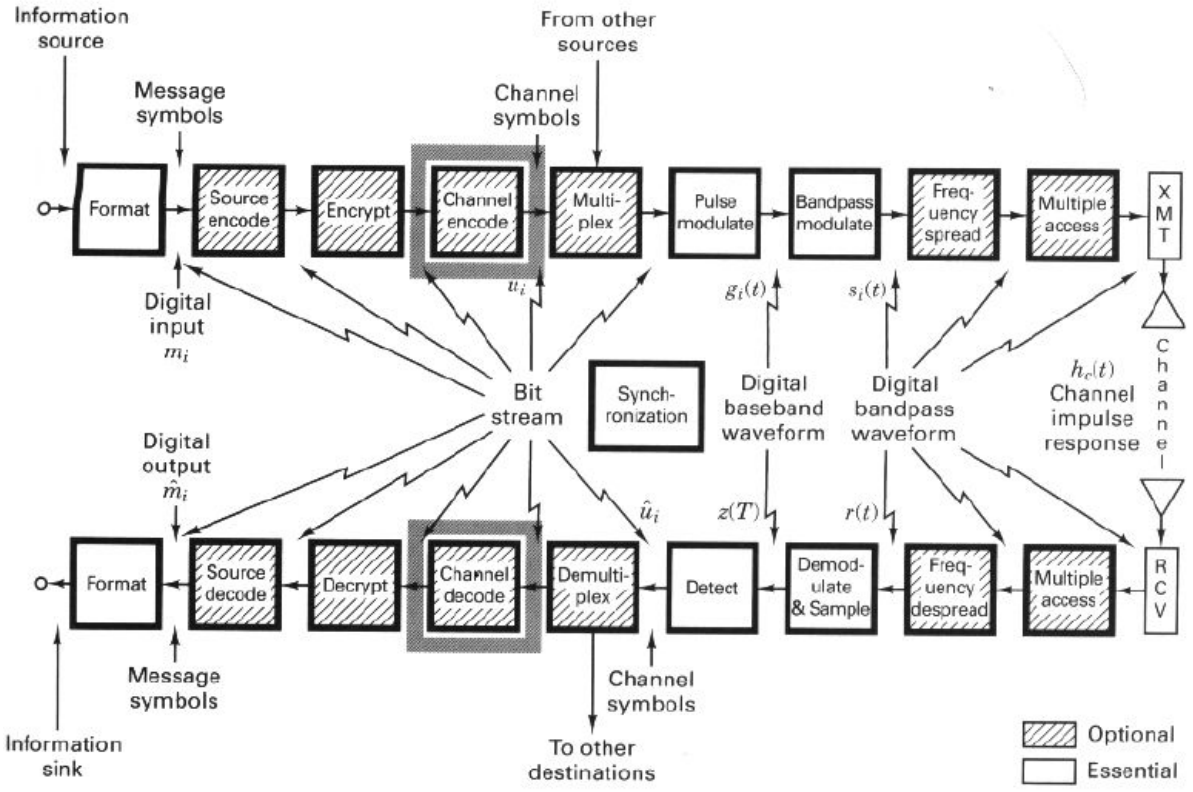
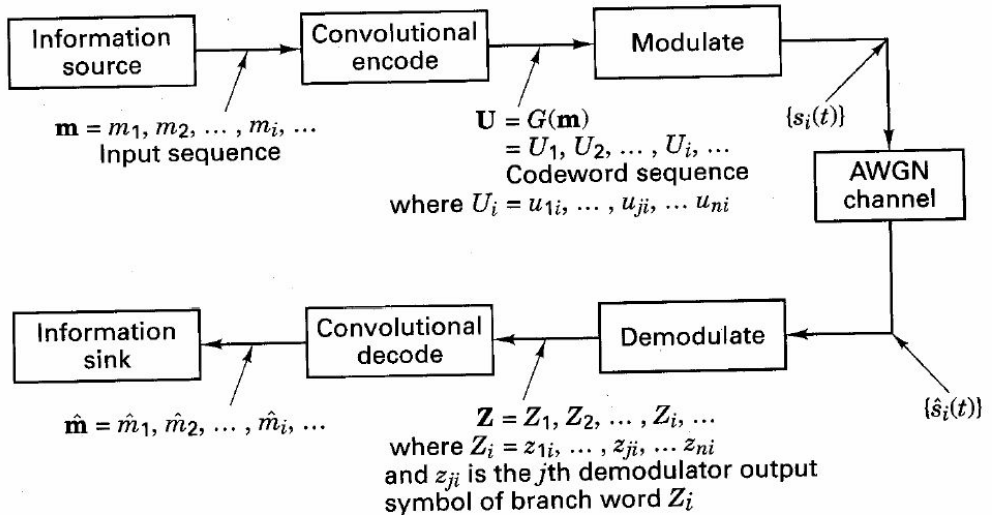
Codificación de Canal Convolucional

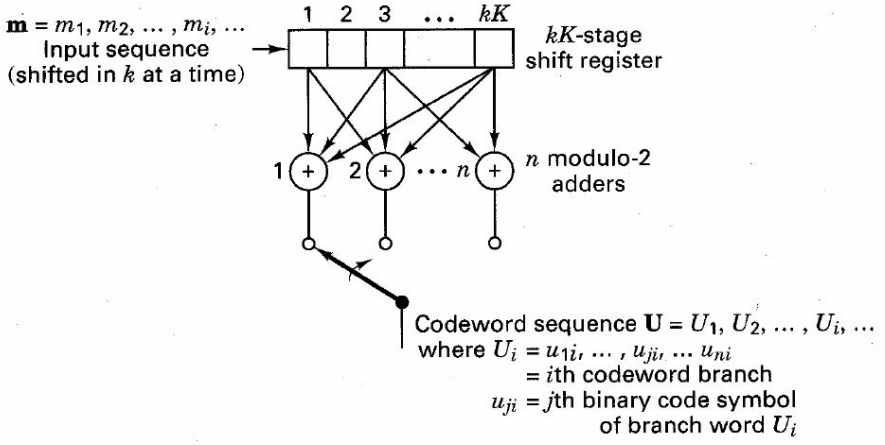


Cuando se habla de codificación de canal del tipo convolucional se hace referencia a un sistema de comunicación digital cuya implementación del codificador y decodificador respeta las características de los códigos convolucionales.



Un código convolucional está descrito por tres variables, definidas como n, k y K, donde “n” es la cantidad de bits que se obtienen producto del proceso de codificación ante la entrada de “k” bits. Es importante destacar que “k” no esta asociado a una agrupación de los mensajes digitales con

una cantidad fija para asignar a una codeword válida como se realizaba en los codificadores de bloques lineales. En este caso, “k” representa la cantidad de bits que se desplazan sobre un registro de desplazamiento por cada clock del sistema. Por otro lado, “K” hace referencia a la dimensión del registro, es decir, se podría pensar como una cantidad “K” de flip-flop tipo D conectados en serie que por cada clock del sistema, desplazan los bits de izquierda a derecha desde el registro 1 hasta el “K”. Una vez establecidos los registros, estos tendrán vínculos hacia diversos sumadores de módulo 2 según sea el criterio de codificación.



