Riassunto

Capitolo 1 – Insiemi

1) Insiemi numerici

→ Insieme N

Primi numeri ad essere stati scoperti nella storia e che si imparano. Vengono utilizzati per effettuare semplici attività, come contare.

L'insieme N è un insieme inferiormente limitato (0) ma non limitato superiormente. Ogni valore numerico ha un successore.

N = {0, 1, 2, 3, 4, …, n, n+1, …}
N = [0, +∞)

I numeri naturali dell'insieme N hanno le seguenti proprietà:
1) 0 non è successore di nessun numero naturale;
2) 0 appartiene ai naturali;
3) ogni numero naturale n ha un successore s(n);
4) due numeri naturale a e b (a, b ∈ N) hanno due successori diversi (¬ (a = b) → ¬(s(a) = s(b)).

Principio di Induzione

Il principio di induzione afferma che una proprietà P è vera in N se e solo se:
1) è vera in 0;
2) se è vera in n, allora sarà vera anche in s(n).

→ Insieme Z

Tutti i valori che possono essere preceduti da un segno – o da un segno +.
Tutti i valori preceduti dal segno + o senza segno rappresentano valori positivi e tutti i valori preceduti dal segno – rappresentano valori negativi.

Ogni valore ha un predecessore e un successore ed è un insieme illimitato sia inferiormente che superiormente.

Z = {…, -n-1, -n, …, -3, -2, -1, 0, 1, 2, 3, …, n, n+1, …}
Z = (-∞, +∞)

Valore Assoluto

Il valore assoluto di a è il valore di a che non ha alcun segno (|a|).
0 non ha segno.

L'opposto di a è -a.
L'opposto di -a è a.

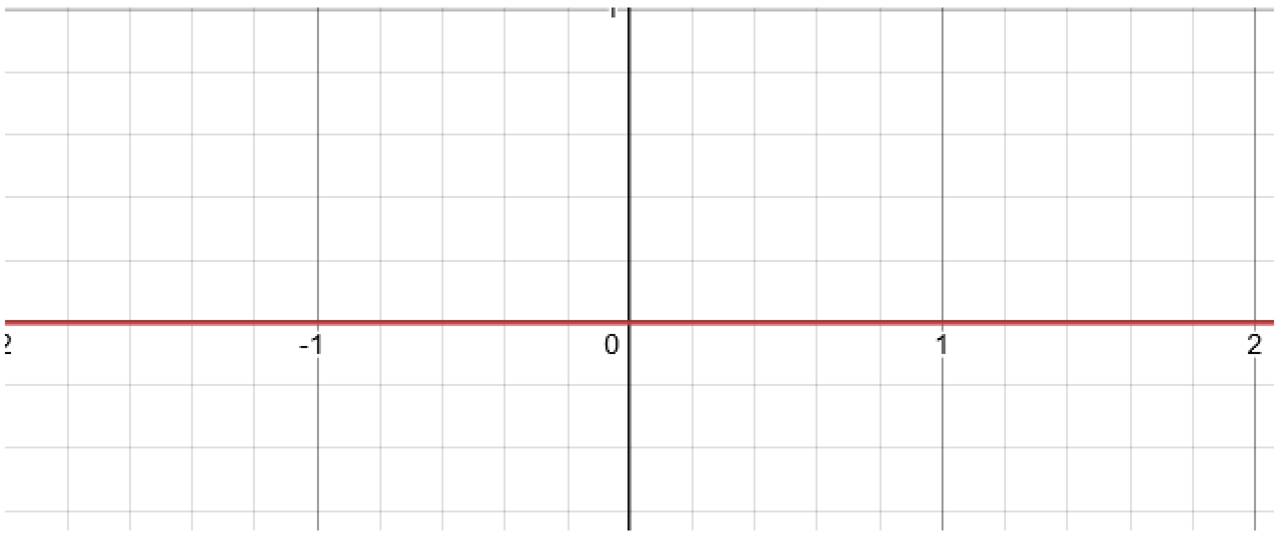
Rapporto tra N e Z
L'insieme N è un sottoinsieme di Q, mentre Q è un ampliamento dell'insieme N.
N < Q.

→ Insieme Q
I numeri razionali sono tutti quei valori che possono essere espressi mediante un rapporto tra due numeri a e b e possono essere negativi (<0) o positivi (>=0).
Anche i numeri interi appartenenti a Z sono razionali e questi sono contenuti in Q (N < Z < Q).
Definizione:
x Є Q = a/b

→ Insieme I
Sono tutti quei valori che non possono essere espressi mediante un rapporto tra due numeri.
Esempio:
√2, π.

→ Insieme R
L'insieme dei numeri reali contiene tutti i numeri razionali e irrazionali (R = Q U I) e possono essere rappresentati mediante una retta reale (o asse reale). Il punto viene rappresentato da una corrispondenza tra un elemento appartenente all'insieme delle ascisse e un elemento appartenente all'insieme delle coordinate.

R = Q U I



→ Insieme C (numeri complessi)

L'insieme dei numeri complessi è l'insieme sul quale è possibile effettuare operazioni non definite nell'insieme R, come l'estrazione della radice quadrata di un numero negativo.

Il valore appartenente a C viene così definito:

c = a + bi con a e b appartenenti a R.

2) Insiemi

Raggruppamento di oggetti distinti e ben definiti.

Esempio:

insieme delle lettere dell'alfabeto;

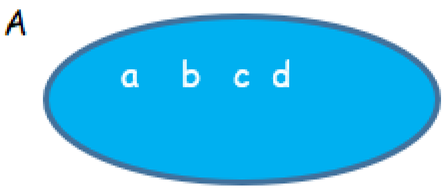
insieme delle vocali;

insieme dei tipi di carne.

→ Come rappresentare un insieme?

Un insieme può essere rappresentato in tre modalità:

1. Grafico Eulero - Venn



2. Rappresentazione Intensionale

L'insieme viene rappresentato mediante una descrizione (detta anche rappresentazione per caratteristica o proprietà).

A = {x | descrizione proprietà}

3. Rappresentazione Estensionale

Vengono elencate tra parentesi graffe tutti gli elementi appartenenti ad un insieme.

A = {a,b,c,d}

→ Simbologie

Gli insiemi vengono rappresentati da una lettera maiuscola (A, B, C, …).

Insieme (A, B, C, …, Z)

L'elemento viene rappresentato mediante una lettera minuscola (a, b, c, …)

Elemento (a, b, c, …, z)

L'appartenenza stabilisce se un determinato elemento appartiene ad un insieme e si indica nella seguente maniera:

a ∈ A → l'elemento a appartiene all'insieme A
a ∉ A → l'elemento a non appartiene all'insieme A

→ Appartenenza e Intensionale
Gli elementi di un insieme A devono soddisfare una proprietà P.
1) Se x ∈ A, allora x soddisfa P.
2) Se x ∉ A, allora x non soddisfa P.

→ Operazione vero – funzionale
L'operazione vero funzionale è utile per rappresentare intensionalmente un insieme attraverso una descrizione leggermente più complessa.
Esempio: si vuole rappresentare l'intervallo di numeri compresi tra 3 e 6 esclusi
A = {x | x > 3 and x < 6}

→ Quali sono le proprietà dell'uguaglianza?
L'uguaglianza tra oggetti (insiemi, elementi) è rappresentata mediante il simbolo =.
Essa ha tre proprietà

Uguaglianza	
Proprieta	Descrizione
Riflessiva	A = A
Simmetrica	A = B -> B = A
Transitiva	A = B , B = C -> A = C

→ Sottoinsiemi
Un sottoinsieme di un insieme dato A è l'insieme di tutti gli elementi appartenenti all'insieme dato A.
Il sottoinsieme B si dice proprio se diverso da 0 e da A.
L'insieme vuoto 0 ammette solo come sottoinsieme 0.
Il singoletto {a} ammette due sottoinsiemi: 0 e {a}

→ Notazione
⊆ → indica che un insieme è contenuto in un altro insieme.
Se A ⊆ B e B ⊆ A, allora A = B.
Se B ⊆ A e B diverso da A, allora A ⊂ B (A strettamente contenuto in B).

Il sottoinsieme soddisfa le seguenti proprietà:

Proprieta	Descrizione
Riflessiva	A ≤ A
Antisimmetrica	A ≤ B, B ≤ A -> A = B
Transitiva	A ≤ B, B ≤ C -> A ≤ C

→ Insieme potenza (insieme delle parti)

L'insieme potenza o insieme delle parti è l'insieme di tutti i possibili sottoinsiemi di un insieme dato.

Proprieta	Descrizione
|P(A)| = |(2^|A|)|	S = {X | X ≤ S}

→ Famiglia di Insiemi

La famiglia di insiemi è un insieme che contiene tutti gli elementi che rappresentano un determinato insieme.

Descrizione

UF = {x | x Є A per qualche elemento A Є F}

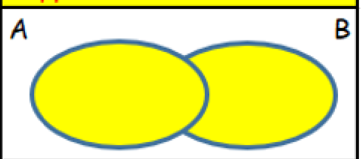
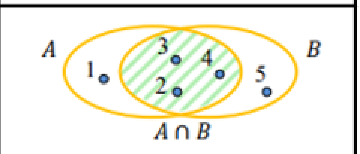
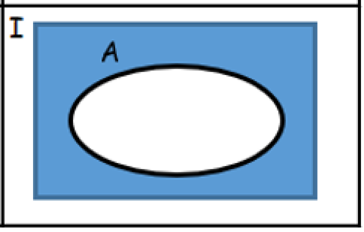
∩F = {x | x Є A per ogni elemento A Є F}

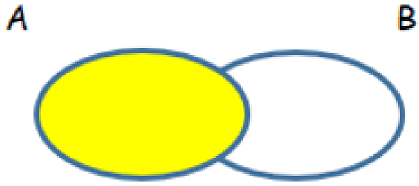
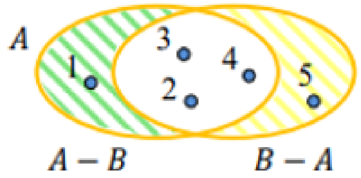
→ Partizione di un insieme

La partizione di un insieme dato A è un insieme che soddisfa tre proprietà:

1) ogni partizione contiene elementi diversi da una seconda partizione;
2) l'unione di tutti i sottoinsiemi deve essere equivalente all'insieme dato A;
3) non deve essere vuoto.

Capitolo 2 - Operazioni Insiemi

A, B, C sono insiemi		
Operazione	Rap. Caratteristica	Rappresentazione Grafica
$A \cup B$	{x | x Є A V x Є B}	
$A \cap B$	{x | x Є A and x Є B}	
¬A	{x | x Є I and x ¬Є A}	

$A \setminus B$	$\{x \mid x \in A \text{ and } x \neg\in B\}$	
$A \triangle B$	$(A \setminus B) \cup (B \setminus A)$	

Unione	
Proprieta'	Descrizione
Idempotente	$A \cup A = A$
Commutativa	$A \cup B = B \cup A$
Associativa	$A \cup (B \cup C) = (A \cup B) \cup C$
Neutro	$A \cup 0 = A$
Assorbimento	$A \cup B = B$ sse $A \leq B$
Monotonicità	$A \leq A \cup B$ e $B \leq A \cup B$

Intersezione	
Proprieta'	Descrizione
Idempotente	$A \cap A = A$
Commutativa	$A \cap B = B \cap A$
Associativa	$A \cap (B \cap C) = (A \cap B) \cap C$
Annichiliazione	$A \cap 0 = 0$
Assorbimento	$A \cap B = B$ sse $B \leq A$
Monotonicità	$A \cap B \leq A$ e $A \cap B \leq B$
Distributività 1	$A \cap (B \cup C) = A \cap B \cup A \cap C$
Distributività 2	$A \cup (B \cap C) = A \cup B \cap A \cup C$

Complementazione	
Proprieta'	
¬I = 0, ¬0 = I	
¬(¬A) = A	
A ∩ ¬A = 0	
A U ¬A = U	
¬(A U B) = ¬A ∩ ¬B	1 Legge di De Morgan
¬(A ∩ B) = ¬A U ¬B	2 Legge di De Morgan
A ≤ B sse ¬B ≤ ¬A	

Differenza
Proprieta'
A\A = 0
A\0 = A
0\A = 0
A\B = A ∩ ¬B
(A\B)\C = A\(C\B)
A\B != B\A

Differenza Simmetrica
Proprieta'
A Δ A = 0
A Δ 0 = A
0\A = 0
A Δ B = B Δ A
A Δ B = (A U B)\(A ∩ B)

Capitolo 3 - Relazioni

→ Premessa

Gli insiemi non sono ordinati

```
{x,y} = {y,x}
```

→ Coppia Ordinata

La coppia ordinata è la coppia dove è possibile distinguere il primo elemento con il secondo elemento.

```
<x,y>
```

L'elemento x è il primo elemento e l'elemento y è il secondo elemento.
`<x,y> != <y,x>`

Esiste anche la coppia ordinata `<x,x>`

→ Formulazione Insiemistica

La coppia ordinata `<x,y>` non è altro che l'insieme `{{x},{x,y}}` .
Sia `F = {{x},{x,y}}`
x è il primo elemento solo se x Є ∩F (appartiene a tutti gli insiemi);
y è il secondo elemento solo se:
→ y Є UF\∩F oppure
→ {y} = UF.

Notare che `<x,x> = {{x},{x,x}} = {{x}}`

→ Definizione Giusta

`<a,b> = <x,y>`
se `{{a},{a,b}} = {{x}{x,y}}`

→ Generalizzazione

È possibile generalizzare le coppie ordinate a tuple ordinate di lunghezza n (n-tuple ordinate) definendo
`<x1,...,xn,xn+1> = <<x1,...,xn>,xn+1>` con n>=2

→ Prodotto Cartesiano

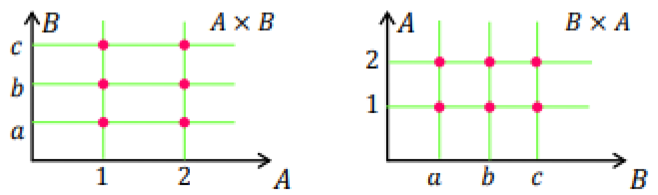
Il prodotto cartesiano è l'insieme di tutte le possibili coppie ordinate in cui il primo elemento appartiene al primo insieme e il secondo elemento al secondo insieme.

$$A = \{\ 1,2\ \} \qquad B = \{a,b,c\}$$

$$A \times B = \{(1,a),(1,b),(1,c),(2,a),(2,b),(2,c)\}$$

$$B \times A = \{(a,1),(b,1),(c,1),(a,2),(b,2),(c,2)\}$$

Grafico



→ Sequenze

Una sequenza è una tupla ordinata `<s1, …, sn>` dove ogni n Є N e ogni s_i Є S.
S^n è l'insieme di tutte le n – tuple di elementi di S.

Una sequenza finita di elementi di S è un elemento di S per qualche n Є N.
Un segmento della sequenza finita δ = <s1, …, sn> è una sequenza δ' = <sk, sk+1, …, sl> dove 1 <= k <= l <= n.
Il segmento è iniziale solo se k=1.

→ Relazioni

Una relazione è un sottoinsieme del prodotto cartesiano di due o più insiemi. La relazione tra A e B è un sottoinsieme di A x B.

