# Cours 5. Compression des données. Introduction au codage de canal

A. Désilles

4 mai 2009

# Résumé

1. Rappels
   - Compression des données
   - Algorithmes statistiques.
2. Méthodes à dictionnaire
   - Méthode LZW
   - Quelques remarques sur les codes à dictionnaire
3. Codage de canal
   - Modèle probabiliste de commuication
   - Règle de décodage d'un canal avec bruit.
   - Analyse de l'erreur de transmission
   - Notion de code de canal.
   - Second théorème de Shannon

# Résumé

1. Rappels
   - Compression des données
   - Algorithmes statistiques.
2. Méthodes à dictionnaire
   - Méthode LZW
   - Quelques remarques sur les codes à dictionnaire
3. Codage de canal
   - Modèle probabiliste de commuication
   - Règle de décodage d'un canal avec bruit.
   - Analyse de l'erreur de transmission
   - Notion de code de canal.
   - Second théorème de Shannon

# Résumé

1. Rappels
   - Compression des données
   - Algorithmes statistiques.
2. Méthodes à dictionnaire
   - Méthode LZW
   - Quelques remarques sur les codes à dictionnaire
3. Codage de canal
   - Modèle probabiliste de commuication
   - Règle de décodage d'un canal avec bruit.
   - Analyse de l'erreur de transmission
   - Notion de code de canal.
   - Second théorème de Shannon

# Résumé

# Compression de données : position de problème

- Soit un texte $T$ composé de symboles d'un alphabet $\Omega = \{s_1, \ldots, s_n\}$.
- Soit $M$ le nombre de symboles dans $T$.
- Codage à longuer fixe, $R$ : la longueur totale de texte codé est $l(T) = RM$.
- On cherche alors un code binaire pour le texte $T$ tel que la longueur de message codé $l(U)$ soit inférieure à $l(T)$.
- On appelle taux de compression la quantité

$$t = \frac{l(U)}{l(T)}$$

# Compression de données : position de problème

- Soit un texte $T$ composé de symboles d'un alphabet $\Omega = \{s_1, \ldots, s_n\}$.
- Soit $M$ le nombre de symboles dans $T$.
- Codage à longuer fixe, $R$ : la longueur totale de texte codé est $l(T) = RM$.
- On cherche alors un code binaire pour le texte $T$ tel que la longueur de message codé $l(U)$ soit inférieure à $l(T)$.
- On appelle taux de compression la quantité

$$t = \frac{l(U)}{l(T)}$$

# Compression de données : position de problème

- Soit un texte $T$ composé de symboles d'un alphabet $\Omega = \{s_1, \ldots, s_n\}$.
- Soit $M$ le nombre de symboles dans $T$.
- Codage à longuer fixe, $R$ : la longueur totale de texte codé est $l(T) = RM$.
- On cherche alors un code binaire pour le texte $T$ tel que la longueur de message codé $l(U)$ soit inférieure à $l(T)$.
- On appelle taux de compression la quantité

$$t = \frac{l(U)}{l(T)}$$

# Compression de données : position de problème

- Soit un texte $T$ composé de symboles d'un alphabet $\Omega = \{s_1, \ldots, s_n\}$.
- Soit $M$ le nombre de symboles dans $T$.
- Codage à longuer fixe, $R$ : la longueur totale de texte codé est $l(T) = RM$.
- On cherche alors un code binaire pour le texte $T$ tel que la longueur de message codé $l(U)$ soit inférieure à $l(T)$.
- On appelle taux de compression la quantité

$$t = \frac{l(U)}{l(T)}$$

# Compression de données : position de problème

- Soit un texte $T$ composé de symboles d'un alphabet $\Omega = \{s_1, \ldots, s_n\}$.
- Soit $M$ le nombre de symboles dans $T$.
- Codage à longuer fixe, $R$ : la longueur totale de texte codé est $l(T) = RM$.
- On cherche alors un code binaire pour le texte $T$ tel que la longueur de message codé $l(U)$ soit inférieure à $l(T)$.
- On appelle taux de compression la quantité

$$t = \frac{l(U)}{l(T)}$$

# Deux approches principales

**Compression sans pertes.** Il s'agit de méthodes inversibles, garantissant que le message initial peut être restauré intégralement à partir du compressé. Dans ce groupe de méthodes on retrouve en particulier, le format zip utilisé pour l'archivage et le format d'images GIF, par exemple. Le taux de compression est limité par l'entropie des données.

**Compression avec pertes.** Il s'agit de techniques basées sur les approximations des données initiales. Lors de la restauration le message obtenu n'est pas exactement le message initial. On retrouve dans ce groupe les formats Jpeg, MP3, MPEG.

# Deux approches principales

**Compression sans pertes.** Il s'agit de méthodes inversibles, garantissant que le message initial peut être restauré intégralement à partir du compressé. Dans ce groupe de méthodes on retrouve en particulier, le format zip utilisé pour l'archivage et le format d'images GIF, par exemple. Le taux de compression est limité par l'entropie des données.

**Compression avec pertes.** Il s'agit de techniques basées sur les approximations des données initiales. Lors de la restauration le message obtenu n'est pas exactement le message initial. On retrouve dans ce groupe les formats Jpeg, MP3, MPEG.

# Compression sans pertes : deux familles d'algorithmes

# Construction de la distribution de probabilités

- pour tout symbole $s_i$ de l'alphabet $\Omega$ soit $m_i$ le nombre d'occurrences de ce symbole dans le texte $T$.
- On peut alors définir $f_i = \frac{m_i}{M}$ la fréquence du symbole $s_i$.
- On a alors

$$\forall i = 1, \ldots, n, \quad 0 \leq f_i \leq 1, \quad \sum_{i=1}^{n} f_i = 1$$

- Ainsi, l'ensemble $\{f_i, \ i = 1, \ldots, n\}$ est une distribution de probabilité sur $\Omega$.

# Construction de la distribution de probabilités

- pour tout symbole $s_i$ de l'alphabet $\Omega$ soit $m_i$ le nombre d'occurrences de ce symbole dans le texte $T$.
- On peut alors définir $f_i = \dfrac{m_i}{M}$ la fréquence du symbole $s_i$.
- On a alors

$$\forall i = 1, \ldots, n, \quad 0 \leq f_i \leq 1, \quad \sum_{i=1}^{n} f_i = 1$$

- Ainsi, l'ensemble $\{f_i, \ i = 1, \ldots, n\}$ est une distribution de probabilité sur $\Omega$.

# Construction de la distribution de probabilités

- pour tout symbole $s_i$ de l'alphabet $\Omega$ soit $m_i$ le nombre d'occurrences de ce symbole dans le texte $T$.
- On peut alors définir $f_i = \frac{m_i}{M}$ la fréquence du symbole $s_i$.
- On a alors

$$\forall i = 1, \ldots, n, \quad 0 \leq f_i \leq 1, \quad \sum_{i=1}^{n} f_i = 1$$

- Ainsi, l'ensemble $\{f_i, \ i = 1, \ldots, n\}$ est une distribution de probabilité sur $\Omega$.

# Construction de la distribution de probabilités

- pour tout symbole $s_i$ de l'alphabet $\Omega$ soit $m_i$ le nombre d'occurrences de ce symbole dans le texte $T$.
- On peut alors définir $f_i = \frac{m_i}{M}$ la fréquence du symbole $s_i$.
- On a alors

$$\forall i = 1, \ldots, n, \quad 0 \leq f_i \leq 1, \quad \sum_{i=1}^{n} f_i = 1$$

- Ainsi, l'ensemble $\{f_i, \ i = 1, \ldots, n\}$ est une distribution de probabilité sur $\Omega$.

# Compression par codage de Huffman : Avantages

1. Compression sans pertes
2. Méthodes applicables à tout type de données numériques (images, son, texte)
3. L'arbre de code n'occupe pas beaucoup de mémoire

# Compression par codage de Huffman : Avantages

1. Compression sans pertes
2. Méthodes applicables à tout type de données numériques (images, son, texte)
3. L'arbre de code n'occupe pas beaucoup de mémoire

# Compression par codage de Huffman : Avantages

1. Compression sans pertes
2. Méthodes applicables à tout type de données numériques (images, son, texte)
3. L'arbre de code n'occupe pas beaucoup de mémoire

# Compression par codage de Huffman : Faiblesses

1. Le taux de compression est variable : il dépend des données elles mêmes (la redondance)
2. Il est nécessaire de construire d'abord l'arbre avant de pourvoir coder
3. Inutilisable en temps réel
4. Il est nécessaire de transmettre l'arbre de code avec les données pour décoder

# Compression par codage de Huffman : Faiblesses

1. Le taux de compression est variable : il dépend des données elles mêmes (la redondance)
2. Il est nécessaire de construire d'abord l'arbre avant de pourvoir coder
3. Inutilisable en temps réel
4. Il est nécessaire de transmettre l'arbre de code avec les données pour décoder

# Compression par codage de Huffman : Faiblesses

1. Le taux de compression est variable : il dépend des données elles mêmes (la redondance)
2. Il est nécessaire de construire d'abord l'arbre avant de pourvoir coder
3. Inutilisable en temps réel
4. Il est nécessaire de transmettre l'arbre de code avec les données pour décoder

# Compression par codage de Huffman : Faiblesses

1. Le taux de compression est variable : il dépend des données elles mêmes (la redondance)
2. Il est nécessaire de construire d'abord l'arbre avant de pourvoir coder
3. Inutilisable en temps réel
4. Il est nécessaire de transmettre l'arbre de code avec les données pour décoder

# Principe

- Le principe commun consiste à encoder des séquences de caractères par les références à leurs emplacements dans un dictionnaire.
- Le dictionnaire est construit à partir du texte lui même
- Il contient toutes les séquences déjà rencontrées