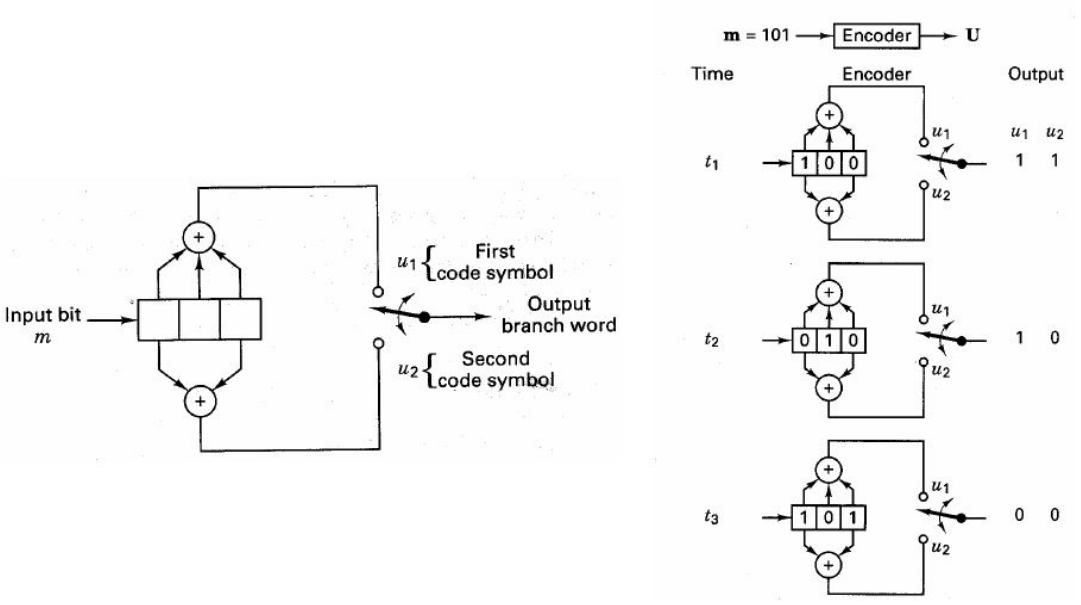
Una característica importante de estos codificadores a diferencia de los bloques lineales es que tienen memoria. La codeword “U_i” es función del mensaje digital “m_i” y de los K-1 anteriores. (Mensajes digitales anteriores presentes desde el segundo registro hasta el último)
La limitación de longitud, es decir, el valor de “K” representa el número de k-bits de corrimiento sobre los cuales un simple bit de información puede influenciar la salida del codificador. En cada unidad de tiempo, “k” bits son desplazados dentro de los primeros k-estados del registro; todos los bits en el registro son desplazados “k” estados a la derecha y las salidas de los “n” sumadores son muestreadas secuencialmente para producir los símbolos binarios del código o bits del código. Estos bits de código son luego usados por el modulador para especificar las formas de ondas a ser transmitidas sobre el canal. Dado que hay “n” bits de código para cada grupo de entrada de “k” bits de mensaje, la relación de código es k/n bits de mensaje por bit de código, donde “k”<”n”.

Entonces ¿Cómo podemos generar diversos codificadores convolucionales?
→ Cambiando las conexiones entre el registro de desplazamiento y los sumadores módulo 2.
→ Cambiando la cantidad de elementos del registro de desplazamiento.
→ Cambiando la cantidad de bits por clock que entran a los registros de desplazamiento y por lo tanto, la cantidad de bits que se desplazan.
Conclusión, modificando los parámetros característicos del codificador. Para entender qué impacto tienen dichas variables sobre la técnica de codificación se debe describir primero la función de codificación G(m), ya que, es la herramienta que permite caracterizar a un codificador convolucional.
Con todo esto queremos decir que para implementar un codificador convolucional deberemos tener 4 características fundamentales, siendo estas: “k”, “n”, “K” y G(m). Una vez que se tengan en cuenta estos aspectos, aparecen diversas formas de implementación para este tipo de

codificación como lo es la conexión pictorial, vectores de conexión o polinomiales, máquina o diagrama de estados, el diagrama de árbol y el diagrama de trellis.

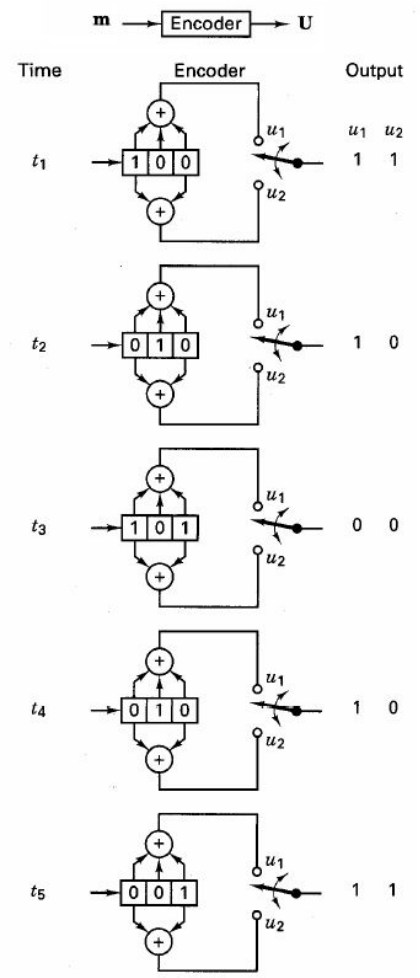
Diagrama pictorial

Una manera de representar el codificador es mediante un dibujo. Este permite obtener la codeword codificada a partir del mensaje digital, analizando la evolución del sistema a lo largo del tiempo. Para ello es necesario especificar sobre el dibujo un set de n vectores de conexión, uno por cada sumador módulo 2. Cada vector tiene dimensión K y describe la conexión del registro de corrimiento del codificador al sumador de módulo 2. Un uno en la i-ésima posición del vector indica que el correspondiente estado en el registro de corrimientos está conectado al sumador módulo 2 y un cero en la posición dada indica que no existe una conexión entre el estado y el sumador de módulo 2.
En el siguiente ejemplo se ilustra el concepto expuesto anteriormente:



Se puede observar en la figura izquierda como el sumador de módulo 2 puede ser representado como g1 = 1 1 1 y el inferior como g2= 1 0 1. Mientras que en el caso derecho vemos como actúa este sistema ante la presencia de un cierto mensaje digital como lo es el m=101, con respecto a los "g_x" implementados. Se observa cómo al haber pasado 3 tiempos, todo el mensaje ya se encuentra dentro del codificador obteniendo las salidas representadas por u_1 y u_2 . En esta situación se puede apreciar la influencia que genera cada dígito binario sobre la salida del codificador. En donde el primer bit influye en la salida durante tres tiempos de bit (tiempo que permanece dentro del codificador), el segundo influye dos tiempos y el último bit influye sobre la salida solamente sobre un tiempo. Es decir que cada bit de información asociado al mensaje digital no tiene el mismo peso en la codificación. Por eso, se busca agregar al final de cada mensaje digital una secuencia de bits (ceros) que permitan limpiar el registro y equilibren la influencia de cada bit del mensaje digital sobre la codeword de salida. La cantidad de ceros binarios debe ser "K" (Longitud del registro), pero estaríamos agregando más redundancia de la necesaria. Para ser eficientes en el uso de este concepto deberemos agregar K-1 ceros, siendo en este caso 2 ceros a los mensajes digitales para que el registro de desplazamiento por cada mensaje digital termine la codificación con el registro en todos ceros menos el último elemento.

De esta forma, cuando un nuevo mensaje digital ingrese al codificador, el primer bit desplazará todo el registro hacia la derecha descartando la influencia del último bit del mensaje digital anterior. De esta manera, las codeword no expresan la influencia entre mensajes digitales diferentes.
Siguiendo con el ejemplo anterior, la siguiente imagen ilustra la situación para un m=101 y otro m=110:



Si se desea transmitir: m=101 y luego m=110.
Caso 1, sin tener en cuenta influencia:

m = 101 110 → Encoder → U = 11 10 00 01 10 01 11 00

Se puede observar como el ultimo bit del primer mensaje influye sobre las primeras 2 codewords del proximo mensaje

Caso 2, considerando la influencia e incluyendo k-1 ceros:

m = 101 → Encoder → U =11 10 00 10 11

m = 110 → Encoder → U=11 01 01 11 00

m = 10100110 → Encoder → U = 11 10 00 10 11 11 01 01 11 00

Respuesta al Impulso

Trabajar con dibujos o diagramas pictóricos lleva su tiempo y es una forma poco eficiente de representar la tarea del codificador. Otra manera, un poco más sencilla está basada en la respuesta al impulso.
Para obtener la respuesta al impulso solo es necesario ingresar un uno binario e ir desplazándolo por todos los elementos del registro y obtener su codeword asociada.