Esempio

```
A = {a,b,c,d,e,f}
B = {1,2,3}

A x B = {<a,1>,<a,2>,<a,3>,<b,1>,<b,2>,<b,3>,
         <c,1>,<c,2>,<c,3>,<d,1>,<d,2>,<d,3>,
         <e,1>,<e,2>,<e,3>,<f,1>,<f,2>,<f,3>}
```

Una relazione può essere:

```
R = {<a,1>,<a,2>,<b,1>}
```

Se la coppia ordinata <x,y> appartiene alla relazione R, significa che x è in relazione con y (x Є A and y Є B).

→ Come rappresentare una relazione?

Una relazione può essere rappresentata mediante le seguenti modalità:

1 – Rappresentazione tabulare

B\A	a	b	c	d	e	f
1	x	x				
2	x					
3						

2 – Matrice booleana (0/1)

1	1	0
1	0	0
0	0	0
0	0	0
0	0	0
0	0	0

Ogni riga rappresenta l'elemento dell'insieme A e ogni colonna rappresenta l'elemento dell'insieme B.

→ Quali sono gli elementi di una realzione?

Sia R ⊆ A x B una relazione:

a) si dice dominio di R (dom(R)), l'insieme di tutti gli oggetti x Є A tali che <x,y> Є R per qualche y Є B.

$dom(R) = \{x \in A \mid \exists y \in B. \langle x, y\rangle \in R\}$

b) si dice codominio di R l'insieme di tutti gli oggetti y ∈ B tali che <x,y> ∈ R per qualche x ∈ A.

$codom(R) = \{y \in B \mid \exists x \in A. \langle x, y\rangle \in R\}$

→ Relazioni n-arie

a) Una relazione si dice binaria se costituita da un insieme di coppie ordinate.

b) Una relazione si dice ternaria se costituita da un insieme di triple ordinate.

c) Una relazione si dice quaternaria se costituita da un insieme di quartuple ordinate.

Se gli elementi delle tuple appartengono allo stesso insieme A, allora una relazione n-aria é un sottoinsieme di A^n.

Esempio

```
1. {<x,x> | x Є A}  relazione binaria su A
2. {<x,y> | x,y ∈ N, x ≤ y} relazione d'ordine naturale su N
3. {<x,y,z> | x,y,z ∈ R, x^2+y^2 = z^2} area geometrica
```

→ Operazioni Relazioni

R,S sono relazioni

Operazione	Risultato
R U S	{x | x Є R V x Є S}
R ∩ S	{x | x Є R ∩ x Є S}
¬R	{<x,y> | <x,y> ¬Є R}
R^(-1)	{<y,x> | <x,y> Є R}

→ Proprietà di relazioni

Proprietà	Conseguenza
R ⊆ S	¬S ⊆ ¬R
¬(R ∩ S)	¬R U ¬S
¬(R U S)	¬R ∩ ¬S
R ⊆ S	R^-1 ⊆ S^-1
(R ∩ S)^-1	R^-1 U S^-1

Esempio

Siano $A = \{a, b\}$, $R = \{\langle a, b\rangle, \langle b, a\rangle\}$, $S = \{\langle a, b\rangle, \langle a, a\rangle\}$

1. $R \cap S = \{\langle a, b\rangle\}$
2. $\overline{(R \cup S)} = \{\langle b, b\rangle\}$
3. $R^{-1} = R$
4. $S^{-1} \neq S$

→ Identità

Dato un insieme A, si dice identità su A

$I_A = \{\langle x,x\rangle \mid x \in A\}$

una relazione che associa ogni elemento x dell'insieme A a se stesso.

→ Proprietà relazioni binarie

R relazione binaria		
Proprietà	Descrizione	Descrizione
riflessiva	$\langle x,x\rangle \in R$ per ogni $x \in A$	$(I \subseteq R)$
simmetrica	se $\langle x,y\rangle \in R$ implica $\langle y,x\rangle \in R$	R = R^-1
asimmetrica	se $\langle x,y\rangle, \langle y,x\rangle \in R$ implica $x = y$	R ∩ R^-1 ⊆ IA
transitiva	se $\langle x,y\rangle, \langle y,z\rangle \in R$ implica $\langle x,z\rangle \in R$	

Esempio

Sia $A = \{a, b, c\}$

1. $R_1 = \{\langle a, a\rangle, \langle a, b\rangle, \langle b, a\rangle, \langle a, c\rangle\}$
   non riflessiva, non simmetrica, non transitiva, non antisimmetrica
2. $R_2 = \{\langle a, a\rangle, \langle b, b\rangle, \langle c, c\rangle, \langle a, b\rangle, \langle b, a\rangle, \langle a, c\rangle\}$
   riflessiva, non simmetrica, non transitiva, non antisimmetrica
3. $R_3 = \{\langle a, a\rangle, \langle b, b\rangle\}$
   non riflessiva, simmetrica, transitiva, antisimmetrica

Capitolo 4 – Funzioni

→ Generalità

Una funzione è una legge (o relazione) che associa un elemento x dell'insieme A un solo elemento y dell'insieme B.

Una funzione $y = f(x)$ ha le seguenti proprietà:

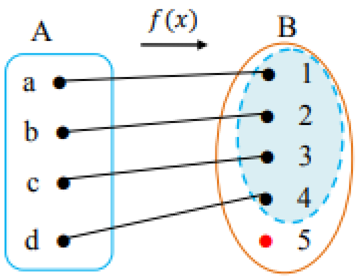
1) l'elemento y appartenente all'insieme B viene detto immagine di x (x contro immagine di y).
2) L'insieme delle x, associate agli elementi y appartenenti all'insieme B, viene detto domino o campo di esistenza.
3) L'insieme delle immagini viene detto codominio.

Una funzione può essere di tre tipologie:

1) Iniettiva: una funzione si dice iniettiva se ogni elemento y appartenente all'insieme B è immagine di al più un elemento x appartenente all'insieme A.

Notazione:

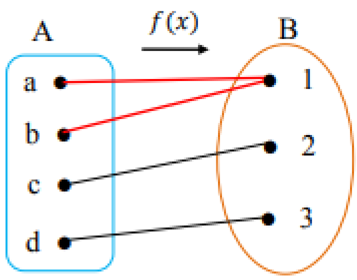
$f(x)$ iniettiva $\Leftrightarrow$ $x_1 \neq x_2$ $\rightarrow$ $f(x_1) \neq f(x_2)$



2) Suriettiva: una funzione si dice suriettiva se ogni elemento y appartenente all'insieme B è immagine di almeno un elemento x appartenente all'insieme A.

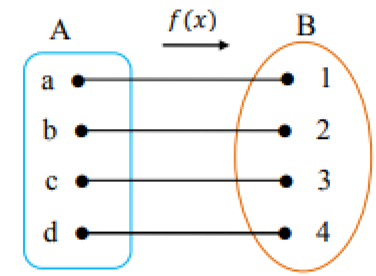
Notazione

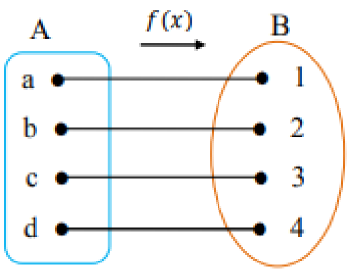
$f(x)$ suriettiva $\Leftrightarrow$ $\forall\, y \in B\ \exists\, x \in A : f(x) = y$



3) Biiettiva (o Biunivoca): una funzione si dice biiettiva (o biunivoca) se è sia iniettiva che suriettiva.

$f(x)$ biunivoca $\Leftrightarrow$ $\forall\, x \in A\ \exists!\ y \in B : f(x) = y$ e viceversa

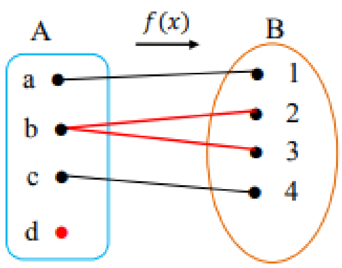




funzione **biunivoca** o biettiva

- una funzione si dice **biunivoca** (o biettiva) quando è sia iniettiva che suriettiva, cioè quando **ad ogni** elemento dell'insieme A corrisponde **uno ed un solo** elemento dell'insieme B e **viceversa**
- $f(x)$ biunivoca $\Leftrightarrow$ $\forall\, x \in A\ \exists!\ y \in B : f(x) = y$ e viceversa

· l'insieme A è il dominio, ***tutto*** l'insieme B è il codominio di $f(x)$



corrispondenza

la legge rappresentata nella figura a sinistra **non** è una funzione perché non ne soddisfa la definizione, infatti:

- all'elemento "b" dell'insieme A sono associati più elementi ("2, 3") dell'insieme B.
- l'elemento "d" dell'insieme A non è associato ad alcun elemento dell'insieme B.

la legge non è una funzione ma prende il nome di **corrispondenza**

→ Funzione Inversa

Sia $f: A \to B$ una funzione e $y \in B$. L'immagine inversa di f in y è

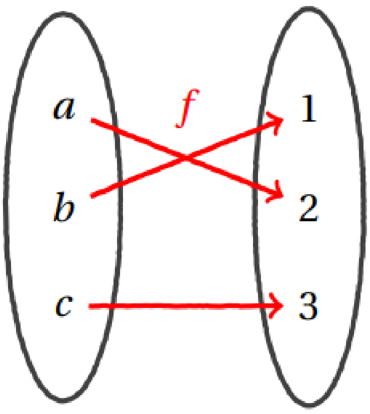
$f^{-1}(y) = \{x \in A \mid f(x) = y\}$

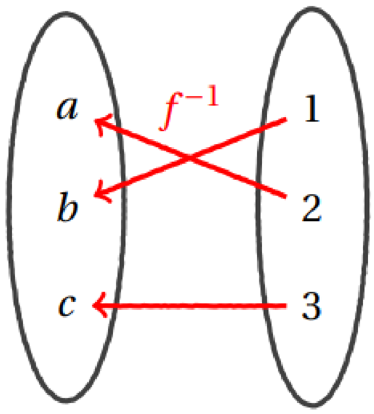
Una funzione $f : A \to B$ è invertibile se esiste una funzione $g : B \to A$ tale che per ogni $x \in A$ e ogni $y \in B$:

$g(f(x)) = x$

$f(g(y)) = y$

In questo caso, g è l'inverso di f e si rappresenta come f^{-1}.





→ Caratterizzazione

Una funzione f è invertibile solo se è iniettiva. L'inverso f^{-1} è totale se f è suriettiva.

Esempi

a. La funzione somma $f: N^2 \to N$

f(x,y) = x + y non è iniettiva (non è invertibile).

b. op è invertibile (op^{-1} = op)

c. $f : N \to N^+$, f(x) = x +1 è invertibile $f^{-1}(y) = y - 1$.

→ Composizione di funzioni

Date due funzioni f e g, si dice funzione composta di f e g, e si indica con il simbolo g o f, la funzione definita da:

(g o f)(x) = g(f(x)).

Proprietà Composizione

Proprietà	Descrizione
f o (g o h)	(f o g) o h
f, g iniettive	f o g iniettiva
f, g suriettive	f o g suriettiva
f, g invertibili	f o g invertibile

→ Funzione Caratteristica

I sottoinsiemi di un insieme A si possono anche rappresentare tramite una funzione detta caratteristica.

La funzione caratteristica di un insieme $S \subseteq A$ é la funzione $f_S : A \to \{0, 1\}$ dove

$$f_S(x) = \begin{cases} 0 & x \notin S \\ 1 & x \in S \end{cases}$$

Proprietà

Per ogni $x \in A$	
Proprietà	Operazione
$f(S \cap T)(x)$	$fS(x) \cdot fT(x)$
$f(S \cup T)(x)$	$fS(x) + fT(x) - fS(x) \cdot fT(x)$
$f(S \Delta T)(x)$	$fS(x) + fT(x) - 2 \cdot fS(x) \cdot fT(x)$

→ Multinsieme

Un multinsieme é una variante di un insieme dove gli elementi si possono ripetere

```
{{a,a,b,c,c,c}} != {{a,b,c}}
```

Formalmente, un multinsieme è una funzione da un insieme a N ($f: A \rightarrow N$) che esprime quante volte si ripete ogni elemento nel multinsieme (A = {a,b,c,d})

{<a,2>,<b,1>,<c,3>,<d,0>}

Capitolo 5 – Cardinalità

→ Definizione di numeri cardinali

I numeri cardinali sono un tipo generalizzato di numeri naturali utilizzati per indicare la grandezza degli insiemi finiti. Essi riescono a classificare gli insiemi infiniti.

Cantor utilizzò il concetto di funzione biunivoca per definire la cardinalità: se due insiemi hanno la stessa cardinalità, allora esiste una funzione biunivoca tra i due insiemi.

Sfruttando i concetti di funzione iniettiva, suriettiva e biunivoca, è possibile stabilire se la cardinalità di un insieme è maggiore della seconda o viceversa.
Dati due insiemi A e B:
a) la funzione si dice iniettiva, se $|A| < |B|$ (A non è più grande di B);
b) la funzione si dice suriettiva, se $|A| > |B|$ (B non è più grande di A);
c) la funzione si dice biiettiva, se $|A| = |B|$ (A e B sono equipotenti).

Se A é equipotente a {1, ... , n} si dice che A ha cardinalità (o potenza) n e si scrive $|A| = n$ (A è un insieme finito).

→ Cardinalità di N

La cardinalità dell'insieme N (che non é finita, e quindi non è un numero naturale) si chiama aleph zero ($\aleph_0$). Questo valore è il minore di tutti i valori transfiniti.
L'insieme dei transfiniti è così definito $\aleph_0 < \aleph_1 < \aleph_2 < \ldots < \aleph_n$}.
Un insieme A é numerabile se é equipotente ad N e quindi ha cardinalità $\aleph_0$.

→ Teorema di Cantor

Il teorema di Cantor afferma che non esiste una funzione biunivoca tra N e P(N).
$\aleph_0 < 2^{\aleph_0}$

Ipotesi del continuo: non ci sono transfiniti intermedi tra $\aleph_0$ e $2^{\aleph_0}$ ($\aleph_1 = 2^{\aleph_0}$).