# Principe

- Le principe commun consiste à encoder des séquences de caractères par les références à leurs emplacements dans un dictionnaire.
- Le dictionnaire est construit à partir du texte lui même
- Il contient toutes les séquences déjà rencontrées

# Principe

- Le principe commun consiste à encoder des séquences de caractères par les références à leurs emplacements dans un dictionnaire.
- Le dictionnaire est construit à partir du texte lui même
- Il contient toutes les séquences déjà rencontrées

# LZ78. Méthode

- Le dictionnaire sera construit au fur et à mesure de la lecture du texte à compresser.
- A tout instant le dictionnaire représente la partie du texte déjà lu, découpée en séquences numérotées.
- Le numéro 0 est réservé à la chaîne de caractères vide.
- Chaque séquence codée sera remplacée par un couple : $(i, c)$.
- $i$ est le numéro d'entrée dans le dictionnaire du préfixe composé des $n-1$ premiers caractères.
- $c$ est le dernier caractère de la séquence codée.
- On parcourt le texte restant à compresser à partir du caractère courent tant que la séquence lue existe dans le dictionnaire. On s'arrête dès qu'une séquence nouvelle, non encore enregistrée dans le dictionnaire, est trouvée. Ce procédé garantit que la nouvelle séquence $s$ est de la forme

# LZ78. Méthode

- Le dictionnaire sera construit au fur et à mesure de la lecture du texte à compresser.
- A tout instant le dictionnaire représente la partie du texte déjà lu, découpée en séquences numérotées.
- Le numéro 0 est réservé à la chaîne de caractères vide.
- Chaque séquence codée sera remplacée par un couple : $(i, c)$.
- $i$ est le numéro d'entrée dans le dictionnaire du préfixe composé des $n-1$ premiers caractères.
- $c$ est le dernier caractère de la séquence codée.
- On parcourt le texte restant à compresser à partir du caractère courent tant que la séquence lue existe dans le dictionnaire. On s'arrête dès qu'une séquence nouvelle, non encore enregistrée dans le dictionnaire, est trouvée. Ce procédé garantit que la nouvelle séquence $s$ est de la forme

# LZ78. Méthode

- Le dictionnaire sera construit au fur et à mesure de la lecture du texte à compresser.
- A tout instant le dictionnaire représente la partie du texte déjà lu, découpée en séquences numérotées.
- Le numéro 0 est réservé à la chaîne de caractères vide.
- Chaque séquence codée sera remplacée par un couple : $(i, c)$.
- $i$ est le numéro d'entrée dans le dictionnaire du préfixe composé des $n-1$ premiers caractères.
- $c$ est le dernier caractère de la séquence codée.
- On parcourt le texte restant à compresser à partir du caractère courent tant que la séquence lue existe dans le dictionnaire. On s'arrête dès qu'une séquence nouvelle, non encore enregistrée dans le dictionnaire, est trouvée. Ce procédé garantit que la nouvelle séquence $s$ est de la forme

# LZ78. Méthode

- Le dictionnaire sera construit au fur et à mesure de la lecture du texte à compresser.
- A tout instant le dictionnaire représente la partie du texte déjà lu, découpée en séquences numérotées.
- Le numéro 0 est réservé à la chaîne de caractères vide.
- Chaque séquence codée sera remplacée par un couple : $(i, c)$.
- $i$ est le numéro d'entrée dans le dictionnaire du préfixe composé des $n-1$ premiers caractères.
- $c$ est le dernier caractère de la séquence codée.
- On parcourt le texte restant à compresser à partir du caractère courent tant que la séquence lue existe dans le dictionnaire. On s'arrête dès qu'une séquence nouvelle, non encore enregistrée dans le dictionnaire, est trouvée. Ce procédé garantit que la nouvelle séquence $s$ est de la forme

# LZ78. Méthode

- Le dictionnaire sera construit au fur et à mesure de la lecture du texte à compresser.
- A tout instant le dictionnaire représente la partie du texte déjà lu, découpée en séquences numérotées.
- Le numéro 0 est réservé à la chaîne de caractères vide.
- Chaque séquence codée sera remplacée par un couple : $(i, c)$.
- $i$ est le numéro d'entrée dans le dictionnaire du préfixe composé des $n-1$ premiers caractères.
- $c$ est le dernier caractère de la séquence codée.
- On parcourt le texte restant à compresser à partir du caractère courent tant que la séquence lue existe dans le dictionnaire. On s'arrête dès qu'une séquence nouvelle, non encore enregistrée dans le dictionnaire, est trouvée. Ce procédé garantit que la nouvelle séquence $s$ est de la forme

# LZ78. Méthode

- Le dictionnaire sera construit au fur et à mesure de la lecture du texte à compresser.
- A tout instant le dictionnaire représente la partie du texte déjà lu, découpée en séquences numérotées.
- Le numéro 0 est réservé à la chaîne de caractères vide.
- Chaque séquence codée sera remplacée par un couple : $(i, c)$.
- $i$ est le numéro d'entrée dans le dictionnaire du préfixe composé des $n-1$ premiers caractères.
- $c$ est le dernier caractère de la séquence codée.
- On parcourt le texte restant à compresser à partir du caractère courent tant que la séquence lue existe dans le dictionnaire. On s'arrête dès qu'une séquence nouvelle, non encore enregistrée dans le dictionnaire, est trouvée. Ce procédé garantit que la nouvelle séquence $s$ est de la forme

# LZ78. Méthode

- Le dictionnaire sera construit au fur et à mesure de la lecture du texte à compresser.
- A tout instant le dictionnaire représente la partie du texte déjà lu, découpée en séquences numérotées.
- Le numéro 0 est réservé à la chaîne de caractères vide.
- Chaque séquence codée sera remplacée par un couple : $(i, c)$.
- $i$ est le numéro d'entrée dans le dictionnaire du préfixe composé des $n-1$ premiers caractères.
- $c$ est le dernier caractère de la séquence codée.
- On parcourt le texte restant à compresser à partir du caractère courent tant que la séquence lue existe dans le dictionnaire. On s'arrête dès qu'une séquence nouvelle, non encore enregistrée dans le dictionnaire, est trouvée. Ce procédé garantit que la nouvelle séquence $s$ est de la forme

# LZ78. Exemple

ELLE EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

- On initialise le dictionnaire avec la séquence vide à l'emplacement 0.
- On lit le caractère "E". Cette séquence n'est pas présente dans le dictionnaire. On le remplace par le couple $(0,'E')$ et enregistre dans le dictionnaire la séquence "E" à l'adresse $i = 1$
- Le caractère lu : "L". Cette séquence n'est pas dans le dictionnaire. On la remplace par le couple $(0,'L')$ et on enregistre "L" dans le dictionnaire à l'emplacement $i = 2$.
- Caractère lu : "L". Séquence présente dans le dictionnaire à l'adresse $i = 2$.
- On lit le caractère suivant. La séquence en attente devient : "LE". Elle n'est pas dans le dictionnaire. On la remplace par le couple $(2, E)$ et on l'enregistre dans le dictionnaire à l'adresse $i = 3$.

# LZ78. Exemple

ELLE EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

- On initialise le dictionnaire avec la séquence vide à l'emplacement 0.
- On lit le caractère "E". Cette séquence n'est pas présente dans le dictionnaire. On le remplace par le couple $(0,'E')$ et enregistre dans le dictionnaire la séquence "E" à l'adresse $i = 1$
- Le caractère lu : "L". Cette séquence n'est pas dans le dictionnaire. On la remplace par le couple $(0,'L')$ et on enregistre "L" dans le dictionnaire à l'emplacement $i = 2$.
- Caractère lu : "L". Séquence présente dans le dictionnaire à l'adresse $i = 2$.
- On lit le caractère suivant. La séquence en attente devient : "LE". Elle n'est pas dans le dictionnaire. On la remplace par le couple $(2, E)$ et on l'enregistre dans le dictionnaire à l'adresse $i = 3$.

# LZ78. Exemple

ELLE EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

- On initialise le dictionnaire avec la séquence vide à l'emplacement 0.
- On lit le caractère "E". Cette séquence n'est pas présente dans le dictionnaire. On le remplace par le couple $(0,'E')$ et enregistre dans le dictionnaire la séquence "E" à l'adresse $i = 1$
- Le caractère lu : "L". Cette séquence n'est pas dans le dictionnaire. On la remplace par le couple $(0,'L')$ et on enregistre "L" dans le dictionnaire à l'emplacement $i = 2$.
- Caractère lu : "L". Séquence présente dans le dictionnaire à l'adresse $i = 2$.
- On lit le caractère suivant. La séquence en attente devient : "LE". Elle n'est pas dans le dictionnaire. On la remplace par le couple $(2, E)$ et on l'enregistre dans le dictionnaire à l'adresse $i = 3$.

# LZ78. Exemple

ELLE EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

- On initialise le dictionnaire avec la séquence vide à l'emplacement 0.
- On lit le caractère "E". Cette séquence n'est pas présente dans le dictionnaire. On le remplace par le couple $(0,'E')$ et enregistre dans le dictionnaire la séquence "E" à l'adresse $i = 1$
- Le caractère lu : "L". Cette séquence n'est pas dans le dictionnaire. On la remplace par le couple $(0,'L')$ et on enregistre "L" dans le dictionnaire à l'emplacement $i = 2$.
- Caractère lu : "L". Séquence présente dans le dictionnaire à l'adresse $i = 2$.
- On lit le caractère suivant. La séquence en attente devient : "LE". Elle n'est pas dans le dictionnaire. On la remplace par le couple $(2, E)$ et on l'enregistre dans le dictionnaire à l'adresse $i = 3$.

