

# Sistemi di numerazione

Andrea Passerini  
passerini@disi.unitn.it

Informatica

## Sommario

- informatica
- rappresentare informazioni
- la differenza Analogico/Digitale
- i sistemi di numerazione posizionali
- il sistema binario

## Definizione

*la scienza della rappresentazione e della elaborazione dell'informazione*

- L'informazione è costituita dal connubio di:
  - dati** rappresentazioni di entità di interesse
  - interpretazione** regole per comprendere e manipolare i dati

## Componenti

**supporto fisico** l'informazione deve essere scritta su un qualche supporto fisico che funga da veicolo (e.g. la carta, un disco magnetico)

**linguaggio** è un insieme di regole che permette di scrivere e leggere informazione, e di interpretarla (e.g. la lingua Italiana, i sistemi di numerazione posizionali)

## Linguaggio

- Un linguaggio è tipicamente costituito da:
  - alfabeto** un insieme di simboli appartenenti al linguaggio
  - regole** un insieme di regole che permettono di combinare tali simboli in costrutti del linguaggio, e di interpretarli.
- Perché ogni simbolo contribuisca un minimo di informazione, un alfabeto deve avere almeno due simboli distinti

## Sistemi automatici

- Per poter elaborare automaticamente informazione, è necessario darne una rappresentazione che sia gestibile da uno strumento automatico
- Esistono fonti di informazione estremamente diverse, e il loro trattamento automatico richiede una uniformazione nella modalità con cui vengono rappresentate
- Come vedremo, tutta l'informazione gestita dal calcolatore viene mantenuta in forma numerica, con il più semplice sistema di numerazione possibile, quello binario (con due soli simboli)
- Inoltre tutti i dati vengono memorizzati tramite numeri interi finiti (possibilmente con un'unità frazionaria)

## Rappresentazione di dati

- Dal punto di vista della rappresentazione, un dato può essere di vari tipi:
  - categorico** rosso,verde,blu
  - ordinale** orrendo,brutto,bello,fantastico
  - numerale discreto** 10, 159, -10
  - numerale continuo**  $1e-13$ ,  $\sqrt{2}$
- Tutti questi tipi possono essere rappresentati o approssimati tramite numeri interi (possibilmente con un'unità frazionaria)

## Dati categorici come interi

- Si usa il metodo del *dizionario*
- Ad ogni possibile valore viene associato un numero intero, con cui viene rappresentato
- Esempio: rosso  $\rightarrow$  0, verde  $\rightarrow$  1, blu  $\rightarrow$  2
- L'unica operazione definita su questo tipo di dati è *l'uguaglianza*, e la codifica preserva tale operazione (rosso=rosso  $\rightarrow$  0=0)

## Dati ordinali come interi

- Si usa il metodo dell'*enumerazione*
- I valori sono ordinati, ed ad ognuno viene associato un numero intero crescente
- Esempio: orrendo  $\rightarrow$  0, brutto  $\rightarrow$  1, bello  $\rightarrow$  2 fantastico  $\rightarrow$  3
- La codifica rispetta la relazione di ordine (orrendo  $<$  bello  $\rightarrow$  0  $<$  2)

## Dati reali come combinazione di interi

- Un reale non può essere rappresentato che come reale, ma può essere *approssimato*
- Un numero razionale può approssimare un numero reale:

$$\sqrt{2} = 1.41421356237309504880168872420969807..$$

- In pratica si sceglie il numero di cifre della rappresentazione, e si approssima il reale con quelle cifre. E.g. con 10 cifre frazionarie:

$$\sqrt{2} \approx 1.4142135624$$

- Il numero risultante può essere scritto come la combinazione di un intero per una unità frazionaria

$$1.4142135624 = 14142135624 * 0.0000000001$$

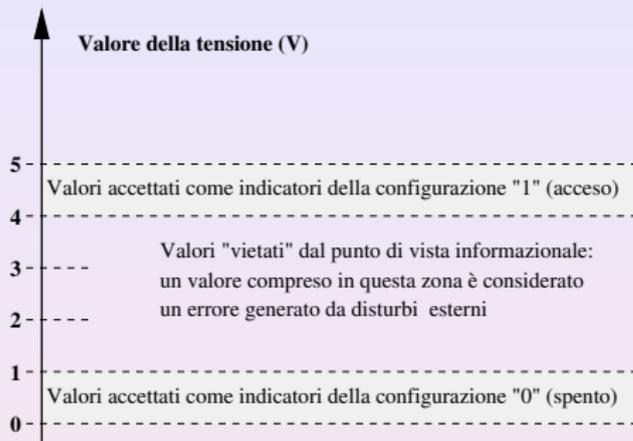
## Codica analogica contro codifica digitale

- Come è meglio codificare l'informazione per la sua elaborazione automatica mediante un calcolatore ?
- Grandezze naturali come l'intensità di un suono assumono valori *continui*.
  - Una codifica si dice *analogica* se mantiene un'analogia tra la struttura dell'entità di informazione e struttura della configurazione (possiamo parlare di codifica *continua*).
  - Una codifica si dice *digitale* se impone un numero di configurazioni distinte ammissibili e converte un'entità di informazione in una di queste configurazioni mediante una regola di codifica (possiamo parlare di codifica *discreta*).