Dimostrazione
Supponiamo che esiste una tale funzione biunivoca $f : N \rightarrow P(N)$ e definiamo

$$Z = \{x \in \mathbb{N} \mid x \notin F(x)\}$$

Come Z ⊆ N e f \`e biunivoca, esiste n ∈ N dove f (n) = Z

$$n \in Z \text{ sse } n \notin f(n) \text{ sse } n \notin Z$$

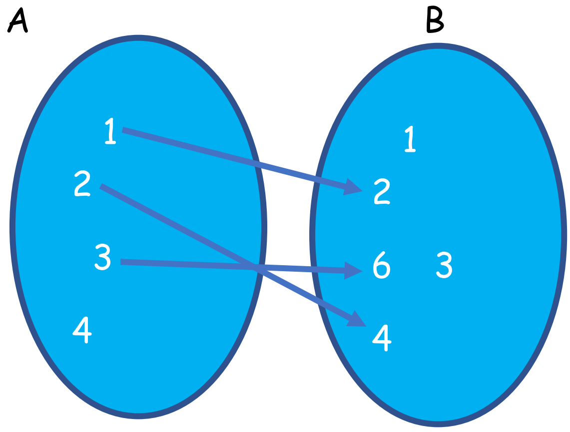
Capitolo 6 – Rappresentazione di una Relazione

Una relazione può essere rappresentata in diverse modalità:

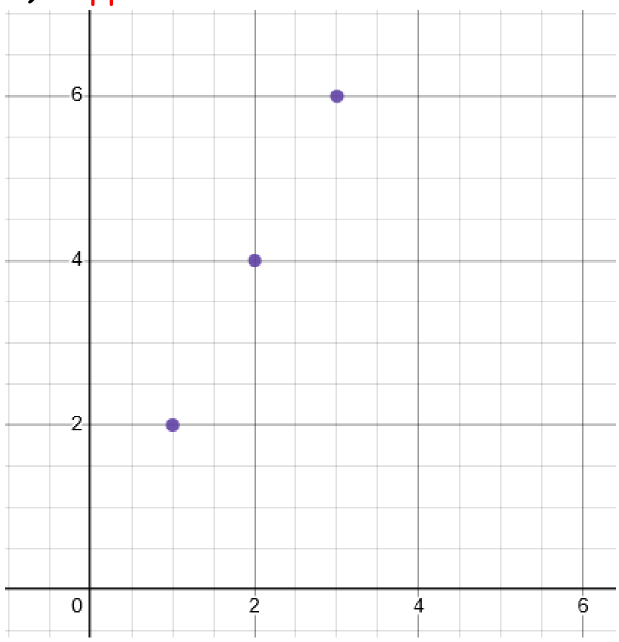
1) Rappresentazione per Elencazione

```
R = {<1,2>,<2,4>,<3,6>}
```

2) Rappresentazione Sagittale



3) Rappresentazione Cartesiana



x_1	y_1
1	2
2	4
3	6

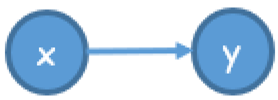
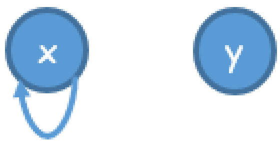
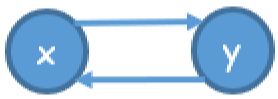
4) Rappresentazione Tabulare

A \ B	1	2	3	4	5	6
0						
1		x				
2				x		
3						x
4						

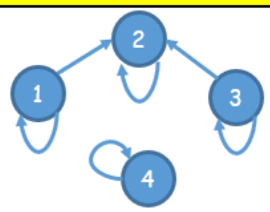
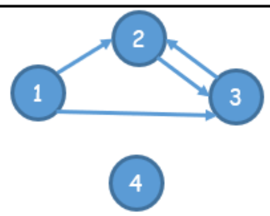
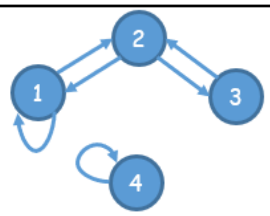
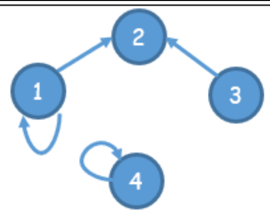
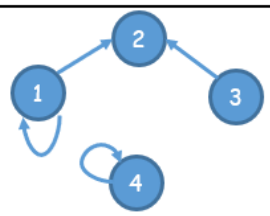
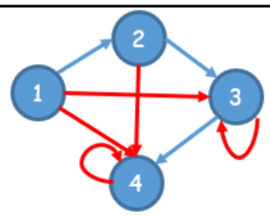
5) Matrice Booleana

0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	1	0	0
0	0	0	0	0	1
0	0	0	0	0	0

6) Grafo

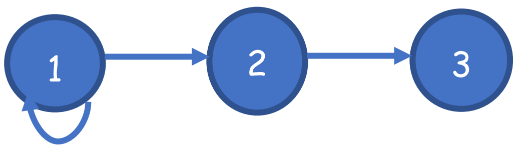
<table>
<tr><td colspan="2">Se ⟨x, y⟩ ∈ R</td></tr>
<tr><td>Proprietà</td><td>Grafo</td></tr>
<tr><td>‹x,y› ∈ R</td><td>
</td></tr>
<tr><td>‹x,x› ∈ R</td><td>
</td></tr>
<tr><td>⟨x,y⟩ ∈ R e ⟨y,x⟩ ∈ R</td><td>
</td></tr>
</table>

→ Proprietà

Se ⟨x, y⟩ ∈ R			
Proprietà	**Descrizione**	**Grafo**	**Matrice Booleana**
riflessiva	se ⟨x, x⟩ ∈ R ∀x ∈ S		$\begin{matrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$
irriflessiva	se ⟨x, x⟩ ¬∈ R ∀x ∈ S		$\begin{matrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$
simmetrica	se ⟨x,y⟩ ∈ R qualora ⟨y,x⟩ ∈ R		$\begin{matrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$
asimmetrica	se ⟨x,y⟩ ∈ R -> ⟨y,x⟩ ∈ R		$\begin{matrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$
antisimmetrica	se ⟨x, y⟩ ∈ R e ⟨y, x⟩ ¬∈ R -> x = y		$\begin{matrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$
transitiva	se ⟨x, y⟩ ∈ R e ⟨y, z⟩ ∈ R -> ‹x,z› Є R		$\begin{matrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{matrix}$

→ Riflessività ed irriflessività

Ci sono relazioni che non sono né riflessive né irriflessive.

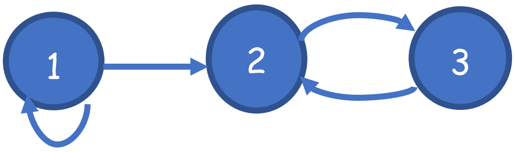


→ Non è riflessiva perché ⟨2,2⟩ /∈ R;

→ Non è irriflessiva perché ⟨1,1⟩ ∈ R;

→ Simmetria ed Anti simmetria

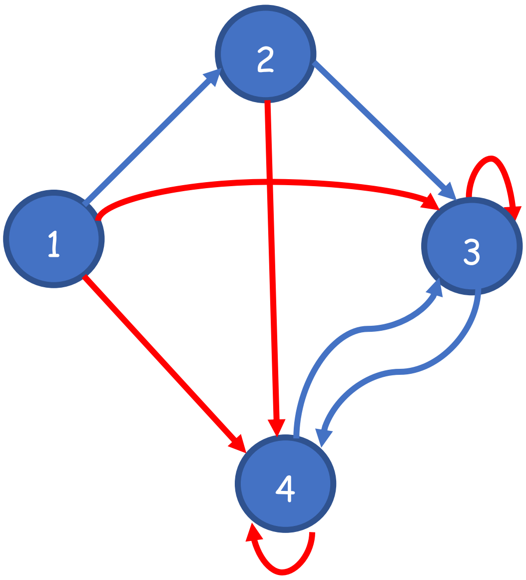
Esistono relazioni che non sono né simmetriche né antisimmetriche.

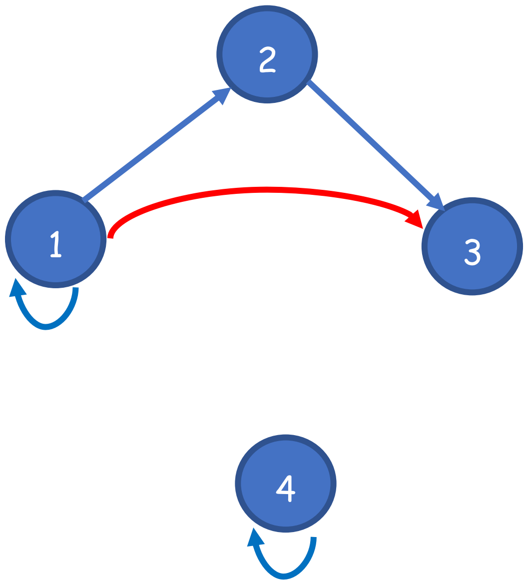


→ Non è simmetrica perché ⟨1, 2⟩ ∈ R ma ⟨2, 1⟩ ∉ R.

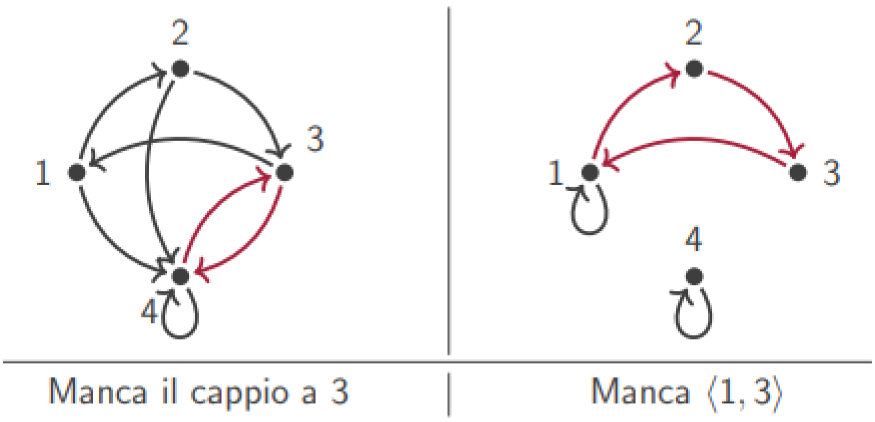
→ Non è antisimmetrica perché ⟨2, 3⟩ ∈ R e ⟨3, 2⟩ ∈ R.

Esempio: Relazioni Transitive





Esempio: Relazioni non transitive



→ Quando una relazione si dice connessa?

Una relazione si dice connessa se ogni due elementi sono collegati. Per ogni x, y ∈ S, se x /= y allora ⟨x, y⟩ ∈ R oppure ⟨y, x⟩ ∈ R.

→ Riflessività ed operazioni

Siano R ed R' due relazioni su S:

1) se R è riflessiva, R^−1 è riflessiva;

2) R è riflessiva solo se R è irriflessiva;

3) se R ed R' sono riflessive, allora anche R ∪ R' e R ∩ R' sono riflessive.

→Transitività e operazioni

Se R ed R' sono transitive allora R ∩ R' è transitiva.

→ Matrice Booleana

La matrice booleana è una matrice a valori {0,1}.

Associata a $R \subseteq A \times B$, si denota con M_R.

Se |A| = n e |B| = m, M_R possiede n righe ed m colonne.

La riga i corrisponde all' elemento $a_i \in A$; la colonna j corrisponde all' elemento $b_j \in B$; ed è tale che:

$$m_{ij} = \begin{cases} 1 & \langle s_i, t_j \rangle \in R \\ 0 & \text{altrimenti} \end{cases}$$

A \ B	1	2	3	4	5	6
0						
1		x				
2				x		
3						x
4						

0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	1	0	0
0	0	0	0	0	1
0	0	0	0	0	0

→ Proprietà di M_R

1) R è riflessiva solo se M_R ha tutti 1 sulla diagonale principale;

2) R è irriflessiva solo se M_R ha tutti 0 sulla diagonale principale;

3) R è simmetrica solo se M_R è simmetrica;

4) R è asimmetrica solo se per ogni i /= j, se $m_{ij} = 1$, allora $m_{ji} = 0$;

5) $M_{R^{-1}}$ è la trasposta di M_R.

6) (!) M_R si ottiene scambiando 0 con 1.

→ Operazioni con matrici booleane

Siano M e N due matrici booleane di dimensioni m x n, le operazioni tra matrici booleane possono essere:

1) join di M e N ($M \sqcup N$) è la matrice booleana L di dimensione m x n, i cui elementi possono essere 1, se $m_{ij} = 1$ o $n_{ij} = 1$, o 0 altrimenti;
2) meet di M e N ($M \sqcap N$) è la matrice booleana L di dimensione m x n, i cui elementi possono essere 1, se $m_{ij} = 1$ e $n_{ij} = 1$, o 0 altrimenti;

→ Prodotto booleano

Siano M e N matrici booleane di dimensioni n × m e m × p rispettivamente, il loro prodotto booleano è la matrice $L = M \odot N$ di dimensioni n × p dove:

$$\ell_{ij} = \begin{cases} 1 & \text{se esiste } k, 1 \leq k \leq m \text{ tale che } m_{ik} = 1 \text{ e } n_{kj} = 1 \\ 0 & \text{altrimenti} \end{cases}$$

1	0	0	1	*	0	0	1		0	1	1
0	0	1	0		0	1	0		1	1	1
					1	1	1				
					0	1	0				

→ Composizione di relazioni

$R1 \subseteq S \times T$;

$R2 \subseteq T \times Q$.

$R2 \circ R1 = \{\langle x, y\rangle \in S \times Q \mid \exists z \in T\ \langle x, z\rangle \in R1, \langle z, y\rangle \in R1\}$

La composizione di relazioni si calcola tramite il prodotto di matrice:

$M_{R2 \circ R1} = M_{R1} \odot M_{R2}$

Capitolo 7 – Relazione di equivalenza

→ Definizione

Una relazione di equivalenza è una relazione che soddisfa tre proprietà: riflessiva, simmetrica e transitiva. Essa permette di creare blocchi di elementi che hanno qualcosa in comune.

Esempio 1:

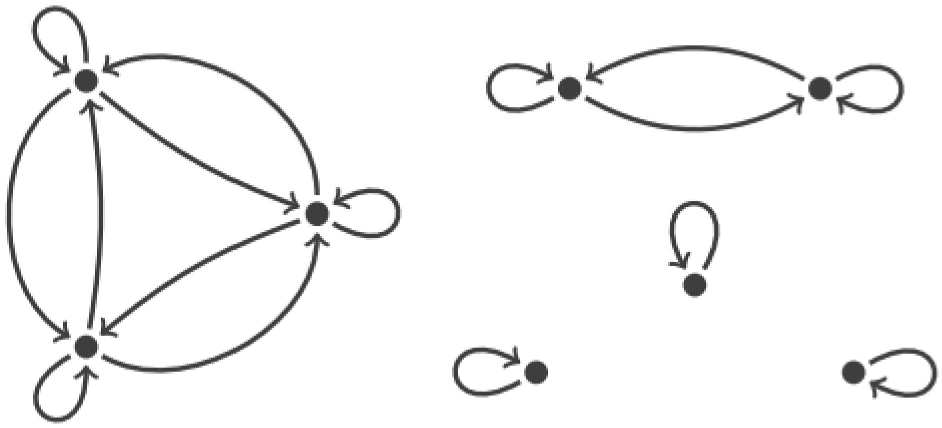
appartenere alla stessa classe;
essere nati nello stesso anno;
essere parallele nell'insieme delle rette.

Esempio 2:

Se $f: A \to B$ è una funzione totale, allora la relazione

R = {⟨x, y⟩ ∈ A × A | f (x) = f (y)}
è una relazione di equivalenza.

→ Visualizzazione
La rappresentazione sagittale di una relazione di equivalenza consiste di diversi grafi totalmente collegati.



→ Classe di equivalenza
La classe di equivalenza è un sottoinsieme che contiene tutti gli elementi equivalenti a un qualche elemento. In una classe di equivalenza tutti gli elementi in essa contenuti sono tra loro equivalenti.