# LZ78. Exemple

ELLE EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

- On initialise le dictionnaire avec la séquence vide à l'emplacement 0.
- On lit le caractère "E". Cette séquence n'est pas présente dans le dictionnaire. On le remplace par le couple $(0,'E')$ et enregistre dans le dictionnaire la séquence "E" à l'adresse $i = 1$
- Le caractère lu : "L". Cette séquence n'est pas dans le dictionnaire. On la remplace par le couple $(0,'L')$ et on enregistre "L" dans le dictionnaire à l'emplacement $i = 2$.
- Caractère lu : "L". Séquence présente dans le dictionnaire à l'adresse $i = 2$.
- On lit le caractère suivant. La séquence en attente devient : "LE". Elle n'est pas dans le dictionnaire. On la remplace par le couple $(2, E)$ et on l'enregistre dans le dictionnaire à l'adresse $i = 3$.

# LZ78. Exemple

ELLE␣EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

| indice | Dictionnaire | Code | indice | Dictionnaire | Code |
|---|---|---|---|---|---|
| 0 | null | | 13 | ␣E | (4,E) |
| | | | 14 | CH | (10,H) |
| | | | 15 | ELL | (8,L) |
| | | | 16 | E␣ | (1,␣) |
| | | | 17 | ETE | (11,E) |
| | | | 18 | R | (0,R) |
| 6 | T | (0,T) | 19 | N | (0,N) |
| 7 | ␣B | (4,B) | 20 | ELLE | (15,E) |
| 8 | EL | (1,L) | 21 | . | (0,.) |
| 9 | LE␣ | (3,␣) | 22 | ␣ES | (13,S) |
| 10 | C | (0,C) | 23 | T␣ | (6,␣) |
| 11 | ET | (1,T) | 24 | RE | (18,E) |

# LZ78. Exemple

ELLE␣EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

| indice | Dictionnaire | Code | indice | Dictionnaire | Code |
|---|---|---|---|---|---|
| 0 | null | | 13 | ␣E | (4,E) |
| 1 | E | (0,E) | 14 | CH | (10,H) |
| | | | 15 | ELL | (8,L) |
| | | | 16 | E␣ | (1,␣) |
| | | | 17 | ETE | (11,E) |
| | | | 18 | R | (0,R) |
| 6 | T | (0,T) | 19 | N | (0,N) |
| 7 | ␣B | (4,B) | 20 | ELLE | (15,E) |
| 8 | EL | (1,L) | 21 | . | (0,.) |
| 9 | LE␣ | (3,␣) | 22 | ␣ES | (13,S) |
| 10 | C | (0,C) | 23 | T␣ | (6,␣) |
| 11 | ET | (1,T) | 24 | RE | (18,E) |

# LZ78. Exemple

ELLE␣EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

| indice | Dictionnaire | Code | indice | Dictionnaire | Code |
|---|---|---|---|---|---|
| 0 | null | | 13 | ␣E | (4,E) |
| 1 | E | (0,E) | 14 | CH | (10,H) |
| 2 | L | (0,L) | 15 | ELL | (8,L) |
| | | | 16 | E␣ | (1,␣) |
| | | | 17 | ETE | (11,E) |
| | | | 18 | R | (0,R) |
| 6 | T | (0,T) | 19 | N | (0,N) |
| 7 | ␣B | (4,B) | 20 | ELLE | (15,E) |
| 8 | EL | (1,L) | 21 | . | (0,.) |
| 9 | LE␣ | (3,␣) | 22 | ␣ES | (13,S) |
| 10 | C | (0,C) | 23 | T␣ | (6,␣) |
| 11 | ET | (1,T) | 24 | RE | (18,E) |

# LZ78. Exemple

ELLE␣EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

| indice | Dictionnaire | Code | indice | Dictionnaire | Code |
|---|---|---|---|---|---|
| 0 | null | | 13 | ␣E | (4,E) |
| 1 | E | (0,E) | 14 | CH | (10,H) |
| 2 | L | (0,L) | 15 | ELL | (8,L) |
| 3 | LE | (2,E) | 16 | E␣ | (1,␣) |
| | | | 17 | ETE | (11,E) |
| | | | 18 | R | (0,R) |
| 6 | T | (0,T) | 19 | N | (0,N) |
| 7 | ␣B | (4,B) | 20 | ELLE | (15,E) |
| 8 | EL | (1,L) | 21 | . | (0,.) |
| 9 | LE␣ | (3,␣) | 22 | ␣ES | (13,S) |
| 10 | C | (0,C) | 23 | T␣ | (6,␣) |
| 11 | ET | (1,T) | 24 | RE | (18,E) |

# LZ78. Exemple

ELLE␣EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

| indice | Dictionnaire | Code | indice | Dictionnaire | Code |
|---|---|---|---|---|---|
| 0 | null | | 13 | ␣E | (4,E) |
| 1 | E | (0,E) | 14 | CH | (10,H) |
| 2 | L | (0,L) | 15 | ELL | (8,L) |
| 3 | LE | (2,E) | 16 | E␣ | (1,␣) |
| 4 | ␣ | (0,␣) | 17 | ETE | (11,E) |
| | | | 18 | R | (0,R) |
| 6 | T | (0,T) | 19 | N | (0,N) |
| 7 | ␣B | (4,B) | 20 | ELLE | (15,E) |
| 8 | EL | (1,L) | 21 | . | (0,.) |
| 9 | LE␣ | (3,␣) | 22 | ␣ES | (13,S) |
| 10 | C | (0,C) | 23 | T␣ | (6,␣) |
| 11 | ET | (1,T) | 24 | RE | (18,E) |

# LZ78. Exemple

ELLE␣EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

| indice | Dictionnaire | Code | indice | Dictionnaire | Code |
|---|---|---|---|---|---|
| 0 | null | | 13 | ␣E | (4,E) |
| 1 | E | (0,E) | 14 | CH | (10,H) |
| 2 | L | (0,L) | 15 | ELL | (8,L) |
| 3 | LE | (2,E) | 16 | E␣ | (1,␣) |
| 4 | ␣ | (0,␣) | 17 | ETE | (11,E) |
| 5 | ES | (1,S) | 18 | R | (0,R) |
| 6 | T | (0,T) | 19 | N | (0,N) |
| 7 | ␣B | (4,B) | 20 | ELLE | (15,E) |
| 8 | EL | (1,L) | 21 | . | (0,.) |
| 9 | LE␣ | (3,␣) | 22 | ␣ES | (13,S) |
| 10 | C | (0,C) | 23 | T␣ | (6,␣) |
| 11 | ET | (1,T) | 24 | RE | (18,E) |

# LZ78. Exemple

ELLE␣EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE.

| indice | Dictionnaire | Code | indice | Dictionnaire | Code |
|---|---|---|---|---|---|
| 0 | null | | 13 | ␣E | (4,E) |
| 1 | E | (0,E) | 14 | CH | (10,H) |
| 2 | L | (0,L) | 15 | ELL | (8,L) |
| 3 | LE | (2,E) | 16 | E␣ | (1,␣) |
| 4 | ␣ | (0,␣) | 17 | ETE | (11,E) |
| 5 | ES | (1,S) | 18 | R | (0,R) |
| 6 | T | (0,T) | 19 | N | (0,N) |
| 7 | ␣$B$ | (4, $B$) | 20 | ELLE | (15,E) |
| 8 | EL | (1,L) | 21 | . | (0,.) |
| 9 | LE␣ | (3, ␣) | 22 | ␣ES | (13,S) |
| 10 | C | (0,C) | 23 | T␣ | (6,␣) |
| 11 | ET | (1,T) | 24 | RE | (18,E) |

# LZ78 : Exemple

Le code définitif obtenu sera

$$(0E, 0L, 2E, 0\sqcup, 1S, 0T, 4B, 1L, 3\sqcup, 0C, 1T, 6E, 4E, 10H, 8L, 1\sqcup, 11E, 0R, 0N,$$

On obtient un code de 50 caractères au lieu de 56.

# LZ78 : Exemple

Le code définitif obtenu sera

$(0E, 0L, 2E, 0\sqcup, 1S, 0T, 4B, 1L, 3\sqcup, 0C, 1T, 6E, 4E, 10H, 8L, 1\sqcup, 11E, 0R, 0N,$

On obtient un code de 50 caractères au lieu de 56.

# Décodage

A chaque couple de code lu le décodeur va en effet effectuer deux actions :

1. Restituer la séquence qu'il représente en concaténant le mot dictionnaire indiqué par l'adresse du couple et le caractère contenu dans le couple ;
2. Enregistrer la nouvelle séquence dans le dictionnaire.

# Décodage

A chaque couple de code lu le décodeur va en effet effectuer deux actions :

1. Restituer la séquence qu'il représente en concaténant le mot dictionnaire indiqué par l'adresse du couple et le caractère contenu dans le couple ;
2. Enregistrer la nouvelle séquence dans le dictionnaire.

# Décodage

Voici le décodage du code de notre premier exemple :

$(0E, 0L, 2E, 0\sqcup, 1S, 0T, 4B, 1L, 3\sqcup, 0C, 1T, 6E, 4E, 10H, 8L, 1\sqcup, 11E, 0R, 0N,$

Le dictionnaire est initialisé par la chaîne de caractères vide à l'adresse 0.

