</i>	$k_C(s,r) \leq k_C(s) + k_C(r) + O(1)$	$k_C(s)$ est sous-additive.
<i>Sous-additivité "forte"</i>		
<i>Signe</i>	$k_C(s) \geq 0$ $k_C(r,s) \geq 0$ $k_C(s r) \geq 0$ $k_C(r:s) \geq 0$	
<i>Symétrie</i>	$k_C(s:r) = k_C(r:s)$ $= k_C(s) + k_C(r) - k_C(s,r)$	L'information contenue dans A à propos de B est la même que l'information contenue dans B à propos de A
<i>Changement de coordonnées</i>		La complexité d'une chaîne n'est pas invariante dans un décalage circulaire des caractères.
<i>Aléatoireté</i>	$\pi(s) = O\left(2^{-k_C(s)}\right)$	La source étant préfixée est probabilisable.

<i>Tableau 5.10</i>	Complexité monotone	(co-énumérable)
<i>Minimum</i>	$K_K(\epsilon) = 0$ $K_K(s s) = 0$	La complexité d'une chaîne vide ou d'une chaîne déjà connue est nulle.
<i>Majoration</i>	$K_K(s) \leq l(s) + O(1)$	La complexité d'une chaîne est au plus égale à sa longueur à une constante près.
<i>Concavité</i>	$K_K(s) \approx \log s$ presque partout	L'allure générale de $K_K(s)$ est concave, à l'exception des zones de faible complexité.
<i>Monotonie</i>	$K_K(s,r) \geq K_K(s) + O(1)$	La complexité du tout est supérieure à la complexité des parties
<i>Additivité</i>		
<i>Sous-additivité</i>		
<i>Sous-additivité "forte"</i>		
<i>Signe</i>	$K_K(s) \geq 0$ $K_K(r,s) \geq 0$ $K_K(s r) \geq 0$ $K_K(r:s) \geq 0$	La complexité d'une chaîne est positive ou nulle
<i>Symétrie</i>		
<i>Changement de coordonnées</i>		La complexité d'une chaîne n'est pas invariante dans un décalage circulaire des caractères.
<i>Aléatoireté</i>	$ K_{k_0}(s) - K_{k_1}(s) = \text{Ack}^{-1}(s)$	Dans le cas discret, une source préfixée est nécessairement probabilisable. Mais l'extension au continu entraîne l'impossibilité de comparer probabilité $\pi(s)$ et complexité. $K_K(s)$

5.5. Conclusion : énumérabilité du message et du contexte

5.5.1. Complexités des processus informationnels

5.5.1.1. Problématique générale

"L'ensemble des sous-ensembles d'un ensemble infini dénombrable n'est pas dénombrable" est un théorème qui fonde l'analyse de la calculabilité en informatique théorique. Un mot est une chaîne de caractères finie écrite dans un alphabet donné, qui code une procédure algorithmique effective. L'ensemble des mots est un ensemble dénombrable. Un langage est un sous-ensemble de l'ensemble des mots, qui code les instances positives d'un problème. L'ensemble des langages est non dénombrable. «Il ne peut donc y avoir une procédure effective pour chaque problème !» conclut P. Wolper [Wolper, 1992] : l'ensemble des questions est bien plus vaste que l'ensemble des réponses.

La démarche exposée dans cette section conduit à étendre ces réflexions générales aux processus informationnels. Les messages que partagent deux sources sont formés de suites finies de signes constituant un ensemble de mots M infini dénombrable. Une source lit ou écrit chaque mot par identification dans un langage formant un sous-ensemble de l'ensemble des langages *a priori* possibles. La connaissance de ce langage suppose la présence d'un contenu d'identification contextuel à l'information transcrite. Cette part d'information, formée d'un élément de $\mathcal{P}(M)$ et non de M lui-même, serait de nature continue.

Nous proposons pour ce schéma général la modélisation suivante.

(i) *Proposition 5.1* : nous considérons un processus informationnel associant (au minimum) deux sources alphabétiques.

Cette proposition accompagne une hypothèse minimale sur la connaissance préalable de la constitution interne des sources : chaque état symbolique d'une source est étiqueté par un nombre. La topologie de l'ensemble de ces nombres est une structure en étoile. Aucun nombre n'a plus de signification *a priori* qu'un autre, aucun ordre n'est affecté à cet ensemble. Les sources sont au minimum des sources lexicales. On dispose d'un alphabet pour transcrire les états symboliques des sources en mots.

