Codici convoluzionali

Un codice convoluzionale (n, k, N) è un codice lineare a memoria finita $m = k(N-1)$, che genera n cifre binarie per k cifre di messaggio ($R_C = \frac{k}{n}$) tenendo conto delle ultime N (lunghezza di vincolo) parole di input (da k bit). N è il numero di volte in cui ciascun bit di input contribuisce a un codeframe.

Si possono implementare con una finestra scorrevole di kN bit sull'input, collegata a sommatori con un output da n bit, e che avanza di un dataframe (k bit) alla volta.

Un codice convoluzionale può essere specificato tramite N generatrici $n \times k$: $g_{ij}^{(l)} = 1$ se il bit j del dataframe l nella finestra scorrevole contribuisce a determinare l'i-esimo bit del codeframe corrente. Quindi:

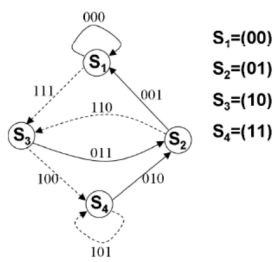
$$x_i^{(t)} = \sum_{l=1}^{N} \sum_{j=1}^{k} g_{ij}^{(l)} u_j^{(t-N+l)} \qquad i \in \{1, \ldots, n\}$$

Rappresentazioni

Gli esempi sono con $k = 1$, quindi $m = N - 1$ (in generale $k(N-1)$). Gli archi continui sono associati alla lettura di 0, quelli tratteggiati di 1.

Diagramma a stati

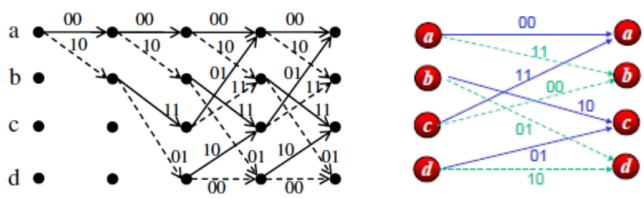
2^m nodi, ciascuno associato a m bit corrispondenti alle $N-1$ parole lette in precedenza, e con transizioni basate sui k bit correnti. Ogni nodo ha 2^k archi uscenti, ciascuno associato a n bit di output (stile Mealy).



Il nuovo stato è la dataword letta seguita dallo stato precedente senza l'ultima dataword (finestra scorrevole).

Diagramma a traliccio (trellis)

Traccia anche l'indice temporale t; per ogni t contiene i 2^m stati, collegati opportunamente agli stati all'istante di tempo successivo.



La prima versione segue il processo di codifica (si inizia sempre dallo stato con i registri nulli), la seconda è una forma compatta.

Albero