

TS226

-

# Codes convolutifs et codes concaténés associés

**Romain Tajan**

15 novembre 2019

# Plan

- 1 Previously on TS226 ...
  - ▷ Rappels sur l'encodeur
  - ▷ Rappels sur le diagramme d'état
  - ▷ Rappels sur le treillis
  - ▷ Rappels sur les décodeurs
  - ▷ Rappels sur l'algorithme de Viterbi
- 2 Décodage MAP-Bit des Codes Convolutifs
  - ▷ Introduction
  - ▷ Décodage MAP-Bit : algorithme BCJR
- 3 Codes concaténés - Turbocodes
- 4 Turbo-Codes

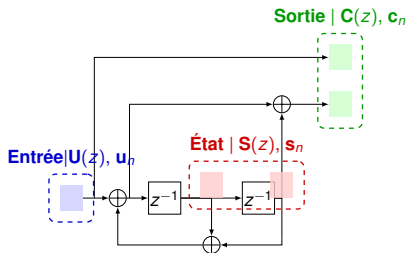
Comment avez-vous trouvé le cours précédent ?

- A Très difficile
- B Difficile
- C Moyen
- D Simple
- E Très simple

# Plan

- 1 Previously on TS226 ...
  - ▷ Rappels sur l'encodeur
  - ▷ Rappels sur le diagramme d'état
  - ▷ Rappels sur le treillis
  - ▷ Rappels sur les décodeurs
  - ▷ Rappels sur l'algorithme de Viterbi
- 2 Décodage MAP-Bit des Codes Convolutifs
- 3 Codes concaténés - Turbocodes
- 4 Turbo-Codes

## Rappels / définitions



• **Entrée** :  $\mathbf{U}(z) = [U^{(0)}(z), U^{(1)}(z) \dots U^{(n_b-1)}(z)]$

→ dans ce cours  $n_b = 1 \Rightarrow \mathbf{U}(z) \rightarrow U(z)$

• **État** :  $\mathbf{S}(z) = [S^{(0)}(z), S^{(1)}(z) \dots S^{(m-1)}(z)]$

→  $m$  est appelé "mémoire du code"

→  $\nu = m + 1$  est appelé "longueur de contrainte"

→ dans l'exemple  $m = 2$

→  $2^m$  : nombre d'états

• **Sortie** :  $\mathbf{C}(z) = [C^{(0)}(z), C^{(1)}(z) \dots C^{(m-1)}(z)]$

→  $n_s$  nombre de sorties

→ dans l'exemple  $n_s = 2$

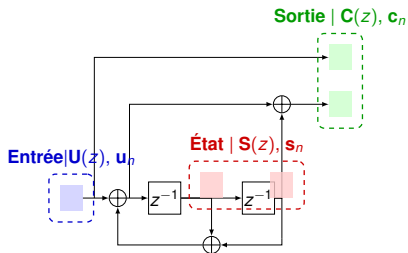
→ De façon générale :  $\mathbf{C}(z) = U(z)\mathbf{G}(z)$

→ où  $\mathbf{G}(z) = \begin{bmatrix} \frac{A^{(1)}(z)}{B^{(1)}(z)} & \dots & \frac{A^{(n_s)}(z)}{B^{(n_s)}(z)} \end{bmatrix}$

• **Rendement du code** :  $R = \frac{\text{\#bits d'info. en entrée}}{\text{\#bits codés en sortie}}$

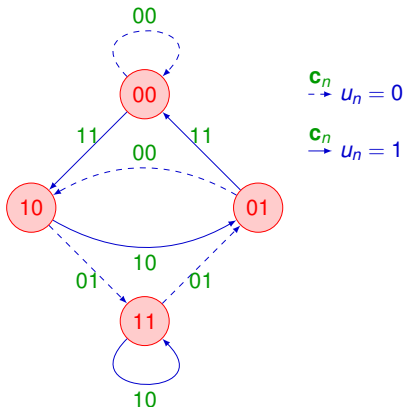
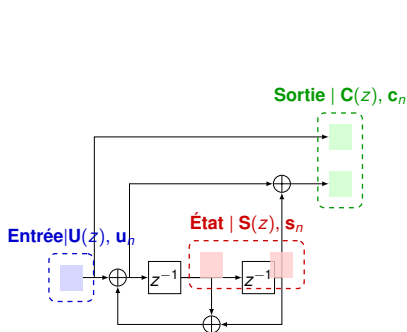
→ Ici :  $R = \frac{n_b}{n_s} = \frac{1}{2}$

# Rappels / définitions

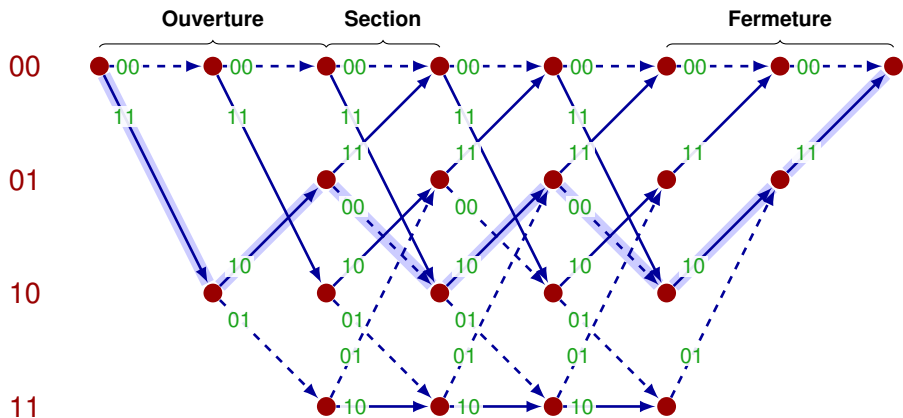


- Code linéaire
- Encodeur récursif / non récursif
- Encodeur systématique / non systématique
- Notation octale

# Rappels / définitions



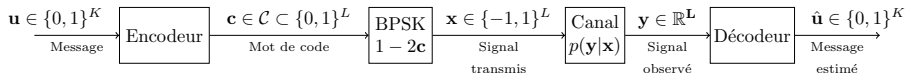
- Message  $\Leftrightarrow$  chemin dans le graphe
- Mot de code  $\Leftrightarrow$  étiquettes le long du chemin dans le graphe
- Nécessité de définir (au moins) un état initial



- On souhaite envoyer le message :  $\mathbf{u} = [1, 1, 0, 1, 0]$
- On calcule le mot de code :  $\mathbf{c} = [1\ 1, 1\ 0, 0\ 0, 1\ 0, 0\ 0, \underline{1\ 0}, \underline{1\ 1}]$
- On envoie le signal :  $\mathbf{x} = [-1\ -1, -1\ 1, 1\ 1, -1\ 1, 1\ 1, \underline{-1\ 1}, \underline{-1\ -1}]$



# Décodage des Codes convolutifs



- Le **Max. A Posteriori (MAP)** :  $\hat{\mathbf{u}} = \underset{\mathbf{u} \in \{0,1\}^K}{\operatorname{argmax}} \mathbb{P}(\mathbf{U} = \mathbf{u} | \mathbf{y})$

→ Minimise la probabilité d'erreur trame

- Le **Max. de vraisemblance (ML)** :  $\hat{\mathbf{u}} = \underset{\mathbf{u} \in \{0,1\}^K}{\operatorname{argmax}} p(\mathbf{y}|\mathbf{u}) = \underset{\mathbf{c} \in \mathcal{C} | \mathbf{u} \in \{0,1\}^K}{\operatorname{argmin}} \sum_{\ell=0}^{L-1} \mathbf{c}_{\ell} \mathbf{y}_{\ell}^T$

→ **CC** : Équivalent moins complexe du MAP, algorithme de Viterbi

- Le **Max. A Posteriori - Bit (MAP-Bit)** :  $\hat{u}_i = \underset{u_i \in \{0,1\}}{\operatorname{argmax}} \mathbb{P}(U_i = u_i | \mathbf{y})$

→ Minimise la probabilité d'erreur binaire

- Nombre de bits dans le message :  $K$
- Taille message + fermeture :  $L = K + m$
- Le message envoyé :  $\mathbf{u} = [u_0, u_1, \dots, u_{K-1}]$
- Le mot de code émis :  $\mathbf{c} = [\mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_{L-1}]$ , où  $\mathbf{c}_\ell = [c_\ell^{(0)}, c_\ell^{(1)}]$
- Le signal observé :  $\mathbf{y} = [\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{L-1}]$ , où  $\mathbf{y}_\ell = [y_\ell^{(0)}, y_\ell^{(1)}]$
- **Décodeur ML :**  
$$\hat{\mathbf{c}} = \underset{\mathbf{c} \in \mathcal{C}}{\operatorname{argmin}} \sum_{\ell=0}^{L-1} \mathbf{c}_\ell \mathbf{y}_\ell^T$$
- Le message reçu :  $\hat{\mathbf{u}} = [\hat{u}_0, \hat{u}_1, \dots, \hat{u}_{K-1}] = \phi^{-1}(\hat{\mathbf{c}})$

• **Solution 1** : explorer tous les messages (eq. tous les mots de codes)