(ii) *Proposition 5.2* : un processus informationnel est effectif quand il termine.

Une communication est opérationnelle si le récepteur dispose de l'information nécessaire pour décoder le message, de l'information nécessaire pour décoder le code, etc. Pour sortir de ce cercle vicieux, il faut qu'au message soit associée *toute* l'information de référence nécessaire au décodage de ce message. "*Toute*" signifie que l'information est complète et autosuffisante.

En généralisant aux processus informationnels quelconques, c'est-à-dire à toute relation entre deux sources A et B ("mise en relation" est le sens étymologique du terme "communication"), tout partage d'information est effectif si l'accès aux ressources communes est équilibré, et que chaque source peut calculer effectivement et de façon autonome les chaînes qu'elle lit ou écrit.

(iii) Proposition 5.3 : un processus informationnel effectif est symétrique.

La symétrie est un corollaire de l'effectivité du processus : dans une communication, le résultat de la transmission de l'information x de A vers B est réellement un partage, si B est en retour en mesure de transmettre x à A . L'effectivité du partage entraîne qu'on doit pouvoir déterminer x aussi bien à partir de A qu'à partir de B , soit :

$$x = I(A:B) = I(A) - I(A|B) = I(B) - I(B|A)$$

Si le processus était asymétrique, cela voudrait dire qu'une partie de l'information lue ou écrite par B serait contenue dans A et inaccessible à B (et réciproquement), qui ne pourrait donc terminer certains calculs faute de données.

(iv) Proposition 5.4 : L'information partagée est discernable et minimale.

On ne se limite pas à un unique message partagé entre deux sources lexicales : le partage concerne des textes. Il faut donc que les messages (et les signes) soient discernables. Cela constitue une première exigence. Un des moyens d'atteindre ce but est le préfixage, qui remplit une deuxième exigence : la compression maximale des messages.

Nous ajoutons donc un encodage préfixé aux sources alphabétiques initiales. La complexité de cet encodage est une complexité préfixée. Le résultat de cet encodage est une source préfixée, transcription de la source lexicale correspondante.

(v) Proposition 5.5 : un processus informationnel effectif associe dans l'information conditionnelle un contenu d'identification explicite et un contenu d'identification implicite.

Cela dérive immédiatement de la notion de complexité effective, qui elle-même est imposée par les conditions de préfixage (discernabilité) et de symétrie. Dans une quantité conditionnelle $I(r|s)$, s permet l'identification de r : c'est en connaissant s que l'on peut

déterminer ce qui appartient en propre à r et à r seul.

Mais il faut maintenant considérer les quantités $\langle s, K_C(s) \rangle$ et $\langle r, K_C(r) \rangle$, où apparaît explicitement une dualité : si s et r sont les contenus (connus), $K_C(s)$ et $K_C(r)$ autorisent l'effectivité des procédures (de calcul, de communication, etc) en les référenciant. Rappelons que la quantité $\langle s, K_C(s) \rangle$ (resp. $\langle r, K_C(r) \rangle$) contient la même information que w^* (resp. v^*), qui est le plus petit programme qui produit s (resp r) :

$$\begin{aligned} s = \Phi(w^*) &\Leftrightarrow K_C(s) = I(w^*) & r = \Phi(v^*) &\Leftrightarrow K_C(r) = I(v^*) \\ k_c(r|s) = K_C(r|\langle s, K_C(s) \rangle) &= K_C(r|w^*) & k_c(s|r) = K_C(s|\langle r, K_C(r) \rangle) &= K_C(s|v^*) \\ k_c(s:r) = K_C(s) - K_C(s|v^*) &= & k_c(r:s) = K_C(r) - K_C(r|w^*) \end{aligned}$$

On peut interpréter cela de deux façons complémentaires :

- On considère s et r comme deux informations associées. Pour expliciter ce qui leur est commun, il faut par exemple pourvoir retrancher de l'information contenue dans r l'information contenue dans s sachant non seulement s (contenu d'identification explicite) mais encore la manière de produire s (l'information w^* à laquelle on a accès par l'intermédiaire de sa "référence" $K_C(s)$: contenu d'identification implicite).