Register contents	Branch word u_1	u_2
1 0 0	1	1
0 1 0	1	0
0 0 1	1	1

Input sequence: 1 0 0
Output sequence: 1 1 1 0 1 1

¿Cómo usamos este concepto para codificar fácilmente los diferentes mensajes digitales?

Input **m**	Output				
1	1 1	1 0	1 1		
0		0 0	0 0	0 0	
1			1 1	1 0	1 1
Modulo-2 sum:	1 1	1 0	0 0	1 0	1 1

Cuando se haga presente en la entrada del sistema un uno binario se generará la respuesta al impulso que durará 3 tiempos. Si al uno binario le sigue un cero, la respuesta del sistema tendrá la misma longitud que la respuesta al impulso pero serán todos ceros binarios y comenzarán a influenciar desde el segundo tiempo, porque el cero ingresó un instante de tiempo posterior. El último dígito binario presente en el mensaje digital influenciará la salida del sistema en los últimos 3 instantes de tiempo porque fue el último en ingresar como se ha visto en figuras anteriores. Finalmente para obtener la codeword asociada a dicho mensaje digital se hace suma en módulo 2 por cada tiempo de análisis.
En conclusión, para la secuencia de entrada deseada, la codeword se encontrará mediante la superposición o la adición lineal de los impulsos de entrada desplazados en el tiempo.

Representación Polinomial

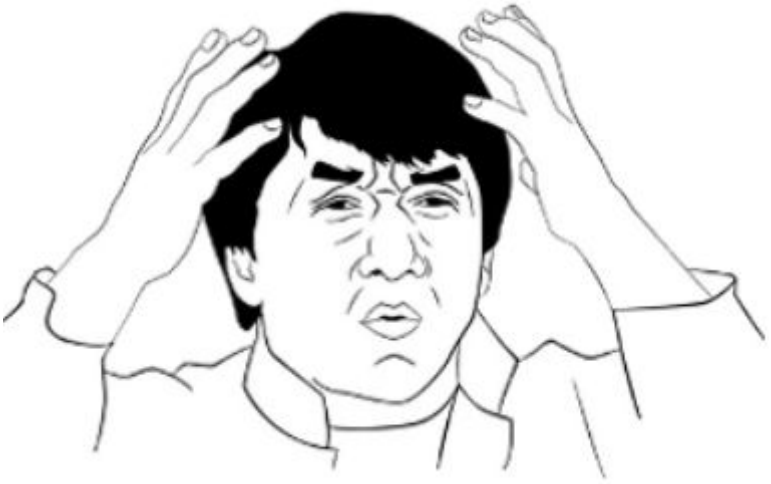
Sabemos que muchas operaciones realizadas por los codificadores pueden ser representadas fácilmente mediante polinomios. Se puede representar a un codificador convolucional con un set de “n” generadores polinomiales, uno para cada uno de los sumadores módulo 2. Cada polinomio es de grado K-1 o menor y describe la conexión del registro de desplazamiento del codificador al sumador módulo 2 del mismo modo que lo hace un vector de conexión como en el análisis mediante diagrama pictorial. La idea es que mediante estos polinomios y el mensaje digital (expresado como polinomio) se haga un producto tal que se obtenga por un lado “$g_1(x).m_D(x)$” y por otro “$g_2(x).m_D(x)$” para terminar encontrando la codeword asociada al mensaje digital mediante entrelazado. Es decir, la codeword U(x) es $g_1(x).m_D(x)$ entrelazado con $g_2(x).m_D(x)$.
Para el ejemplo con el que se está trabajando, la representación polinomial de las conexiones y del mensaje digital “101” es la siguiente:

$$g_1(X) = 1 + X + X^2$$
$$g_2(X) = 1 + X^2$$
$$m_D(x) = 1 + x^2$$

De esta manera, operando matemáticamente se obtiene la codeword resultante:

$$\mathbf{m}(X)\mathbf{g}_1(X) = (1 + X^2)(1 + X + X^2) = 1 + X + X^3 + X^4$$
$$\mathbf{m}(X)\mathbf{g}_2(X) = (1 + X^2)(1 + X^2) = 1 + X^4$$
$$\mathbf{m}(X)\mathbf{g}_1(X) = 1 + X + 0X^2 + X^3 + X^4$$
$$\mathbf{m}(X)\mathbf{g}_2(X) = 1 + 0X + 0X^2 + 0X^3 + X^4$$
$$\mathbf{U}(X) = (1,1) + (1,0)X + (0,0)X^2 + (1,0)X^3 + (1,1)X^4$$
$$\mathbf{U} = 1\ 1 \quad 1\ 0 \quad 0\ 0 \quad 1\ 0 \quad 1\ 1$$

No hay que olvidar que dichos polinomios responden a dígitos binarios por lo tanto la suma es en realidad una suma módulo 2 de los términos correspondientes. Como se observa en la imagen anterior, una vez que se obtienen $g_1(x).m_D(x)$ y $g_2(x).m_D(x)$ se ordenan los polinomios resultantes en orden ascendente y se completan con ceros los términos faltantes para entrelazar. Entrelazar significa agrupar términos de ambos polinomios que tienen igual orden.
Yo creo que siempre que hablamos de polinomios en estos capítulos todo el mundo se queda estilo



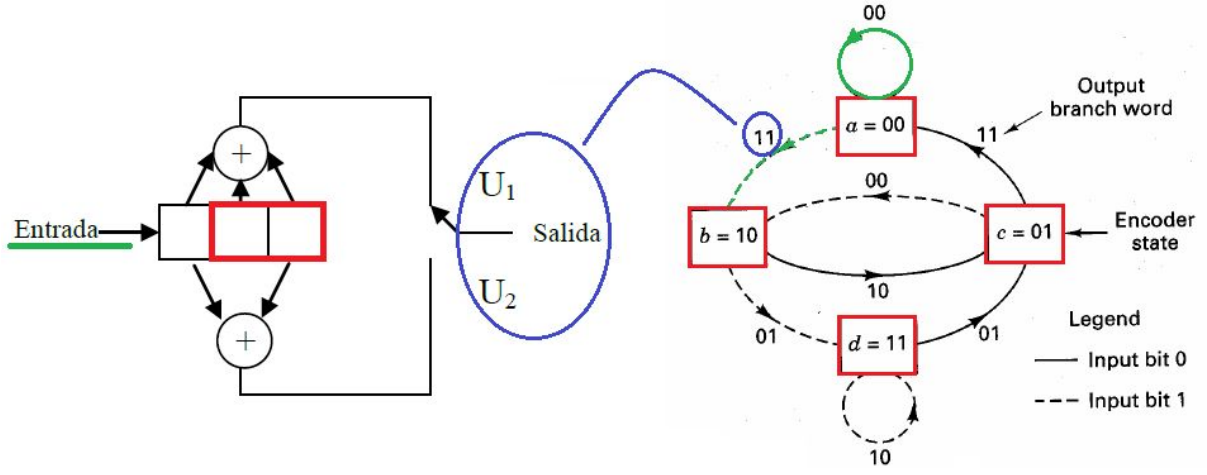
¿Por que siempre en los codificadores tanto cíclicos como en los convolucionales se habla de **polinomios**?
Esta demostrado que es una pinchila el laburo matemático de los polinomios. Siempre llevan asociado aplicar propiedades de distributiva re largas y aplicar propiedades de potencia. Entonces ¿Por qué? Porque lo importante de esta herramienta matemática es que dejan en evidencia la secuencia temporal. Es decir, cómo fueron sucediendo los hechos para que se conforme la codeword. En donde cada coeficiente del polinomio está hablando del tiempo. Siempre! Es la razón de la herramienta. En realidad somos tan putos telecos que si no usan la transformada zeta para señales discretas no nos damos cuenta. ¿Qué lo qué dice este tarado? Que siempre estamos acostumbrados a asociar secuencias discretas con el tiempo para las telecomunicaciones de la siguiente forma: $MD(z)= 1.z^0 + 0.z^{-1} + 1.z^{-2}$. El Sklar trabaja con polinomios de coeficientes positivos y con la variable “x” pero no se tiene que perder la idea.
En conclusión, en codificación convolucional es fundamental la influencia de los bits del mensaje digital **a lo largo del tiempo** en las codeword resultantes y en codificación cíclica es fundamental la rotación cíclica del mensaje, es decir, la manera en que se maneja la secuencia en el dominio del tiempo. Entonces los polinomios vienen al pelo para expresar de forma matemática estas situaciones.