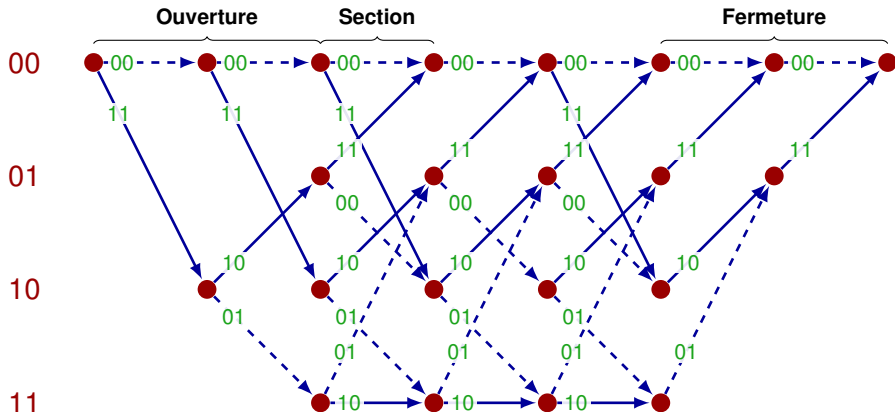
- Nombre de bits dans le message :  $K$
- Taille message + fermeture :  $L = K + m$
- Le message envoyé :  $\mathbf{u} = [u_0, u_1, \dots, u_{K-1}]$
- Le mot de code émis :  $\mathbf{c} = [\mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_{L-1}]$ , où  $\mathbf{c}_\ell = [c_\ell^{(0)}, c_\ell^{(1)}]$
- Le signal observé :  $\mathbf{y} = [\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{L-1}]$ , où  $\mathbf{y}_\ell = [y_\ell^{(0)}, y_\ell^{(1)}]$
- **Décodeur ML** :  
$$\hat{\mathbf{c}} = \underset{\mathbf{c} \in \mathcal{C}}{\operatorname{argmin}} \sum_{\ell=0}^{L-1} \mathbf{c}_\ell \mathbf{y}_\ell^T$$
- Le message reçu :  $\hat{\mathbf{u}} = [\hat{u}_0, \hat{u}_1, \dots, \hat{u}_{K-1}] = \phi^{-1}(\hat{\mathbf{c}})$

• **Solution 1** : explorer tous les messages (eq. tous les mots de codes)  $\rightarrow$  **il y en a  $2^K$ ...**

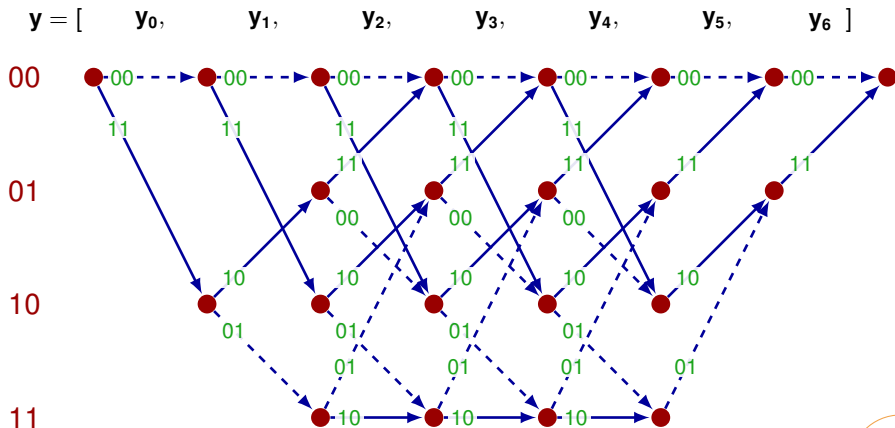
- Nombre de bits dans le message :  $K$
- Taille message + fermeture :  $L = K + m$
- Le message envoyé :  $\mathbf{u} = [u_0, u_1, \dots, u_{K-1}]$
- Le mot de code émis :  $\mathbf{c} = [\mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_{L-1}]$ , où  $\mathbf{c}_\ell = [c_\ell^{(0)}, c_\ell^{(1)}]$
- Le signal observé :  $\mathbf{y} = [\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{L-1}]$ , où  $\mathbf{y}_\ell = [y_\ell^{(0)}, y_\ell^{(1)}]$
- **Décodeur ML** :
$$\hat{\mathbf{c}} = \underset{\mathbf{c} \in \mathcal{C}}{\operatorname{argmin}} \sum_{\ell=0}^{L-1} \mathbf{c}_\ell \mathbf{y}_\ell^T$$
- Le message reçu :  $\hat{\mathbf{u}} = [\hat{u}_0, \hat{u}_1, \dots, \hat{u}_{K-1}] = \phi^{-1}(\hat{\mathbf{c}})$

- **Solution 1** : explorer tous les messages (eq. tous les mots de codes)  $\rightarrow$  **il y en a  $2^K$ ...**
- **Solution 2** : utiliser le treillis + la forme de la fonction de coût  $\rightarrow$  **algorithme de Viterbi**

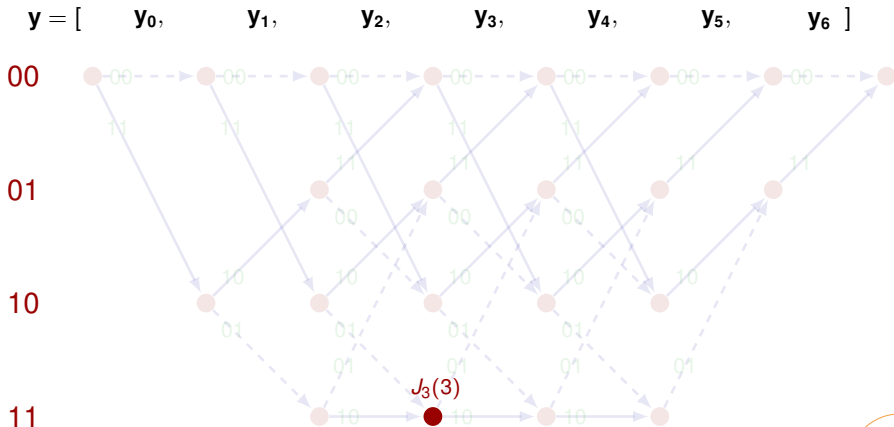
- Soit  $J_n(s_n) = \min_{\mathbf{c} \in \mathcal{C}(s_0 \rightarrow s_n)} \sum_{\ell=0}^{n-1} \mathbf{c}_\ell \mathbf{y}_\ell^T$
- Le décodage ML est équivalent à trouver l'antécédent de  $J_L(s_L)$  où  $s_0 = s_L = 0$



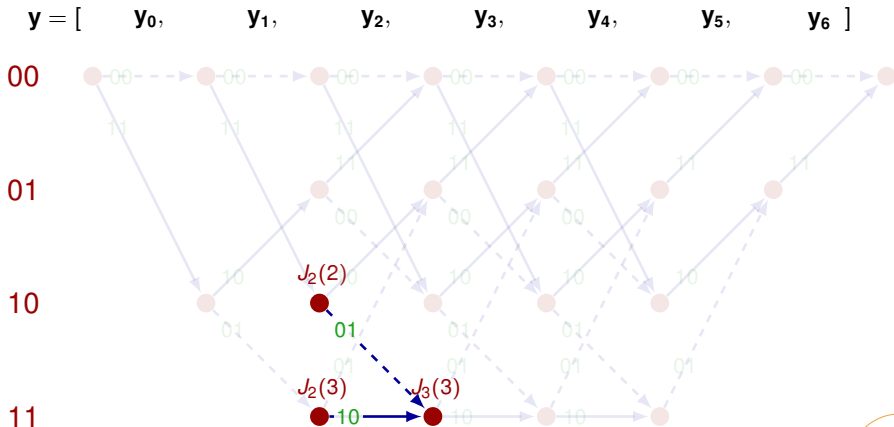
- $\mathcal{P}(s_n)$  : **parents de l'état**  $s_n$ , i.e. ensemble des  $s_{n-1}$  tels que  $s_{n-1} \rightarrow s_n$  existe
- **Algorithme de Viterbi** : pour chaque  $n \in [0, L - 1]$  et chaque  $s_n \in \mathcal{S}_n$  calculer
 
$$J_n(s_n) = \min_{s_{n-1} \in \mathcal{P}(s_n)} \left[ J_{n-1}(s_{n-1}) + \mathbf{c}_{n-1}(s_{n-1} \rightarrow s_n) \mathbf{y}_{n-1}^T \right]$$
- $\hat{\mathbf{u}}$  : **chemin survivant minimisant**  $J_L(s_L)$



- $\mathcal{P}(s_n)$  : **parents de l'état**  $s_n$ , i.e. ensemble des  $s_{n-1}$  tels que  $s_{n-1} \rightarrow s_n$  existe
- **Algorithme de Viterbi** : pour chaque  $n \in [0, L-1]$  et chaque  $s_n \in \mathcal{S}_n$  calculer
 
$$J_n(s_n) = \min_{s_{n-1} \in \mathcal{P}(s_n)} [J_{n-1}(s_{n-1}) + \mathbf{c}_{n-1}(s_{n-1} \rightarrow s_n) \mathbf{y}_{n-1}^T]$$
- $\hat{\mathbf{u}}$  : **chemin survivant minimisant**  $J_L(s_L)$

