Principes de fonctionnement des machines binaires

2019/2020

Pierluigi Crescenzi
Université de Paris, IRIF

- Tests et examens
  - CC : résultat des tests en TD / TP (semaine 4 et semaine 10)
  - E0 : partiel (**samedi 26 octobre 9h30-11h30**)
    - **Amphi 4C : de A à D**
    - **Amphi 5C : de E à N**
    - **Amphi 9E : de O à ZA**
    - **Amphi 6C : MATH-INFO**

Arrivez à 9h !

  - E1 : examen mi décembre
  - E2 : examen fin juin
- Notes finales
  - Note session 1 : 25% CC + 25% E0 + 50% E1
  - Note session 2 : max(E2, 33% CC + 67% E2)
- Rappel
  - Pas de note ⇒ pas de moyenne ⇒ pas de semestre
- Site web
  - `moodlesupd.script.univ-paris-diderot.fr`

- Numération et arithmétique
- Numération et arithmétique en machine
- Numérisation et codage (texte, images)
- Compression, cryptographie, **contrôle d'erreur**
- Logique et calcul propositionnel
- Circuits numériques

Contrôle d'erreur

- Le principe est de fournir de la **redondance**
  - Doubler/tripler le message

- Le principe est de fournir de la **redondance**
  - Doubler/tripler le message
- Exemple
  - L’alphabet radio international (encore très très employé, y compris dans l’aviation civile)
    - On rajoute de l’information permettant d’assurer la bonne lisibilité du message en cas de bruit

- Le principe est de fournir de la **redondance**
  - Doubler/tripler le message
- Exemple
  - L'alphabet radio international (encore très très employé, y compris dans l'aviation civile)
    - On rajoute de l'information permettant d'assurer la bonne lisibilité du message en cas de bruit

A – Alpha	***J – Juliet***	***S – Sierra***
B – Bravo	***K – Kilo***	***T – Tango***
C – Charlie	***L – Lima***	***U – Uniform***
D – Delta	***M – Mike***	***V – Victor***
E – Echo	***N – November***	***W – Whiskey***
F – Foxtrot	***O – Oscar***	***X – X-Ray***
G – Golf	***P – Papa***	***Y – Yankee***
H – Hotel	***Q – Quebec***	***Z – Zulu***
I – India	***R – Romeo***	

- Le principe est de fournir de la **redondance**
  - Doubler/tripler le message
- Exemple
  - L’alphabet radio international (encore très très employé, y compris dans l’aviation civile)
    - On rajoute de l’information permettant d’assurer la bonne lisibilité du message en cas de bruit

A – Alpha	***J – Juliet***	***S – Sierra***
B – Bravo	***K – Kilo***	***T – Tango***
C – Charlie	***L – Lima***	***U – Uniform***
D – Delta	***M – Mike***	***V – Victor***
E – Echo	***N – November***	***W – Whiskey***
F – Foxtrot	***O – Oscar***	***X – X-Ray***
G – Golf	***P – Papa***	***Y – Yankee***
H – Hotel	***Q – Quebec***	***Z – Zulu***
I – India	***R – Romeo***	

    - **PF1** en radio international **Papa**, **Foxtrot**, **One**

- Le **CRC** était employé dans la transmission du code ASCII, en ajoutant un 8-ième bit de contrôle, appelé **bit de parité**
  - De sorte que la somme des huit bits soit toujours **paire**

- Le **CRC** était employé dans la transmission du code ASCII, en ajoutant un 8-ième bit de contrôle, appelé **bit de parité**
  - De sorte que la somme des huit bits soit toujours **paire**
    - CRC du mot `0110001`

- Le **CRC** était employé dans la transmission du code ASCII, en ajoutant un 8-ième bit de contrôle, appelé **bit de parité**
  - De sorte que la somme des huit bits soit toujours **paire**
    - CRC du mot `0110001`
      - `1`

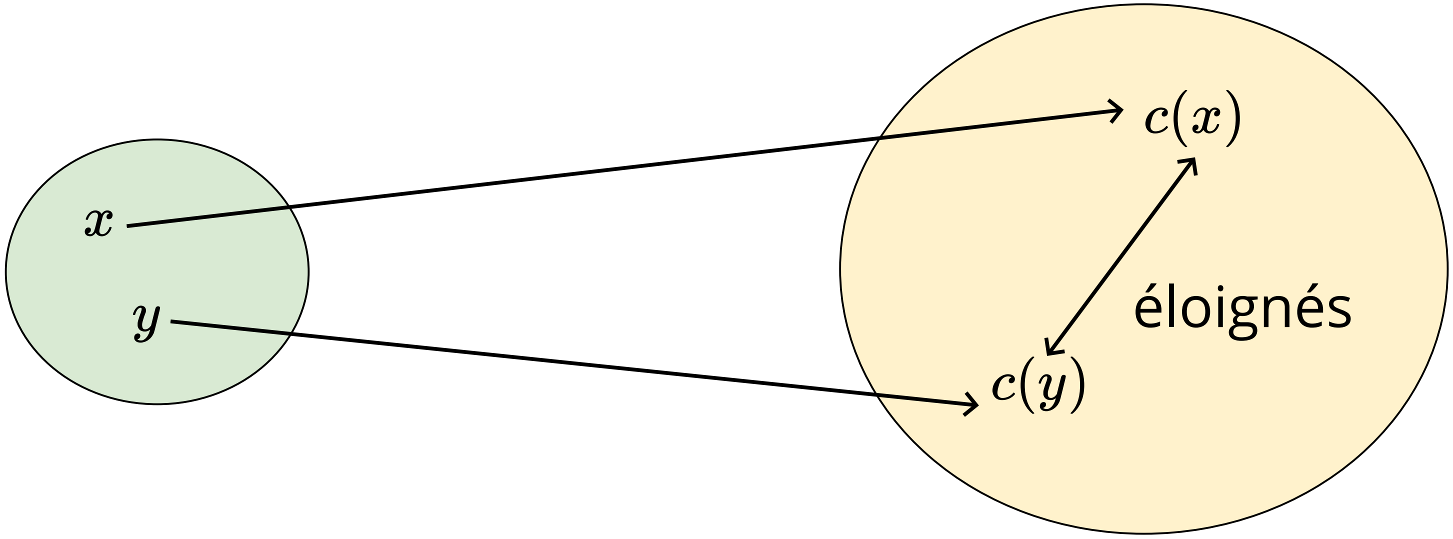
- Le **CRC** était employé dans la transmission du code ASCII, en ajoutant un 8-ième bit de contrôle, appelé **bit de parité**
  - De sorte que la somme des huit bits soit toujours **paire**
    - CRC du mot `0110001`
      - `1`
    - CRC du mot `0110110`

- Le **CRC** était employé dans la transmission du code ASCII, en ajoutant un 8-ième bit de contrôle, appelé **bit de parité**
  - De sorte que la somme des huit bits soit toujours **paire**
    - CRC du mot `0110001`
      - `1`
    - CRC du mot `0110110`
      - `0`

