INGEGNERIA PER LA SOSTENIBILITA' INDUSTRIALE - UnivPM

Fondamenti di Informatica

Prof. Ing. Loris Penserini, PhD

elpense@gmail.com

https://orcid.org/0009-0008-6157-0396

Materiale:

https://github.com/penserini/Lezioni_UnivPM.git

Codifica dell'Informazione (overview)

Rappresentazione delle Informazioni

L'informazione digitale all'interno dell'elaboratore è rappresentata in codifica binaria 0 e 1 (**bit**).

Le cifre binarie all'interno di un calcolatore vengono trattate a gruppi o pacchetti contenenti un numero costante di bit: in particolare, per essere elaborate, le cifre binarie vengono raggruppate in sequenze o stringhe di 8 bit. Una stringa di 8 bit prende il nome di **byte**.

Per il trattamento dei dati, gli elaboratori operano su sequenze composte da un numero fisso di byte. Tali stringhe di byte prendono il nome di **parole (word)**.

Rappresentazione dei numeri

Per i numeri decimali si usano le seguenti rappresentazioni:

Virgola fissa (fixed point)

Es. **1.8** **0.00347** **32.321**

Virgola mobile (floating point)

Es. **5E-3** **10E+3** **2.5E-4**

Quest'ultimo modo di rappresentazione si chiama notazione scientifica o rappresentazione esponenziale. La lettera **E** sta al posto di «10 elevato a» ed è seguita dall'esponente a cui elevare la base 10: la potenza di 10 va poi moltiplicata per il numero (**mantissa**) che precede la lettera E.

Per esempio: **2.5E-4 ➔ 2.5 x 10^{-4} = 0.00025**

Project Work

Fornire una rappresentazione esponenziale dei seguenti numeri:

0.00125

1.5455

77300000

99554433

Project Work – Possibile Soluzione

Fornire una rappresentazione esponenziale dei seguenti numeri:

0.00125 = 12.5E-4

1.5455 = 154.55E-2

77300000 = 773E+5

99554433 = 9955.4433E+4

Rappresentazione dei numeri in Memoria

La rappresentazione interna dei numeri in memoria RAM subisce delle limitazioni dovute alle dimensioni fisiche della cella di memoria.

Con il termine **precisione** della rappresentazione interna dei numeri si indica il numero di byte utilizzati per la rappresentazione dei numeri che può variare per diversi sistemi di elaborazione in commercio.

- **precisione semplice** (o precisione singola): quando i numeri reali sono rappresentati, per esempio, con 4 byte (cioè 32 bit) ,
- **precisione doppia**: quando i numeri reali sono rappresentati, per esempio, con 8 byte (cioè 64 bit).

Rappresentazione Alfanumerica

Le informazioni esprimibili mediante una combinazione di lettere, cifre o caratteri speciali devono avere una corrispondenza binaria affinché un elaboratore riesca a riconoscere e a trattare questo tipo di informazione digitale.

L'associazione di una combinazione binaria del byte ad un determinato simbolo (lettera, cifra o carattere speciale) e chiamata **codifica**.

La prima codifica nella storia dell'informatica si chiama **ASCII** (*American Standard Code for Information Interchange*), proposto dall'ingegnere dell'IBM, Bob Berner nel 1961, e fu poi pubblicato dall'*American National Standards Institute* nel 1963.

Codifica ASCII (estesa)

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`	128	80	Ç	160	A0	á	192	C0	└	224	E0	α
1	01	Start of heading	33	21	!	65	41	A	97	61	a	129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
2	02	Start of text	34	22	"	66	42	B	98	62	b	130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
3	03	End of text	35	23	#	67	43	C	99	63	c	131	83	â	163	A3	ú	195	C3	├	227	E3	π
4	04	End of transmit	36	24	$	68	44	D	100	64	d	132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
5	05	Enquiry	37	25	%	69	45	E	101	65	e	133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
6	06	Acknowledge	38	26	&	70	46	F	102	66	f	134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
7	07	Audible bell	39	27	'	71	47	G	103	67	g	135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
8	08	Backspace	40	28	(	72	48	H	104	68	h	136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i	137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j	138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k	139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l	140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m	141	8D	ì	173	AD	¡	205	CD	═	237	ED	ø
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n	142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o	143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
16	10	Data link escape	48	30	0	80	50	P	112	70	p	144	90	É	176	B0	░	208	D0	╨	240	F0	≡
17	11	Device control 1	49	31	1	81	51	Q	113	71	q	145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
18	12	Device control 2	50	32	2	82	52	R	114	72	r	146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
19	13	Device control 3	51	33	3	83	53	S	115	73	s	147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
20	14	Device control 4	52	34	4	84	54	T	116	74	t	148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u	149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v	150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
23	17	End trans. block	55	37	7	87	57	W	119	77	w	151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
24	18	Cancel	56	38	8	88	58	X	120	78	x	152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
25	19	End of medium	57	39	9	89	59	Y	121	79	y	153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z	154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{	155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|	156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}	157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~	158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	⌂	159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