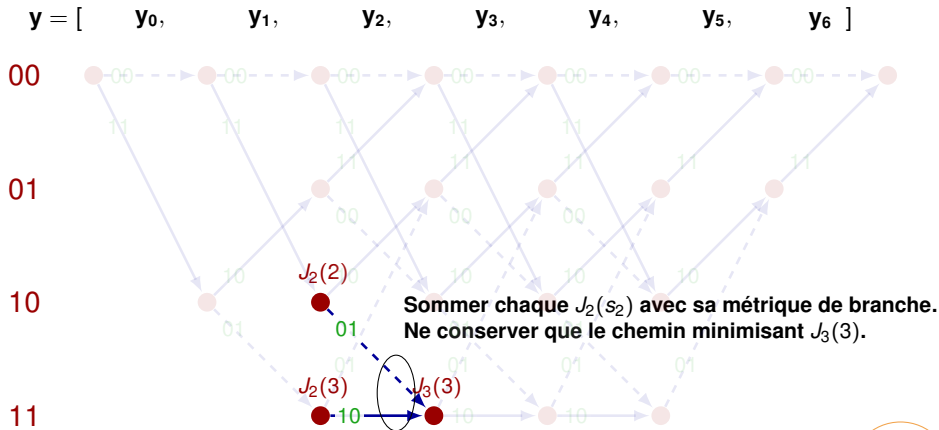


- $\mathcal{P}(s_n)$  : **parents de l'état**  $s_n$ , i.e. ensemble des  $s_{n-1}$  tels que  $s_{n-1} \rightarrow s_n$  existe
- **Algorithme de Viterbi** : pour chaque  $n \in [0, L - 1]$  et chaque  $s_n \in \mathcal{S}_n$  calculer
 
$$J_n(s_n) = \min_{s_{n-1} \in \mathcal{P}(s_n)} [J_{n-1}(s_{n-1}) + \mathbf{c}_{n-1}(s_{n-1} \rightarrow s_n) \mathbf{y}_{n-1}^T]$$
- $\hat{\mathbf{u}}$  : **chemin survivant minimisant**  $J_L(s_L)$



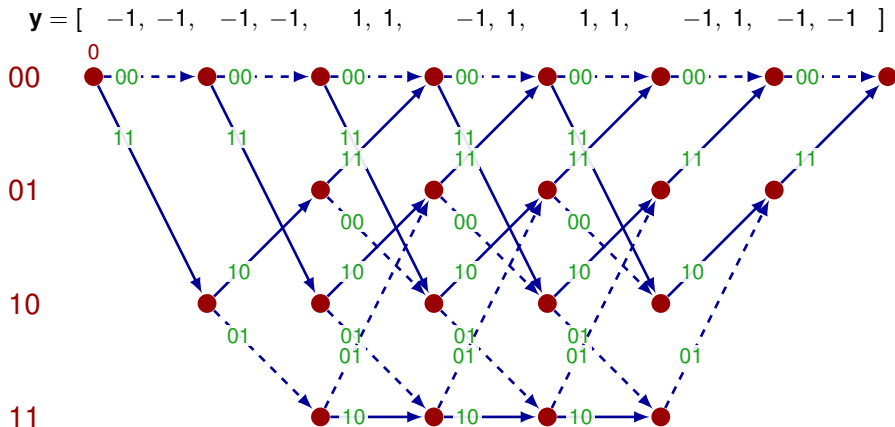


- $\mathcal{P}(s_n)$  : **parents de l'état**  $s_n$ , i.e. ensemble des  $s_{n-1}$  tels que  $s_{n-1} \rightarrow s_n$  existe
- **Algorithme de Viterbi** : pour chaque  $n \in [0, L - 1]$  et chaque  $s_n \in \mathcal{S}_n$  calculer 
$$J_n(s_n) = \min_{s_{n-1} \in \mathcal{P}(s_n)} \left[ J_{n-1}(s_{n-1}) + \mathbf{c}_{n-1}(s_{n-1} \rightarrow s_n) \mathbf{y}_{n-1}^T \right]$$
- $\hat{u}$  : **chemin survivant minimisant**  $J_L(s_L)$



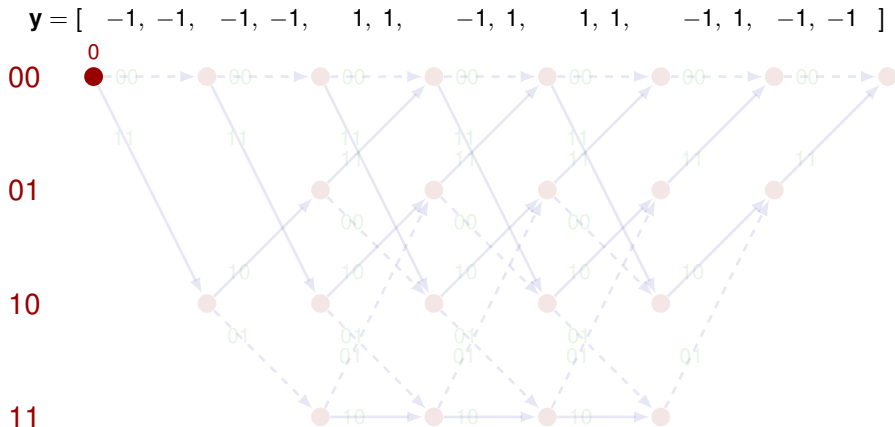
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



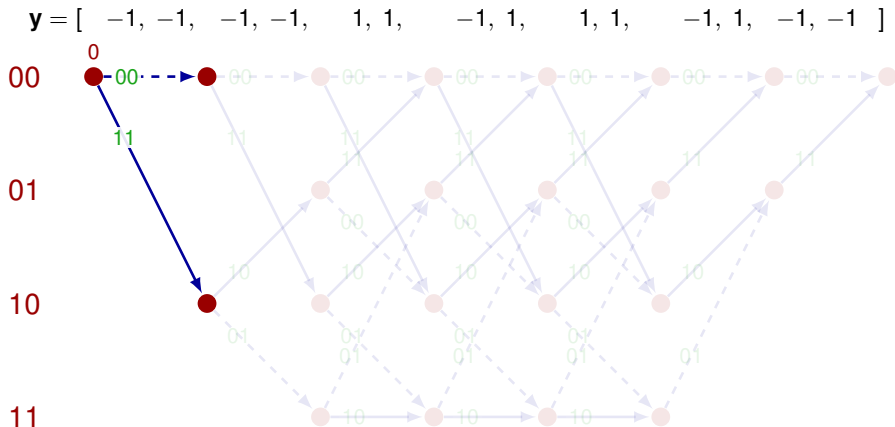
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



## Exemple de question...

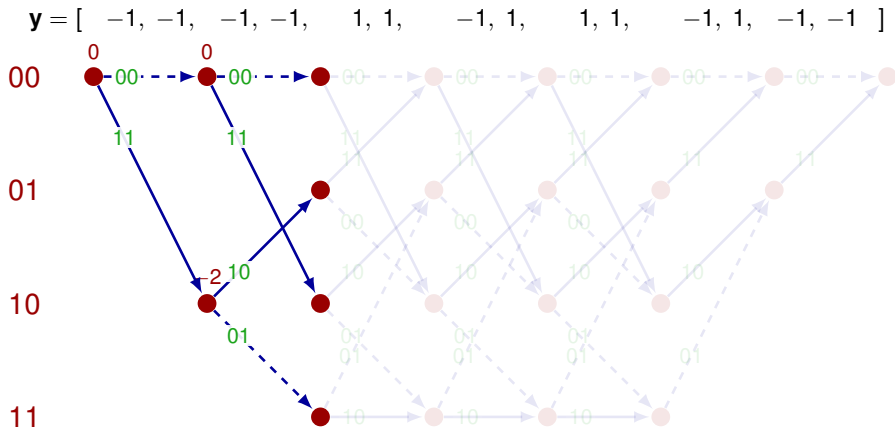
Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?