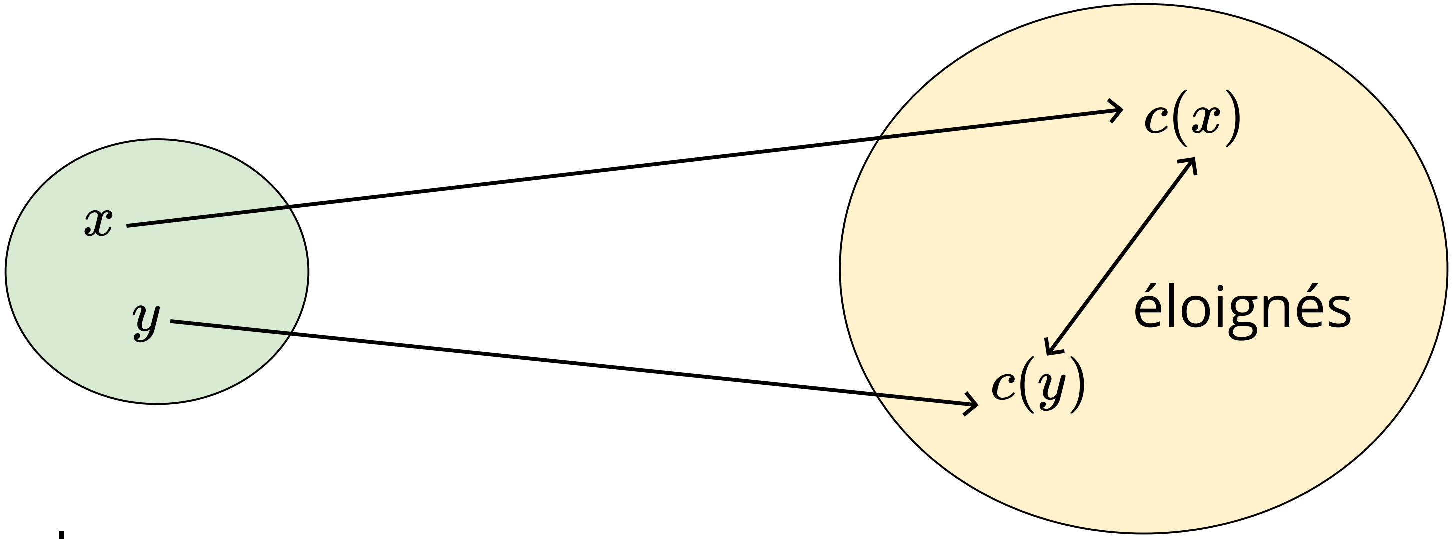
- Le **CRC** était employé dans la transmission du code ASCII, en ajoutant un 8-ième bit de contrôle, appelé **bit de parité**
  - De sorte que la somme des huit bits soit toujours **paire**
    - CRC du mot `0110001`
      - `1`
    - CRC du mot `0110110`
      - `0`
  - Donc si je reçois 8 bits dont le nombre de 1 est impair, je sais que la transmission est erronée
    - Dans les autres cas, je ne sais pas

- Le **CRC** était employé dans la transmission du code ASCII, en ajoutant un 8-ième bit de contrôle, appelé **bit de parité**
  - De sorte que la somme des huit bits soit toujours **paire**
    - CRC du mot `0110001`
      - `1`
    - CRC du mot `0110110`
      - `0`
  - Donc si je reçois 8 bits dont le nombre de 1 est impair, je sais que la transmission est erronée
    - Dans les autres cas, je ne sais pas
  - Permet de détecter si **une** erreur s'est produite, mais pas où
    - Ne permet pas de détecter deux erreurs

- Idée de base
  - Coder les mots d'origine en **mots de code** assez éloignés les uns des autres
    - Difficile à rater même en présence d'erreurs

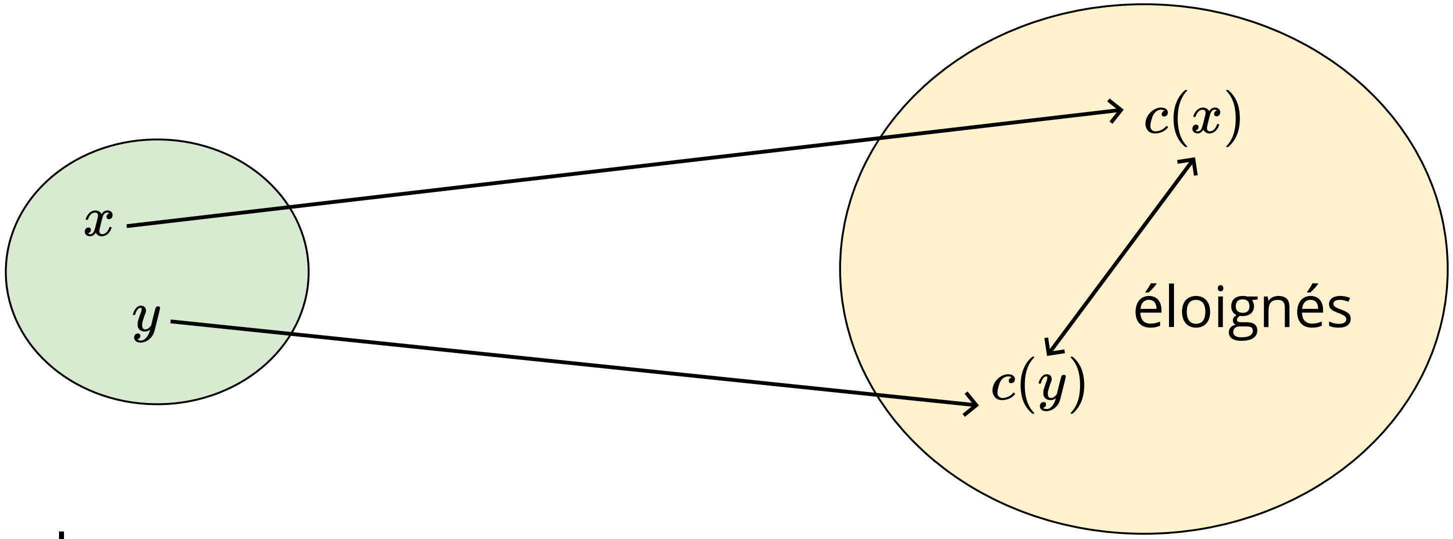


- Idée de base
  - Coder les mots d'origine en **mots de code** assez éloignés les uns des autres
    - Difficile à rater même en présence d'erreurs



- Exemple
  - **BravoDelta** plus loin de **PapaTango** que **BD** de **PT**

- Idée de base
  - Coder les mots d'origine en **mots de code** assez éloignés les uns des autres
    - Difficile à rater même en présence d'erreurs



- Exemple
  - **BravoDelta** plus loin de **PapaTango** que **BD** de **PT**
- Nous devons définir une notion de distance
  - Entre mots binaires

- **Distance de Hamming** $d_H(a,b)$ entre deux mots binaires de longeur n $a = a_1a_2\cdots a_n$ et $b = b_1b_2\cdots b_n$
  - Nombre d'indices i tels que $a_i \neq b_i$

- **Distance de Hamming** $d_H(a,b)$ entre deux mots binaires de longeur n $a = a_1 a_2 \cdots a_n$ et $b = b_1 b_2 \cdots b_n$
  - Nombre d'indices i tels que $a_i \neq b_i$
- Exemples
  - $a = 0001111$ et $b = 1101011$
  - $a = 1011101$ et $b = 1001001$
  - $a = 1011101$ et $b = 1011101$
  - $a = 1011101$ et $b = 0100010$

- **Distance de Hamming** $d_H(a,b)$ entre deux mots binaires de longeur n $a = a_1a_2\cdots a_n$ et $b = b_1b_2\cdots b_n$
  - Nombre d'indices i tels que $a_i \neq b_i$
- Exemples
  - $a = 0001111$ et $b = 1101011$
    - $d_H(a,b) = 1+1+0+0+1+0+0 = 3$
  - $a = 1011101$ et $b = 1001001$
  - $a = 1011101$ et $b = 1011101$
  - $a = 1011101$ et $b = 0100010$