Pour exprimer ce que deux objets A et B ont en commun, il ne suffit pas de comparer A à B , il faut aussi savoir produire B (ou vice versa). On retrouve ici le sens étymologique du mot "in-formation" : une mise en forme.

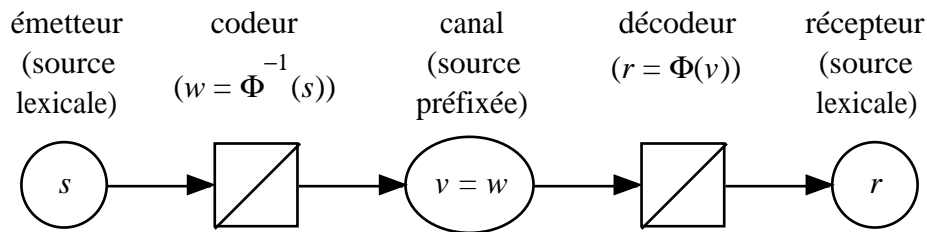
- On considère s et r comme les deux sources d'un processus communicant. La transmission $k_c(s:r)$ est effective si le récepteur (par ex. r) est capable de distinguer ce qui lui appartient en propre, l'émetteur étant connu, soit $K_C(r|w^*)$, de l'information totale $K_C(r)$ qu'il contient. Mais cette connaissance de l'émetteur par le récepteur est en quelque sorte une connaissance "pratique", puisqu'elle porte non seulement sur l'information s de l'émetteur mais aussi sur l'information $K_C(s)$ nécessaire au calcul du programme de calcul de s .

Pour que deux sources A et B communiquent, il ne suffit pas que A et B dialoguent, il faut aussi qu'elles échangent les connaissances qui permettent ce dialogue, ou posséder ces connaissances à l'origine.

Exemple : système de transmission :

- La modélisation usuelle d'un système de communication non bruité, comme le

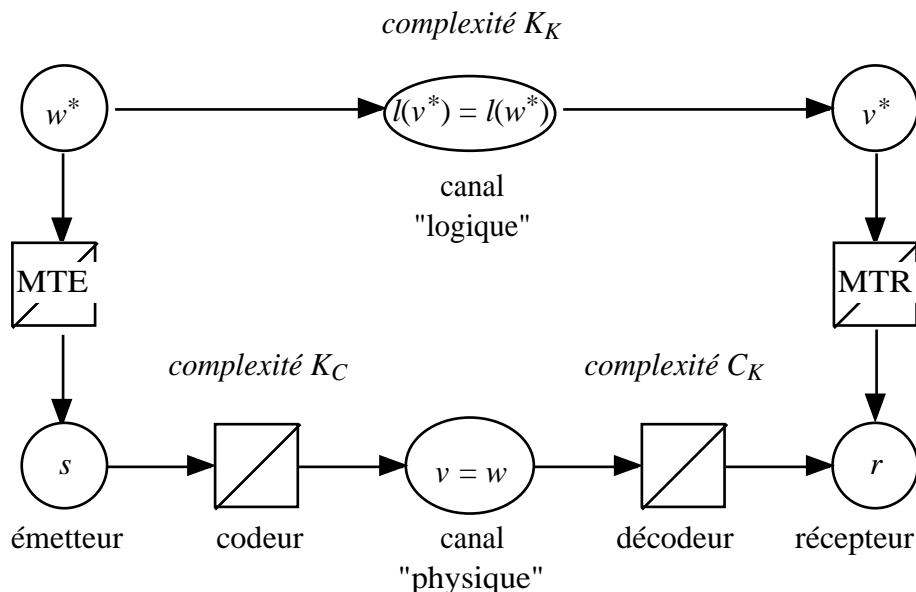
proposait Shannon dans son article initial, est simple. Il existe une bijection entre les codes émis (w est le mot-code correspondant à la source s) et les codes reçus (v est le mot-code correspondant au destinataire r). Il est implicitement convenu que les procédures de codage/décodage sont effectives, et telles que $v = w$.