- Couple $(0, E)$. Séquence décodée : 'E'. Adresse dans le dictionnaire : $i = 1$. Texte='E'
- Couple $(0, L)$. Séquence décodée : 'L'. Adresse dans le dictionnaire : $i = 2$. Texte='EL'
- Couple $(2, E)$. Séquence décodée : 'LE'. Adresse dans le dictionnaire : $i = 3$. Texte='ELLE'
- Couple $(0, \sqcup)$. Séquence décodée : '$\sqcup$'. Adresse dans le dictionnaire : $i = 4$. Texte='ELLE '
- Couple $(1, S)$. Séquence décodée : 'ES'. Adresse dans le dictionnaire : $i = 5$. Texte='ELLE ES'

# Décodage

Voici le décodage du code de notre premier exemple :

$(0E, 0L, 2E, 0\sqcup, 1S, 0T, 4B, 1L, 3\sqcup, 0C, 1T, 6E, 4E, 10H, 8L, 1\sqcup, 11E, 0R, 0N,$

Le dictionnaire est initialisé par la chaîne de caractères vide à l'adresse 0.

- Couple $(0, E)$. Séquence décodée : 'E'. Adresse dans le dictionnaire : $i = 1$. Texte='E'
- Couple $(0, L)$. Séquence décodée : 'L'. Adresse dans le dictionnaire : $i = 2$. Texte='EL'
- Couple $(2, E)$. Séquence décodée : 'LE'. Adresse dans le dictionnaire : $i = 3$. Texte='ELLE'
- Couple $(0, \sqcup)$. Séquence décodée : '$\sqcup$'. Adresse dans le dictionnaire : $i = 4$. Texte='ELLE '
- Couple $(1, S)$. Séquence décodée : 'ES'. Adresse dans le dictionnaire : $i = 5$. Texte='ELLE ES'

# Décodage

Voici le décodage du code de notre premier exemple :

$(0E, 0L, 2E, 0\sqcup, 1S, 0T, 4B, 1L, 3\sqcup, 0C, 1T, 6E, 4E, 10H, 8L, 1\sqcup, 11E, 0R, 0N,$

Le dictionnaire est initialisé par la chaîne de caractères vide à l'adresse 0.

- Couple $(0, E)$. Séquence décodée : 'E'. Adresse dans le dictionnaire : $i = 1$. Texte='E'
- Couple $(0, L)$. Séquence décodée : 'L'. Adresse dans le dictionnaire : $i = 2$. Texte='EL'
- Couple $(2, E)$. Séquence décodée : 'LE'. Adresse dans le dictionnaire : $i = 3$. Texte='ELLE'
- Couple $(0, \sqcup)$. Séquence décodée : '$\sqcup$'. Adresse dans le dictionnaire : $i = 4$. Texte='ELLE '
- Couple $(1, S)$. Séquence décodée : 'ES'. Adresse dans le dictionnaire : $i = 5$. Texte='ELLE ES'

# Décodage

Voici le décodage du code de notre premier exemple :

$$(0E, 0L, 2E, 0\sqcup, 1S, 0T, 4B, 1L, 3\sqcup, 0C, 1T, 6E, 4E, 10H, 8L, 1\sqcup, 11E, 0R, 0N,$$

Le dictionnaire est initialisé par la chaîne de caractères vide à l'adresse 0.

- Couple $(0, E)$. Séquence décodée : 'E'. Adresse dans le dictionnaire : $i = 1$. Texte='E'
- Couple $(0, L)$. Séquence décodée : 'L'. Adresse dans le dictionnaire : $i = 2$. Texte='EL'
- Couple $(2, E)$. Séquence décodée : 'LE'. Adresse dans le dictionnaire : $i = 3$. Texte='ELLE'
- Couple $(0, \sqcup)$. Séquence décodée : '$\sqcup$'. Adresse dans le dictionnaire : $i = 4$. Texte='ELLE '
- Couple $(1, S)$. Séquence décodée : 'ES'. Adresse dans le dictionnaire : $i = 5$. Texte='ELLE ES'

# Décodage

Voici le décodage du code de notre premier exemple :

$$(0E, 0L, 2E, 0\sqcup, 1S, 0T, 4B, 1L, 3\sqcup, 0C, 1T, 6E, 4E, 10H, 8L, 1\sqcup, 11E, 0R, 0N,$$

Le dictionnaire est initialisé par la chaîne de caractères vide à l'adresse 0.

- Couple $(0, E)$. Séquence décodée : 'E'. Adresse dans le dictionnaire : $i = 1$. Texte='E'
- Couple $(0, L)$. Séquence décodée : 'L'. Adresse dans le dictionnaire : $i = 2$. Texte='EL'
- Couple $(2, E)$. Séquence décodée : 'LE'. Adresse dans le dictionnaire : $i = 3$. Texte='ELLE'
- Couple $(0, \sqcup)$. Séquence décodée : '$\sqcup$'. Adresse dans le dictionnaire : $i = 4$. Texte='ELLE '
- Couple $(1, S)$. Séquence décodée : 'ES'. Adresse dans le dictionnaire : $i = 5$. Texte='ELLE ES'

# Décodage

- Couple $(0, T)$. Séquence décodée : 'T'. Adresse dans le dictionnaire : $i = 6$. Texte='ELLE EST'
- Couple $(4, B)$. Séquence décodée : '␣B'. Adresse dans le dictionnaire : $i = 7$. Texte='ELLE EST B'
- Couple $(1, L)$. Séquence décodée : 'EL'. Adresse dans le dictionnaire : $i = 8$. Texte='ELLE EST BEL'
- Couple $(3, ␣)$. Séquence décodée : 'LE '. Adresse dans le dictionnaire : $i = 9$. Texte='ELLE EST BELLE '
- Couple $(0, C)$. Séquence décodée : 'C'. Adresse dans le dictionnaire : $i = 10$. Texte='ELLE EST BELLE C'
- Couple $(1, T)$. Séquence décodée : 'ET'. Adresse dans le dictionnaire : $i = 11$. Texte='ELLE EST BELLE CET'
- etc

# Décodage

- Couple $(0, T)$. Séquence décodée : 'T'. Adresse dans le dictionnaire : $i = 6$. Texte='ELLE EST'
- Couple $(4, B)$. Séquence décodée : '␣B'. Adresse dans le dictionnaire : $i = 7$. Texte='ELLE EST B'
- Couple $(1, L)$. Séquence décodée : 'EL'. Adresse dans le dictionnaire : $i = 8$. Texte='ELLE EST BEL'
- Couple $(3, \sqcup)$. Séquence décodée : 'LE '. Adresse dans le dictionnaire : $i = 9$. Texte='ELLE EST BELLE '
- Couple $(0, C)$. Séquence décodée : 'C'. Adresse dans le dictionnaire : $i = 10$. Texte='ELLE EST BELLE C'
- Couple $(1, T)$. Séquence décodée : 'ET'. Adresse dans le dictionnaire : $i = 11$. Texte='ELLE EST BELLE CET'
- etc

# Décodage

- Couple $(0, T)$. Séquence décodée : 'T'. Adresse dans le dictionnaire : $i = 6$. Texte='ELLE EST'
- Couple $(4, B)$. Séquence décodée : '␣B'. Adresse dans le dictionnaire : $i = 7$. Texte='ELLE EST B'
- Couple $(1, L)$. Séquence décodée : 'EL'. Adresse dans le dictionnaire : $i = 8$. Texte='ELLE EST BEL'
- Couple $(3, ␣)$. Séquence décodée : 'LE '. Adresse dans le dictionnaire : $i = 9$. Texte='ELLE EST BELLE '
- Couple $(0, C)$. Séquence décodée : 'C'. Adresse dans le dictionnaire : $i = 10$. Texte='ELLE EST BELLE C'
- Couple $(1, T)$. Séquence décodée : 'ET'. Adresse dans le dictionnaire : $i = 11$. Texte='ELLE EST BELLE CET'
- etc

# Décodage

- Couple $(0, T)$. Séquence décodée : 'T'. Adresse dans le dictionnaire : $i = 6$. Texte='ELLE EST'
- Couple $(4, B)$. Séquence décodée : '␣B'. Adresse dans le dictionnaire : $i = 7$. Texte='ELLE EST B'
- Couple $(1, L)$. Séquence décodée : 'EL'. Adresse dans le dictionnaire : $i = 8$. Texte='ELLE EST BEL'
- Couple $(3, ␣)$. Séquence décodée : 'LE '. Adresse dans le dictionnaire : $i = 9$. Texte='ELLE EST BELLE '
- Couple $(0, C)$. Séquence décodée : 'C'. Adresse dans le dictionnaire : $i = 10$. Texte='ELLE EST BELLE C'
- Couple $(1, T)$. Séquence décodée : 'ET'. Adresse dans le dictionnaire : $i = 11$. Texte='ELLE EST BELLE CET'
- etc

# Décodage

- Couple $(0, T)$. Séquence décodée : 'T'. Adresse dans le dictionnaire : $i = 6$. Texte='ELLE EST'
- Couple $(4, B)$. Séquence décodée : '␣B'. Adresse dans le dictionnaire : $i = 7$. Texte='ELLE EST B'
- Couple $(1, L)$. Séquence décodée : 'EL'. Adresse dans le dictionnaire : $i = 8$. Texte='ELLE EST BEL'
- Couple $(3, ␣)$. Séquence décodée : 'LE '. Adresse dans le dictionnaire : $i = 9$. Texte='ELLE EST BELLE '
- Couple $(0, C)$. Séquence décodée : 'C'. Adresse dans le dictionnaire : $i = 10$. Texte='ELLE EST BELLE C'
- Couple $(1, T)$. Séquence décodée : 'ET'. Adresse dans le dictionnaire : $i = 11$. Texte='ELLE EST BELLE CET'
- etc