- **Distance de Hamming** $d_H(a,b)$ entre deux mots binaires de longeur n $a = a_1a_2\cdots a_n$ et $b = b_1b_2\cdots b_n$
  - Nombre d’indices i tels que $a_i \neq b_i$
- Exemples
  - $a = 0001111$ et $b = 1101011$
    - $d_H(a,b) = 1+1+0+0+1+0+0 = 3$
  - $a = 1011101$ et $b = 1001001$
    - $d_H(a,b) = 0+0+1+0+1+0+0 = 2$
  - $a = 1011101$ et $b = 1011101$
  - $a = 1011101$ et $b = 0100010$

- **Distance de Hamming** $d_H(a,b)$ entre deux mots binaires de longeur n $a = a_1a_2\cdots a_n$ et $b = b_1b_2\cdots b_n$
  - Nombre d'indices i tels que $a_i \neq b_i$
- Exemples
  - $a = 0001111$ et $b = 1101011$
    - $d_H(a,b) = 1+1+0+0+1+0+0 = 3$
  - $a = 1011101$ et $b = 1001001$
    - $d_H(a,b) = 0+0+1+0+1+0+0 = 2$
  - $a = 1011101$ et $b = 1011101$
    - $d_H(a,b) = 0+0+0+0+0+0+0 = 0$
  - $a = 1011101$ et $b = 0100010$

- **Distance de Hamming** $d_H(a,b)$ entre deux mots binaires de longeur n $a = a_1 a_2 \cdots a_n$ et $b = b_1 b_2 \cdots b_n$
  - Nombre d'indices i tels que $a_i \neq b_i$
- Exemples
  - $a = 0001111$ et $b = 1101011$
    - $d_H(a,b) = 1+1+0+0+1+0+0 = 3$
  - $a = 1011101$ et $b = 1001001$
    - $d_H(a,b) = 0+0+1+0+1+0+0 = 2$
  - $a = 1011101$ et $b = 1011101$
    - $d_H(a,b) = 0+0+0+0+0+0+0 = 0$
  - $a = 1011101$ et $b = 0100010$
    - $d_H(a,b) = 1+1+1+1+1+1+1 = 7$

- Un code binaire de longueur n est un ensemble de mots binaires de longueur n
- La distance de Hamming d'un code binaire est le minimum des distances entre deux mots du code

- Un code binaire de longueur n est un ensemble de mots binaires de longueur n
- La distance de Hamming d'un code binaire est le minimum des distances entre deux mots du code
- Exemples
  - $\{000, 111\}$
  - $\{000, 011, 101, 110\}$
  - $\{000, 001, 011, 101, 110\}$

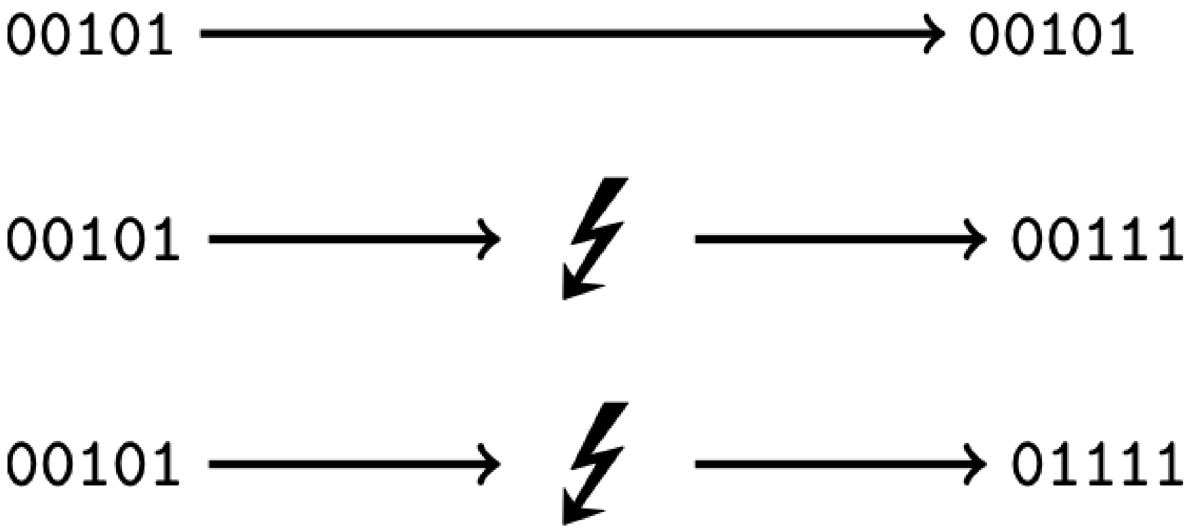
- Un code binaire de longueur n est un ensemble de mots binaires de longueur n
- La distance de Hamming d'un code binaire est le minimum des distances entre deux mots du code
- Exemples
  - $\{000, 111\}$
    - Distance de Hamming : 3
  - $\{000, 011, 101, 110\}$
  - $\{000, 001, 011, 101, 110\}$

- Un code binaire de longueur n est un ensemble de mots binaires de longueur n
- La distance de Hamming d'un code binaire est le minimum des distances entre deux mots du code
- Exemples
  - $\{000, 111\}$
    - Distance de Hamming : 3
  - $\{000, 011, 101, 110\}$
    - Distance de Hamming : 2
  - $\{000, 001, 011, 101, 110\}$

- Un code binaire de longueur n est un ensemble de mots binaires de longueur n
- La distance de Hamming d'un code binaire est le minimum des distances entre deux mots du code
- Exemples
  - $\{000, 111\}$
    - Distance de Hamming : 3
  - $\{000, 011, 101, 110\}$
    - Distance de Hamming : 2
  - $\{000, 001, 011, 101, 110\}$
    - Distance de Hamming : 1

- Le mot du code c est émis
  - Après d'éventuelles erreurs de transmission, le mot r est reçu
- On décode le mot r selon le principe du *maximum de vraisemblance*
  - On le décode comme un mot du code à distance minimum de r

- Le mot du code c est émis
  - Après d'éventuelles erreurs de transmission, le mot r est reçu
- On décode le mot r selon le principe du *maximum de vraisemblance*
  - On le décode comme un mot du code à distance minimum de r
- Exemples : code $\{00101, 01010, 11111\}$



- Le mot du code c est émis
  - Après d'éventuelles erreurs de transmission, le mot r est reçu
- On décode le mot r selon le principe du *maximum de vraisemblance*
  - On le décode comme un mot du code à distance minimum de r
- Exemples : code $\{00101, 01010, 11111\}$

  $$00101 \longrightarrow 00101$$

  - Pas d'erreur : 00101

- Le mot du code c est émis
  - Après d'éventuelles erreurs de transmission, le mot r est reçu
- On décode le mot r selon le principe du *maximum de vraisemblance*
  - On le décode comme un mot du code à distance minimum de r
- Exemples : code $\{00101, 01010, 11111\}$

  - Pas d'erreur : 00101

  - Un erreur : mot plus proche 00101 (correct)

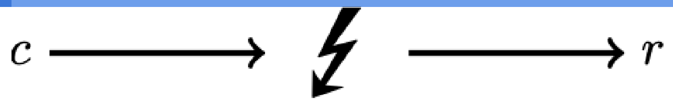
- Le mot du code c est émis
  - Après d'éventuelles erreurs de transmission, le mot r est reçu
- On décode le mot r selon le principe du *maximum de vraisemblance*
  - On le décode comme un mot du code à distance minimum de r
- Exemples : code $\{00101, 01010, 11111\}$

  - Pas d'erreur : 00101

  - Un erreur : mot plus proche 00101 (correct)