• *Figure 5.14* : schéma statistique de communication

Notons que ce n'est pas en soi le fait de devoir calculer la fonction $\Phi^{-1}(s)$ qui pose problème. En effet, si la fonction Φ est calculable, la fonction inverse Φ^{-1} l'est aussi. Pour calculer $y = \Phi^{-1}(x)$, il suffit d'énumérer \mathbb{N} en calculant Φ jusqu'à ce que l'on retrouve x (quoique cette méthode ne marcherait pas dans le cas semi-calculable, car les valeurs de y qui font que $\Phi(y)$ ne termine pas empêchent la poursuite de l'énumération de \mathbb{N}). D'un point de vue général, encoder une source s sous la forme d'un programme minimum w^* , puis décoder ce message pour produire un résultat r identique à s , revient à calculer le plus petit programme qui produit un résultat donné, et ce problème est récursivement énumérable.

- Le point de vue algorithmique ajoute au schéma précédent une connaissance supplémentaire concernant les procédures de calcul employées dans la communication. Nous proposons le schéma de communication suivant :



- *Figure 5.15* : schéma algorithmique de communication (MTE : Machine de Turing de l'émetteur ; MTR : MT récepteur). Dans le cas idéal d'un compression maximale, on aurait $v = v^*$ et $w = w^*$

Dans ce schéma figurent explicitement les procédures de calcul : $s = (w^*)$, notée "MTE". Et $r = \Phi(v^*)$, notée MTR. Pour transmettre de l'information depuis l'émetteur vers le récepteur, on pourrait croire qu'il suffit de communiquer $w = \Phi^{-1}(s)$ au récepteur, en s'assurant que $v = w$. Or les résultats de calculabilité introduisent une dissymétrie fondamentale dans le flux informationnel : on sait il n'y a pas moyen de savoir si le programme de calcul w expédié au récepteur est équivalent au programme v qui produit r . Le problème de savoir si deux programmes sont équivalents est un problème indécidable. On ne peut donc être certain que le résultat de la communication r est bien tel que $r = s$, parce qu'il est impossible de savoir si $v = w$, *sauf...* si l'on transmet non seulement s , mais aussi le programme w lui-même qui calcule s . D'où la notion de "double" canal de communication.

Les chaînes s et w^* composent l'ensemble réel du message : s représente les données effectivement transmises, w^* le contenu implicite et explicite par rapport auquel les mots s doivent être décodés. Ce contenu d'identification w^* est un programme, élément d'un ensemble récursivement énumérable, partagé par l'émission (codage) et la réception (décodage, avec $v^* = w^*$). Donc la voie de communication est double : à une voie "physique" qui supporte les données explicites, il faut adjoindre une voie "logique" qui traite de la façon avec laquelle les messages sont construits. L'effectivité d'une procédure communicante repose sur l'existence d'un double canal de communication.

(vi) Proposition 5.6 : la présence d'une information contextuelle implicite dans l'information conditionnelle rend celle-ci non énumérable.

On a vu que la fonction $k_c(r|s) = K_C(r|w^*) = K_C(r| \langle s, K_C(s) \rangle)$ qui contient une information contextuelle implicite nécessaire au calcul effectif de l'information r sachant s n'est pas énumérable.

D'un point de vue général, à quels langages appartiennent les messages émis et reçus ? Il n'y a pas de raison a priori pour que les langages L et L' des deux sources coïncident ou alors, s'ils coïncident, l'information concernant leur identité est elle-même une information, qu'il faut transmettre... mais dans quel langage ? Au message émis s_i doit correspondre le message reçu r_j par l'intermédiaire d'un mot-code commun x_{ij} . L'ensemble des messages s_i est dénombrable, celui des r_j aussi, mais il existe une infinité non dénombrable de façons \mathcal{R}_{ij} d'associer les réponses r_j aux données s_i par une fonction de \mathbb{N} dans \mathbb{N} . Or c'est précisément cette relation \mathcal{R} qui ne fait l'objet d'aucun transfert informationnel au cours d'une transaction : il s'agit d'un "standard" sur lequel émetteur et récepteur doivent être préalablement accordés.

(vii) Proposition 5.7 : la complexité d'un contenu d'identification implicite est monotone.

Sous-jacent à la notion d'information duale, existe le problème de la communication des chaînes programmes v^* et w^* codant les chaînes sources s et r . Peut-on expliciter totalement un processus informationnel ? Il faudrait envisager de partager les chaînes autodélimitées v^* et w^* de façon discernable, donc sous forme elle-même autodélimitée, ce qui revient à invoquer l'existence des complexités monotones $K_K(v^*)$ et $K_K(w^*)$, longueurs des programmes t^* et u^* qui codent respectivement ces chaînes.