La nuova codifica Unicode

Così come avvenne per altri standard dell'informatica (es. TCP/IP) anche la codifica ASCII estesa (8bit) risultò alla fine degli anni '80 non più sufficiente a codificare i simboli di tutte le lingue…

Nel 1991 nacque la codifica **Unicode** (estensione dell'ASCII) che a sua volta ebbe delle evoluzioni: UTF-8 (8bit), UTF-16 (16bit) e UTF-32 (32 bit).

L'obiettivo generale di Unicode è di creare una codifica che comprenda tutti i caratteri, con tutte le variazioni possibili, di tutte le lingue esistenti, oltre ai simboli utilizzati in matematica e nelle scienze.

Sia internamente ad un elaboratore, sia in ambiente distribuito, ogni carattere viene convertito in bit per essere trasmesso. Più bit vengono utilizzati, più caratteri differenti possono essere utilizzati e più lunga sarà la sequenza di bit da trasmettere.

Le tabelle dei codici Unicode sono disponibili sul sito
http://www.unicode.org/charts

Project Work

Prendiamo in considerazione la stringa di testo «**Ciao, mondo!**» contenente 12 caratteri (si considerano anche spazi e punto esclamativo).

Calcolare quanto spazio di memoria occuperebbe la stessa stringa di testo nelle codifiche ASCII standard e UTF-32.

Project Work - soluzione

Prendiamo in considerazione la stringa di testo «**Ciao, mondo!**» contenente 12 caratteri (si considerano anche spazi e punto esclamativo).

Calcolare quanto spazio di memoria occuperebbe la stessa stringa di testo nelle codifiche ASCII standard e UTF-32.

Soluzione

ASCII standard => 12 car X 7 bit = 84 bit

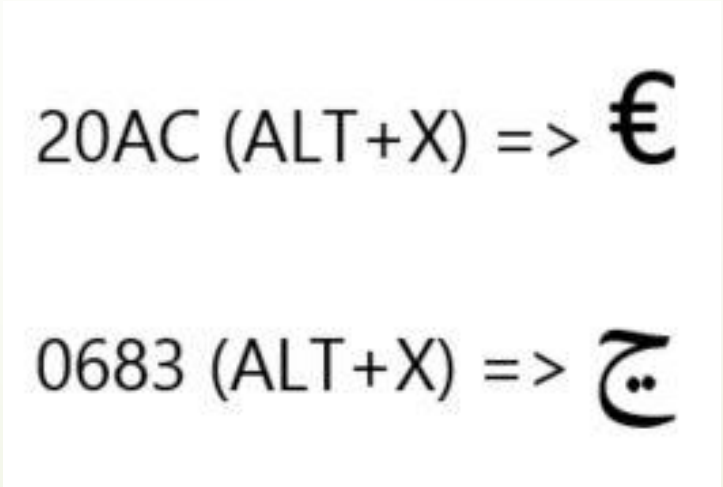
UTF-32 => 12 car X 32 bit = 384 bit

Cioè la codifica UTF-32 richiede più di 4 volte lo spazio necessario alla codifica ASCII standard.

Project Work - soluzione

Utilizzando le tabelle di codifica UTF disponibili in rete, aprite una pagina vuota di MS-Word e generate i simboli desiderati nel seguente modo:

«codice esadecimale» ALT+X



Logica delle Proposizioni (overview)

Logica delle Proposizioni

Nella programmazione si usa molto spesso la logica delle proposizioni o **algebra delle proposizioni** le cui fondamenta sono l'algebra booleana dal nome del matematico inglese George Boole (1815-1864). La **Logica Proposizionale** costituisce la base di molti linguaggi formali usati nell'Intelligenza Artificiale.

