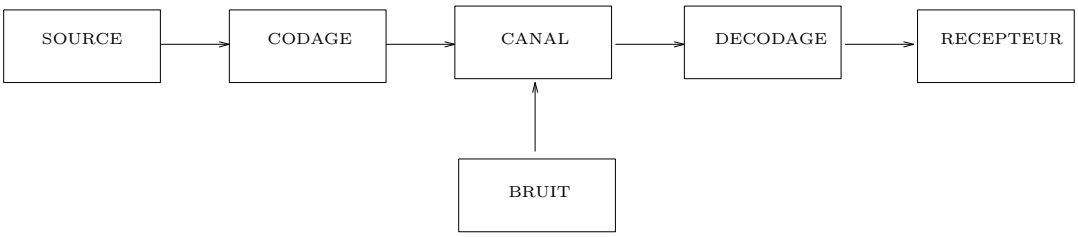
Codage
Première partie

Carla Selmi, Département d'Informatique, Université de Rouen

1 Introduction

Ce cours est dédié à l'étude des techniques qui permettent de coder sans ambiguïté et efficacement, par rapport aux objectifs que l'on désire atteindre, les messages émis par une source. Ces messages sont codés en utilisant un ensemble de mots sur un alphabet différent de celui utilisé par la source, de manière telle à pouvoir les envoyer en le faisant passer par un canal de transmission. Le schéma suivant, dit de Shannon, illuste cette thématique.



Dans ce dessin, *SOURCE* représente la source qui émet les messages, *CODAGE* le procédé qui permet de coder les messages émis par la source, *DECODAGE* le processus qui permet de retrouver le message émis par la source, *RECEPTEUR* le récepteur du message et *CANAL* est le canal de transmission par lequel le message est transmis de la source au récepteur.
Entre le codage et le décodage, le message peut subir des altérations et être partiellement modifié. On dit, dans ce cas, que le message est soumis à un *BRUIT* ou que le canal est bruité.
Le couple codage-décodage doit être adpté au canal de transmission et aux objectifs que l'on désire atteindre :

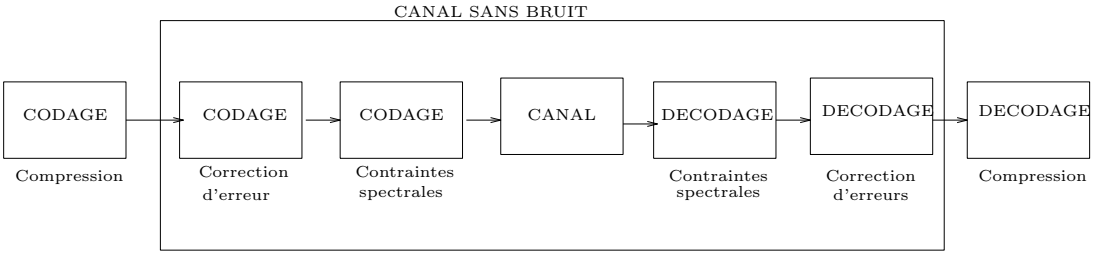
- En *cryptographie*, on cherche à coder un message de manière telle qu'une personne non autorisée ne puisse retrouver ou modifier ce message.

- En *correction d'erreurs*, on cherche à coder un message de manière telle que l'on puisse repérer et corriger les erreurs survenus dans le canal de transmission.

- En *compression de données*, on cherche à réduire la taille des messages transmis ou stockés.

- Dans le *codage à contraintes spectrales*, le canal de transmission est un support phisique qui est modélisé par le type de contraintes que le message doit satisfaire pour être transmis. Le message émis par la source doit être transformé pour qu'il puisse vérifier ces contraintes. On appelle *codage à contrainte spectrales* ce type de codage. Un exemple est celui d'un canal qui ne supporte pas les alternances trop rapides de 0 et des 1 et qui ne sait pas faire la différence entre une longue suite de 1 et une longue suite de 0. On cherche, dans ce cas, un codage qui permette de transformer une suite de 0 et de 1 en une suite qui satisfait la contrainte de ne jamais contenir l'un des blocs suivants : $010101, 00000, 11111$.

L'ordre de composition de ces différents codages est important. Le codage de compression, par exemple, doit opérer dans un canal sans bruit. Comme, en général, les canaux de transmission sont bruités, un codage correcteurs d'erreurs opérant sur un canal bruité constitue un canal sans bruit.



2 Codes : définitions et exemples

Un *alphabet* est un ensemble fini $A = \{a_1, \ldots, a_n\}$ de symboles appelés lettres. On note A^* l'ensemble des mots sur l'alphabet A, y compris le *mot vide*, noté ε (le seul mot qui ne contient aucune lettre), et $A^+ = A^* \setminus \{\varepsilon\}$. La *longueur* d'un mot $u \in A^*$, notée $|u|$, est le nombre de lettres qui la composent, le mot vide étant le seul mot de longueur zéro.

Définition 2.1 *Code*
Un code C sur l'alphabet A est un ensemble de mots sur A qui ne contient pas le mot vide.

Les éléments d'un code C sont appelés *mots de code*.

Exemple 2.2 *Quelques exemples de code.*

- $C = \{0, 10, 110\}$ *est un code sur l'alphabet* $\{0, 1\}$.