# Décodage

- Couple $(0, T)$. Séquence décodée : 'T'. Adresse dans le dictionnaire : $i = 6$. Texte='ELLE EST'
- Couple $(4, B)$. Séquence décodée : '␣B'. Adresse dans le dictionnaire : $i = 7$. Texte='ELLE EST B'
- Couple $(1, L)$. Séquence décodée : 'EL'. Adresse dans le dictionnaire : $i = 8$. Texte='ELLE EST BEL'
- Couple $(3, ␣)$. Séquence décodée : 'LE '. Adresse dans le dictionnaire : $i = 9$. Texte='ELLE EST BELLE '
- Couple $(0, C)$. Séquence décodée : 'C'. Adresse dans le dictionnaire : $i = 10$. Texte='ELLE EST BELLE C'
- Couple $(1, T)$. Séquence décodée : 'ET'. Adresse dans le dictionnaire : $i = 11$. Texte='ELLE EST BELLE CET'
- etc

# Décodage

- Couple $(0, T)$. Séquence décodée : 'T'. Adresse dans le dictionnaire : $i = 6$. Texte='ELLE EST'
- Couple $(4, B)$. Séquence décodée : '␣B'. Adresse dans le dictionnaire : $i = 7$. Texte='ELLE EST B'
- Couple $(1, L)$. Séquence décodée : 'EL'. Adresse dans le dictionnaire : $i = 8$. Texte='ELLE EST BEL'
- Couple $(3, ␣)$. Séquence décodée : 'LE '. Adresse dans le dictionnaire : $i = 9$. Texte='ELLE EST BELLE '
- Couple $(0, C)$. Séquence décodée : 'C'. Adresse dans le dictionnaire : $i = 10$. Texte='ELLE EST BELLE C'
- Couple $(1, T)$. Séquence décodée : 'ET'. Adresse dans le dictionnaire : $i = 11$. Texte='ELLE EST BELLE CET'
- etc

# LZW : principe

- L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv.
- Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout.
- On lit le texte "caractère par caractère" jusqu'à ce qu'on rencontre une séquence qui n'est pas dans le disctionnaire
- Chaque nouvelle séquence est forcément de la forme $(m_i, c)$ où $m_i$ est un mot du dictionnaire à la position $i$ et $c$ est un caractère
- la nouvelle séquence est ajoutée au dictionnaire et sera utilisée pour le codage la prochaine fois qu'elle sera rencontrée
- On remplace alors $d_i$ par $i$, son adresse dans le dictionnaire
- et on reprend la lecture à partir du caractère $c$

# LZW : principe

- L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv.
- Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout.
- On lit le texte "caractère par caractère" jusqu'à ce qu'on rencontre une séquence qui n'est pas dans le disctionnaire
- Chaque nouvelle séquence est forcément de la forme $(m_i, c)$ où $m_i$ est un mot du dictionnaire à la position $i$ et $c$ est un caractère
- la nouvelle séquence est ajoutée au dictionnaire et sera utilisée pour le codage la prochaine fois qu'elle sera rencontrée
- On remplace alors $d_i$ par $i$, son adresse dans le dictionnaire
- et on reprend la lecture à partir du caractère $c$

# LZW : principe

- L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv.
- Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout.
- On lit le texte "caractère par caractère" jusqu'à ce qu'on rencontre une séquence qui n'est pas dans le disctionnaire
- Chaque nouvelle séquence est forcément de la forme $(m_i, c)$ où $m_i$ est un mot du dictionnaire à la position $i$ et $c$ est un caractère
- la nouvelle séquence est ajoutée au dictionnaire et sera utilisée pour le codage la prochaine fois qu'elle sera rencontrée
- On remplace alors $d_i$ par $i$, son adresse dans le dictionnaire
- et on reprend la lecture à partir du caractère $c$

# LZW : principe

- L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv.
- Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout.
- On lit le texte "caractère par caractère" jusqu'à ce qu'on rencontre une séquence qui n'est pas dans le disctionnaire
- Chaque nouvelle séquence est forcément de la forme $(m_i, c)$ où $m_i$ est un mot du dictionnaire à la position $i$ et $c$ est un caractère
- la nouvelle séquence est ajoutée au dictionnaire et sera utilisée pour le codage la prochaine fois qu'elle sera rencontrée
- On remplace alors $d_i$ par $i$, son adresse dans le dictionnaire
- et on reprend la lecture à partir du caractère $c$

# LZW : principe

- L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv.
- Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout.
- On lit le texte "caractère par caractère" jusqu'à ce qu'on rencontre une séquence qui n'est pas dans le disctionnaire
- Chaque nouvelle séquence est forcément de la forme $(m_i, c)$ où $m_i$ est un mot du dictionnaire à la position $i$ et $c$ est un caractère
- la nouvelle séquence est ajoutée au dictionnaire et sera utilisée pour le codage la prochaine fois qu'elle sera rencontrée
- On remplace alors $d_i$ par $i$, son adresse dans le dictionnaire
- et on reprend la lecture à partir du caractère $c$

# LZW : principe

- L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv.
- Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout.
- On lit le texte "caractère par caractère" jusqu'à ce qu'on rencontre une séquence qui n'est pas dans le disctionnaire
- Chaque nouvelle séquence est forcément de la forme $(m_i, c)$ où $m_i$ est un mot du dictionnaire à la position $i$ et $c$ est un caractère
- la nouvelle séquence est ajoutée au dictionnaire et sera utilisée pour le codage la prochaine fois qu'elle sera rencontrée
- On remplace alors $d_i$ par $i$, son adresse dans le dictionnaire
- et on reprend la lecture à partir du caractère $c$

# LZW : principe

- L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv.
- Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout.
- On lit le texte "caractère par caractère" jusqu'à ce qu'on rencontre une séquence qui n'est pas dans le disctionnaire
- Chaque nouvelle séquence est forcément de la forme $(m_i, c)$ où $m_i$ est un mot du dictionnaire à la position $i$ et $c$ est un caractère
- la nouvelle séquence est ajoutée au dictionnaire et sera utilisée pour le codage la prochaine fois qu'elle sera rencontrée
- On remplace alors $d_i$ par $i$, son adresse dans le dictionnaire
- et on reprend la lecture à partir du caractère $c$

# Remarques. Taille de dictionnaire

- En théorie, l'algorithme fonctionne avec un dictionnaire illimité.
- Dans la pratique, il est toujours de taille fixe.
- Si le dictionnaire est plein plusieurs choix sont possibles
- certaines méthodes permettent de doubler la taille du dictionnaire
- d'autres, en effacent une partie, la plus ancienne

# Remarques. Taille de dictionnaire

- En théorie, l'algorithme fonctionne avec un dictionnaire illimité.
- Dans la pratique, il est toujours de taille fixe.
- Si le dictionnaire est plein plusieurs choix sont possibles
- certaines méthodes permettent de doubler la taille du dictionnaire
- d'autres, en effacent une partie, la plus ancienne

# Remarques. Taille de dictionnaire

- En théorie, l'algorithme fonctionne avec un dictionnaire illimité.
- Dans la pratique, il est toujours de taille fixe.
- Si le dictionnaire est plein plusieurs choix sont possibles
- certaines méthodes permettent de doubler la taille du dictionnaire
- d'autres, en effacent une partie, la plus ancienne

# Remarques. Taille de dictionnaire

- En théorie, l'algorithme fonctionne avec un dictionnaire illimité.
- Dans la pratique, il est toujours de taille fixe.
- Si le dictionnaire est plein plusieurs choix sont possibles
- certaines méthodes permettent de doubler la taille du dictionnaire
- d'autres, en effacent une partie, la plus ancienne

# Remarques. Taille de dictionnaire

- En théorie, l'algorithme fonctionne avec un dictionnaire illimité.
- Dans la pratique, il est toujours de taille fixe.
- Si le dictionnaire est plein plusieurs choix sont possibles
- certaines méthodes permettent de doubler la taille du dictionnaire
- d'autres, en effacent une partie, la plus ancienne

# Remarques. Taux de compression

- Comme pour la méthode de Huffman, ce taux n'est pa fixe et ne peut être connu à l'avance
- Il dépend de la qualité intrinsèque des données
- Il peut être amélioré en utilisant une technique de codage statistique (Huffman, par exemple) sur les adresses.
- On exploite alors aussi les fréquences d'occurrence des séquences de taille variable.

# Remarques. Taux de compression

- Comme pour la méthode de Huffman, ce taux n'est pa fixe et ne peut être connu à l'avance
- Il dépend de la qualité intrinsèque des données
- Il peut être amélioré en utilisant une technique de codage statistique (Huffman, par exemple) sur les adresses.
- On exploite alors aussi les fréquences d'occurrence des séquences de taille variable.

# Remarques. Taux de compression

- Comme pour la méthode de Huffman, ce taux n'est pa fixe et ne peut être connu à l'avance
- Il dépend de la qualité intrinsèque des données
- Il peut être amélioré en utilisant une technique de codage statistique (Huffman, par exemple) sur les adresses.
- On exploite alors aussi les fréquences d'occurrence des séquences de taille variable.

# Remarques. Taux de compression

- Comme pour la méthode de Huffman, ce taux n'est pa fixe et ne peut être connu à l'avance
- Il dépend de la qualité intrinsèque des données
- Il peut être amélioré en utilisant une technique de codage statistique (Huffman, par exemple) sur les adresses.
- On exploite alors aussi les fréquences d'occurrence des séquences de taille variable.

# Remarques. Applications des codes à dictionnaire

- **La commande compress** sous UNIX applique l'algorithme LZW avec une taille initiale du dictionnaire de 512 mots ( codés sur 9 bits). Si le dictionnaire est rempli, on double sa taille ( codage sur 10 bits). Il est possible de préciser la taille maximale de dictionnaire. Lorsque l'algorithme l'atteint, il poursuit le codage de façon statique, sans modifier le dictionnaire.
- **GZIP, PKZIP, WINZIP** utilisent une variante de l'algorithme LZ77
- **Formats d'images GIF, PNG** utilisent également une variante de codage par dictionnaire.

