TS226
-
Codes convolutifs et codes concaténés associés

Romain Tajan

1er octobre 2020

Plan

QCM

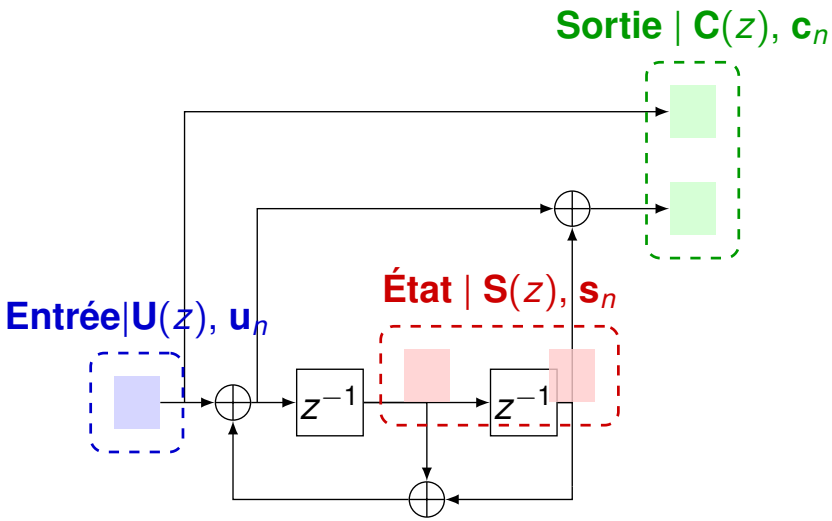
Comment avez-vous trouvé le cours précédent ?

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

#QDLE#S#ABCDE#30#

Plan

Rappels / définitions



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

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

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

$\rightarrow$ m est appelé "mémoire du code"

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

$\rightarrow$ dans l'exemple $m = 2$

$\rightarrow$ 2^m : nombre d'états

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

$\rightarrow$ n_s nombre de sorties

$\rightarrow$ dans l'exemple $n_s = 2$

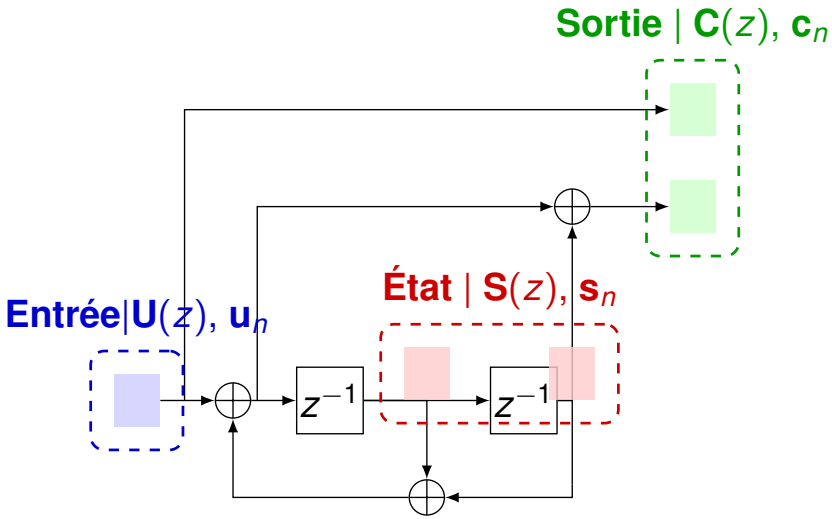
$\rightarrow$ De façon générale : $\mathbf{C}(z) = U(z)\mathbf{G}(z)$

$\rightarrow$ où $\mathbf{G}(z) = [\frac{A^{(1)}(z)}{B^{(1)}(z)}\ \ldots\ \frac{A^{(n_s)}(z)}{B^{(n_s)}(z)}]$

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

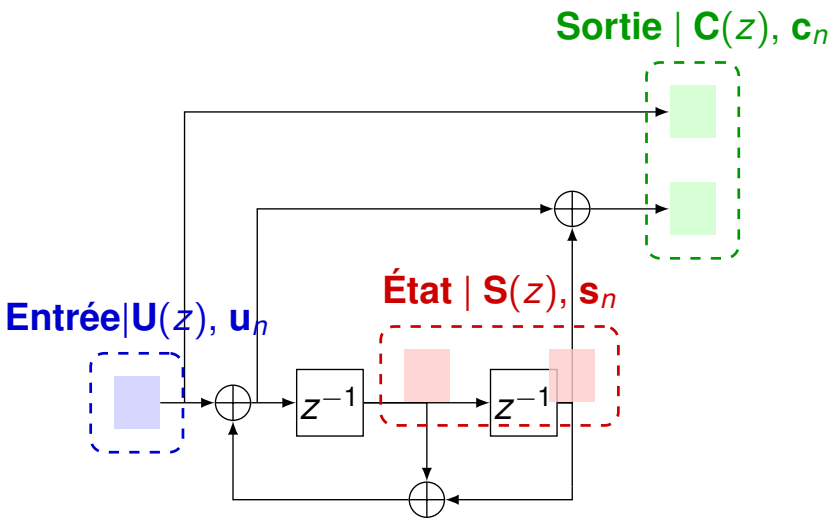
$\rightarrow$ Ici : $R = \frac{n_b}{n_s} = \frac{1}{2}$

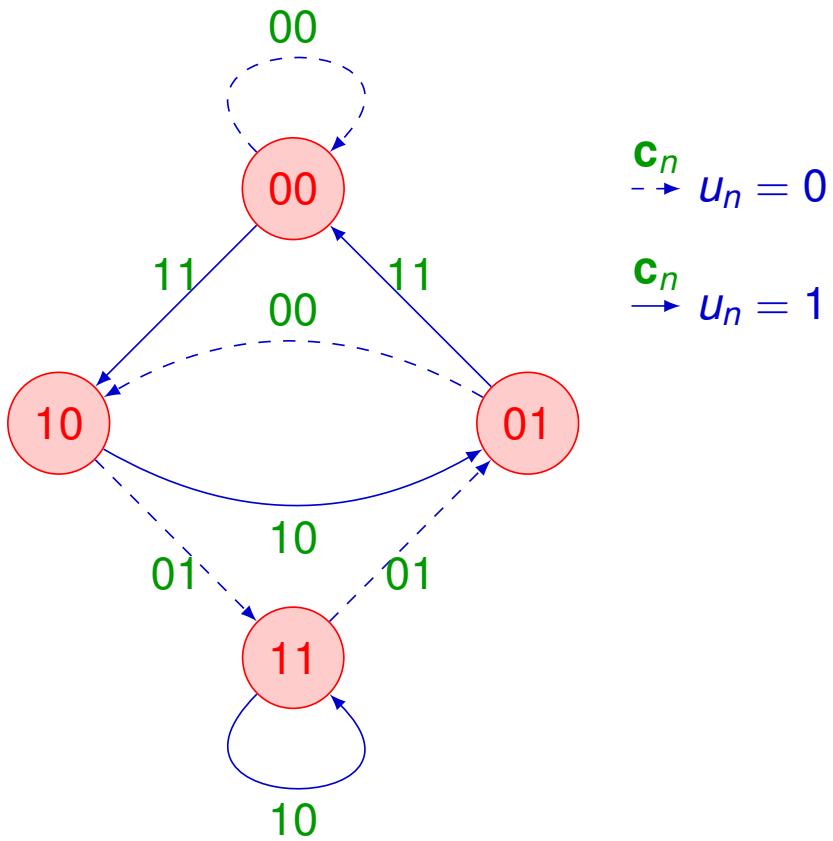
Rappels / définitions



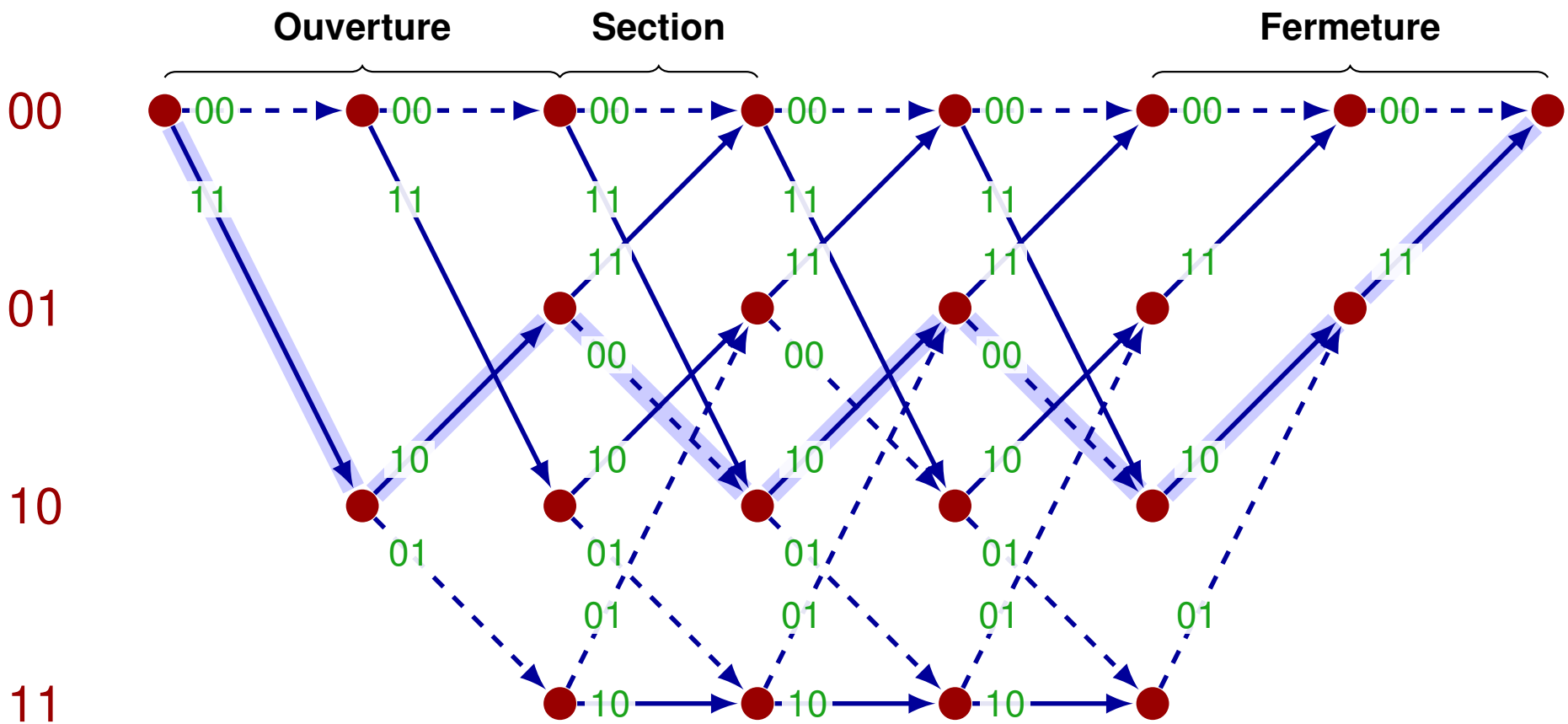
- **Code linéaire**
- **Encodeur récursif / non récursif**
- **Encodeur systématique / non systématique**
- **Notation octale**

Rappels / définitions



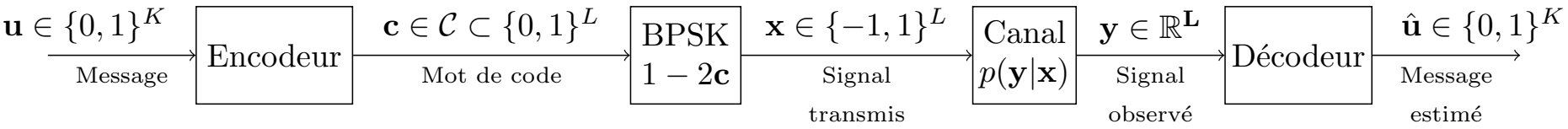


- Message ⇔ chemin dans le graphe
- Mot de code ⇔ étiquettes le long du chemin dans le graphe
- Nécessité de définir (au moins) un état initial



- 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)** : $\hat{\mathbf{u}} = \underset{\mathbf{u}\in\{0,1\}^K}{\text{argmax}}\ \mathbb{P}(\mathbf{U}=\mathbf{u}|\mathbf{y})$

  $\rightarrow$ **Minimise la probabilité d'erreur trame**

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

  $\rightarrow$ **CC : Équivalent moins complexe du MAP, algorithme de Viterbi**

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

  $\rightarrow$ **Minimise la probabilité d'erreur binaire**

- Nombre de bits dans le message : K
- Taille message + fermeture : $L = K + m$
- Le message envoyé : $\mathbf{u} = [u_0,\ u_1,\ \ldots,\ u_{K-1}]$
- Le mot de code émis : $\mathbf{c} = [\mathbf{c}_0,\ \mathbf{c}_1,\ \ldots,\ \mathbf{c}_{L-1}]$, où $\mathbf{c}_\ell = [c_\ell^{(0)}, c_\ell^{(1)}]$
- Le signal observé : $\mathbf{y} = [\mathbf{y}_0,\ \mathbf{y}_1,\ \ldots,\ \mathbf{y}_{L-1}]$, 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 : $\hat{\mathbf{u}} = [\hat{u}_0,\ \hat{u}_1,\ \ldots,\ \hat{u}_{K-1}] = \phi^{-1}(\hat{\mathbf{c}})$

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

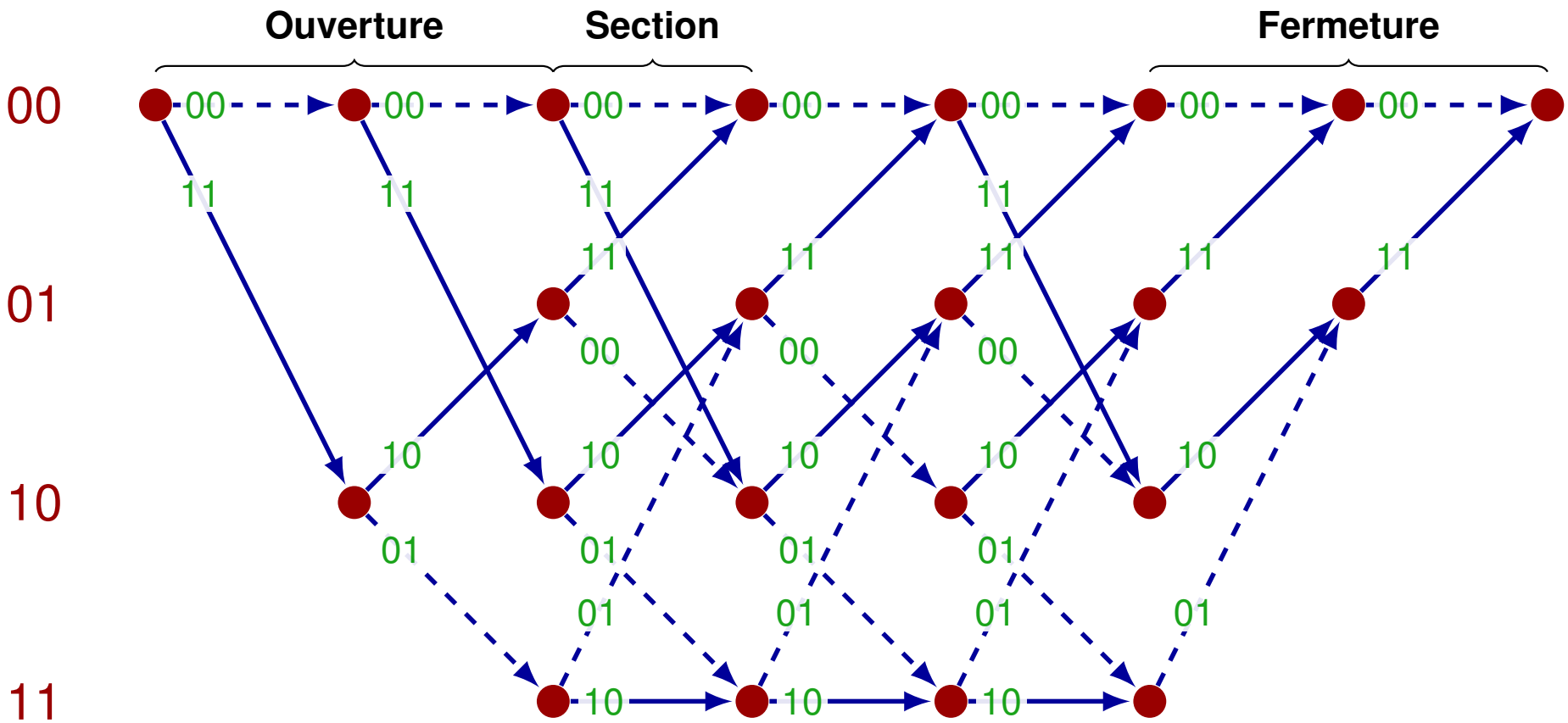
- Nombre de bits dans le message : K
- Taille message + fermeture : $L = K + m$
- Le message envoyé : $\mathbf{u} = [u_0,\ u_1,\ \ldots,\ u_{K-1}]$
- Le mot de code émis : $\mathbf{c} = [\mathbf{c}_0,\ \mathbf{c}_1,\ \ldots,\ \mathbf{c}_{L-1}]$, où $\mathbf{c}_\ell = [c_\ell^{(0)}, c_\ell^{(1)}]$
- Le signal observé : $\mathbf{y} = [\mathbf{y}_0,\ \mathbf{y}_1,\ \ldots,\ \mathbf{y}_{L-1}]$, 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 : $\hat{\mathbf{u}} = [\hat{u}_0,\ \hat{u}_1,\ \ldots,\ \hat{u}_{K-1}] = \phi^{-1}(\hat{\mathbf{c}})$

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

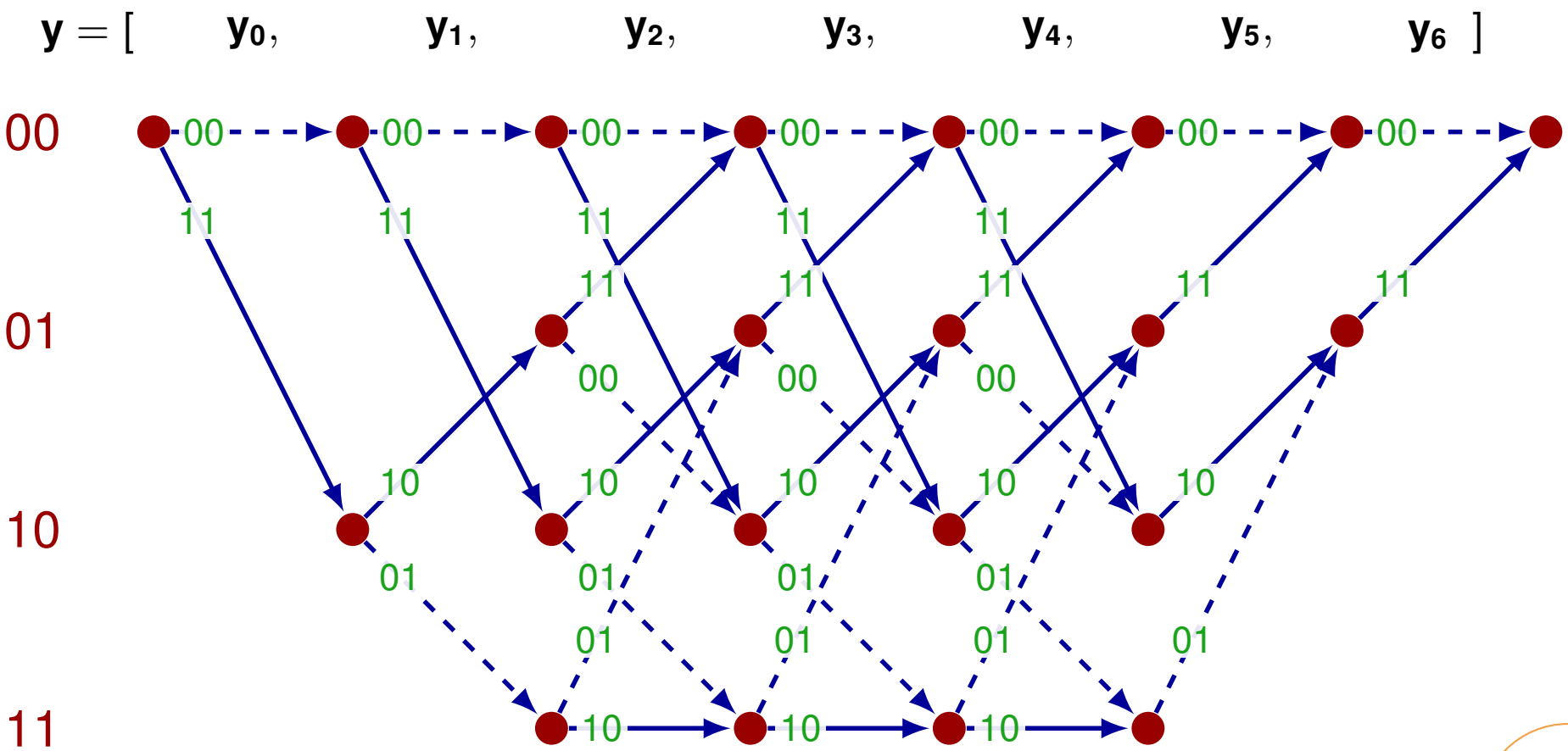
- Nombre de bits dans le message : K
- Taille message + fermeture : $L = K + m$
- Le message envoyé : $\mathbf{u} = [u_0,\ u_1,\ \ldots,\ u_{K-1}]$
- Le mot de code émis : $\mathbf{c} = [\mathbf{c}_0,\ \mathbf{c}_1,\ \ldots,\ \mathbf{c}_{L-1}]$, où $\mathbf{c}_\ell = [c_\ell^{(0)}, c_\ell^{(1)}]$
- Le signal observé : $\mathbf{y} = [\mathbf{y}_0,\ \mathbf{y}_1,\ \ldots,\ \mathbf{y}_{L-1}]$, 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 : $\hat{\mathbf{u}} = [\hat{u}_0,\ \hat{u}_1,\ \ldots,\ \hat{u}_{K-1}] = \phi^{-1}(\hat{\mathbf{c}})$