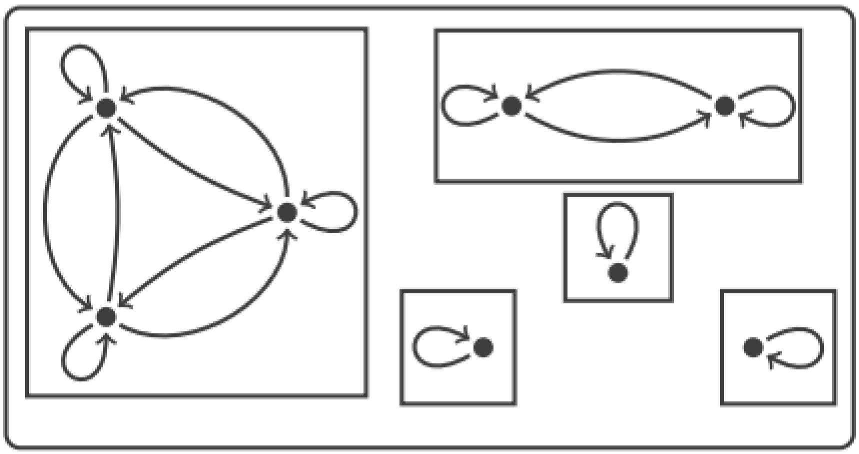
→ Partizione di un insieme S
Sia S un insieme, una partizione di S è una famiglia di insiemi
P = {T1, …, Tn}, Ti ⊆ S, 1 ≤ i ≤ n tali che:
Ti ≠ ∅ per ogni i, 1 ≤ i ≤ n;
Ti ∩ Tj = ∅ per ogni 1 ≤ i < j ≤ n; e
UP = S

Se R è una relazione di equivalenza su S, allora T ≠ ∅ ⊆ S è una classe di equivalenza se per ogni x ∈ S:
x ∈ T solo se {y ∈ S | ⟨x, y⟩ ∈ R} = T

Sia S un insieme e R una relazione di equivalenza su S.
Ogni elemento x ∈ S definisce una classe di equivalenza:
[x]R = {y ∈ S | ⟨x, y⟩ ∈ R}

La famiglia di insiemi {[x]R | x ∈ S} (gli elementi sono le classi di equivalenza di S) è chiamato insieme quoziente di S rispetto a R (si indica con S/R).

Esempio 1



Esempio 2

Sia $n \in N$. La relazione $\simeq n \subseteq N \times N$ definita come $x \simeq n\, y$ solo se $x \equiv y \bmod n$ (ossia ($x \bmod n$) = ($y \bmod n$)) è una relazione di equivalenza.

Per n = 4, $\simeq 4$ definisce 4 classi di equivalenza:

$[x] = \{x + 4k \mid k \in N\}$

$[0] = \{0, 4, 8, 12, \ldots\}$
$[1] = \{1, 5, 9, 13, \ldots\}$
$[2] = \{2, 6, 10, 14, \ldots\}$
$[3] = \{3, 7, 11, 15, \ldots\}$

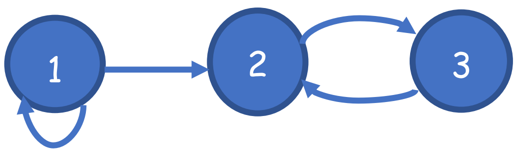
Capitolo 8 – Grafi

→ Generalità

Un grafo è una struttura matematica che è costituita da:
un insieme di nodi (detti vertici);
collegamenti tra vertici che possono essere
- orientati (archi) (grafo orientato);
- non orientati (spigoli) (grafo non orientato);
dati associati a nodi e collegamenti (etichette).

→ Come si rappresentano?

Un grafo viene rappresentato disegnando punti per i nodi, e segmenti o curve per i collegamenti tra i nodi.
Importante:
la posizione o forma dei nodi e dei collegamenti sono irrilevanti.
soltanto la loro esistenza definisce il grafo.



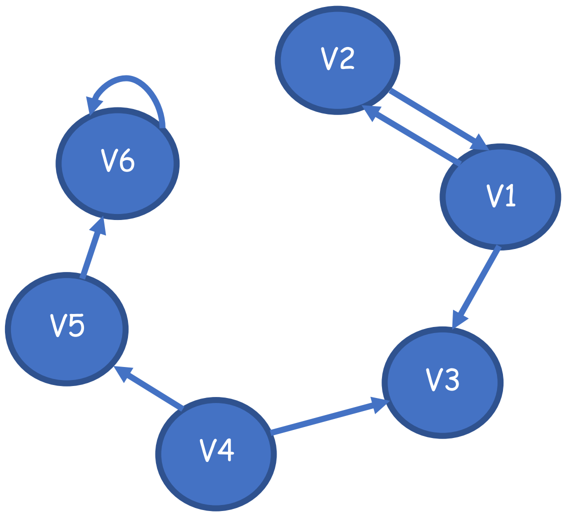
→ Relazioni binarie

I grafi possono rappresentare relazioni binarie

Esempio:

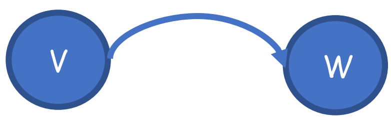
V = {V1, V2, V3, V4, V5, V6}

V1	V1
V1	V2
V1	V3
V2	V1
V4	V3
V4	V5
V5	V6
V6	V6



1	1	1	0	0	0
1	0	0	0	0	0
0	0	0	0	0	0
0	0	1	0	1	0
0	0	0	0	0	1
0	0	0	0	0	1

→ Terminologie: gradi



L'arco che connette V e W è detto uscente da V ed entrante in W.

Il numero di archi uscenti dal nodo V è il grado di uscita di V e il numero di archi entranti in V è il grado di ingresso di V.

Un nodo è chiamato:

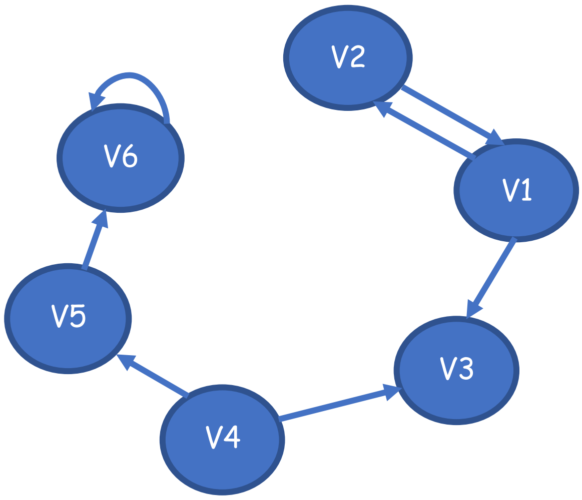
sorgente: se non ha archi entranti (grado di ingresso è 0);

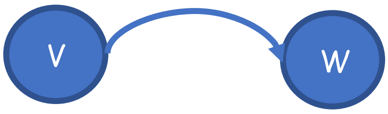
pozzo: se non ha archi uscenti (grado di uscita 0);

isolato: se non ha archi né uscenti né entranti.

Esempio:

	Grado Uscita	Grado Ingresso	Sorgente	Pozzo
V1	2	1	no	no
V2	1	1	no	si
V3	0	2	no	si
V4	2	0	si	no
V5	1	1	no	no
V6	1	2	no	no





I nodi V e W sono adiacenti se vi è un arco che li connette (qualunque sia la direzione). L'arco è incidente su V e W e il grado di V è il numero di nodi adiacenti a V.

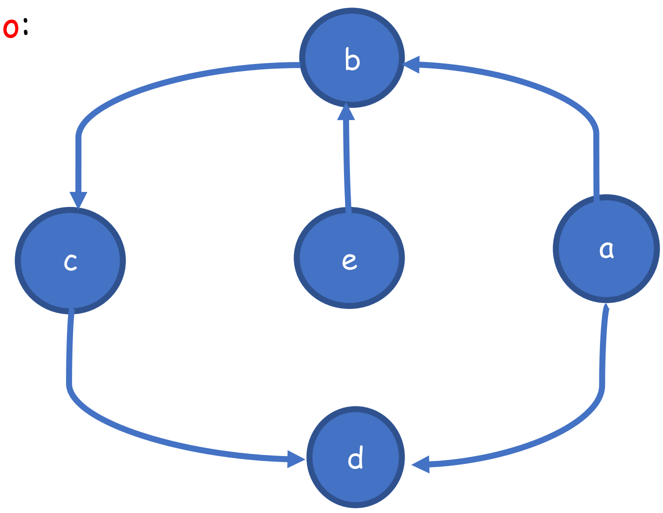
→ Terminologia: cammino

Un cammino è una sequenza finita di nodi <v1, v2, …, vn> tali che per ogni i, 1 <= i <n, esiste un arco uscente da v_i ed entrante in v_{i+1}. Il cammino parte da V a W se v1 = V e vn = W

→ Terminologia: semi cammino

Un semi - cammino è una sequenza finita di nodi <v1, v2, …, vn> tali che per ogni i, 1 <= i <n, esiste un arco che collega v_i e v_{i+1} in direzione arbitraria. La lunghezza di un semi – cammino è il numero di archi che lo compongono (n – 1). Un semi – cammino è semplice se tutti i nodi nella sequenza sono diversi. Un grafo è connesso se esiste sempre un semi – cammino tra due nodi qualsiasi.

Esempio:



Cammini	Lunghezza
<a,d>	1
<a,b,c,d>	3

Semi- Cammini	Lunghezza
<a,b,e>	2
<a,b,c,d,a,b,e>	6

→ Terminologia: ciclo

Un ciclo intorno al nodo V è un cammino tra V e V.

→ Terminologia: semi – ciclo

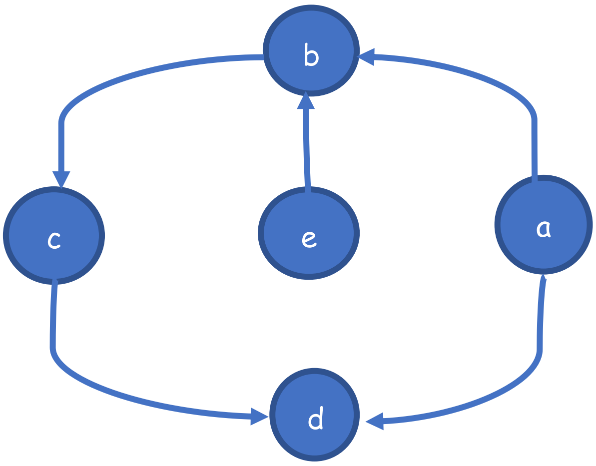
Un semi – ciclo intorno al nodo V è un semi – cammino tra V e V.

→ Terminologia: cappio

Un cappio intorno a V è un ciclo di lunghezza 1.

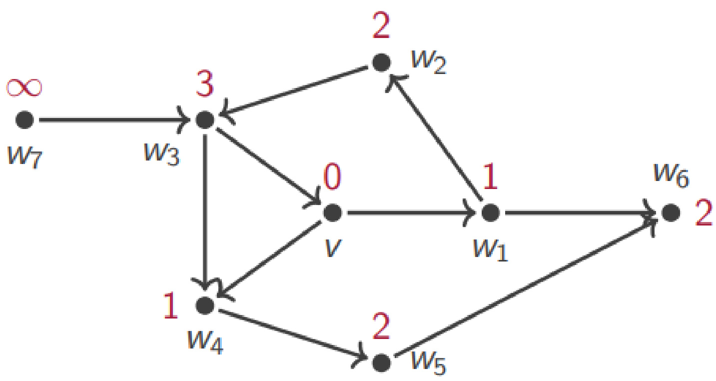
→ Terminologia: distanza

La distanza da V a W è la lunghezza del cammino più corto tra V e W.
La distanza da V a V è sempre 0.
Se non vi è nessun cammino tra V e W allora la distanza è infinita (∞).
In un grafo ordinato, la distanza da V e W non è sempre equivalente alla distanza da W a V.



Distanza	Valore
<a,b>	1
<a,d>	1
<b,a>	3
<d,a>	1
<e,d>	3
<a,e>	∞

→ Trovare le distanze



Le distanze da v ad ogni nodo del grafo.

→ Trovare distanze: Algoritmo

Ricerca in ampiezza delle distanze da v ad ogni nodo.

Inizializzazione:

1) segnare v come visitato con distanza `d(v) = 0`;

2) segnare altri nodi non visitato;

Ciclo: finché ci sono nodi visitato

3) trovare un nodo w visitato con distanza minima `d(w) = n`;

4) segnare w come esplorato;

5) per ogni nodo w' incidente da w: se w' è non visitato, segnare w' come visitato e `d(w') = n+1`

Finalizzazione

ad ogni nodo w non visitato assegnare `d(w) = ∞`

→ Trovare distanze: Algoritmo

Grafo Orientato: è una coppia `G = (V, E)` dove:

1) V è un insieme di nodi;

2) $E \subseteq V \times V$ è una relazione binaria in V.

Grafo non orientato: è un grafo orientato dove E è una relazione simmetrica. Gli archi sono rappresentati come coppie non ordinate `(v,w) [(v,w) = (w,v)]`.

→ Sotto grafo

Il grafo $G_1 = (V_1, E_1)$ è un sotto grafo di $G_2 = (V_2, E_2)$ solo se $V_1 \subseteq V_2$ ed $E_1 \subseteq E_2$.

Un sotto grafo si ottiene togliendo nodi e/o archi dal grafo.

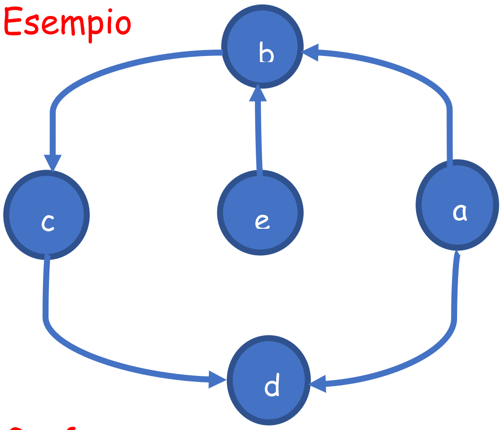
Sia $G = (V, E)$ un grafo.

Il sotto grafo indotto da $V' \subseteq V$ è il grafo che ha soltanto archi adiacenti agli elementi di V'.

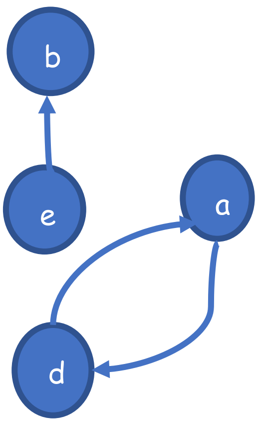
Formalmente, è il grafo $G = (V', E')$ dove

$$E' = \{\langle v, w\rangle \in E \mid v, w \in V'\}$$

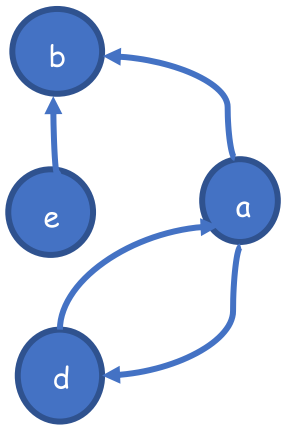
Esempio



Grafo



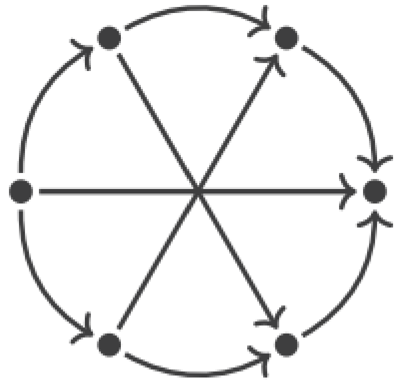
Sotto Grafo



Sotto Grafo

→ DAG – Grafo Aciclico Orientato

Un grafo orientato senza cicli, in cui non esiste nessun cammino da un nodo a se stesso.