Representación mediante diagrama de estado

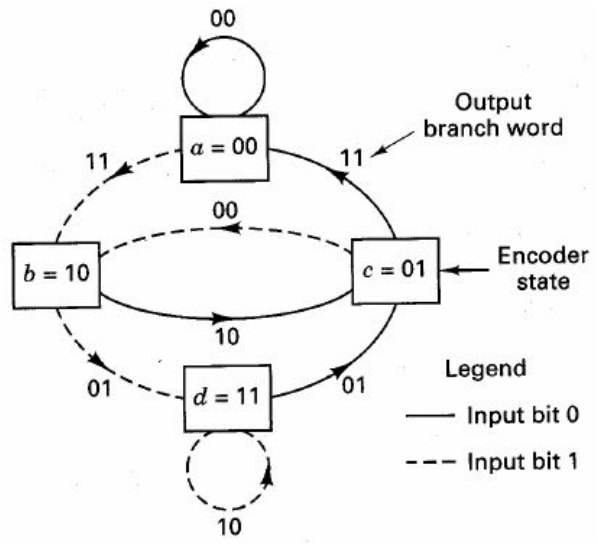
Los codificadores convolucionales pertenecen a una clase de aparatos conocidos como “máquina de estado finito”. El nombre en gran medida se debe a que hay solamente un número discreto y finito de estados posibles y únicos que la máquina puede adquirir a medida que transcurre el tiempo. Estos dispositivos tienen memoria de las señales pasadas, lo que significa que el estado futuro que asume la máquina dependerá del pasado, es decir de todos aquellos eventos anteriores que ocurrieron desde que comenzó a funcionar. Esto aplica a las codeword de salida, ¿pero a los registros de desplazamientos cómo se los puede modelar?
En este caso, estamos en presencia de una cadena de Markov. En la teoría de la probabilidad, se conoce como cadena de Márkov o modelo de Márkov a un tipo especial de proceso estocástico discreto en el que la probabilidad de que ocurra un evento depende solamente del evento inmediatamente anterior. Esta característica de falta de memoria recibe el nombre de propiedad de Markov. Lo que está ocurriendo es que el estado presente resume toda la información relevante (toda la historia del sistema) para describir en probabilidad su estado futuro. Para conocer el estado futuro solo necesitamos conocer el estado actual (combinación binaria que generan los k-1 registros) y la señal de entrada.
Desde el punto de vista probabilístico, podemos expresar la siguiente ecuación:

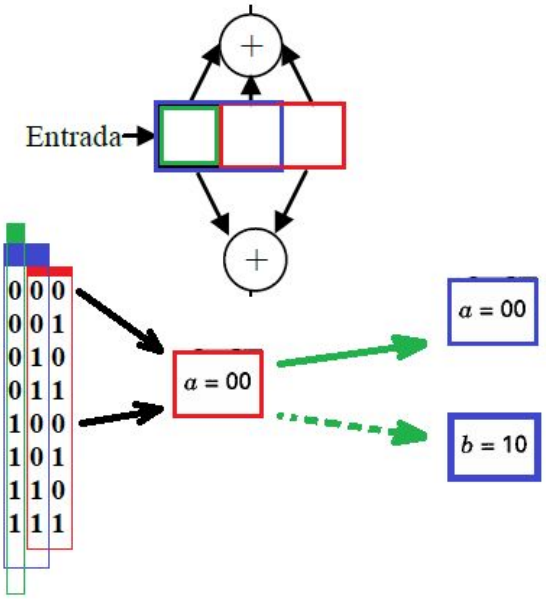
$$P(X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_2 = x_2, X_1 = x_1) = P(X_{n+1} = x_{n+1} | X_n = x_n).$$

En resumen, los registros de desplazamiento del codificador (estados) son Markov y pueden representarse mediante un diagrama de estado.



El funcionamiento y conformación del diagrama de estado es más sencillo de observar mediante la implementación de un ejemplo, para ello primero armaremos paso a paso el diagrama ya presentado y luego veremos cómo implementarlo para un mensaje de entrada m=11011 seguido por 2 ceros para vaciar los registros y dar igual influencia a los m bits.





Input bit m_i	Register contents	State at time t_i	State at time t_{i+1}	Branch word at time t_i u_1	u_2
—	0 0 0	0 0	0 0	—	
1	1 0 0	0 0	1 0	1	1
1	1 1 0	1 0	1 1	0	1
0	0 1 1	1 1	0 1	0	1
1	1 0 1	0 1	1 0	0	1
1	1 1 0	1 0	1 1	0	1
0	0 1 1	1 1	0 1	0	1
0	0 0 1	0 1	0 0	1	1

state t_i
state t_{i+1}

Output sequence: **U** = 1 1 0 1 0 1 0 0 0 1 0 1 1 1

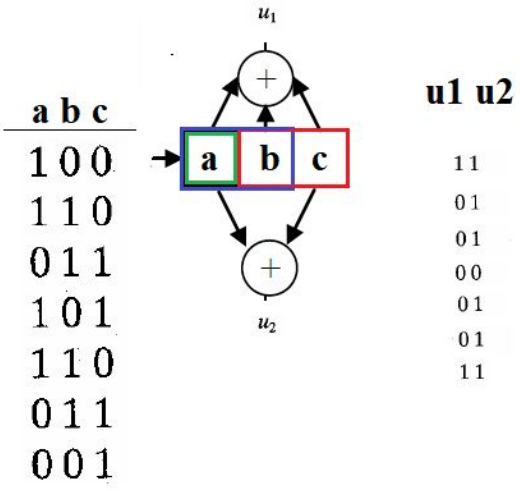
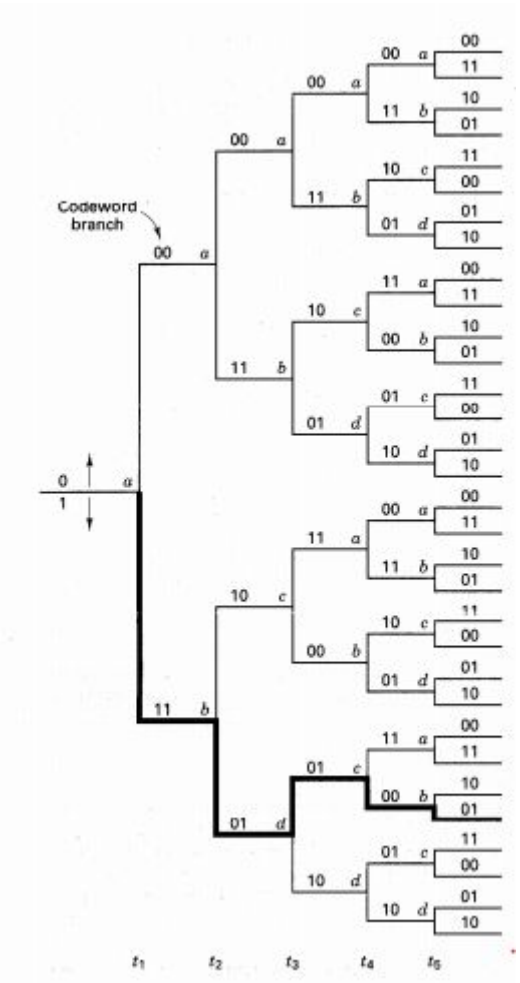


Diagrama de árbol

Aunque el diagrama de estado caracteriza completamente al codificador, no se lo puede utilizar fácilmente para rastrear las transmisiones del codificador como una función del tiempo, porque el diagrama de estado no representa la historia del tiempo, ya que no da ninguna referencia temporal. El diagrama de árbol agrega la dimensión del tiempo al diagrama de estado y permite describir dinámicamente al codificador como una función de una secuencia particular de entrada. Sin embargo, cuando los mensajes digitales son demasiados largos necesitaremos varias hojas A4 y mucha paciencia porque el diagrama crece rápidamente.