- **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**

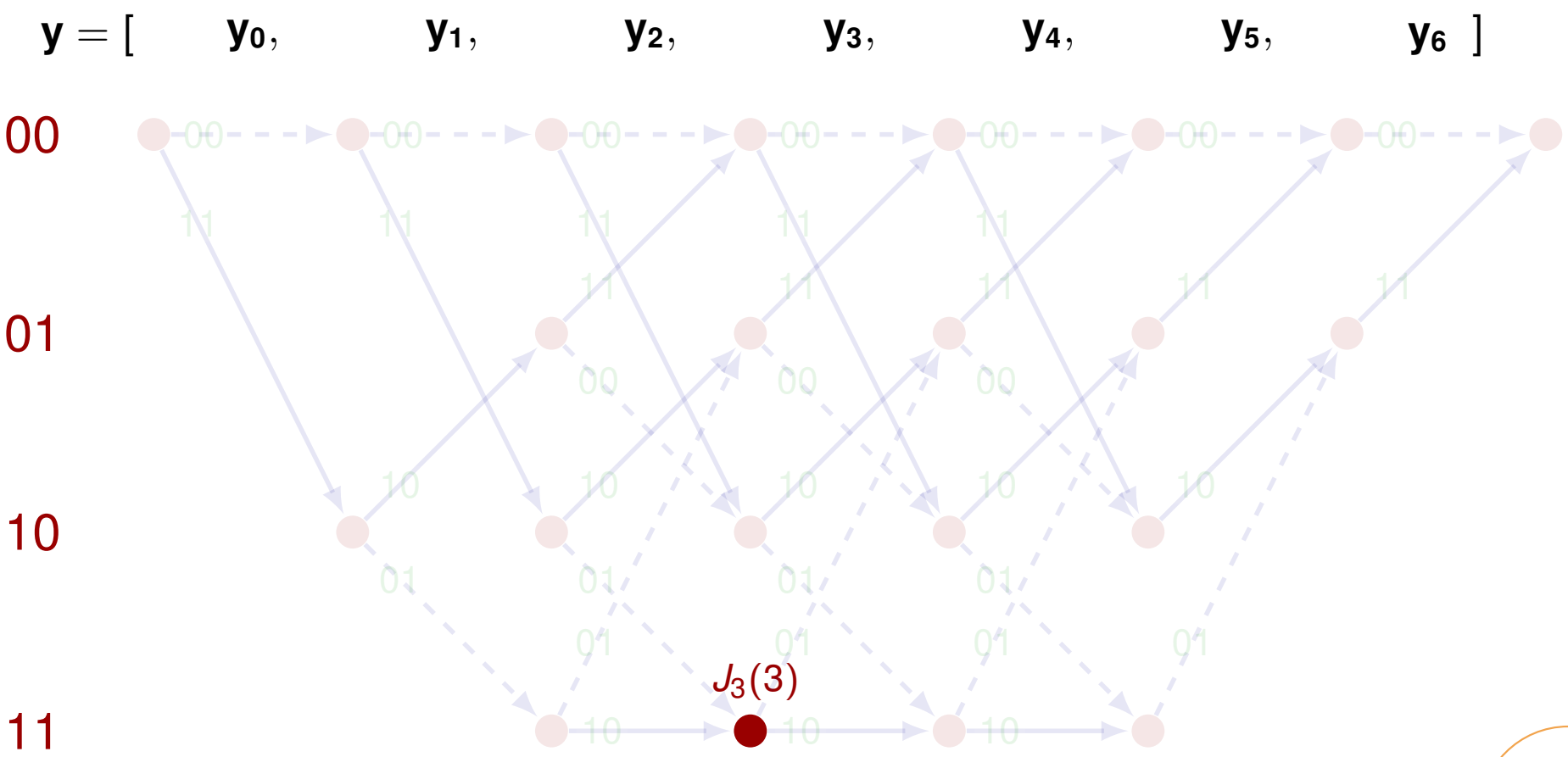
- Soit $J_n(s_n) = \min_{\mathbf{c} \in \mathcal{C}(s_0 \to s_n)} \sum_{\ell=0}^{n-1} \mathbf{c}_\ell \mathbf{y}_\ell^T$
- Le décodage ML est équivalent à trouver l'antécédent de $J_L(s_L)$ où $s_0 = s_L = 0$



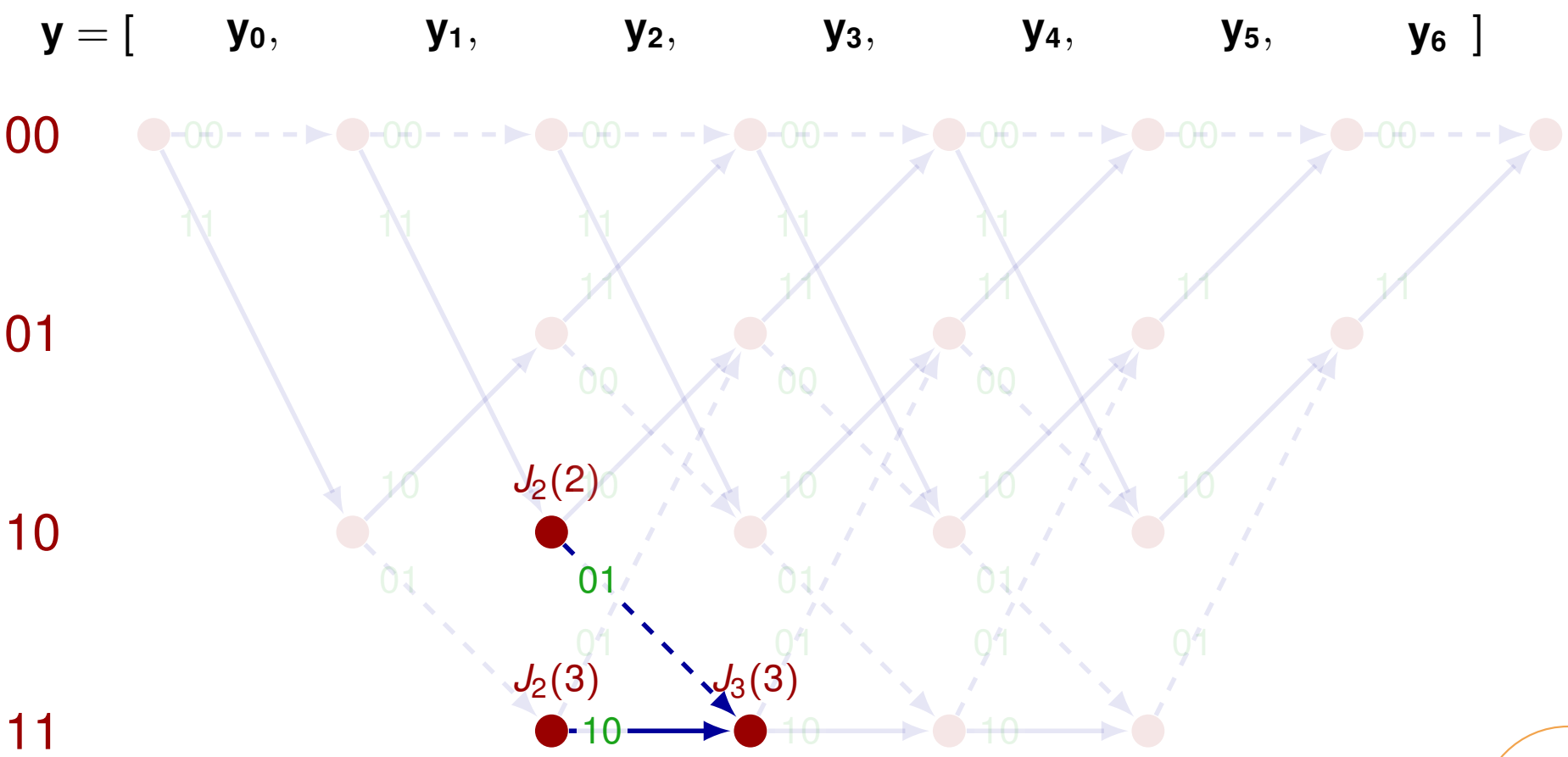
- $\mathcal{P}(s_n)$: **parents de l'état** s_n, i.e. ensemble des s_{n-1} tels que $s_{n-1} \to s_n$ existe
- **Algotithme de Viterbi** : pour chaque $n \in [0, L-1]$ et chaque $s_n \in \mathcal{S}_n$ calculer
$$J_n(s_n) = \min_{s_{n-1} \in \mathcal{P}(s_n)} \left[J_{n-1}(s_{n-1}) + \mathbf{c}_{n-1}(s_{n-1}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

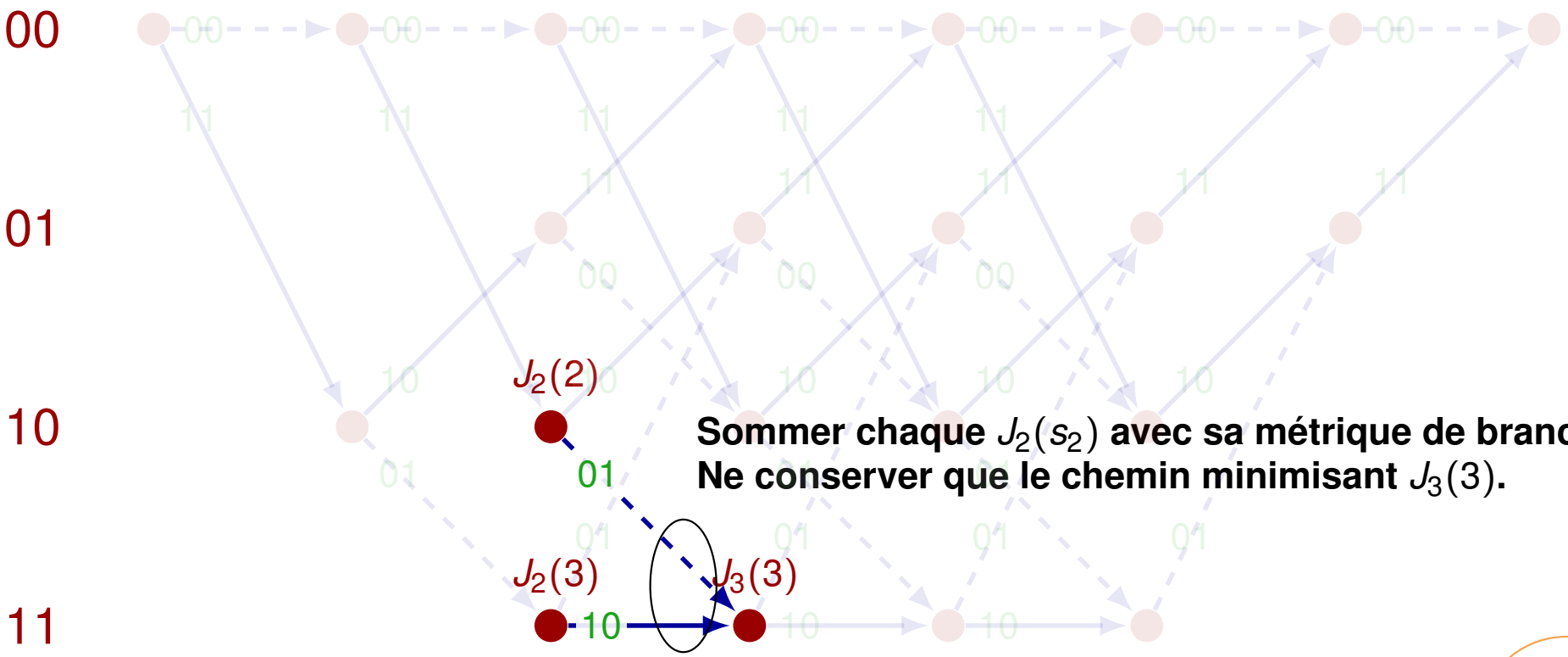


- $\mathcal{P}(s_n)$: **parents de l'état** s_n, i.e. ensemble des s_{n-1} tels que $s_{n-1} \to s_n$ existe
- **Algotithme de Viterbi** : pour chaque $n \in [0, L-1]$ et chaque $s_n \in \mathcal{S}_n$ calculer
$$J_n(s_n) = \min_{s_{n-1} \in \mathcal{P}(s_n)} \left[J_{n-1}(s_{n-1}) + \mathbf{c}_{n-1}(s_{n-1}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

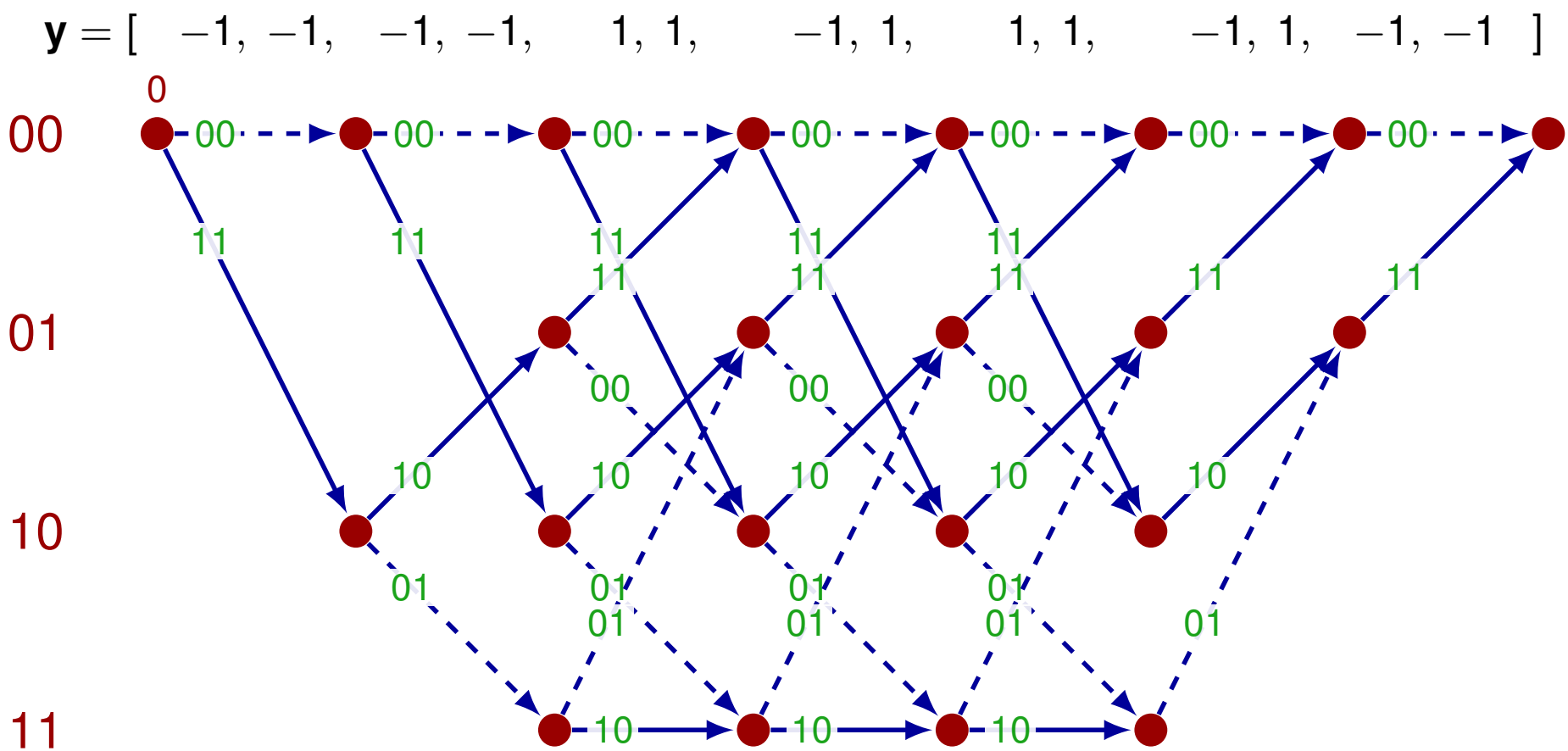
$\mathbf{y} = [\ \mathbf{y_0},\ \mathbf{y_1},\ \mathbf{y_2},\ \mathbf{y_3},\ \mathbf{y_4},\ \mathbf{y_5},\ \mathbf{y_6}\ ]$



Sommer chaque $J_2(s_2)$ avec sa métrique de branche.
Ne conserver que le chemin minimisant $J_3(3)$.

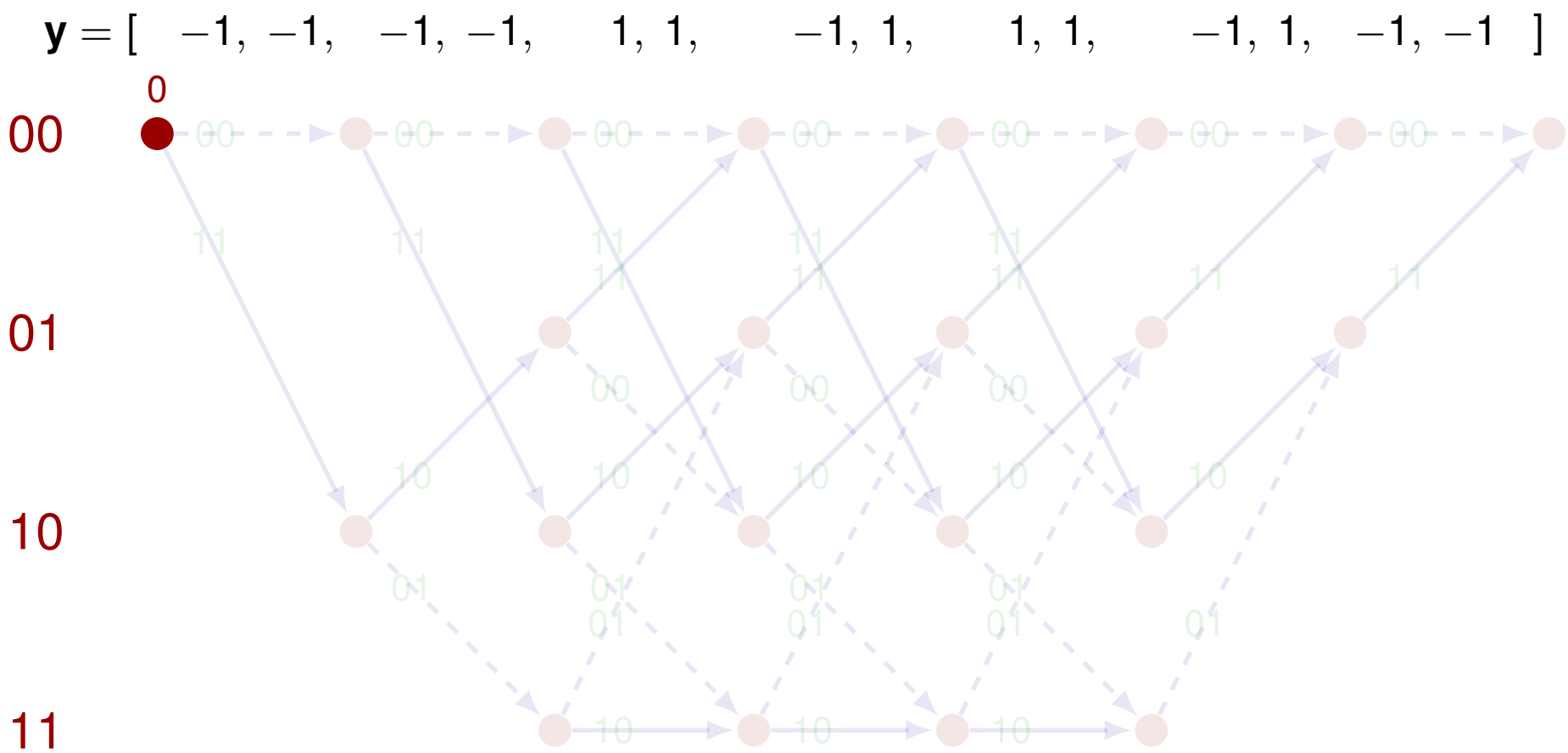
Exemple de question...

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 ?