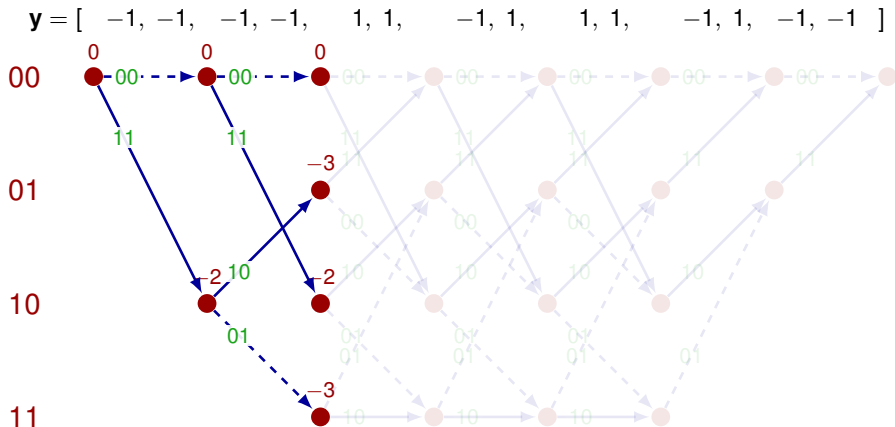
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



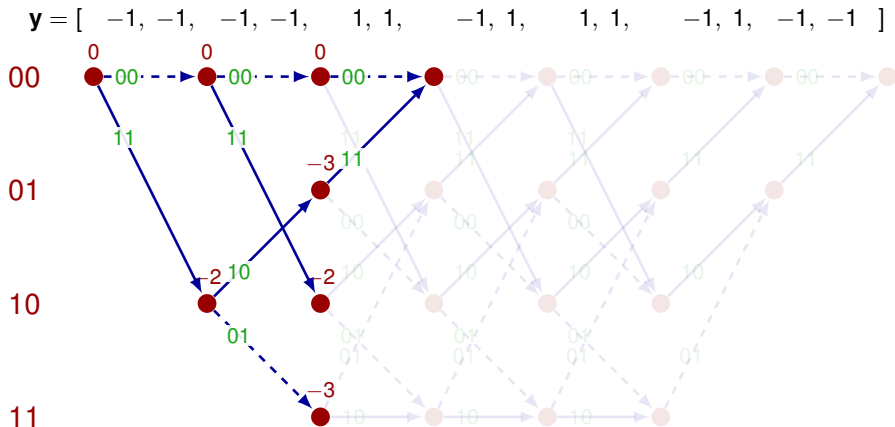
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



## Exemple de question...

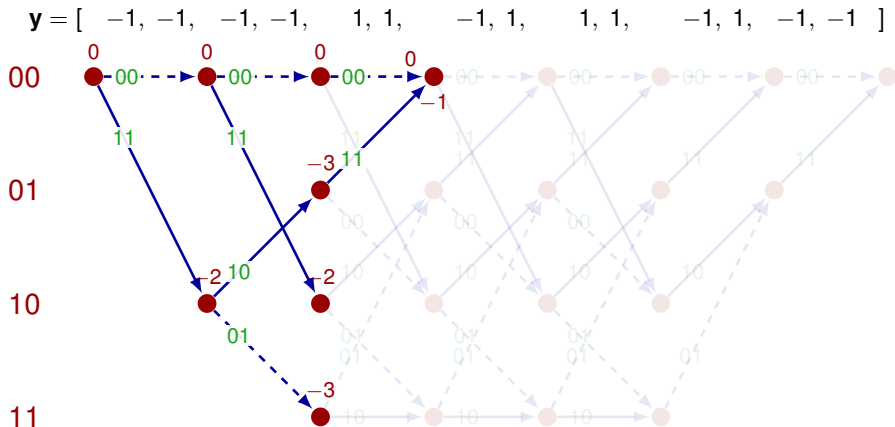
Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?





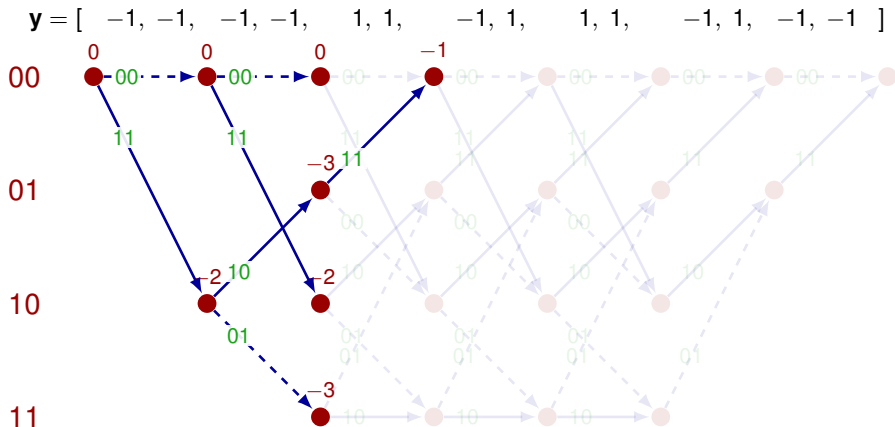
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



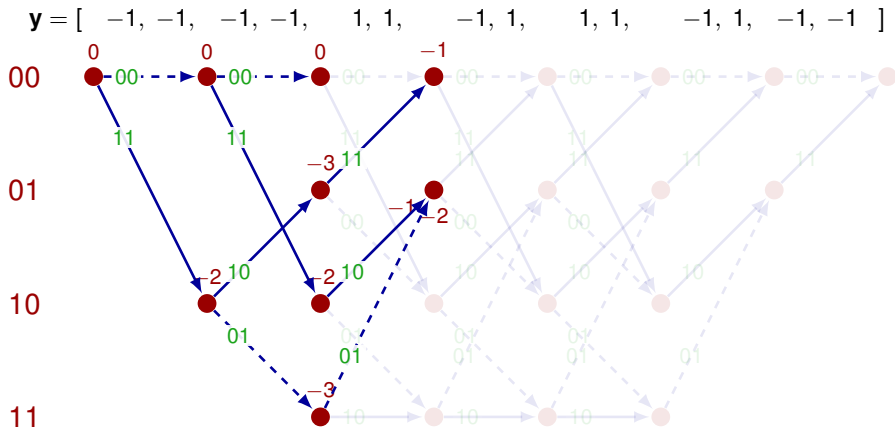
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



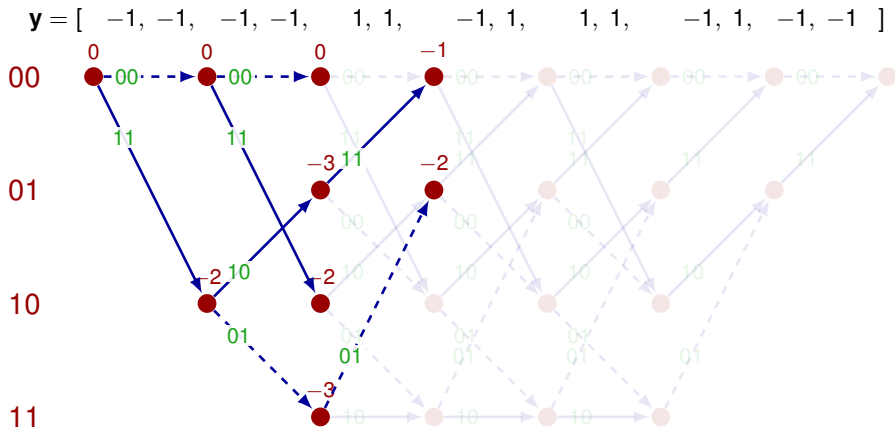
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



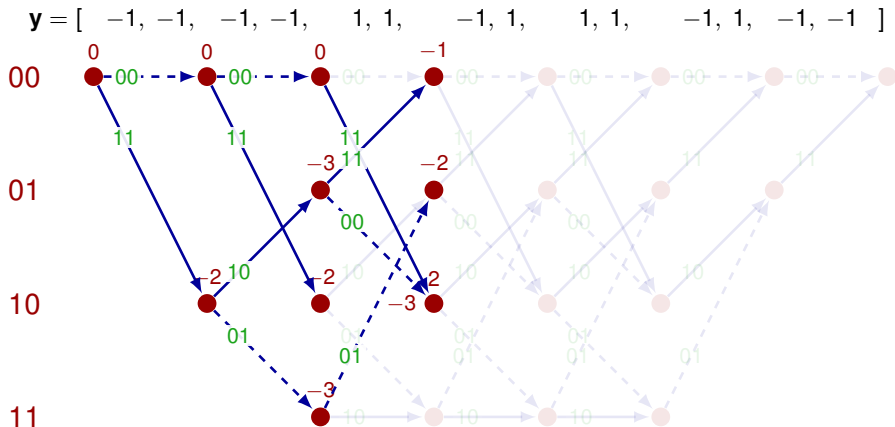
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



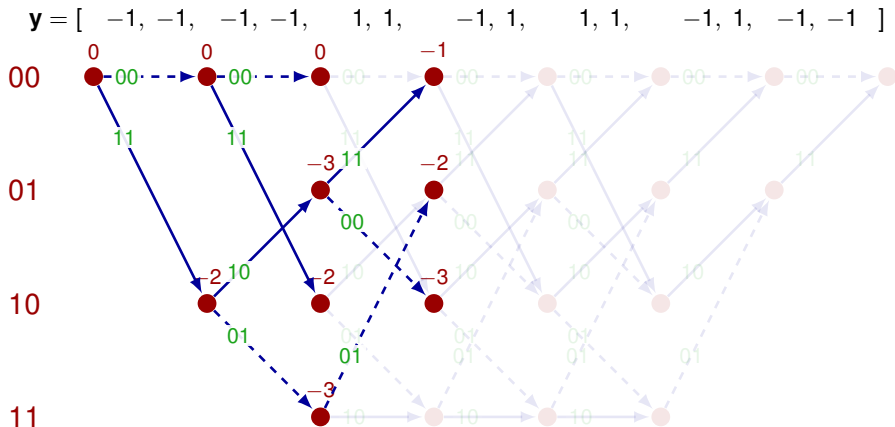
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