Para el mensaje digital 101 y el codificador convolucional de referencia, tenemos el siguiente diagrama:



La regla para encontrar la codeword resultante es: Si el bit de entrada es un cero, la palabra asociada a la rama se encuentra moviéndose a la próxima rama de más a la derecha, en dirección ascendente. Si la entrada es un uno, la palabra asociada a la rama se encuentra moviéndose a la próxima rama de más a la derecha, en dirección descendente.

Para construir el árbol, siempre se asume el estado inicial de todos ceros y utilizando el diagrama de estado se construyen las bifurcaciones que contienen los estados futuros dependiendo del estado actual y la entrada binaria. Una vez especificadas las bifurcaciones, se agrega a cada rama correspondiente la salida del codificador.

Es importante mencionar que a partir del tiempo t_4,o sea en K+1, el resultado de las ramas y las bifurcaciones se repiten. ¿Dónde? Es necesario dividir el diagrama de árbol en dos a partir de la primera bifurcación. Luego, analizar ambos diagramas a partir del tiempo inmediato posterior al valor de “K” asociado a los registros de desplazamiento. Esto demuestra que es fácilmente replicable pero requiere muchísimo espacio y tiempo para codificar mensajes largos.

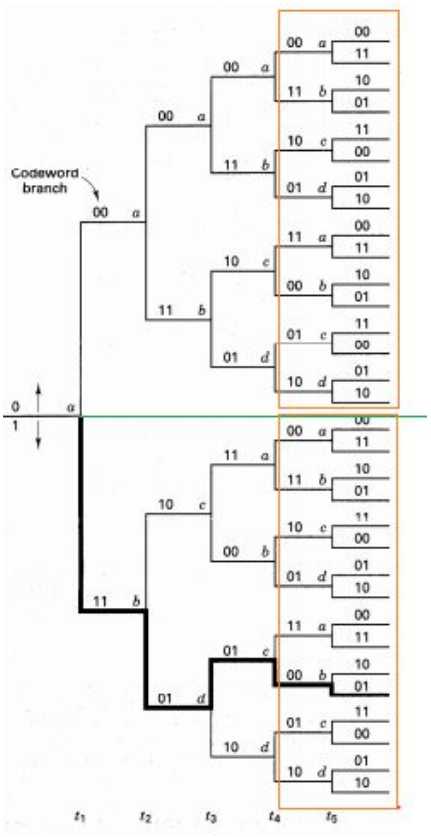
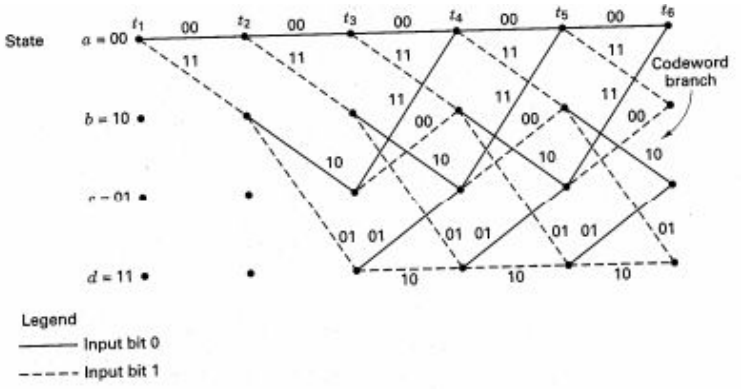


Diagrama de Trellis

Dada la repetición que presenta el diagrama de árbol, en los tiempos t_i posteriores a “K” se encontrarán siempre al menos dos nodos que poseen los mismos estados, entonces podrían unirse, ya que todos los caminos posteriores serían indistinguibles.
Si efectivamente se realiza dicha operación sobre el diagrama de árbol se obtiene un nuevo diagrama denominado diagrama de Trellis. Este aprovecha la estructura repetitiva del diagrama de árbol para presentar una nueva forma de caracterizar al codificador como una función temporal más manejable.
Es importante mencionar que los nodos de trellis representan los estados del registro de desplazamiento. Para cada unidad de tiempo, se requiere 2^{k-1} nodos para representar los 2^{k-1} posibles estados del codificador. En líneas generales, la estructura fija prevalece después de que “K” es alcanzado. En este punto y después de esto, cada uno de los estados puede pasar a uno de dos estados. De las dos ramas salientes, uno corresponde a un bit de entrada cero y el otro corresponde a un bit de entrada uno.



Lo importante de Trellis es la simplicidad con la que puedo codificar. Solamente es necesario especificar los bits del mensaje digital y desde el estado “a” o lo que es lo mismo decir “00” construir el camino correspondiente. **Cada codeword válida esta definida por un camino en el diagrama de Trellis**. En cada intervalo de tiempo, las secciones de los caminos permiten visualizar la salida del codificador, por lo tanto se puede observar la evolución temporal de la secuencia de salida correspondiente a la codeword.
Sin embargo este concepto no fue explotado al máximo hasta que apareció Viterbi. El cual se dio cuenta que con esta herramienta y una métrica de distancia inventada por él, se podía resolver el problema clásico de detección para sistemas de comunicación digital. Se sabe que la detección consiste en establecer cuál es la señal que se ha querido transmitir en un determinado instante de tiempo en función de adquirir una muestra de la señal recibida, denominada Z. Esto matemáticamente implica resolver una ecuación de probabilidad. Dado que tenemos una muestra de la señal recibida Z, cuál es la probabilidad de que se haya querido transmitir S_1 , $S_{2,}$ ó S_m. En definitiva hay que resolver la siguiente ecuación:

$$P\left(S_1/Z\right) \underset{>}{<} P\left(S_2/Z\right)$$

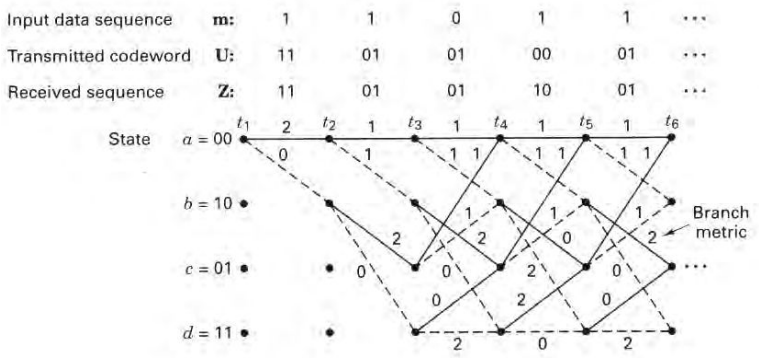
Ecuación de probabilidades condicionales a posteriori que Viterbi simplificó mediante teorema simple de Bayes porque las probabilidades conjuntas son nulas. Esto se debe a que el ruido afecta de forma independiente a cada símbolo transmitido por el canal. Lo que hizo Viterbi es resolver esta ecuacion:

$$p(z/s_1) \underset{\geq}{\leq} p(z/s_2)$$

Esto implica comparar el Z recibido con cada una de las codeword válidas. Es decir, computar similitudes y diferencias entre cada uno de los caminos en el diagrama de Trellis con la secuencia recibida. Aquella codeword que acuse menor cantidad de diferencias con la secuencia Z constituye la codeword más probable de ser transmitida. A este concepto, Viterbi le dio el nombre de “máxima probabilidad” y lo formalizó mediante la siguiente ecuación:

$$P\left(Z\middle|U^{(m')}\right) = \max P\left(Z\middle|U^{(m)}\right)$$

sobre todas las $U^{(m)}$



La imagen anterior dice que para el tiempo t_6 se deberían armar todos los caminos posibles que resultan hasta dicho instante de tiempo (codeword válidas) y calcular la probabilidad

$P\left(Z/U^{(m)}\right)$. Habría que realizar alrededor de 2^6 operaciones de probabilidad ya que por cada intervalo de tiempo, los nodos se bifurcan en dos, aportando el doble de caminos posibles para llegar al mismo punto.

