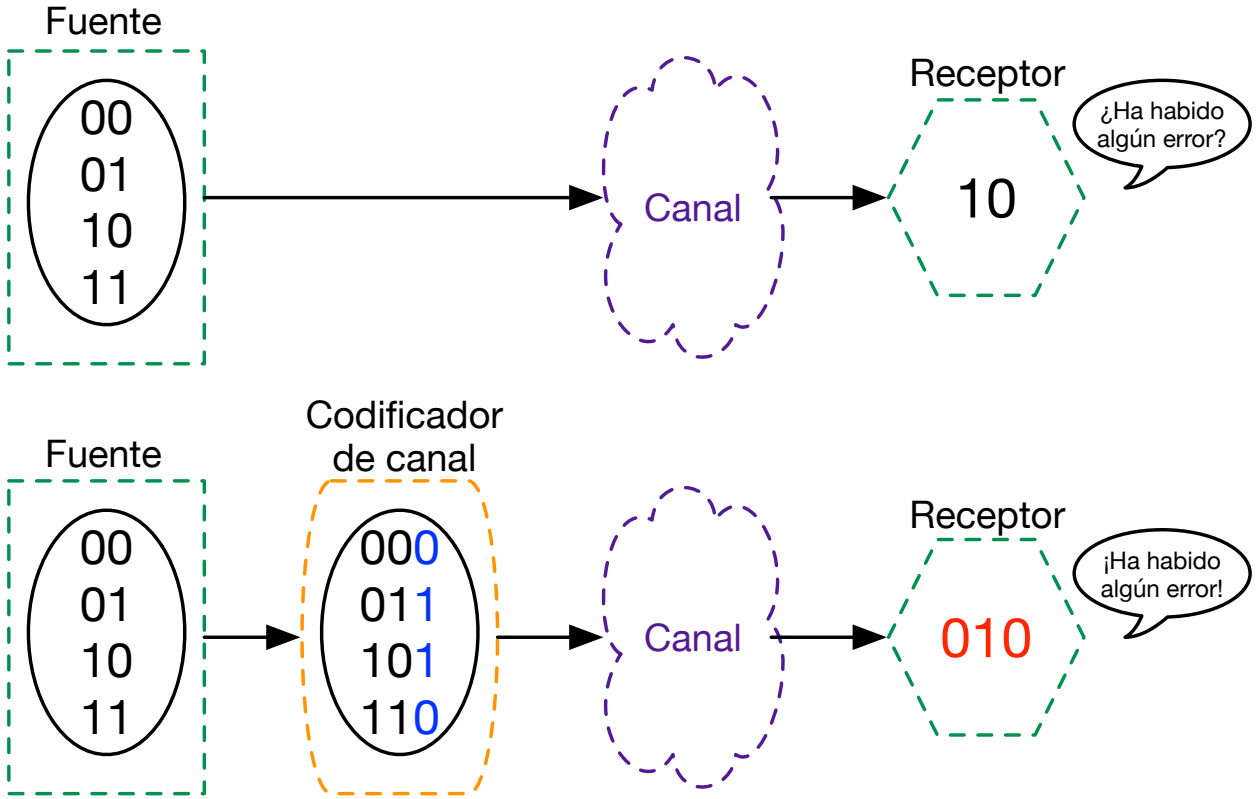
Codificación de canal

Motivación

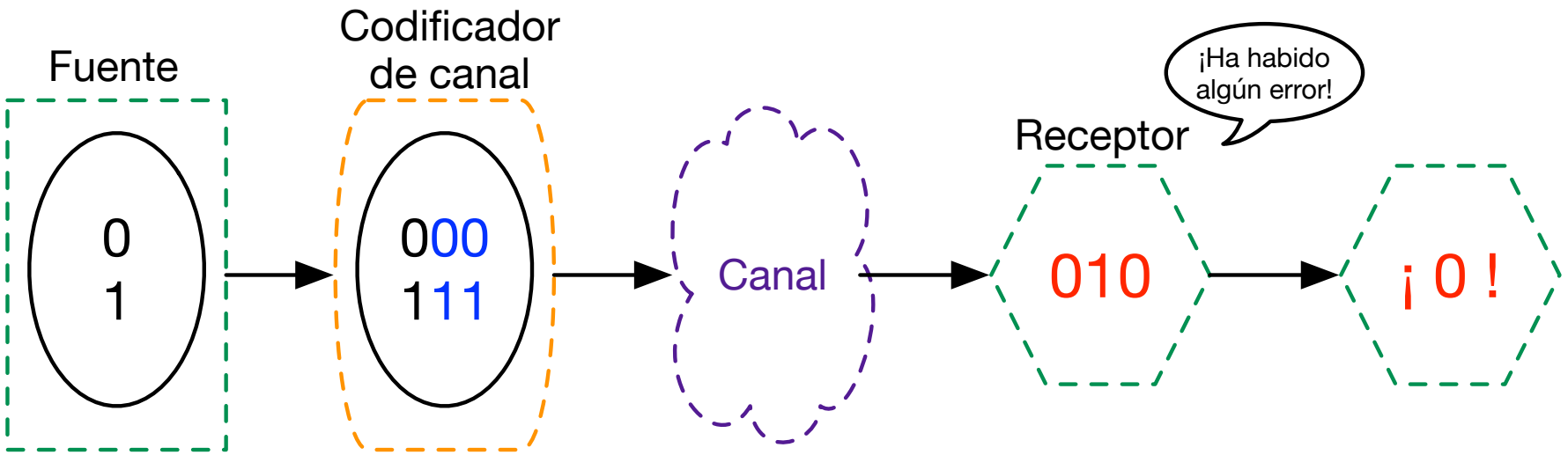
- Existencia de canales ruidosos:
  - Necesidad de transmitir información libre de errores



Códigos de repetición

- Posiblemente la solución más sencilla:
  - Repito cada bit transmitido n veces

- Por ejemplo, si $n = 3$:



Parámetros importantes

- **Tasa de codificación**: Relación entre el número de bits de información y el número de bits totales transmitidos
- **Distancia Hamming**: número de elementos distintos entre dos vectores
- **Distancia mínima**: La menor distancia Hamming entre dos palabras código válidas

Detección y corrección de errores

- Dado un código con distancia mínima d_{min}:
  - Se pueden **detectar** $d_{min} - 1$ errores
  - Se pueden **corregir** $(dmin - 1)/2$ errores

¿Y la probabilidad de error?

