Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma

Fondamenti di Informatica

Laurea in
Ingegneria Civile e Ingegneria per l'Ambiente e il Territorio

Rappresentazione dell'Informazione

Stefano Cagnoni e Monica Mordonini

Informatica = **Infor**mazione + Auto**matica**

- **Informazione** : notizia, dato o elemento che consente di avere conoscenza più o meno esatta di fatti, situazioni, modi di essere.
- **Dato**: ciò che è immediatamente presente alla conoscenza, prima di ogni elaborazione; (in informatica) elemento di informazione costituito da simboli che devono essere elaborati.
  (dal *Vocabolario della Lingua Italiana*, Istituto dell'Enciclopedia Italiana)

Informazione

- E' l'oggetto del complesso processo di comunicazione fra due entità che condividono un medesimo codice d'interpretazione
- Informazione & rumore: l'informazione si dice significativa se ha un effetto sull'utilizzatore, altrimenti è rumore

Rappresentazione dell'Informazione

- L'informazione può essere rappresentata in due forme:
  - Analogica
  - Digitale
- Nella forma analogica una grandezza è rappresentata in modo continuo.
- Nella forma digitale una grandezza è rappresentata in modo discreto.

Rappresentazione dell'Informazione

- La maggior parte delle grandezze fisiche è di tipo continuo.
- Tuttavia alcuni tipi di informazione "artificiali" sono di tipo discreto (ad esempio, un testo scritto).

Elaboratore elettronico
(*computer o calcolatore*)

- Calcolatore = strumento per fare calcoli ?
- Definizione legata alle origini, oggi troppo limitativa!
- In francese = ordinateur (elaboratore)
- Un computer è uno strumento "universale" per l'elaborazione dei dati.

Elaboratore elettronico
(*computer o calcolatore*)

- Strumento per la rappresentazione, la memorizzazione e l'elaborazione delle informazioni
  - uno strumento in grado di eseguire insiemi di *azioni elementari*
  - le azioni vengono eseguite su oggetti (*dati*) per produrre altri oggetti (*risultati*)
  - l'esecuzione di azioni viene richiesta all'elaboratore attraverso frasi scritte in un qualche linguaggio (*istruzioni*)

Programmazione

- L'attività con cui si predispone l'elaboratore a eseguire un particolare insieme di azioni su un particolare insieme di dati allo scopo di risolvere un problema

Risoluzione di un problema

- La descrizione del problema non indica direttamente (in genere) un modo per ottenere il risultato voluto
- differenza tra *specifica di un problema* e *processo di risoluzione*
- La risoluzione di un problema è quel processo che
  - dato un problema
  - individuato un opportuno metodo risolutivo

  trasforma

  - i dati iniziali
  - nei corrispondenti risultati finali

Cenni storici I

- 1600 - Blaise Pascal realizza la prima macchina programmata meccanicamente per eseguire operazioni aritmetiche. E' la cosi detta Pascalina. Poteva effettuare addizioni e sottrazioni.
- 1671 – Leibniz realizza una macchina dello stesso tipo che poteva eseguire le quattro operazioni elementari e la radice quadrata.

Cenni storici II

- 1804 – Joseph Jacquard realizza un telaio per tessitura controllato automaticamente da schede perforate.
- 1833 Charles Babbage progetta la "Analytical Engine" . Possedeva una memoria fatta da pile di ruote dentate e una unità di calcolo in grado di effettuare le quattro operazioni. I dati erano immessi in memoria tramite schede perforate.

Cenni storici III

- Hollerith fonda la *Computing Tabulating Recording Company* che nel 1923 diventerà *International Business Machine* (IBM)
- *1904: invenzione del tubo a vuoto*
- 1945 - All' Università di Princeton J. von Neumann sviluppa l'idea di calcolatore "a memoria programmata". La macchina immagazzina nella memoria non solo i dati da elaborare ma anche le istruzioni del programma. L'architettura di von Neumann è quella usata ancora oggi.

Cenni storici IV

- *1947: invenzione del transistor*
- 1951 – nasce il primo calcolatore commerciale l'UNIVAC I (Universal Automatic Computer).
- *1969: invenzione dei circuiti integrati*
- 1981: Personal Computer IBM

Le generazioni I

- **I generazione – Inizia nel 1951.**
  - I calcolatori sono realizzati con la tecnologia delle valvole termoioniche e con memorie a tamburo magnetico.
  - Hanno capacità di memoria limitata, occupano molto spazio e consumano molta energia.
  - Vengono programmati in linguaggio macchina.
  - Solo più tardi compaiono i primi linguaggi di alto livello come il FORTRAN (1957).

Le generazioni II

- **II generazione – 1959-1965.**
  - Vengono utilizzati i transistor (che sostituiscono le valvole) e le memorie sono costituite da nuclei magnetici. Nascono dischi e nastri magnetici,
  - Nascono il COBOL (linguaggio per applicazioni congressuali) e l'ALGOL (1960) per applicazioni scientifiche (precursore di Pascal, C etc.).
  - Compaiono i primi sistemi operativi.