- $C = \{ab, abcd, cdab\}$ *est un code sur l'aphabet* $A = \{a, b, c, d\}$

Un procédé simple de codage, appelé *codage par blocs à longuer variable*, ou plus simplement *codage à longueur variable*, consiste à substituer à chaque lettre du message émis par la source, un mot sur un alphabet différent de celui utilisé par la source. L'exemple le plus repandu est le code Morse, où l'alphabet de source est l'alphabet latin est celui de codage est composé de trois éléments : point, trait et blanc. Ce code est à déchiffrage immédiat, ce qui n'est pas le cas de tous les codes comme nous le verrons dans la suite.

Considérons une source qui émet des messages sur l'alphabet B et soit $C \subset A^+$ un code de même cardinal que B, $|C| = |B|$. Une fonction bijective $f : B \longrightarrow C$ est appelée *fonction d'encodage* de B dans C. L'ensemble B est appelé *alphabet source* et les mots que l'on peut obtenir avec les lettres de B *messages*. Le triplet (B, C, f) est appelé *schéma de codage*. Nous étendons la fonction d'encodage f à un morphisme $f : B^* \longrightarrow C^*$ de l'ensemble des mots sur l'alphabet B à l'ensemble des mots sur l'alphabet C. Cela permet d'envoyer un message $u = b_1 \dots b_n,\ b_1, \dots, b_n \in B^+$, émis par la source, en le traduisant sous la forme $f(u) = f(b_1 \dots b_n) = f(b_1) \dots f(b_n)$, comme le montre l'exemple suivant.

Exemple 2.3 *L'ordinateur ne travaille qu'avec des données numériques. Pour la représentation numérique des caractères d'un clavier, il existe un schéma de codage appelé ASCII (American Standard Code for Information Interchange). Dans le schéma de codage ASCII, l'alphabet de source* B *est l'ensemble des* $2^8 = 256$ *caractères (majuscules, minuscules, caractères spéciaux et de contrôle), le code* C *est composé par l'ensemble des mots de longueur huit sur l'alphabet binaire* $A = \{0, 1\}$ *et la fonction d'encodage est donné selon un schéma dont un extrait est donné ci-dessous.*

A	01000001	J	01001010	S	01010011
B	01000010	K	01001011	T	01010100
C	01000011	L	01001100	U	01010101
D	01000100	M	01001101	V	01010110
E	01000101	N	01001110	W	01010111
F	01000110	O	01001111	X	01011000
G	01000111	P	01010000	Y	01011001
H	01001000	Q	01010001	Z	01011010
I	01001001	R	01010010		00100000

Selon ce schéma le message SALUT est encodé par le mot :

$$0101001101000001010011000101010101010100$$

3 Déciffrabilité d'un code

Soit (B, C, f) un schéma d'encodage. Le fait que f soit bijective ne garantit pas que le décodage d'un message puisse se faire sans ambiguïté par le récepteur.

Exemple 3.1 *Considérons le schéma de codage des lettres de l'alphabet* $B = \{A, \ldots, Z\}$ *par les entiers* $C = \{0, \ldots, 25\}$ *donné par* $f(A) = 0, f(B) = 1, \ldots f(Z) = 25$. *Le mot reçu 1209 peut alors correspondre aux messages BCAJ, MAJ ou BUJ.*

Définition 3.2 *Code uniquement déchiffrable*
Un code $C \subseteq A^+$ *est* uniquement déchiffrable *ou* non ambigu *si et seulement si pour tout couple de suites de mots sur* C, $\{x_i\}_{i=0}^n$, $x_i \in C$, $\forall\ 0 \leq i \leq n$, *et* $\{y_j\}_{j=0}^m$, $y_j \in C$, $\forall\ 0 \leq j \leq m$, *vérifiant*

$$x_1 \ldots x_n = y_1 \ldots y_m$$

on a

$$\begin{cases} m & = & n \\ x_i & = & y_i, \quad \forall 1 \leq i \leq n. \end{cases}$$

Nous utiliserons l'adjectif *ambigu* pour un code qui n'est pas uniquement déchiffrable.

Exemple 3.3 *Codes ambigus et uniquement déchiffrables sur l'alphabet* $A = \{0, 1\}$.

- $C_1 = \{00, 01, 10, 11\}$ *est uniquement déchiffrable. En effet, comme les mots de* C_1 *ont tous la même longueur, il existe au plus une décomposition d'un mot de* A^+ *comme concatenation de mots de* C_1.

- $C_2 = \{0, 01, 10\}$ *est ambigu car le mot* 010 *admet deux décompositions différentes comme concatenation de mots de* C_2, $010 = (0)(10) = (01)(0)$.

- $C_3 = \{0, 10, 110\}$ *est uniquement déchiffrable car aucun mot de* C_3 *ne commence par un autre mot de* C_3.

4 Codes préfixes, suffixes, bipréfixes

Le dernier exemple de code uniquement déchiffrable fait partie d'une famille de codes très utilisés dans les applications, les codes préfixes, dont le code Morse fait partie. Nous définissons d'abord les notions de préfixe, suffixe et de facteur.

Définition 4.1 *Préfixes, suffixes et facteurs*
Soient $u, v \in A^*$. *Nous dirons que :*

- *Le mot* u *est un* préfixe *de* v *s'il existe* $x \in A^*$ *tel que* $v = ux$. *Si* $x \neq \varepsilon$, *nous dirons que* u *est un* préfixe propre *de* v.

- *Le mot* u *est un* suffixe *de* v *s'il existe* $x \in A^*$ *tel que* $v = xu$. *Si* $x \neq \varepsilon$, *nous dirons que* u *est un* suffixe propre *de* v.

- *Le mot* u *est un* facteur *de* v *s'ils existent* $x, y \in A^*$ *tel que* $v = xuy$. *Si* $x \neq \varepsilon$ *ou* $y \neq \varepsilon$, *nous dirons que* u *est un* facteur propre *de* v.

