

1 Alfabeti

Quello che non mi piace degli alfabeti, è che in fondo è definito solo quello di destinazione: quello del mondo reale ne è una conseguenza. Che senso ha definirlo? Si definiscono le regole, i simboli del mondo-computer, e si ottengono i simboli del mondo reale.

2 Introduzione

In generale, lo scopo della comunicazione è lo scambio di un'informazione. Questo scambio può avvenire tra qualsiasi tipo di essere: umano, animale, vegetale. Per gli essere umani, la comunicazione avviene attraverso il linguaggio, sia in forma orale che in forma scritta.

1) Parzialità: una comunicazione non *trasmette* mai la realtà completa, ma solo una sua **parte**.

Ad esempio, quando vediamo un cane che sta correndo, possiamo trasmettere questa informazione dicendo:

“Guarda c'è un cane che corre”

Ma questa non è una descrizione completa della realtà, infatti le seguenti domande non hanno risposta:

- Chi è quel cane?
- Verso dove corre?
- Quanto veloce?
- Di che razza è?

2) Interpretabilità: una comunicazione ricevuta da due diversi soggetti potrà essere interpretata **diversamente**.

Rispetto l'esempio del cane, ogni ricevente potrà immaginarsi diversi scenari:

- Una cane che corre in mezzo alla strada.
- Un cane che corre sul marciapiedi.
- Uno può immaginarsi un Dobermann, un altro un Chihuahua.

E tutto questo a seconda della personalità di chi riceve.

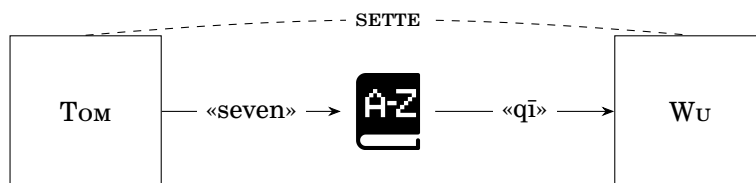
3) Limitatezza: una comunicazione potrà essere effettuata utilizzando un numero finito di simboli.

Rispetto all'esempio di cui sopra i simboli sono le parole, e sebbene quella

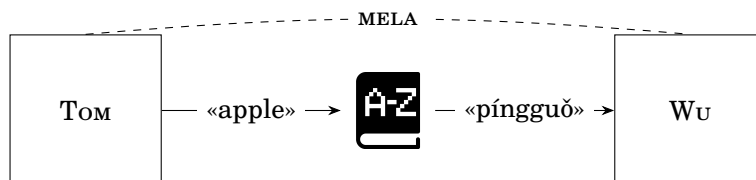
frase potrà essere detta in modi diversi, le parole usate saranno limitate a quelle della lingua italiana. La limitatezza nel caso degli essere umani è dovuta alla possibilità di poter conoscere solamente un numero limitato di parole.

2.1 Esempio

Iniziamo da un esempio. In un bar ci sono due studenti di informatica, un inglese di nome Tom ed un cinese di nome Wu, entrambi parlano solamente le proprie lingue. Tom vuole comunicare a Wu il numero SETTE. Per fare questo Tom prende un dizionario Inglese-Cinese e cerca il termine «seven» e trova che si dice «qī», e lo comunica a Wu:



Comunicare mela: Adesso Tom vuole comunicare a Wu MELA, e lo fa utilizzando sempre lo stesso dizionario.



Ricapitolando:

	SETTE		MELA	
	TOM	WU	TOM	WU
Termine	«seven»	«qī»	«apple»	«píngguǒ»

2.2 La lingua è un codice?

Dato questo esempio, possiamo dire che quello che Tom fa è codificare il termine «seven» in «qī» ed il termine «apple» in «píngguǒ». Ma quando Wu ascolta «píngguǒ» gli sorge un dubbio:

Tom intende il frutto?
oppure Tom intende l'azienda Apple?