(viii) Question ouverte : sur l'effectivité des processus informationnels.

On sait que l'extension au continu de la complexité uniforme K_K est impossible, car cette extension conduit à distinguer la complexité uniforme proprement dite des chaînes émises, de la complexité a priori des chaînes reçues.

Cela conduit à la conclusion suivante :

- l'échange explicite de contenu d'information (les données) est nécessairement discret.

Cette conclusion est en accord avec ce que l'on savait déjà de l'entropie statistique classique et classique quantique.

- le partage de contenu d'identification n'est pas énumérable.

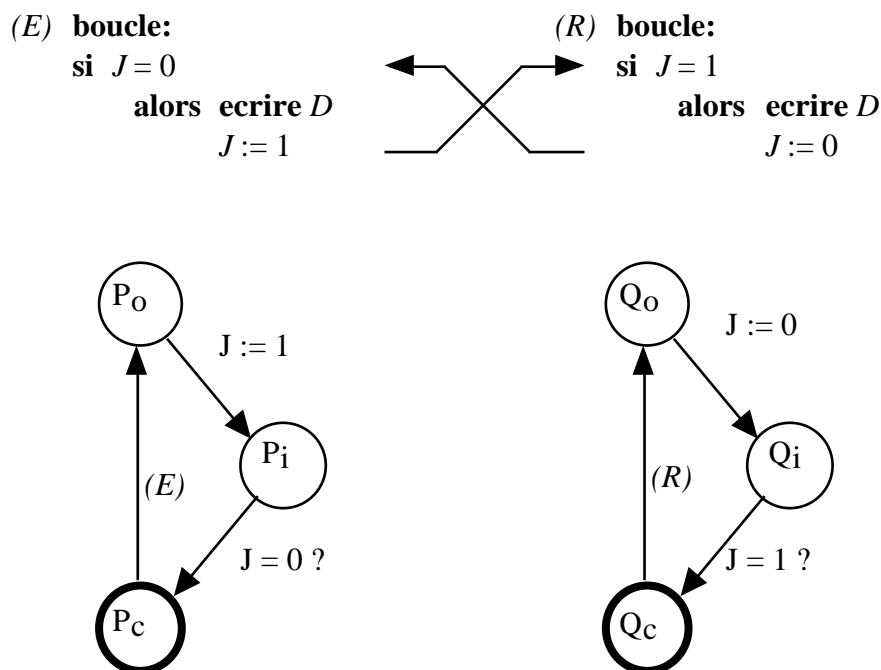
- un processus informationnel associe un contexte partagé à des données échangées. Pour être effectif, il faudrait que ce processus transmette de façon explicite toute l'information contextuelle, ce est impossible, car cela reviendrait à échanger sous forme discrète une information par nature continue. Donc un processus informationnel ne saurait être effectif, sauf si les deux sources partagent préalablement l'information contextuelle nécessaire à son effectivité.

Cette dernière éventualité s'appliquerait notamment aux sources faibles et à l'entropie quantique quantique (la corrélation quantique jouant le rôle d'un partage effectif).

5.5.1.2. Exemple

Des exemples ont déjà été donnés de l'association entre contenu d'information et contenu d'identification. On exposera ici le cas de l'exclusion mutuelle entre deux processus communicants, pour montrer que les propriétés qui ont été évoquées – dualité du canal, non-énumérabilité du contexte – apparaissent dès le niveau élémentaire des procédés de transmission.

Soit le processus suivant :