Exemple 4.2 *Sur l'alphabet* $A = \{a, b, \ldots, z\}$*, le mot* auto *est préfixe du mot* automate*, le mot* tomate *en est un suffixe et le mot* toma *un facteur.*

Définition 4.3 *Code préfixe, suffixe et bipréfixe*
Soit $C \subseteq A^+$*.*

- *Le code* C *est* préfixe *si aucun mot de* C *n'est préfixe propre d'aucun autre mot de* C*.*
- *Le code* C *est* suffixe *si aucun mot de* C *n'est suffixe propre d'aucun autre mot de* C*.*
- *Le code* C *est* bipréfixe *s'il est préfixe et suffixe à la fois.*

Exemple 4.4 *Des exemples sur l'alphabet* $\{a, b\}$

1. $\{ab, aab, ba, bbba\}$ *est préfixe mais pas suffixe.*
2. $\{ba, aba, bb\}$ *est suffixe mais pas préfixe.*
3. $\{ab, aa, aab\}$ *n'est ni préfixe ni suffixe.*
4. *Les codes uniformes, codes dont les éléments ont tous la même longueur, sont bipréfixe.*

Nous prouvons la propriété fondamentale des codes préfixes (en TD).

Proposition 4.5 *Tout code préfixe (suffixe) est uniquement déchiffrable.*

Preuve. Soit $C \subset A^+$ un code préfixe. Supposons, par l'absurde, qu'il soit ambigu. Alors l'ensemble D des mots ayant deux factorisations différentes comme produit de mots de C est non vide. Soit $w \in A^+$ un mot appartenant à D de longueur minimale. Le mot w admet deux factorisations différentes :

$$w = x_1 \ldots x_m = y_1 \ldots y_n,\ x_i, y_j \in C.$$

Les mots x_1 et y_1 sont différents du mot vide car C est un code et ils sont différents entre eux car w est un mot de D de longueur minimale.Donc, ou bien x_1 est un préfixe propre de y_1 ou bien y_1 est un préfixe propre de x_1, ce qui contredit l'hypothèse que C est préfixe.

■

5 L'algorithme de Sardinas et Pattersons

L'algorithme de Sardinas et Patterson, prouvé en 1950, permet de décider si un code fini $C \subset A^+$ est uniquement déchiffrable. On associe à C la suite $\{R_i\}_{i\geq 0}$, d'ensembles de mots sur A, définie par :

$$\begin{cases} R_0 &= C^{-1}C \setminus \{\varepsilon\} \\ R_{i+1} &= C^{-1}R_i \cup R_i^{-1}C, \quad \forall\, i \geq 0. \end{cases}$$

La suite $\{R_i\}_{i\geq 0}$ est appelée *suite des restes droits* associée à C.

Soient $X, Y \subseteq A^*$, le *reste droit* ou *quotient gauche* de X par Y est l'ensemble de mots de A^* défini par :

$$Y^{-1}X = \{w \in A^* \mid \exists y \in Y : yw \in X\}.$$

Exemple 5.1 *Constructions de quelques suites de restes droits*

- *Soit $C = \{a, ba, bb\}$. On a $R_0 = C^{-1}C \setminus \{\varepsilon\} = \emptyset$. Cela implique que $R_i = \emptyset$ pour tout $i \geq 0$. On remarque que, en général, si X est un ensemble préfixe alors la suite des restes droits associés à X est composée du seul ensemble vide.*

- *Soit $C = \{a, ab, bb\}$. On a $R_i = \{b\}$ pour tout $i \geq 0$.*

- *Soit $C = \{ab, abcd, cdab\}$. On a $R_0 = \{cd\}$, $R_1 = \{ab\}$, $R_2 = \{\varepsilon, cd\}$ et $R_3 = C$ et $R_4 = R_2$. La suite est donc périodique. On remarque que $R_1 \cap C = \{ab\}$. Nous allons démontrer que cela implique qu'il existe un mot de C^+ qui admet deux factorisations différentes comme produit de mots de C. En effet, $R_0 \cap C = \{ab\}$. On sait que $r_1 = ab \in R_0^{-1}C$. Donc, il existe un mot $r_0 \in R_0$ tel que $r_0 r_1 \in C$. Ce mot est $r_0 = cd$. Le mot r_0 appartient à $R_0 = C^{-1}C \setminus \{\varepsilon\}$. Cela implique qu'il existe un mots $c \in C$ tel que $cr_0 \in C$. Ce mot est $c = ab$. On obient, $(c)(r_0 r_1) = abcdab = (cr_0)(r_1) \in C^+$. Il existe donc un mot de C^+ qui admet deux factorisations différentes comme produit de mots de C.*

Exemple 5.2 *Le code $C\{0, 01, 010\}$ es ambigu car le mot $u = 010$ admet deux factorisations différentes comme produit de mots de C. Nous allons montrer qu'il existe un $i \geq 0$ tel que l'ième terme de la suite des restes droits associé à C est différent de l'ensemble vide, $R_i \neq \emptyset$. De $(0)(10) = (01)(0)$ on déduit qu'il existe un mot de C, $r_0 = 0$, qui est préfixe d'un autre mot de C, $c = 01$. Nous obtenons que $r_0 \in R_0$ et que $r_0^{-1}c = 1 = r_1 \in R_1$. D'où $r_1^{-1}c = r_0 \in R_2 \cap C$. Donc $R_2 \cap C \neq \emptyset$.*

Nous pouvons à présent énoncer le théorème de Sardinas et Patterson.