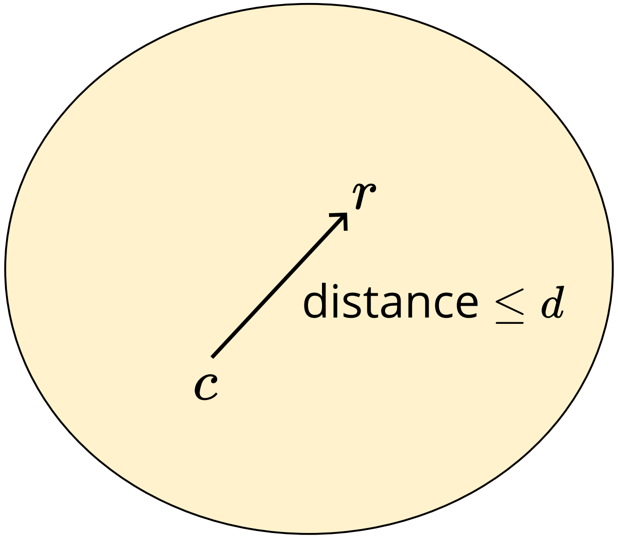
  - Deux erreurs : mot plus proche 11111 (incorrect)

- c est émis et r est reçu



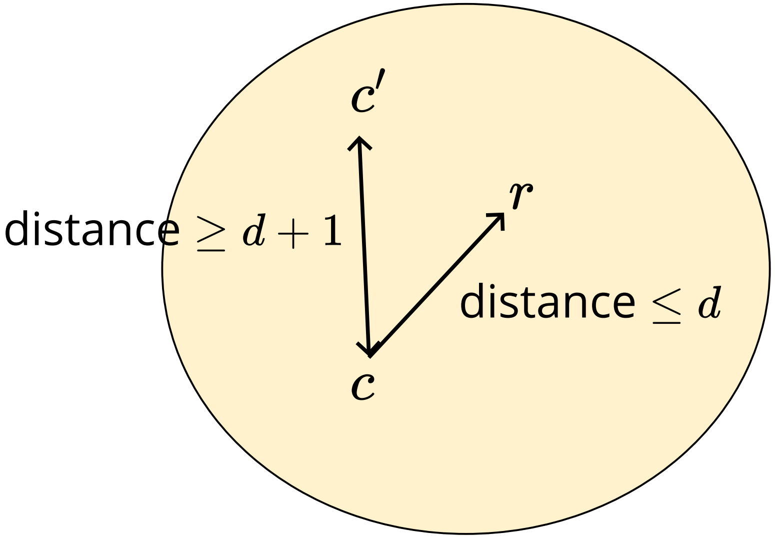
- c est émis et r est reçu

- Le code doit **détecter** d erreurs



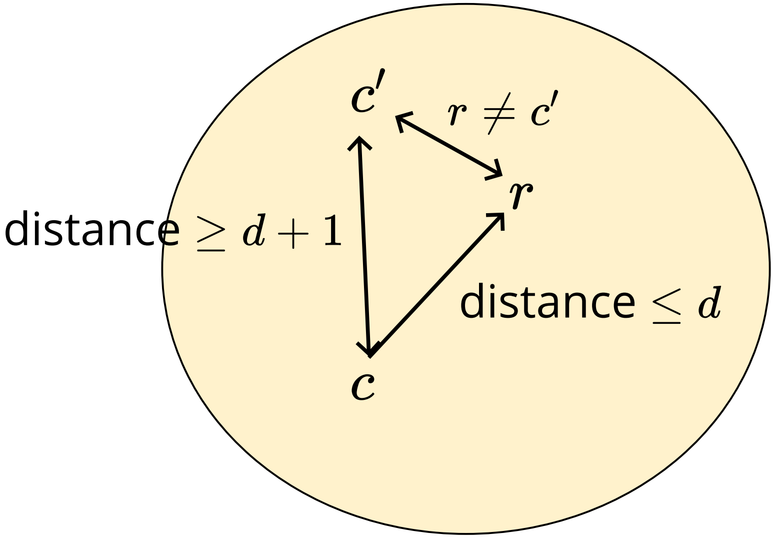
- c est émis et r est reçu

- Le code doit **détecter** d erreurs
  - La distance de Hamming du code doit être au moins $d+1$



- c est émis et r est reçu

- Le code doit **détecter** d erreurs
  - La distance de Hamming du code doit être au moins $d+1$
    - Dans ce cas r ne peut pas être un mot du code c'
    - Je sais que r est faux

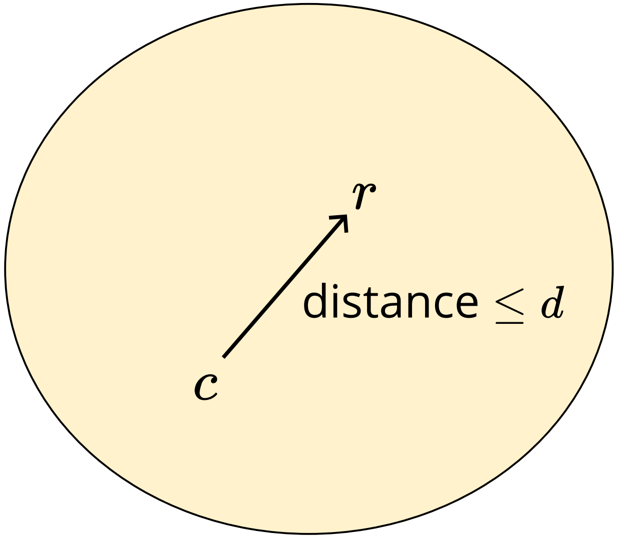


- c est émis et r est reçu

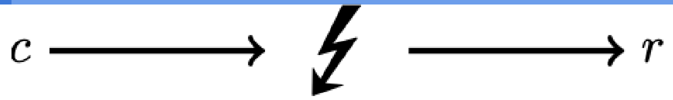
- Le code doit **détecter** d erreurs
  - La distance de Hamming du code doit être au moins $d+1$
    - Dans ce cas r ne peut pas être un mot du code c'
    - Je sais que r est faux

- c est émis et r est reçu

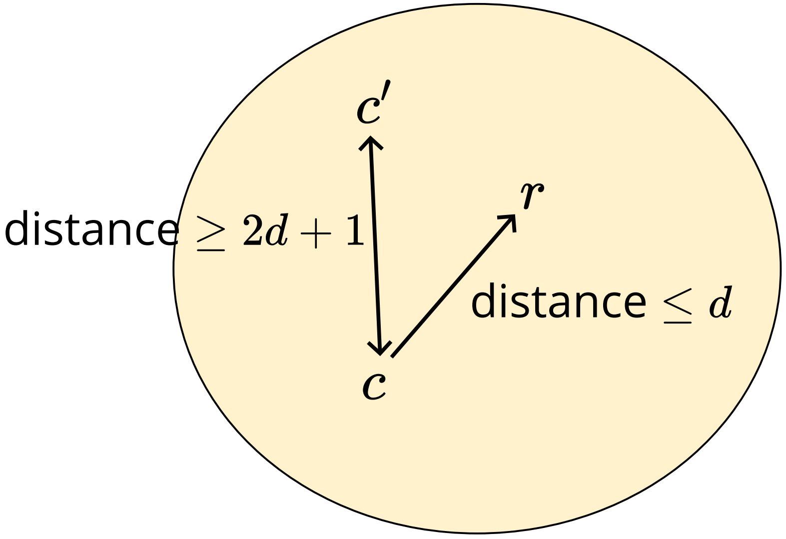
- Le code doit **détecter** d erreurs
  - La distance de Hamming du code doit être au moins $d+1$
    - Dans ce cas r ne peut pas être un mot du code c'
    - Je sais que r est faux
- Le code doit **corriger** d erreurs