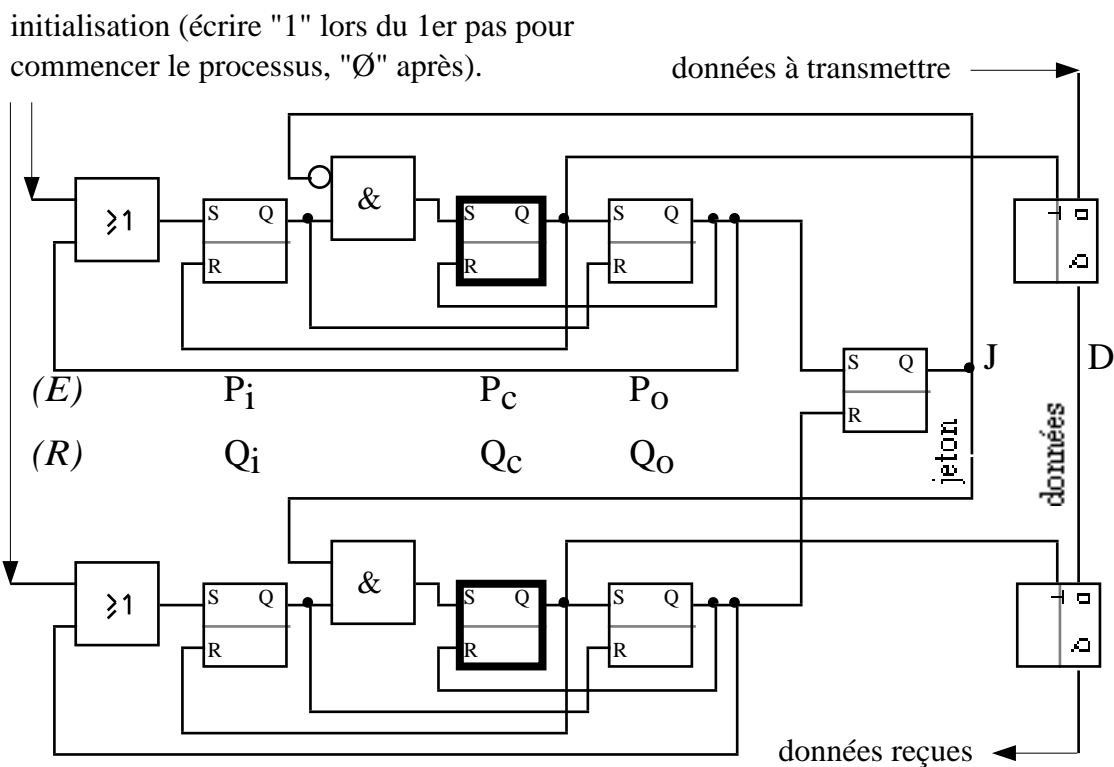


- Figure 5.16 : Algorithme d'exclusion mutuelle entre deux processus communicants. Le transfert informationnel par partage de la variable D constitue une section critique (notée P_c et Q_c) de chaque processus : si l'un parle, l'autre écoute. Il ne faut donc pas que E et R accèdent en même temps

à leurs sections critiques respectives. Cela est assuré par le partage d'une deuxième variable, le "jeton" J .

- *Remarque* : on peut raffiner cet algorithme pour éviter qu'un des deux interlocuteurs ne "monopolisent" la parole en restant indéfiniment dans sa section critique (algorithme de Peterson).

On donne généralement, pour satisfaire le principe d'exclusion mutuelle entre deux processus partageant une même variable, le système concurrent (type "poignée de main") représenté figure 8, comme la solution minimale à ce problème [cf par ex. Gochet, Gribomont, 1994]. La variable J (jeton), partagée, détermine quel processus a le droit d'accéder à sa section critique, et écrire ou lire la variable partagée D . Le schéma ci-dessous propose une version matérielle de ce processus, qui ne fait donc intervenir que des fonctions booléennes élémentaires (ET, OU, NON, ...), voire une seule fonction (ET-NON par ex.) :



- *Figure 5.17* : réalisation matérielle à l'aide de mémoires RS et D. La section critique de chaque processus est matérialisée par une mémoire RS repérée par un trait épais.

Un invariant du système est la conjonction des quatre assertions suivantes :

$$\begin{aligned}
 & [at P_i \Rightarrow (J = 1 \vee at Q_i)] \\
 & \wedge [at Q_i \Rightarrow (J = 0 \vee at P_i)] \\
 & \wedge [at P_c \Rightarrow (J = 0 \vee \neg at Q_0)]
 \end{aligned}$$

$$\wedge [at Q_c \Rightarrow (J = 1 \vee \neg at P_0)]$$

On en déduit :

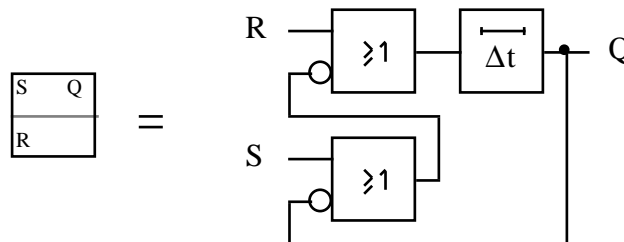
$$\Rightarrow (at P_c \Rightarrow J = 0) \wedge (at Q_c \Rightarrow J = 1)$$

qui décrit le rôle de la variable J : à aucun moment les deux processus ne peuvent accéder à leur section critique en même temps (chacun attend la fin du message de son interlocuteur pour prendre la parole à son tour...)