Théorème 5.3 *Soit $C \subset A^+$ un code fini. C est uniquement déchiffrable si et seulement s'il existe un entier $i \geq 0$ tel que $R_i \cap C \neq \emptyset$.*

On peut déduire du théorème de Sardinas e Pattersons un algorithme qui permet de tester si un code fini est uniquement déchiffrable dont la preuve est laissée au lecteur.

6 Mise en oeuvre de l'algorithme de Sardinas et Pattersons

Nous pouvons associer à un code C un graphe orientè et étiquetté, appelé *graphe préfixe* et noté $\mathcal{SP}(\mathcal{C}) = \langle S, A\rangle$. Il est défini par :

- $S =$*Pref*$(C) \setminus \{\varepsilon\}$.
- $A = A_c \cup A_{av}$ où :
  $A_c = \{(u, \varepsilon, v) \mid u, v \in S,\ uv \in C\}$. Un arc appartenant à A_c est dit *arc croisé*. L'étiquette d'un arc croisé est ε.
  $A_c = \{(u, c, v) \mid u, v \in S,\ c \in C,\ uc = v\}$. Un appartenant à A_{av} est dit *arc avant*. L'étiquette d'un arc avant est un mot de C.

Exemple 6.1 *On construit le graphe préfixe $\mathcal{SP}(C) = \langle S, A_c, A_{av}\rangle$ associé au code $C = \{a, bb, abbba, babab\}$.*

$$S = \{a, ab, abb, abbb, abbba, b, bb, ba, bab, baba\},$$

$$A_c = \{(b, \varepsilon, b), (abb, \varepsilon, ba), (abbb, \varepsilon, a), (bab, \varepsilon, ab), (baba, \varepsilon, b), (ba, \varepsilon, bab)\},$$

$$A_{av} = \{(a, bb, abb), (ab, bb, abbb), (abbb, a, abbba), (b, a, ba), (bab, a, baba)\}.$$

Définition 6.2 *Soit $C \subset A^+$ un code. Une C-*factorisation *d'un mot $v \in A^+$ est une factorisation de la forme*

$$v = sx_1 \dots x_k t$$

où $s, t \in$ Pref$(C), x_i \in C,\ \forall 1 \leq i \leq k$.

Définition 6.3 *Soit $C \subset A^+$ un code. Les C-factorisations $(s, x_1, \dots, x_n, t)$ et $(\overline{s}, y_1, \dots, y_m, \overline{t})$ d'un mot $v \in A^+$ sont* disjointes *si*

1. *$s \neq \overline{s}$;*
2. *$sx_1 \dots x_i \neq \overline{s}y_1 \dots x_j\ \forall 1 \leq i \leq n, \forall 1 \leq j \leq m$.*

On peut montrer le résultat suivant:

Lemme 6.4 *Il existe dans $\mathcal{SP}(C)$ un chemin de longueur $n \geq 1$ du sommet s au sommet t si et seulement ils existent $x_1, \dots, x_k, y_1, \dots, y_l \in C$ tels que :*

1. *$sy_1 \dots y_l t = x_1 \dots x_k$, ou bien*
2. *$sy_1 \dots y_l = x_1 \dots x_k t$*

où $n = k + l$ et les C-factorisations $(s, y_1, \dots, y_l, t)$ et $(x_1, \dots, x_k)$ et $(s, y_1, \dots, y_l,)$ et $(x_1, \dots, x_k, t)$ sont disjointes.

Exemple 6.5 *Considérons le graphe préfixe de l'Exemple 6.1.*
Le chemin

$$abb \xrightarrow{\varepsilon} ba \xrightarrow{\varepsilon} bab \xrightarrow{\varepsilon} ab \xrightarrow{bb} abbb$$

correspond à une égalité entre C-factorisations de type 1. On peut remarquer que, dans ce cas, le nombre d'arcs croisés est impair.

$$sy_1t = (abb)(babab)(abbb) = (abbba)(babab)(bb) = x_1x_2x_3.$$

avec $l = 1, k = 3, n = l + k = 4$.
Le chemin

$$abb \xrightarrow{\varepsilon} ba \xrightarrow{\varepsilon} bab \xrightarrow{a} baba$$

correspond à une égalité entre C-factorisations de type 2. On peut remarquer que, dans ce cas, le nombre d'arcs croisés est pair.

$$sy_1t = (abb)(babab)(a) = (abbba)(baba) = x_1t$$

avec $l = 2, k = 1, n = l + k = 3$.
Considérons maintenant le chemin

$$a \xrightarrow{bb} abb \xrightarrow{\varepsilon} ba \xrightarrow{\varepsilon} bab \xrightarrow{\varepsilon} ab \xrightarrow{bb} abbb \xrightarrow{\varepsilon} a.$$

Il correspond à l'égalité

$$sy_1y_2t = (a)(bb)(babab)(abbba) = (abbba)(babab)(bb)(a) = x_1x_2x_3x_4.$$

Comme $s, t \in C$, le code C est ambigu.

Comme corollaire du lemme précédent, on obtient que la condition d'unique déchiffrabilité d'un code C peut se tester sur le graphe préfixe associé.