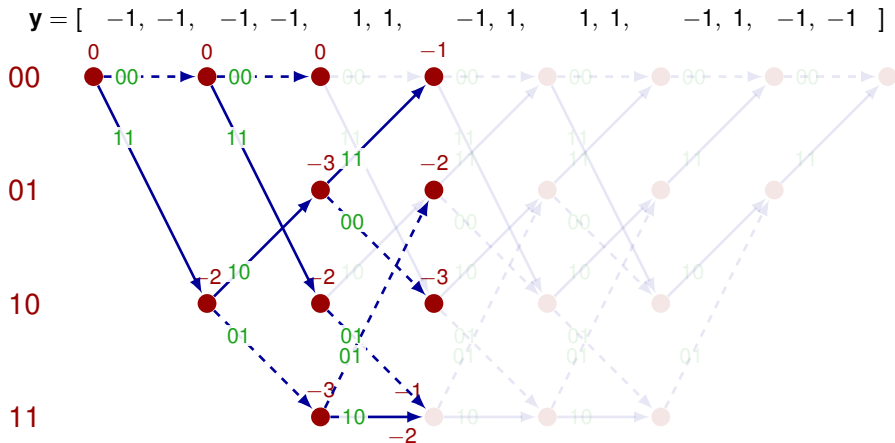
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



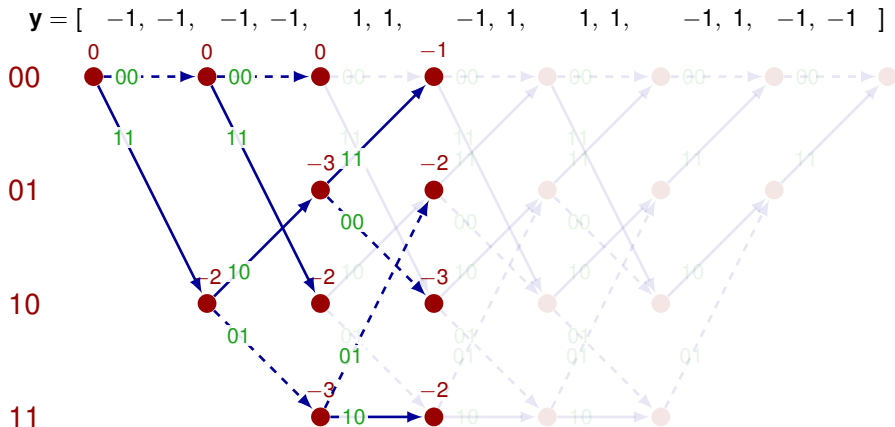
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



## Exemple de question...

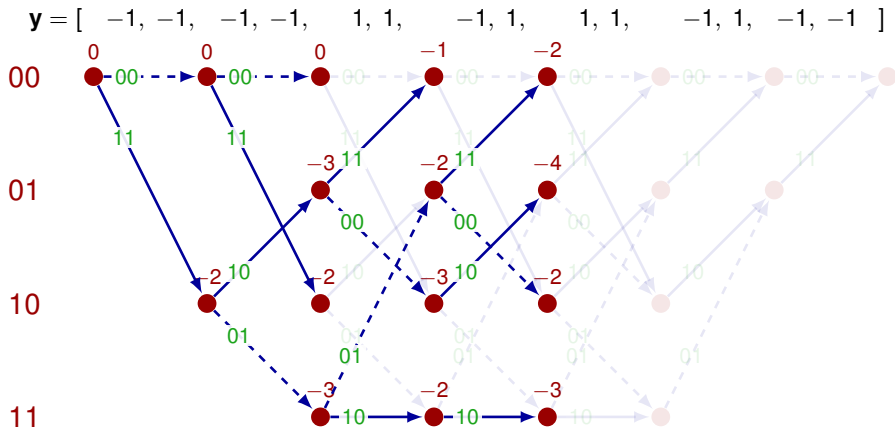
Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?





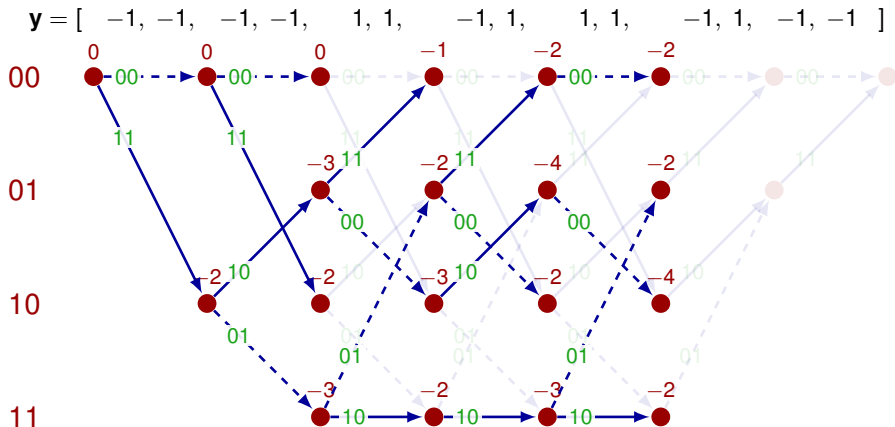
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



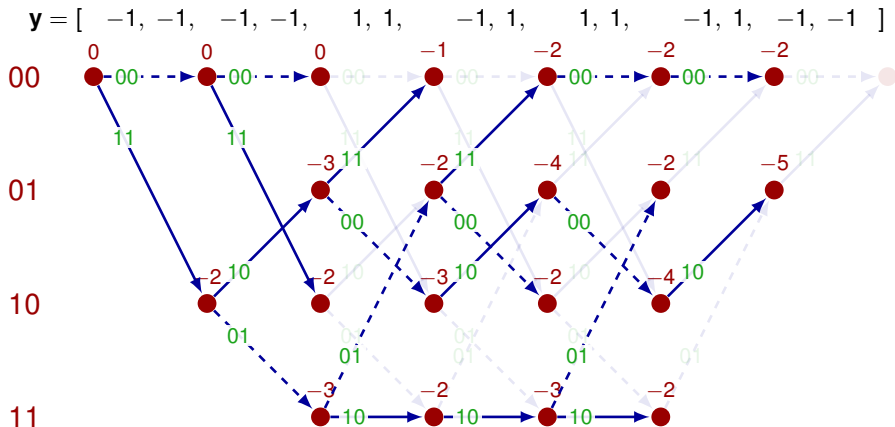
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



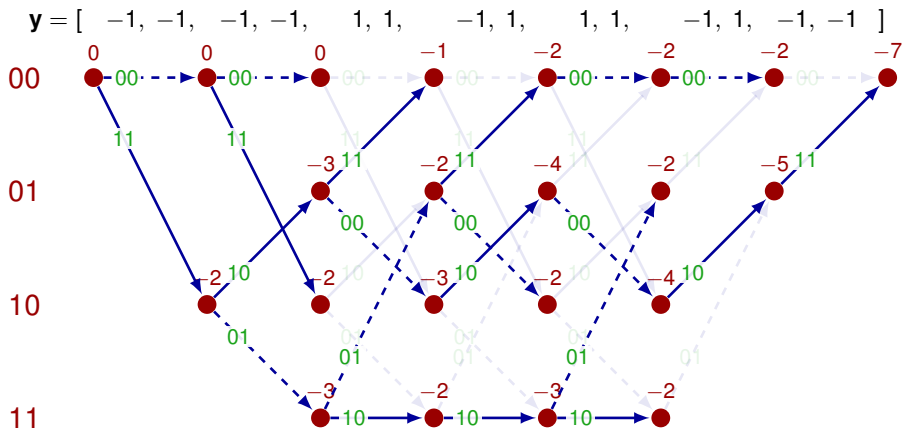
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



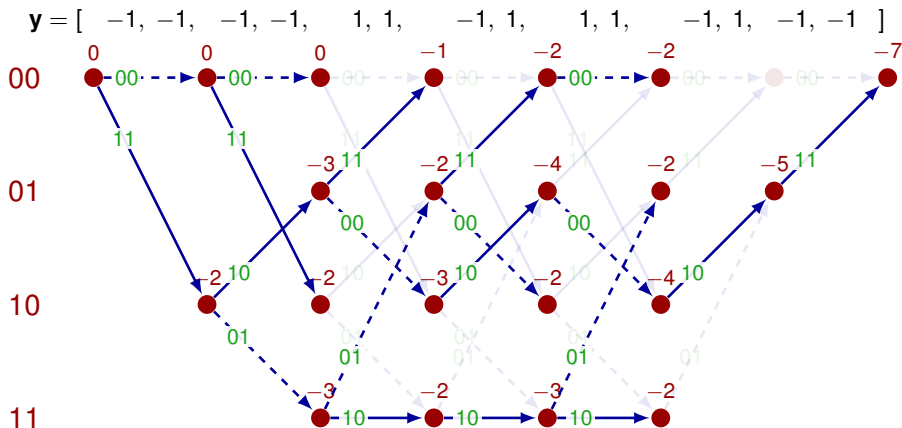
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