En este sentido, Viterbi se dio cuenta de algo fundamental. En lugar de analizar todas las codewords válidas en el instante de tiempo t_6, planteó analizar las codeword posibles por ramas dentro del diagrama de trellis. Esto lo llevó al desarrollo de la siguiente ecuación:

$$p\left(Z/U^{(m)}\right)=\prod_{i=1}^{\infty}p\left(Z_i/U_i^{(m)}\right)=\prod_{i=1}^{\infty}\prod_{j=1}^{n}p\left(Z_{ji}/U_{ji}^{(m)}\right)$$

donde:

Z_i = i-ecima rama de la secuencia recibida Z

$U_i^{(m)}$ = i-ecima rama de una secuencia de codeword en particular

Z_{ji} = j-ecimo codigo de un simbolo Z_i

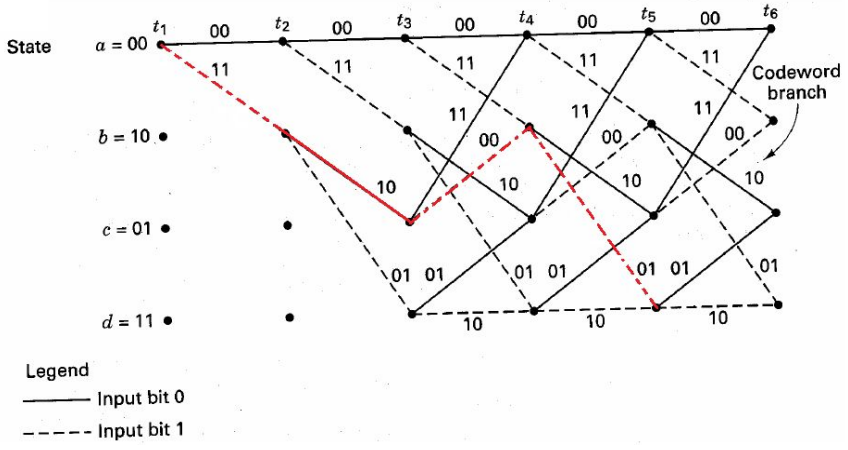
$U_{ji}^{(m)}$ = j-ecimo codigo de un simbolo U_i

¿Cómo se lee la ecuación anterior de probabilidad?

Para responder dicha pregunta necesitamos formular la probabilidad de transmitir una codeword válida desde el punto de vista del codificador. Esta probabilidad para un ejemplo particular puede pensarse como:

La codeword U se obtiene con las siguientes probabilidades:

$$P\left(\frac{Estado\ b}{Estado\ a}\right)$$
$$y$$
$$P\left(\frac{Estado\ c}{Estado\ b}\right)$$
$$y$$
$$P\left(\frac{Estado\ b}{Estado\ c}\right)$$
$$y$$
$$P\left(\frac{Estado\ d}{Estado\ b}\right)$$



$$P\left(\frac{Estado\ b}{Estado\ a}\right) -> Probabilidad\ de\ estar\ en\ el\ estado\ b\ en\ t_2\ dado\ que\ estaba\ en\ el\ a\ para\ t_1$$

y

$$P\left(\frac{Estado\ c}{Estado\ b}\right) -> Probabilidad\ de\ estar\ en\ el\ estado\ c\ en\ t_3\ dado\ que\ estaba\ en\ el\ b\ para\ t_2$$

y

$$P\left(\frac{Estado\ b}{Estado\ c}\right) -> Probabilidad\ de\ estar\ en\ el\ estado\ b\ en\ t_4\ dado\ que\ estaba\ en\ el\ c\ para\ t_3$$

y

$$P\left(\frac{Estado\ d}{Estado\ b}\right) -> Probabildiad\ de\ estar\ en\ el\ estado\ d\ en\ t_5\ dado\ que\ estaba\ en\ el\ estado\ b\ en\ t_4$$

Con esta lógica y conociendo que los eventos son independientes, la intersección (representada por los “y” que unen a las probabilidades condicionales) resulta en un producto. De ahí es la productoria que se observa en el decodificador. Sin embargo en el decodificador el procedimiento consiste en todo momento en plantear: “Dado el Z recibido, cuál es la probabilidad de que haya seleccionado este camino (rama) para el momento actual”. Lo que hace el decodificador es analizar cada uno de los caminos por segmentos. De ahí surge la doble productoria, porque el decodificador debe realizar una comparación entre la secuencia binaria recibida Z y todos los posibles caminos y para cada caso particular, analizar sección a sección las ramas de dicho camino.

Para que se entienda mejor, apliquemos el ejemplo anterior:

El decodificador plantea para dicha codeword

$$P\left(\frac{Z}{Estado\ A->B}\right)_{t_1->t_2} \quad -> \textit{Cual es la probabilidad de que el } z \textit{ recibido corresponda a la rama del camino asociado al cambio de estado de } A \textit{ a } B$$

y

$$P\left(\frac{Z}{Estado\ B->C}\right)_{t_2->t_3} \quad ->-> \textit{Cual es la probabilidad de que el } z \textit{ recibido corresponda a la rama del camino asociado al cambio de estado de } B \textit{ a } C$$

y

...

y

...

En conclusión, se busca maximizar las probabilidades para las diferentes secciones.

$$\prod_{i=1}^{\infty}\prod_{j=1}^{n} p\left(Z_{ji}/U_{ji}^{(m)}\right)$$

Viterbi considero en su análisis que es más conveniente desde el punto de vista operativo usar el logaritmo de la función de probabilidad, ya que permite aplicar sumas de términos en lugar de productos. Se puede utilizar dicha transformación ya que el logaritmo es monotónicamente creciente y por lo tanto no alterará el resultado final en la selección de la codeword.

$$\gamma_U(m) = \log P\left(Z\middle|U^{(m)}\right) = \sum_{i=1}^{\infty}\log P\left(Z_i\middle|U_i^{(m)}\right) = \sum_{i=1}^{\infty}\sum_{j=1}^{n}\log P\left(z_{ji}\middle|u_{ji}^{(m)}\right)$$

Entonces, el problema de detección se reduce a encontrar aquel camino a través del diagrama de árbol o el diagrama de trellis, tal que $\gamma_U(m)$ sea máxima.