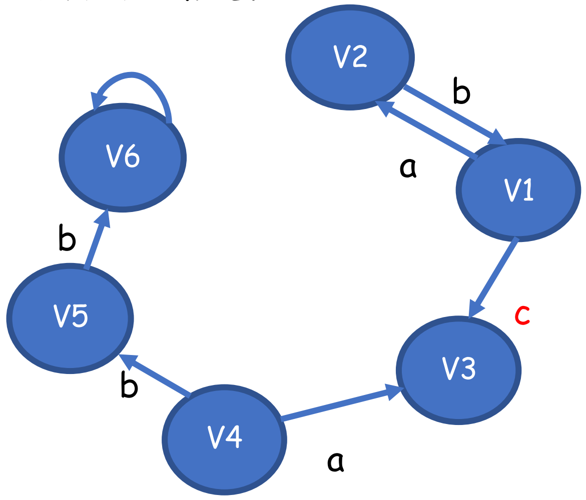


→ Grafo etichettato

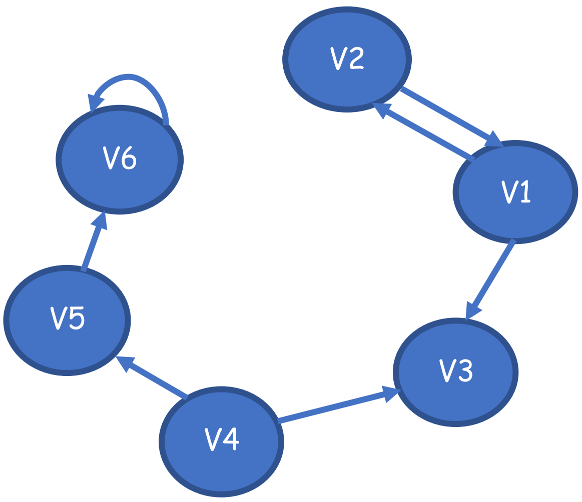
Un grafo etichettato è una tripla `G = (V,E,l)` dove:

1) `(V,E)` è un grafo;

2) `l:E → L` è una funzione totale che associa ad ogni arco e ∈ E una etichetta da un insieme L.



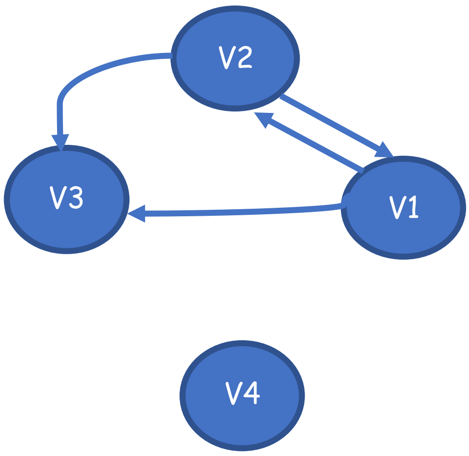
→ Relazioni ternarie



V1	V1	a
V1	V2	b
V1	V3	c
V2	V1	a
V4	V3	a
V4	V5	b
V5	V6	b
V6	V6	c

→ Matrice di adiacenza

La matrice di adiacenza di un grafo G = (V,E) è la matrice booleana della relazione E.



0	1	1	0
1	0	1	0
0	0	0	0
0	0	0	0

La matrice di adiacenza di grafi non orientati è sempre simmetrica.

→ Grafo completo

Un grafo si dice completo collega ogni nodo con tutti gli altri nodi (ma non con se stesso). La matrice di adiacenza ha 0 su tutta la diagonale, ed 1 sulle altre posizioni.

0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0

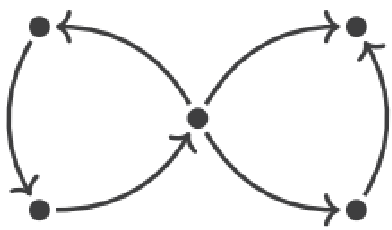
→ Grafo fortemente connesso

Un grafo G si dice fortemente connesso se per ogni due nodi v, w ∈ V esiste un cammino da v a w.

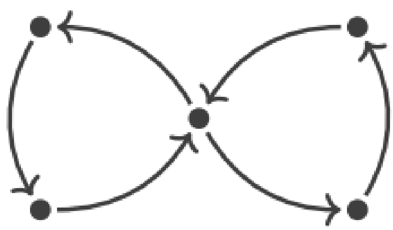
In un grafo fortemente connesso:

1) esiste sempre un ciclo che visita ogni nodo (non necessariamente semplice);
2) non ci sono né sorgenti né pozzi;

Esempio



connesso



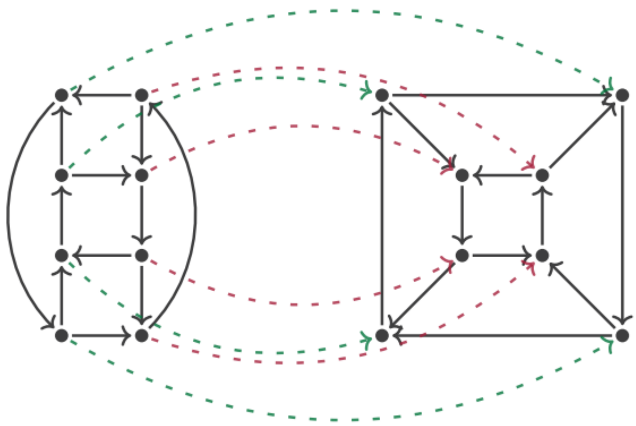
fortemente connesso

→ Isomorfismi tra grafi

Due grafi `G1 = (V1, E1)` e `G2 = (V2, E2)` sono isomorfi se esiste una funzione biunivoca f : V1 → V2 tale che

$$\langle v, w \rangle \in E_1 \quad \text{sse} \quad \langle f(v), f(w) \rangle \in E_2$$

L'isomorfismo f mantiene la struttura del grafo G1, ma sostituisce i nomi dei vertici per quelli di G2. Due grafi isomorfi sono in realtà lo stesso grafo con i nodi rinominati.



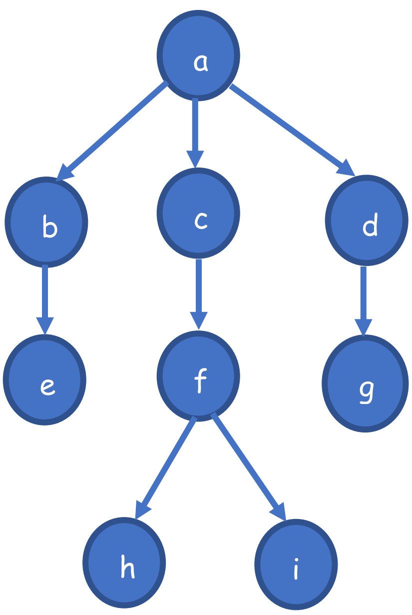
Capitolo 9 – Alberi

→ Generalità

Un albero è un DAG connesso tale che:

1) esiste esttamente un nodo sorgente (radice dell'albero);
2) ogni nodo diverso dalla radice ha un solo arco entrante.

I nodi pozzo di un'albero sono chiamati foglie oppure nodi esterni.
Tutti gli altri nodi sono chiamati interni.



Nodo	Tipologia
<a>	radice
<b,c,d,f>	padre
<e,g,h,i>	foglia
<e,g,h,i>	figlio

→ Proprietà

Il grado di ingresso di un nodo è:

1 se non è la radice;

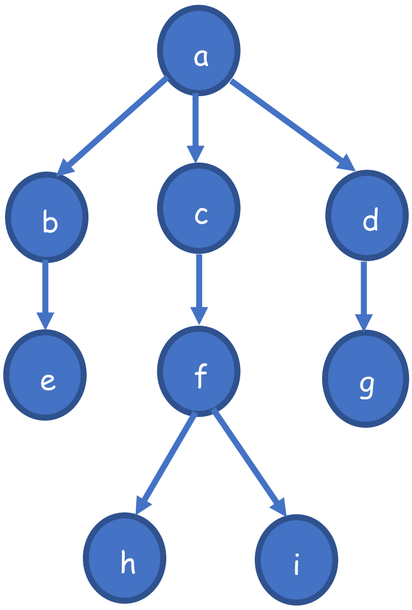
0 se è la radice;

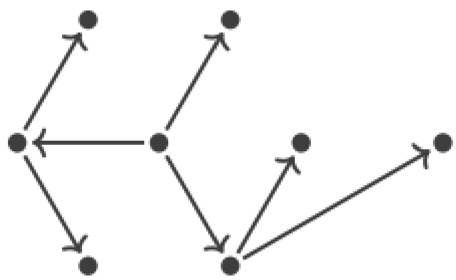
Il grado di uscita di un nodo non ha restrizioni.

Per ogni nodo v che non è la radice, esiste esattamente un cammino della radice a v e non può essere mai vuoto (la radice esiste sempre).

Se un albero è finito, allora esiste almeno una foglia (può essere anche la radice). I nodi intermedi sono sia il padre che il figlio.

→ Rappresentazione





→ Cammini in un albero

In un albero esiste esattamente un cammino dalla radice a qualunque nodo v diverso dalla radice.

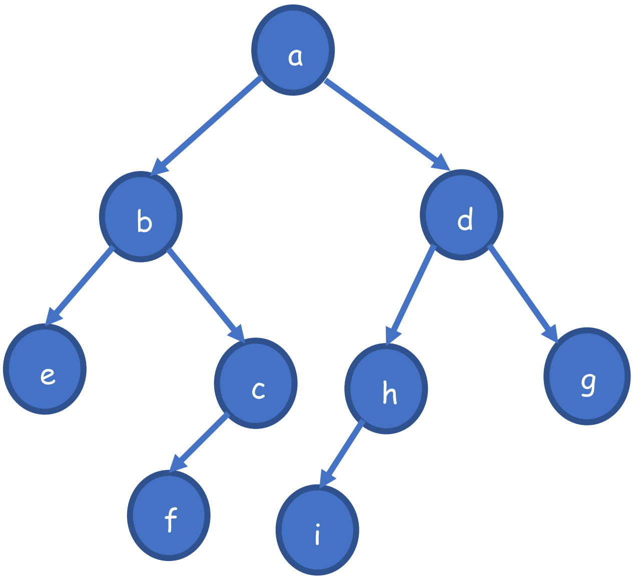
Ogni nodo w in questo cammino è un ascendente di v (avo) e v è un discendente di w (radice è l'unico nodo che non ha ascendenti).

Se il cammino da w a v ha lunghezza 1, allora w è il padre di v, e v è un figlio di w.

→ Profondità dell'albero

La profondità di un nodo v è la lunghezza del cammino della radice a v.

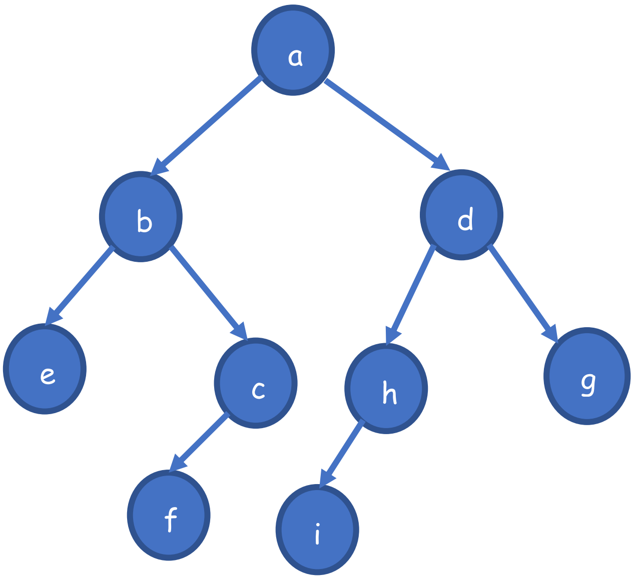
L'altezza di un albero è la profondità massima dei suoi nodi.



Nodo	Profondità
<a>	radice
<b,c,d,f>	padre
<e,g,h,i>	foglia
<e,g,h,i>	figlio

→ Albero binario

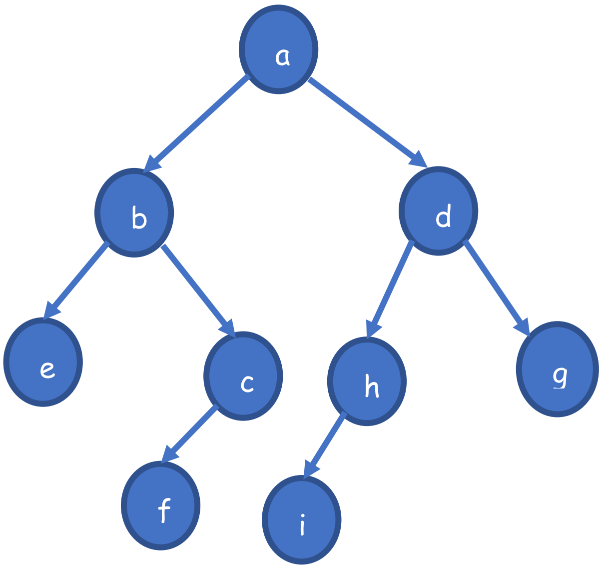
Un albero si dice binario se ogni nodo ha al più due figli. I figli di un nodo in un albero binario sono ordinati (figlio sinistro e figlio destro).



→ Proprietà degli alberi binari

Un albero binario ha al massimo 2^p nodi di profondità p.

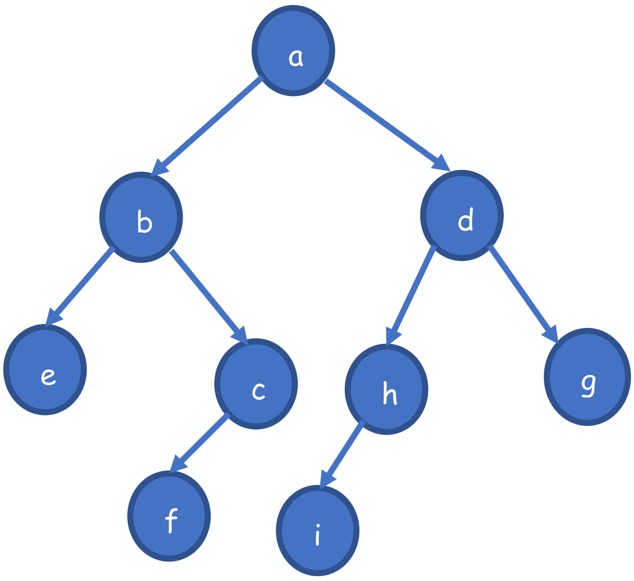
Un albero di altezza n ha al più $\sum_{i=0}^{n} 2^i = 2^{n+1}-1$ nodi.