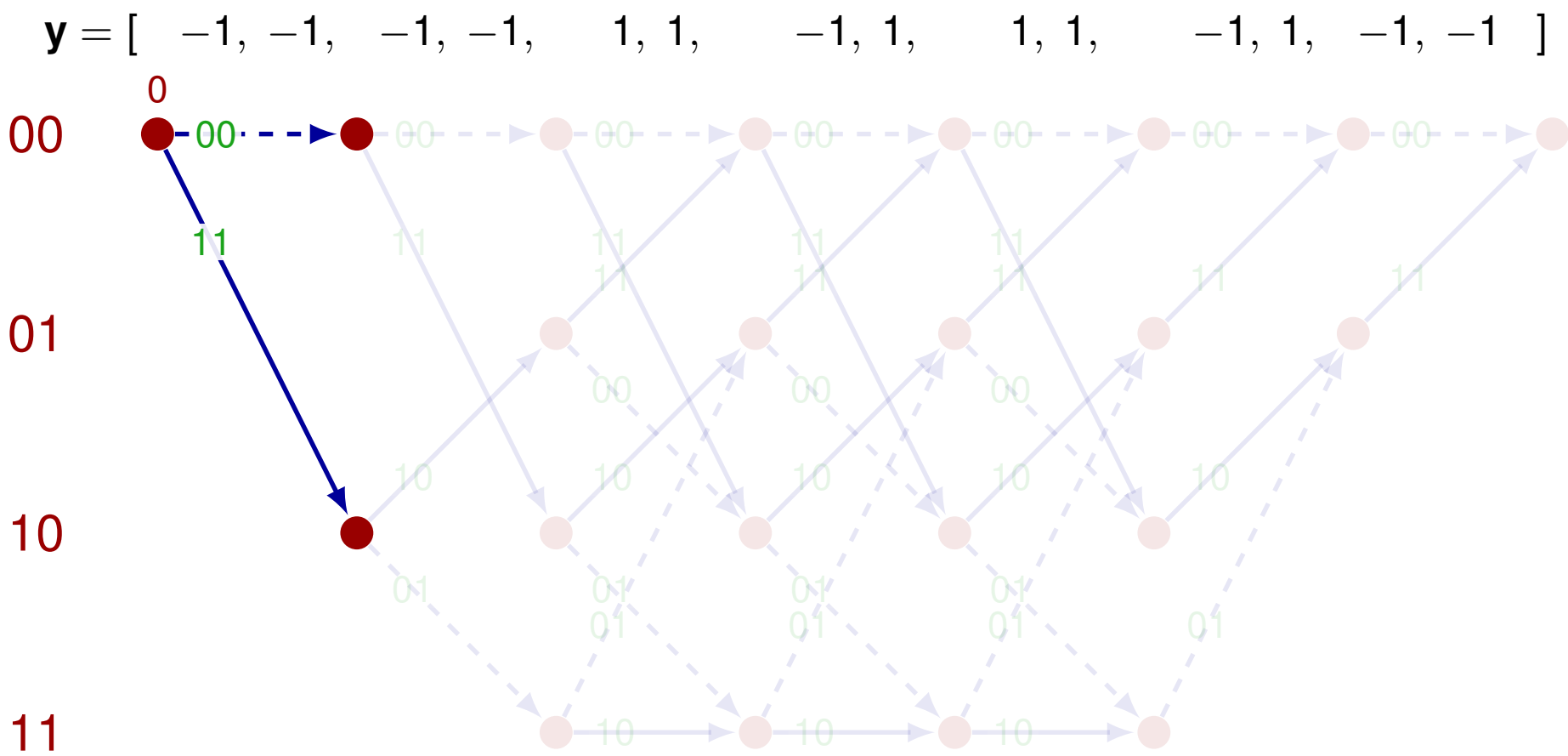
Exemple de question...

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 ?



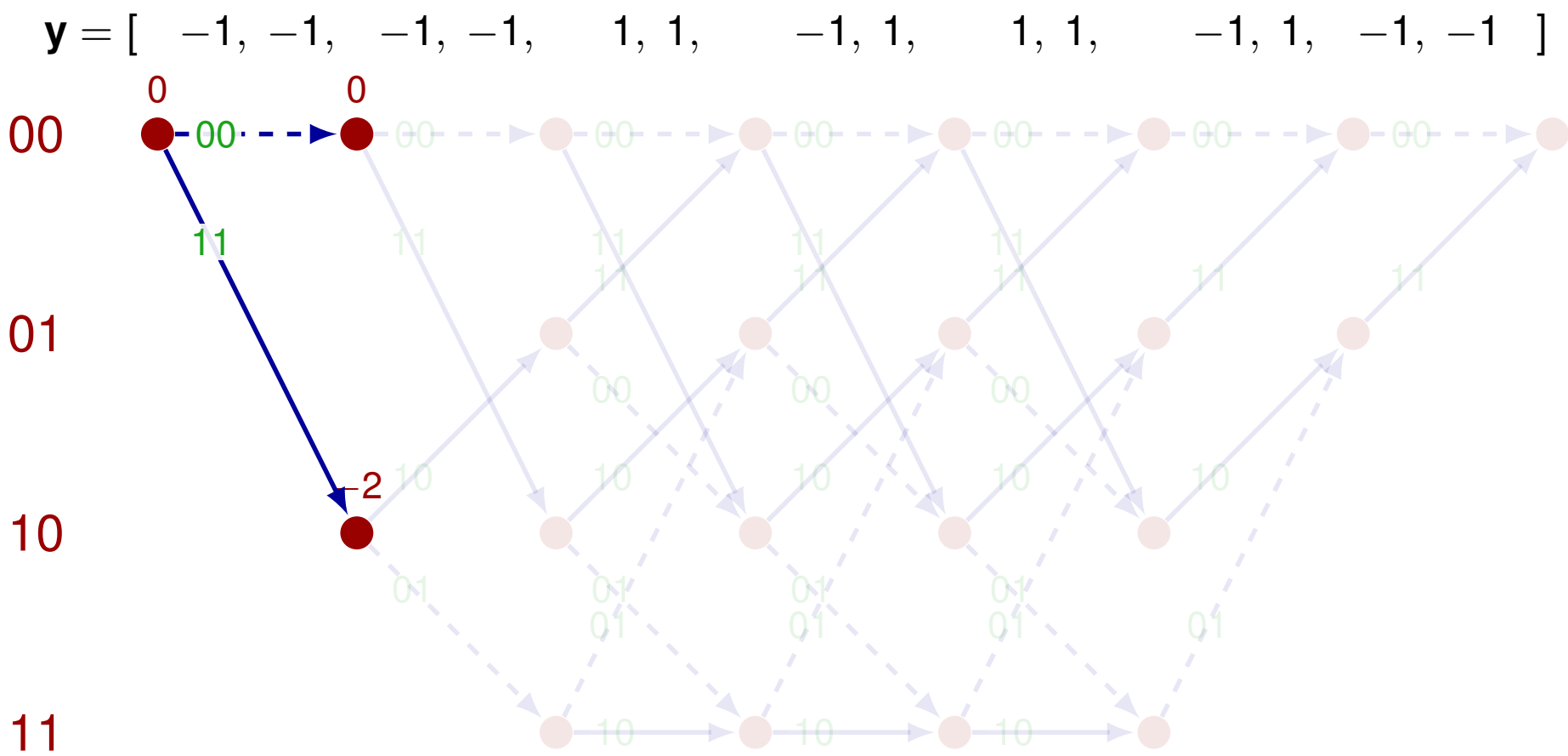
Exemple de question...

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 ?



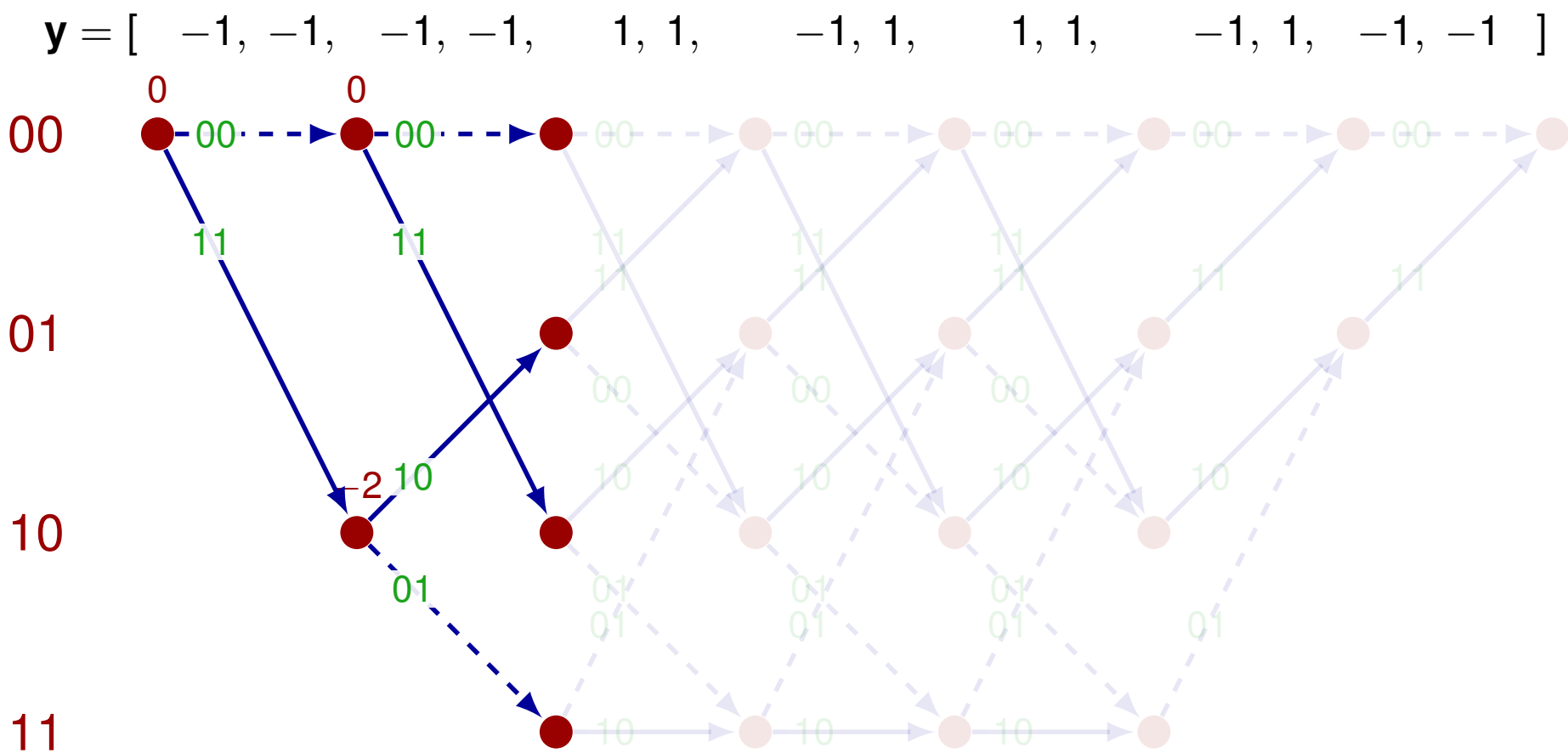
Exemple de question...

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 ?



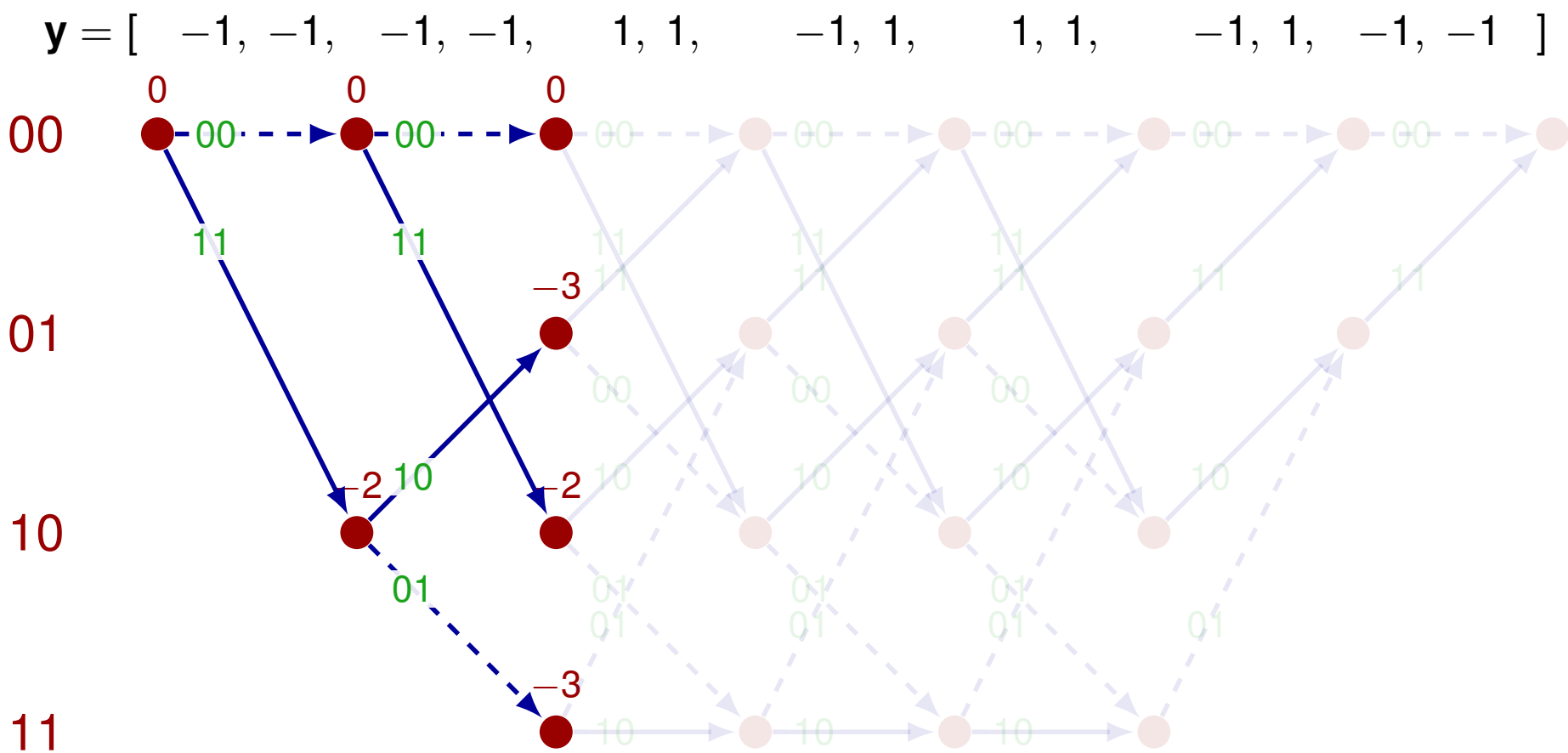
Exemple de question...

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 ?



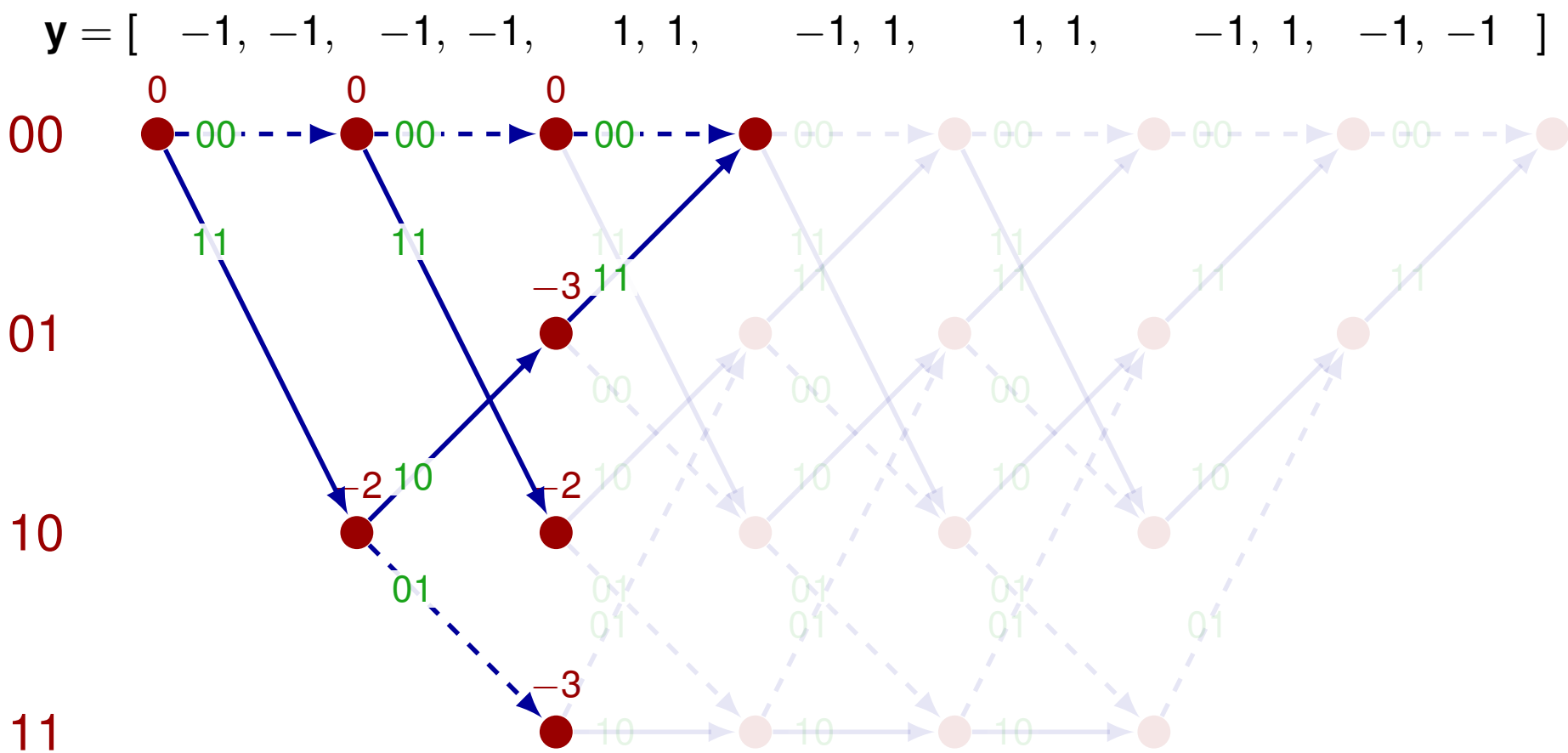
Exemple de question...

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 ?



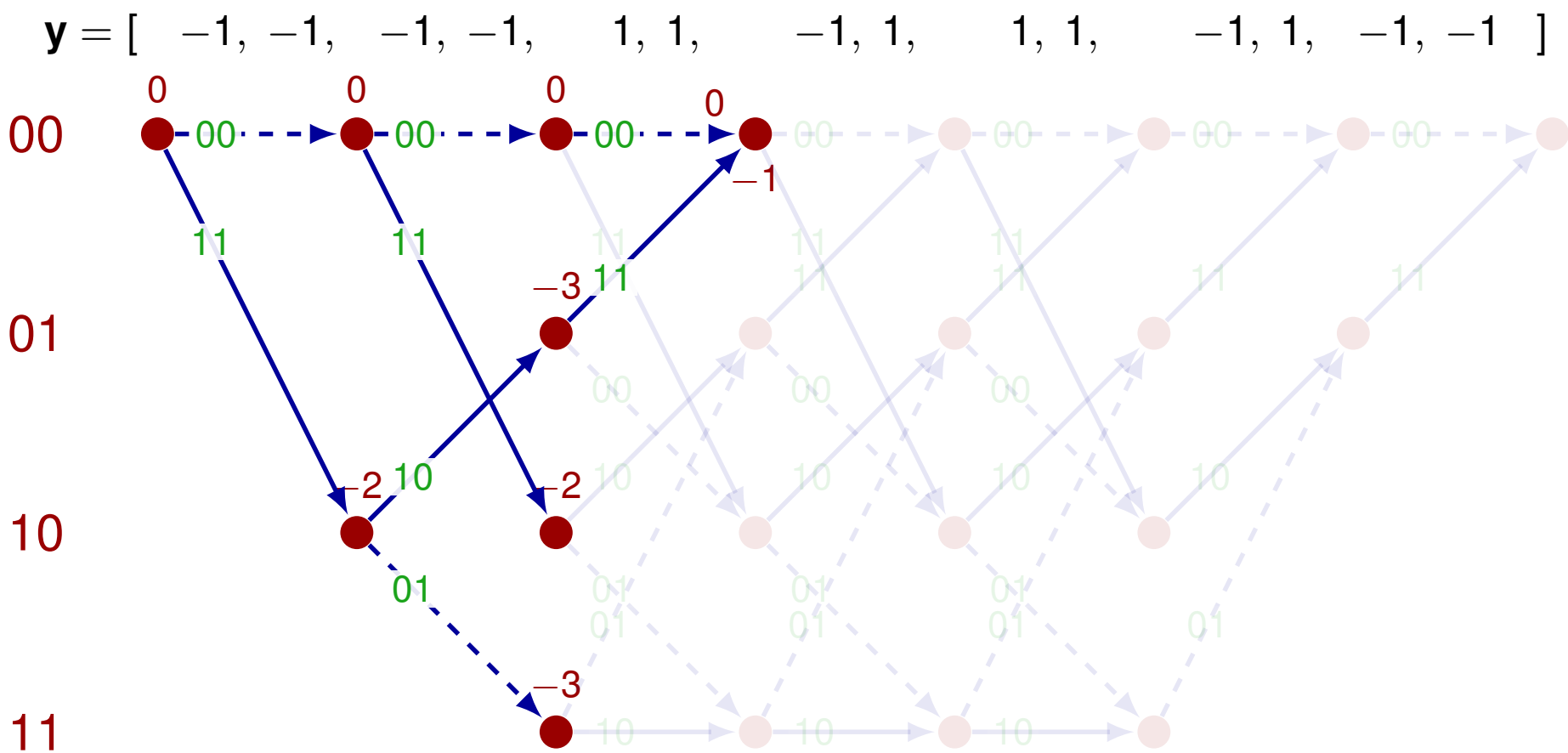
Exemple de question...

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 ?