Le generazioni III

- **III generazione – 1965-1972**.
  - Vengono utilizzati i circuiti integrati.
  - Più transistor vengono "integrati" su di una unica piastrina di silicio detta "chip".
  - Nascono i sistemi in multiprogrammazione ed i sistemi in time-sharing in cui più utenti, collegati tramite terminali, possono utilizzare contemporaneamente lo stesso computer.

Le generazioni IV

- **IV generazione – 1972 - oggi.**
  - Nel 1972 nascono i microprocessori. Una unità centrale (CPU) può essere contenuta in un unico circuito integrato.
  - Collegando la CPU alla memoria e ai dispoasitivi di I/O (input/output)si ottengono i minicomputer. Velocità di eleborazione raggiunte 100 MIPS (Milioni di istruzioni per secondo).
  - Contemporaneo sviluppo della telematica (applicazione delle telecomunicazioni all'informatica).
  - Nascono i primi computer paralleli.

Tipi di Calcolatore

- **Supercalcolatori**: elevata potenza di calcolo per applicazioni scientifiche e modellazione di sistemi complessi
- **Mainframe**: elevata capacità di gestire periferiche per applicazioni gestionali su ampia scala
- **Minicalcolatori**: caratteristiche simili ai mainframe ma su scala più ridotta, per piccole aziende
- **Workstation**: stazioni di lavoro per applicazioni avanzate di progettazione e grafica computerizzata
- **Personal Computer (PC)**: sistemi destinati ad uso personale utilizzabili in diverse configurazioni (postazione singola di lavoro, sistemi multimediali, server di rete, workstation economiche, laptop)

Tipi di dati

- Dati **numerici** (interi e reali)
- Dati **simbolici** (codifica di concetti o simboli: es. vero e falso, caratteri alfanumerici, ecc.)
- Dati **multimediali**
  - testi
  - suoni
  - immagini (fisse o in movimento)
- La potenza e la versatilità del calcolatore derivano dalla possibilità di utilizzare una codifica comune per i diversi tipi di dati.
- Eseguendo operazioni dello stesso tipo a livello *fisico* si possono ottenere risultati interpretabili in modo molto diverso a livello *logico*

Codifica dell'informazione

- Gli esseri viventi ricevono informazione direttamente dal mondo circostante e dai propri simili attraverso i sensi (**percezione**).
- La percezione, tuttavia, è un fatto immediato. L'informazione percepita deve poter anche essere **memorizzata** e **trasmessa** agli altri.
- La memorizzazione e la trasmissione dell'informazione richiedono che questa sia **codificata**.
- Se devo descrivere un fenomeno che non posso riprodurre direttamente o un oggetto che non ho a portata di mano o un concetto astratto ho bisogno di **simboli**.

Codifica dell'informazione

- Un insieme di **simboli** e di regole che determinano come interpretarli costituiscono un **codice.**

Es. la scrittura

SIMBOLI = a b c d e f g h … x y z , . +' ( ) …
ARCO
(potremmo anche leggerlo ocra, cane ecc.)

- Esistono regole che creano una corrispondenza fra ogni simbolo e un suono, che determinano come interpretare gruppi di più simboli ecc.

Codifica dell'informazione

- Alcuni codici usati quotidianamente:
  - la scrittura
  - le lingue
  - i gesti
  - la rappresentazione dei numeri
- Utilizzare gli stessi codici permette la COMUNICAZIONE (dell'informazione)
- COMUNICARE permette di:
  - descrivere oggetti, idee, percezioni, emozioni
  - trasmettere conoscenza, esperienza

Comunicazione

Problema

- Un uomo atterra su un pianeta sconosciuto e incontra una diversa forma di vita.
  Come può far capire che non ha cattive intenzioni se non esiste alcun codice riconosciuto da entrambi ?
- *Per comunicare è necessario condividere gli stessi codici!*
- Due persone di provenienza diversa hanno lingue e gestualità diverse. Però trasmettono le proprie emozioni attraverso espressioni spontanee simili.
- I codici possono collocarsi a livelli diversi.

Rappresentazione dell'informazione

- La codifica dell'informazione può avere due forme:
  - **Analogica**
  - **Digitale**
- Nella forma *analogica* una grandezza è rappresentata in modo continuo da un'altra grandezza continua (es. una tensione elettrica).