# Remarques. Applications des codes à dictionnaire

- **La commande compress** sous UNIX applique l'algorithme LZW avec une taille initiale du dictionnaire de 512 mots ( codés sur 9 bits). Si le dictionnaire est rempli, on double sa taille ( codage sur 10 bits). Il est possible de préciser la taille maximale de dictionnaire. Lorsque l'algorithme l'atteint, il poursuit le codage de façon statique, sans modifier le dictionnaire.
- **GZIP, PKZIP, WINZIP** utilisent une variante de l'algorithme LZ77
- **Formats d'images GIF, PNG** utilisent également une variante de codage par dictionnaire.

# Remarques. Applications des codes à dictionnaire

- **La commande compress** sous UNIX applique l'algorithme LZW avec une taille initiale du dictionnaire de 512 mots ( codés sur 9 bits). Si le dictionnaire est rempli, on double sa taille ( codage sur 10 bits). Il est possible de préciser la taille maximale de dictionnaire. Lorsque l'algorithme l'atteint, il poursuit le codage de façon statique, sans modifier le dictionnaire.
- **GZIP, PKZIP, WINZIP** utilisent une variante de l'algorithme LZ77
- **Formats d'images GIF, PNG** utilisent également une variante de codage par dictionnaire.

# Modèle complet d'un canal de communication

Un canal de transmission **stationnaire et sans mémoire** peut être modélisé par le triplet $(X, Y, P(Y|X))$ où

- $X$ est la variable aléatoire de la source
- $Y$ est la variable aléatoire du récepteur
- $P(Y|X)$ est appelée matrice de transition et est définie par

$$p_{i|j} = P[y_j|\, x_i],\ i = 1, \ldots, n,\ j = 1, \ldots, m$$

Cette matrice décrit les propriétés du bruit dans le canal.

- Le terme "**sans mémoire**" signifie chaque symbole reçu est indépendant des symboles reçus précédemment.
- Le terme "**stationnaire**" signifie que les caractéristiques probabilistes du bruit sont indépendantes du temps.

# Modèle complet d'un canal de communication

Un canal de transmission **stationnaire et sans mémoire** peut être modélisé par le triplet $(X, Y, P(Y|X))$ où

- $X$ est la variable aléatoire de la source
- $Y$ est la variable aléatoire du récepteur
- $P(Y|X)$ est appelée matrice de transition et est définie par

$$p_{i|j} = P[y_j|\, x_i],\ i = 1, \ldots, n,\ j = 1, \ldots, m$$

Cette matrice décrit les propriétés du bruit dans le canal.

- Le terme "**sans mémoire**" signifie chaque symbole reçu est indépendant des symboles reçus précédemment.
- Le terme "**stationnaire**" signifie que les caractéristiques probabilistes du bruit sont indépendantes du temps.

# Modèle complet d'un canal de communication

Un canal de transmission **stationnaire et sans mémoire** peut être modélisé par le triplet $(X, Y, P(Y|X))$ où

- $X$ est la variable aléatoire de la source
- $Y$ est la variable aléatoire du récepteur
- $P(Y|X)$ est appelée matrice de transition et est définie par

$$p_{i|j} = P[y_j|\, x_i],\ i = 1, \ldots, n,\ j = 1, \ldots, m$$

Cette matrice décrit les propriétés du bruit dans le canal.

- Le terme **"sans mémoire"** signifie chaque symbole reçu est indépendant des symboles reçus précédemment.
- Le terme **"stationnaire"** signifie que les caractéristiques probabilistes du bruit sont indépendantes du temps.

# Modèle complet d'un canal de communication

Un canal de transmission **stationnaire et sans mémoire** peut être modélisé par le triplet $(X, Y, P(Y|X))$ où

- $X$ est la variable aléatoire de la source
- $Y$ est la variable aléatoire du récepteur
- $P(Y|X)$ est appelée matrice de transition et est définie par

$$p_{i|j} = P[y_j|\, x_i],\ i = 1, \ldots, n,\ j = 1, \ldots, m$$

Cette matrice décrit les propriétés du bruit dans le canal.

- Le terme **"sans mémoire"** signifie chaque symbole reçu est indépendant des symboles reçus précédemment.
- Le terme **"stationnaire"** signifie que les caractéristiques probabilistes du bruit sont indépendantes du temps.

# Modèle complet d'un canal de communication

Un canal de transmission **stationnaire et sans mémoire** peut être modélisé par le triplet $(X, Y, P(Y|X))$ où

- $X$ est la variable aléatoire de la source
- $Y$ est la variable aléatoire du récepteur
- $P(Y|X)$ est appelée matrice de transition et est définie par

$$p_{i|j} = P[y_j|\, x_i],\ i = 1, \ldots, n,\ j = 1, \ldots, m$$

Cette matrice décrit les propriétés du bruit dans le canal.

- Le terme **"sans mémoire"** signifie chaque symbole reçu est indépendant des symboles reçus précédemment.
- Le terme **"stationnaire"** signifie que les caractéristiques probabilistes du bruit sont indépendantes du temps.

# Capacité de canal

## Information mutuelle moyenne

$$I(X;Y) = H(X) - H(X|Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(i,j)\log\left(\frac{P(i,j)}{P(x_i)P(y_j)}\right).$$

$I(X;Y)$ représente la quantité d'information transportée par le canal.

## Capacité de canal

$$C = \max_{P(X)} I(X;Y) = \max_{P(X)}(H(X) - H(X|Y))$$

Le maximum est pris sur toutes les distributions de probabilité possibles de la source.

# Capacité de canal

## Information mutuelle moyenne

$$I(X;Y) = H(X) - H(X|Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(i,j)\log\left(\frac{P(i,j)}{P(x_i)P(y_j)}\right).$$

$I(X;Y)$ représente la quantité d'information transportée par le canal.

## Capacité de canal

$$C = \max_{P(X)} I(X;Y) = \max_{P(X)}(H(X) - H(X|Y))$$

Le maximum est pris sur toutes les distributions de probabilité possibles de la source.

# Capacité de canal

## Information mutuelle moyenne

$$I(X;Y) = H(X) - H(X|Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(i,j)\log\left(\frac{P(i,j)}{P(x_i)P(y_j)}\right).$$

$I(X;Y)$ représente la quantité d'information transportée par le canal.

## Capacité de canal

$$C = \max_{P(X)} I(X;Y) = \max_{P(X)}(H(X) - H(X|Y))$$

Le maximum est pris sur toutes les distributions de probabilité possibles de la source.

# Exemple : capacité d'un canal sans bruit

Soit un canal sans bruit et une source d'alphabet $\Omega = \{x_i,\ i = 1, \ldots, n\}$.

Dans ce cas $I(X;Y) = H(X) - H(X|Y) = H(X)$.

Alors

$$C = \max_{P(X)} I(X;Y) = \max_{P(X)} H(X) = \log n.$$

# Exemple : capacité d'un canal sans bruit

Soit un canal sans bruit et une source d'alphabet $\Omega = \{x_i,\ i = 1, \ldots, n\}$.
Dans ce cas $I(X;Y) = H(X) - H(X|Y) = H(X)$.

Alors

$$C = \max_{P(X)} I(X;Y) = \max_{P(X)} H(X) = \log n.$$

# Exemple : capacité d'un canal sans bruit

Soit un canal sans bruit et une source d'alphabet $\Omega = \{x_i,\ i = 1, \ldots, n\}$.
Dans ce cas $I(X;Y) = H(X) - H(X|Y) = H(X)$.
Alors

$$C = \max_{P(X)} I(X;Y) = \max_{P(X)} H(X) = \log n.$$

# Exemple : capacité d'un canal sans bruit

Soit un canal sans bruit et une source d'alphabet $\Omega = \{x_i,\ i = 1, \ldots, n\}$.
Dans ce cas $I(X;Y) = H(X) - H(X|Y) = H(X)$.
Alors

$$C = \max_{P(X)} I(X;Y) = \max_{P(X)} H(X) = \log n.$$

## Problème de codage avec bruit

Étant donné un canal $X, Y, P(Y|X)$ de capacité $C$ est il possible de transmettre des messages avec un débit aussi proche que possible de $C$ et avec une probabilité d'erreur aussi petite que possible ?

# Codage avec bruit : analyse du problème

- Codage sans bruit : la fonction de décodage est la transformation inverse de la fonction de codage ;
- En présence de bruit un message reçu peut correspondre à plusieurs messages en entrée
- La matrice de transition définit la loi de probabilité
- Il est donc nécessaire de définir une règle de décision permettant de décoder le message reçu

# Codage avec bruit : analyse du problème

- Codage sans bruit : la fonction de décodage est la transformation inverse de la fonction de codage ;
- En présence de bruit un message reçu peut correspondre à plusieurs messages en entrée
- La matrice de transition définit la loi de probabilité
- Il est donc nécessaire de définir une règle de décision permettant de décoder le message reçu

# Codage avec bruit : analyse du problème

- Codage sans bruit : la fonction de décodage est la transformation inverse de la fonction de codage ;
- En présence de bruit un message reçu peut correspondre à plusieurs messages en entrée
- La matrice de transition définit la loi de probabilité
- Il est donc nécessaire de définir une règle de décision permettant de décoder le message reçu

# Codage avec bruit : analyse du problème

- Codage sans bruit : la fonction de décodage est la transformation inverse de la fonction de codage ;
- En présence de bruit un message reçu peut correspondre à plusieurs messages en entrée
- La matrice de transition définit la loi de probabilité
- Il est donc nécessaire de définir une règle de décision permettant de décoder le message reçu

