1. Sistemas de numeración

Un **sistema de numeración** es un conjunto de símbolos empleados para representar información numérica.

La **base** m de un sistema de numeración es la cantidad de símbolos distintos que utiliza.

En un **sistema de numeración posicional**, el valor de un dígito depende de su posición dentro del número. La posición de cada dígito tiene asignado un **peso**. Para los números enteros, se asignan las potencias positivas de la base m que aumentan de derecha a izquierda. Para los números fraccionarios, se asignan las potencias negativas de la base m que decrecen de izquierda a derecha.

El **sistema decimal** es un sistema posicional de base 10.

El **sistema binario** es un sistema posicional de base 2. Con n bits se puede contar hasta un número igual a $2^n - 1$.

El **sistema hexadecimal** es un sistema posicional de base 16. Un dígito decimal equivale a un número binario de 4 dígitos.

El bit más a la izquierda de un número binario con signo es el **bit de signo**. Los números binarios con signo se pueden representar con:

- **Formato magnitud y signo:** un número negativo tiene los mismos bits de magnitud que el correspondiente número positivo, pero el bit de signo es un 1 en lugar de un 0. Tiene doble representación del 0.
- **Complemento a 1:** un número negativo es el complemento a 1 del correspondiente número positivo. Tiene doble representación del 0.
- **Complemento a 2:** un número negativo es el complemento a 2 del correspondiente número positivo. No tiene doble representación del 0.

2. Códigos

Un **código** es un sistema de reglas para convertir información de una forma o representación a otra con propiedades deseadas para su transmisión, almacenamiento y procesamiento.

Se utilizan códigos para facilitar la comunicación en lugares donde el lenguaje oral o escrito es difícil o imposible de utilizar.

Formalmente, un código C es una función $C : \mathcal{S} \to \mathcal{T}$ que asigna a toda palabra $s_i = \{s_1, ..., s_n\}$ perteneciente al alfabeto fuente $\mathcal{S}$, una palabra $t_i = C(s_i) = \{t_1, ..., t_m\}$ perteneciente al alfabeto objetivo $\mathcal{T}$.

La **longitud** de una palabra código es la cantidad de símbolos que la componen.

La **longitud esperada** de un código es un promedio ponderado de la longitud de palabra de cada símbolo fuente codificado y su probabilidad de aparecer.

En un código de **longitud fija**, un número fijo de símbolos de entrada es codificado en un número fijo de símbolos de salida.

En un código de **longitud variable**, un número fijo de símbolos de entrada es codificado en un número variable de símbolos de salida.

La **distancia** es una función que mide la similitud entre dos palabras código y cumple con las siguientes propiedades: $d(a, b) = d(b, a)$, $d(a, a) = 0$, $d(a, b) \geq 0$.

La **distancia de Hamming** es el número de posiciones en las que los símbolos de dos palabras son diferentes.

Un **código continuo** es un código en el que la distancia de Hamming entre las palabras código sucesivas es 1.

Un **código cíclico** es un código en el que, al aplicar una rotación a una palabra código, se obtiene otra palabra que pertenece al código.

Un **código completo** es un código que utiliza todas las palabras código disponibles y no tiene redundancia.

Si $C : \mathcal{S} \to \mathcal{T}$ es inyectiva ($s_i \neq s_j \Rightarrow t_i \neq t_j$), entonces el código C es **invertible** y **unívocamente decodificable**.

Un **código prefijo** es un código de longitud variable donde ninguna palabra código es prefijo de cualquier otra palabra código.

Un **código instantáneo** es un código que se puede decodificar de forma instantánea palabra a palabra.

Un **código óptimo** es un código unívocamente decodificable cuya longitud media es mínima comparada a los demás códigos sobre los mismos alfabetos fuente y objetivo.

Dada una fuente de n símbolos a codificar con un alfabeto de r símbolos utilizando un conjunto de n palabras de longitudes l_1 a l_n. La **desigualdad de Kraft** corresponde a

$$\sum_{i=1}^{n} \left(\frac{1}{r}\right)^{l_i} \leq 1$$

- Es condición necesaria para que un código sea **prefijo**.
- Es condición suficiente para que exista algún código prefijo con la secuencia de longitudes l_1 a l_n.
- Es condición necesaria para que un código sea **unívocamente decodificable**.

La **decodificación** es el proceso inverso que convierte un código a su forma de origen para ser utilizado y comprendido.

Los códigos se utilizan para **comprimir datos** y transmitirlos de forma más eficiente, **controlar errores** agregando redundancia, **proteger** información privada de terceros.

2.1. Detección y corrección de errores

La transmisión y almacenamiento de datos se realizan a través de canales físicos que pueden causar alteraciones. Las técnicas de **detección y corrección de errores** incluyen información redundante para poder detectar y corregir los errores.

Los esquemas de detección y corrección pueden ser:

- **No sistemático:** el mensaje original se transforma en un mensaje codificado que lleva la misma información y que tiene al menos la misma longitud del mensaje original.
- **Sistemático:** al mensaje original se le adjunta información de verificación que se deriva de los datos mediante algún algoritmo.