Es.
suono -> *microfono* -> tensione
tensione -> *altoparlante* -> suono

- La tensione prodotta dal microfono è tanto più alta quanto più elevato è il livello del suono.
- La vibrazione dell'altoparlante è tanto più ampia quanto più elevata è la tensione

Rappresentazione dell'informazione

- Nella forma *digitale* una grandezza è rappresentata in modo *discreto* da una sequenza di *campioni* (interpretabili come numeri interi).
- Un campione può rappresentare:
  - il livello di colore di un pixel (punto colorato)
  - l'ampiezza di un suono in un certo istante
  - un carattere
  - un numero!
- La rappresentazione digitale usata all'interno di un calcolatore:
  - è una approssimazione della realtà (continua).
  - l'errore di approssimazione dipende dalla precisione (numero di cifre a disposizione) della rappresentazione digitale.

Rappresentazione dell'informazione

- La rappresentazione digitale della realtà è una rappresentazione basata su numeri (digit = cifra, quindi digitale = numerico) che necessita di un CODICE per poterli rappresentare.
- I circuiti di un calcolatore lavorano a due diversi livelli di tensione (di solito 0 e 5 Volt, ma anche 0 e 3.3 Volt)
- E' possibile usare i 2 livelli per rappresentare due SIMBOLI diversi, a cui associare due diversi significati

Es. Vero/Falso Positivo/Negativo Presenza/Assenza
Ma anche le *quantità* (cifre) **0/1**

- All'interno del calcolatore i numeri vengono rappresentati con 2 cifre (rappresentazione **binaria**)

Codifica binaria

- Rappresentazione di numeri
- Notazione di tipo **posizionale** (come la notazione decimale).
- Ogni numero è rappresentato da una sequenza di simboli
- Il valore del numero dipende non solo dalla quantità rappresentata da ciascun simbolo, ma anche dalla posizione in cui si trovano i simboli.

3456 è diverso da **6543**

1001 è diverso da **1100**

- Utilizza solo 2 simboli (0 e 1)

Notazione posizionale (caratteristiche generali)

- Scelta una **base** di rappresentazione B
  - ogni numero è rappresentato da una sequenza di simboli (**cifre**) appartenente a un alfabeto di **B simboli** distinti
  - ogni cifra rappresenta un valore compreso **fra 0 e B-1**
  - a ogni posizione corrisponde un **peso**, uguale ad una **potenza della base** crescente da dx a sx.
  - Valore del numero = somma dei prodotti di ciascuna cifra per il peso associato alla sua posizione

Esempio di rappresentazione su N cifre:

$$d_{N-1}\, d_{N-2} \ldots d_1\, d_0 = d_{N-1} * B^{N-1} + d_{N-2} * B^{N-2} + \ldots + d_1 * B^1 + d_0 * B^0$$

Cifra più significativa — Cifra meno significativa

Rappresentazioni medianti basi diverse

- Decimale (B=10)
- Binaria (B=2)
  - Un calcolatore rappresenta l'informazione attraverso la codifica binaria.
  - Ogni elemento di una sequenza binaria viene detto **bit** (**B**inary dig**IT**).
  - Una sequenza di 8 bit si dice **byte**.
- Ottale (B=8)
- Esadecimale (B=16)

Esempi

$(102)_{10} = 1*10^2 + 0*10^1 + 2*10^0 = 100 + 0 + 2$

$(1100110)_2 = 1*2^6+1*2^5+0*2^4+0*2^3+1*2^2+1*2^1+0*2^0$
$= 64 + 32 + 0 + 0 + 4 + 2 = (102)_{10}$

$(146)_8 = 1*8^2 + 4*8^1 + 6*8^0 = 64 + 32 + 6 = (102)_{10}$

Rappresentazioni mediante basi diverse

- Quanto più è piccola la base tanto più lunga sarà la rappresentazione di una stessa quantità.

Es. $(109)_{10} = (1101101)_2 = (6D)_{16}$

- Qualunque sia la notazione, il valore della base B è codificato con la sequenza 10
  Infatti: $10 = 1 * B^1$ $(+ 0 * B^0)$

- Con sequenze di N simboli in base B posso rappresentare B^N numeri diversi

Multipli del byte

1 Byte = 8 bit

1 KiloByte (kB) = 1024 byte ($2^{10} = 1024$)

1 MegaByte (MB) = 1024 KB = 2^{20} Byte

1 GigaByte (GB) = 1024 MB = 2^{30} Byte

1 TeraByte (TB) = 1024 GB = 2^{40} Byte

Conversione da una base ad un'altra

La regola per convertire un numero da una base B (es. B=10) ad una diversa base b è (es. b=2):

1. Dividere il numero (decimale) per b (due) e annotare quoziente e resto.
2. Continuare a dividere per b il quoziente finché non diventa uguale a zero.
3. Assegnare i resti ottenuti come valore delle cifre per la codifica, partendo dall'ultimo.