- Supongamos que la probabilidad de error en un bit es p_e
- La probabilidad de que ocurran *i* errores en una palabra código de n bits es:

$$p(i,n) = \binom{n}{i} \cdot p_e^i \cdot (1-p_e)^{n-i} \approx \binom{n}{i} \cdot p_e^i$$

$$\binom{n}{i} = \frac{n!}{i!\,(n-i)!}$$

Ejemplo

- Supongamos lo siguiente:
  - $p_e = 10^{-6}$
  - Código de repetición con $k = 1$ y $n = 3$

- *La probabilidad de que haya algún error pero no lo detectemos es:*

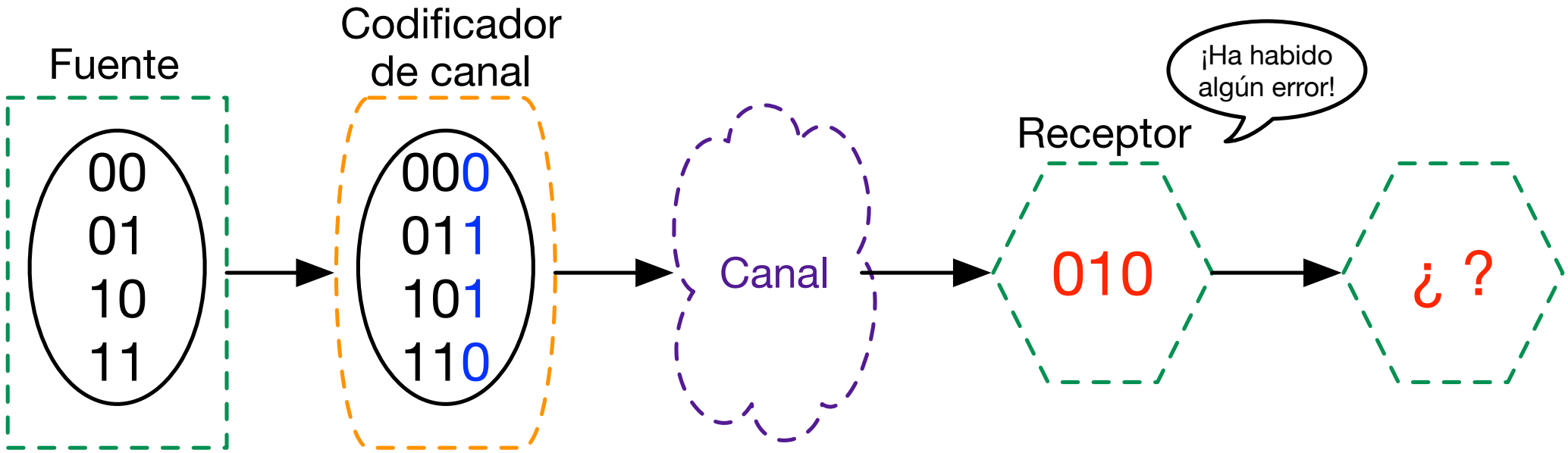
$$p(3,3) = p_e^3 = 10^{-18}$$

- *La probabilidad de que haya algún error y no lo podamos corregir:*

$$p = p(2,3) + p(3,3) = 3 \cdot p_e^2 - 2p_e^3 \approx 3 \cdot 10^{-12}$$

Códigos de paridad

- Se utiliza $n = k + 1$
- El bit de paridad es tal que el número de unos en la palabra código sea par (paridad par) o impar (paridad impar)



Códigos bloque

- Es una generalización de todo lo anterior
- Antes, un par de definiciones:
  - **Código lineal**:
    - La suma de dos vectores código es otro vector código
    - El vector 0 forma parte del código
  - **Peso de un vector**:
    - Número de unos que tiene el vector
  - **Distancia mínima:**
    - Se puede calcular como el peso mínimo del código

Códigos bloque (II)

- Supongamos un mensaje a transmitir $\boldsymbol{m} = (m_1, m_2, \ldots, m_k)$
- Y el correspondiente vector transmitido $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$
- En un código bloque, ambos vectores se relacionan como:

$$\boldsymbol{x} = \boldsymbol{m} \cdot \boldsymbol{G}$$

- **G** es la **matriz generadora** del código ($k \times n$)
- Un **código es sistemático** si la matriz generadora es:

$$\boldsymbol{G} = [\boldsymbol{I_k} \mid \boldsymbol{P}]$$

Códigos bloque (III)

- ¿Y cómo decodificamos el mensaje?
- Se utiliza la **matriz de paridad** $(n-k \times n)$:

$$\boldsymbol{H} = [\boldsymbol{P^T} \mid \boldsymbol{I_{n\_k}}]$$

- Si recibimos un vector $\boldsymbol{y}$ (que será el vector transmitido $\boldsymbol{x}$ más ruido, $\boldsymbol{e}$), haremos:

$$\boldsymbol{s} = \boldsymbol{y}\,\boldsymbol{HT}$$

- A $\boldsymbol{s}$ se le llama **síndrome**, y si es nulo, la palabra código $\boldsymbol{y}$ es válida.

Códigos bloque (IV)

- Vale, bien, pero ¿y qué hay de lo de corregir errores?
- Se puede demostrar que el síndrome depende exclusivamente del error cometido ($\boldsymbol{e}$), y no del vector transmitido ($\boldsymbol{x}$)
- Problema:
  - Si x tiene n bits, hay 2^n errores posibles
  - El síndrome sólo tiene $n - k$ bits -> Sólo hay 2^{n-k} síndromes posibles

Códigos bloque (V)

- Decodificación de máxima verosimilitud:
  - Tenemos que crear una tabla con los síndromes correspondientes a los 2^{n-k-1} vectores de error más probables.
  - El decodificador:
    - Calcula el síndrome ($\boldsymbol{s} = \boldsymbol{y}\,\boldsymbol{H}^{T}$)
    - Busca en la tabla ese síndrome
    - Obtiene el vector de error correspondiente (ê)
    - Obtiene cuál sería la palabra decodificada ($\boldsymbol{y} + \hat{e}$)

Códigos bloque (VI)

- ¿Y cómo se define la matriz generadora?
- Hay muchos métodos, pero vamos a ver sólo uno: **códigos Hamming**
- Se caracterizan por:
  - Bits de control: $q = n - k \geq 3$
  - $n = 2^q - 1$
  - Independientemente de q, $d_{min} = 3$
    - Podemos detectar hasta 2 errores
    - Podemos corregir hasta un error

Códigos Hamming

- Para crear la matriz de comprobación de paridad colocamos todos los vectores binarios posibles en las columnas, ordenados de forma que quede la matriz identidad al final:

$$H = \left[\begin{array}{cccc|ccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

Códigos Hamming (II)