Essendo sia Tom che Wu due studenti di informatica.

Questo problema può presentarsi non solo con MELA ma con qualsiasi termine.

Conclusione Per capire il **significato** di un termine, che sia in italiano, cinese o inglese, non basta il termine stesso. Ogni termine è interpretabile a seconda del contesto, del tono e di altri fattori.

- SETTE, che può essere «seven» o «qī» è poco interpretabile (ma può sempre confondersi col film di David Fincher).
- MELA, soprattutto se detto da un inglese ad un cinese, entrambi studenti di informatica, è molto interpretabile.

Nel linguaggio l'interpretabilità di ogni termine è considerato anche un vantaggio, in informatica no:

Ogni termine in informatica deve avere un significato preciso, ovvero non interpretabile.

Quando un linguaggio è definito da termini ognuno con un preciso e non-interpretabile significato, viene detto **codice**.

3 Codice

In informatica il codice viene utilizzato per legare due mondi: quello reale e quello del computer. Per questo quando si parla di codice si parla anche di codifica e decodifica. La codifica è il passaggio dal mondo reale a quello del computer, la decodifica è il passaggio dal mondo del computer a quello del reale.

Nel mondo reale, come abbiamo visto precedentemente con l'esempio di Tom e Wu, un significato può assumere diverse forme. Ad esempio prendiamo il numero 10, esso può assumere diverse *forme*:

- «10» con i numeri arabi.
- «dieci» in italiano.
- «ten» in inglese.
- «X» nei numeri romani.
- «IIIIIIII» nella numerazione unaria.
- «1010» in base 2.
- «12» in base 8.
- «A» in base 16.
- «K» in base 64.

Tutte queste forme vengono dette **significanti** del significato 10. Il numero dieci nel mondo dei computer invece diventa qualcosa definito molto precisamente:

- Si stabilisce come convertirlo in binario.
- Si stabilisce in quanti bit.

Ora proviamo ad effettuare diverse codifiche:

- 8-bit, MSBA sinistra: 0000·1010.
- 16-bit, MSBA sinistra: 0000·0000·0000·1010.
- 8-bit, MSBA destra: 0101·0000.
- 16-bit, MSBA destra: 0101·0000·0000·0000.

Tutte queste sono possibili codifiche del numero 10, il calcolatore dovrà poi utilizzarle sapendo a quale codifica esse appartengono.

Analizziamo ora in questi termini tutte le codifiche fin'ora viste.

Dopo aver visto varie tipologie di conversione, possiamo finalmente dare la definizione di *codice*:

Un codice è un insieme di regole per convertire una informazione in un altro oggetto o in un'altra azione, non necessariamente dello stesso tipo.

Facciamo un paio di esempi:

	tipo infor- mazione	di	si trasforma in	come?
Semaforo	Colore		Azione	Rosso \rightarrow <i>stop</i> ; verde \rightarrow <i>procedere</i> .
Codice Morse	Sequenza di punti/linee		Lettera/numero	$\dots \rightarrow$ S; --- \rightarrow 0; \dots --- $\dots \rightarrow$ SOS.

Conversione tra basi La conversione tra basi che abbiamo visto noi (tra basi 2/8/10/16) può rientrare nella definizione di codice data sopra, ma in senso molto ampio. Fintanto che convertiamo un numero tra una base ed un'altra, stiamo solamente cambiando la sua rappresentazione, e lo scopo di questa conversione, potremmo dire, non *va al di là del foglio di carta in cui abbiamo effettuato la conversione*. 1111·0000·1010·1100

Scopo Un codice, invece, non cambia solo la rappresentazione di una informazione, ma la cambia avendo già in mente un preciso scopo. Quindi:

- Effettuiamo un **cambio di rappresentazione** quando convertiamo una informazione in un'altra forma senza ulteriore scopo.
- Effettuiamo una **codifica** quando convertiamo una informazione in un'altra forma con uno scopo preciso.