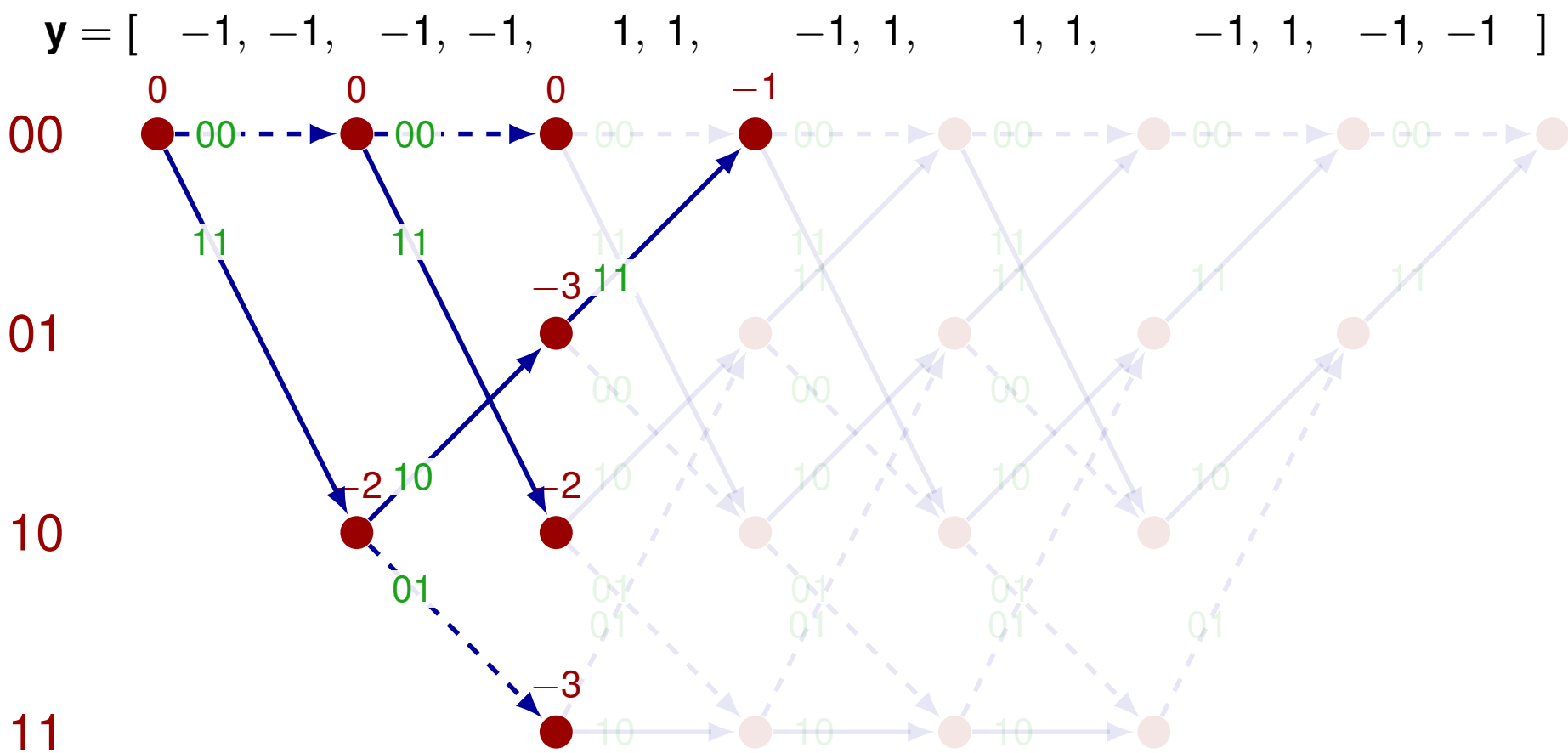
Exemple de question...

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 ?



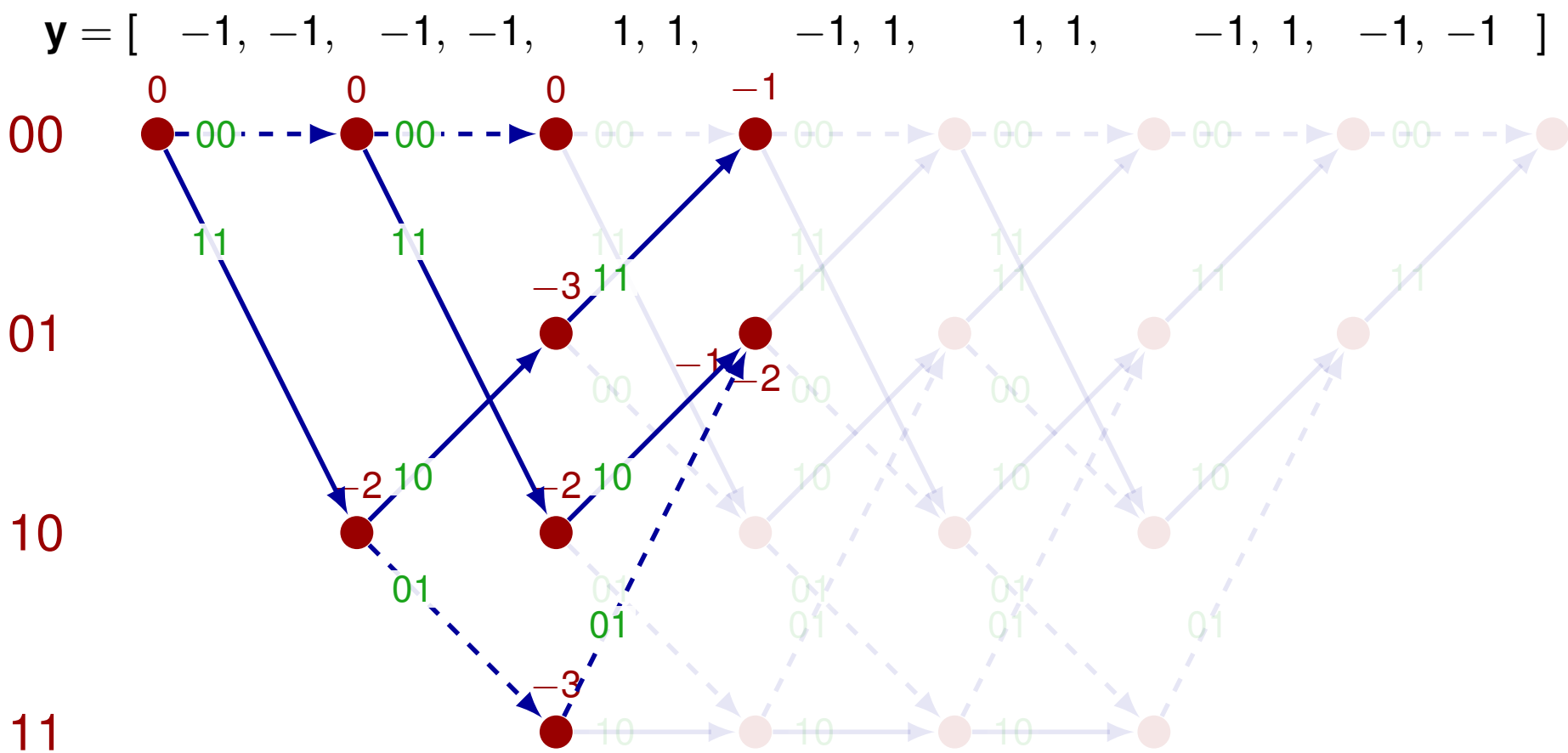
Exemple de question...

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 ?



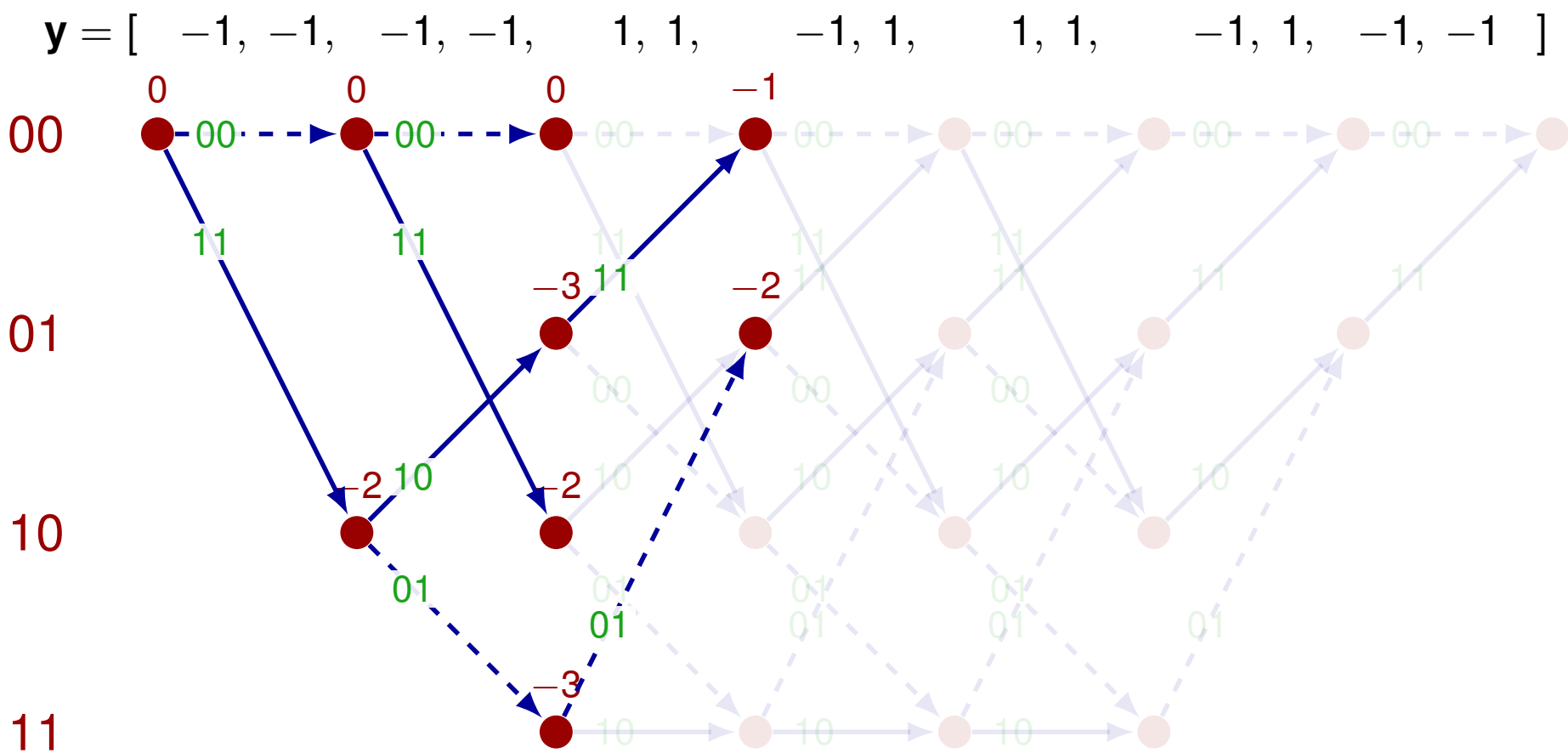
Exemple de question...

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 ?



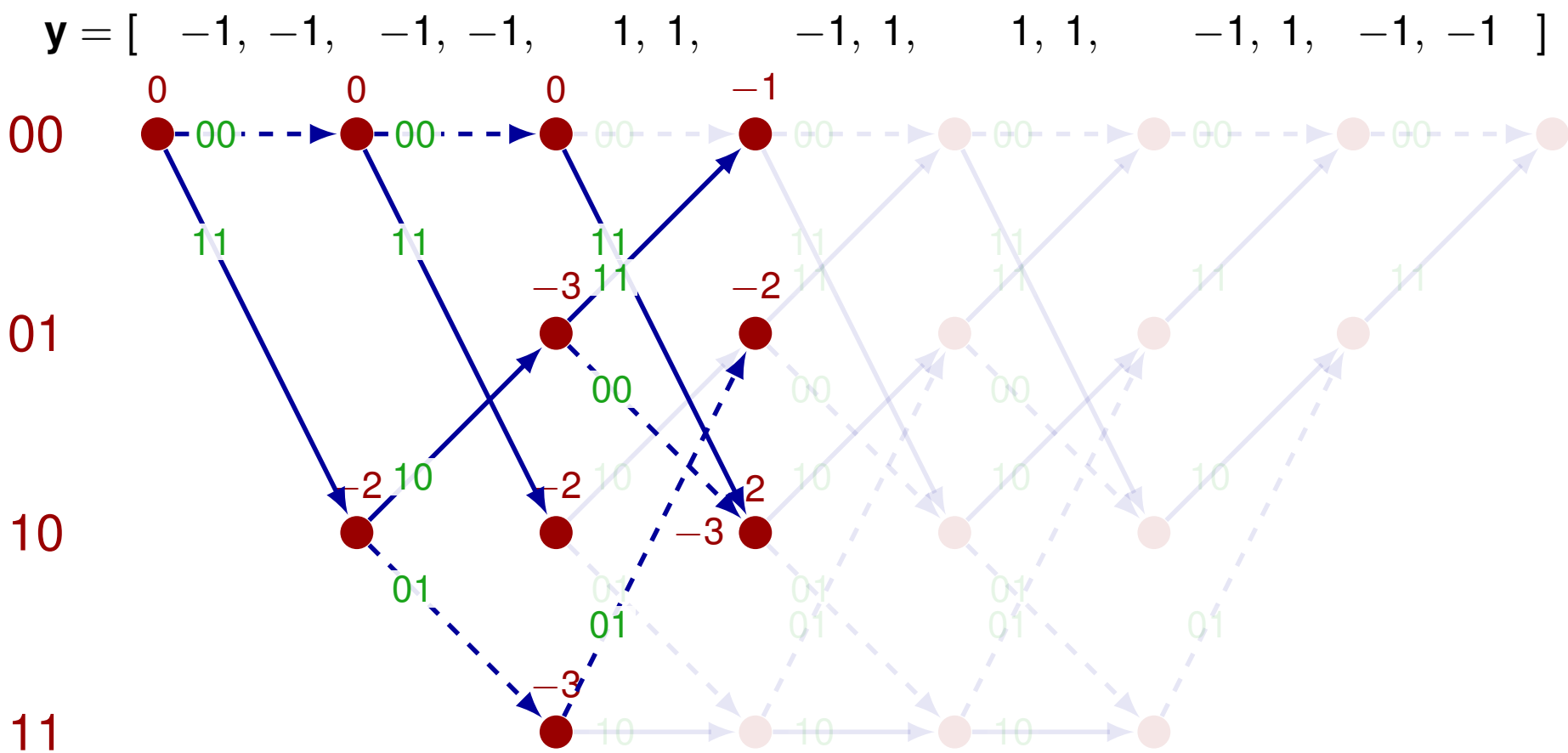
Exemple de question...

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 ?



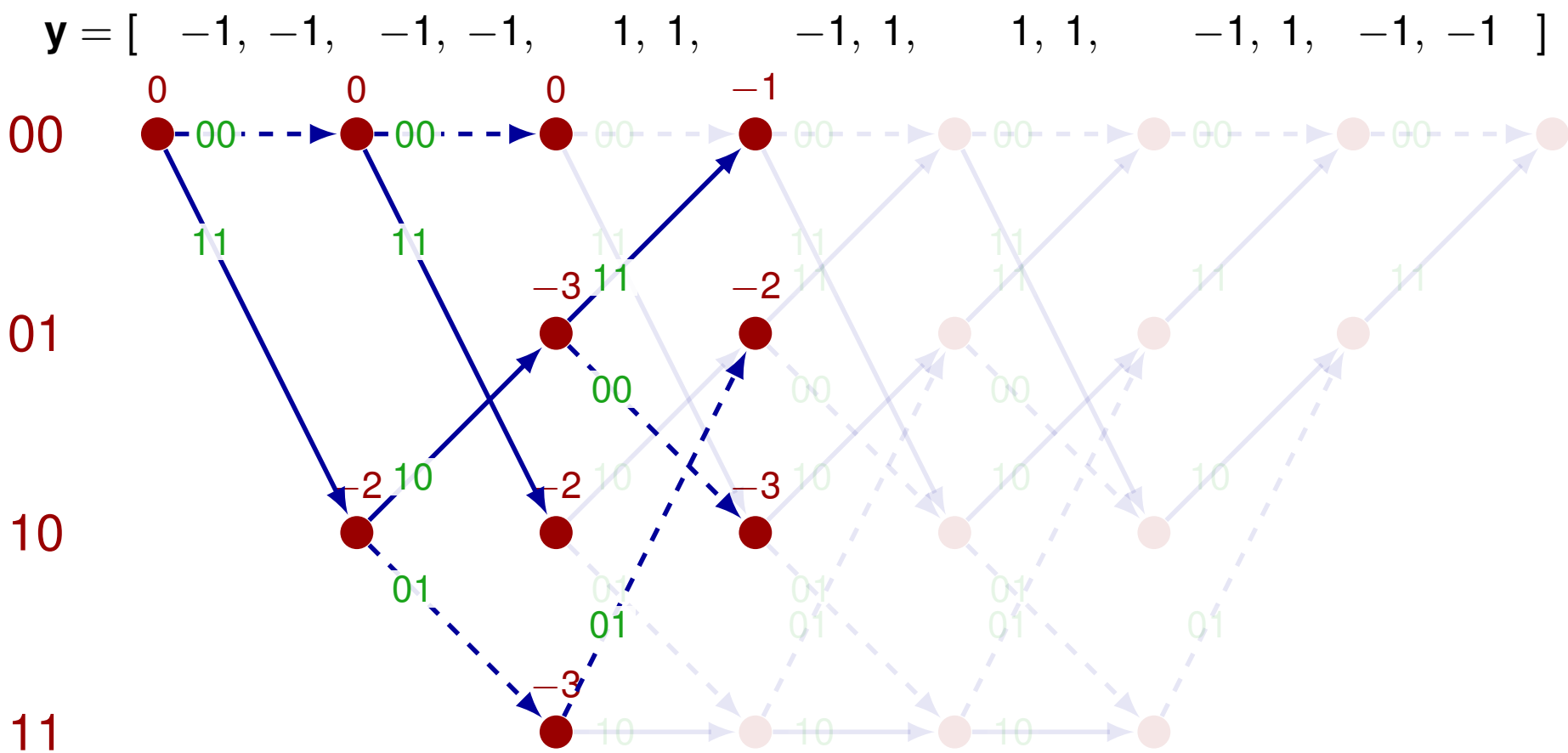
Exemple de question...

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 ?



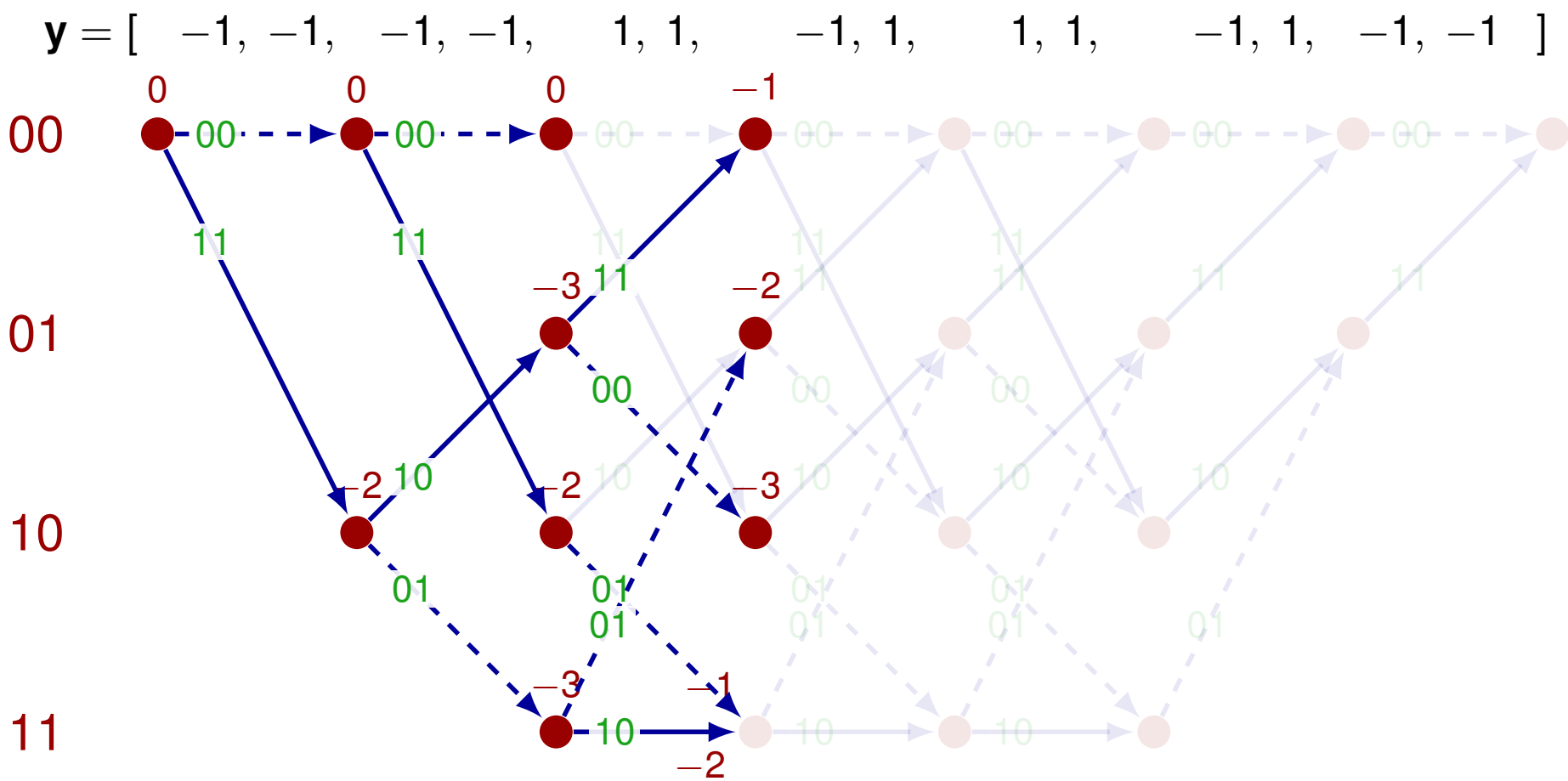
Exemple de question...