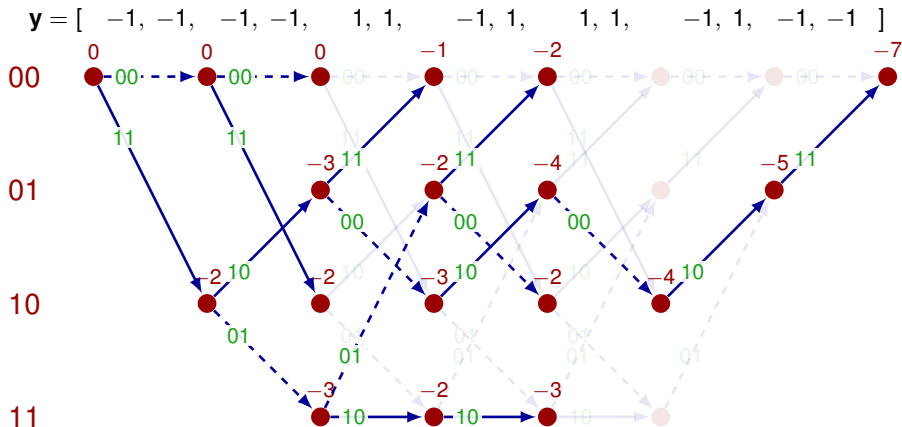
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



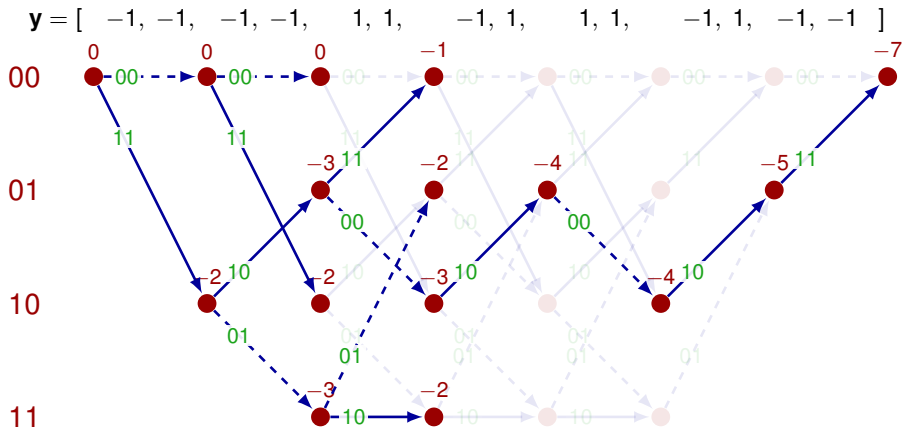
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



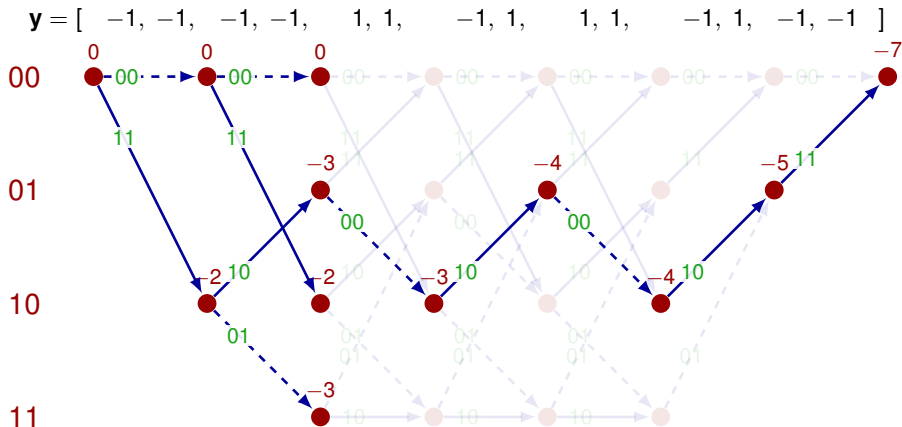
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



## Exemple de question...

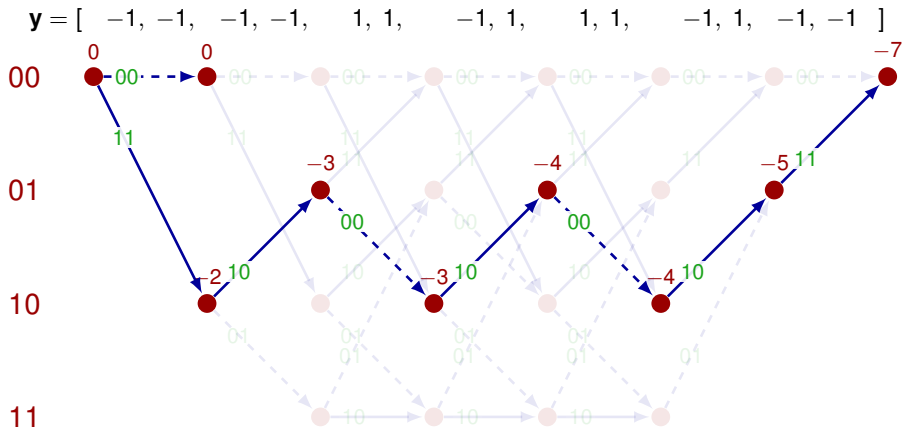
Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?





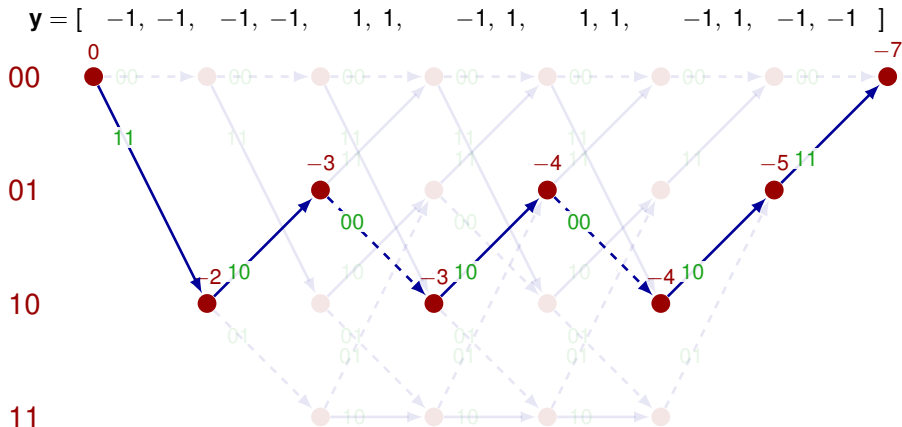
## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



## Exemple de question...

Soit le signal observé  $\mathbf{y} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1]$ , sachant que  $\mathbf{u} = [1, 1, 0, 1, 0]$ , reçoit-on le message sans erreur ?



# Plan

- 1 Previously on TS226 ...
- 2 **Décodage MAP-Bit des Codes Convolutifs**
  - ▷ Introduction
  - ▷ Décodage MAP-Bit : algorithme BCJR
- 3 Codes concaténés - Turbocodes
- 4 Turbo-Codes

# Introduction



- Le **décodeur** du **Maximum A Posteriori - Bit (MAP-Bit)** est la fonction de  $\mathbf{y}$  définie par

$$\hat{u}_\ell = \arg \max_{u_\ell \in \{0,1\}} \mathbb{P}(U_\ell = u_\ell | \mathbf{y})$$

- Le **décodeur MAP-Bit** minimise la probabilité d'erreur binaire.

- En pratique, on calcule le Logarithme du Rapport de Vraisemblance (LLR) suivant :

$$L(u_\ell) = \log \left( \frac{\mathbb{P}(U_\ell = 0 | \mathbf{y})}{\mathbb{P}(U_\ell = 1 | \mathbf{y})} \right)$$

- Le signe de  $L(u_\ell)$  **représente la décision** :  $\hat{u}_\ell = \begin{cases} 0, & \text{si } L(u_\ell) \geq 0 \\ 1, & \text{sinon.} \end{cases}$
- Le **module** de  $L(u_\ell)$  i.e.  $|L(u_\ell)|$  représente la **fiabilité** de la décision.
- Un décodeur produisant de telles valeurs est dit **soft**.

# Algorithme BCJR (Bahl Cocke Jelinek Raviv)

L'algorithme BCJR repose sur deux principes :

- 1 Les LLR ( $L(u_\ell)$ ) peuvent être calculés sur le treillis :

$$L(u_\ell) = \log \left( \frac{\sum_{(s \rightarrow s') \in \mathcal{U}_0} \alpha_\ell(s) \gamma_\ell(s, s') \beta_{\ell+1}(s')}{\sum_{(s \rightarrow s') \in \mathcal{U}_1} \alpha_\ell(s) \gamma_\ell(s, s') \beta_{\ell+1}(s')} \right)$$

où

- $\alpha_\ell(s) = p(s_\ell = s, \mathbf{y}_0^{\ell-1})$  est une **métrique de nœud** appelée **métrique aller**
- $\beta_\ell(s) = p(\mathbf{y}_\ell^{L-1} | s_\ell = s)$  est une **métrique de nœud** appelée **métrique retour**
- $\gamma_\ell(s, s') = p(s_{\ell+1} = s' | s_\ell = s, \mathbf{y}_\ell)$  est une **métrique de branche**

- 2 Les métriques de nœuds et de branches se calculent elles-mêmes sur le treillis :