Questo è coerente con gli esempi di cui sopra:

- Il codice morse viene utilizzato per convertire caratteri alfanumerici in sequenze di linee e punti in quanto linee e punti, una volta convertiti in segnali radio, sono più facili e meno soggetti ad errore durante la trasmissione e la ricezione in una comunicazione radiofonica, per di più, funzionano bene anche se utilizzati in forma luminosa o sonora.
- Il codice *semaforo* viene utilizzato per convertire un colore in una azione in quanto per un essere umano è più immediato e meno soggetto ad errore convertire un colore in un'azione¹.

Conversione in base 2 Pensiamo quindi ad una conversione del numero 125 in base 2. Tale numero è 1111101. Questa conversione è senza scopo ulteriore, e quindi è un *cambio di rappresentazione*. Se volessimo invece effettuare una codifica, dovremmo capire qual è lo scopo di tale conversione? Allora potremmo modificare la conversione in questo modo:

Convertire il numero 125 in base 2 per un computer con un processore di 16 bit.

In questo caso il risultato diventerà: 0000 0000 0111 1101. Avendo specificato lo scopo, ovvero di essere utilizzato da un computer di 16 bit, abbiamo applicato una *codifica*. Una volta stabilito lo scopo, passare da base 2 a base 8 o base 16, non è altro che un cambio di rappresentazione, ma all'interno dello scopo definito, quindi quello di essere utilizzato con un computer con un processore di 16 bit. Quindi, solo per quanto riguarda i numeri:

- *Codifica*: convertire un numero qualsiasi in qualsiasi base in un numero compatibile con l'utilizzo di un calcolatore.
- *Cambio di rappresentazione*: convertire un numero qualsiasi in qualsiasi base in un'altra base.

Infine useremo i seguenti termini:

- **Valore**: il numero come quantità astratta, cioè indipendentemente da come lo scriviamo.
- **Rappresentazione**: lo stesso valore può essere scritto in più modi (ad esempio base 2, base 8, base 10).
- **Codice**: il valore viene rappresentato in una forma compatibile ad uno specifico scopo.

3.1 Codice numeri naturali

I numeri naturali \mathbb{N} sono i numeri interi positivi che vanno da 0 a infinito. La codifica utilizzata converte il numero da base 10 a base 2 e aggiunge gli zeri necessari a raggiungere il numero di bit richiesti.

¹Si pensi se invece dei colori ci fossero le scritte.

Esempio Dato il valore 56 rappresentato in base 10, esso potrà essere rappresentato anche come:

- in base 2: «11 1000»₂.
- in base 8: «70»₈.
- in base 16: «38»_H.

La codifica per un computer a 16 bit sarà:

- «11 1000»₂ a 16-bit \rightarrow $\langle 0000 \cdot 0000 \cdot 0011 \cdot 1000 \rangle$.

Limiti della codifica Con questa codifica avremmo i seguenti limiti:

- Numero massimo codificabile: $2^{16} - 1 = 65535$.

3.1.1 Rappresentazione in altre basi

Le basi 16 e le basi 8 sono utili per **compattare** un numero binario in un formato più leggibile. Comune è l'esempio dei colori codificati in RGB a 24-bit. In ogni caso, i computer usano sempre e solo i bit. *Attenzione* a rappresentare diversamente 0000·0000·0011·1000:

- In base 16 \rightarrow «0038»₁₆.
- In base 8 \rightarrow «00070»₈.