- c est émis et r est reçu

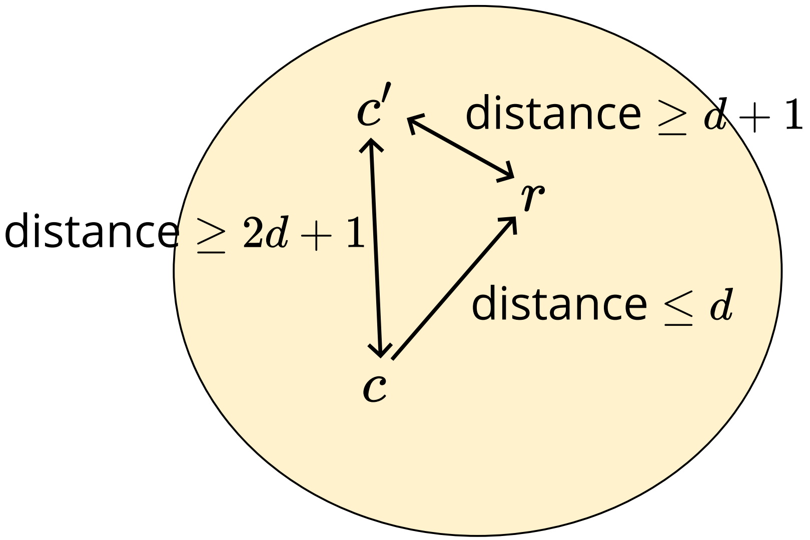


- Le code doit **détecter** d erreurs
  - La distance de Hamming du code doit être au moins $d+1$
    - Dans ce cas r ne peut pas être un mot du code c'
    - Je sais que r est faux
- Le code doit **corriger** d erreurs
  - La distance de Hamming du code doit être au moins $2d+1$



- c est émis et r est reçu

- Le code doit **détecter** d erreurs
  - La distance de Hamming du code doit être au moins $d+1$
    - Dans ce cas r ne peut pas être un mot du code c'
    - Je sais que r est faux
- Le code doit **corriger** d erreurs
  - La distance de Hamming du code doit être au moins $2d+1$
    - La distance entre r et les autres mots du code doit être au moins $d+1$
    - Je sais que r est faux et je corrige r pour obtenir c



- Une famille de codes qui permet de détecter et corriger
  - Son principe est de calculer plusieurs bits de parité sur différentes parties du mot de sorte qu'il soit possible de détecter si erreur il y a et où

- Une famille de codes qui permet de détecter et corriger
  - Son principe est de calculer plusieurs bits de parité sur différentes parties du mot de sorte qu'il soit possible de détecter si erreur il y a et où
- Le code de Hamming le plus simple est le [7,4]
  - Il code des messages de 4 bits sur 7 bits (c'est le prix à payer)
    - Il permet de détecter et corriger 1 erreur et de détecter 2 erreurs

- Une famille de codes qui permet de détecter et corriger
  - Son principe est de calculer plusieurs bits de parité sur différentes parties du mot de sorte qu'il soit possible de détecter si erreur il y a et où
- Le code de Hamming le plus simple est le [7,4]
  - Il code des messages de 4 bits sur 7 bits (c'est le prix à payer)
    - Il permet de détecter et corriger 1 erreur et de détecter 2 erreurs
  - Le message est $d_1d_2d_3d_4$
    - On y rajoute 3 bits p_1, p_2, p_3 de sorte que

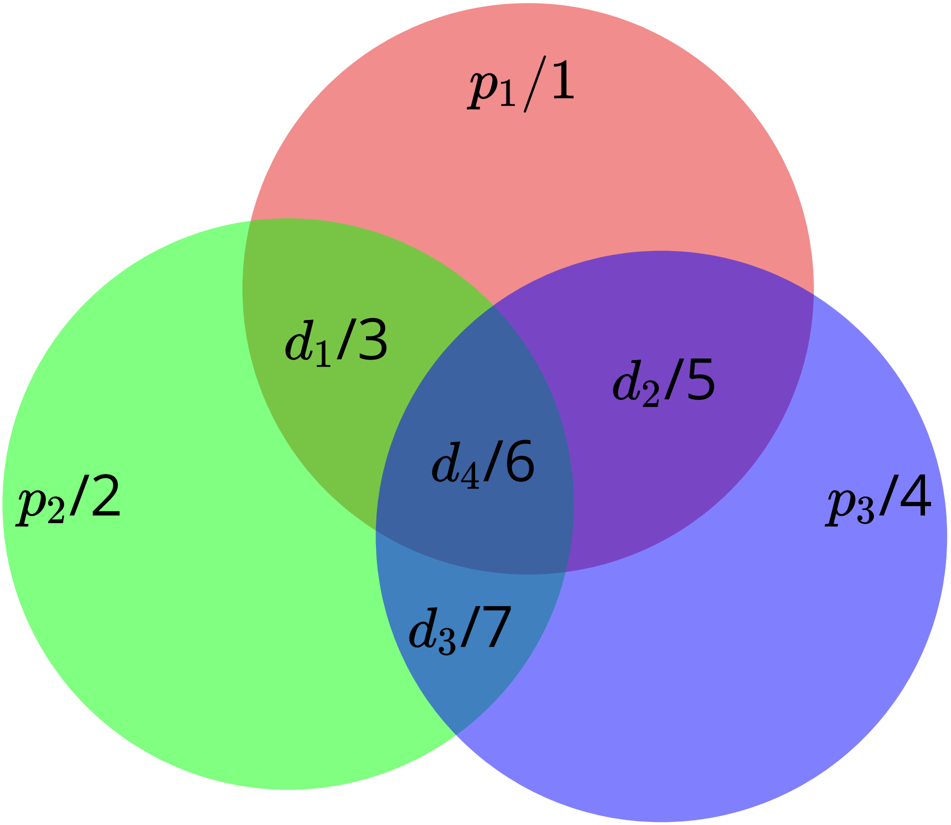
- Une famille de codes qui permet de détecter et corriger
  - Son principe est de calculer plusieurs bits de parité sur différentes parties du mot de sorte qu'il soit possible de détecter si erreur il y a et où
- Le code de Hamming le plus simple est le [7,4]
  - Il code des messages de 4 bits sur 7 bits (c'est le prix à payer)
    - Il permet de détecter et corriger 1 erreur et de détecter 2 erreurs
  - Le message est $d_1 d_2 d_3 d_4$
    - On y rajoute 3 bits p_1, p_2, p_3 de sorte que
      - p_1 est le bit de parité de $d_1 d_2 d_4$

- Une famille de codes qui permet de détecter et corriger
  - Son principe est de calculer plusieurs bits de parité sur différentes parties du mot de sorte qu’il soit possible de détecter si erreur il y a et où
- Le code de Hamming le plus simple est le [7,4]
  - Il code des messages de 4 bits sur 7 bits (c’est le prix à payer)
    - Il permet de détecter et corriger 1 erreur et de détecter 2 erreurs
  - Le message est $d_1 d_2 d_3 d_4$
    - On y rajoute 3 bits p_1, p_2, p_3 de sorte que
      - p_1 est le bit de parité de $d_1 d_2 d_4$
      - p_2 est le bit de parité de $d_1 d_3 d_4$