# Règle de décodage de canal

## Définition

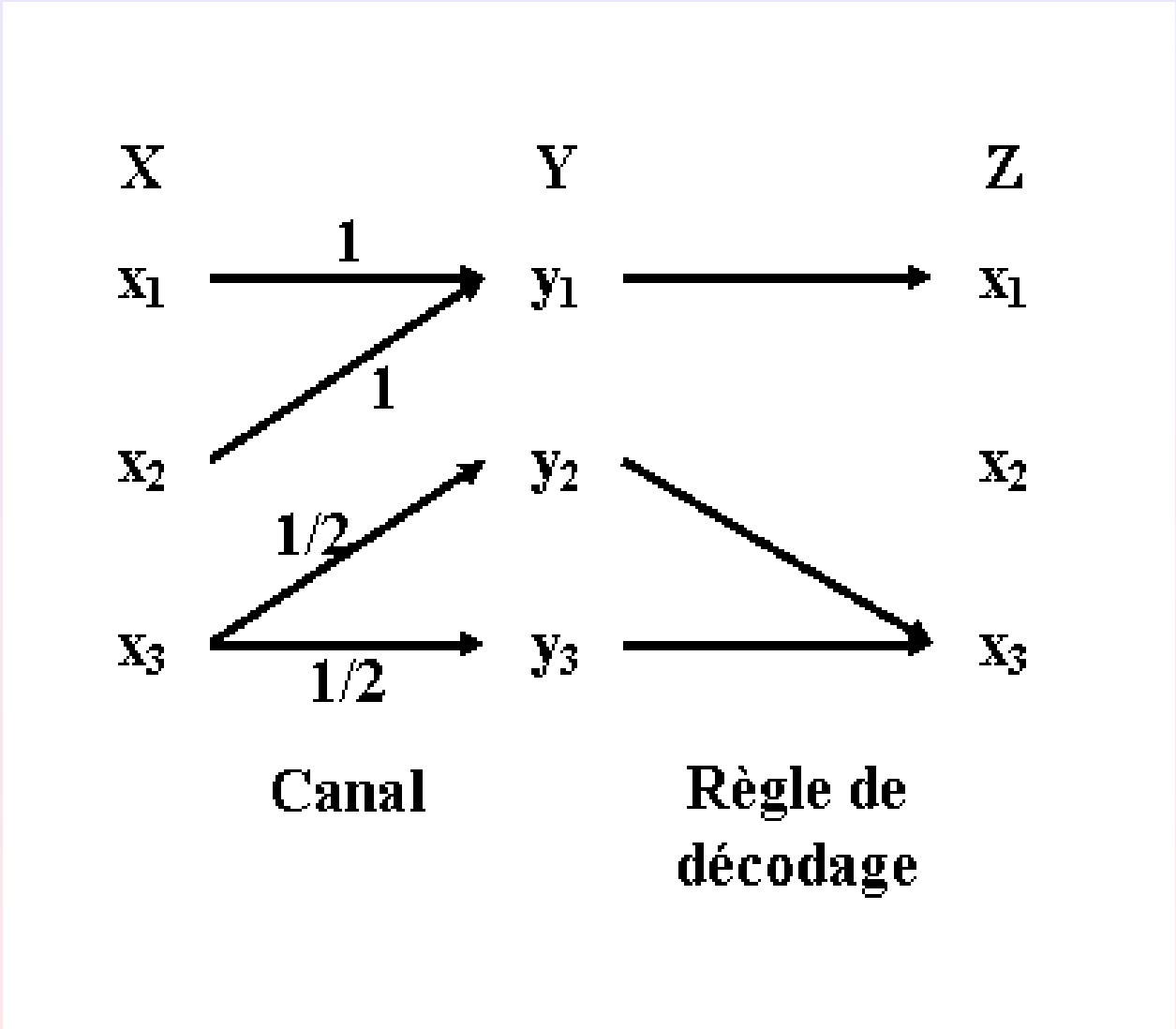
Soit un canal avec un alphabet d'entrée $\Omega_X = \{x_1, \ldots, x_n\}$ et un alphabet de sortie $\Omega_Y = \{y_1, \ldots, y_d\}$. La règle de décodage de canal est une fonction déterministe

$$g : \{y_1, \ldots, y_D\} \to \{x_1, \ldots, x_M\}$$

qui à chaque symbole reçu $y_j$ associe un symbole de l'alphabet d'entrée $x_j^* = g(y_j)$.

# Exemple

Soient $X$, $Y$ et $Z = g(Y)$ respectivement les symboles émis, reçu et décodé.



# Règle de décodage

$$g(y_1) = x_1,\ g(y_2) = x_3,\quad g(y_3) = x_2$$

Le seul cas où une erreur a lieu est l'émission de symbole $x_2$.
Ainsi la probabilité d'erreur est égale à la probabilité d'émission de $x_2$ donc à $1/4$.

# Règle de décodage

$$g(y_1) = x_1,\ g(y_2) = x_3,\quad g(y_3) = x_2$$

Le seul cas où une erreur a lieu est l'émission de symbole $x_2$.

Ainsi la probabilité d'erreur est égale à la probabilité d'émission de $x_2$ donc à $1/4$.

# Règle de décodage

$$g(y_1) = x_1,\ g(y_2) = x_3,\quad g(y_3) = x_2$$

Le seul cas où une erreur a lieu est l'émission de symbole $x_2$.
Ainsi la probabilité d'erreur est égale à la probabilité d'émission de $x_2$ donc à $1/4$.

# Erreur de transmission

Dans le cas général soit $E$ l'événement correspondant à une erreur lors de transmission d'un seul symbole.

Soient les variables aléatoires $X$, $Y$ et $Z = g(Y)$ représentant respectivement les symboles émis, reçu et décodé.

Comment peut on calculer $P[E]$ ?

Formule de Bays :

$$P(E) = \sum_{i=1}^{n} p(x_i) p(E|x_i)$$

# Erreur de transmission

Dans le cas général soit $E$ l'événement correspondant à une erreur lors de transmission d'un seul symbole.
Soient les variables aléatoires $X$, $Y$ et $Z = g(Y)$ représentant respectivement les symboles émis, reçu et décodé.
Comment peut on calculer $P[E]$ ?
Formule de Bays :

$$P(E) = \sum_{i=1}^{n} p(x_i) p(E|x_i)$$

# Erreur de transmission

Dans le cas général soit $E$ l'événement correspondant à une erreur lors de transmission d'un seul symbole.
Soient les variables aléatoires $X$, $Y$ et $Z = g(Y)$ représentant respectivement les symboles émis, reçu et décodé.
Comment peut on calculer $P[E]$ ?

Formule de Bays :

$$P(E) = \sum_{i=1}^{n} p(x_i) p(E|x_i)$$

# Erreur de transmission

Dans le cas général soit $E$ l'événement correspondant à une erreur lors de transmission d'un seul symbole.
Soient les variables aléatoires $X$, $Y$ et $Z = g(Y)$ représentant respectivement les symboles émis, reçu et décodé.
Comment peut on calculer $P[E]$ ?
Formule de Bays :

$$P(E) = \sum_{i=1}^{n} p(x_i) p(E|x_i)$$

Sachant que le symbole $x_i$ est transmis, l'erreur $E$ équivaut à $g(Y) \neq x_i$ :

$$p\left[E|x_i\right]$$

$$\left\{\right.$$

Sachant que le symbole $x_i$ est transmis, l'erreur $E$ équivaut à $g(Y) \neq x_i$ :

$$p\left[E|x_i\right]$$

Sachant que le symbole $x_i$ est transmis, l'erreur $E$ équivaut à $g(Y) \neq x_i$ :

$$p\left[E|x_i\right] = P\left[g(Y) \neq x_i|x_i\right]$$

Sachant que le symbole $x_i$ est transmis, l'erreur $E$ équivaut à $g(Y) \neq x_i$ :

$$p\left[E|x_i\right] = P\left[g(Y) \neq x_i|x_i\right] = \sum_{j=1}^{d} P\left[Y = y_j \text{ et } g(y_j) \neq x_i|x_i\right]$$

Sachant que le symbole $x_i$ est transmis, l'erreur $E$ équivaut à $g(Y) \neq x_i$ :

$$p\left[E|x_i\right] = P\left[g(Y) \neq x_i|x_i\right] = \sum_{j=1}^{d} P\left[Y = y_j \text{ et } g(y_j) \neq x_i|x_i\right]$$

Sachant que le symbole $x_i$ est transmis, l'erreur $E$ équivaut à $g(Y) \neq x_i$ :

$$p[E|x_i] = P[g(Y) \neq x_i|x_i] = \sum_{j=1}^{d} P[Y = y_j \text{ et } g(y_j) \neq x_i|x_i]$$

$$P[Y = y_j \text{ et } g(y_j) \neq x_i|x_i] = \left\{ \right.$$

Sachant que le symbole $x_i$ est transmis, l'erreur $E$ équivaut à $g(Y) \neq x_i$ :

$$p[E|x_i] = P[g(Y) \neq x_i|x_i] = \sum_{j=1}^{d} P[Y = y_j \text{ et } g(y_j) \neq x_i|x_i]$$

$$P[Y = y_j \text{ et } g(y_j) \neq x_i|x_i] = \begin{cases} p(y_j|x_i), & \text{si } g(y_j) \neq x_i \\ \end{cases}$$

Sachant que le symbole $x_i$ est transmis, l'erreur $E$ équivaut à $g(Y) \neq x_i$ :

$$p[E|x_i] = P[g(Y) \neq x_i|x_i] = \sum_{j=1}^{d} P[Y = y_j \text{ et } g(y_j) \neq x_i|x_i]$$

$$P[Y = y_j \text{ et } g(y_j) \neq x_i|x_i] = \begin{cases} p(y_j|x_i), & \text{si } g(y_j) \neq x_i \\ 0 & \text{si } g(y_j) = x_i \end{cases}$$