[Panti et al., 2001] [Aldewereld et al., 2008] [Morandini et al., 2009]

Un **enunciato** rappresenta una proposizione che può essere vera o falsa, ma non entrambe le cose. La verità o falsità di un enunciato è anche detta **valore di verità**.

La logica proposizionale è il linguaggio (con proposizioni e connettivi), mentre l'algebra booleana è la struttura matematica che gli dà fondamento e permette di trattare proposizioni come variabili numeriche (0/1).

ESEMPI

Proposizione

«Oggi c'è il sole!», «Domani si parte» => sono enunciati

«Speriamo che sia promosso», «come è andato il viaggio?» => non sono enunciati perché non sono né veri né falsi.

Proposizione composta

Supponiamo:

- p = "È giorno"
- q = "È soleggiato"

Una proposizione composta può essere:

p ∧ q (p AND q)

"È giorno **e** è soleggiato"

In **logica proposizionale**, questo è un enunciato che è vero solo se p e q sono entrambe vere.

Nel modello dell'algebra booleana, p e q sono variabili booleane con valori {0,1}, e l'operazione ∧ è l'AND booleana.

Operatori di Confronto

Gli operatori di relazione (o di confronto) permettono di esprimere una proposizione atomica (condizione) confrontando due operandi. I più noti sono quelli che confrontano insiemi numerici:

- uguale (simbolo '=' o '==')
- diverso (simbolo '≠' o '<>' o '!=')
- maggiore (simbolo '>')
- minore (simbolo '<')
- maggiore o uguale (simbolo '≥')
- minore o uguale (simbolo '≤')

Il risultato di un confronto assume valore **Vero** o **Falso** (ovvero, un valore logico) pertanto lo possiamo comporre in predicati:

- 5 < 3 ? Falso
- -1 > -5 ? Vero

Proposizioni Composte

In alcuni casi, gli enunciati possono essere composti, cioè formati da sottoenunciati collegati tra loro da **connettivi logici**. Le proposizioni composte sono espressioni logiche dette anche funzioni booleane.

Esempio

«Domani si parte oppure si resta a casa» => è un enunciato composto da due sottoenunciati:

- **p** = «Domani si parte»
- **q** = «Domani si resta a casa»,

collegati tra loro dal connettivo «oppure»

- **p OR q**

Operatori/Connettivi Logici

- **Oggetto della logica**
  - La logica studia **il legame tra premesse e conclusioni**, cioè se una conclusione segue necessariamente dalle premesse.
  - La logica non stabilisce **se le premesse siano vere o false** (quello è compito, ad esempio, della scienza, dell'esperienza o di altre discipline).
- La logica delle proposizioni tratta questi enunciati come **atomi logici** e li combina con operatori o connettivi:
  - ¬ (negazione) => "non" , **NOT**
  - ∧ (congiunzione) => "e" , **AND**
  - ∨ (disgiunzione) => "o" , **OR**
  - ⊕ (or-esclusivo) => **XOR**
  - → (implicazione) => "se… allora…"
  - ↔ (doppia implicazione) => "se e solo se"

Proposizioni Composte

- Il valore di verità di una proposizione composta dipende dal valore di verità delle proposizioni atomiche che la compongono.
- I connettivi logici in quanto operatori sono funzioni che associano ad ogni valore di verità un valore corrispondente.
- Una proposizione composta (o funzione booleana) è esprimibile (e valutabile) mediante tabelle di verità.
- Per un enunciato composto il valore di verità è definito dai valori di verità dei suoi sottoenunciati e dal connettivo logico che li unisce.

Esempi di Proposizioni Composte

Proposizione A: oggi è martedì

Proposizione B: oggi è il 2 novembre

A AND B (oggi è martedi 2 novembre)

NOT A AND B (oggi non è martedi ed è il 2 novembre)

A AND NOT B (oggi è martedi e non è il 2 novembre)

A XOR B (o oggi è martedi, o è il 2 novembre)

Tabella di Verità

In generale dunque una **tabella di verità** è la definizione tabellare di una funzione booleana in K variabili (in questo caso le variabili sono **proposizioni atomiche**)

- Per ogni possibile combinazione dei valori delle variabili deve essere specificato il valore assunto dalla funzione
- Le variabili possono assumere solo due valori (VERO o
- FALSO)
- Le possibili combinazioni sono 2^k

Inferenza

- **Un'inferenza è un passaggio da una o più premesse a una conclusione.**
  - Si dice valida (corretta) se, assumendo vere le premesse, la conclusione non può essere falsa.