On remarque qu'un tel protocole minimal met en jeu *deux* voies de communication, par D et J : le fait de communiquer D implique nécessairement de communiquer aussi J , pour assurer un déroulement bien ordonné des échanges. On peut généraliser ce procédé au cas où les variables partagées D et J seraient constituées de messages (préfixés) indépendants et quelconques. Puisque cette communication minimale nécessite de toutes façons deux voies d'échange, on peut en toute généralité considérer D et J comme un couple de transferts informationnels jouant des rôles symétriques.

Un examen approfondi de l'algorithme d'exclusion mutuelle fait apparaître une composante continue inhérente au processus communicant, dûe aux propriétés qu'il est nécessaire d'attribuer à la variable *temps* pour en assurer le bon fonctionnement.

Le schéma fait intervenir, pour donner un équivalent booléen de l'algorithme, un certain nombre de mémoires RS et D. Mais l'équivalent exprimé en termes de logique combinatoire d'une mémoire implique l'existence d'opérateurs retard Δt rendus nécessaires pour assurer un fonctionnement non aléatoire de ces mémoires (figure 5.18)



• Figure 5.18 : schéma équivalent d'une mémoire RS en logique combinatoire

Le fonctionnement d'une bascule est représenté par l'équation :

$$Q_{n+1} = S \vee \neg (R \vee \neg Q_n)$$

Et les états du processus par le système :

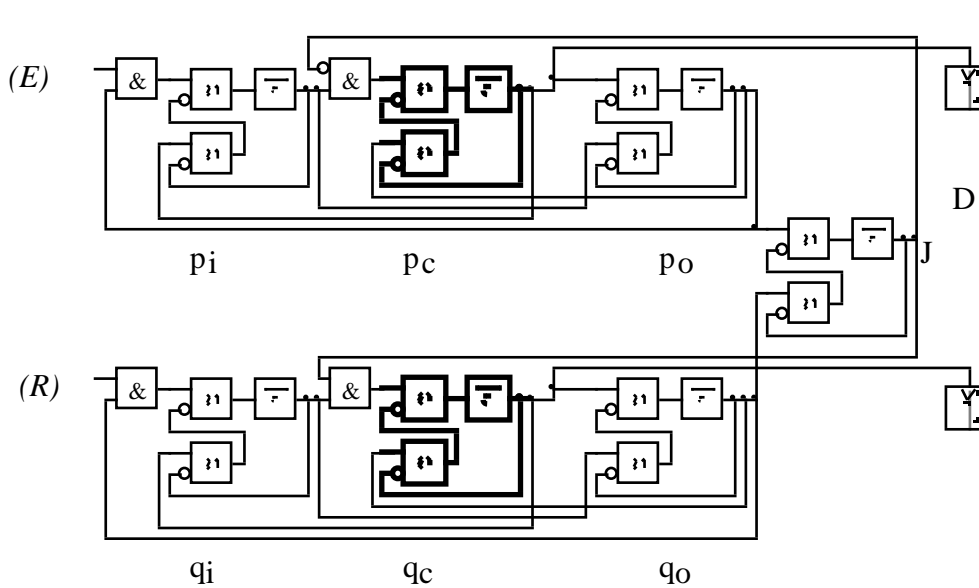
$$P_{i_{n+1}} = (E \wedge P_{o_n}) \vee \neg (P_{c_n} \vee \neg P_{i_n}) \quad Q_{i_{n+1}} = (R \wedge Q_{o_n}) \vee \neg (Q_{c_n} \vee \neg Q_{i_n})$$

$$P_{c_{n+1}} = (\neg J \wedge P_{i_n}) \vee \neg (P_{o_n} \vee \neg P_{c_n}) \quad Q_{c_{n+1}} = (J \wedge Q_{i_n}) \vee \neg (Q_{o_n} \vee \neg Q_{c_n})$$

$$P_{o_{n+1}} = P_{c_n} \vee \neg (P_{i_n} \vee \neg P_{o_n}) \quad Q_{o_{n+1}} = Q_{c_n} \vee \neg (Q_{i_n} \vee \neg Q_{o_n})$$

$$J_{n+1} = P_{O_n} \vee \neg (Q_{O_n} \vee \neg J_n)$$

Parmi toutes les mémoires qui entrent dans la composition du schéma général, il est particulièrement important d'examiner le comportement de celles qui correspondent à P_c et à J . Or il est facile de voir que, sous certaines conditions, les opérateurs retard introduisent des aléas dans la marche du processus :