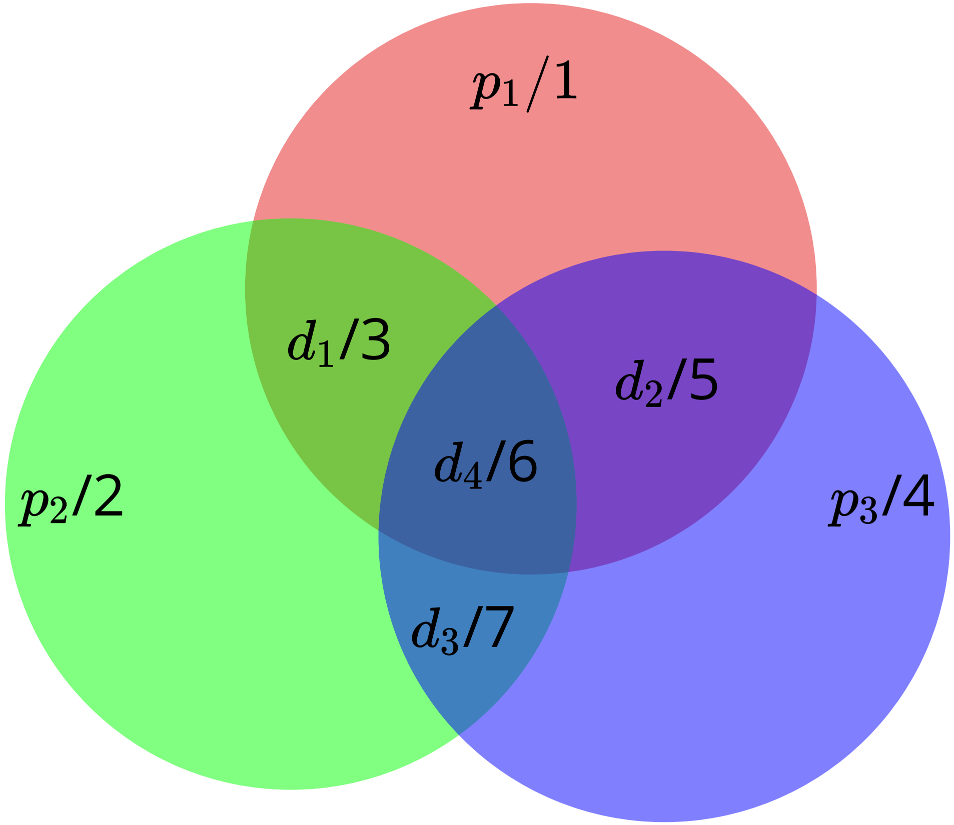
- Une famille de codes qui permet de détecter et corriger
  - Son principe est de calculer plusieurs bits de parité sur différentes parties du mot de sorte qu'il soit possible de détecter si erreur il y a et où
- Le code de Hamming le plus simple est le [7,4]
  - Il code des messages de 4 bits sur 7 bits (c'est le prix à payer)
    - Il permet de détecter et corriger 1 erreur et de détecter 2 erreurs
  - Le message est $d_1 d_2 d_3 d_4$
    - On y rajoute 3 bits p_1, p_2, p_3 de sorte que
      - p_1 est le bit de parité de $d_1 d_2 d_4$
      - p_2 est le bit de parité de $d_1 d_3 d_4$
      - p_3 est le bit de parité de $d_2 d_3 d_4$

- Une famille de codes qui permet de détecter et corriger
  - Son principe est de calculer plusieurs bits de parité sur différentes parties du mot de sorte qu'il soit possible de détecter si erreur il y a et où
- Le code de Hamming le plus simple est le [7,4]
  - Il code des messages de 4 bits sur 7 bits (c'est le prix à payer)
    - Il permet de détecter et corriger 1 erreur et de détecter 2 erreurs
  - Le message est $d_1 d_2 d_3 d_4$
    - On y rajoute 3 bits p_1, p_2, p_3 de sorte que
      - p_1 est le bit de parité de $d_1 d_2 d_4$
      - p_2 est le bit de parité de $d_1 d_3 d_4$
      - p_3 est le bit de parité de $d_2 d_3 d_4$
  - Le mot de code est $p_1 p_2 d_1 p_3 d_2 d_3 d_4$

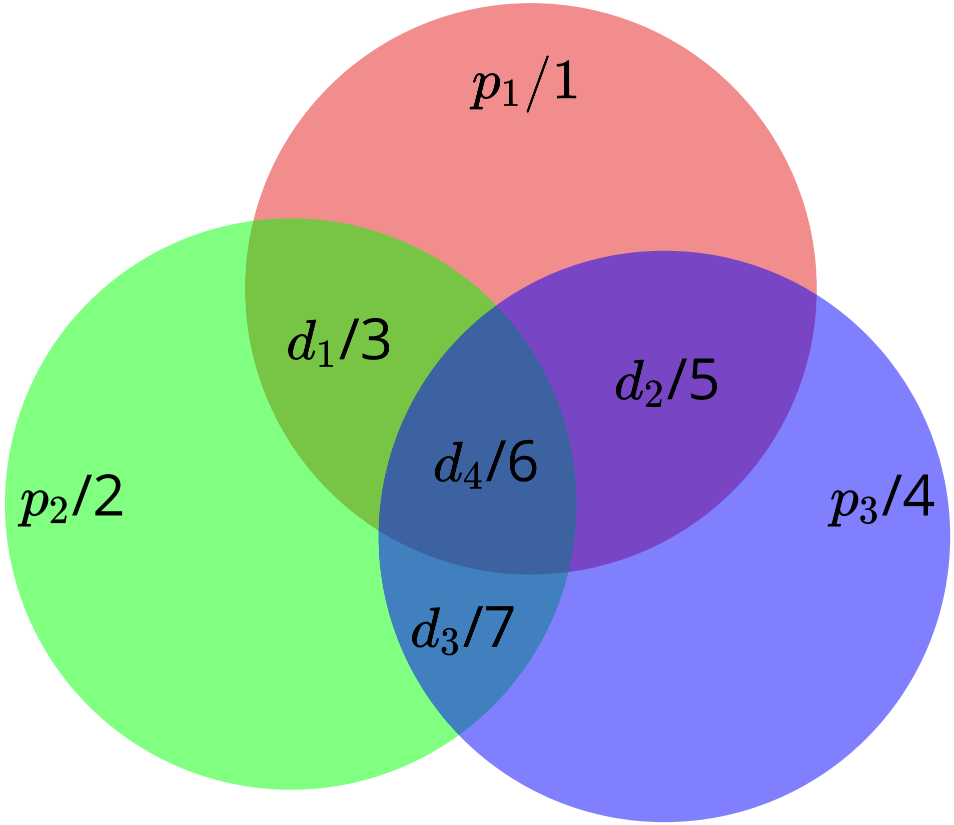
- Exemple
  - Le message est $\overset{d_1}{0}\ \overset{d_2}{1}\ \overset{d_3}{0}\ \overset{d_4}{1}$



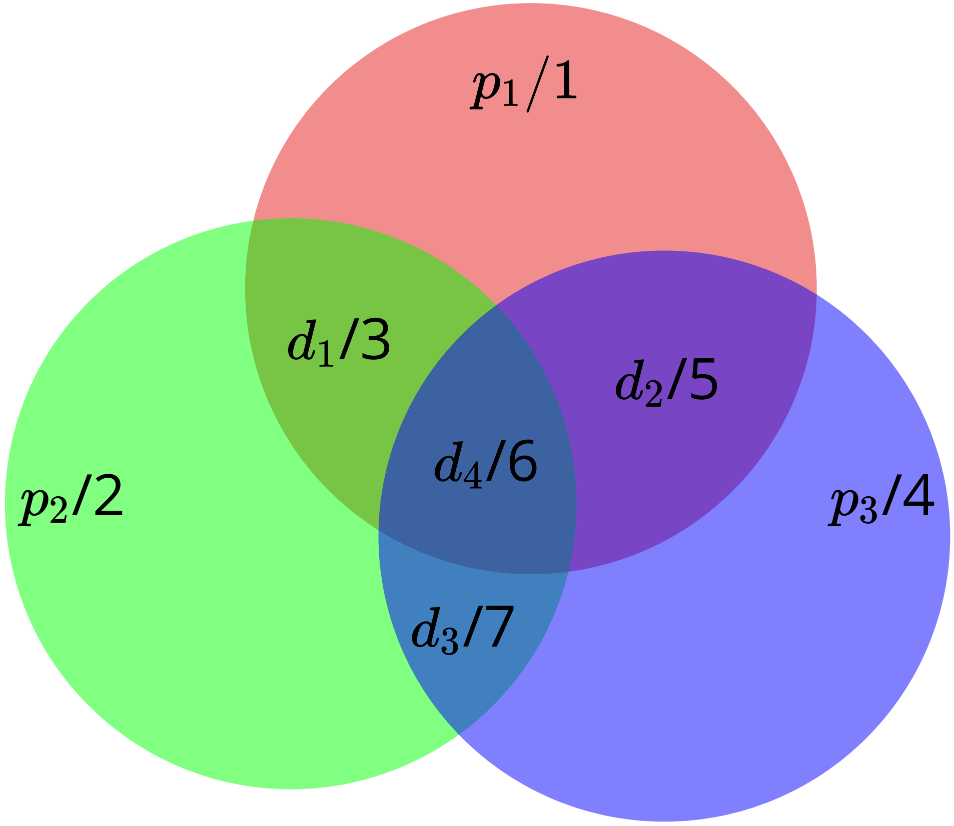
- Exemple
  - Le message est $\overset{d_1}{0}\ \overset{d_2}{1}\ \overset{d_3}{0}\ \overset{d_4}{1}$
  - On calcule
    - p_1
    - p_2
    - p_3