- A partir de ahí es inmediato calcular $\boldsymbol{G}$, sabiendo que $\boldsymbol{H} = [\boldsymbol{P}^T | \boldsymbol{I}_q]$:

$$G = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right]$$

Decodificación dura y blanda

- **Decodificación dura:**
  - Es lo que hemos venido haciendo hasta ahora
  - La palabra recibida se decodifica bit a bit
  - Se elige aquella palabra código a una distancia Hamming menor

- **Decodificación blanda:**
  - En este caso se elige el símbolo a una menor distancia euclídea.
  - La probabilidad de error va a ser menor

Ejemplo

- Código de paridad con n=3
- Utilizamos una PAM unipolar con amplitudes 0V y 1V
- Supongamos que:
  - Se transmite el símbolo **011**
  - Es decir, transmitimos por el canal la señal **0V – 1V – 1V**
  - En recepción tenemos: **0.2V – 0.45V – 0.7V**

Decodificación dura

- Al decodificar la señal recibida, el receptor obtiene:
  - 0.2V -> 0
  - 0.45V -> 0
  - 0.7V -> 1
- Es decir, recibimos la secuencia **001**, que sabemos que es incorrecta.
- Hay cuatro posibilidades: 000, 011, 101 y 110.
- Las distancias Hamming son 1, 1, 1 y 3
- Elegiremos una de las tres primeras al azar.
- Acertaremos **1 de cada 3 veces**

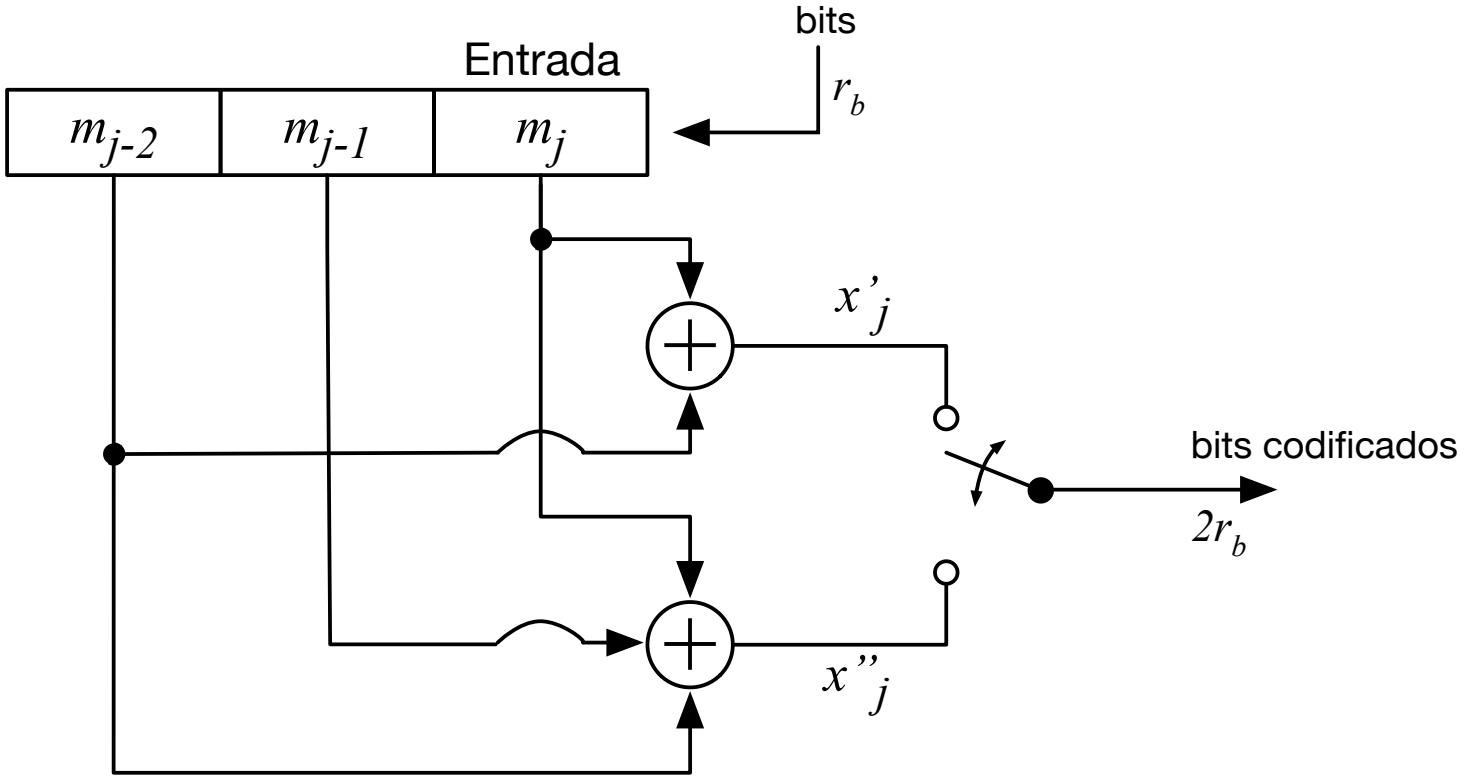
Decodificación blanda

- Recibimos lo mismo que antes: 0.2V – 0.45V - 0.7V
- En este caso calculo la distancia euclídea a cada uno de los posibles símbolos de mi constelación:
  - 000 -> $(0-0.2)^2+(0-0.45)^2+(0-0.7)^2=$ 0.73
  - 011 -> $(0-0.2)^2+(1-0.45)^2+(1-0.7)^2=$ 0.43
  - 101 -> $(1-0.2)^2+(0-0.45)^2+(1-0.7)^2=$ 0.93
  - 110 -> $(1-0.2)^2+(1-0.45)^2+(0-0.7)^2=$ 1.43