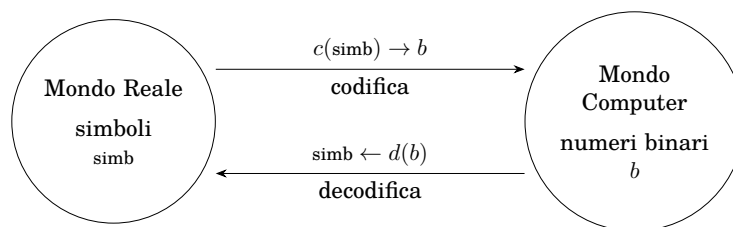
MA ATTENZIONE al procedimento inverso:

- 0038₈ in base 2 diventa 0000·0000·0111·1101, che è esattamente il codice.
- 00070_H in base 2 diventa 00·0000·0000·0111·1101, che **non** è esattamente il codice in quanto abbiamo due zeri in più che non esistono nel codice.

Per cui si deve sempre ricordare le regole della codifica applicata.

3.2 Funzioni

Abbiamo quindi due mondi:



Indichiamo con:

- $c(\text{simb}) \rightarrow b$ la funzione di codifica, che trasforma un qualsiasi simbolo simb in un numero binario b .
- $d(b) \rightarrow \text{simb}$ la funzione di decodifica, che trasforma un qualsiasi numero binario b in un simbolo simb .

Nel caso della codifica con numeri interi precedente avremmo:

- simb : numeri interi.
- b : numeri binari a n -bit.
- $c(\text{simb})$: conversione da base 10 a base 2.

3.2.1 Funzione da base10 a base2

Definiamo la seguente funzione

$$f_{10 \rightarrow 2}^n(d)$$

la funzione che riceve in input un numero decimale e lo converte in un numero binario a n bit. Esempio:

$$\begin{aligned} f(13) &= 1101 \\ f^5(13) &= 0 \cdot 1101 \\ f^{16}(13) &= 0000 \cdot 0000 \cdot 0000 \cdot 1101 \end{aligned}$$

Se n non è definito, allora si userà il minor numero possibile di bit necessari.

4 Numeri interi con segno

Con la codifica dei numeri naturali è impossibile codificare il simbolo $-$, dato che nel mondo dei computer esistono solamente i numeri binari. Per cui è necessario utilizzare un diverso tipo di codifica. Per codificare i numeri relativi (ovvero sia maggiori che minori di zero) le seguenti sono le codifiche principali:

- Modulo e segno.
- Complemento a 1.
- Complemento a 2.

Numeri positivi Per quanto riguarda i numeri positivi, tutte e tre le codifiche –una volta stabilito il numero di bit– usano la stessa funzione $c(\text{simb})$, quella dei numeri naturali. Per cui il valore $+56$ codificato per un calcolatore di 16 bit sarà

$$+56 \rightarrow 0000 \cdot 0000 \cdot 0111 \cdot 1101_2$$

per tutte e tre le codifiche.

4.1 Modulo e segno

Nella codifica “Modulo e segno” il msb viene utilizzato per indicare il segno:

- $\text{msb} = 0$: allora avremo segno $+$ (numero positivo).
- $\text{msb} = 1$: allora avremo segno $-$ (numero negativo).

Definiamo la funzione

$$\text{MS}(b)$$

la quale richiede come input un numero binario b e ne inverte il MSB (ovvero la prima cifra da sinistra). Esempi:

$$\begin{aligned}\text{MS}(\underline{0}010) &= \underline{1}010 \\ \text{MS}(\underline{1}011000010) &= \underline{0}011000010\end{aligned}$$

Regole codifica

- Nel caso dei numeri positivi, si convertono utilizzando $\text{BIN}(\text{simb})$.
- Nel caso dei numeri negativi, si convertono utilizzando $\text{MS}(\text{BIN}(\text{simb}))$.