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 ?



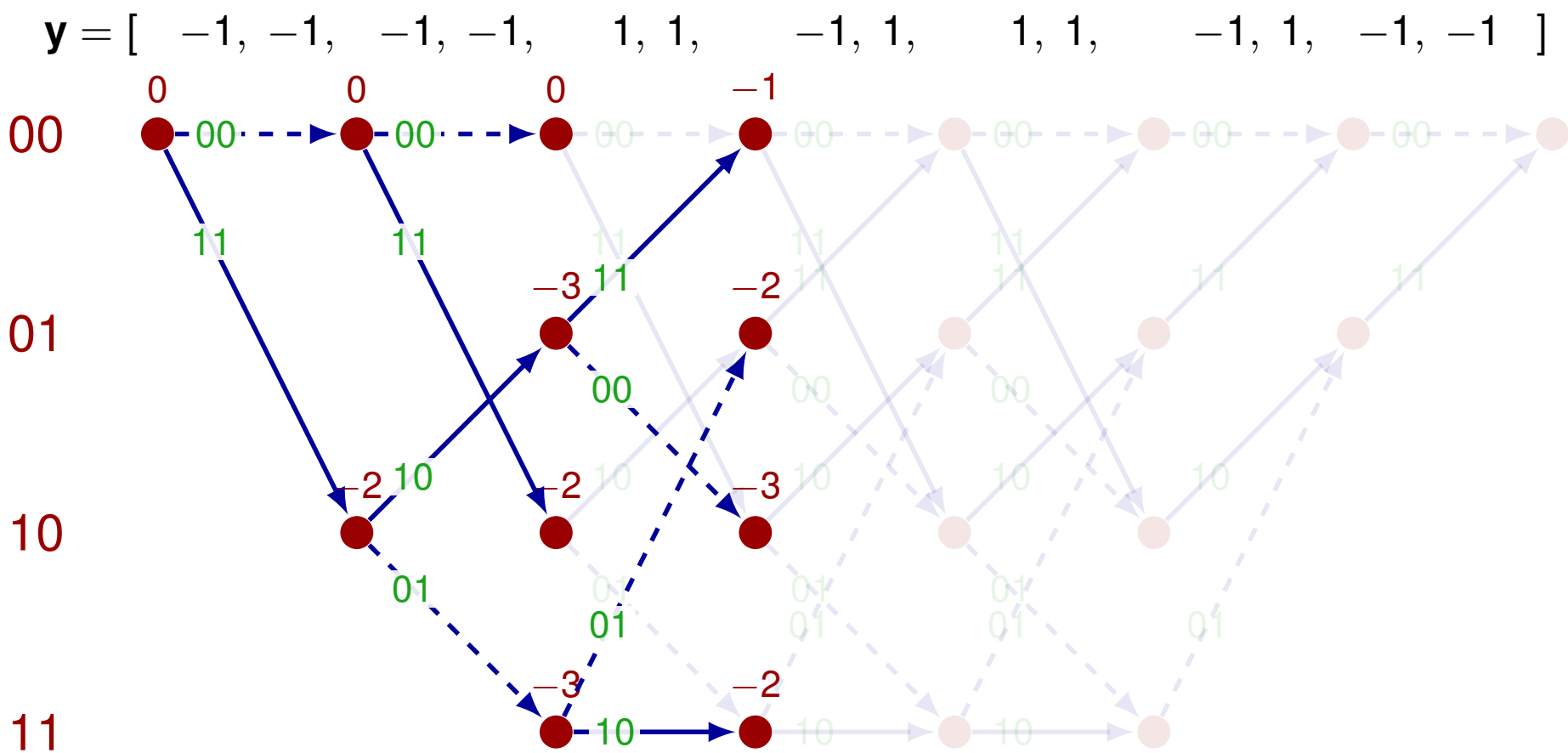
Exemple de question...

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 ?



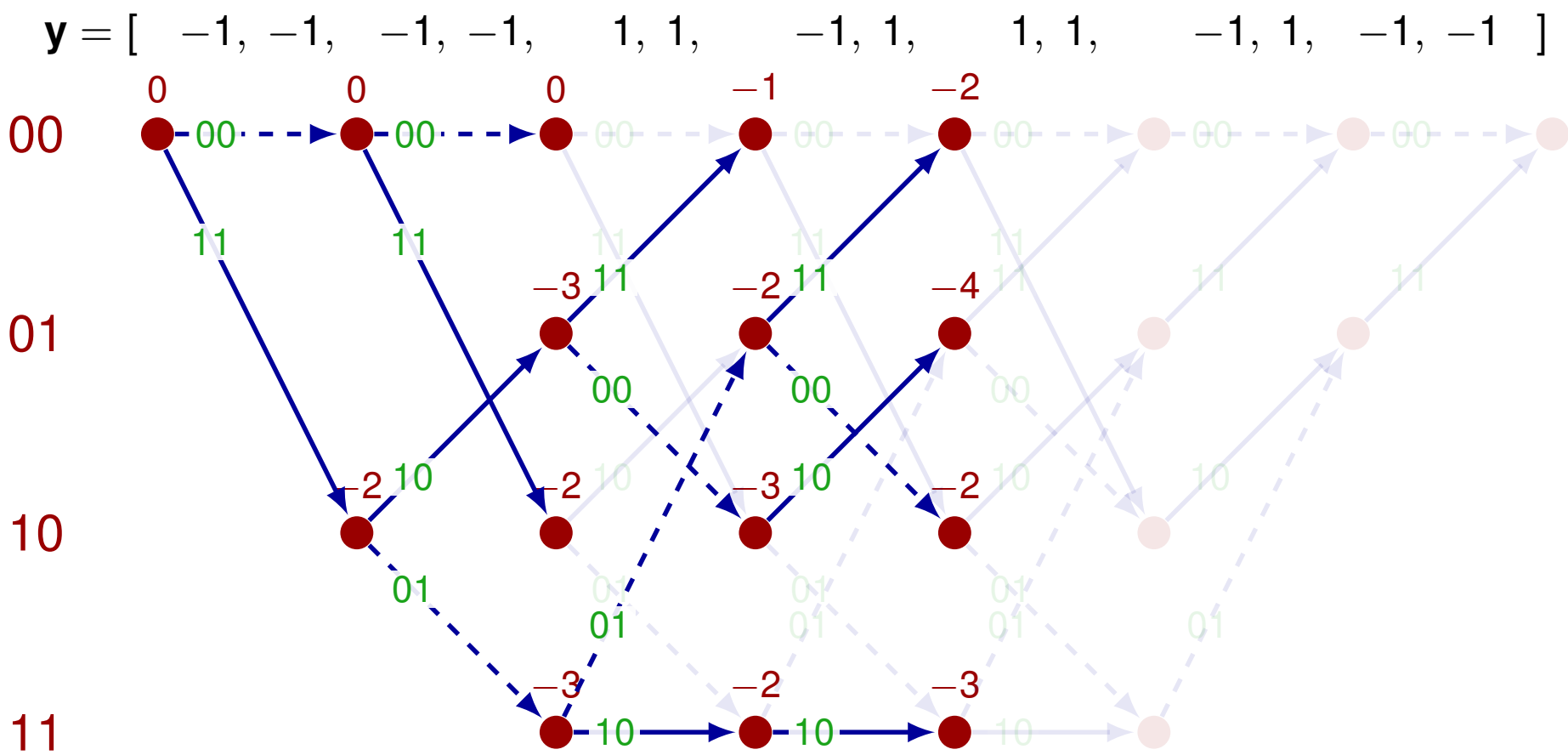
Exemple de question...

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 ?



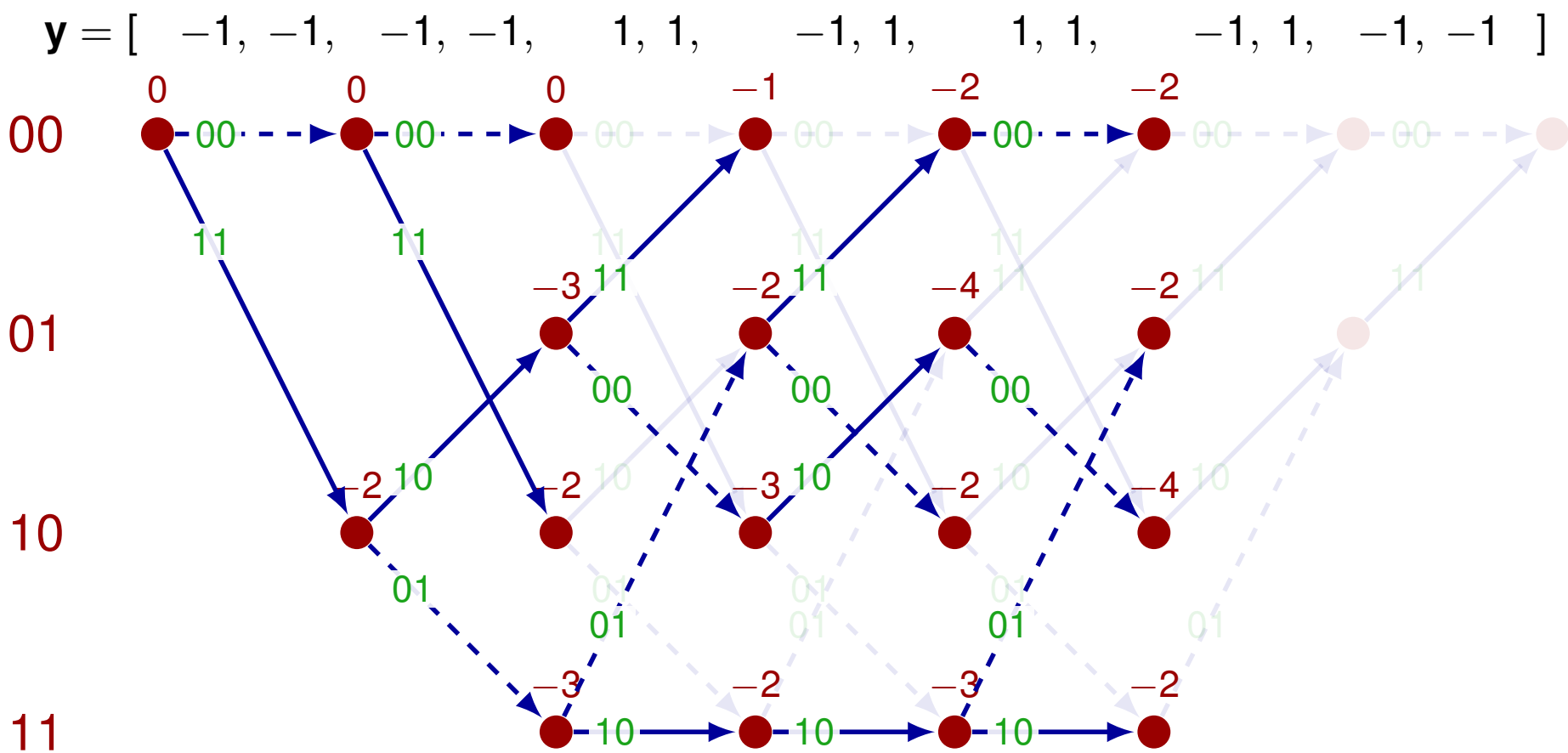
Exemple de question...

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 ?



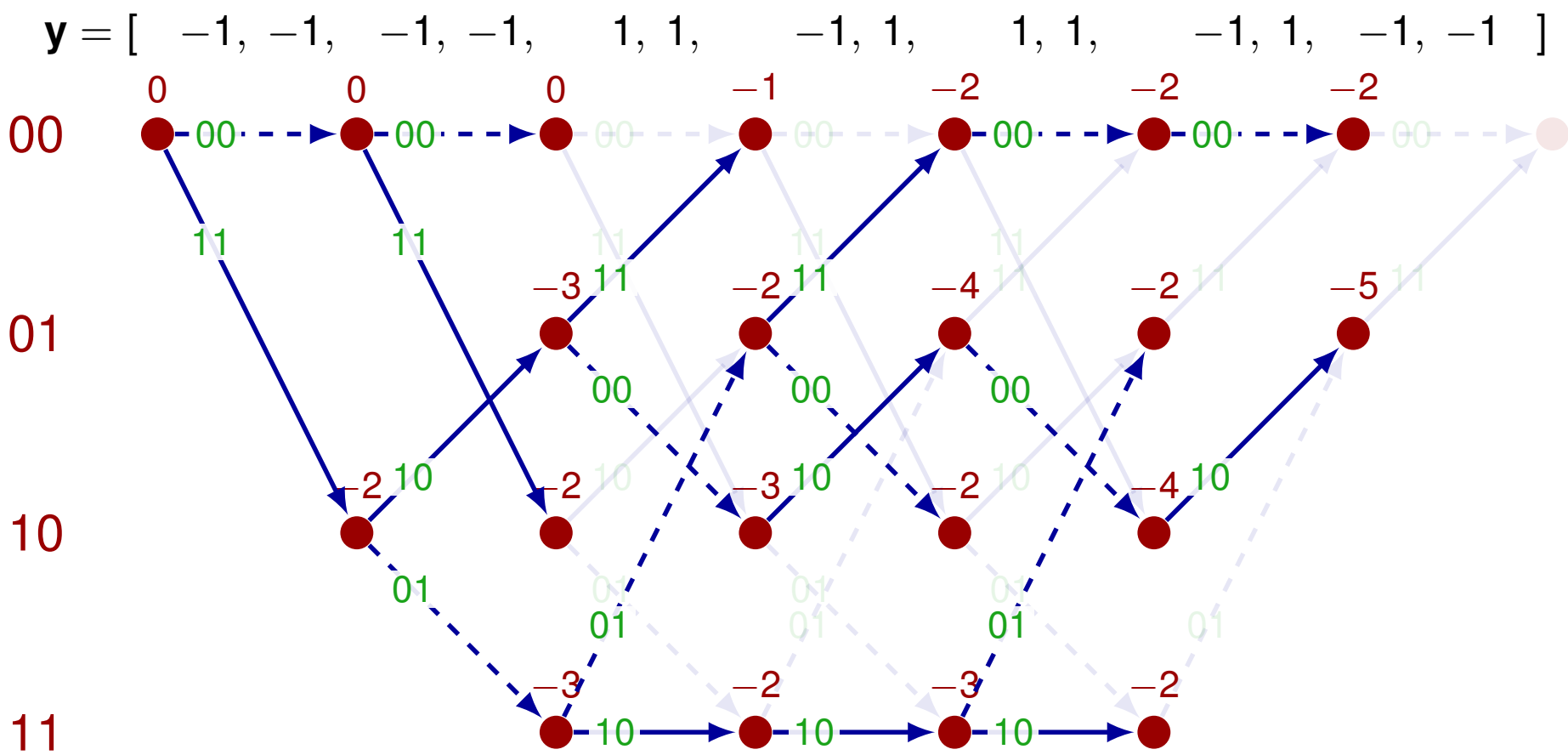
Exemple de question...

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 ?



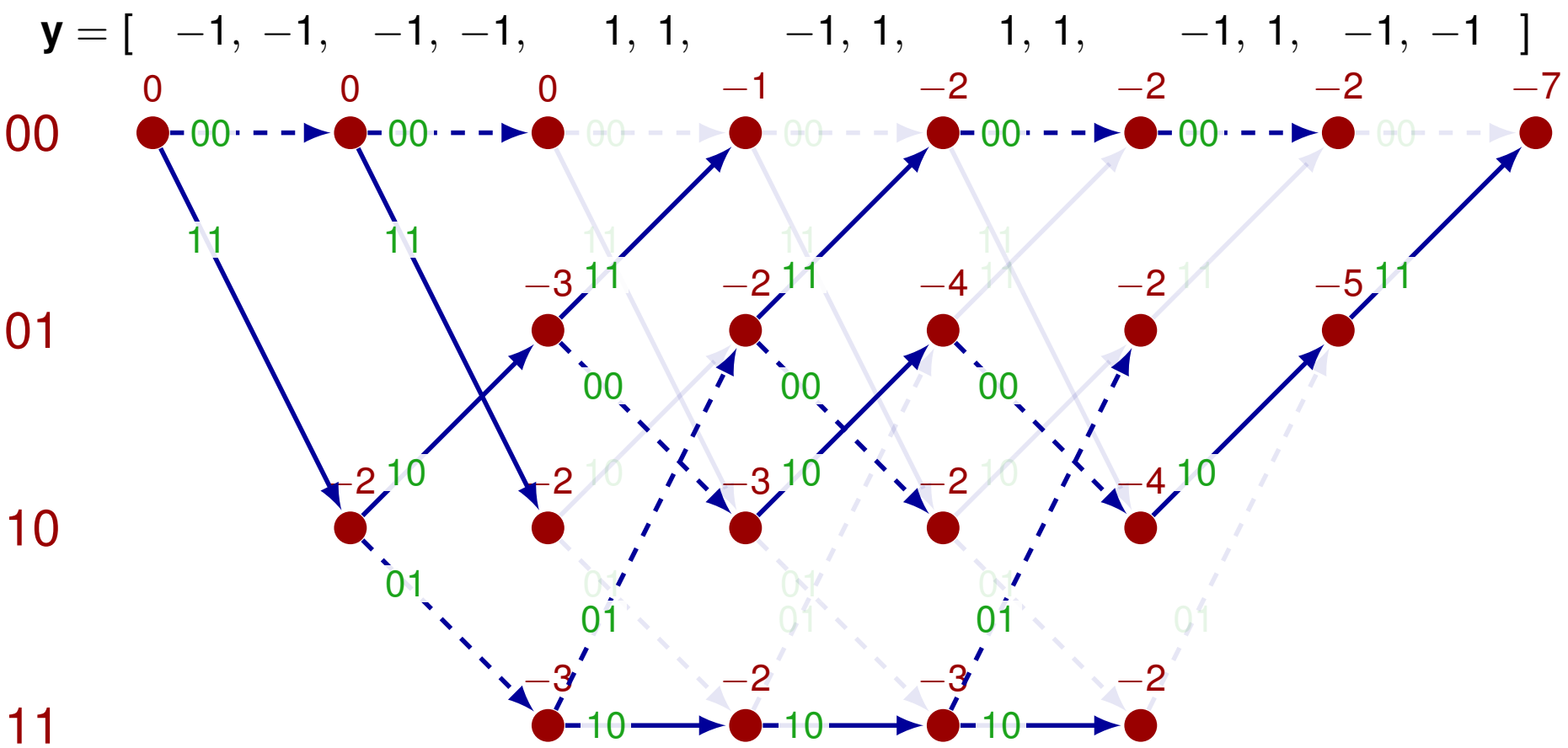
Exemple de question...

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 ?



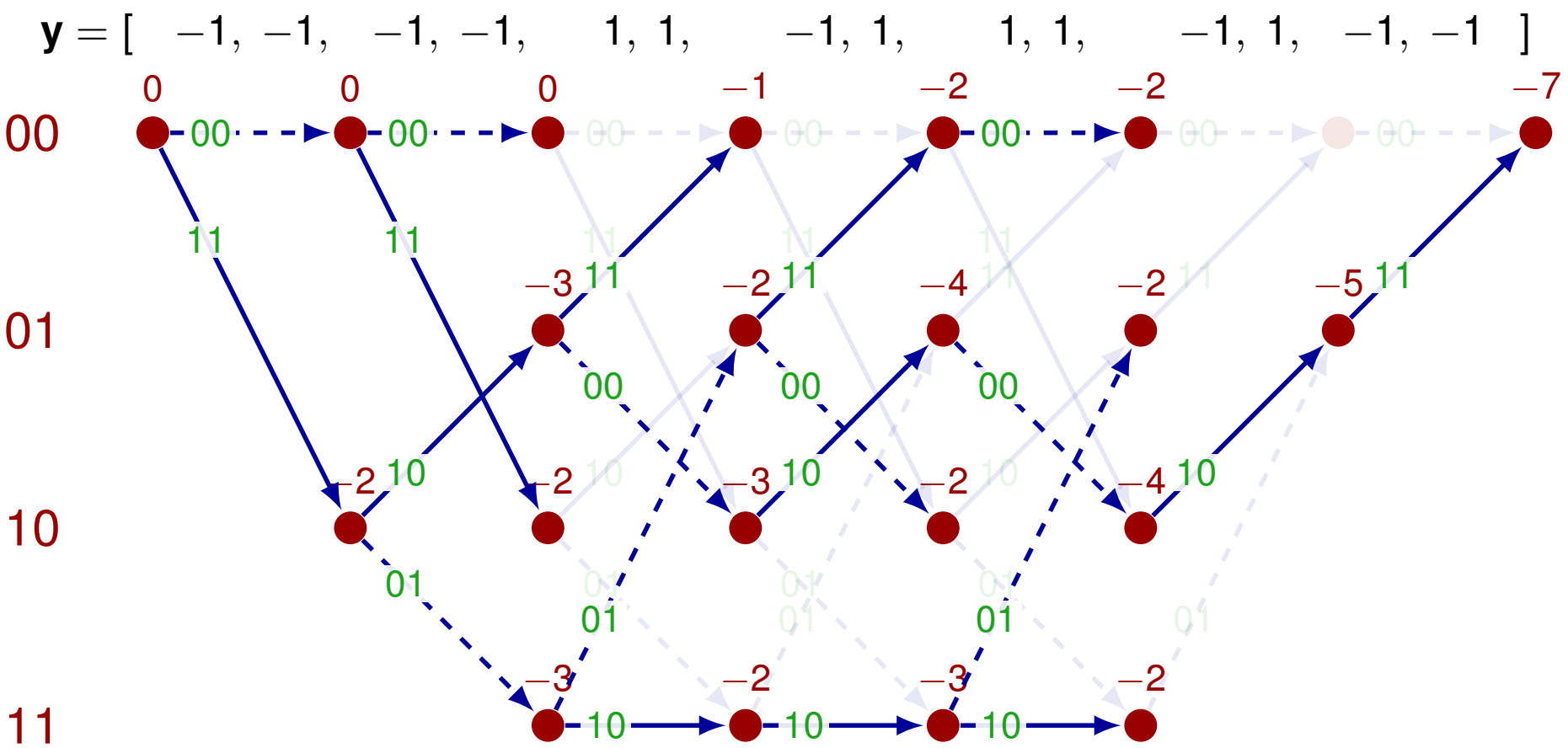
Exemple de question...