- Elegiremos el símbolo **011**

Códigos convolucionales

- Principal diferencia: tienen memoria
- Ejemplo: código (n,k,L) = (2,1,2)



$$x'_j = m_j \oplus m_{j-2}$$
$$x''_j = m_j \oplus m_{j-1} \oplus m_{j-2}$$

Árbol del código

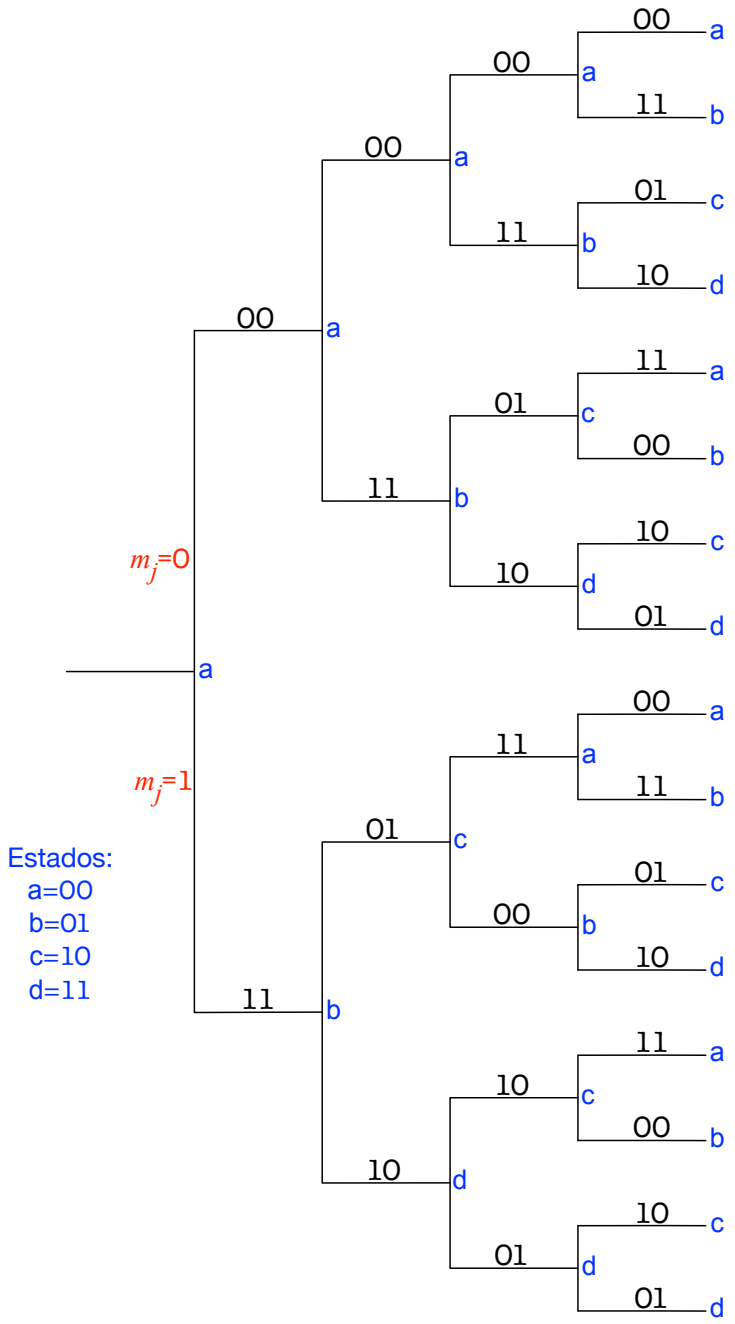
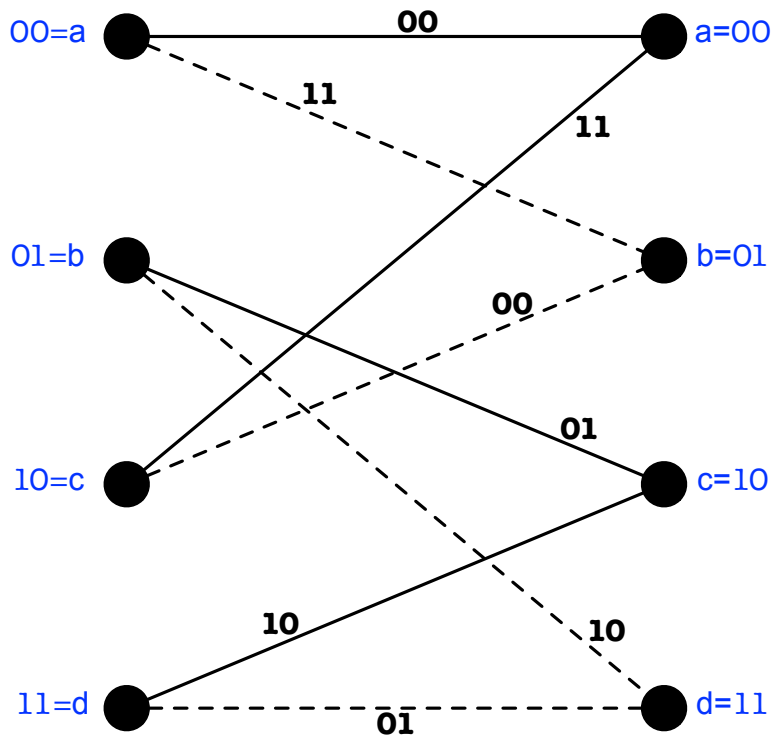
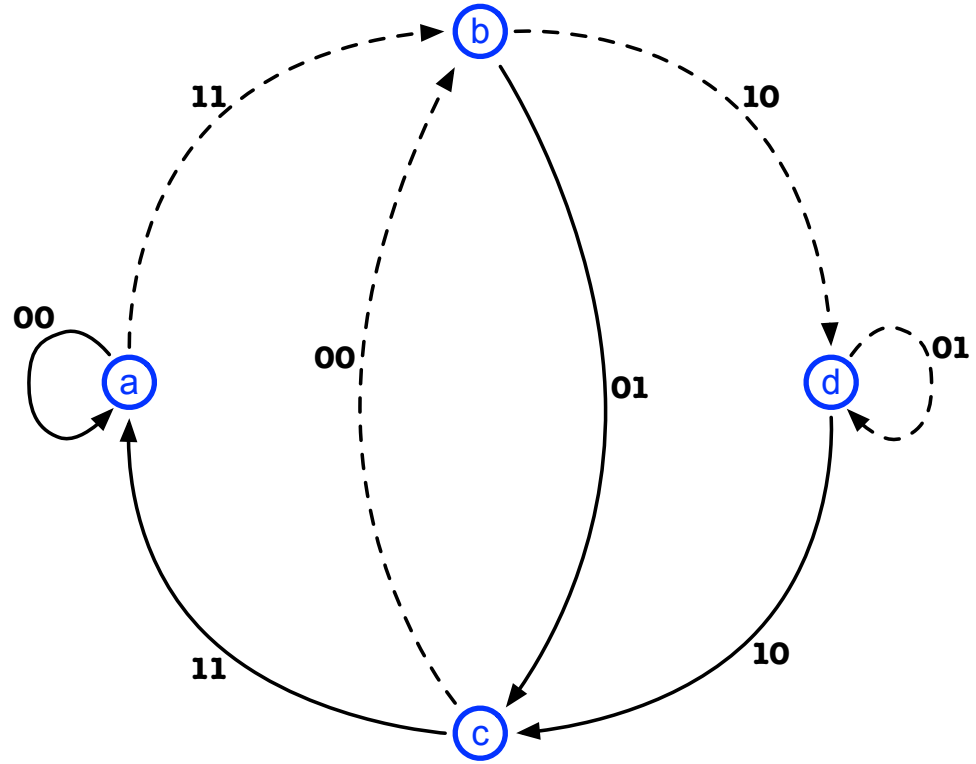
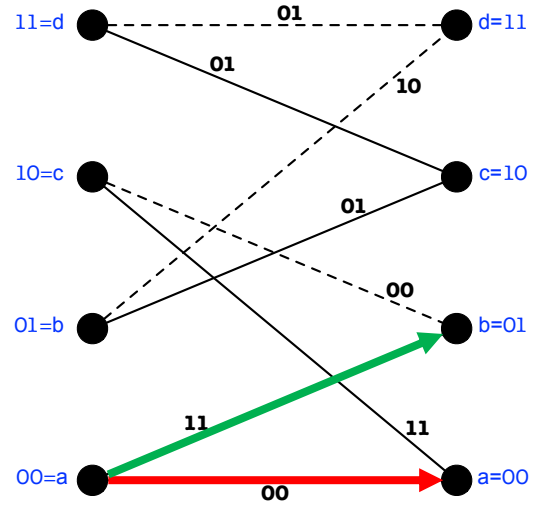