## Vantaggi della codifica digitale

- Qualsiasi sistema fisico (incluso il calcolatore) è sottoposto all'influenza dell'ambiente circostante che ne perturba la configurazione introducendo *rumore*.
  - E' necessario rendere il sistema fisico che elabora l'informazione il più immune possibile al rumore.
  - La codifica analogica è fortemente sensibile al rumore, poichè tutte le configurazioni sono lecite e non si può distinguere la componente di informazione dal contributo dovuto al rumore.
  - Minore è il numero di configurazioni possibili maggiore è la possibilità di isolare l'informazione dal rumore.
  - E' questa la ragione principale del successo della codifica *binaria* nei calcolatori elettronici (la più estrema codifica discreta: due soli valori per ogni simbolo)

# Codifica binaria di un segnale elettrico

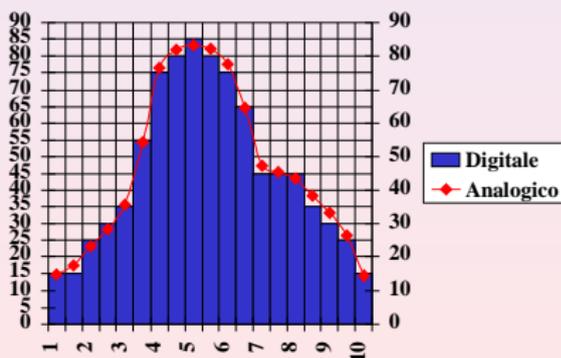
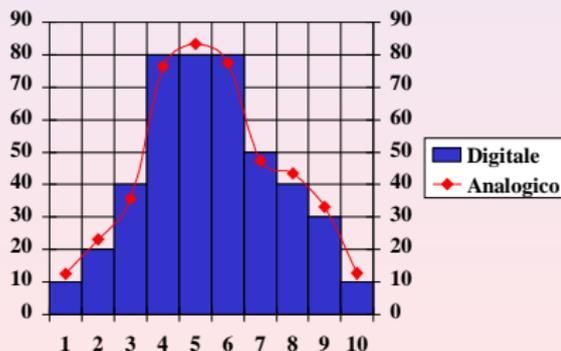


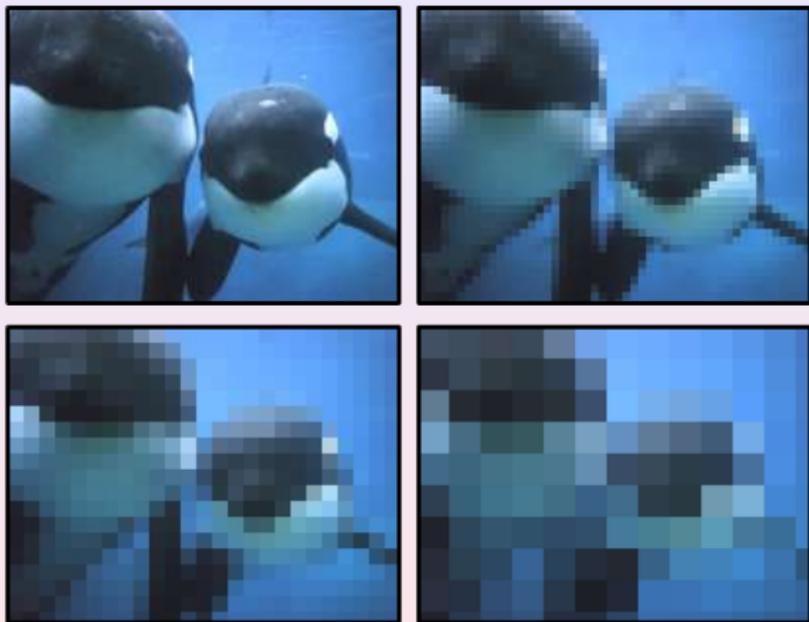
- Si introduce una tolleranza a piccoli effetti di rumore (anche in presenza di una tensione spuria di 0.5 volt le configurazioni rimangono stabili)
- Si introduce la possibilità di riconoscimento automatico della presenza di rumore (una tensione di 3 volt corrisponde ad uno stato informativo vietato)

# Codifica digitale di segnali analogici

## Campionamento

- Permette di approssimare in maniera arbitrariamente accurata informazione continua (analogica).
- Consiste nel:
  - 1 misurare il segnale ad intervalli regolari (di tempo se evolve nel tempo, come un suono, di spazio se immagine, entrambi se video)
  - 2 approssimare ogni misura continua con un valore discreto





## Premessa

- Cosa intendiamo quando scriviamo (ad es.) ?