Los modelos más comunes son:

- **Modelos sin memoria:** cuando los errores ocurren al azar.
- **Modelos dinámicos:** cuando los errores ocurren en ráfagas.

Existen tres tipos principales de corrección de errores:

- **Solicitud de repetición automática (ARQ):** el receptor tiene un tiempo para confirmar si recibió el mensaje. Si se agota el tiempo de espera, el mensaje se reenvía hasta que sea recibido correctamente.
- **Adelanto de información (FEC):** se agregan datos redundantes para recuperar la información cuando hay errores.
- **Esquemas híbridos:** el receptor verifica si el mensaje tuvo errores con FEC, y solicita un reenvío con ARQ de ser necesario.

2.2. Códigos de detección y correción de errores

Los **códigos convolucionales** son códigos en los que cada símbolo de palabra código es una suma ponderada de los símbolos del mensaje de entrada. Se decodifican con **algoritmos de decisión suave**. Los **códigos de bloque lineales** procesan la información en bloques de longitud fija. Se decodifican con **algoritmos de decisión dura**.

Existen muchos tipos de códigos de bloque lineales:

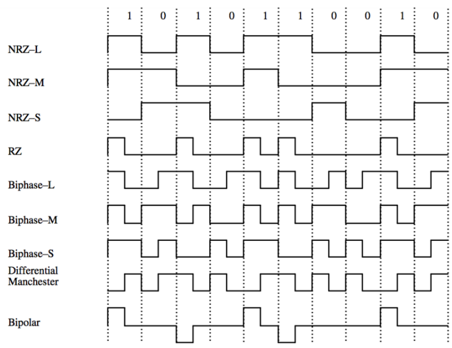
- **Códigos de repetición:** repiten el mensaje múltiples veces.
- **Códigos cíclicos:** rotaciones de las palabras código dan otra palabra código.
- **Códigos polinomiales:** las palabras código válidas son los polinomios divisibles por un polinomio fijo.

La **detección de errores** se realiza utilizando una **función *hash***.

- **Bit de paridad:** se agrega un bit al mensaje para garantizar que el número de bits con valor 1 en el resultado sea par o impar.

Los **códigos de Hamming** son códigos capaces de detectar errores de hasta dos bits o corregir errores de un bit.

2.3. Códigos de línea



Un **código de línea** es un patrón de voltaje, corriente, magnetismo o fotones utilizado para representar datos digitales a ser transmitidos o almacenados.

- Reducen la complejidad de procesamiento,
- Aumentan la tolerancia al ruido e interferencia.
- Facilitan la detección y corrección de errores.
- Facilitan la sincronización.

Los canales físicos más utilizados son las líneas eléctricas, las ondas inalámbricas, las señales ópticas, la impresión en papel o los campos magnéticos.

La **disparidad** de un patrón de bits es la diferencia en el número de unos frente al número de ceros. Se puede eliminar con:

- **Códigos de peso constante:** las palabras código contienen niveles opuestos, de modo que el nivel promedio sobre cada palabra es 0.
- **Códigos de disparidad emparejados**: cada palabra que promedia un nivel negativo se empareja con otra que promedia un nivel positivo.

2.4. Códigos digitales

El **código BCD** expresa cada uno de los dígitos decimales con un código binario.

Dígito decimal	0	1	2	3	4	5	6	7	8	9
BCD 8421	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
2421	0000	0001	0010	0011	0100	1011	1100	1101	1110	1111
Exceso-3	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100

El **código Gray** varía un único bit entre códigos sucesivos. Para convertir un número binario a Gray, se aplica una operación `XOR` con el mismo número desplazado un bit a la derecha. Para convertir de código Gray a binario, el MSB no cambia y cada bit generado es el `XOR` entre el bit generado anterior y el bit Gray actual.

3. Álgebra de Boole

El **álgebra de Boole** es una estructura matemática que sistematiza operaciones lógicas utilizando técnicas algebraicas para tratar expresiones de la lógica proposicional.

Dado un conjunto formado por al menos dos elementos $\mathfrak{B} = \{\varnothing, U\}$ en el que se define:

- Una **operación unitaria** interna llamada **complemento**: $a \to b = \sim a$.
- Una **operación binaria** interna llamada **suma** $(a, b) \to c = a \oplus b$.
- Una **operación binaria** interna llamada **producto** $(a, b) \to c = a \odot b$.

El conjunto y las operaciones son un álgebra de Boole si se cumplen los siguientes axiomas:

- Suma y producto son **asociativas**.
- Suma y producto son **conmutativas**.
- Suma y producto son **distributivas**.
- Existencia de **elemento neutro**.
- Existencia de **elemento complementario**.

A partir de los axiomas se deducen los teoremas fundamentales:

- **Idempotencia** para la suma y el producto.
- **Absorción** para la suma y el producto.
- **Identidad** para la suma y el producto.
- **Involución**.
- **Complemento**.
- **DeMorgan**.

El **principio de dualidad** enuncia que a toda función lógica le corresponde su dual formada intercambiando los operadores $\oplus$ con $\odot$, y los U con $\varnothing$.