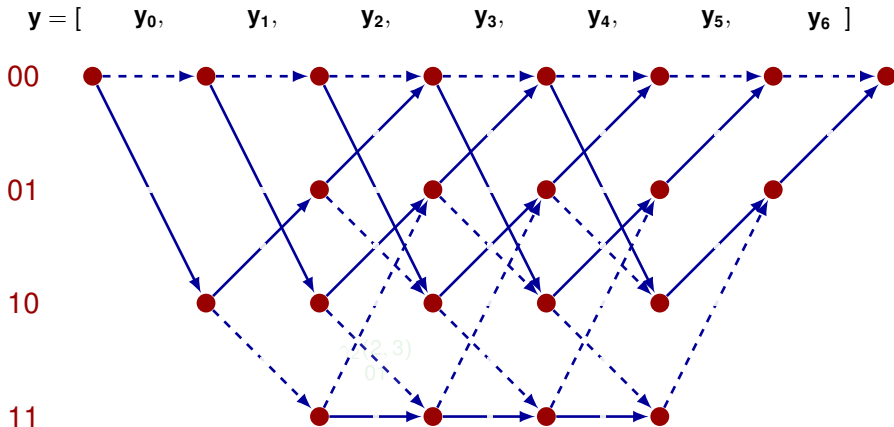
$$\alpha_{\ell+1}(s') = \sum_{s \in \mathcal{S}} \alpha_\ell(s) \gamma_\ell(s, s')$$

$$\beta_\ell(s) = \sum_{s' \in \mathcal{S}} \beta_{\ell+1}(s') \gamma_\ell(s, s')$$

$$\gamma_\ell(s, s') = \frac{p(u_\ell(s \rightarrow s'))}{2\pi\sigma^2} \exp \left( -\frac{\|\mathbf{y}_\ell - \mathbf{x}_\ell(s \rightarrow s')\|^2}{2\sigma^2} \right)$$

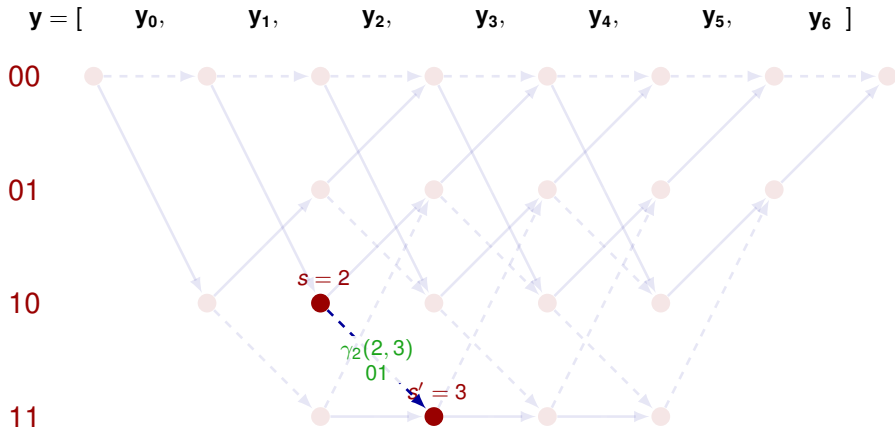
# Métrique de branche $\gamma_\ell(s, s')$

$$\gamma_\ell(s, s') = \frac{p(u_\ell(s \rightarrow s'))}{2\pi\sigma^2} \exp\left(-\frac{\|y_\ell - \mathbf{x}_\ell(s \rightarrow s')\|^2}{2\sigma^2}\right)$$



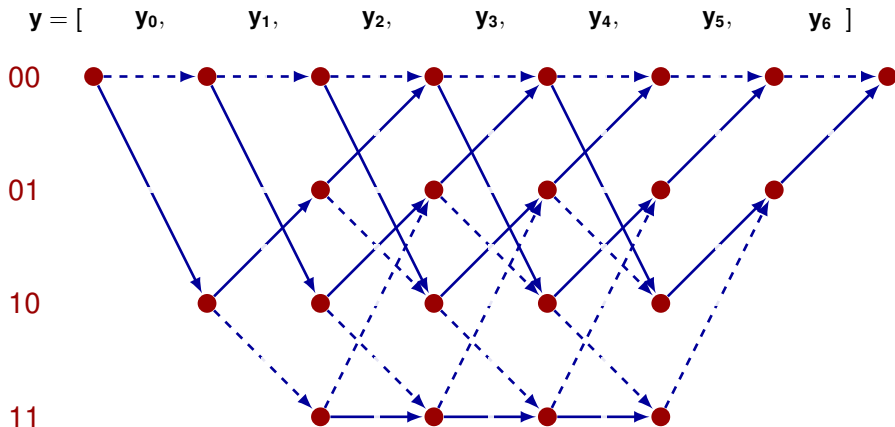
# Métrique de branche $\gamma_\ell(s, s')$

$$\gamma_\ell(s, s') = \frac{p(u_\ell(s \rightarrow s'))}{2\pi\sigma^2} \exp\left(-\frac{\|\mathbf{y}_\ell - \mathbf{x}_\ell(s \rightarrow s')\|^2}{2\sigma^2}\right)$$



# Récursion "aller" : calcul des $\alpha_\ell(s)$

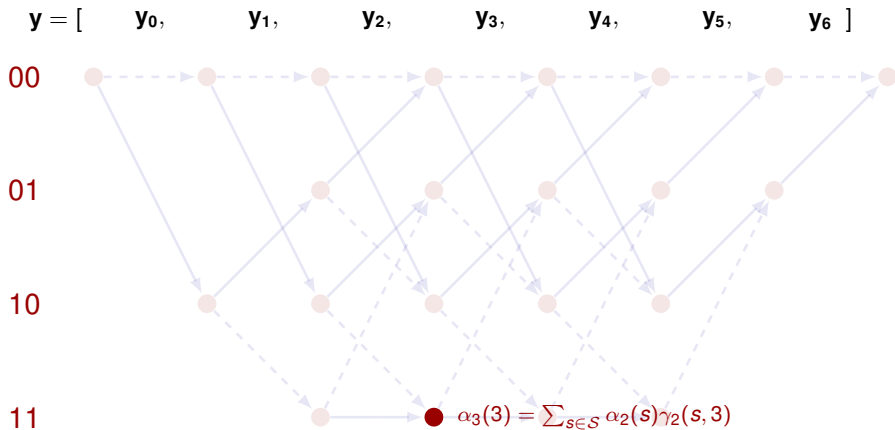
$$\alpha_{\ell+1}(s') = \sum_{s \in \mathcal{S}} \alpha_\ell(s) \gamma_\ell(s, s')$$





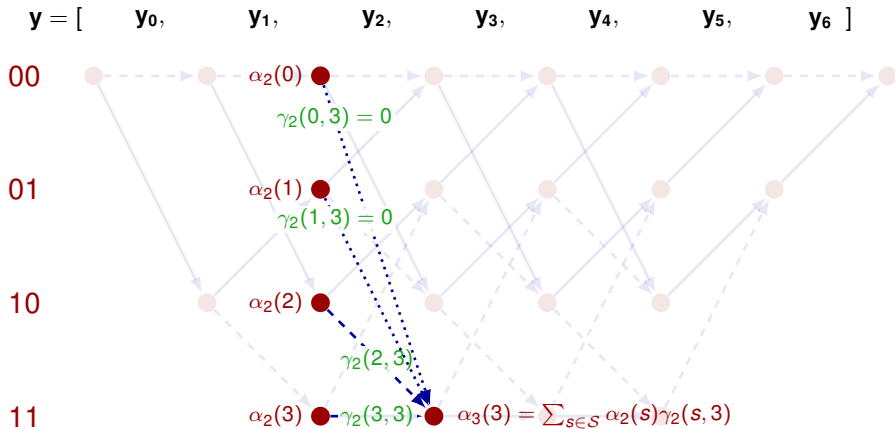
# Récursion "aller" : calcul des $\alpha_\ell(s)$

$$\alpha_{\ell+1}(s') = \sum_{s \in \mathcal{S}} \alpha_\ell(s) \gamma_\ell(s, s')$$



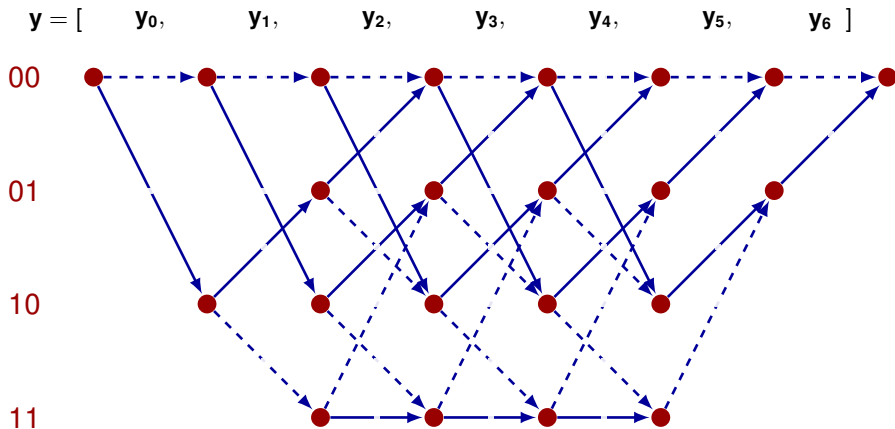
# Récursion "aller" : calcul des $\alpha_\ell(s)$

$$\alpha_{\ell+1}(s') = \sum_{s \in \mathcal{S}} \alpha_\ell(s) \gamma_\ell(s, s')$$