Théorème 6.6 *Un code fini $C \subset A^+$ est ambigu si et seulement si le graphe $\mathcal{SP}(\mathcal{C})$ possède un chemin entre deux sommets dont les étiquettes appartiennent à C.*

En ce qui concerne la taille du graphe préfixe, nous avons le résultat suivant:

Proposition 6.7 *Soient $C \subset A^+$, $n =\mid X \mid$, $N = \sum_{x \in C} \mid x \mid$. Le graphe préfixe $\mathcal{SP}(\mathcal{C})$ possède au plus N sommets et $n \times N$ arcs*

Preuve. Les sommets de C sont tous les préfixes non vides des mots de C. Chaque mot de $x \in C$ engendre $\mid x \mid$ préfixes. Le nombre de somméts de $\mathcal{SP}(\mathcal{C})$ est au plus N. Pour évaluer le nombre d'arcs, considérons un sommet t de $\mathcal{SP}(\mathcal{C})$. Nous avons que si (s, ε, t) est un arc croisé entrant en t alots $\mid t \mid <\mid st \mid$, $st \in C$. Si (s, x, t) est un arc avant entrant en t alors $t = sx$ et $\mid t \mid <\mid x \mid$, $x \in C$.
Pour un sommet $t \in C$, on note C_t^+ l'ensemble des éléments de C dont la longueur est supérieure à celle de t et par C_t^- l'ensemble des éléments de C dont la longueur est inférieure à celle de t. Le nombre d'arcs croisés entrant en t sont au plus $\mid C_t^+ \mid$ et le nombre d'arcs avant entrant en t sont au plus $\mid C_t^- \mid$; au total, le nombre d'arcs entrant en t est au plus $\mid C_t^+ \mid + \mid C_t^- \mid \leq n$. Comme le nombre de sommets de $\mathcal{SP}(\mathcal{C})$ est N, le nombre d'arcs est inférieur ou égal à $n \times N$.

■

Toutes les propriétés de $\mathcal{SP}(\mathcal{C})$ peuvent être testées en temps linéaire par rapport à sa taille.

6.1 Construction de $\mathcal{SP}(\mathcal{C})$

La construction de $\mathcal{SP}(\mathcal{C})$ se fait en plusieurs étapes :

1. Contruction du *trie*, automate déterministe dont le graphe est connexe et sans cycles, $\mathcal{T}(C)$ qui reconnaît C : $\mathcal{T}(C) = \langle Q, A, \cdot, i, F\rangle$ où :

   - $Q =$*Pref*(C), $i = \varepsilon$, $F = Q \cap C$.
   - $\forall p \in Q$ et $\forall a \in A$: $p \cdot a = pa$ si $pa \in$*Pref*(C) et $p \cdot a$ n'est pas défini sinon.

2. Calcul de la *fontion de suppléance* sur les états de $\mathcal{T}(C)$,

   $$f : \textit{Pref}(C) \longrightarrow \textit{Pref}(C),$$

   définie par $f(p)$ est le plus long suffixe propre de $p \in$*Pref*(C) qui est aussi un préfixe d'un mot de C, si $p \neq \varepsilon$ et $f(\varepsilon) = \varepsilon$.

3. Construction de *l'automate de localisation (pattern matching)* $\mathcal{PM}(C)$ associé à $C \subset A^+$. Cet automate est est défini par : $\mathcal{PM}(C) = \langle Q, A, \star, i, F\rangle$ où :

   - $Q =$*Pref*(C), $i = \varepsilon$, $\overline{F} = F \cup \{p \in Q \mid f(p) \in \overline{F}\}$.
   - $\forall p \in Q$ et $\forall a \in A$: $p \star a = pa$ si $pa \in$*Pref*(C) et $p \star a = f(p)$ sinon.

4. Calcul de l'ensemble des arcs croisés par l'utilisation de la fonction de suppléance. Pour tout mot $x \in C$, un arc croisé est déterminé par un suffixe t de x qui est aussi un préfixe, $x = st$. Tous les suffixes qui sont aussi de préfixes sont énumérés par la fontion de suppléance. Nous obtenons les arcs croisés de la manière suivante :

   Procédure *Arcs-Croisés*$(C \subseteq A^+)$;
   Début
   $A_c \longleftarrow \emptyset$;
   Pour tout $x \in C$ **faire**
   Début
   $t \longleftarrow f(x)$;
   Tantque $t \neq \varepsilon$ **faire**
   Début
   $s \longleftarrow xt^{-1}$;
   $Ajouter(A_c, (s, \varepsilon, t))$;
   $t \longleftarrow f(t)$;
   Fintanque
   Finpour
   $Retourner(A_c)$
   Finprocédure

   La notation xt^{-1} signifie un mot s tel que $st = x$. Pour calculer de manière

efficace $s = xt^{-1}$, on associe à x un vecteur de pointeurs sur les préfixes de x. Ainsi, en connaissant la longueur de x et de t, on peut obtenir s en temps constant.

5. Calcul de l'ensemble des arcs avant par l'utilisation de la fonction de suppléance. Pour déterminer les arcs avant on utilise la fonction σ qui permet de retrouver le plus long suffixe propre d'un préfixe d'un mot de C qui appartienne à C. La fonction σ est définie seulement sur les états finaux de $\mathcal{PM}(C)$ par :

$$\sigma(x) = \begin{cases} f(x) = & \text{si } f(x) \in F \\ \sigma(f(x)) & \text{si } f(x) \in \overline{F} \setminus F \\ \varepsilon & \text{sinon} \end{cases}$$

Nous obtenons les arcs avant de la manière suivante :

Procédure *Arcs-avant*($C \subseteq A^+$);
Début
  $A_{av} \longleftarrow \emptyset$;
  Pour tout $t \in \overline{F}$ **faire**
    Début
      $x \longleftarrow \sigma(t)$;
      Tantque $(x \neq \varepsilon)$ **faire**
        Début
        $s \longleftarrow tx^{-1}$;
        $Ajouter(A_{av}, (s, x, t))$;
        $x \longleftarrow \sigma(x)$;
        Fintanque
    Finpour
  $Retourner(A_{av})$
Finprocédure

7 Codes à délai de déchiffrage fini

L'encodage par un code préfixe des messages émis par une source S permet de déchiffrer les messages reçus avant la fin de la reception du mot. Dans ce cas, nous dirons que le code est à délai de déchiffrage zéro.