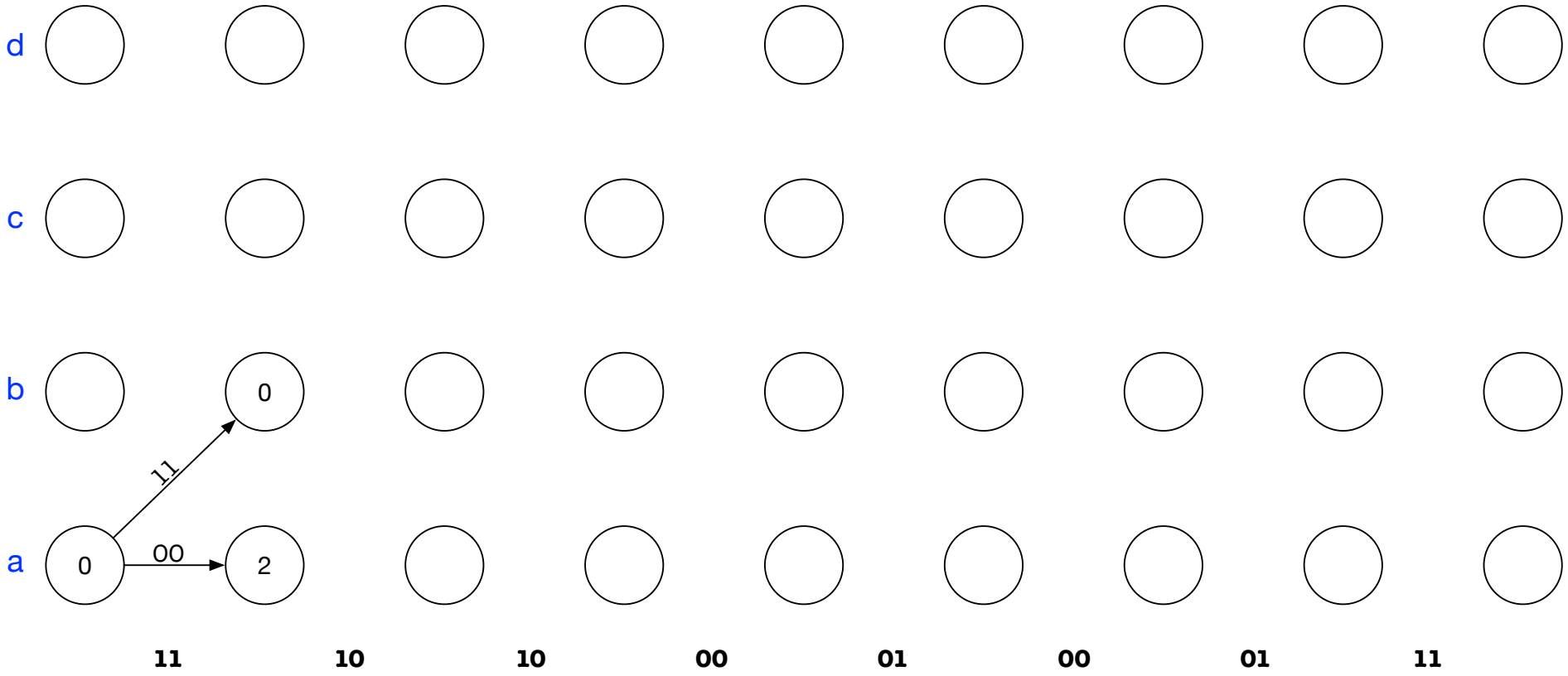
Diagrama de rejilla y de estados



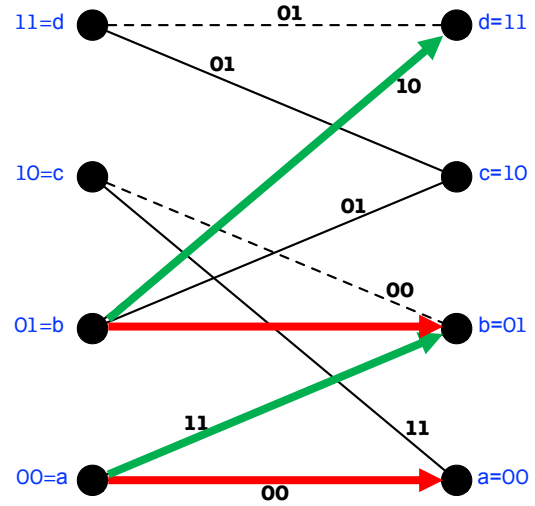


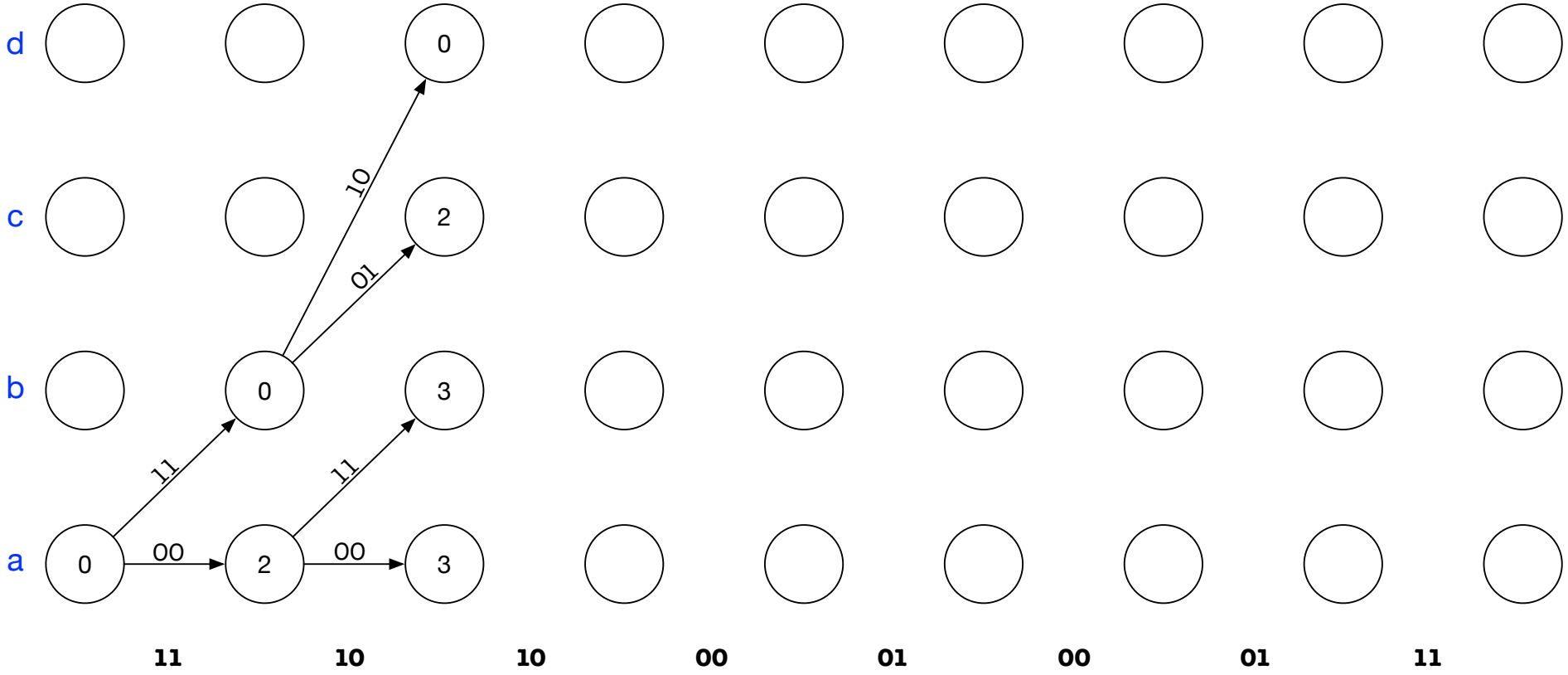