ESEMPIO NUMERO NEGATIVO: Dato il numero -56 , lo si vuole codificare per l'utilizzo con un calcolatore a 16 bit:

- Si converte il numero senza segno 56 in binario a 16 bit:

$$b = \text{BIN}(56) \rightarrow \langle 0000 \cdot 0000 \cdot 0011 \cdot 1000 \rangle$$

- Si applica la funzione $\text{MS}()$:

$$\text{MS}(b) = \underline{1}000 \cdot 0000 \cdot 0011 \cdot 1000$$

Decodifica Per decodificare, ovvero passare da codice binario a simbolo, si devono effettuare due passaggi:

1. Capire se il numero è negativo: msb uguale a 1.
2. Calcolare il modulo applicando $\text{B10}(\text{MS}(b))$.

Esempio:

- Numero positivo $b = \langle 0000 \cdot 0011 \cdot 1110 \cdot 1000 \rangle$:

$$\text{simb} = \text{B10}(b) \rightarrow 1000$$

- Numero negativo $b = \langle 1000 \cdot 0011 \cdot 1110 \cdot 1000 \rangle$:

$$\begin{aligned}\text{simb} &= \text{B10}(\text{MS}(b)) \rightarrow \\ &\rightarrow \text{B10}(0111 \cdot 1100 \cdot 0001 \cdot 0111) \\ &\rightarrow -1000\end{aligned}$$

Limiti

- Dato che il MSB è utilizzato per indicare il segno, solamente 15 bit dei 16 bit possono essere utilizzati per il valore numerico, quindi il massimo numero codificabile sarà: $2^{15} - 1 = 32767$.
- E lo stesso vale per il numero più piccolo: $-2^{15} + 1 = -32767$.
- Si ha una doppia rappresentazione dello zero:

$$+0 \rightarrow 0000 \cdot 0000 \cdot 0000 \cdot 0000$$

$$-0 \rightarrow 1000 \cdot 0000 \cdot 0000 \cdot 0000$$

4.2 Complemento a 1

Con la codifica “Complemento a 1” si codificano i numeri negativi invertendo² tutti i bit del numero positivo. Esempio a 16 bit:

$$+10 \rightarrow 0000 \cdot 0000 \cdot 0000 \cdot 1010$$

$$-10 \rightarrow 1111 \cdot 1111 \cdot 1111 \cdot 0101$$

Definiamo la funzione

$$\text{CA1}(b)$$

la quale richiede come input un numero binario b e ne inverte tutti i bit.

Codifica “Complemento a 1” Dato il numero -56 , lo si vuole codificare per l'utilizzo con un calcolatore a 16 bit:

- Si converte il numero senza segno 56 in binario a 16 bit:

$$56 \rightarrow \langle 0000 \cdot 0000 \cdot 0011 \cdot 1000 \rangle.$$

- Si applica la funzione $\text{CA1}()$:

$$\text{CA1}(0000 \cdot 0000 \cdot 0011 \cdot 1000) = \langle 1111 \cdot 1111 \cdot 1100 \cdot 0111 \rangle.$$

Regole

- Nel caso dei numeri positivi, si convertono utilizzando $\text{BIN}(\text{simb})$.
- Nel caso dei numeri negativi, si convertono utilizzando $\text{CA1}(\text{BIN}(\text{simb}))$.

Limiti

- Possiamo applicare la funzione $\text{CA1}()$ solamente a metà dei numeri (quelli che con $\text{MSB} = 0$), per cui il massimo numero codificabile sarà: $2^{15} - 1 = 32767$.
- Lo stesso vale per il numero più piccolo: $-2^{15} + 1 = -32767$.
- Si ha una doppia rappresentazione dello zero:

$$+0 \rightarrow 0000 \cdot 0000 \cdot 0000 \cdot 0000$$

$$-0 \rightarrow 1111 \cdot 1111 \cdot 1111 \cdot 1111$$

²Con *invertire* si intende che gli 0 diventano 1 e gli 1 diventano 0.

Perché solo ai numeri con MSB=1? Se applicassimo $CA1()$ anche ai numeri con $MSB = 1$ riottenremmo codici che rappresentano i numeri positivi. Ad esempio il numero 40000:

$$BIN(40000) \rightarrow 1001 \cdot 1100 \cdot 0100 \cdot 0000$$

Ora applichiamo $CA1()$:

$$CA1(1001 \cdot 1100 \cdot 0100 \cdot 0000) \rightarrow 0110 \cdot 0011 \cdot 1011 \cdot 1111$$

che però rappresenta già 25535.