Exemple 7.1 *Code à délai de déchiffrage zéro*

$$X = \{a, ba, bb\}$$

La décomposition d'un message encodé par X est déterminée dés la lecture d'un mot du code, car aucun mot du code est préfixe d'un autre mot di code.

Exemple 7.2 *Code à délai de déchiffrage $d = 1$*

$$Y = \{ab, baab, abb\}$$

Si l'on reçoit un message encodé par Y dont le début est $w = abb$, on ne peut pas savoir si la décomposition du début du message commence par ab ou abb. Dans le cas où la lettre suivante est a, le début de la décomposition de $wa = abba$ n'est pas encore déterminé. Si la lettre suivante est b la décomposition de $wb = abbb$ est par contre déterminée, elle commence par abb. La décomposition de $waa = abbaa =$ commence par ab et celle de $wab = abbab$ par abb.
Pour le code Y, il faut attendre l'apparition d'au moins deux mots du code pour déterminer la décomposition du début du message, car l'apparition d'un seul mots de C ne suffit pas. Nous dirons que Y est un code à délai de déchiffrage $d = 1$.

Exemple 7.3 *Code à délai de déchiffrage infini*

$$Z = \{aa, b, ba\}$$

Dans le début du message $w = b(aa)^n$, on a l'apparition de $n+1$ mots du code et le début de la décomposition de w n'est pas encore déterminé. On dit que Z est à délai de déchiffrage infini.

Définition 7.4 *Code à délai de dechiffrage fini*
Soit $C \subset A^+$ un code. C est à délai de déchiffrage fini *s'il existe un entier d tel que à chaque fois que $f = x_1 \dots x_{d+1}, x_i \in C,\ \forall 1 \leq i \leq d+1$, est préfixe d'un mot $g = y_1 \dots y_r, y_j \in C,\ \forall 1 \leq i \leq r$ alors $x_1 = y_1$.*

Autrement dit, C est un code à délai de déchiffrage fini d si le premier mot de la décomposition d'un message f composé d'au moins $d+1$ mots de C est déterminé de manière unique.

Il est clair que, si un code C est à délai de déchiffrage d, il est aussi à délai de déchiffrage $k \geq d$. Le délai de déchiffrage d'un code C est défini comme le plus petit entier d pour lequel le code est à délai de déchiffrage.

Nous provons maintenant la propriété fondamentale des codes à délai de déchiffrage fini.

Proposition 7.5 *Soit $C \subset A+$ un code à délai de dechiffrage fini d. Alors C est uniquement déchiffrable.*

Preuve. A faire

8 Décodage des codes à délai de déchiffrage fini

Nous proverons dans cette section que le décodage d'un message codé par un code à délai de déchiffrage fini peut être réalisé par automate particulier, appelé transducteur. Informellement, un transducteur est un automate qui réalise une relation sur $A^* \times B^*$.

Définition 8.1 *Un transducteur est un automate de la forme*

$$\mathcal{T} = (Q, A, B, I, T, E, i, t)$$

défini par :

- *Q est l'ensemble des états,*

- *A est l'alphabet d'entrée et B celui de sortie,*

- *$I \subset Q$ est l'ensemble des états initiaux,*

- *$T \subset Q$ l'ensemble des états finaux,*

- *$E \subset Q \times A^* \times B^* \times Q$ est l'ensemble des transitions,*

- *$i : I \longrightarrow B^*$ est la fonction d'entrée et $t : T \longrightarrow B^*$ celle de sortie.*

Si $e = (p, u, v, q), p, q \in Q,\ u \in A^*, v \in B^*$, est une transition, le mot u est l'*étiquette d'entrée* et v est l'*étiquette de sortie* de e. L'ensemble des transitions peut se définir aussi avec une fonction, dite *fonction de transition*, $\delta : Q \times A^* \longrightarrow B^* \times Q$. Pour la transition e, on aura $\delta(p, u) = (v, q)$.

Un transducteur est à *temps réel* si les étiquettes d'entrée des ses transitions appartiennent à A. On appelle *automate d'entrée* associé à $\mathcal{T}$, l'automate que l'on obtient à partir de $\mathcal{T}$ en effaçant les étiquettes de sortie et *automate de sortie* associé à $\mathcal{T}$, l'automate que l'on obtient à partir de $\mathcal{T}$ en effaçant les étiquettes d'entrée. Un transducteur est dit *séquentiel* s'il est à temps réel et si l'automate d'entrée est déterministe. Un automate séquentiel réalise une *fonction* sur $A^* \times B^*$, dite *fonction séquentielle*.

8.1 Décodage

Soit $\gamma : B^* \longrightarrow A^*$ un morphisme. On peut montrer que le code $X = \gamma(X)$ est uniquement déciffrable ssi γ est un morphisme injectif. On dit que γ est un *morphisme de codage* s'il est injectif.

Soit $\gamma : B^* \longrightarrow A^*$ un morphisme de codage. Comme γ est injectif, il existe une fonction partielle $\gamma^{-1} : A^* \longrightarrow B^*$, de domaine $X^* = \gamma(B)^*$, telle que

$$\gamma^{-1}(\gamma(u)) = u,\ \forall u \in B^*.$$

La fonction γ^{-1} est appelée *fonction de décodage* associée à γ.