Numero in base B **Base b**

Quozienti	resti
35	2
17	1
8	1
4	0
2	0
1	0
0	1

Resti

$(35)_{10} = (100011)_2$

Conversione binario - ottale/esadecimale

- Nella rappresentazione ottale (B=8) si usano gli 8 simboli *0, 1, 2, 3, 4, 5, 6, 7*
- In quella esadecimale (B=16) i 16 simboli *0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F*
- Con 3 bit si rappresentano $2^3 = 8$ simboli
- Con 4 bit si rappresentano $2^4 = 16$ simboli
- Dato un numero binario, la corrispondente codifica ottale si ottiene considerando gruppi di 3 bit (a partire dalla cifra meno significativa) e sostituendo al valore rappresentato da tali bitla cifra ottale equivalente
- Per convertire in esadecimale si fa la stessa cosa considerando gruppi di 4 bit

Conversione binario - ottale/esadecimale

Esempio:

Dato il numero binario 1100011001011011

la corrispondente codifica ottale è
1/100/011/001/011/011 = 1 4 3 1 3 3

La corrispondente codifica esadecimale è
1100/0110/0101/1011 = C 6 5 B

Rappresentazione di Numeri
Interi Positivi (numeri naturali)

- Un calcolatore assegna un numero fisso N di bit per ogni tipo di dato. N è di solito multiplo di 8.
- Con N bit si possono rappresentare 2^N valori distinti (es. con 8 bit, 2^8=256), associabili, ad esempio, ai numri naturali da 0 a 2^N -1.

Rappresentazione di Numeri Naturali

- **Dati N bit, quali numeri naturali si possono rappresentare?**

I numeri da 0 a $2^N - 1$

- **Dato un numero naturale, quanti bit sono necessari per rappresentarlo ?**

  Per rappresentare un numero naturale I con N bit è necessario che $2^N > I$, cioè $N > \log_2 I$

Es. I = 90 $N > \log_2 90$ $N > 6.492$

  Quindi N = 7. Infatti con 7 bit si possono rappresentare i numeri da 0 a 127

Somma

0 + 0 = 0 1 + 0 = 1

0 + 1 = 1 1 + 1 = 10

Somma e sottrazione si eseguono esattamente come nel caso decimale, sulla base dei 4 casi riportati a fianco.

Es. 22 + 21

1		1	← riporto			
	1	0	1	1	0 +	
	1	0	1	0	1 =	
1	0	1	0	1	1	

Rappresentazione di Numeri
Numeri relativi (interi con segno)

- E' possibile estendere in modo naturale la rappresentazione dei numeri naturali ai numeri relativi.
- I numeri relativi possono essere:
  - positivi (segno +)
  - negativi (segno -)
- Il segno può assumere 2 valori

  => Basta aggiungere 1 bit per rappresentare il segno.

Rappresentazione di numeri interi relativi (con N bit)

Prima rappresentazione possibile:

- Segno e Valore Assoluto
  - Il primo bit indica il segno (0=positivo, 1=negativo), cui segue la rappresentazione (su N-1 bit) del valore assoluto del numero, cioè del numero naturale che si ottiene eliminando il segno.

Es. (con 8 bit)

10000011 = - 3

Rappresentazione di numeri interi relativi (con N bit)

Rappresentazione alternativa :

- Complemento a due
  - Un numero negativo **-n** è rappresentato attraverso il **complemento a 2 del suo valore assoluto**, cioè dal numero intero positivo 2^N - n.

Es. N=8 2^N=256

$(6)_{10} = (00000110)_2$

$(-6)_{10} = (11111010)_2$ [Infatti: 256 - 6 = 250]

Osservazioni

- I numeri naturali e i corrispondenti numeri relativi positivi hanno la stessa rappresentazione e hanno come cifra più significativa 0

  $(35)_{10} = (00100011)_2$ in tutte le rappresentazioni

- Tutti i numeri negativi hanno 1 come cifra più significativa (come nella rappresentazione in segno e valore assoluto)

  $(-35)_{10} = (10100011)_2$ rappr. segno + val. ass.

  $(11111010)_2$ rappr. mediante compl. a 2

- Lo zero ha un'unica rappresentazione

- E' possibile realizzare la proprietà dei numeri relativi per cui m – n = m + (-n)

Calcolo rapido del complemento a 2

- Il complemento a 2 di un numero in una rappresentazione ad N bit si definisce come:

$$C(n) = 2^N - n$$

- Possiamo scrivere anche $C(n) = 2^N - n - 1 + 1$

N.B. !!

Dati n (ad es. 00100011) ed N (ad es. 8)

la rappresentazione di $2^N - 1 - n$ **è uguale alla rappresentazione di n con le cifre invertite** (11011100)

Allora: **il complemento di un numero si calcola invertendo tutti i suoi bit e sommando 1**

E quindi………..

- Per calcolare il complemento a 2 di un numero
  1. Si rappresenta il numero in binario
  2. Si invertono tutte le cifre (1 ->0 e 0 -> 1)
  3. Si somma 1.

Es.

32 = 00100000

- 32 = 11011111 + 1 = 11100000

N.B.