Decodifica Per decodificare, ovvero passare da codice binario a simbolo, si devono effettuare due passaggi:

- Capire se il numero è negativo: MSB uguale a 1.
- Calcolare il modulo applicando $B10(CA1(b))$.

Esempio:

- Numero positivo, $\langle 0000 \cdot 0011 \cdot 1110 \cdot 1000 \rangle$:

$$B10(0000 \cdot 0011 \cdot 1110 \cdot 1000) \rightarrow 1000$$

- Numero negativo, $\langle 1000 \cdot 0011 \cdot 1110 \cdot 1000 \rangle$:

$$\begin{aligned} & B10(CA1(1000 \cdot 0011 \cdot 1110 \cdot 1000)) \rightarrow \\ & \rightarrow B10(0111 \cdot 1100 \cdot 0001 \cdot 0111) \\ & \rightarrow -31767 \end{aligned}$$

4.3 Complemento a 2

Con la codifica “Complemento a 2” si codificano i numeri negativi applicando la codifica “Complemento a 1” e poi sommando 1. Esempio a 16 bit:

$$\begin{aligned} +10 & \rightarrow 0000 \cdot 0000 \cdot 0000 \cdot 1010 \\ -10 & \rightarrow 1111 \cdot 1111 \cdot 1111 \cdot 0101 + \\ & \quad 0000 \cdot 0000 \cdot 0000 \cdot 0001 = \\ & \quad 1111 \cdot 1111 \cdot 1111 \cdot 0110 \end{aligned}$$

Definiamo la funzione

$$CA2(b) = CA1(b) + 1$$

la quale richiede come input un numero binario b e ne inverte tutti i bit.

Esempio Codifica “Complemento a 1” di -56 per l'utilizzo con un calcolatore a 16 bit:

- Si converte il numero senza segno 56 in binario a 16 bit:

$$56 \rightarrow \langle 0000 \cdot 0000 \cdot 0011 \cdot 1000 \rangle.$$

- Si applica la funzione $CA1()$:

$$CA1(0000 \cdot 0000 \cdot 0011 \cdot 1000) = \langle 1111 \cdot 1111 \cdot 1100 \cdot 0111 \rangle.$$

- Si somma 1:

$$\langle 1111 \cdot 1111 \cdot 1100 \cdot 0111 \rangle + 1 = \langle 1111 \cdot 1111 \cdot 1100 \cdot 1000 \rangle.$$

Regole

- Nel caso dei numeri positivi, si convertono utilizzando $BIN(\text{simb})$.
- Nel caso dei numeri negativi, si convertono utilizzando $CA2(BIN(\text{simb}))$.

Limiti

- Possiamo applicare la funzione $CA1()$ solamente ai numeri con $MSB = 0$, per cui il massimo numero codificabile sarà: $2^{15} - 1 = 32767$.
- Non vale lo stesso per il numero più piccolo grazie alla somma $+1$, infatti sarà: $-2^{15} + 1 = -32768$.
- Non si ha una doppia rappresentazione dello zero.

Perché non si ha una doppia rappresentazione dello zero? Applichiamo il procedimento inverso a $1111 \cdot 1111 \cdot 1111 \cdot 1111$:

- Sommiamo uno ottenendo $\langle 0000 \cdot 0000 \cdot 0000 \cdot 0000 \rangle$.
- $CA1(0000 \cdot 0000 \cdot 0000 \cdot 0000) \rightarrow 1111 \cdot 1111 \cdot 1111 \cdot 1111$.

Per cui otteniamo che $1111 \cdot 1111 \cdot 1111 \cdot 1111 \rightarrow -32768$.