→ Struttura albero binario

Un albero binario è una struttura ricorsiva composta da:

1) un nodo (radice);
2) un albero binario sinistro (eventualmente vuoto);
3) un albero binario destro (eventualmente vuoto).



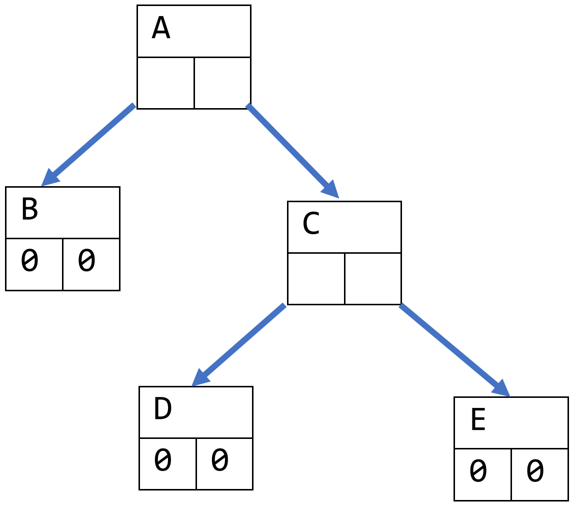
Nodo	Tipologia
<a>	radice
<b,c,e,f>	albero sinistro
<d,g,h,i>	albero destro

→ Come rappresentarli?

E' possibile rappresentare un albero binario sia come:

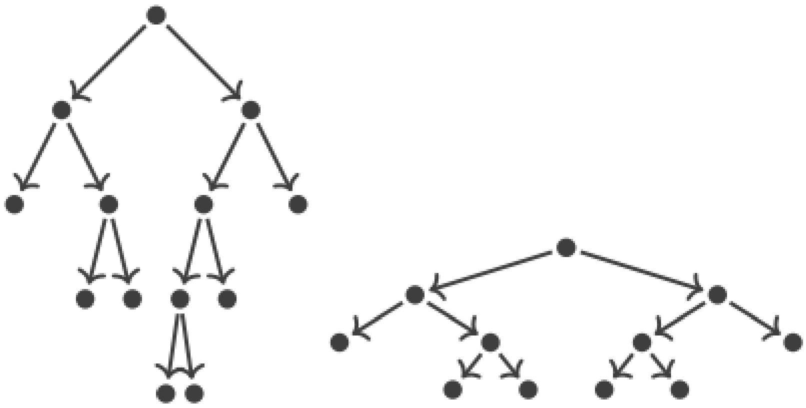
1) una collezione di nodi, dove la radice è segnalata e ogni nodo ha due puntatori (alle radici degli alberi destro e sinistro);
2) come una tabella di $2^{n+1}-1$ righe, dove n è l'altezza dell'albero.

1	A	1
2	B	1
3	C	1
4	0	0
5	0	0
6	D	1
7	E	1



→ Alberi binari pieni

Un albero binario è pieno se ogni nodo interno ha due figli.



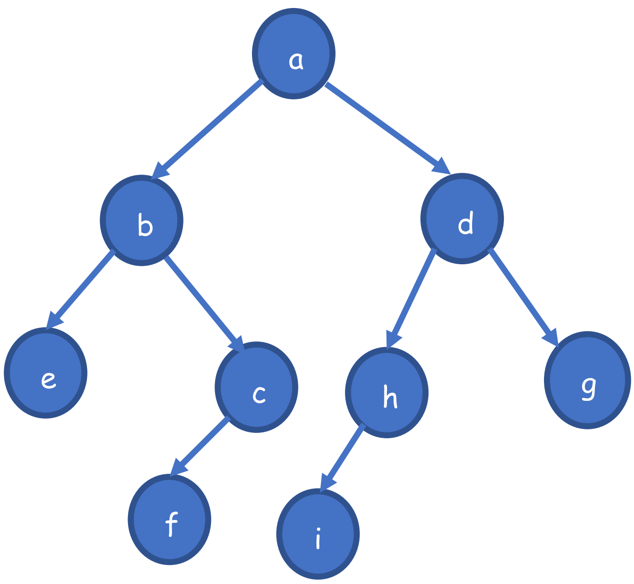
→ Alberio binario completo

Un albero binario è completo se:

1) ha altezza n;

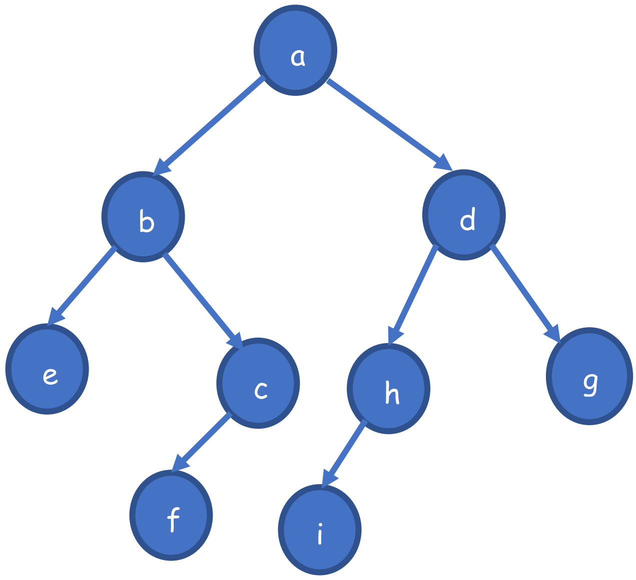
2) ad ogni profondità i, $0 \le i < n$ ci sono 2^i nodi;

3) l'ultimo livello `e riempito da sinistra a destra;



→ Albero bilanciato

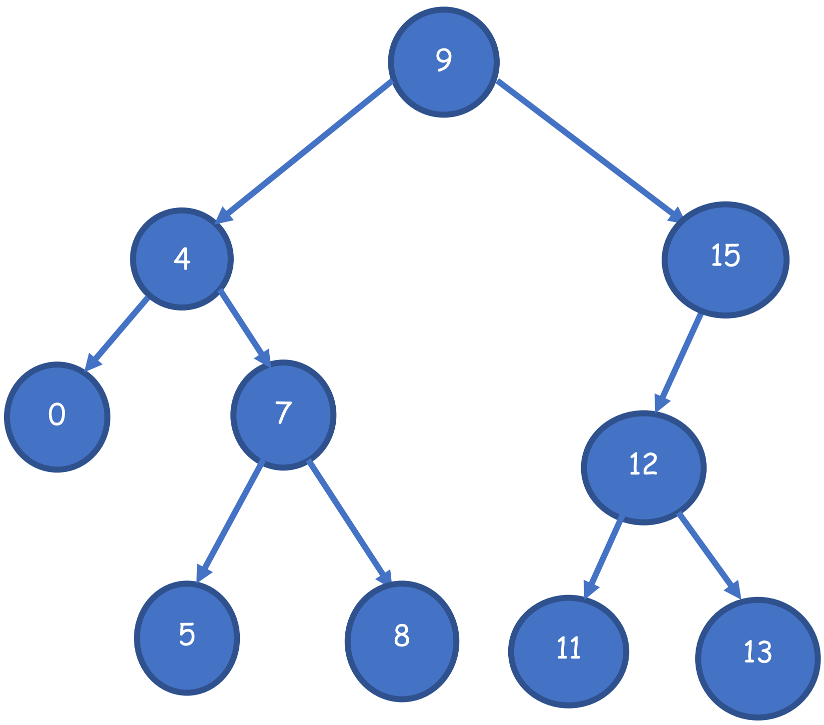
Un albero binario è bilanciato se per ogni nodo v la differenza fra il numero di nodi nell'albero sinistro di v e il numero di nodi nell'albero destro di v è al massimo 1.



→ Albero binario di ricerca

Un albero di ricerca è un albero binario G = (V, E) tale che per ogni nodo z:

1) z ∈ Z;
2) ogni nodo dell'albero sinistro di z è minore a z;
3) ogni nodo dell'albero destro di z è maggiore a z.



→ Attraversamento albero binario

Un attraversamento è un processo che visita tutti i nodi di un albero.
Una enumerazione dei nodi è un attraversamento che elenca ogni nodo esattamente una volta.

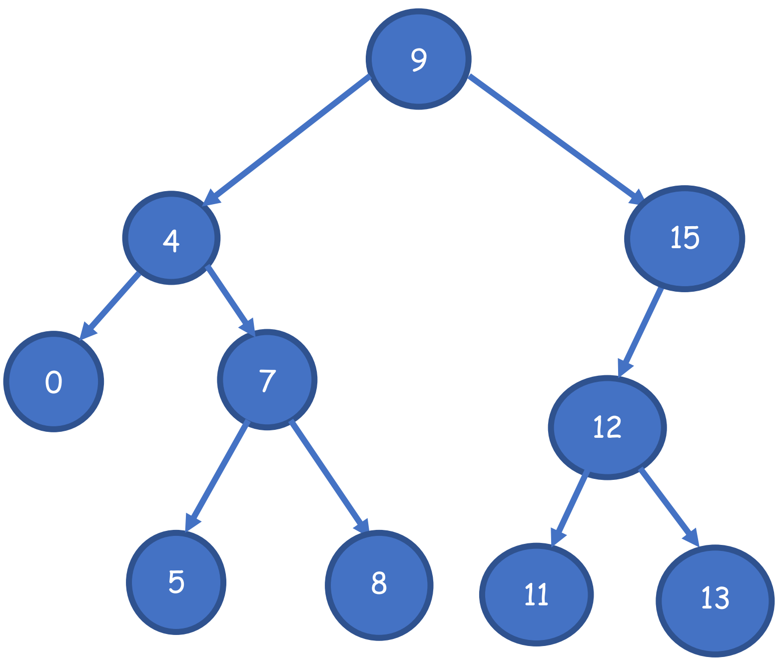
→ Tipi di attraversamento

Un attraversamento può essere di due tipologie:
1) in profondità: in cui si esplora ogni ramo dell'albero fino in fondo (figli prima dei fratelli);
2) in ampiezza: in cui si esplora prima i nodi più vicini ala radice (fratelli prima dei figli).

→ Enumerazione in profondità

Ci sono tre tipi diversi di ordini di profondità:
1) L (sinistra);
2) R (destra);
3) V (enumerazione).

I tre ordini di enumerazione in profondità sono:
1) pre – order: si visita prima la radice, poi il sotto albero di sinistra e infine il sotto albero di destra (VLR);
2) in – order: si visita prima il sotto albero sinistro, poi la radice e inifine il sotto albero di destra (LVR);
3) post – order: si visita prima il sotto albero sinistro, poi il sotto albero destro e infine la radice (LRV).



VLR

9	4	0	7	5	8	15	12	11	13

LVR

0	4	5	7	8	9	11	12	13	15

LRV

0	5	8	7	4	11	13	12	15	9

→ Numero di foglie in un albero

Un albero finito ha sempre almeno una foglia e per massimizzare il numero di foglie è necessario avere un albero pieno.
Un albero pieno con n nodi interni ha n+1 foglie.
Il numero di puntatori nulli in un albero binario con n nodi è n + 1.

→ Dimostrazione per induzione

Teorema: un albero pieno con n nodi interni ha n+1 foglie.
Per dimostrare per induzione il teorema, è necessario eseguire tre passi:
1) dimostrare il caso base;
2) specificare l'ipotesi di induzione basata su un numero n;
3) dimostrare il passo di induzione: è vero anche per n+1.

Caso base: un albero non è mai vuoto. Ha almeno 0 nodi interni e 1 foglia.
Ipotesi Induttiva: il risultato è vero per alberi con n nodi interni (hanno n+1 foglie).
Passo Induttivo: dimostrare il caso n + 1; cioè, se un albero pieno ha n + 1 nodi interni, allora ha n + 1 + 1 = n + 2 foglie.

Capitolo 10 – Ordinamenti e Reticoli

→ Definizioni

Una relazione R su un insieme S è un:
1) preordine solo se R è riflessiva e transitiva;
2) ordine parziale solo se R è un preordine antisimmetrico (riflessiva, antisimmetrica e transitiva);
3) ordine stretto solo se R è irriflessiva e transitiva (quindi anche asimmetrica).

1) Un ordine parziale viene rappresentato con ≤; uno stretto con <;
2) Un poset (insieme parzialmente ordinato) è la coppia (S, ≤).

→ Ordine totale

1) Un ordine totale è un ordine parziale fortemente connesso: per ogni $x, y \in S$: $x \leq y$ oppure $y \leq x$.

2) Un ordine totale stretto è un ordine stretto connesso: per ogni $x, y \in S$ tali che $x \neq y$: $x < y$ oppure $y < x$. Ogni coppia di elementi è paragonabile.

Esempio

1) "essere nato prima di" è un ordine stretto (non totale);

2) la relazione $\leq$ su N è un ordine totale;

3) la relazione $\subseteq$ su insiemi è un ordine parziale;

4) $<$ su R è un ordine totale stretto;

5) la chiusura transitiva di un DAG è un ordine parziale.

→ Relazione fra ordini totali

Ordini totali stretti e non-stretti sono molto vicini e se R è un ordine totale (non-stretto) allora $R \setminus I_S$ è un ordine totale stretto. Se R è un ordine totale stretto allora $R \cup I_S$ è un ordine totale.

→ Triconomia

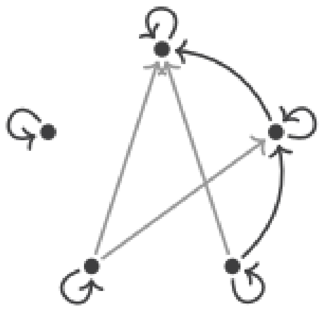
In un ordine totale stretto R per ogni $x, y \in S$ si soddisfa esattamente una fra:

1) $x = y$;

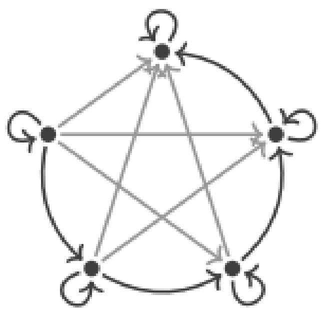
2) $\langle x, y \rangle \in R$

3) $\langle y, x \rangle \in R$

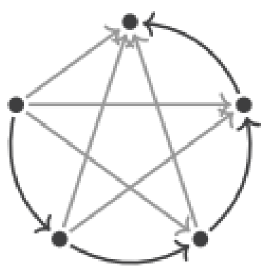
Descrizione grafica



ordine parziale



ordine totale



ordine totale stretto

→ Prodotto di ordinamenti

Siano (S, ≤S) e (T, ≤T) due poset, si definisce la relazione ≤S×T su S × T come

$\langle s, t\rangle \leq_{S\times T} \langle s', t'\rangle$ sse $s \leq_S s', t \leq_T t'$

$(S \times T, \leq_{S\times T})$ → è un poset.

→ Ordine lessicografico

Paragona tuple di elementi posizione per posizione.

La relazione ≤lex su S × T è definita da ⟨s, t⟩ ≤lex ⟨s ', t '⟩ solo se:

1) s < s*s';

2) s = s', t ≤T * t'.

$(S \times T, \leq_{\text{lex}})$ è un poset che preserva ordini totali.

→ Copertura

In un poset (S, ≤), una copertura di x ∈ S è un elemento minimo più grande di x.