Il complemento del complemento di un numero è il numero stesso

C (32) = 256 - 32 = 224; (uso 8 bit)

C (C (32)) = 256 - (256 -32) = 32

Esempi di rappresentazioni

- Avendo a disposizione un byte per la rappresentazione, il numero naturale 35 ha la seguente rappresentazione binaria:

  00100011

  Il numero –35 in segno e valore assoluto:

  10100011

  Il numero –35 in complemento a due:

rappr. di 35	00100011
scambio 0<->1	11011100
aggiungo 1	$\boxed{11011101}$

Rappresentazione dello 0

- **modulo e segno**
  - rappresentazione ambigua
  - +0 = 00000000
  - \- 0 = 10000000
- **complemento a due**
  - rappresentazione univoca infatti il complemento a due di 00000000 è ancora 0 (*primo vantaggio*!)

Quindi:

in una rappresentazione a N bit con complemento a 2 posso rappresentare i numeri da $-(2^{N-1})$ **a** $+2^{N-1}-1$

Es. con 8 bit rappresento i numeri da

$-128 = -(2^7)$ a $127 = (2^7-1)$

Addizione

Se si utilizza la notazione con complemento a 2, si può ragionare in termini "algebrici":

Es. 22 – 21

Rappresento -21 in complemento a 2

```
  1 1 1  1 1 1
  0 0 0 1 0 1 1 0
  1 1 1 0 1 0 1 1
  ---------------
  0 0 0 0 0 0 0 1
```

La sottrazione equivale alla somma del minuendo con la negazione (rappresentazione complementata) del sottraendo.

Secondo vantaggio della rappresentazione in complemento a 2: **all'interno del calcolatore somme e sottrazioni sono eseguite utilizzando la stessa operazione (circuito).**

Esercizi

- Verificare che, in una rappresentazione senza segno, $(10000101)_2 = (133)_{10}$
- Data una rappresentazione intera a 4 bit senza segno, rappresentare e calcolare in binario le seguenti operazioni (decimali) :

  5 + 5

  13 + 5 (che risultato "apparente" ottengo ???)
- Data una rappresentazione a 8 bit in complemento a 2
  - determinare il minimo numero rappresentabile
  - determinare il massimo numero rappresentabile

Rappresentazioni a lunghezza fissa: problemi

Usare rappresentazioni di lunghezza fissa porta ad avere valori non rappresentabili (solo un certo numero di configurazioni è disponibile). Questo può provocare errori di due tipi:

Overflow

Underflow

Rappresentazioni a lunghezza fissa: problemi

- **Overflow** indica un errore nella rappresentazione di un certo numero (di solito il risultato di una operazione) dovuto al fatto che la quantità di cifre disponibili è minore rispetto a quelle necessarie a rappresentare il numero.

Es. i due addendi di una somma possono essere rappresentabili ma il risultato della somma no.

Es. (interi senza segno su 8 bit) 130 + 150 =280
Ma con 8 bit rappresento al max. il numero 255.

- **Underflow** indica che il risultato è troppo piccolo per essere rappresentato, cioè minore del più piccolo numero rappresentabile.

Es. divisione fra interi quando il dividendo è minore del divisore.
Il risultato in questo caso è 0.
25/50 = 0.5, ma io posso rappresentare solo 0, 1, …

Esempio 1 (overflow)

- Con 8 bit posso rappresentare:

interi positivi da 0 a 255

interi con segno da -128 a +127

- Supponiamo di essere nel primo caso e di avere 11111111 = $(255)_{10}$

```
255 + 1 =   ?        1 1111111
                       11111111+
                       00000001
                       --------------
                     1 00000000
```

- Ma ho solo 8 bit => il risultato dell'operazione è 0!

Esempio 2 (overflow)

- Con 8 bit posso rappresentare:

interi positivi da 0 a 255

interi con segno da -128 a +127

- Supponiamo di essere nel secondo caso e di avere 01111111 = $(127)_{10}$

127 + 1 = ????

```
 1111111
01111111+
00000001
---------
10000000
```

Ma 10000000 è negativo! (-128)

Rappresentazione di Numeri Reali

- Un numero reale è una grandezza continua
- Può assumere infiniti valori
- In una rappresentazione di lunghezza limitata, deve di solito essere approssimato.
- Esistono due forme per rappresentare un numero reale
  - Segno, parte intera, parte decimale (rappresentazione in virgola fissa)
  - Segno, mantissa, esponente (rappresentazione in virgola mobile)

Rappresentazione in virgola fissa (fixed-point)

- Utilizzando una rappresentazione su N cifre si assume che la posizione della virgola sia fissa in un preciso punto all'interno della sequenza. Quindi si assegnano k cifre per la parte intera e N-k cifre per la parte inferiore all'unità (che si può impropriamente definire *parte decimale*)

Es. con 8 cifre e 3 cifre 'decimali'

in base 10	123.45	00123.450
in base 2	111.1	00111.100

Rappresentazione in virgola fissa (fixed-point)

NB In una rappresentazione di tipo posizionale le cifre alla destra della virgola vengono moltiplicate per potenze negative della base

Es.