Algoritmo de Viterbi



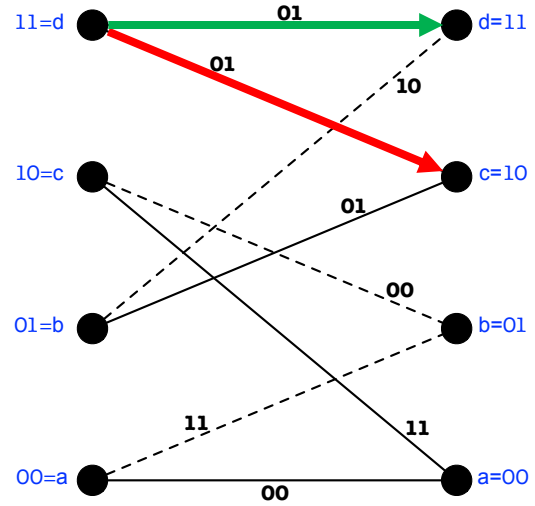


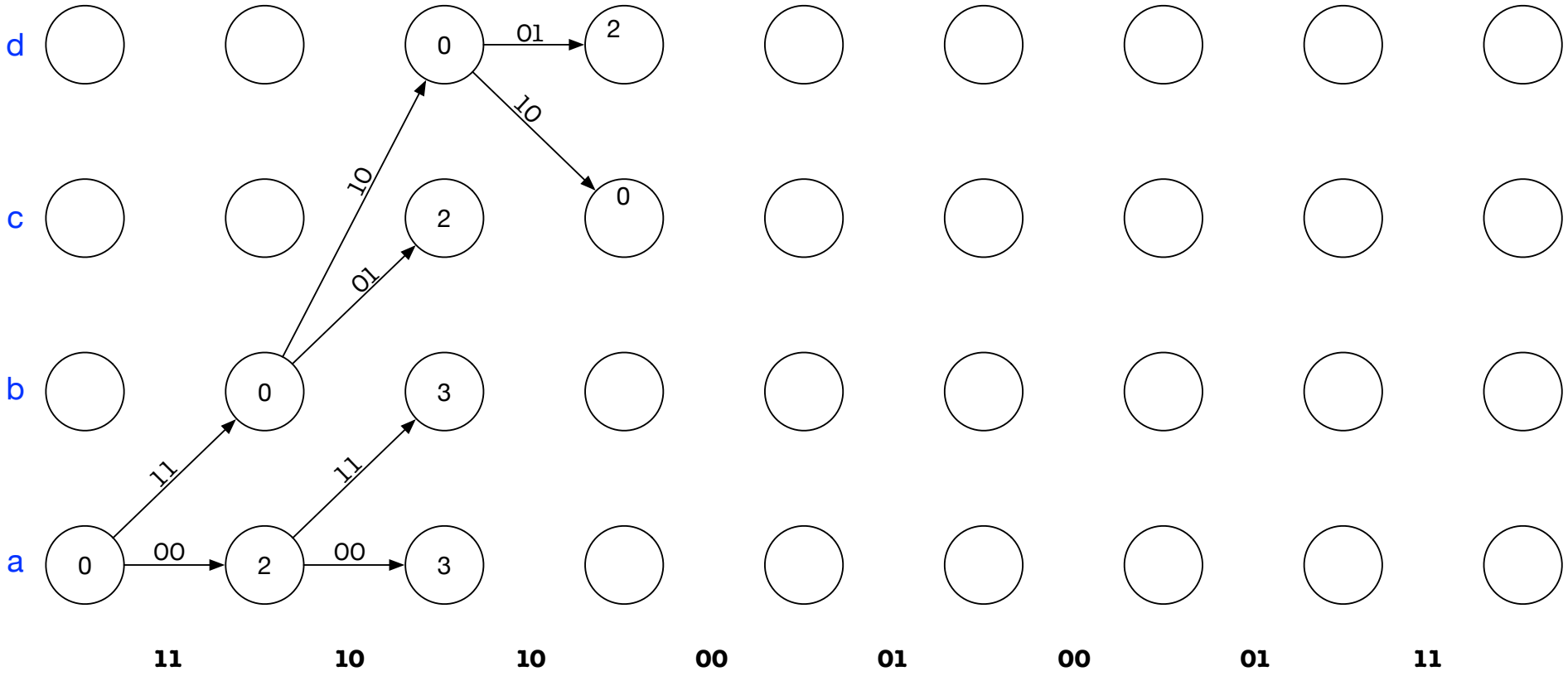
Algoritmo de Viterbi