Exercice 1 *Soit $B = \{b_1, b_2, b_3\}$ et $A = \{0, 1\}$. Construire un transducteur qui réalise le décodage associé au morphisme de codage $\gamma : B^* \longrightarrow A^*$ engendré par $\gamma(b_1) = 00,\ \gamma(b_2) = 1,\ \gamma(b_3) = 01$.*

Exercice 2 *Soit $B = \{b_1, b_2, b_3\}$ et $A = \{0, 1\}$. Construire un transducteur qui réalise le décodage associé au morphisme de codage $\gamma : B^* \longrightarrow A^*$ engendré par $\gamma(b_1) = 00,\ \gamma(b_2) = 10,\ \gamma(b_3) = 100$.*

L'automate d'entrée de l'Exemple 1 est déterministe. Celui de l'Exemple 2 n'est pas déterministe mais il vérifie une propriété remarquable, il est non ambigu.

Définition 8.2 *Un automate est* non ambigu *si deux chemins de même étiquette et de mêmes extrémités sont égaux.*

Un automate qui n'est pas non ambigu est dit *ambigu.*

Exercice 3 *Donner un exemple d'automate ambigu.*

8.2 Trasducteur préfixe

Soit $\gamma : B^* \longrightarrow A^*$, $X = \gamma(B)$, un morphisme de codage et soit $\gamma^{-1} : A^* \longrightarrow B^*$ la fonction de décodage associé à γ. Le *transducteur préfixe* associé à γ est le transducteur $\mathcal{T}_\gamma^p = (Q, A, B, I, T, E, i, t)$, qui est défini par :

- $Q =$*Pref*(X),
- $I = T = \{\varepsilon\}$,
- $i(\varepsilon) = t(\varepsilon) = \varepsilon$,
- $(p, a, \varepsilon, pa) \in E$ si $pa \in$*Pref*$(X) \setminus X$,
- $(p, a, b, \varepsilon) \in E$ si $pa \in X$ et $b = \gamma^{-1}(pa)$.

Le transducteur $\mathcal{T}_\gamma^p$ réalise la fonction de décodage associée à γ. En effet, l'automate d'entrée associé à $\mathcal{T}_\gamma$ reconnaît X^*. A chaque fois que un chemin de $\mathcal{T}_\gamma^p$ part de l'état initial revient à l'état initial, un mot de X^* est reconnu et son décodage est l'étiquette de sortie de ce chemin. On peut montrer la proposition suivante :

Proposition 8.3 *L'automate d'entrée du transducteur préfixe $\mathcal{T}_\gamma^p$, qui réalise la fonction de décodage associée au morphisme de codage $\gamma : B^* \longrightarrow A^*$, est non ambigu. Il est déterministe si $X = \gamma(B)$ est préfixe.*

Remarque 8.4 *Pour un code $X \subset A^+$ qui n'est pas uniquement déchiffrable, l'automate d'entrée est non déterministe.*

Exemple 8.5 *Soit $B = \{b_1, b_2, b_3\}$ et $A = \{a, b\}$. Construire un transducteur de décodage pour le morphisme de codage $\gamma : B^* \longrightarrow A^*$, engendré par $\gamma(b_1) = a$; $\gamma(b_2) = ab$, $\gamma(b_3) = ba$.*

9 Transducteur de décodage pour les codes à délai de déchiffrage fini

Soit $\gamma : B^* \longrightarrow A^*$, $X = \gamma(B)$, un morphisme de codage et soit X un code à délai de déchiffrage fini. On peut démontrer la proposition suivante:

Proposition 9.1 *Soit $\gamma : B^* \longrightarrow A^*$ un morphisme de codage. L'ensemble $X = \gamma(B)$ est un code à délai de déchiffrage fini si et seulement si γ^{-1} est une fonction séquentielle.*

Preuve. La preuve de cette proposition consiste en la construction d'un trasducteur qui réalise le décodage associé à γ. Ce transducteur, dit transducteur à délai, est $\mathcal{T}_\gamma^d = (Q, A, B, I, T, E, i, t)$, il est défini par :

- $Q = Pref(X^{d+1})$,
- $I\{\varepsilon\}$, $i(\varepsilon) = \varepsilon$,
- $T = Q \cap X^{\leq d}$, pour $(x_{i_1} \cdots x_{i_k}) \in X^{\leq d}$ on a $t(x_{i_1} \cdots x_{i_k}) = b_{i_1} \cdots b_{i_k}, x_{i_j} \in X$ et $\gamma(b_{i_j}) = x_{i_j}$, $\forall 1 \leq j \leq k$,
- $(p, a, \varepsilon, pa) \in E$ si $pa \in Pref(X^{d+1}) \setminus X^{d+1}$,
- $(p, a, b_{i_1}, x_{i_2} \cdots x_{i_{d+1}}) \in E$ si $pa = x_{i_1} x_{i_2} \cdots x_{i_{d+1}}, x_{i_k} \in X$, $1 \leq k \leq d+1$ et $\gamma(b_{i_1}) = x_{i_1}$.