$(5.75)_{10} = 5*10^{0} + 7*10^{-1} + 5*10^{-2}$

$(11.011)_{2} = 1*2^{1} + 1*2^{0} + 0*2^{-1} + 1*2^{-2} + 1*2^{-3}$

Si ricordi che: $N^{-k} = 1 / N^{k}$

Conversione decimale-binario di numeri non interi

- La parte intera si converte col metodo delle divisioni successive
- Per la parte alla destra della virgola:
  - si moltiplica la parte a destra della virgola per 2
  - si prende la parte intera del risultato come cifra utile
  - si ripetono i 2 passi precedenti finché la parte a destra della virgola non diventa 0

Es. 19.375 Sappiamo che $(19)_{10} = (10011)_{2}$

0.375 * 2 = **0**.75

0.75 * 2 = **1**.5

0.5 * 2 = **1**.0

Quindi $(19.375)_{10} = (10011.011)_{2}$

Rappresentazione in virgola mobile (floating-point)

- Se ho una rappresentazione in virgola fissa (es. su segno e 8 cifre con 3 cifre alla destra della virgola) rappresento numeri (base 10) compresi fra

-99999.999 e 99999.999

Non posso rappresentare, quindi:

- numeri che richiedono più di 5 cifre intere, cioè maggiori di 99999.999 (es. 1000000)
- numeri che richiedono più di 3 cifre alla destra della virgola, come ad es. 123.0001

Rappresentazione in virgola mobile (floating-point)

- Utilizza la cosiddetta *notazione scientifica* (*esponenziale*)
- Nel formato esponenziale un numero N viene espresso nella seguente forma

$$N = \pm m \cdot b^{e}$$

- *b* base del sistema di numerazione
- *m* mantissa del numero
- *e* esponente (intero con segno)

Rappresentazione in virgola mobile (floating-point)

$$N = \pm m \cdot b^{e}$$

- Fissata la base, per rappresentare un numero reale è necessario rappresentare *segno, mantissa* ed *esponente*. La mantissa si suppone in virgola fissa con la virgola all'inizio, seguita sempre da una cifra diversa da zero.

Es (base 10) 523.45 = $.52345 * 10^{3}$

(base 2) 11.1011 = $.111011 * 2^{2}$

- **Ricorda:** *moltiplicare (dividere) un numero per una potenza della base equivale a far scorrere a sinistra (destra) il numero di un numero di posizioni pari all'esponente, ovvero a spostare la virgola di un uguale numero di posizioni in senso opposto.*

Rappresentazione in virgola mobile (floating-point)

- Permette di manipolare numeri con ordini di grandezza molto differenti utilizzando per la rappresentazione un insieme limitato di cifre: con N cifre a disposizione per la mantissa (più un certo numero per l'esponente) posso rappresentare in modo esatto i numeri che richiedono meno di N cifre fra la cifra più significativa e quella meno significativa.
- Il numero di cifre usate per l'esponente determina di quante posizioni posso spostare la virgola rispetto alla posizione 'standard' (a sinistra della cifra più significativa)

Approssimazioni nelle operazioni in floating point

- Quando il numero di cifre necessarie per una rappresentazione corretta del risultato di una operazione è maggiore del numero di cifre a disposizione, il numero viene *troncato*: si trascurano cioè le cifre meno significative per le quali "non c'è spazio".

Es. Supponendo di usare 4 bit per la mantissa

13 + 0.5

13 = $.1101 * 2^4$ 0.5 = $0.1 * 2^0$

Il risultato sarebbe $0.11011 * 2^4$, ma ho solo 4 bit

Quindi il risultato è $0.1101 * 2^4$, e quindi 13+0.5=13!

Esercizi

- In una rappresentazione binaria in virgola fissa con 6 bit per la parte intera e 4 per la parte 'decimale':
  1. Posso rappresentare il numero (in base 10) 16^{-1} ?
  2. Quale è il massimo intero che posso rappresentare ?
- In una rappresentazione in virgola mobile con 8 cifre di mantissa e 3 di esponente:
  1. Posso rappresentare senza approssimarlo il numero (in base 10) 1.03125017 ?
  2. Come rappresento il numero 122.625 ?

Algebra di Boole

- L'algebra di Boole è un formalismo che opera su variabili (dette *variabili booleane* o *variabili logiche* o *asserzioni*) che possono assumere due soli valori:
  - Vero
  - Falso
- L'algebra booleana nasce come tentativo di definire in forma algebrica processi di tipo logico-deduttivo
- Tuttavia, poiché di fatto l'algebra di Boole opera su variabili binarie (*vero* e *falso* sono i 2 soli simboli), i suoi operatori possono essere inclusi fra gli operatori dell'algebra binaria.