1945732

unmilionenovecentoquarantacinquemilasettecentotrentadue

- Questo si ottiene come:  
due + trenta + settecento + cinquemila + ... + unmilione
- Ogni cifra viene moltiplicata per un peso. Da destra a sinistra i pesi sono 1, 10, 100, 1000, ..., 1000000, ossia  $10^0, 10^1, 10^2, 10^3, \dots, 10^6$
- Il totale si ottiene sommando il valore di ciascuna cifra moltiplicato per il peso corrispondente alla sua posizione.

# Sistema di numerazione posizionale

## sistema *posizionale*

- Si parla di sistema *posizionale* perché il valore di una cifra dipende dalla sua posizione
- Le posizioni si contano da destra a sinistra a partire da 0
- Il valore della cifra viene moltiplicato per la base (10) elevata alla posizione
- Esempio: 1456

cifra	posizione	valore
6	0	$6 * 10^0 = 6$
5	1	$5 * 10^1 = 50$
4	2	$4 * 10^2 = 400$
1	3	$1 * 10^3 = 1000$

- Tale sistema di numerazione facilita notevolmente le operazioni aritmetiche rispetto ai sistemi non posizionali (e.g. quello dei Romani)

## Ingredienti

- Un numero naturale  $b$  detto *base*. (e.g. 10)
- Un insieme ordinato di  $b$  simboli distinti detti *cifre* (e.g. 0,1,2,3,4,5,6,7,8,9)
- Un *codice di interpretazione* per determinare il numero rappresentato da una stringa di cifre.
- Un insieme di procedure (algoritmi) per le quattro operazioni aritmetiche  $+, -, \times, /$

## Codice di interpretazione

- Un numero intero rappresentato in base  $b$  con  $n$  cifre è una stringa di cifre:

$$(c_{n-1} \cdots c_0)_b$$

- Ad ogni posizione nella stringa è associato un peso. Da destra a sinistra i pesi sono:

$$b^0, \dots, b^{n-1}$$

- Ogni cifra rappresenta il numero di volte in cui deve essere considerato il peso corrispondente alla posizione in cui si trova la cifra stessa

# Forma polinomiale

- Possiamo dunque definire la seguente relazione detta *forma polinomiale*

$$(c_{n-1} \cdots c_0)_b = c_0 \times b^0 + \cdots + c_{n-1} \times b^{n-1}$$

- analogamente possiamo scrivere numeri frazionari:

$$(.c_{-1} \cdots c_{-m})_b = c_{-1} \times b^{-1} + \cdots + c_{-m} \times b^{-m}$$

- o numeri con parte intera e parte frazionaria:

$$(c_{n-1} \cdots c_0.c_{-1} \cdots c_{-m})_b = c_{n-1} \times b^{n-1} + \cdots + c_0 \times b^0 + c_{-1} \times b^{-1} + \cdots + c_{-m} \times b^{-m}$$

## Nota

Il sistema di numerazione in base 10, o *sistema decimale*, è il sistema comunemente usato, ed i numeri in base 10 sono rappresentati di norma senza l'indicazione della base.

## Sistema binario (base 2)

- E' il sistema di numerazione con la base più piccola possibile
- In questo caso le cifre sono  $\{0,1\}$
- Si parla di cifra binaria (*binary digit* o *bit*)
- Il bit è l'unità minima di informazione.
- Conversione binario  $\rightarrow$  decimale tramite la sua forma polinomiale:

$$\begin{aligned}(1011.01)_2 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} \\ &= (11.25)_{10}\end{aligned}$$

# Conversione decimale $\rightarrow$ binario di numeri interi

- Problema formulabile come segue: dato l'intero decimale  $N$ , determinare la stringa di bit  $c_{n-1} \cdots c_2 c_1 c_0$  tale che

$$N = c_0 \times 2^0 + c_1 \times 2^1 + c_2 \times 2^2 + \cdots + c_{n-1} \times 2^{n-1}$$

- Si applica il metodo delle *divisioni successive*.
- Dividiamo entrambi i membri dell'uguaglianza per 2:

$$N/2 = c_0 \times 2^{-1} + c_1 \times 2^0 + c_2 \times 2^1 + \cdots + c_{n-1} \times 2^{n-2} = R \times 2^{-1} + Q$$

dove  $R = N \bmod 2$  e  $Q = \lfloor N/2 \rfloor$  sono rispettivamente resto e quoziente della divisione.