Remarque 9.2 *Si le code $X = \gamma(B)$ est préfixe, c'est à dire à délai de déchiffrage $d = 0$, le transducteur construit par la méthode de la preuve de la Proposition 9.1 est le transducteur préfixe.*

10 Décodage bidirectionnel : la méthode de Girod

L'encodage par un code préfixe permet le décodage sans délai des messages reçus. On a vu que ce décodage peut être réalisé par un transducteur séquentiel en lisant le message de la gauche vers la droite. Pour limiter la propagation des erreurs, dans le cas d'un canal bruité, il est utile de pouvoir décoder les messages reçus aussi de la droite vers la gauche. Bernd Girod a élaboré en 1999 une méthode qui permet la construction d'un transducteur de décodage bidirectionnel, dans le cas d'un code préfixe. Pour expliquer cette méthode, on considère :

- Un alphabet source $B = \{b_1, \ldots, b_n\}$.
- L'alphabet de codage $A = \{0, 1\}$.
- Un code préfixe $X = \{x_1, \ldots, x_n\}$, $x_i \in A^+$, $\forall 1 \leq i \leq n$.
- Le morphisme de codage $\gamma : B^* \longrightarrow A^*$, défini par $\gamma(b_i) = x_i$, $\forall 1 \leq i \leq n$.
- La somme binaire ou ou-exclusif sur A,

$\oplus$	0	1
0	0	1
1	1	0

Remarque 10.1 *Une propriété remarquable de la somme binaire, qui sera utilisée dans la suite, est la suivante :*

$$a \oplus b = c \longrightarrow a = c \oplus b,\ b = c \oplus a.$$

Soit $v \in B^+$, et soit $y = \gamma(v) = x_{i_1} \ldots x_{i_k}, x_{i_j} \in B$, $\forall 1 \leq j \leq k$, on note y' le mot $y' = \widetilde{x}_{i_1} \ldots \widetilde{x}_{i_k}$ où $\widetilde{x}_{i_j}$ est l'image miroir de x_{i_j} pour tout $1 \leq j \leq k$.

Exemple 10.2 *Soit $B = \{b_1, b_2\}$ et soit $A = \{0, 1\}$. On considère le morphisme de codage défini par $x_1 = \gamma(b_1) = 11, x_2 = \gamma(b_2) = 011$. On associe au mot $y = \gamma(b_1 b_2 b_2) = \gamma(b_1)\gamma(b_2)\gamma(b_2) = 11011011$ le mot $y' = 11110110$.*

Soit $L = Maximum\{|x| \mid x \in X\}$ et soit $x_L \in X^+$ un mot de longueur L. On appelle x_L la *clé* du décodage. On associe à tout mot $y = \gamma(v) = x_{i_1} \dots x_{i_k}, x_{i_j} \in B$, $\forall 1 \leq j \leq k$, le mot $z = yx_L \oplus \widetilde{x}_L y'$, que l'on appelle le codage de Girod de v.

Exemple 10.3 *Dans le codage de l'Exemple 10.2, $L = 3$. Si l'on choisit comme clé $x_L = 011$, le codage de Girod du mot $y = 11011011$ est $z = 11011011011 \oplus 11011110110 = 00000101101$*

Le problème que l'on veut résoudre est le suivant : Étant donnés $z \in A^+$ et x_L, trouver $v \in B^+$ tel que $z = \gamma(v)x_L \oplus \widetilde{x}\gamma(v)'$.

Exemple 10.4 *On considère le codage de l'Exemple 10.2. On applique la méthode de décodage de Girod pour décoder le mot $z = 00000101101$, de l'Exemple 10.3, en utilisant la clé $x_L = 011$.*
On commence par faire la somme entre la clé et le préfixe de longueur $L = 3$ de z. C'est à dire, $v_1 = 000 \oplus 110 = 110$.

z	$=$	$000 \mid 00101101$
	$\oplus$	110
v_1	$=$	110

Comme la longueur de v_1 est $L = 3$, un préfixe de v_1 appartient à X et comme X est préfixe, il n'y a que un seul mot de X qui est préfixe de v_1. Ce préfixe est $x_1 = 11$. Le début du décodage de z est $\gamma^{-1}(x_1) = b_1$. On note u_1 le préfixe de longueur L de z. On recommence le décodage en faisant la somme entre le préfixe de longueur L de $z_1 = (u_1)^{-1}z$ et de $(x_1)^{-1}v_1\widetilde{x}_1$.

z_1	$=$	$000 \mid 101101$
	$\oplus$	011
v_2	$=$	011

Le mot v_2 contient $x_2 = 011$ comme préfixe. Le début du décodage de z_1 est $\gamma^{-1}(x_2) = b_2$. Donc, le début du décodage de z est b_1b_2. On note u_2 le préfixe de longueur L de z_1.On recommence le décodage en faisant la somme entre le préfixe de longueur L de $z_2 = (u_2)^{-1}z_1$ et de $\widetilde{x}_2$.

z_2	$=$	$101 \mid 101$
	$\oplus$	110
v_3	$=$	011

Le mot v_3 contient $x_2 = 011$ comme préfixe. Le début du décodage de z_1 est $\gamma^{-1}(x_2) = b_2$. Donc, le début du décodage de z est b_1b_2. Donc, le début du décodage de z est $b_1b_2b_2$. On note u_3 le préfixe de longueur L de z_2. La longueur du mot qui reste à décoder est L. Le décodage du mot y, dont le mot z est le codage de Girod, est terminé. En effet, pour obtenir le codage de Girod de z, nous avons ajouté L lettres. Le décodage de z est $b_1b_2b_2$.

Exercice 4 *La méthode de décodage de Girod peut s'appliquer en lisant z de la droite vers la gauche en utlisant x_L à la place de $\widetilde{x}_L$ et $\widetilde{X}$ à la place de X. Appliquez la méthode de Girod de la droite vers a gauche au mot z avec la clé x_L de l'exemple précédent.*