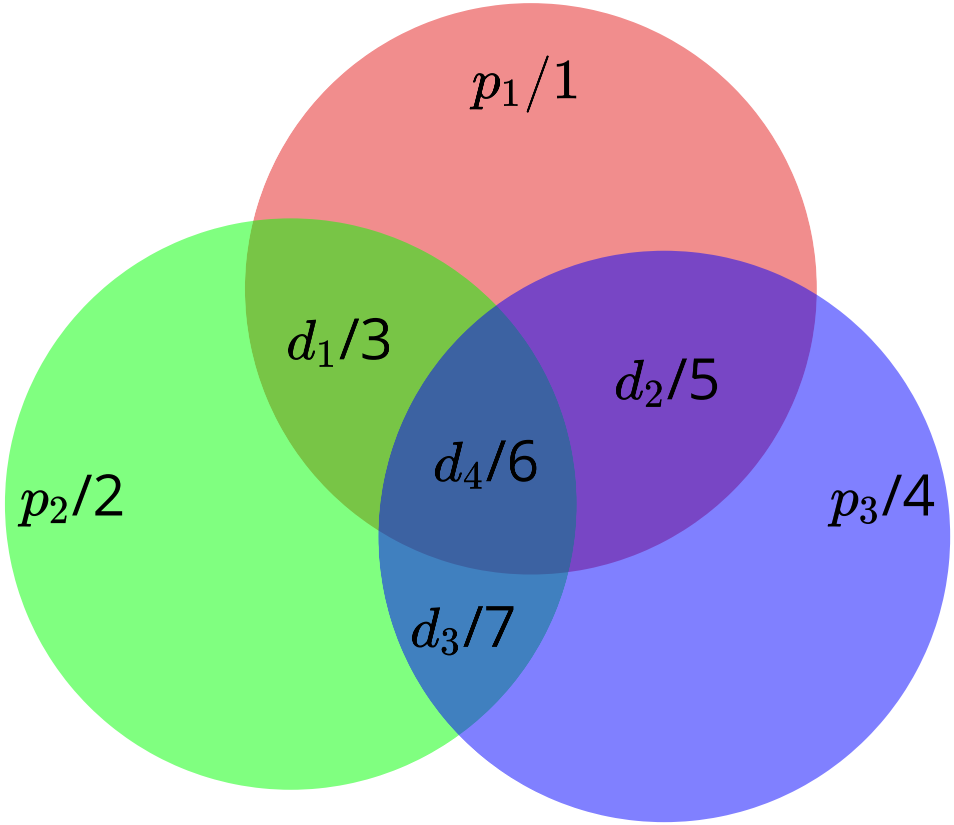
- Exemple
  - Le message est $\overset{d_1}{0}\ \overset{d_2}{1}\ \overset{d_3}{0}\ \overset{d_4}{1}$
  - On calcule
    - p_1
      - 0
    - p_2
    - p_3



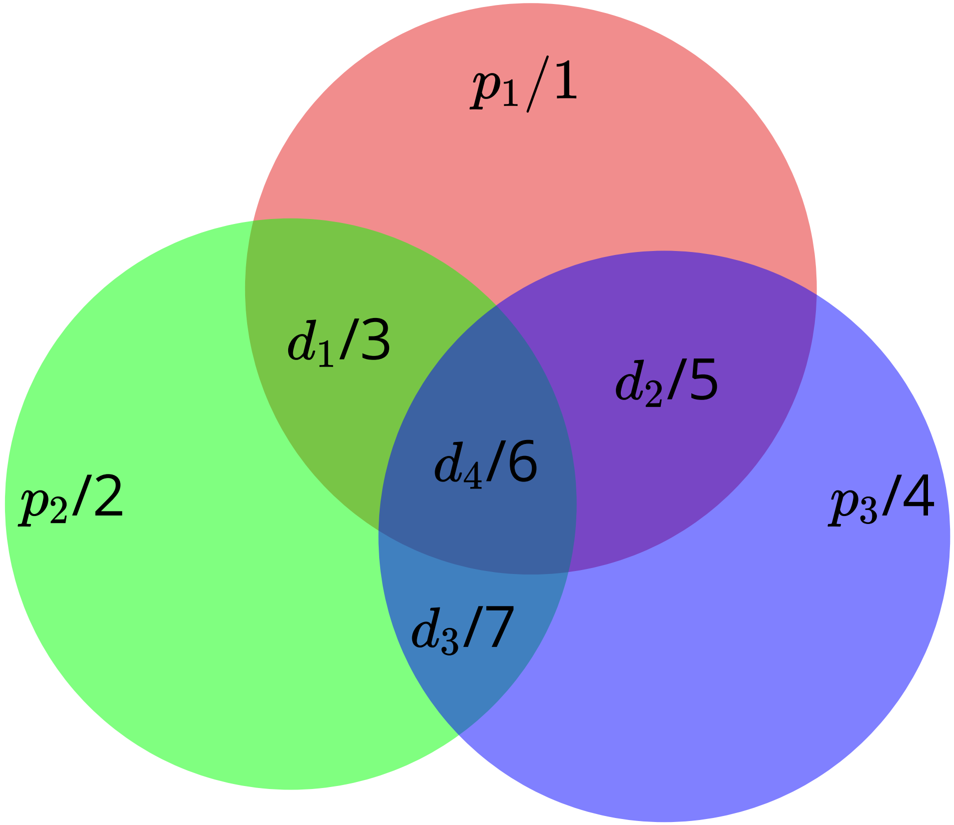
- Exemple
  - Le message est $\overset{d_1}{0}\ \overset{d_2}{1}\ \overset{d_3}{0}\ \overset{d_4}{1}$
  - On calcule
    - p_1
      - 0
    - p_2
      - 1
    - p_3



- Exemple
  - Le message est $\overset{d_1}{0}\ \overset{d_2}{1}\ \overset{d_3}{0}\ \overset{d_4}{1}$
  - On calcule
    - p_1
      - 0
    - p_2
      - 1
    - p_3
      - 0



- Exemple
  - Le message est $\overset{d_1}{0}\ \overset{d_2}{1}\ \overset{d_3}{0}\ \overset{d_4}{1}$
  - On calcule
    - p_1
      - 0
    - p_2
      - 1
    - p_3
      - 0
  - Le mot de code est $\overset{p_1}{0}\ \overset{p_2}{1}\ \overset{d_1}{0}\ \overset{p_3}{0}\ \overset{d_2}{1}\ \overset{d_3}{0}\ \overset{d_4}{1}$



d_1	d_2	d_3	d_4	p_1	p_2	p_3
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	0	0	0
1	1	1	1	1	1	1