Esempio:

- Premessa 1: Se piove, allora la strada è bagnata:
  - p = «piove» , q = «la strada è bagnata» , **p** → **q**
- Premessa 2: Piove
- Conclusione: La strada è bagnata
  → Questa inferenza è corretta

Implicazione: →

In alcuni casi, gli enunciati possono essere collegati dal connettivo «implica» (**p** → **q**) che in logica proposizionale significa che se **p** è vera allora **q** dev'essere vera. Quando **p** è falsa **q** può assumere qualsiasi valore (non possiamo dire nulla di q).

Se indichiamo con le lettere **p** e **q** i due enunciati, la tabella o valore di verità relativa alla loro congiunzione (**p** → **q**) risulta:

p	q	p → q
V	V	V
V	F	F
F	V	V
F	F	V

Dove «V» e «F» sono i valori di verità, «Vero» e «Falso».

Esempio con Implicazione

p = «oggi piove»

q = «la strada è bagnata»

Quando piove sicuramente la strada è bagnata, quindi non può essere che piova e non sia bagnata!

Non piove → la frase non viene messa alla prova, quindi resta "vera per default".

p	q	p → q
V	V	V
V	F	F
F	V	V
F	F	V

Non piove → anche se la strada non è bagnata, la frase non è smentita. Rimane "vera per default".

Doppia Implicazione: ↔

L'operatore logico **"se e solo se"** (in latino *bi-implicazione*) collega due proposizioni p e q:

- **p ↔ q**
- si legge: "**p se e solo se q**"
- oppure: "**p è equivalente a q**".

Significa che **p** e **q** devono avere **lo stesso valore di verità**:

- entrambi veri
- entrambi falsi

In tutti gli altri casi, la frase è falsa.

p	q	p ↔ q
V	V	V
V	F	F
F	V	F
F	F	V

Congiunzione - AND

In alcuni casi, gli enunciati possono essere collegati dal connettivo «e» che in logica proposizionale è rappresentato da **AND** (forma inglese) e che viene chiamato **congiunzione**.

Se indichiamo con le lettere **p** e **q** i due enunciati, la tabella o valore di verità relativa alla loro congiunzione (**p AND q**) risulta:

p	q	p AND q
V	V	V
V	F	F
F	V	F
F	F	F

Dove «V» e «F» sono i valori di verità, «Vero» e «Falso».

Disgiunzione - OR

In alcuni casi, gli enunciati possono essere collegati dal connettivo «oppure» che in logica proposizionale è rappresentato da **OR** (forma inglese) e che viene chiamato **disgiunzione**.

Se indichiamo con le lettere **p** e **q** i due enunciati, la tabella o valore di verità relativa alla loro disgiunzione (**p OR q**) risulta:

p	q	p OR q
V	V	V
V	F	V
F	V	V
F	F	F

Dove «V» e «F» sono i valori di verità, «Vero» e «Falso».

Disgiunzione Esclusiva - XOR

In alcuni casi, gli enunciati possono essere collegati dal connettivo «o esclusivo» che in logica proposizionale è rappresentato da **XOR** (forma inglese) e che viene chiamato **disgiunzione esclusiva**.

Se indichiamo con le lettere **p** e **q** i due enunciati, la tabella o valore di verità relativa alla loro disgiunzione (**p XOR q**) risulta:

p	q	p XOR q
V	V	F
V	F	V
F	V	V
F	F	F

Dove «V» e «F» sono i valori di verità, «Vero» e «Falso».