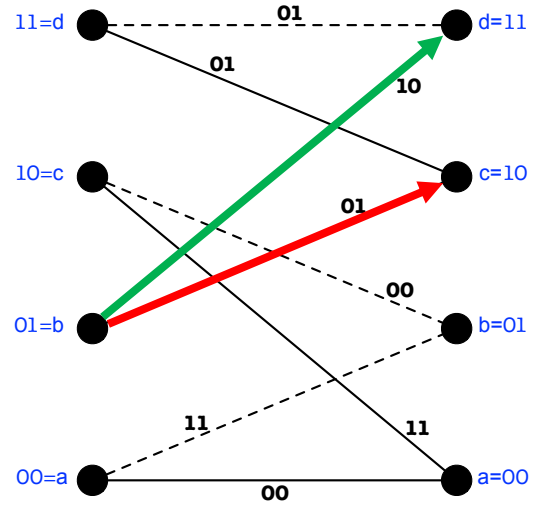


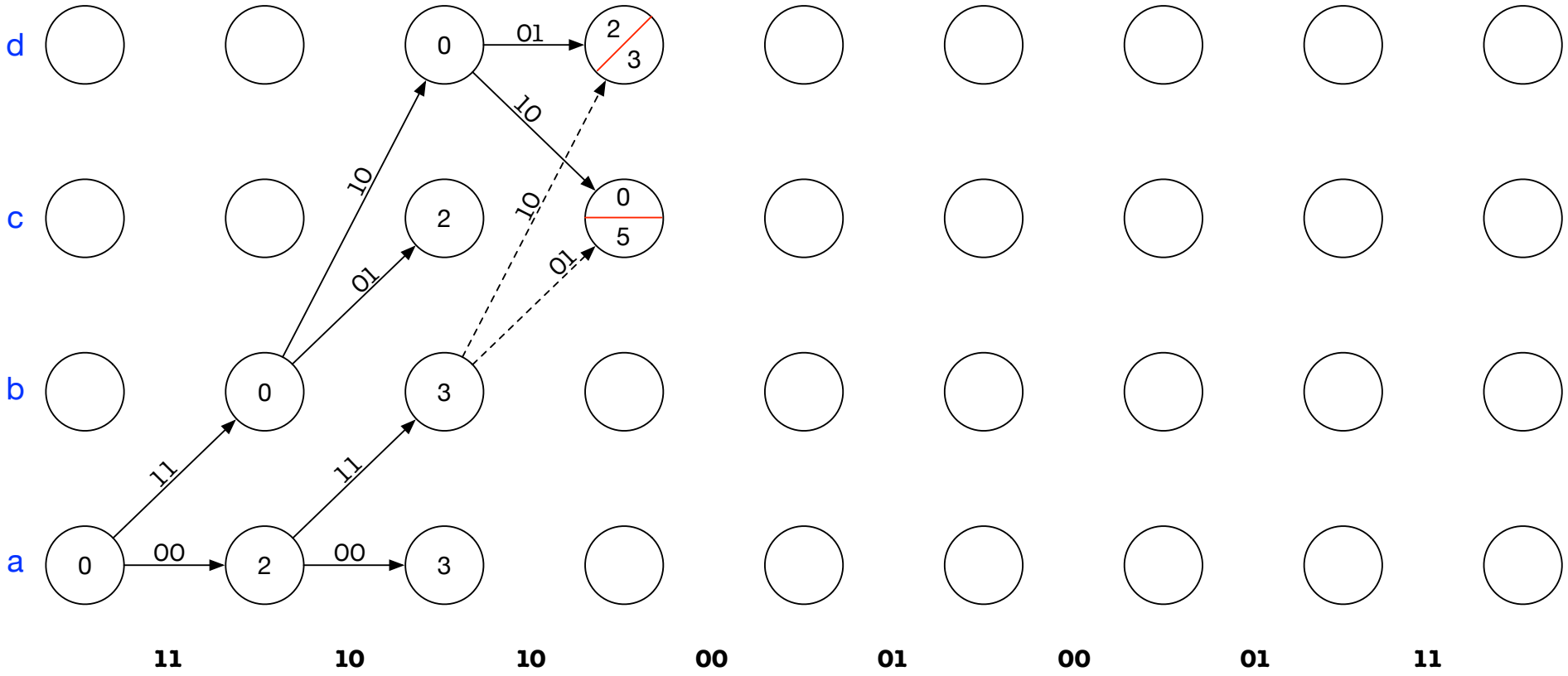
Algoritmo de Viterbi



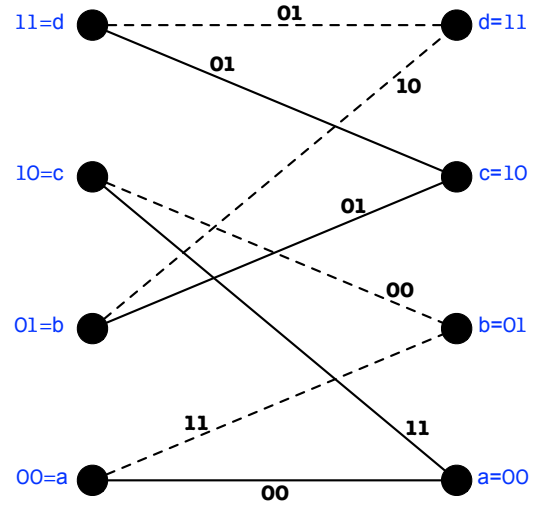


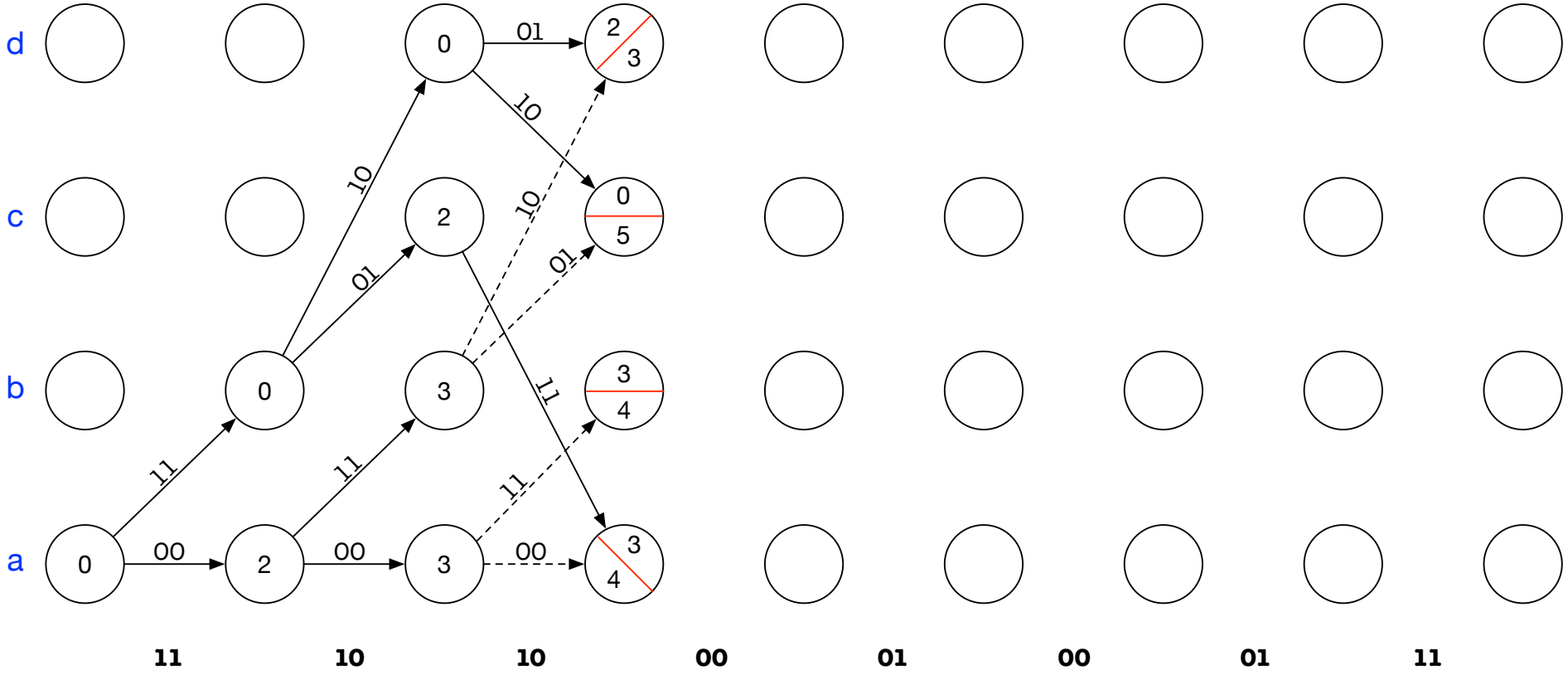