L'elemento y ∈ S è una copertura di x ∈ S solo se:

1) x ≤ y, x != y;

2) non esiste z, x != z != y tale che x ≤ z, z ≤ y.

Esempio

In (N,<=)

1) 5 è una copertura di 4;

2) 6 non è una copertura di 4.

In (P(S),<=) con `S = {0,1,2}`

1) ogni singoletto è una copertura di ∅;

2) S non è una copertura di ∅;

3) S è una copertura di {0,1}, di {0,2} e di {1,2};

4) {0,1} non è una copertura di {0,2}.

→ Elementi estremali (massimali e minimali)

In un poset (S, ≤), un elemento s ∈ S' è

1) minimale: se non esiste un elemento s' != s tale che s' <= s;

2) massimale: se non esiste un elemento s' != s tale che s <= s'.

Un poset può avere nessuno, uno o tanti elementi minimali e massimali.

→ Minoranti e maggioranti

Dato un poset (S, ≤) e un insieme X ⊆ S, un elemento s ∈ S è:

1) minorante di X: solo se s <= x per ogni x ∈ X;

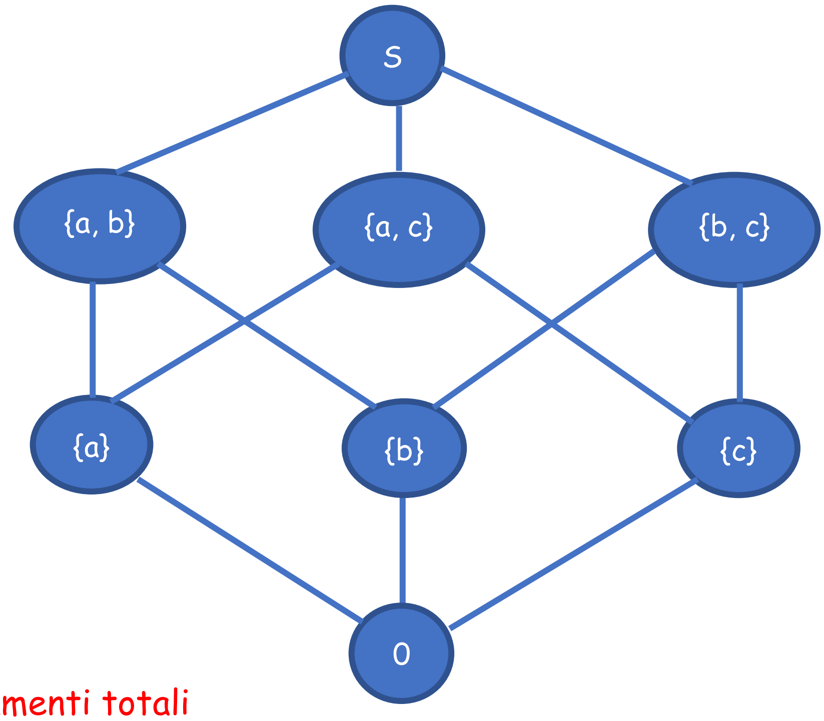
2) massimo minorante di X (⊓X): solo se s' <= s per ogni minorante s' di X;

3) maggiorante di X: solo se s >= x per ogni x ∈ X;
4) minimo maggiorante di X (⊔X): solo se s ≤ s' per ogni maggiorante s' di X;
5) minimo di X: solo se s = ⊓X ∈ X;
6) massimo di X: solo se s = ⊔X ∈ X.

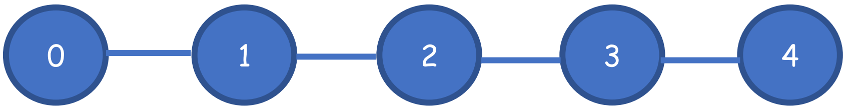
→ Proprietà
Ogni X ⊆ S ha al più un massimo minorante e un minimo maggiorante.
Se ogni X ⊆ S ha un minimo, allora (S,<=) è un insieme ben ordinato (o ben fondato).
Se esiste, ⊓S è il minimo di S, denotato da 0.
Se esiste, ⊔S è il massimo di S, denotato da 1.

→ Diagramma di Hasse
Un diagramma di Hasse è una rappresentazione compatta di un poset. Utilizza la posizione per rappresentare l'ordine e considera la riflessività e transitività implicite.
Il diagramma di Hasse è un grafo non orientato tale che per ogni x,y:
→ se x ≤ y allora x appare sotto di y;
→ x e y sono collegati solo se y è una copertura di x.
L'ordine è la chiusura riflessiva e transitiva del grafo orientato da giù verso su.

Esempio: Il diagramma di Hasse di (P(S),⊆) per S = {a,b,c} è



→ Ordinamenti totali
Il diagramma di Hasse di un ordinamento totale formerà sempre una catena.
Per esempio ({0,1,2,3,4},<=)



→ Reticolo

Il reticolo è un insieme parzialmente ordinato in cui ogni coppia di elementi ha sia un estremo inferiore (inf) che un estremo superiore (sup). I reticoli possono anche essere caratterizzati come strutture algebriche che soddisfano determinate identità.

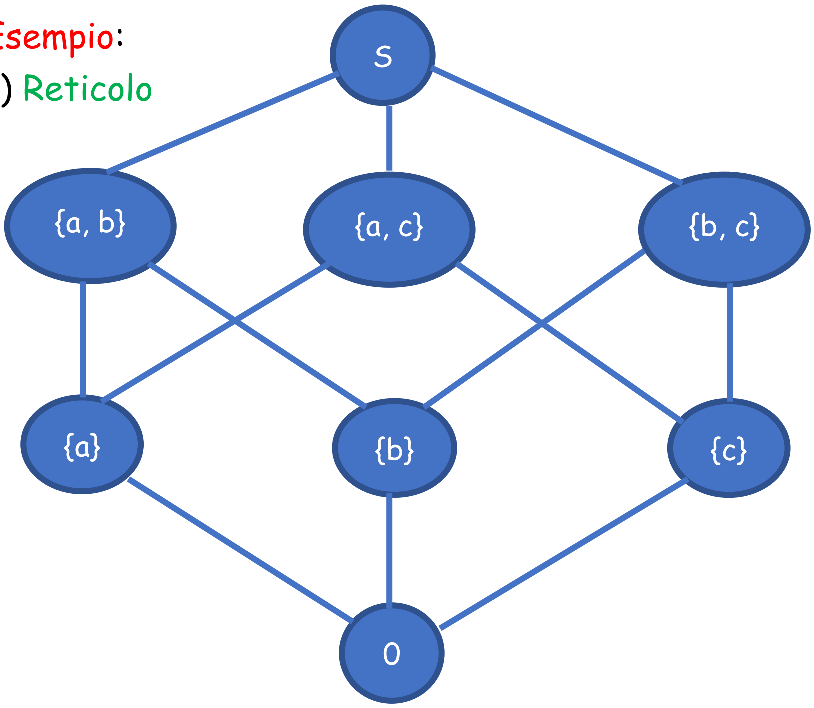
Un reticolo è un poset (S,<=) tale che per ogni x, y ∈ S:

a) esiste un minimo maggiorante $x \sqcup y$ (join);

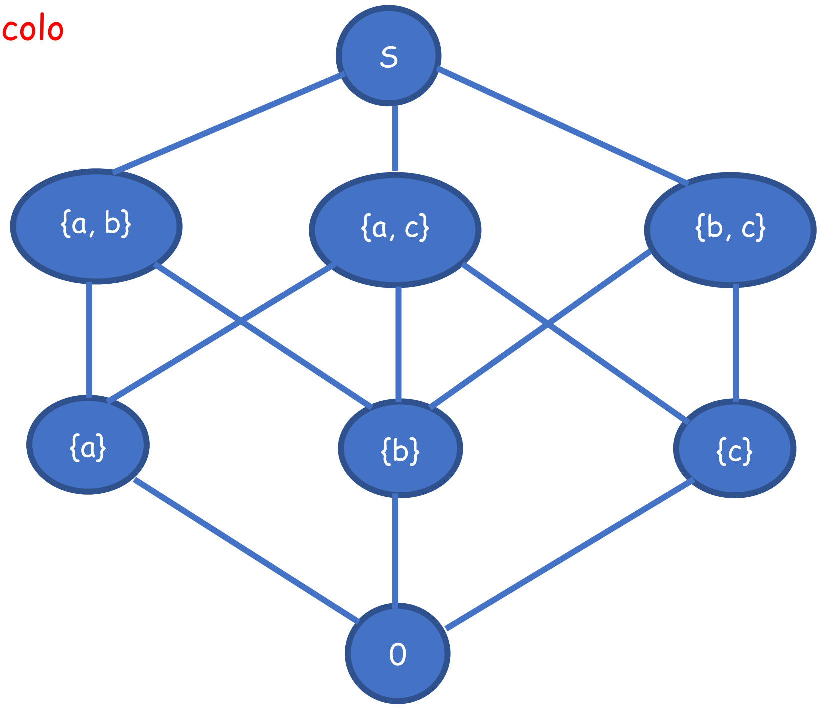
b) esiste un massimo minorante $x \sqcap y$ (meet).

Esempio:

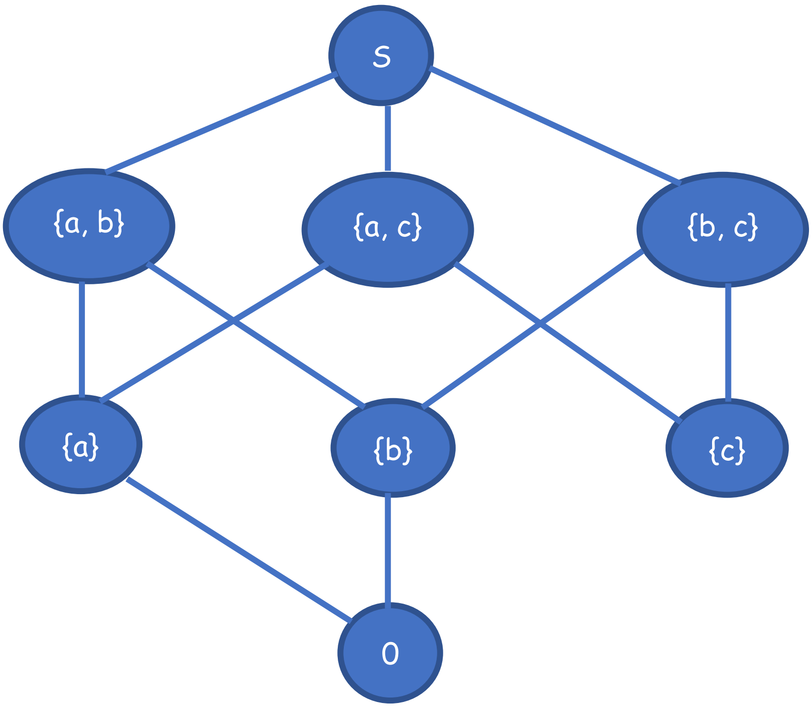
1) Reticolo



2) Non Reticolo



3) Non Reticolo



Proprietà 1

Proprietà	Descrizione
idempotenza	$a \sqcup a = a = a \sqcap a$
commutatività	$a \sqcup b = b \sqcup a$
associatività 1	$a \sqcup (b \sqcup c) = (a \sqcup b) \sqcup c$
associatività 2	$a \sqcap (b \sqcap c) = (a \sqcap b) \sqcap c$
assorbimento	$a \sqcup (a \sqcap b) = a = a \sqcap (a \sqcup b)$

Proprietà 2

Se $(L, \leq)$ è un reticolo, allora per ogni $a, b, c \in L$:

$a \leq a \sqcup b$
se $a \leq c$ e $b \leq c$ allora $a \sqcup b \leq c$
$a \sqcap b \leq a$
se $c \leq a$ e $c \leq b$ allora $c \leq a \sqcap b$
$a \sqcup b = b$ solo se $a \leq b$
$a \sqcap b = a$ solo se $a \leq b$

→ Monoticità

Le operazioni join e il meet sono monotoni; cioè, se $a \leq c$ e $b \leq d$ allora

1) $a \sqcup b \leq c \sqcup d$;

2) $a \sqcap b \leq c \sqcap d$.

→ Tipologie di reticoli

Un reticolo (L, <=) può essere:

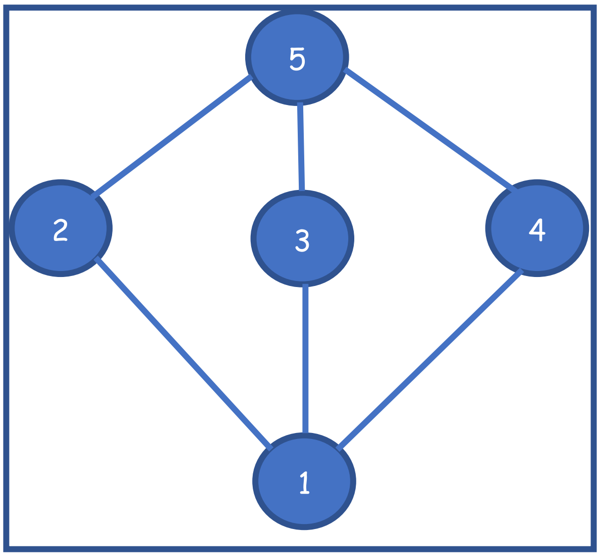
1) completo: solo se per ogni M ⊆ L esistono ⊔M e ⊓M;
2) limitato: solo se esistono <u>1</u> = ⊔L e <u>0</u> = ⊓L;
3) distributivo: solo se meet e join distribuiscono tra di loro:

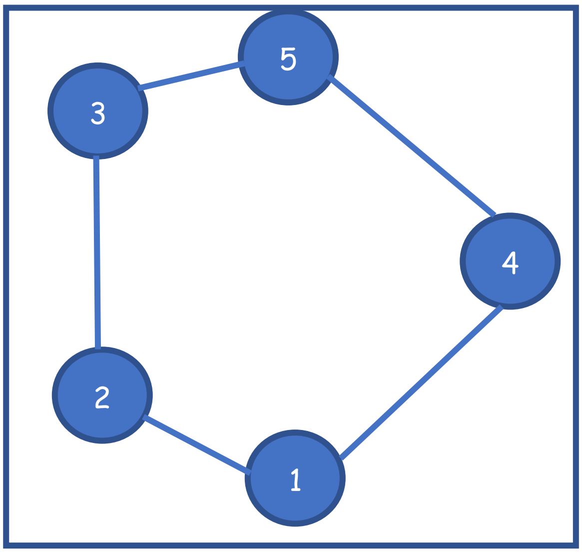
$$a \sqcap (b \sqcup c) = (a \sqcap b) \sqcup (a \sqcap c)$$
$$a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$$

Imparare: Ogni reticolo completo è limitato e ogni reticolo finito è completo e limitato.

→ Reticoli non distributivi

Esempio:





→ Complemento

Siano (L, ≤) un reticolo distributivo limitato e a ∈ L, un elemento b ∈ L è il complemento di a solo se:

a ⊓ b = <u>0</u>
a ⊔ b = <u>1</u>

Se a ∈ L ha un complemento, allora questo è unico.
(L, ≤) è un reticolo di complemento solo se ogni a ∈ L ha un complemento.