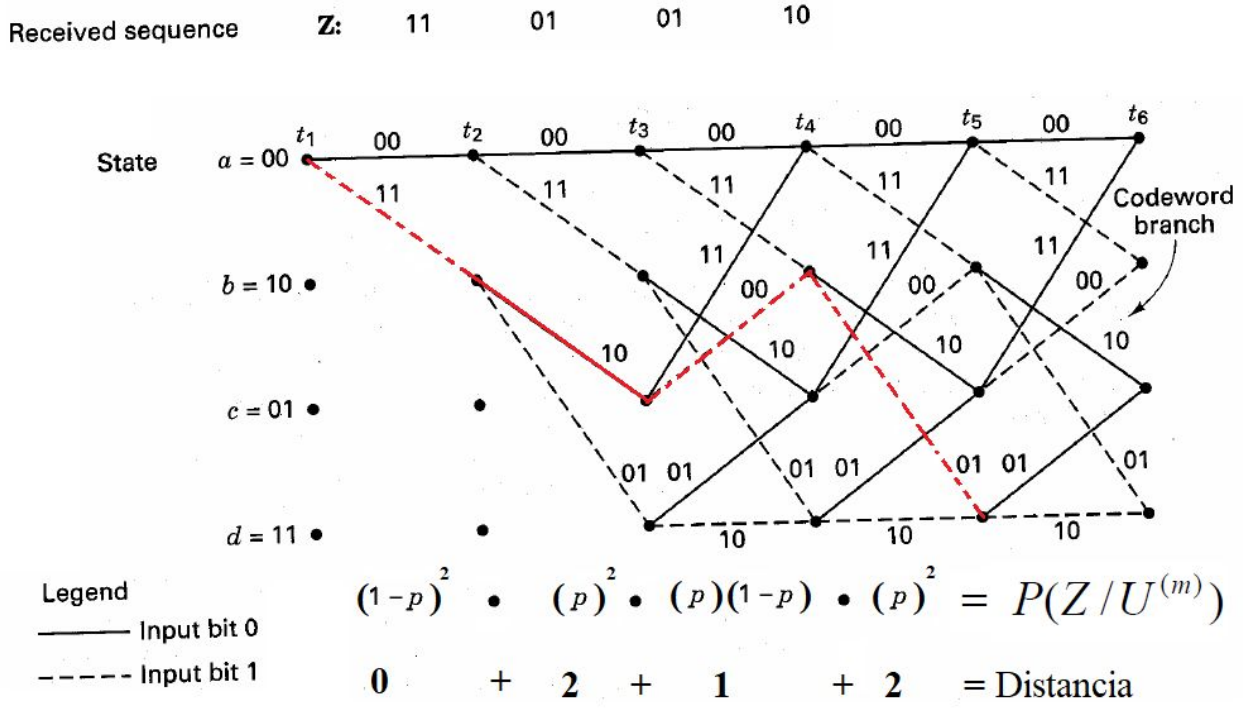
Todo esto nos habla de que un decodificador de máxima probabilidad escoge por el codeword $U^{(m')}$ que maximice la probabilidad $P(Z/U^{(m)})$ o su logaritmo. Para el caso del canal simétrico binario esto es equivalente a escoger el codeword $U^{(m')}$ que esté más cerca en distancia de Hamming a Z. De esta manera, la distancia de Hamming es una métrica para describir la distancia o la proximidad apropiada entre $U^{(m)}$ y Z. De todas las posibles secuencias $U^{(m)}$ transmitidas, el decodificador escoge la secuencia $U^{(m')}$ con la cual la distancia a Z es mínima.

Se supone que U(m) y Z son todas secuencias de longitud de “L” bit y que ellas difieren en las posiciones “d_m” [es decir, las distancias Hamming entre U(m) y Z es d_m]

$$P(Z/U^{(m)}) = p^{d_m}(1-p)^{L-d_m}$$

$$\log P(Z/U^{(m)}) = -d_m \log\left(\frac{1-p}{p}\right) + L\log(1-p)$$

Por lo tanto, para un BSC, queda demostrado que el logaritmo de la probabilidad es convenientemente reemplazada por la distancia de Hamming, y un decodificador de máxima probabilidad escogerá, en el diagrama de árbol o diagrama de trellis, el camino cuya secuencia correspondiente U(m’) está a la mínima distancia de Hamming de la secuencia recibida Z.
Un ejemplo de aplicación de este concepto es:



El algoritmo de Viterbi, aplica estos conceptos (Máxima probabilidad) y de forma eficiente. La ventaja de esta decodificación a comparación con la decodificación de fuerza bruta es que no influye el número de símbolos en la secuencia de codeword. El algoritmo involucra el cálculo de una medida de similitud o distancia, entre el símbolo recibido en el tiempo t_i y todos los caminos de Trellis evaluados en el intervalo t_i. Viterbi quita de las consideraciones aquellos caminos en el diagrama de trellis que posiblemente no podrían ser candidatos para la opción de máxima probabilidad. Cuando dos caminos entran a un mismo estado en un determinado tiempo t_i, el que tiene mejor métrica (menor distancia de hamming hasta ese punto) es seleccionado. Este camino es definido como camino superviviente. La selección de camino superviviente es realizada para todos los estados donde se encuentran dos caminos posibles. El decodificador continua desde este punto con los caminos de mejor métrica obteniendo ventajas sobre el diagrama de trellis a medida que transcurre el tiempo. El mismo va tomando decisiones para

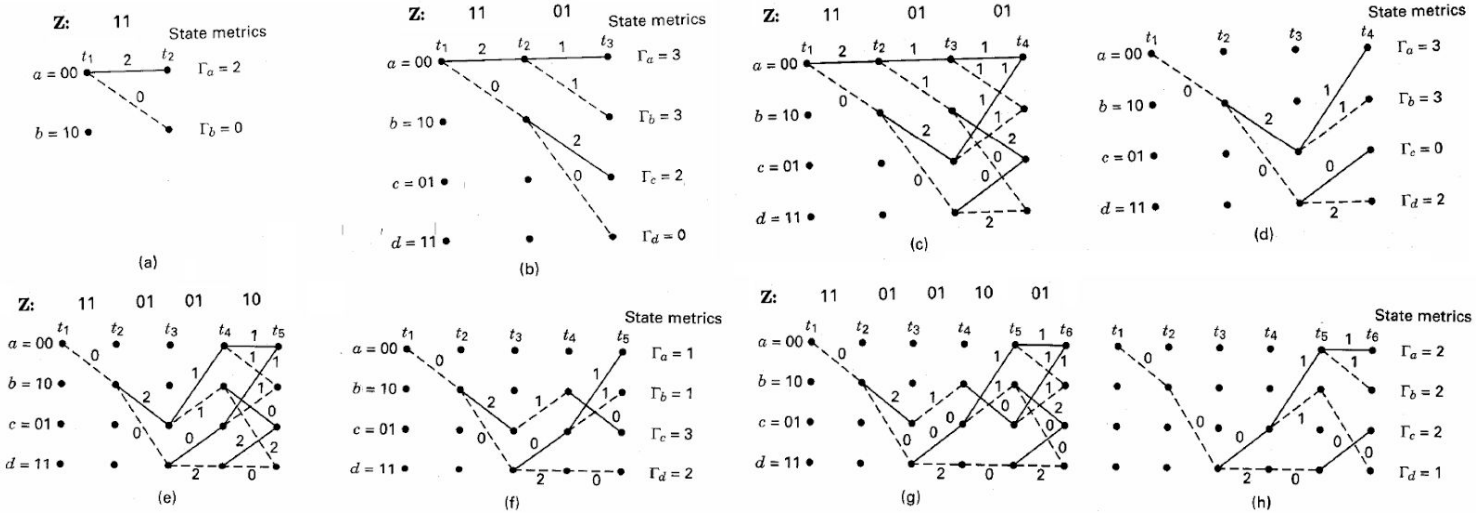
eliminar tempranamente los caminos menos probables. Esto implica una reducción en la complejidad de la decodificación.

Implementando en un caso práctico el algoritmo de Viterbi como se ve en la siguiente figura, partimos desde un estado definido, el cual suele ser comúnmente el a=00. Al recibir una codeword Z, conformada por 2 bits, tendremos dos posibles caminos o ramas debido a la consideración de canal binario simétrico. A estas ramas se les asocia el peso de hamming que se asignará en relación entre el Z recibido y la rama a designar, habiendo únicamente 3 valores posibles (0, 1 y 2) ya que estamos tratando con sistemas binarios y de 2 bits. Para el caso “a)” tenemos que la codeword recibida Z_1 es 11 y los pesos asignados son 2 y 0.