Algebra di Boole

- Sulle variabili booleane è possibile definire delle funzioni (dette funzioni booleane o logiche). Anch'esse possono assumere i due soli valori vero e falso.
- Le funzioni booleane possono essere definite tramite le *tabelle di verità*. Una tabella di verità di una funzione di N variabili ha 2^N righe, una per ogni possibile combinazione delle variabili, e N+1 colonne, N per rappresentare la combinazione delle variabili più una per il valore corrispondente della funzione

X_1	X_2	X_3	**F**
0	0	0	**1**
1	0	0	**0**
0	1	0	**0**
0	0	1	**0**
1	1	0	**1**
0	1	1	**1**
1	0	1	**1**
1	1	1	**1**

Operatori ed Espressioni Booleane

- L'algebra di Boole si basa su un insieme di operatori:
  - **AND** (indicato in genere dal simbolo ×)
  - **OR** (indicato in genere dal simbolo +)
  - **NOT** (indicato in genere dal simbolo ‾)
  - **XOR** (indicato in genere dal simbolo ⊕)
  - **NAND** (indicato in genere dal simbolo ↑)
  - **NOR** (indicato in genere dal simbolo ↓)
- In realtà, qualunque funzione booleana può essere realizzata utilizzando 2 soli operatori: AND e NOT oppure OR e NOT

NOT - AND - OR

X	**NOT**
0	**1**
1	**0**

Il risultato è la negazione della variabile

X_1	X_2	**AND**
0	0	**0**
1	0	**0**
0	1	**0**
1	1	**1**

Il risultato è 1 (Vero) se entrambe le variabili hanno valore 1

X_1	X_2	**OR**
0	0	**0**
1	0	**1**
0	1	**1**
1	1	**1**

Il risultato è 1 (Vero) se almeno una delle variabili ha valore 1

XOR - NAND - NOR

X_1	X_2	**XOR**
0	0	**0**
1	0	**1**
0	1	**1**
1	1	**0**

Il risultato è 1 (Vero) se una sola delle due variabili ha valore 1

X_1	X_2	**NAND**
0	0	**1**
1	0	**1**
0	1	**1**
1	1	**0**

NAND (X_1, X_2) = NOT (AND (X_1,X_2))

NOR (X_1, X_2) = NOT (OR (X_1, X_2))

X_1	X_2	**NOR**
0	0	**1**
1	0	**0**
0	1	**0**
1	1	**0**

Interpretazione logica degli operatori

- Se si ha una operazione del tipo:

A * B (* indica una generica operazione),

il risultato è vero se:

*	condizione
OR	A o B (o entrambe) sono vere
AND	sia A che B sono vere
XOR	A o B (ma non entrambe) sono vere

Operatori ed Espressioni Booleane

- Questi operatori possono essere combinati in espressioni booleane che rappresentano funzioni booleane e si compongono con le stesse regole utilizzate per l'algebra tradizionale.

$F(x_1, x_2, x_3)= ((\text{NOT } x_1) \text{ AND } x_2) \text{ OR } x_3$

Esercizio di logica booleana

- Ricavare la tavola di verità della seguente funzione booleana

F(a,b,c)=(a AND b) OR (NOT c)

Espressioni equivalenti

- Due espressioni si dicono *equivalenti* quando per ogni combinazioni di valori delle variabili assumono lo stesso risultato

Esempio

- $a \oplus b$
- $\overline{a} \times b + a \times \overline{b}$

Espressioni complementari

- T1 e T2 sono *complementari* se per quelle combinazioni in cui T1 risulta 1 T2 risulta 0 e viceversa

Esempio

- $t1=(\overline{a} \times \overline{c}) + (a \times b)$
- $t2=(\overline{a} \times c) + (a \times \overline{b})$

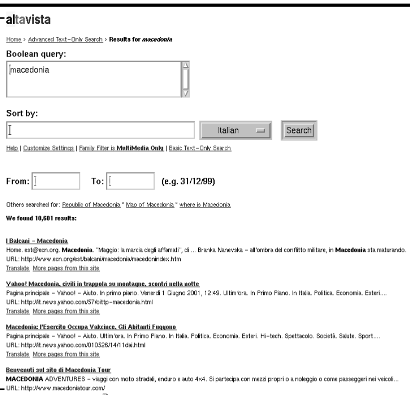
Espressioni duali

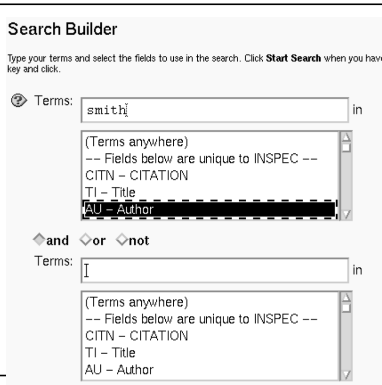
- T2 è detta *duale* di T1 se è ottenuta da essa sostituendo l'operatore AND con l'OR e viceversa, e la costante 0 con 1 e viceversa
  - $T1 = (\bar{a} \times \bar{c}) + (a \times b) + 0$
  - $T2 = (\bar{a} + \bar{c}) \times (a + b) \times 1$