En utilisant le symbole de Kronecker $\delta_{ik} = \begin{cases} 1, & \text{si } i \neq k \\ 0 & \text{si } i = k \end{cases}$

on a

$$P\left[Y = y_j \text{ et } g(y_j) \neq x_i | x_i\right] = p(y_j|x_i)(1 - \delta_{g(y_j), x_i})$$

On définit ainsi

la probabilité d'erreur conditionnelle pour la transmission d'un seul symbole

$$p(E|x_i) = \sum_{j=1}^{d} p(y_j|x_j)(1 - \delta_{g(y_j), x_i}).$$

En utilisant le symbole de Kronecker $\delta_{ik} = \begin{cases} 1, & \text{si } i \neq k \\ 0 & \text{si } i = k \end{cases}$

on a

$$P\left[Y = y_j \text{ et } g(y_j) \neq x_i | x_i\right] = p(y_j|x_i)(1 - \delta_{g(y_j), x_i})$$

On définit ainsi

la probabilité d'erreur conditionnelle pour la transmission d'un seul symbole

$$p(E|x_i) = \sum_{j=1}^{d} p(y_j|x_j)(1 - \delta_{g(y_j), x_i}).$$

En utilisant le symbole de Kronecker $\delta_{ik} = \begin{cases} 1, & \text{si } i \neq k \\ 0 & \text{si } i = k \end{cases}$
on a

$$P\left[Y = y_j \text{ et } g(y_j) \neq x_i | x_i\right] = p(y_j|x_i)(1 - \delta_{g(y_j), x_i})$$

On définit ainsi

la probabilité d'erreur conditionnelle pour la transmission d'un seul symbole

$$p(E|x_i) = \sum_{j=1}^{d} p(y_j|x_j)(1 - \delta_{g(y_j), x_i}).$$

# Notion d'extension de canal

Soit un canal défini par le triplet $X, Y, P(Y|X)$ où

- $X$ est la variable aléatoire correspondante à l'émission d'un symbole de l'alphabet d'entrée $\Omega_X = \{x_1, \ldots, x_n\}$
- $Y$ est la variable correspondante à l'observation d'un symbole reçu dans l'alphabet de sortie $\Omega_Y = \{y_1, \ldots, y_d\}$
- $P(Y|X)$ est la matrice de transition du canal.

Soit $l \in \mathbb{N}$. On définit l'extension de source

$$X^{(l)} = (X_1, \ldots, X_l)$$

et l'extension du récepteur

$$Y^{(l)} = (Y_1, \ldots, Y_l)$$

# Notion d'extension de canal

Soit un canal défini par le triplet $X, Y, P(Y|X)$ où

- $X$ est la variable aléatoire correspondante à l'émission d'un symbole de l'alphabet d'entrée $\Omega_X = \{x_1, \ldots, x_n\}$
- $Y$ est la variable correspondante à l'observation d'un symbole reçu dans l'alphabet de sortie $\Omega_Y = \{y_1, \ldots, y_d\}$
- $P(Y|X)$ est la matrice de transition du canal.

Soit $l \in \mathbb{N}$. On définit l'extension de source

$$X^{(l)} = (X_1, \ldots, X_l)$$

et l'extension du récepteur

$$Y^{(l)} = (Y_1, \ldots, Y_l)$$

# Notion d'extension de canal

Soit un canal défini par le triplet $X, Y, P(Y|X)$ où

- $X$ est la variable aléatoire correspondante à l'émission d'un symbole de l'alphabet d'entrée $\Omega_X = \{x_1, \ldots, x_n\}$
- $Y$ est la variable correspondante à l'observation d'un symbole reçu dans l'alphabet de sortie $\Omega_Y = \{y_1, \ldots, y_d\}$
- $P(Y|X)$ est la matrice de transition du canal.

Soit $l \in \mathbb{N}$. On définit l'extension de source

$$X^{(l)} = (X_1, \ldots, X_l)$$

et l'extension du récepteur

$$Y^{(l)} = (Y_1, \ldots, Y_l)$$

# Notion d'extension de canal

Soit un canal défini par le triplet $X, Y, P(Y|X)$ où

- $X$ est la variable aléatoire correspondante à l'émission d'un symbole de l'alphabet d'entrée $\Omega_X = \{x_1, \ldots, x_n\}$
- $Y$ est la variable correspondante à l'observation d'un symbole reçu dans l'alphabet de sortie $\Omega_Y = \{y_1, \ldots, y_d\}$
- $P(Y|X)$ est la matrice de transition du canal.

Soit $l \in \mathbb{N}$. On définit l'extension de source

$$X^{(l)} = (X_1, \ldots, X_l)$$

et l'extension du récepteur

$$Y^{(l)} = (Y_1, \ldots, Y_l)$$

# Notion d'extension de canal

Soit un canal défini par le triplet $X, Y, P(Y|X)$ où

- $X$ est la variable aléatoire correspondante à l'émission d'un symbole de l'alphabet d'entrée $\Omega_X = \{x_1, \ldots, x_n\}$
- $Y$ est la variable correspondante à l'observation d'un symbole reçu dans l'alphabet de sortie $\Omega_Y = \{y_1, \ldots, y_d\}$
- $P(Y|X)$ est la matrice de transition du canal.

Soit $l \in \mathbb{N}$. On définit l'extension de source

$$X^{(l)} = (X_1, \ldots, X_l)$$

et l'extension du récepteur

$$Y^{(l)} = (Y_1, \ldots, Y_l)$$

# Notion d'extension de canal

Nous considérons alors la transmission d'un message de longueur $l$ comme un nouveau canal $X^{(l)}, Y^{(l)}, P(X^{(l)}|X^{(l)})$ appelé **$l$ème extension de canal** initial.

La matrice de transition de ce nouveau canal peut être déduite de

$$P((y_1, \ldots, y_l)|(x_1, \ldots, x_l)) = \prod_{k=1}^{l} p(y_k|x_k)$$

# Notion d'extension de canal

Nous considérons alors la transmission d'un message de longueur $l$ comme un nouveau canal $X^{(l)}, Y^{(l)}, P(X^{(l)}|X^{(l)})$ appelé **$l$ème extension de canal** initial.
La matrice de transition de ce nouveau canal peut être déduite de

$$P((y_1, \ldots, y_l)|(x_1, \ldots, x_l)) = \prod_{k=1}^{l} p(y_k|x_k)$$

# Notion de code de canal

## Définition

Soit un canal $X, Y, P(Y|X)$. Un code $(n, l)$ pour ce canal est un couple $(W, g)$ où

1. $W = \{w_1, \ldots, w_n\}$ est un ensemble de mots de longueur $l$ dans l'alphabet d'entrée $\Omega_X$, appelés mots du code.
2. $g$ est une règle de décodage

$$g : (\Omega_Y)^l \to W$$

qui associe à toute séquence reçue de longueur $l$ dans l'alphabet de sortie $\Omega_Y$ un des mots du code.

# Notion de code de canal

## Définition

Soit un canal $X, Y, P(Y|X)$. Un code $(n, l)$ pour ce canal est un couple $(W, g)$ où

1. $W = \{w_1, \ldots, w_n\}$ est un ensemble de mots de longueur $l$ dans l'alphabet d'entrée $\Omega_X$, appelés mots du code.
2. $g$ est une règle de décodage

$$g : (\Omega_Y)^l \to W$$

qui associe à toute séquence reçue de longueur $l$ dans l'alphabet de sortie $\Omega_Y$ un des mots du code.

# Probabilité d'erreur conditionnelle

## Définition

Soient un canal $X, Y, P(Y|X)$ et un code $(n, l)$ pour ce canal $(W, g)$. Pour chaque mot du code $w_i$ on définit la probabilité d'erreur conditionnelle

$$\lambda_i^{(l)} = P[E|w_i] = \sum_{(y_1,\ldots,y_l)\in(\Omega_Y)^l} P((y_1,\ldots,y_l)|w_i)(1-\delta_{g(y_1,\ldots,y_l),\,w_i}).$$

# Probabilité d'erreur moyenne et maximale

## Probabilité d'erreur moyenne

Soient un canal $X, Y, P(Y|X)$ et un code $(n, l)$ pour ce canal $(W, g)$. On définit la probabilité d'erreur moyenne **(algébrique)** du code par :

$$\bar{\lambda}^{(l)} = \frac{1}{n} \sum_{i=1}^{n} \lambda_i^{(l)}.$$

## Probabilité d'erreur maximale

$$\lambda^{(l)} = \max_{i=1,\ldots,n} \lambda_i^{(l)}.$$

# Probabilité d'erreur moyenne et maximale

## Probabilité d'erreur moyenne

Soient un canal $X, Y, P(Y|X)$ et un code $(n, l)$ pour ce canal $(W, g)$. On définit la probabilité d'erreur moyenne **(algébrique)** du code par :

$$\bar{\lambda}^{(l)} = \frac{1}{n} \sum_{i=1}^{n} \lambda_i^{(l)}.$$

## Probabilité d'erreur maximale

$$\lambda^{(l)} = \max_{i=1,\ldots,n} \lambda_i^{(l)}.$$

# Débit de communication d'un code

## Définition

Soient un canal $X, Y, P(Y|X)$ et un code $(n, l)$ pour ce canal $(W, g)$. On définit le débit de communication du code par :

$$R = \frac{log(n)}{l}.$$

l'unité de mesure est Shannon par symbole transmis.

# Second théorème de Shannon

## Théorème

Soit un canal $X, Y, P(Y|X)$ de capacité $C > 0$. Pour tout $R < C$ il est possible de trouver un code de canal avec un débit $R$ et une probabilité d'erreur aussi petite que possible. Plus précisément, il existe une suite de codes $(M(l), l)$ tels que $M(l) = 2^{lR}$ telle que

$$\lim_{l \to \infty} \lambda^{(l)} = 0$$