OR – XOR differenze semantiche

Nella lingua italiana la particella «o» può assumere due significati diversi:

- «p o q o entrambi» => disgiunzione OR
- «p o q ma non entrambi» => disgiunzione esclusiva XOR

Per esempio:

- «ho vinto 100€ oppure ho vinto una bicicletta» => si intende anche che possono essere vere entrambe
- «ora sta piovendo oppure ora non sta piovendo» => si intende anche che **non possono essere vere (o false) entrambe**

Negazione - NOT

Per cui dato un enunciato **p** è possibile ricavare un altro enunciato dato dalla negazione del primo: **NOT p => negazione** di **p**

Nel linguaggio naturale siamo soliti dire «non è vero che …» oppure semplicemente anteponendo la parola «non» davanti all'enunciato.

p	NOT p
V	F
F	V

Dove «V» e «F» sono i valori di verità, «Vero» e «Falso».

Project Work

Utilizzando le tabelle di verità, rispondere alle seguenti domande:

- Se a = 3 e b = 5 l'espressione: (a < 2) AND (b > 7) produce una proposizione vera o falsa?
- Se a = 6 e b = 15 l'espressione: (a > 2) OR (b > 11) produce una proposizione vera o falsa?
- Se a = 6 e b = 15 l'espressione: (a > 2) XOR (b > 11) produce una proposizione vera o falsa?
- Se a = 6 e b = 15 l'espressione: (a > 2) AND (NOT(b < 11)) produce una proposizione vera o falsa?
- Per quali valori interi di q la seguente espressione è vera
  (q > 10) AND (q < 15)

Project Work - soluzione

Utilizzando le tabelle di verità, rispondere alle seguenti domande:

- Se a = 3 e b = 5 l'espressione: (a < 2) AND (b > 7) produce una proposizione vera o falsa? => **FALSA**
- Se a = 6 e b = 15 l'espressione: (a > 2) OR (b > 11) produce una proposizione vera o falsa? => **VERA**
- Se a = 6 e b = 15 l'espressione: (a > 2) XOR (b > 11) produce una proposizione vera o falsa? => **FALSA**
- Se a = 6 e b = 15 l'espressione: (a > 2) AND (NOT(b < 11)) produce una proposizione vera o falsa? => **VERA**
- Per quali valori interi di q la seguente espressione è vera
  (q > 10) AND (q < 15) => **q = {11,12,13,14}**

Spunti Bibliografici dell'Autore

[Morandini et al., 2017] Mirko Morandini, Loris Penserini, Anna Perini, Alessandro Marchetto: Engineering requirements for adaptive systems. Requirements Engineering Journal, 22(1): 77-103 (2017)

[Morandini et al., 2009] Morandini M., Penserini L., and Perini A. (2009b). Operational Semantics of Goal Models in Adaptive Agents. In 8th Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'09). IFAAMAS.

[Morandini et al., 2008] Morandini, M., Penserini, L., and Perini, A. (2008b). Automated mapping from goal models to self-adaptive systems. In Demo session at the 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008), pages 485–486.

[Aldewereld et al., 2008] Huib Aldewereld, Frank Dignum, Loris Penserini, Virginia Dignum: Norm Dynamics in Adaptive Organisations. NORMAS 2008: 1-15

[Penserini et al., 2007a] Loris Penserini, Anna Perini, Angelo Susi, John Mylopoulos: High variability design for software agents: Extending Tropos. ACM Trans. Auton. Adapt. Syst. 2(4): 16 (2007)

[Penserini et al., 2007b] Loris Penserini, Anna Perini, Angelo Susi, Mirko Morandini, John Mylopoulos: A design framework for generating BDI-agents from goal models. AAMAS 2007: 149

[Pagliarecci et al., 2007] Francesco Pagliarecci, Loris Penserini, Luca Spalazzi: From a Goal-Oriented Methodology to a BDI Agent Language: The Case of Tropos and Alan. OTM Workshops (1) 2007: 105-114

[Penserini et al., 2006a] Loris Penserini, Anna Perini, Angelo Susi, John Mylopoulos: From Stakeholder Intentions to Software Agent Implementations. CAiSE 2006: 465-479

[Penserini et al., 2006b] Loris Penserini, Anna Perini, Angelo Susi, John Mylopoulos: From Capability Specifications to Code for Multi-Agent Software. ASE 2006: 253-256

[Panti et al., 2003] Maurizio Panti, Loris Penserini, Luca Spalazzi: A critical discussion about an agent platform based on FIPA specification. SEBD 2000: 345-356

[Panti et al., 2001] Maurizio Panti, Luca Spalazzi, Loris Penserini: A Distributed Case-Based Query Rewriting. IJCAI 2001: 1005-1010

GRAZIE!