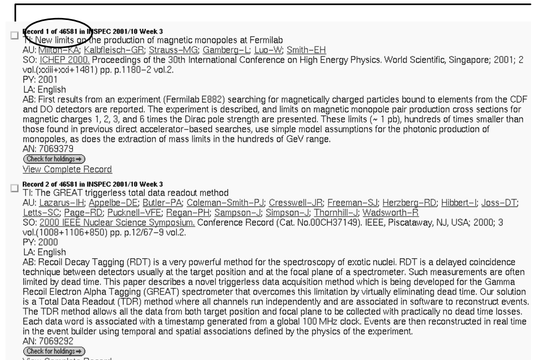
Esercizio di logica booleana

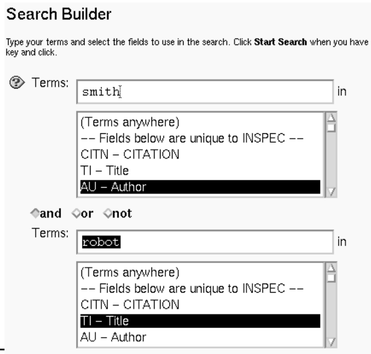
- Una cassaforte ha 4 lucchetti x,y,v,w che devono essere aperti tutti. Le chiavi sono distribuite fra 3 persone A,B,C.
  - A possiede le chiavi v e y
  - B possiede le chiavi v ed x
  - C possiede le chiavi w e y

Dire alla presenza di quali combinazioni di persone la cassaforte può essere aperta e costruire la tavola di verità della corrispondente funzione booleana









Record 1 of 49 in INSPEC 2001/01-2001/07
TI: **Robot** space exploration by trial and error
AU: Jakobi-N; Husbands-P; Smith-T
SO: Genetic Programming 1998. Proceedings of the Third Annual Conference. Morgan Kaufmann Publishers, San Francisco, CA, USA; 1998; xxv+892 pp. p.807-15.
PY: 1998
LA: English
AB: We argue that evolutionary robotics (ER) techniques can act as useful and potentially wide ranging tools in the scientific investigation of adaptive behaviour. After discussing the kinds of investigations ER can play a central role in, a concrete example is presented. We conclude that these kinds of studies are not only scientifically useful, but are necessary for the field to develop as an engineering methodology for autonomous robotics.
AN: 6976975
Check for holdings
View Complete Record

Record 2 of 49 in INSPEC 2001/01-2001/07
TI: Classification of textured surfaces for **robot** navigation using continuous transmission frequency-modulated sonar signatures
AU: Politis-Z; Probert-Smith-PJ
SO: International-Journal-of-Robotics-Research. vol.20, no.2; Feb. 2001; p.107-28.
PY: 2001
LA: English
AB: Whereas in the past ultrasonic sensors have been largely used only to estimate the proximity of objects and the location and identification of primitive targets in a robot workspace the development of biomimetic sonar has opened up new possibilities for their application. Broadband sonar echoes have sufficient resolution so that characteristics on reflection, especially geometry and texture, can be distinguished with only a few measurements. We describe how a model of texture can be used to distinguish between a number of different surfaces using only a single measurement of each, showing results on a number of surfaces that might be considered typical pathways for a mobile robot, both those with periodicity in pattern and those with statistically homogeneous features. In particular, we consider textures corresponding to hard smooth floors, carpets and asphalts, and surfaces with a repeating pattern made up of tiles. Each random rough surface is modeled using an extension of the Kirchhoff approximation method describing the scattering of the acoustic wave on the surface while the periodic surfaces are modeled assuming distinctive reflections from the tile borders. The continuous transmission frequency-modulated sonar signature corresponding to each class is derived and compared with the experimental measurement. A set of features is extracted that exploits the differences between the surface models, and a hierarchical classification scheme is proposed for recognition.
AN: 6854966
*LHM: Posseduto da: IF(134): 8(1989)-
Check for holdings
View Complete Record

Esercizi

- Verificare se le seguenti coppie di funzioni booleane sono equivalenti (cioè hanno la stessa tabella di verità):

C AND (A OR NOT B) e (NOT B OR A) AND C

NOT (C AND B) NOR A e A NOR (B NAND C)

C AND (NOT A OR B) e C NAND (A OR B)

A AND (B AND NOT C) e (NOT A) OR ((NOT C) NAND B)