- Si ricava così che:

$$c_0 = R = N \bmod 2$$

$$Q = \lfloor N/2 \rfloor = c_1 \times 2^0 + c_2 \times 2^1 + \cdots + c_{n-1} \times 2^{n-2}$$

# Conversione decimale $\rightarrow$ binario di numeri interi

- Se  $Q \neq 0$  possiamo ripetere la procedura:

$$\begin{aligned}Q &= c_1 \times 2^0 + c_2 \times 2^1 + \dots + c_{n-1} \times 2^{n-2} \\Q/2 &= c_1 \times 2^{-1} + c_2 \times 2^0 + \dots + c_{n-1} \times 2^{n-3} \\&= R' \times 2^{-1} + Q'\end{aligned}$$

- da cui

$$\begin{aligned}c_1 &= R' = Q \bmod 2 \\Q' &= \lfloor Q/2 \rfloor = c_2 \times 2^0 + \dots + c_{n-1} \times 2^{n-3}\end{aligned}$$

- Iterando finché si ottiene una divisione con quoziente nullo.

- Convertire in binario il numero decimale  $(61)_{10}$

61	$\text{mod } 2 = 1 = c_0$	(least significant bit),	$\lfloor 61/2 \rfloor =$	30
30	$\text{mod } 2 = 0 = c_1$		$\lfloor 30/2 \rfloor =$	15
15	$\text{mod } 2 = 1 = c_2$		$\lfloor 15/2 \rfloor =$	7
7	$\text{mod } 2 = 1 = c_3$		$\lfloor 7/2 \rfloor =$	3
3	$\text{mod } 2 = 1 = c_4$		$\lfloor 3/2 \rfloor =$	1
1	$\text{mod } 2 = 1 = c_5$	(most significant bit)	$\lfloor 1/2 \rfloor =$	0

- quindi  $(61)_{10} = (111101)_2$

# Conversione decimale $\rightarrow$ binario di numeri frazionari

- Problema formulabile come segue: dato il numero frazionario decimale  $F$ , determinare la stringa di bit  $c_{-1}c_{-2}c_{-3}\dots c_{-m}$  tale che

$$F = c_{-1} \times 2^{-1} + c_{-2} \times 2^{-2} + c_{-3} \times 2^{-3} + \dots + c_{-m} \times 2^{-m}$$

- Si applica il metodo delle *moltiplicazioni successive*.
- Moltiplichiamo entrambi i membri dell'uguaglianza per 2:

$$F \times 2 = c_{-1} + c_{-2} \times 2^{-1} + c_{-3} \times 2^{-2} + \dots + c_{-m} \times 2^{-m+1} = N + F'$$

dove  $N = c_{-1}$  è la parte intera del risultato ed

$F' = c_{-2} \times 2^{-1} + c_{-3} \times 2^{-2} + \dots + c_{-m} \times 2^{-m+1}$  è la parte frazionaria.

- Si itera il procedimento sulla parte frazionaria finchè si verifica una delle due condizioni:
  - 1 Si raggiunge il numero massimo di cifre binarie con cui si intende rappresentare il numero frazionario (si ottiene un' *approssimazione per difetto* del numero decimale).
  - 2 Si ottiene come risultato di una moltiplicazione un numero con parte frazionaria nulla.



# Operazioni aritmetiche binarie

- Tabella per la sottrazione binaria:

$$0 - 0 = 0$$

$$0 - 1 = 1 \quad \text{con prestito di } 1$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

- Esempio

$$\begin{array}{r} \phantom{1} \phantom{0} \phantom{1} \phantom{1} \phantom{0} \phantom{0} \phantom{1} \phantom{-} \\ 1 \phantom{0} \phantom{1} \phantom{1} \phantom{0} \phantom{0} \phantom{1} \phantom{1} \phantom{-} \\ \phantom{1} \phantom{0} \phantom{1} \phantom{1} \phantom{0} \phantom{0} \phantom{1} \phantom{1} \phantom{-} \\ \hline 0 \phantom{1} \phantom{0} \phantom{0} \phantom{1} \phantom{1} \phantom{0} \phantom{0} \phantom{-} \end{array}$$