- Distance de Hamming

d_1	d_2	d_3	d_4	p_1	p_2	p_3
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	0	0	0
1	1	1	1	1	1	1

- Distance de Hamming

$$d_H(a,b) = 3$$

	d_1	d_2	d_3	d_4	p_1	p_2	p_3
a	0	0	0	0	0	0	0
	0	0	0	1	1	1	1
	0	0	1	0	0	1	1
	0	0	1	1	1	0	0
b	0	1	0	0	1	0	1
	0	1	0	1	0	1	0
	0	1	1	0	1	1	0
	0	1	1	1	0	0	1
	1	0	0	0	1	1	0
	1	0	0	1	0	0	1
	1	0	1	0	1	0	1
	1	0	1	1	0	1	0
	1	1	0	0	0	1	1
	1	1	0	1	1	0	0
	1	1	1	0	0	0	0
	1	1	1	1	1	1	1

- Distance de Hamming

$$d_H(a,b) = 3$$

	d_1	d_2	d_3	d_4	p_1	p_2	p_3
	0	0	0	0	0	0	0
	0	0	0	1	1	1	1
	0	0	1	0	0	1	1
	0	0	1	1	1	0	0
	0	1	0	0	1	0	1
	0	1	0	1	0	1	0
	0	1	1	0	1	1	0
a	0	1	1	1	0	0	1
	1	0	0	0	1	1	0
	1	0	0	1	0	0	1
	1	0	1	0	1	0	1
	1	0	1	1	0	1	0
	1	1	0	0	0	1	1
	1	1	0	1	1	0	0
b	1	1	1	0	0	0	0
	1	1	1	1	1	1	1

- Distance de Hamming

$$d_H(a,b) = 3$$

	d_1	d_2	d_3	d_4	p_1	p_2	p_3
	0	0	0	0	0	0	0
a	0	0	0	1	1	1	1
	0	0	1	0	0	1	1
	0	0	1	1	1	0	0
	0	1	0	0	1	0	1
	0	1	0	1	0	1	0
	0	1	1	0	1	1	0
	0	1	1	1	0	0	1
	1	0	0	0	1	1	0
	1	0	0	1	0	0	1
	1	0	1	0	1	0	1
	1	0	1	1	0	1	0
	1	1	0	0	0	1	1
	1	1	0	1	1	0	0
	1	1	1	0	0	0	0
b	1	1	1	1	1	1	1

- Distance de Hamming
  - 3

d_1	d_2	d_3	d_4	p_1	p_2	p_3
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	0	0	0
1	1	1	1	1	1	1

- Distance de Hamming
  - 3
- Le code peut corriger 1 erreur et détecter 2 erreurs

d_1	d_2	d_3	d_4	p_1	p_2	p_3
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	0	0	0
1	1	1	1	1	1	1

- Exemple : message $\overset{d_1}{0}\,\overset{d_2}{1}\,\overset{d_3}{0}\,\overset{d_4}{1}$

d_1	d_2	d_3	d_4	p_1	p_2	p_3
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	0	0	0
1	1	1	1	1	1	1

- Exemple : message $\overset{d_1}{0}\,\overset{d_2}{1}\,\overset{d_3}{0}\,\overset{d_4}{1}$

d_1	d_2	d_3	d_4	p_1	p_2	p_3
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	0	0	0
1	1	1	1	1	1	1

- Exemple : message $\overset{d_1}{0}\,\overset{d_2}{1}\,\overset{d_3}{0}\,\overset{d_4}{1}$
  - $c = \overset{p_1}{0}\,\overset{p_2}{1}\,\overset{d_1}{0}\,\overset{p_3}{0}\,\overset{d_2}{1}\,\overset{d_3}{0}\,\overset{d_4}{1}$

d_1	d_2	d_3	d_4	p_1	p_2	p_3
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	0	0	0
1	1	1	1	1	1	1

- Exemple : message $\begin{array}{cccc} d_1 & d_2 & d_3 & d_4 \\ 0 & 1 & 0 & 1 \end{array}$
  - $c = \begin{array}{ccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$
  - Three cases
    - Pas d'erreur :

      $r = \begin{array}{ccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$ est mot de code

d_1	d_2	d_3	d_4	p_1	p_2	p_3
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	0	0	0
1	1	1	1	1	1	1

- Exemple : message $\overset{d_1\, d_2\, d_3\, d_4}{0\,1\,0\,1}$
  - $c = \overset{p_1\, p_2\, d_1\, p_3\, d_2\, d_3\, d_4}{0\,1\,0\,0\,1\,0\,1}$
  - Three cases
    - Pas d'erreur : $r = \overset{p_1\, p_2\, d_1\, p_3\, d_2\, d_3\, d_4}{0\,1\,0\,0\,1\,0\,1}$ est mot de code
    - 1 erreur $r = \overset{p_1\, p_2\, d_2\, p_3\, d_2\, d_3\, d_4}{0\,1\,1\,0\,1\,0\,1}$ n'est pas mot de code

d_1	d_2	d_3	d_4	p_1	p_2	p_3
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	0	0	0
1	1	1	1	1	1	1

- Exemple : message $\begin{array}{cccc} d_1 & d_2 & d_3 & d_4 \\ 0 & 1 & 0 & 1 \end{array}$
  - $c = \begin{array}{ccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$
  - Three cases
    - Pas d'erreur : $r = \begin{array}{ccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$ est mot de code
    - 1 erreur $r = \begin{array}{ccccccc} p_1 & p_2 & d_2 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}$ n'est pas mot de code
      - Mot à distance minimum de r $c = \begin{array}{ccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$ correct

d_1	d_2	d_3	d_4	p_1	p_2	p_3
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	0	0	0
1	1	1	1	1	1	1

- Exemple : message $\begin{array}{cccc} d_1 & d_2 & d_3 & d_4 \\ 0 & 1 & 0 & 1 \end{array}$
  - $c = \begin{array}{ccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$
  - Three cases
    - Pas d'erreur :
      $r = \begin{array}{ccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$ est mot de code
    - 1 erreur
      $r = \begin{array}{ccccccc} p_1 & p_2 & d_2 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}$ n'est pas mot de code
      - Mot à distance minimum de r
        $c = \begin{array}{ccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$ correct
    - 2 erreurs
      $r = \begin{array}{ccccccc} p_1 & p_2 & d_2 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{array}$ n'est pas mot de code

d_1	d_2	d_3	d_4	p_1	p_2	p_3
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	0	0	0
1	1	1	1	1	1	1

- Exemple : message $\begin{array}{cccc} d_1 & d_2 & d_3 & d_4 \\ 0 & 1 & 0 & 1 \end{array}$
  - $c = \begin{array}{ccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$
  - Three cases
    - Pas d'erreur :
      $r = \begin{array}{ccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$ est mot de code
    - 1 erreur
      $r = \begin{array}{ccccccc} p_1 & p_2 & d_2 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}$ n'est pas mot de code
      - Mot à distance minimum de r
        $c = \begin{array}{ccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$ correct
    - 2 erreurs
      $r = \begin{array}{ccccccc} p_1 & p_2 & d_2 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{array}$ n'est pas mot de code
      - Mot à distance minimum de r
        $c = \begin{array}{ccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{array}$ faux

d_1	d_2	d_3	d_4	p_1	p_2	p_3
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	0	0	0
1	1	1	1	1	1	1

- Des codes plus compliqués existent
  - Par exemple le BCH (Bose, Ray-Chaudhuri et Hocquenghem)
    - Utilisé pour les communications satellites, les SSD, et les codes-barres