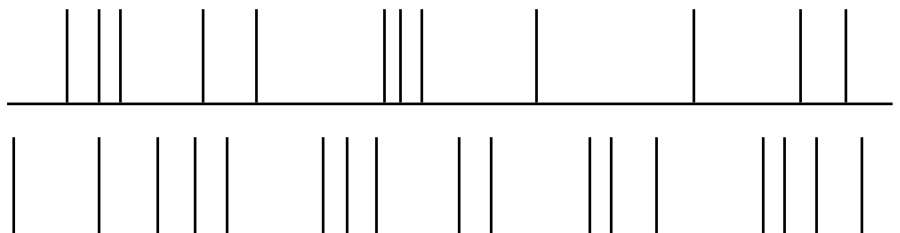


• Figure 5.19 : réalisation de l'algorithme d'exclusion mutuelle à l'aide de circuits combinatoires

Une description logique du temps est à l'origine de ces aléas. A deux RS en interaction correspondent deux équations récursives :

$$\begin{aligned} Q_{n+1} &= S \vee \neg (R \vee \neg Q_n) & \text{et} & & Q'_{m+1} &= S' \vee \neg (R' \vee \neg Q'_m) \\ \Rightarrow \forall n, \exists m [n < m < n+1] & & \text{et} & & \forall m, \exists n [m < n < m+1] \end{aligned}$$

Pour que la récursivité "fonctionne" de chaque côté, indépendamment, il faut qu'à tout moment, entre deux instants quelconques définis par exemple chez le récepteur, on puisse définir un instant propre à l'émetteur, et réciproquement :



• Figure 5.20 : dans tout couple d'événements successifs appartenant à l'une

des sources, il faut pouvoir intercaler un événement survenant dans l'autre source.

Ce qui conduit à formuler l'hypothèse d'un ordre continu (bon ordre, total, dense) pour modéliser le temps commun aux deux systèmes.

5.5.2. Trois classes de calculabilité

1°) Dans un processus communicant, la nécessité de transmettre les chaînes autodélimitées u^* et t^* entraîne qu'une part de l'évaluation de l'information liée au récepteur est continue. Mais la transmission de l'information de l'émetteur vers le récepteur ne peut être qu'un processus discret. Un processus d'échange informationnel met en jeu des messages dénombrables dans un contexte en partie non-énumérable.

2°) Ces résultats s'étendent à tout système informationnel bipolaire. En fait, si l'on considère les fonctions comme des processus communicants [Milner, 1991], ces considérations théoriques conduisent à faire la distinction entre trois classes de calculabilité [Anceau, 1997] :

- celle des automates finis, qui manipulent de l'information booléenne (les ordinateurs...)
- celle des machines de Turing, qui manipulent des entiers.
- celle des processus communicants effectifs, qui supposent la manipulation implicite du continu.

L'approximation que réalise un ordinateur du fonctionnement d'une machine de Turing s'apparente à un "fenêtrage" (*windowing*, au sens du traitement des signaux), c'est-à-dire une coupure dans un ensemble potentiellement infini de données. Elle se traduit par le codage, dans un alphabet de valence finie, des entiers de l'ensemble \mathbb{N} par des mots de longueur finie, qui sont des suites de bits dans le cas d'un alphabet de valence 2.

Maintenant, on peut ajouter que l'approximation que réalise ce même ordinateur des processus de calcul vus sous l'angle général de processus de communication effectifs s'apparente à un échantillonnage (*sampling*, selon la terminologie du traitement des signaux), qui quantifie le continu en unités élémentaires de volume non nul.

La question qui se pose est alors la suivante : peut-on approcher par échantillonnage les caractéristiques continues des processus informationnels ?