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 ?



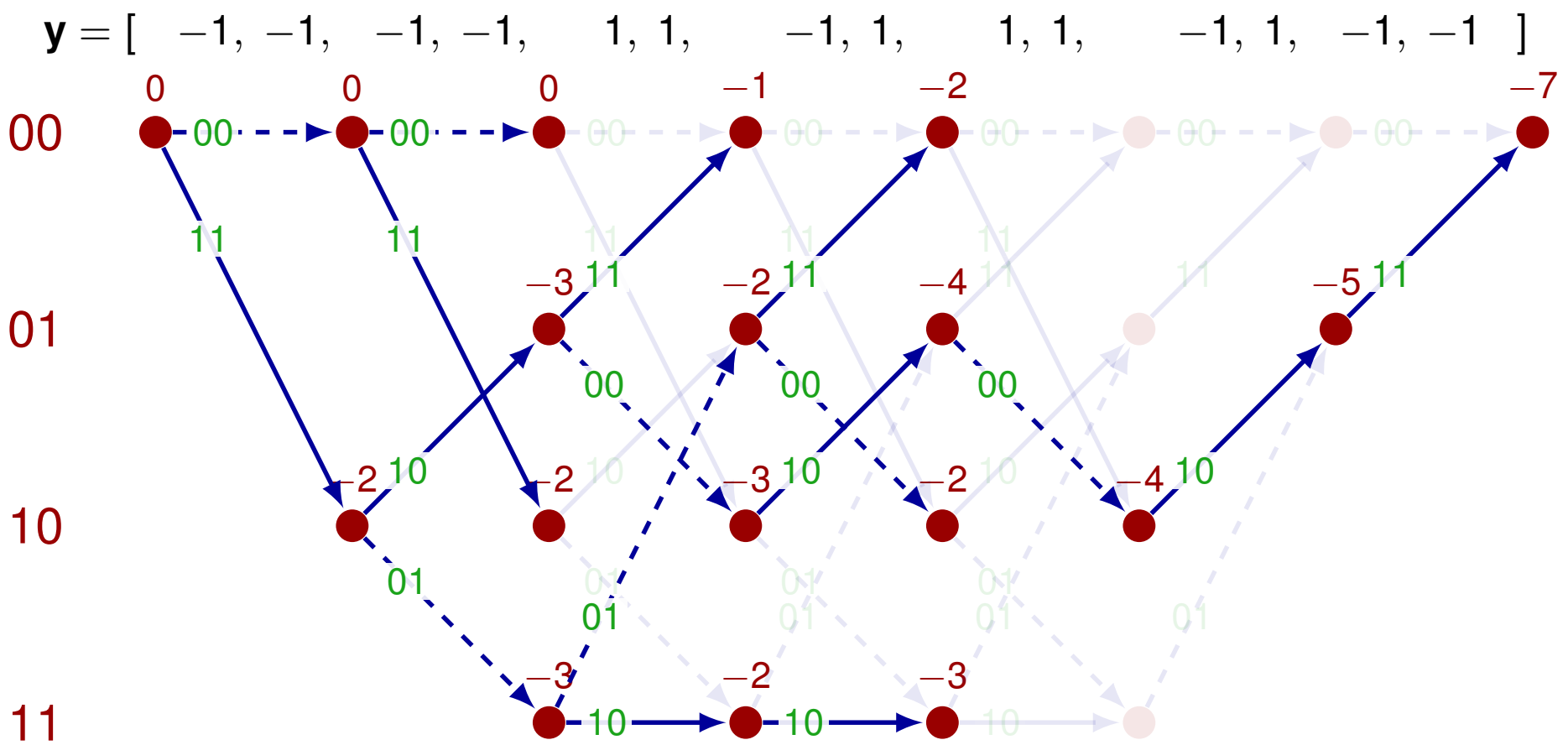
Exemple de question...

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 ?



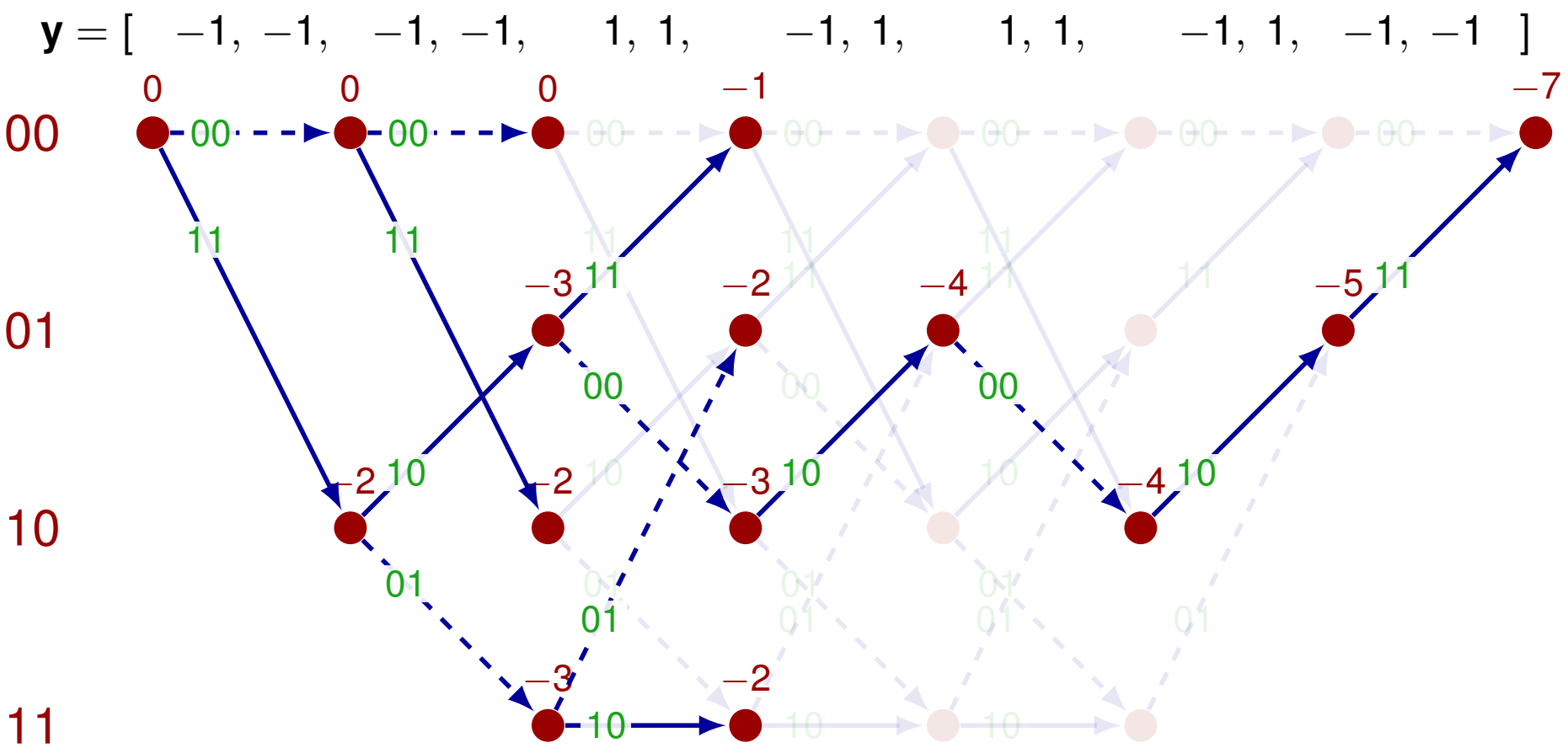
Exemple de question...

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 ?



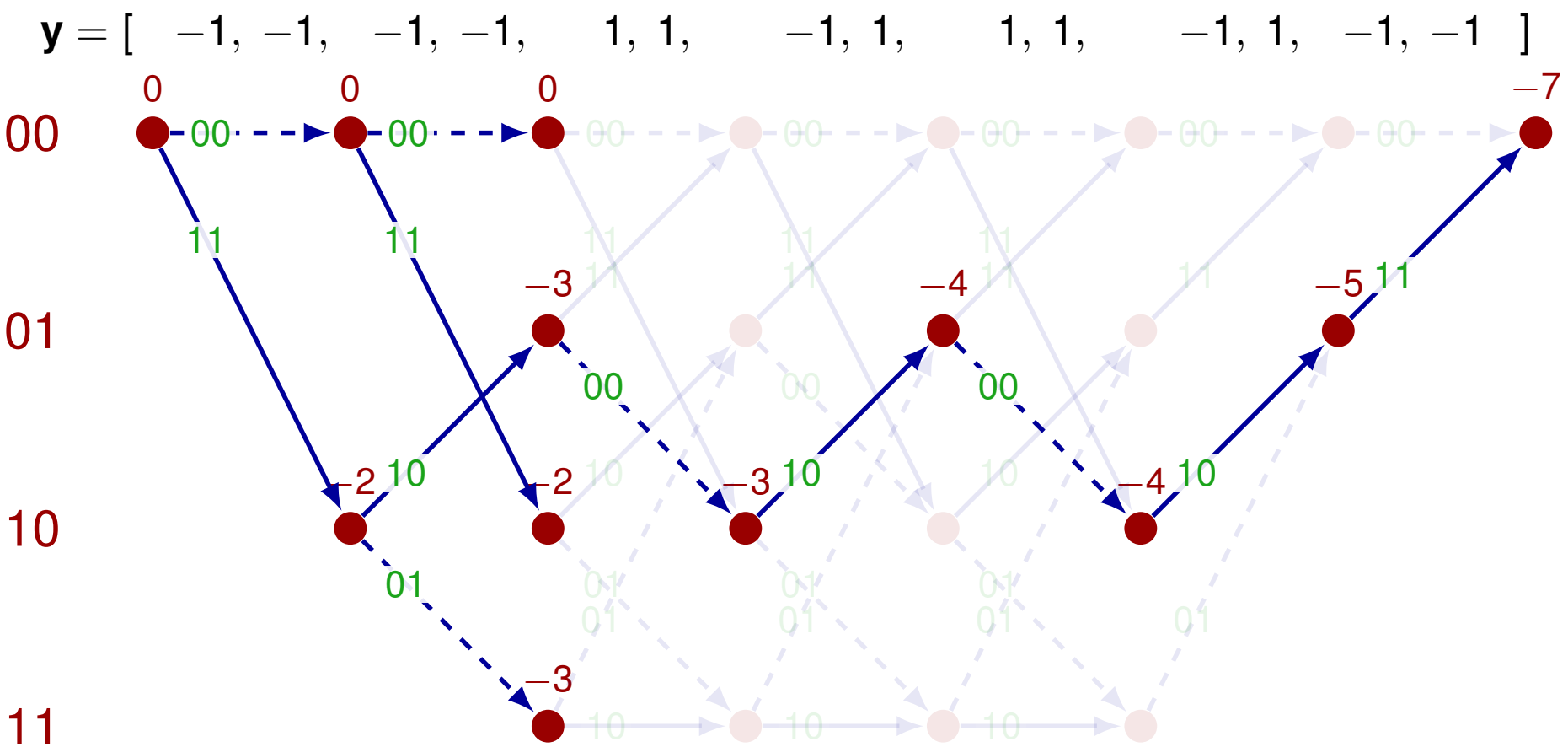
Exemple de question...

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 ?



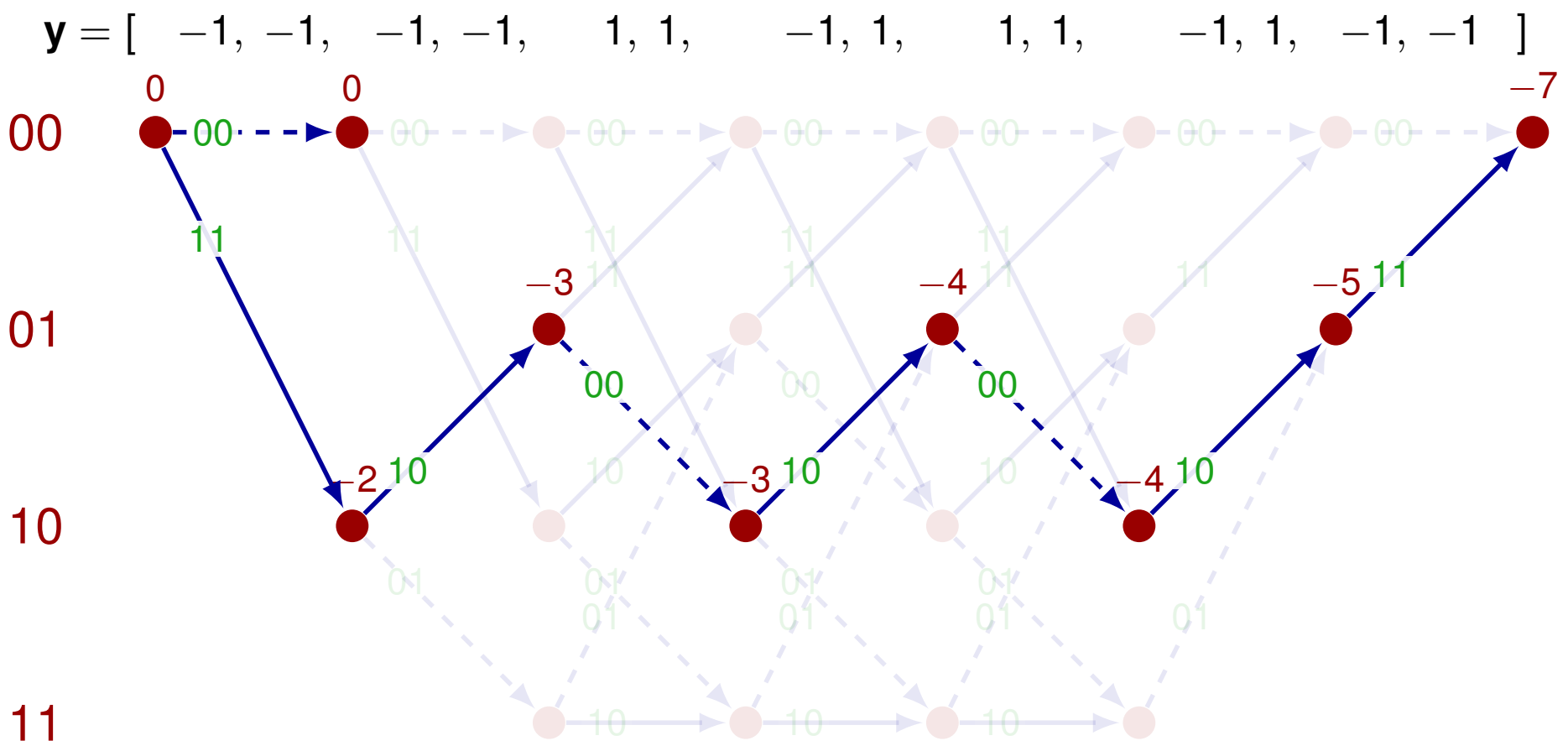
Exemple de question...

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 ?



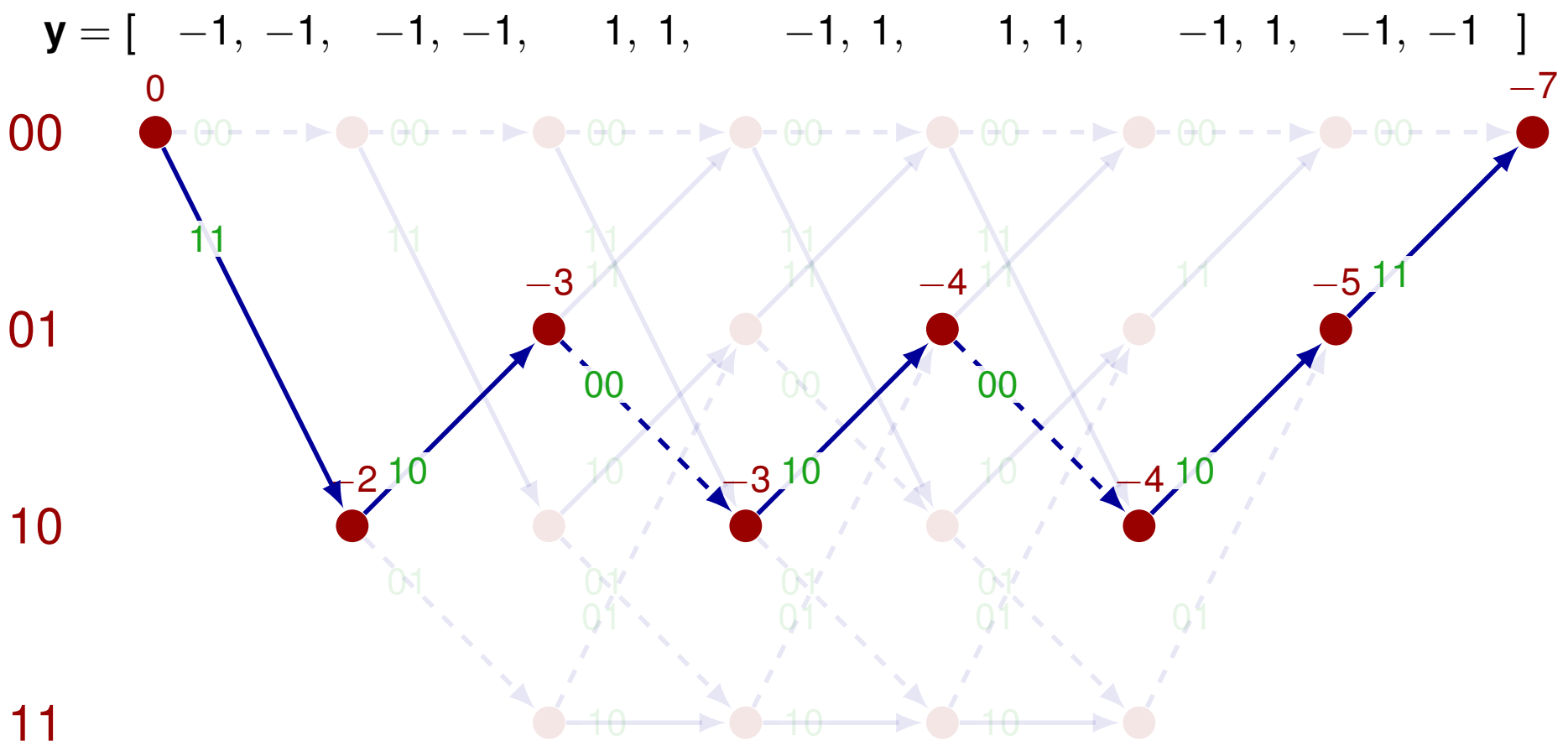
Exemple de question...

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 ?



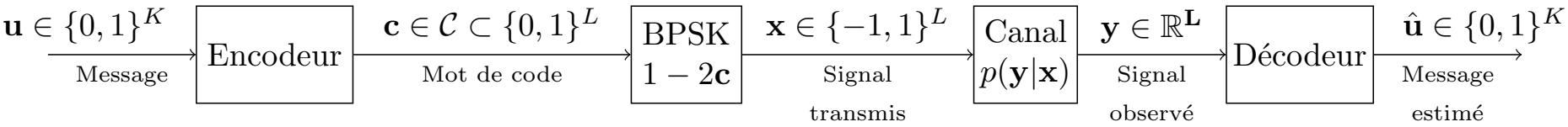
Exemple de question...

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

Introduction



- Le **décodeur** du **Maximum A Posteriori - Bit** (**MAP-Bit**) est la fonction de **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 **souple**.