En la segunda recepción, caso “b)”, siendo Z_2 = 01 asignaremos pesos a 4 ramas ya que ahora hay 2 posibles puntos de partida desde t_2, y ambos con 2 caminos posibles. Pero como todavía ninguna de esas ramas coinciden a un mismo punto de destino, veremos que sucede con la codeword z_3.

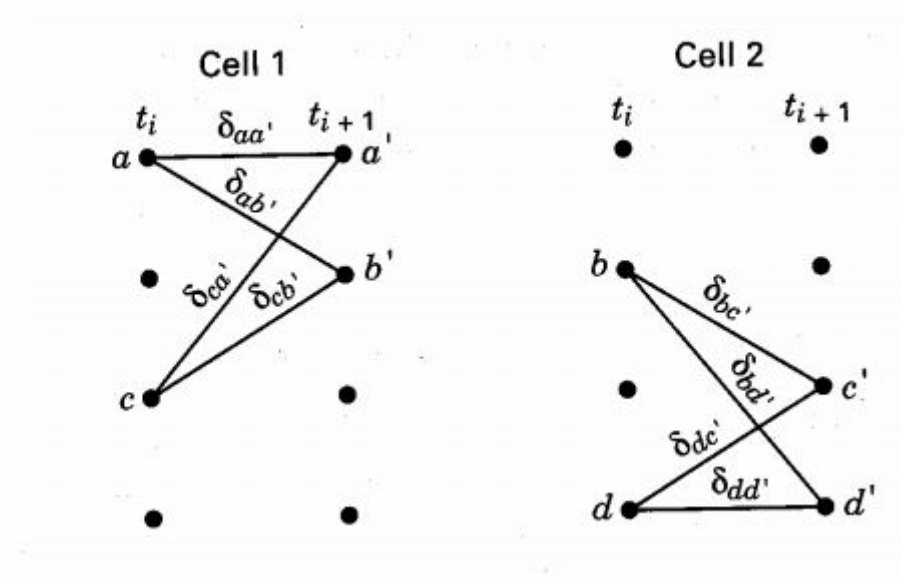
En “c)” con Z_3 = 01 tendremos 8 ramas, que irán de t_3 a t_4 siendo el máximo de ramas posibles para un diagrama que trabaja con codewords de 2 bits. Se asignan los pesos correspondientes pero ahora sucede que las ramas llegan de a pares a t_4 debido a que contamos con solo 4 posibles estados, es acá en donde debemos de guiarnos según las distancias de Hamming acumuladas y optar por las ramas que tienen menor valor de entre las 2 que están en análisis por cada punto, una vez realizado este procedimiento, el diagrama se simplificará a solo 4 posibles vías como se observa en “d” y además, al haber una única rama entre t_1 y t_2 podremos decir el bit que se ha recibido para ese tiempo.

Para “e)” con Z_4 = 10 se realizará nuevamente el procedimiento realizado en “c” y en t_5 haremos una nueva cancelación de ramas como se ve en “f)”. Este procedimiento se realizará sucesivamente a medida de que vayamos recibiendo las distintas codewords a decodificar o hasta que ya se hayan recibido todas las codewords esperadas y decidir por la vía de menor distancia de Hamming acumulada.

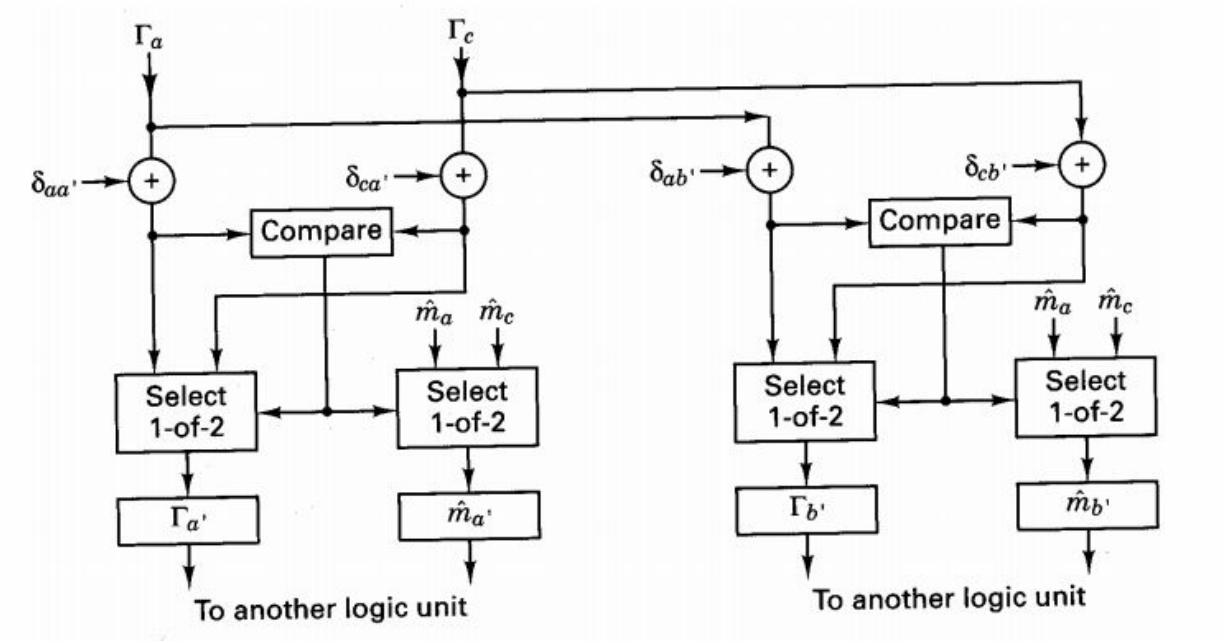


Aplicación del decodificador de Viterbi

Las transiciones entre dos instantes de tiempo consecutivos cualesquiera en el diagrama de Trellis pueden agruparse en 2^{v-1} celdas disjuntas. Para cada celda se grafican cuatro posibles transiciones donde v=K-1. Esta variable se denomina memoria del codificador. Para el ejemplo que se trabaja siempre K= 3, v= 2 y 2^{v-1}= 2 celdas. Estas celdas se muestran en la figura presentada a continuación, donde a,b,c y d se refieren a los estados en el tiempo t_i y a',b',c' y d' se refieren a los estados en el tiempo t_{i+1}. En cada transición se muestra la métrica de la rama $\boldsymbol{\delta}_{xy}$ donde los subíndices responden a la transición del estado "x" al estado "y" en los tiempos comprendidos entre t_i y t_{i+1}.



A continuación se presenta una unidad lógica para resolver la celda 1.



En este diagrama vemos que la métrica de estado "a" en el tiempo t_{i+1},**(□$_{a'}$)** es calculada sumando la métrica del estado previo al estado "a", **(□$_a$)** a la métrica de rama $\boldsymbol{\delta}_{aa'}$. También, es

factible considerar que para obtener la métrica de estado “a” en el tiempo t_{i+1} **(□$_{a`}$)**, se debe sumar la métrica de estado previo al estado “c” **(□$_c$)** a la métrica de rama $\boldsymbol{\delta}_{ca`}$.
Esto da como resultado dos posibles caminos a partir de los cálculos realizados previamente. Tenemos dos métricas candidatas para **(□$_{a`}$).** Ambas son comparadas en el diagrama de la figura anterior y se selecciona la más probable. Aquella métrica más probable (de menor distancia) es almacenada como nueva métrica del estado “a” en el tiempo $t_{i+1}$ , **(□$_{a`}$).** Es también almacenada la nueva historia de camino $m_{a`}$ para el estado “a” donde $m_{a`}$ es el mensaje con la historia del camino que corresponde al estado aumentado por los datos del camino ganador.
Se opera de la misma manera sobre la celda uno para obtener la métrica del estado “b” en el tiempo t_{i+1} **(□$_{b`}$).**
Ejecutamos la operación suma-comparación-selección ACS cuando hay dos transiciones arribando a un estado.
Se verifican los conceptos planteados anteriormente mediante el siguiente ejemplo.