La **lógica binaria** es un sistema lógico cuyas variables adoptan sólo dos valores.

El conjunto es $\mathfrak{B} = \{0, 1\}$ en el que se define:

- El complemento se llama **negación**: $a \to b = \overline{a}$.
- La suma se llama **suma** $(a, b) \to c = a + b$.
- El producto se llama **producto** $(a, b) \to c = a \cdot b$.

La lógica binaria es un álgebra de Boole porque cumple con los axiomas.

4. Funciones lógicas

Una **función lógica** es una función matemática y lógica cuyos argumentos de entrada y su valor de salida asumen uno de los dos valores lógicos. Toman la forma

$$f : \mathfrak{B}^n \to \mathfrak{B}$$

donde $\mathfrak{B} = \{0, 1\}$ y n es un entero no negativo. Hay 2^{2^n} posibles funciones.

Existen tres formas básicas de representar una función lógica:

- **Expresión algebraica:** representación matemática de las expresiones lógicas utilizando las tres operaciones básicas. Combinando las tres se puede construir cualquier expresión lógica.
- **Tabla de verdad:** tabla que tiene una columna para cada variable de entrada y filas en las que toman cada valor posible. En una última columna se muestra el resultado correspondiente a cada fila. Los términos indiferentes se designan con una X y sirven como *comodines* a la hora de operar.
- **Gráfica:** utilizada en el desarrollo de circuitos electrónicos, con símbolos normalizados y conexiones entre ellos. Cada operación tiene su correspondiente **puerta lógica**.

La **estandarización** de los términos de una función lógica simplifica y sistematiza su análisis e implementación.

Un **término canónico** es un término donde aparecen todas las variables.

- **Minitérmino:** consiste de producto y negación. Representa a los términos que en la tabla de verdad son 1.
- **Maxitérmino:** consiste de suma y negación. Representa a los términos que en la tabla de verdad son 0.

Todas las expresiones booleanas, independientemente de su forma, pueden convertirse en cualquiera de las dos **formas estándar**:

- **Suma de productos:** Suma de minitérminos.
- **Producto de sumas:** Producto de maxitérminos.

Las funciones lógicas se pueden **simplificar** utilizando los axiomas y teoremas del álgebra de Boole.

Los **mapas de Karnaugh** reducen la necesidad de cálculos extensos para simplificar expresiones booleanas, aprovechando el reconocimiento de patrones. Se organiza una estructura matricial similar a una tabla de verdad, donde cada celda representa el valor de salida dependiendo de las entradas. Se seleccionan rectángulos de tamaño 2^n (1, 2, 4, 8, 16) que encierren todos los unos o ceros.

5. Circuitos combinacionales

Un **circuito combinacional** es un sistema digital cuyas salidas dependen únicamente del valor de sus entradas. Están compuestos por puertas lógicas interconectadas entre sí, que representan ecuaciones booleanas.

- **Lógicos:** generador/detector de paridad, conversor de código, codificador, generador de códigos detectores de error.
- **Gestión de datos:** multiplexor, demultiplexor, codificador/decodificador para transmisión de datos.
- **Aritmético-lógicos:** sumador, restador, multiplicador, divisor, comparador, desplazador, rotador, ALU.

5.1. Codificadores y decodificadores

Un **codificador** es un circuito combinacional que convierte símbolos de un alfabeto en otro, implementando un proceso de codificación.

Una de sus aplicaciones más comunes es la **compresión** de información para hacer más sencillo su procesamiento, almacenamiento y transmisión.

Los **codificadores binarios** codifican la información proveniente de 2^n entradas en un código binario de n bits, todas teniendo la misma importancia.

Un **codificador binario con prioridad** contempla la posibilidad de dos entradas activas en simultáneo y genera el código correspondiente a la entrada que se prefiera.

Otra de las aplicaciones de la codificación es la **conversión de código**. Por ejemplo, los **conversores BCD-7 segmentos** convierten un BCD en información que puede ser mostrada en un display digital. Un **decodificador** implementa la función inversa de un codificador.

Se pueden implementar para **descomprimir información**.

Un **decodificador binario** identifica combinaciones de bits en sus entradas e indica su presencia mediante una salida dada. Tiene n líneas de entrada y 2^n líneas de salida.

Decodificadores de más salidas pueden construirse a partir de decodificadores más pequeños

5.2. Multiplexores y demultiplexores

Los **multiplexores** son circuitos combinacionales con 2^n entradas de datos y una única salida. Una entrada de selección de tamaño n permite seleccionar la entrada de datos que se conectará con la salida. Se implementan con una suma de productos, con 2^n puertas `AND` y una puerta `OR`.

Multiplexores de muchas entradas se pueden construir a partir de multiplexores de una menor cantidad de entradas.

Los **demultiplexores** son circuitos combinacionales con una única entrada de datos y 2^n salidas. Una entrada de selección de tamaño n permite seleccionar a que salida se conectará la entrada de datos. Se implementan con 2^n puertas `AND`.