Algorithme BCJR (Bahl Cocke Jelinek Raviv)

L'algorithme BCJR repose sur deux principes :

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

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

où

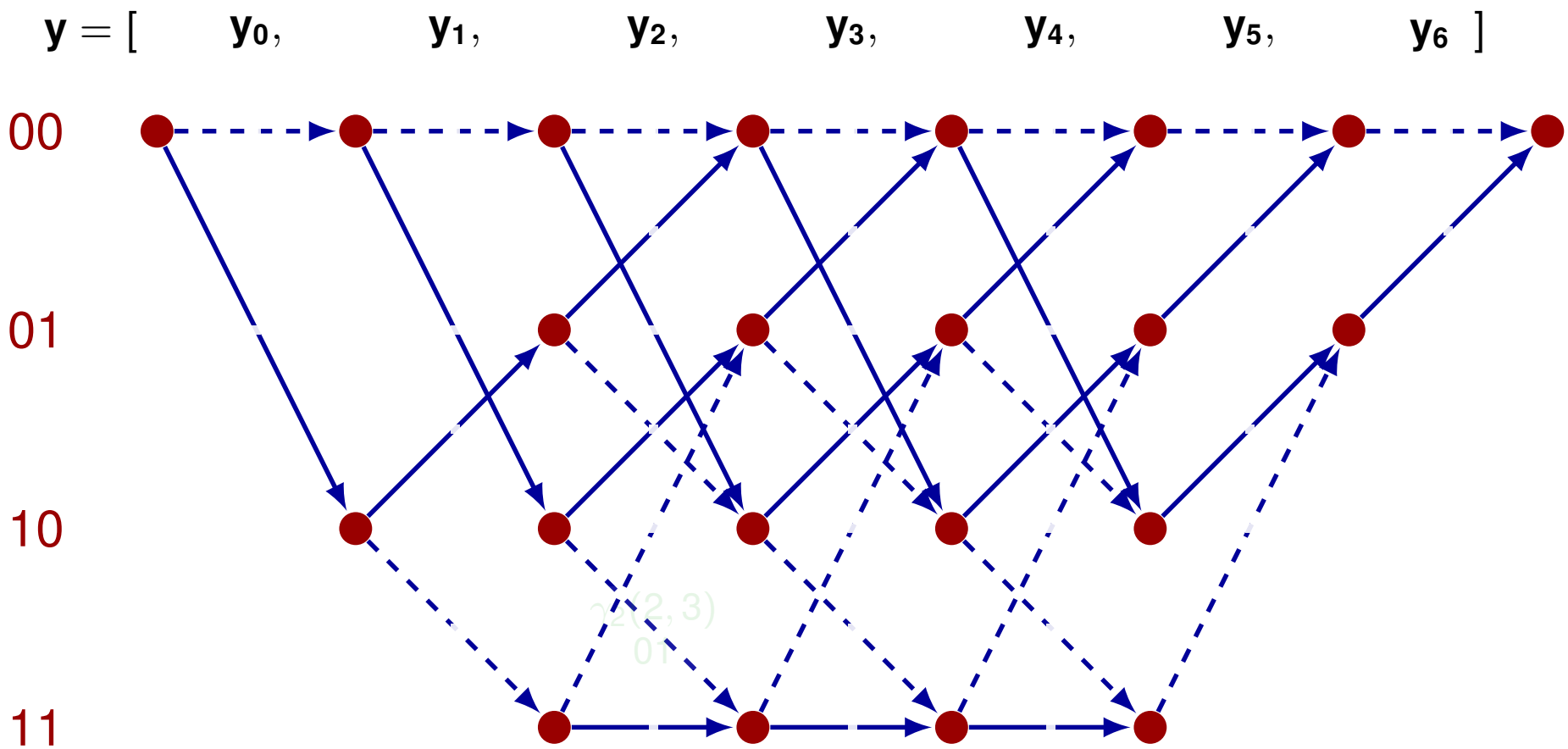
- $\alpha_\ell(s) = p(s_\ell = s, \mathbf{y}_0^{\ell-1})$ est une **métrique de nœud** appelée **métrique aller**
- $\beta_\ell(s) = p(\mathbf{y}_\ell^{L-1} | s_\ell = s)$ est une **métrique de nœud** appelée **métrique retour**
- $\gamma_\ell(s,s') = p(s_{\ell+1} = s' | s_\ell = s, \mathbf{y}_\ell)$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

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

$$\begin{aligned}
\alpha_{\ell+1}(s') &= \sum_{s\in\mathcal{S}} \alpha_\ell(s)\gamma_\ell(s,s') \\
\beta_\ell(s) &= \sum_{s'\in\mathcal{S}} \beta_{\ell+1}(s')\gamma_\ell(s,s') \\
\gamma_\ell(s,s') &= \frac{p(u_\ell(s\to s'))}{2\pi\sigma^2} \exp\left(-\frac{||\mathbf{y}_\ell - \mathbf{x}_\ell(s\to s')||^2}{2\sigma^2}\right)
\end{aligned}$$

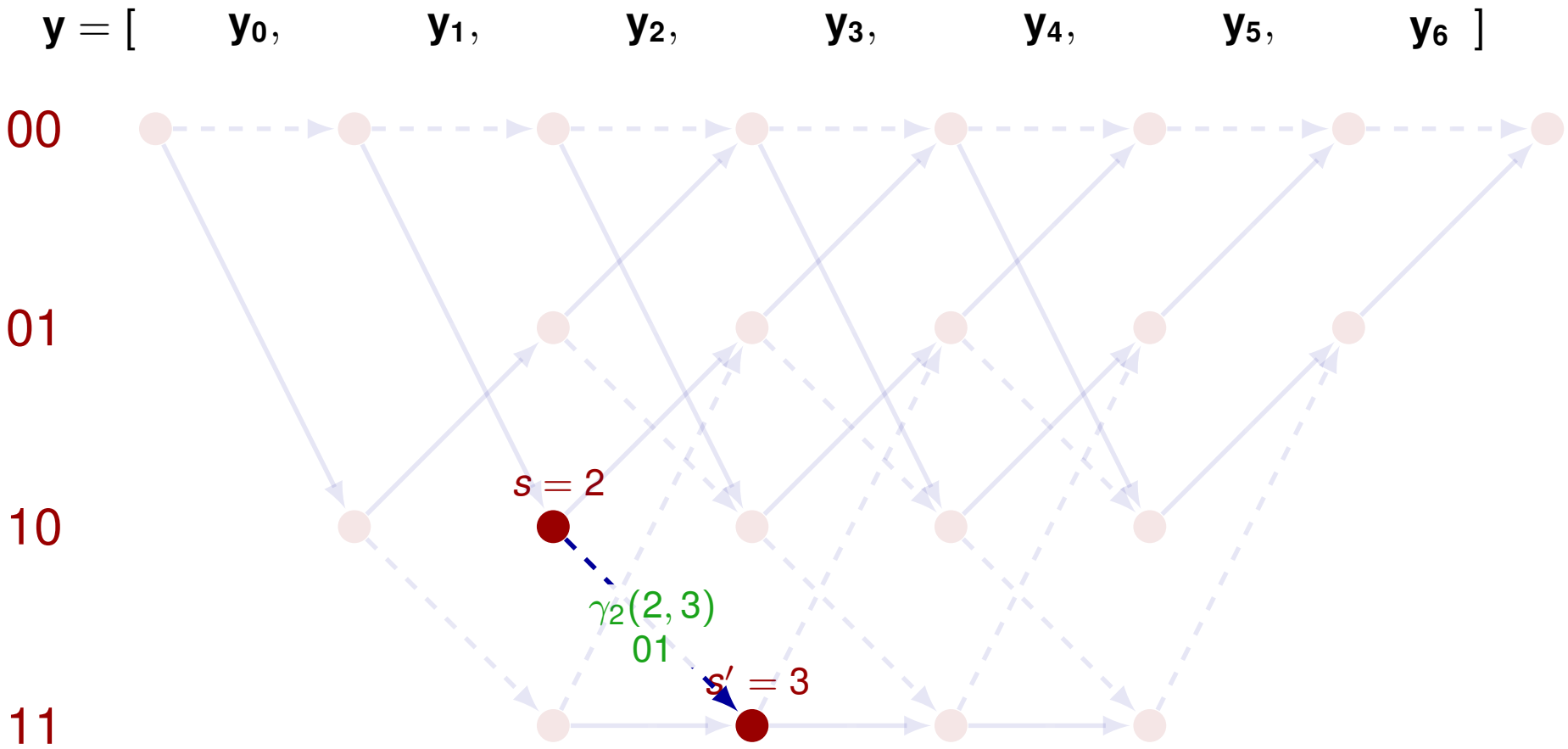
Métrique de branche $\gamma_\ell(s, s')$

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



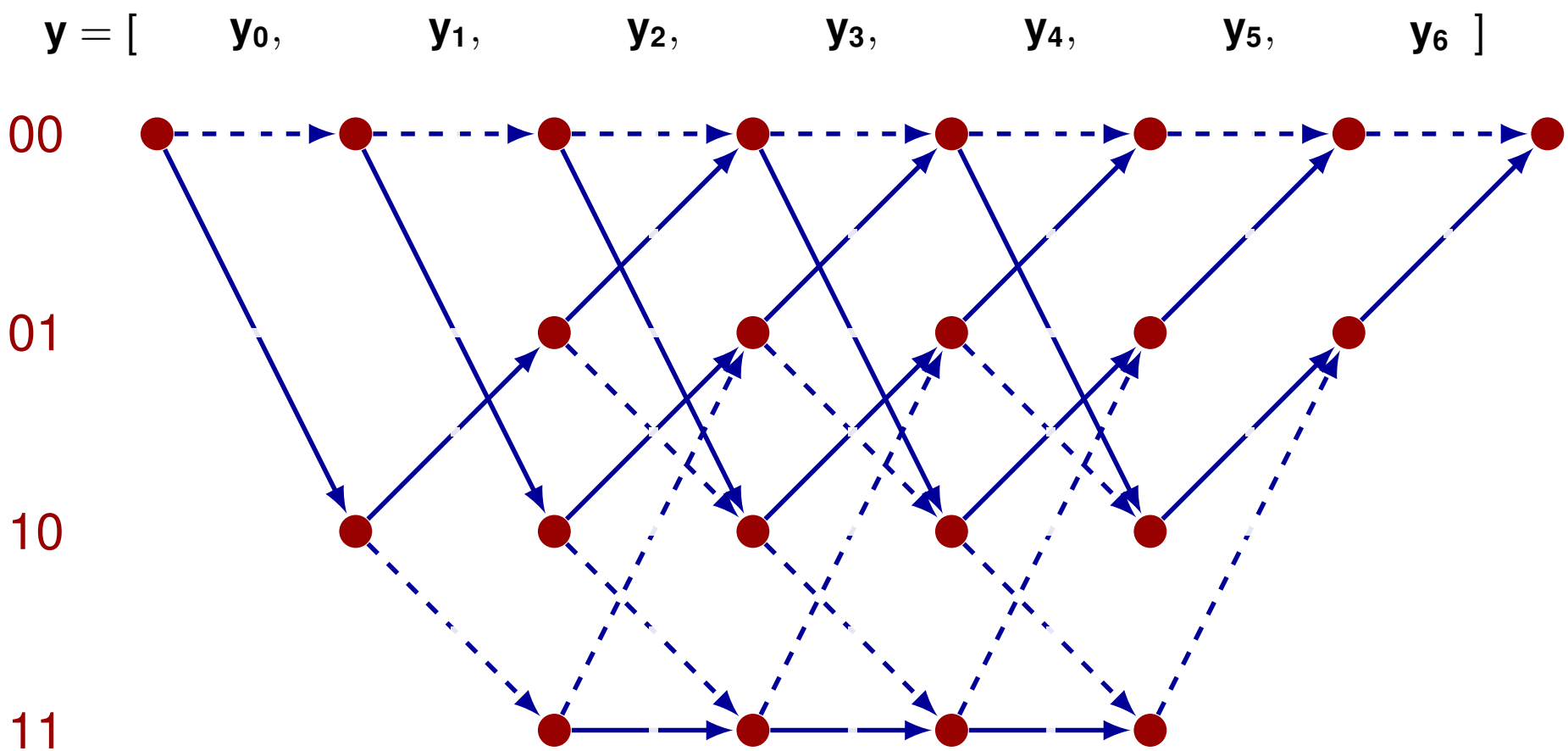
Métrique de branche $\gamma_\ell(s, s')$

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



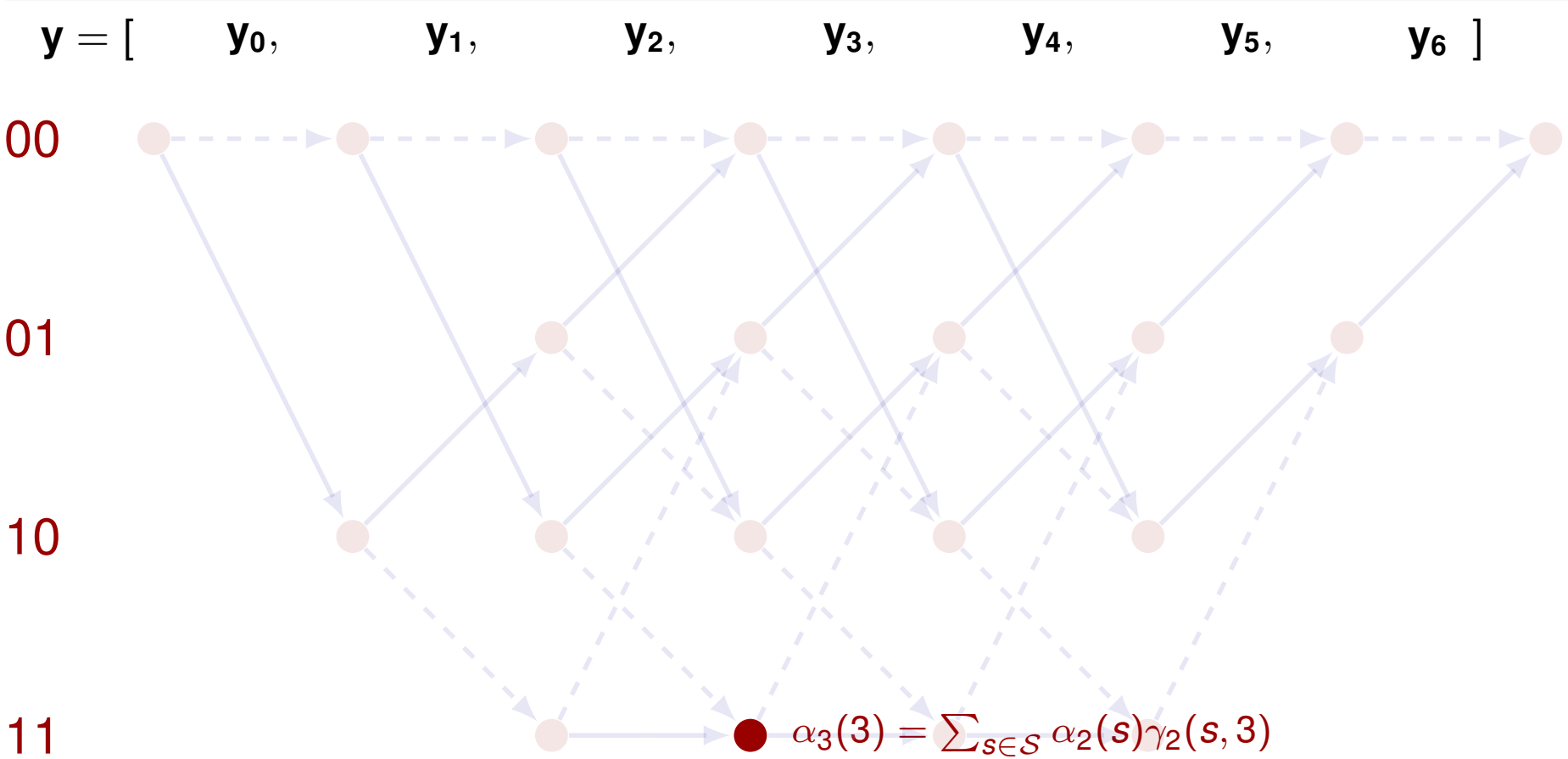
Récursion "aller" : calcul des $\alpha_\ell(s)$

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



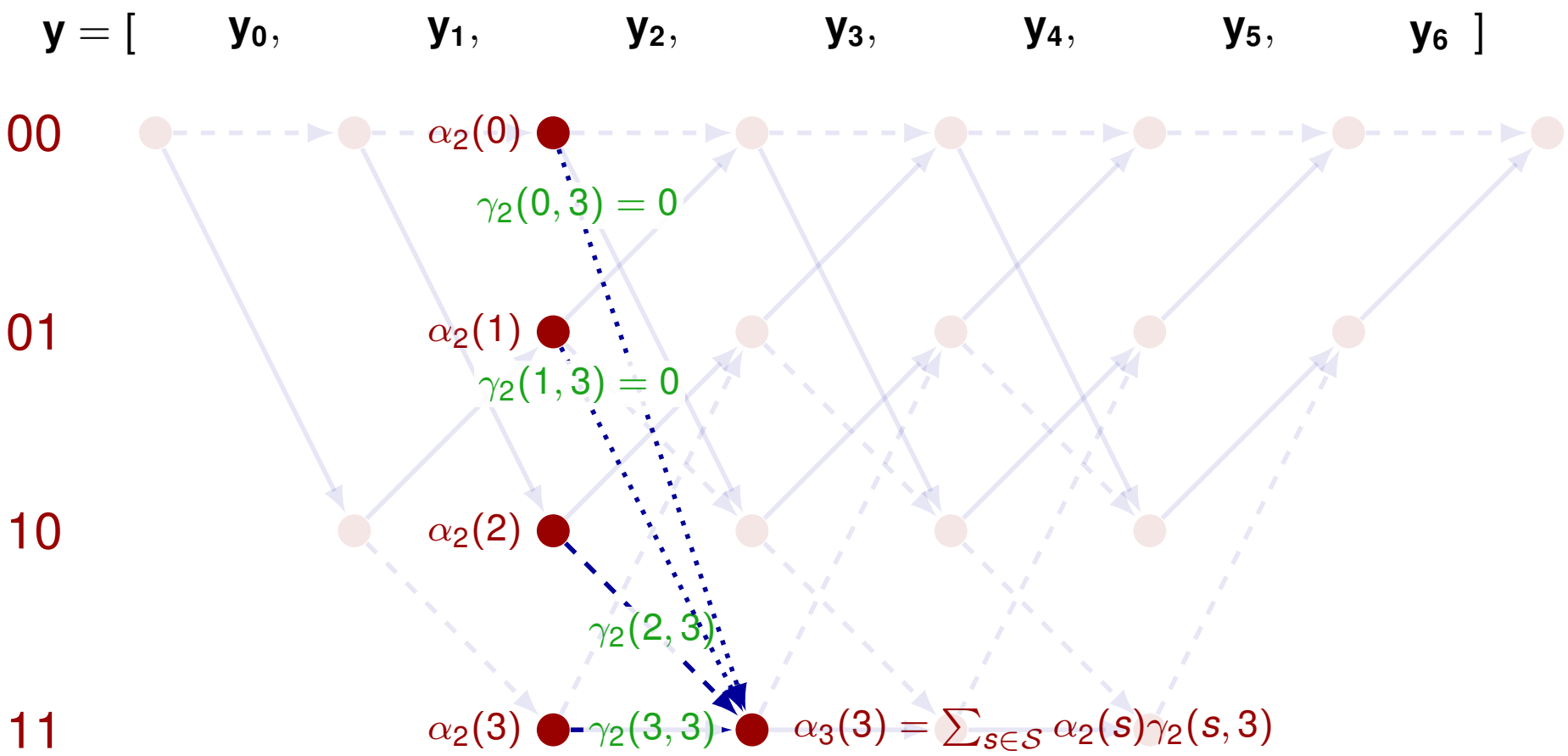
Récursion "aller" : calcul des $\alpha_\ell(s)$

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



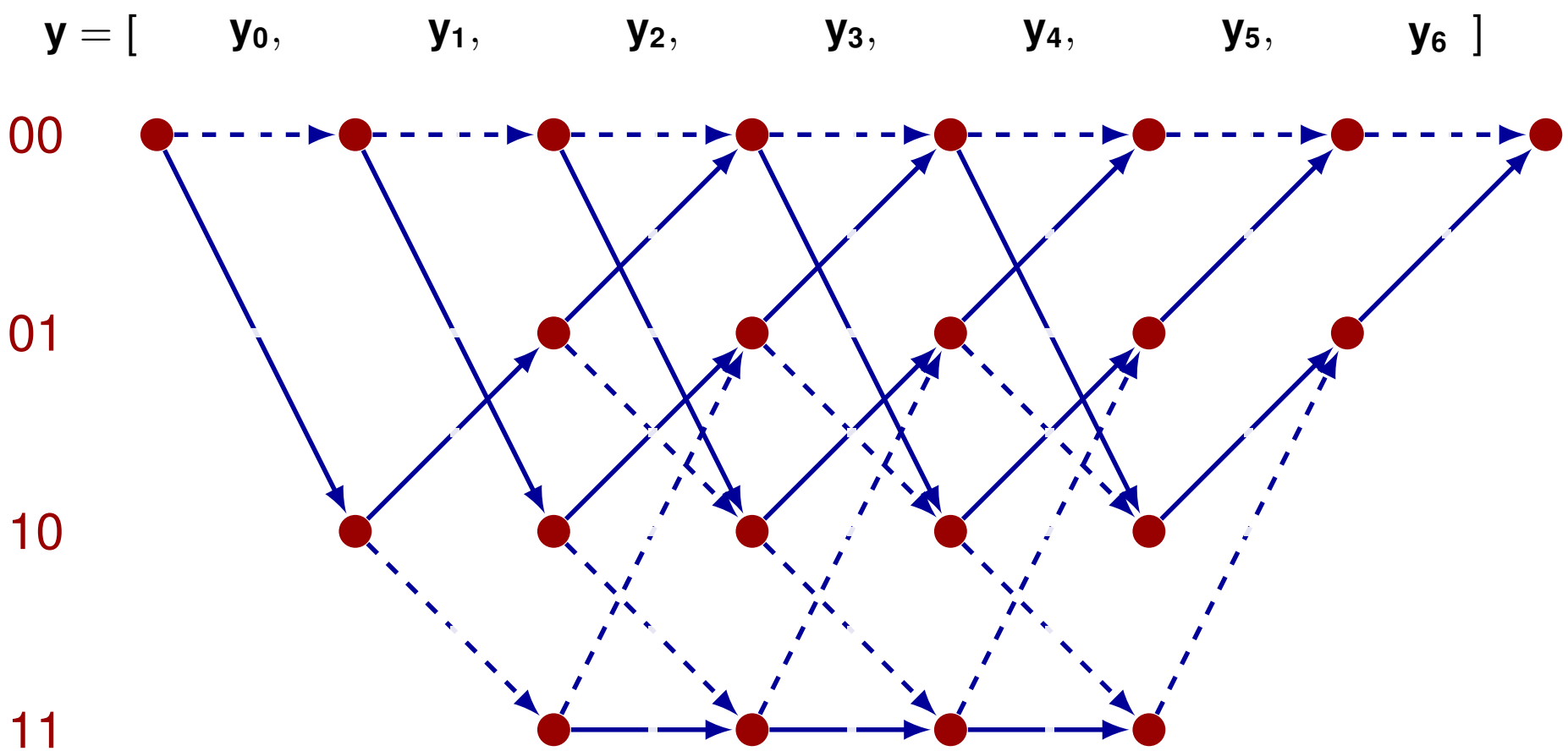
Récursion "aller" : calcul des $\alpha_\ell(s)$