Algoritmo de Viterbi



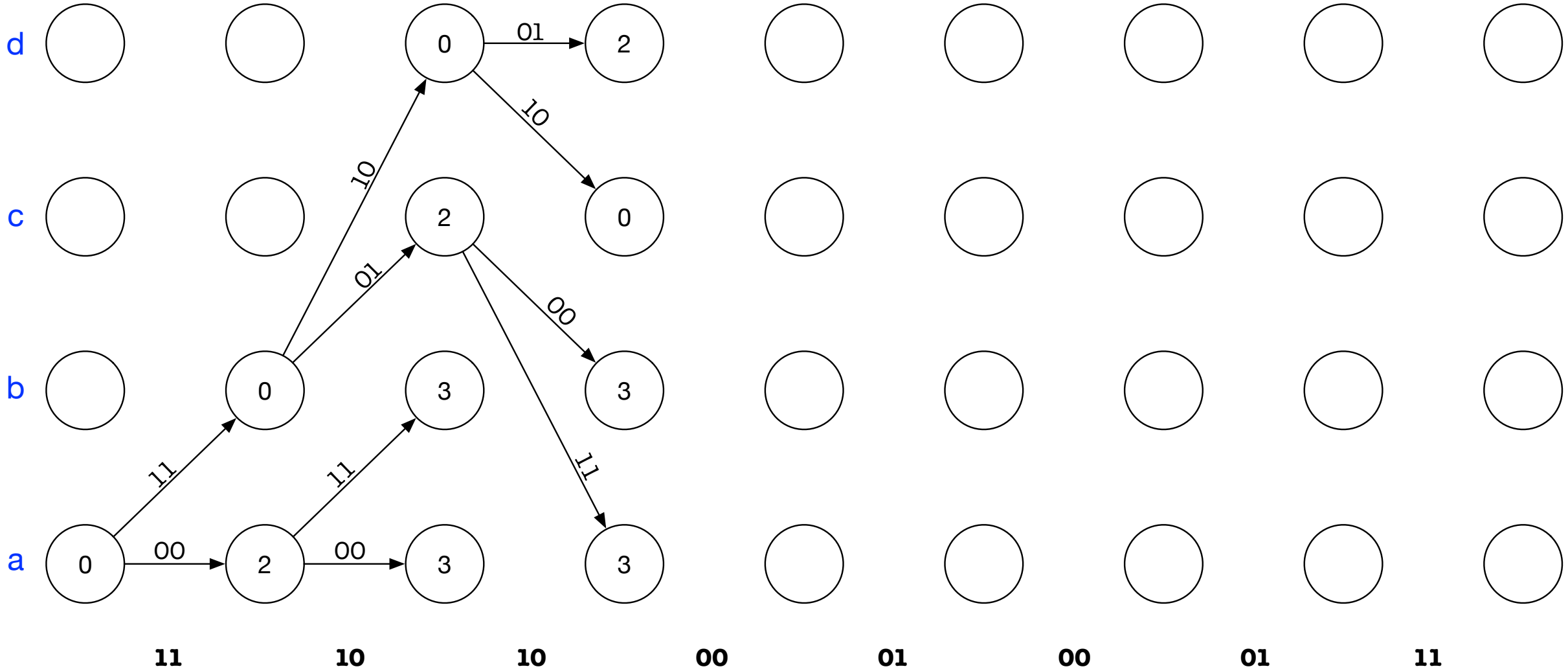


Algoritmo de Viterbi





Algoritmo de Viterbi



Algoritmo de Viterbi