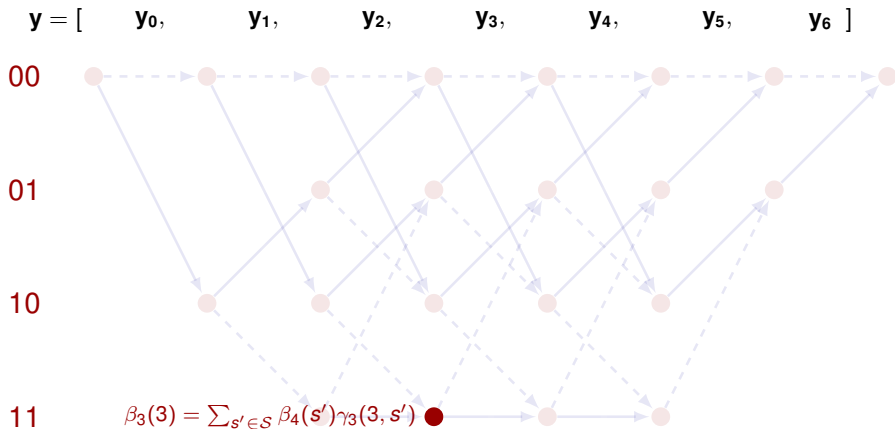
# Récursion "retour" : calcul des $\beta_\ell(s)$

$$\beta_\ell(s) = \sum_{s' \in \mathcal{S}} \beta_{\ell+1}(s') \gamma_\ell(s, s')$$



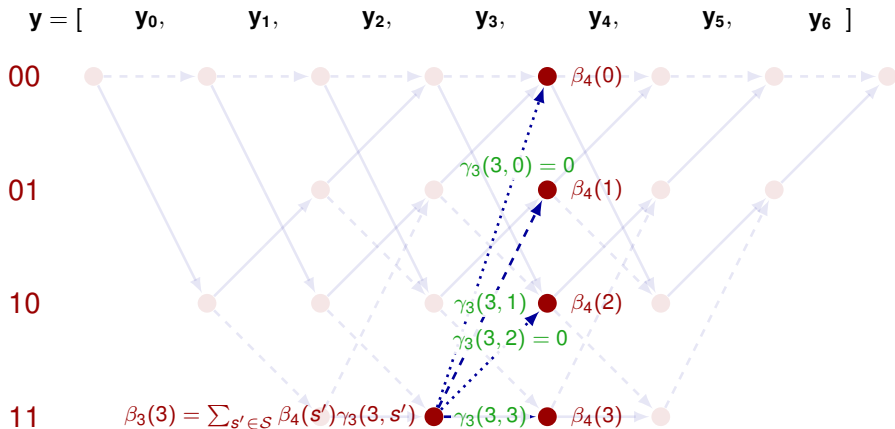
# Récursion "retour" : calcul des $\beta_\ell(s)$

$$\beta_\ell(s) = \sum_{s' \in \mathcal{S}} \beta_{\ell+1}(s') \gamma_\ell(s, s')$$



# Récursion "retour" : calcul des $\beta_\ell(s)$

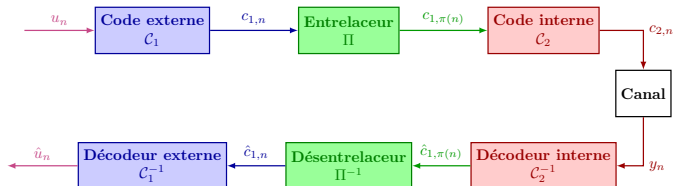
$$\beta_\ell(s) = \sum_{s' \in \mathcal{S}} \beta_{\ell+1}(s') \gamma_\ell(s, s')$$



# Plan

- 1 Previously on TS226 ...
- 2 Décodage MAP-Bit des Codes Convolutifs
- 3 Codes concaténés - Turbocodes**
- 4 Turbo-Codes

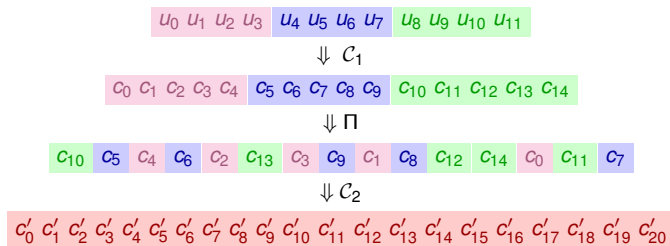
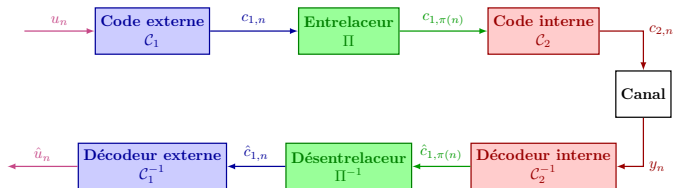
## Concaténation série - Définition



### Remarques

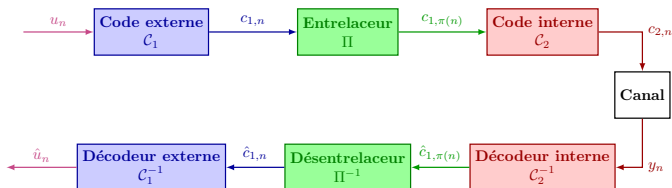
- Jusqu'aux années 1980 :
  - $C_1$  : **code linéaire en bloc** corrigeant jusqu'à  $t$  erreurs (code RS, code BCH)
  - $C_2$  : **code convolutif**
- Rendement** :  $R = R_1 R_2$
- Entrelaceur** : permute les éléments de  $c_1$
- Désentrelaceur** : remet les éléments à leur place avant décodage
  - Permet de disperser les erreurs commises en **rafale** par  $C_2^{-1}$  pour qu'elles apparaissent **isolées** et soient corrigées par  $C_1^{-1}$

## Concaténation série - Émetteur

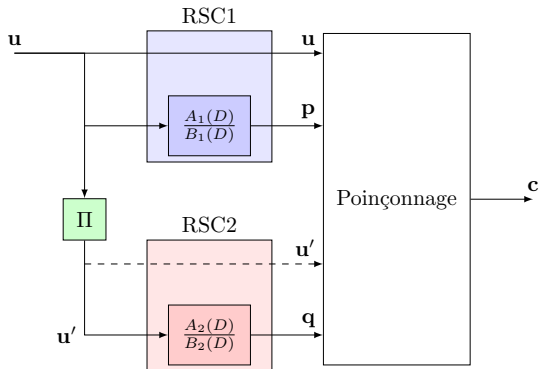




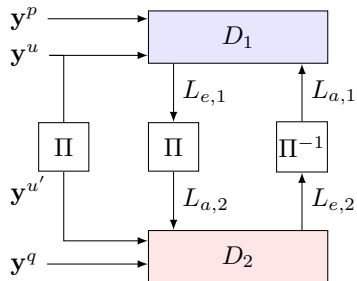
## Concaténation série - Récepteur


 $y_0 \ y_1 \ y_2 \ y_3 \ y_4 \ y_5 \ y_6 \ y_7 \ y_8 \ y_9 \ y_{10} \ y_{11} \ y_{12} \ y_{13} \ y_{14} \ y_{15} \ y_{16} \ y_{17} \ y_{18} \ y_{19} \ y_{20}$ 
 $\Downarrow C_2^{-1}$ 
 $\hat{c}_{10} \ \hat{c}_5 \ \hat{c}_4 \ \hat{c}_6 \ \hat{c}_2 \ \hat{c}_{13} \ \hat{c}_3 \ \hat{c}_9 \ \hat{c}_1 \ \hat{c}_8 \ \hat{c}_{12} \ \hat{c}_{14} \ \hat{c}_0 \ \hat{c}_{11} \ \hat{c}_7$ 
 $\Downarrow \Pi^{-1}$ 
 $\hat{c}_0 \ \hat{c}_1 \ \hat{c}_2 \ \hat{c}_3 \ \hat{c}_4 \ \hat{c}_5 \ \hat{c}_6 \ \hat{c}_7 \ \hat{c}_8 \ \hat{c}_9 \ \hat{c}_{10} \ \hat{c}_{11} \ \hat{c}_{12} \ \hat{c}_{13} \ \hat{c}_{14}$ 
 $\Downarrow C_1^{-1}$ 
 $u_0 \ u_1 \ u_2 \ u_3 \ u_4 \ u_5 \ u_6 \ u_7 \ u_8 \ u_9 \ u_{10} \ u_{11}$

# Turbocode et concaténation parallèle



# Turbocode et concaténation parallèle



# Dernier QCM

Comment avez-vous trouvé ce cours ?

- ☐ A Très difficile
- ☐ B Difficile
- ☐ C Moyen
- ☐ D Simple
- ☐ E Très simple

#QDLE#S#ABCDE#30#

# Plan

- 1 Previously on TS226 ...
- 2 Décodage MAP-Bit des Codes Convolutifs
- 3 Codes concaténés - Turbocodes
- 4 Turbo-Codes**