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



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

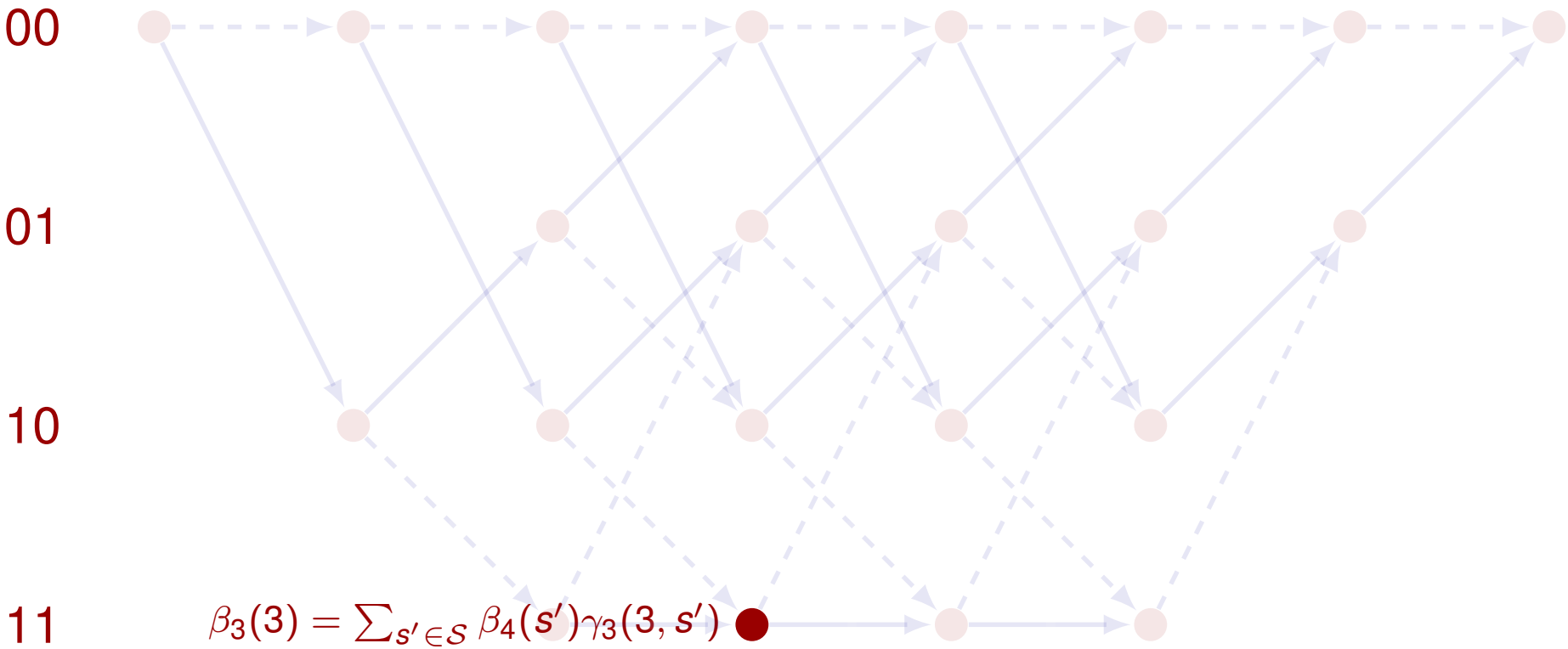
$$\beta_\ell(s) = \sum_{s' \in \mathcal{S}} \beta_{\ell+1}(s')\gamma_\ell(s, s')$$



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

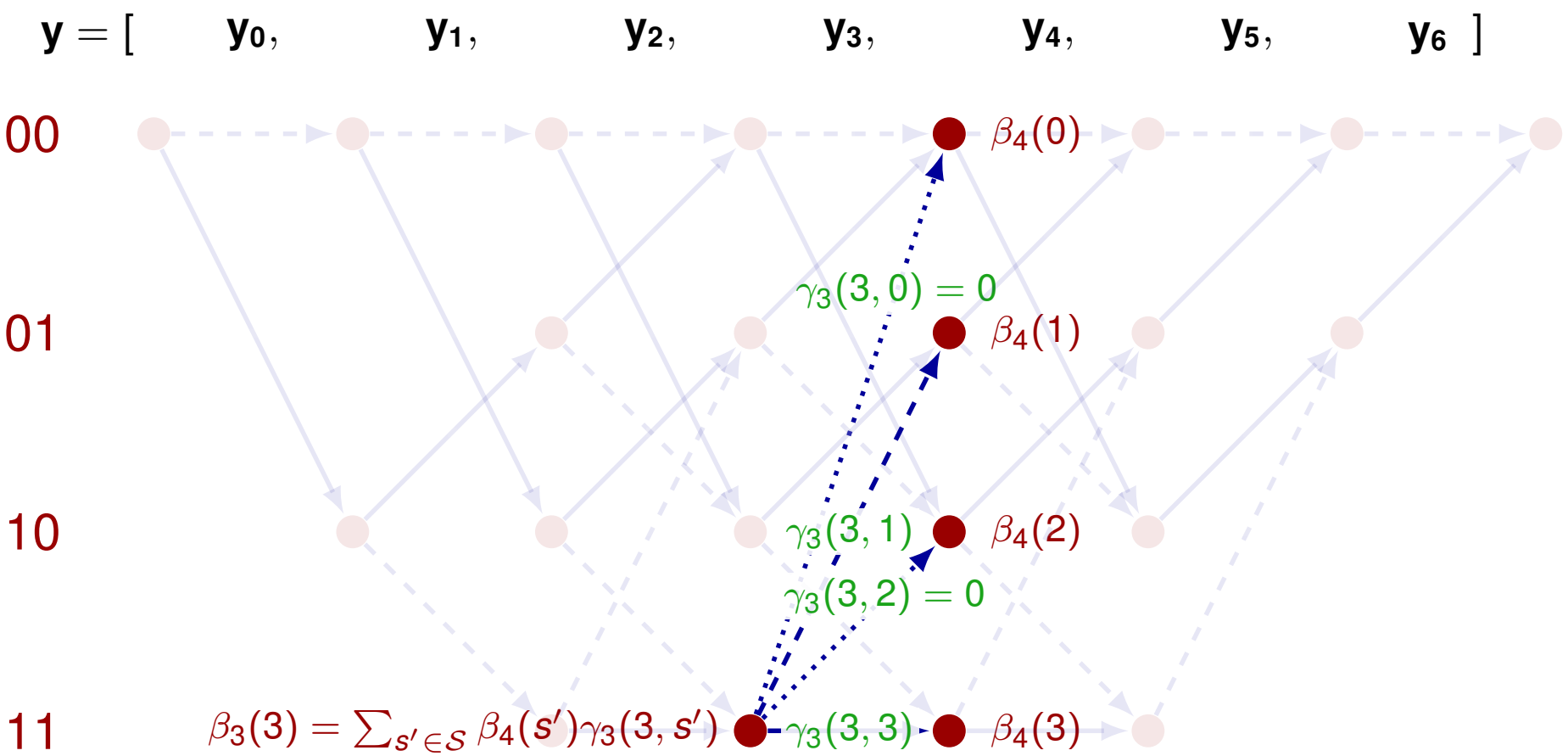
$$\beta_\ell(s) = \sum_{s' \in \mathcal{S}} \beta_{\ell+1}(s')\gamma_\ell(s, s')$$

$\mathbf{y} = [\ \mathbf{y_0},\ \mathbf{y_1},\ \mathbf{y_2},\ \mathbf{y_3},\ \mathbf{y_4},\ \mathbf{y_5},\ \mathbf{y_6}\ ]$



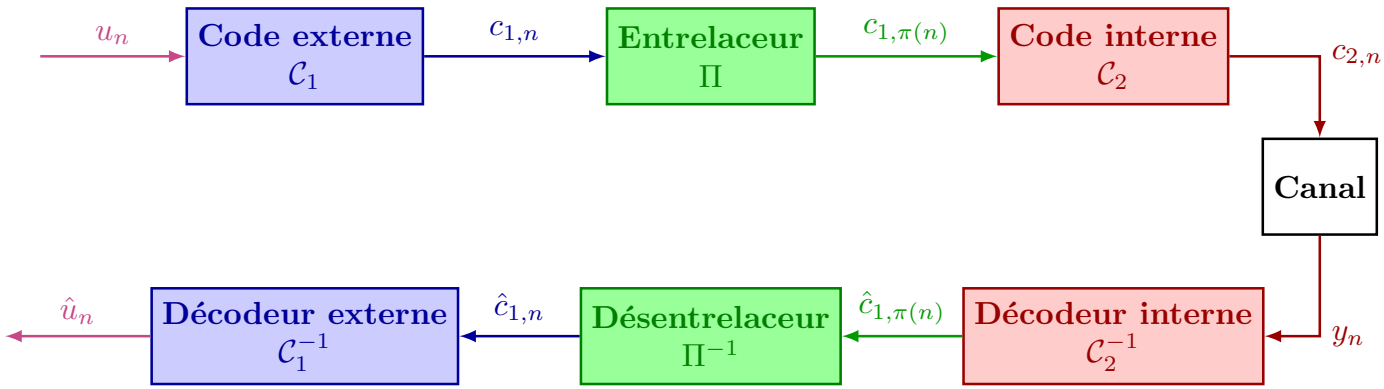
Récursion "retour" : calcul des $\beta_\ell(s)$

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



Plan

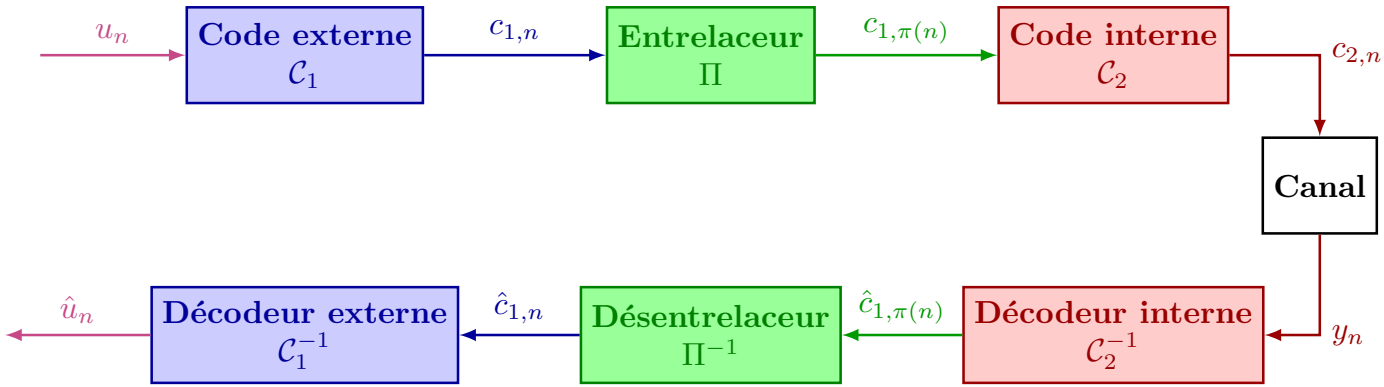
Concaténation série - Définition



Remarques

- Jusqu'aux années 1980 :
  - $\rightarrow$ $\mathcal{C}_1$: **code linéaire en bloc** corrigeant jusqu'à t erreurs (code RS, code BCH)
  - $\rightarrow$ $\mathcal{C}_2$: **code convolutif**
- **Rendement** : $R = R_1 R_2$
- **Entrelaceur** : permute les éléments de c_1
- **Désentrelaceur** : remet les éléments à leur place avant décodage
  - $\rightarrow$ Permet de disperser les erreurs commises en **rafale** par $\mathcal{C}_2^{-1}$ pour qu'elles apparaissent **isolées** et soient corrigées par $\mathcal{C}_1^{-1}$

Concaténation série - Émetteur



$u_0\ u_1\ u_2\ u_3\ |\ u_4\ u_5\ u_6\ u_7\ |\ u_8\ u_9\ u_{10}\ u_{11}$

$\Downarrow\ \mathcal{C}_1$

$c_0\ c_1\ c_2\ c_3\ c_4\ |\ c_5\ c_6\ c_7\ c_8\ c_9\ |\ c_{10}\ c_{11}\ c_{12}\ c_{13}\ c_{14}$

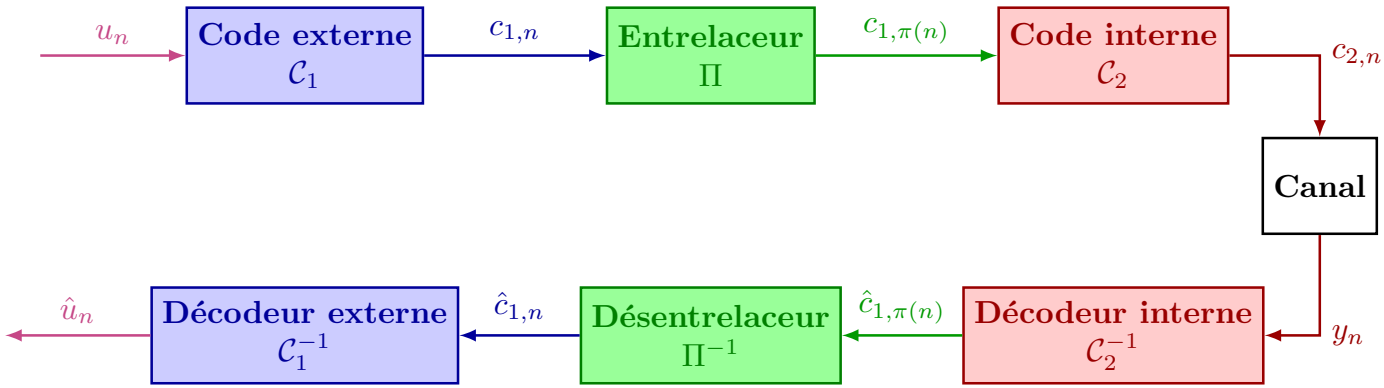
$\Downarrow\ \Pi$

$c_{10}\ |\ c_5\ |\ c_4\ |\ c_6\ |\ c_2\ |\ c_{13}\ |\ c_3\ |\ c_9\ |\ c_1\ |\ c_8\ |\ c_{12}\ |\ c_{14}\ |\ c_0\ |\ c_{11}\ |\ c_7$

$\Downarrow\ \mathcal{C}_2$

$c'_0\ c'_1\ c'_2\ c'_3\ c'_4\ c'_5\ c'_6\ c'_7\ c'_8\ c'_9\ c'_{10}\ c'_{11}\ c'_{12}\ c'_{13}\ c'_{14}\ c'_{15}\ c'_{16}\ c'_{17}\ c'_{18}\ c'_{19}\ c'_{20}$

Concaténation série - Récepteur



$y_0\ y_1\ y_2\ y_3\ y_4\ y_5\ y_6\ y_7\ y_8\ y_9\ y_{10}\ y_{11}\ y_{12}\ y_{13}\ y_{14}\ y_{15}\ y_{16}\ y_{17}\ y_{18}\ y_{19}\ y_{20}$

$\Downarrow \mathcal{C}_2^{-1}$

$\hat{c}_{10}\ \hat{c}_5\ \hat{c}_4\ \hat{c}_6\ \hat{c}_2\ \hat{c}_{13}\ \hat{c}_3\ \hat{c}_9\ \hat{c}_1\ \hat{c}_8\ \hat{c}_{12}\ \hat{c}_{14}\ \hat{c}_0\ \hat{c}_{11}\ \hat{c}_7$

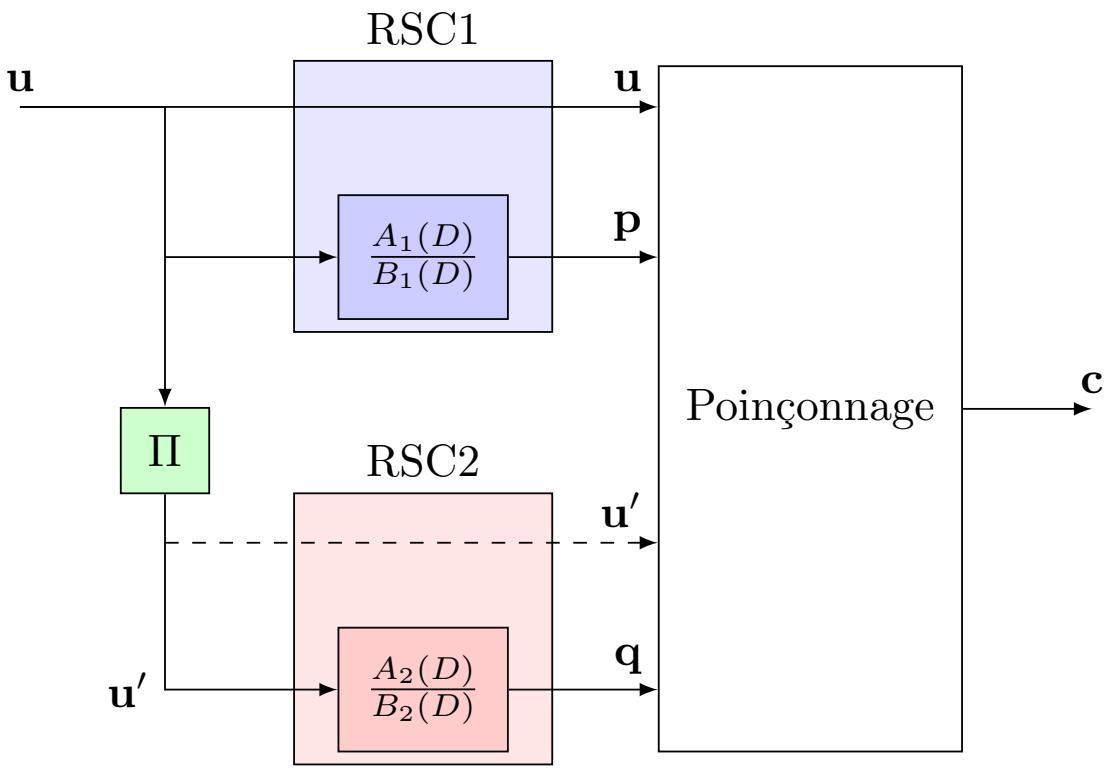
$\Downarrow \Pi^{-1}$

$\hat{c}_0\ \hat{c}_1\ \hat{c}_2\ \hat{c}_3\ \hat{c}_4\ \hat{c}_5\ \hat{c}_6\ \hat{c}_7\ \hat{c}_8\ \hat{c}_9\ \hat{c}_{10}\ \hat{c}_{11}\ \hat{c}_{12}\ \hat{c}_{13}\ \hat{c}_{14}$

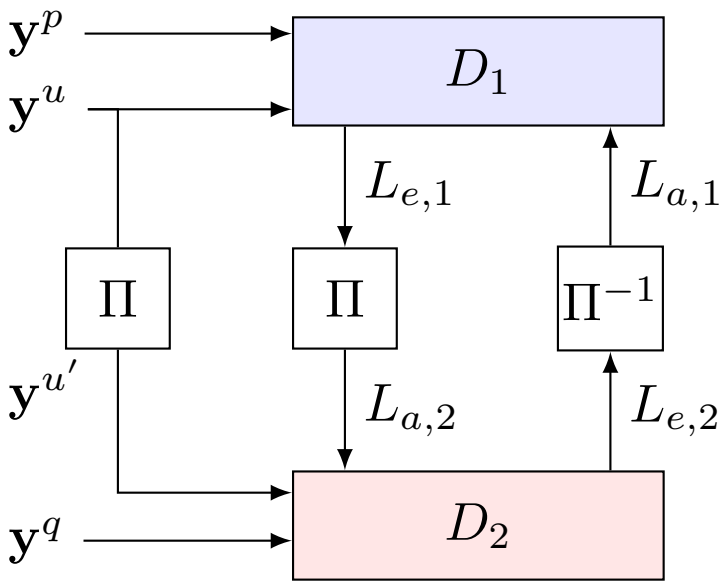
$\Downarrow \mathcal{C}_1^{-1}$

$u_0\ u_1\ u_2\ u_3\ u_4\ u_5\ u_6\ u_7\ u_8\ u_9\ u_{10}\ u_{11}$

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#