4.4 Numeri interi (complemento a 2)

Definiamo la funzione

$$f_{CA2}(n) = f_{CA1}(n) + 1$$

Esempi:

$$\begin{aligned} f_{CA2}(0010) &= 1101 + 1 = 1110 \\ f_{CA1}(0011000010) &= 1100111101 + 1 \\ &= 1100111110 \end{aligned}$$

La codifica complemento a 2 è la seguente:

- Se il numero è maggiore di zero si converte da base10 a base2.
- Se il numero è minore di zero si utilizza $f_{CA2}(n)$.

Codifica di -125 per l'utilizzo con un calcolatore a 16 bit (complemento a 2):

$$\begin{aligned} f_{CA1}(0000000001111101)_2 + 1 &= \\ &= 1111111110000010_2 + 1 = \\ &= 1111111110000011_2 \end{aligned}$$

Cambio di rappresentazione del codice di cui sopra:

- da base 2 a base 16: $0000000001111101_2 = 007D_{16}$; $1111111110000011_2 = FF83_{16}$.
- da base 2 a base 8: $0000000001111101_2 = 000175_8$; $1111111110000011_2 = 177603_8$.

attenzione: quando si cambia la rappresentazione del codice, non bisogna mai dimenticare lo *scopo* (rappresentare su 16 bit in complemento a 2):

- lo riporto in binario (mantenendo 16 bit): ritorna esattamente lo stesso.
- lo riporto in binario dall'ottale: ottengo 18 bit (6 cifre ottali), quindi devo scartare i 2 bit in più a sinistra e mantenere solo i 16 bit meno significativi.

5 Codifica binaria dei numeri

5.1 Numeri interi

Calcoliamo il numero 125 in base 10 in un numero in base 2. Il procedimento è quello di eseguire la divisione per 2 e raccogliere i resti:

$$\begin{array}{r|l} 125 & 2 = 62 + 1 \\ 62 & 2 = 31 + 0 \\ 31 & 2 = 15 + 1 \\ 15 & 2 = 7 + 1 \\ 7 & 2 = 3 + 1 \\ 3 & 2 = 1 + 1 \\ 1 & 2 = 0 + 1 \end{array}$$

Quindi il numero 125 in base 2 sarà: 1111101, raccogliendo i resti dall'ultima riga fino alla prima.

Informatica:

- Elaborazione (in loco): come l'informazione esiste e viene gestita dentro un sistema.

- Memorizzazione (RAM, file system, database, caching).
- Trasformazione/decisione (algoritmi, computazione, OS, compilatori, ML).
- Affidabilità locale (concorrenza, consistenza, isolamenti, crash recovery).
- Rappresentazione (bit, tipi, formati, strutture dati).
- Comunicazione: come l'informazione passa tra sistemi/processi.
 - Protocolli, reti, distribuzione, sincronizzazione, web.
 - Sicurezza “in transit” (TLS, autenticazione, integrità, ecc.).

Comunicazione richiede la codifica, la quale può essere:

- A lunghezza fissa.
- A lunghezza variabile.
- Significante: 10, 1010, 0x0A, 12.
- Codifica: Numeri arabi in base 10, base2, base16, base8.
- Significato: dieci, dieci, dieci, dieci.
- Insieme dei dati da rappresentare:
 - Alfabeto sorgente (T).
 - Composto da n elementi (cardinalità $n = |T|$).
 - *Esempio*: numeri fino a mille, tutti i numeri reali.
- Insieme dei simboli:
 - Alfabeto codice (E).
 - Composto da k elementi (cardinalità $k = |E|$).
 - *Esempio*: lettere dell'alfabeto, 0/1 dei numeri binari.
- Una sequenza di simboli dell'alfabeto E è detta **stringa**.
- Passare da un dato/significato ad un significativo, ovvero una sequenza di simboli, è detto codifica o trasformazione (funzioni di).

Codici ridondanti:

-