Capitolo 11 – Boole

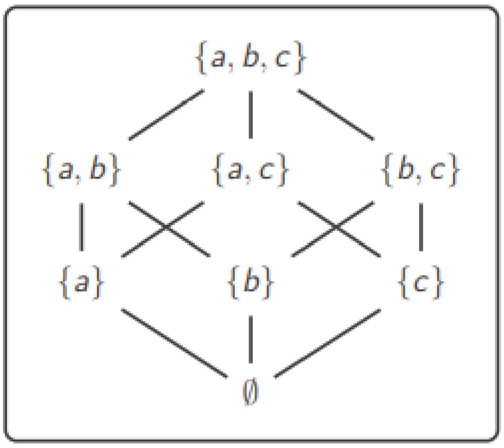
→ Reticolo booleano

Un reticolo booleano è un reticolo limitato, distributivo e complementato.

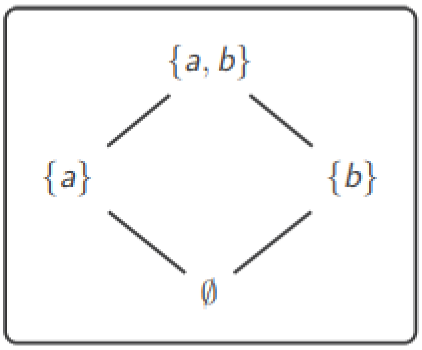
Il reticolo booleano (L, <=) definisce l'algebra di Boole $(L, \sqcup, \sqcap, \bar{\cdot}, \underline{0}, \underline{1})$ con operazioni per disgiunzione, congiunzione e negazione.

Esempio tipico:
Per ogni insieme S, il reticolo $(P(S), \subseteq)$ è un reticolo booleano.
Per ogni $T, T' \subseteq S$:
1) $T \sqcup T' = T \cup T'$;
2) $T \sqcap T' = T \cap T'$;
3) $\neg T = S \setminus T$.

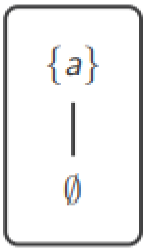
→ Reticoli di insiemi delle parti



3 elementi



2 elementi



1 elemento

→ Algebra di Boole
L'algebra di Boole "tradizionale" è quella definita dal reticolo $(P(\{a\}), \subseteq)$.
Nell'algebra di Boole si effettua la seguente associazione:
- 0 = 0
- {a} = 1

Operazioni:
- la disgiunzione è data dal join (V)
- la congiunzione è data dal meet (∧)
- la negazione è data dal complemento (¬)

→ Operazioni Logiche
Disgiunzione

A	B	A V B
0	0	0
0	1	1
1	0	1
1	1	1

Congiunzione

A	B	A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

Negazione

A	¬A
0	1
1	0

→ Proprietà Operazioni Logiche

• ∧ e ∨ sono idempotenti, commutative e associative

• ¬ è involutivo ($\neg\neg x = x$)

• ∧ e ∨ distribuiscono fra di loro

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$
$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$

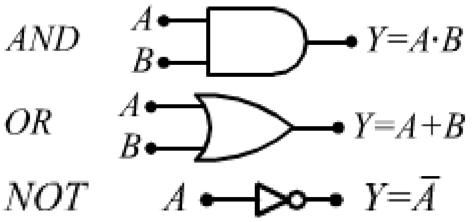
Leggi di De Morgan

$$\neg(x \wedge y) = \neg x \vee \neg y$$
$$\neg(x \vee y) = \neg x \wedge \neg y$$

→ Circuiti Logici

Le operazioni booleane possono essere implementate su porte logiche e ci permettono inoltre di manipolare valori logici in applicazioni molto complesse.

Porte Logiche



Capitolo 12 – Automi a Stato Finito

→ Descrizione

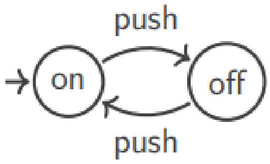
Un automa è un dispositivo che legge una sequenza di simboli, ed esegue istruzioni basati su di essi.

Essa ha tre proprietà:

1) memoria finita;
2) legge senza scrivere;
3) ha un determinato ordine di lettura (legge in ordine senza tornare indietro);
4) legge un'istruzione alla volta.

Gli automi vengono utilizzati ad esempio per progettare circuiti digitali, analizzare espressioni lessicali, cercare parole in un file, verificare sistemi temporali ed ecc.

Esempio: un interruttore



→ Elementi di un automa

Gli elementi di un automa sono:
- un alfabeto (istruzioni);
- un insieme finito di stati (memoria);
- un insieme di regole di transizione (azioni);
- uno o più stati iniziali;
- stati designati come finali.

Le regole di transizione descrivono il nuovo stato della memoria in base all'istruzione. Dopo aver letto la sequenza, può finire in uno stato finale (accetta), o no (rifiuta).

→ Definizione formale

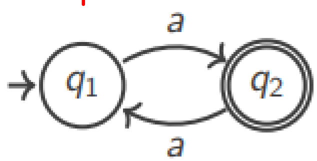
Un automa a stati finiti è una quintupla $A = \langle Q, \Sigma, \Delta, I, F\rangle$, dove:
- Q è un insieme finito non vuoto di stati;
- Σ è un insieme finito non vuoto di simboli (alfabeto);
- $\Delta \subseteq Q \times \Sigma \times Q$ è la relazione di transizione;
- $I \subseteq Q$ è l'insieme di stati iniziali;
- $F \subseteq Q$ è l'insieme di stati finali.

Esempio: $\langle\{q1, q2\}, \{a\}, \{\langle q1, a, q2\rangle, \langle q2, a, q1\rangle\}, \{q1\}, \{q2\}\rangle$

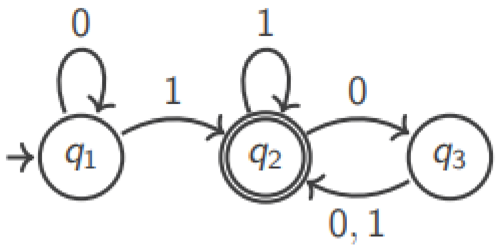
→ Rappresentazione grafica

Un automa può essere rappresentato come un grafo etichettato $\langle Q, E, \ell\rangle$ con $\ell : E \rightarrow P(\Sigma)$. I nodi del grafo rappresentano gli stati, e gli archi le transizioni. Gli stati iniziali si rappresentano con un semiarco e quelli finali con un doppio bordo.

Esempio 1:



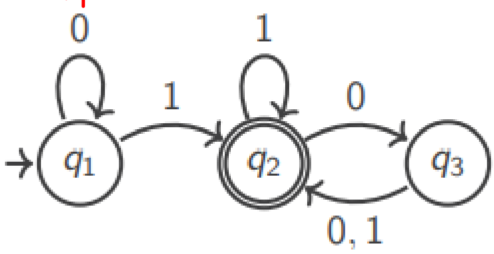
Esempio 2:



L'automa ⟨Q, Σ, Δ, I, F⟩ dove

• Q = {q1, q2, q3};
• Σ = {0,1};
• I = {q1};
• F = {q2};
• Δ = {⟨q1, 0, q1⟩, ⟨q1, 1, q2⟩,⟨q2, 0, q3⟩, ⟨q2, 1, q2⟩,⟨q3, 0, q2⟩, ⟨q3, 1, q2⟩};

Esempio 3:



W = 1101

L'automa esegue la sequenza di stati ⟨q1, q2, q2, q3, q2⟩ e finisce sullo stato finale q2; quindi la accetta.

Rifiuta la sequenza W = 0010.

→ Alfabeto

Un alfabeto è un insieme finito non – vuoto Σ e i suoi elementi vengono denominato simboli.

→ Stringa

Una stringa (o word) è una sequenza finita di simboli. Essa può essere anche vuota (ε).

→ Linguaggio

Un linguaggio è un insieme di parole può essere anche vuoto oppure infinito.

Esempio:

1) Il linguaggio {ε, 11, 1111, 111111, . . .} contiene tutte le parole di lunghezza pari sull'alfabeto {1}.
2) Il linguaggio su {a, b} con tutte le parole che iniziano con 'a' {a, aa, ab, aaa, aab, aba, abb, aaaa, aaab, aaba, aabb, . . .}.
3) Il linguaggio vuoto ∅.
4) Il linguaggio con la parola vuota {ε}.

→ Concatenazioni di parole e linguaggi

La concatenazione $x \cdot y$ di due parole x, y è la sequenza ottenuta mettendo y immediatamente dopo di x

ab · aaab = abaaab

La concatenazione $L \cdot M$ di due linguaggi L, M è il linguaggio ottenuto di concatenare ogni parola di L e ogni parola di M

```
{ε, ab, aaa} · {bb, ba} = {bb, ba, abbb, abba, aaabb, aaaba}
```

→ Stella

Le potenze M^k di un linguaggio M sono definite da:

- $M^0 = \{\varepsilon\}$;
- $M^{k+1} = M \cdot M^k$, $k \geq 0$;

I linguaggi M^* e M^+ sono:

- $M^* = M^0 \cup M^1 \cup M^2 \cup \ldots$
- $M^+ = M^1 \cup M^2 \cup M^3 \cup \ldots$

Nota: se Σ è un alfabeto, allora

- Σ^* è l'insieme di tutte le parole su Σ;
- Σ^+ è l'insieme di tutte le parole non-vuote.

Esempio

- $\{11\}^*$ è il linguaggio di tutte le parole di lunghezza pari su {1}
- $\{a\} \cdot \{a, b\}^*$ il linguaggio delle parole che iniziano con a su {a, b}
- $\{a\}^+ \cdot \{b\}^+$ tutte le parole formate da una sequenza di a seguita da una sequenza di b.

→ Linguaggi regolari (Definizione ricorsiva)

La definizione ricorsiva dei linguaggi regolari è:

- tutti i linguaggi finiti sono regolari

• se L, M sono linguaggi regolari, allora L U M, L • M, L*, L^+ sono regolari.

Esempio

$L = \{0,1\}^* \cdot \{01\} \cdot \{0,1\}^* = \{x01y \mid x, y \in \{0, 1\}^*\}$ è regolare.

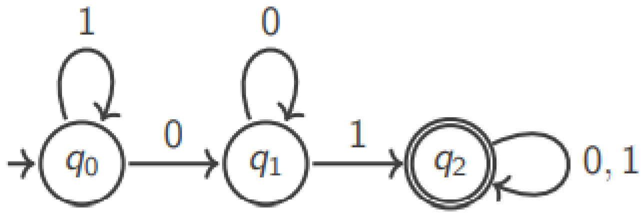
L'insieme di tutti i palindromi su un alfabeto Σ non è regolare.

→ Teorema di equialenza

Il linguaggio riconosciuto da un automa A è l'insieme delle parole accettate da A. Il linguaggio riconosciuto da un automa è regolare.

Esempio

$L = \{0,1\}^* \cdot \{01\} \cdot \{0,1\}^* = \{x01y \mid x, y \in \{0, 1\}^*\}$



⟨{q0, q1, q2}, {0, 1}, Δ, {q0}, {q2}⟩

Δ = {⟨q0, 0, q1⟩, ⟨q0, 1, q0⟩, ⟨q1, 0, q1⟩, ⟨q1, 1, q2⟩, ⟨q2, 0, q2⟩, ⟨q2, 1, q2⟩}

→ Costruzioni di automi

Il principio di induzione permette di dimostrare che un linguaggio regolare è riconosciuto da un automa.

Infatti si afferma che:

1) ∅ e {ε} sono riconosciuti (caso base);

2) si ipotizza che ogni parola sia riconosciuta (ipotesi induttiva);

3) per affermare ciò si deduce che se L e M sono riconosciuti, allora L ∪ M, L · M e L^+ sono riconosciuti (tesi).

→ Determinismo

In un automa, il processo per accettare una parola è nondeterminista. Quando si legge una parola, ci sono diverse strategie da seguire per arrivare ad uno stato finale.

In un automa determinista, a differenza dell'automa tradizionale, il processo di lettura è prefissato.

In automa determinista:

• Δ è una funzione Q × Σ → Q;

• I è un singoletto I = {q}.

→ Caratteristiche

• sia gli automi deterministi che nondeterministi accettano la stessa classe dei linguaggi (regolari);
• gli automi non deterministi si possono trasformare in automi deterministi che accettano lo stesso linguaggio;
• l'automa determinista ha bisogno di molti più stati.

Capitolo 13 – Induzione e Ricorsione

→ Induzione e Ricorsione: descrizione generale

Sono due principi matematici utilizzati per definire, studiare e manipolare oggetti e strutture complesse a partire da elementi semplici.

La loro caratterisitca principale deve essere ben fondata.

→ Ricorsione

È un principio che definisce una struttura basata in sé stesso.

Esempi:

• N: 0 ∈ N; se n ∈ N allora s(n) ∈ N;
• Fibonacci: partendo da 1, 1, un numero di Fibonacci è la somma dei due numeri di Fibonacci precedenti;
• antenato: un genitore è un antenato; l'antenato di un antenato è un antenato.

```
public static int calcola_successione(int a, int n) {
    if (n==0)
        return a;
    else
        return calcola_successione(3*a-2,--n);
}
```

→ Induzione

È un principio che si riferisce al processo di derivare (indurre) una proprietà generale a partire di casi particolari. In matematica, è un metodo di dimostrazione per gestire le strutture definite ricorsivamente.

Esempi:

• per tutti n ∈ N, 2n ∈ N;
• se un numero di Fibonacci è pari, i due numeri di Fibonacci seguenti sono dispari;
• se il genitore di una persona è una persona, allora tutti gli antenati di una persona sono persone;

→ Scopo della Ricorsione e Induzione

Lo scopo dell'uso autoreferenziale in ricorsione e induzione è utile per:

- definire insiemi, strutture dati, ecc.
- verificare proprietà di insiemi (induzione);
- descrivere metodi di calcolo e programmi su di essi (algoritmi ricorsivi).

```
public static int calcola_successione(int a, int n) {
    if (n==0)
        return a;
    else
        return calcola_successione(3*a-2,--n);
}
```

→ Definizione

Una definizione caratterizza e descrive le proprietà che distinguono un oggetto di interesse dagli altri oggetti.

Esempio:

- pari: numeri interi divisibili per 2;
- dispari: numeri interi non pari;
- primi: numeri divisibili soltanto per 1 e sè stessi.

→ Assioma

Un assioma è un principio che è considerato vero senza bisogno di dimostrarlo. Costituisce il punto di partenza per lo sviluppo e lo studio di una disciplina formale.

Esempio:

- 0 non è successore di nessun numero naturale;
- ogni numero naturale ha al massimo 1 predecessore;
- la geometria euclidea si basa su cinque assiomi.

→ Ipotesi

Un'ipotesi è una proposizione considerata temporalmente vera durante il processo di dimostrazione. L'ipotesi è fondamentale per l'induzione, ma anche utile in dimostrazioni dove ci sono diversi casi da analizzare.

→ Teorema

Un teorema è una conseguenza logica degli assiomi. Per appurare che una determinata proposizione sia un teorema, è necessario dimostrarla.

I teoremi sono anche chiamati lemma, corollario e proposizione.

→ Definizione ricorsiva

Una definizione ricorsiva necessita di:

- uno o più casi base (base della ricorsione);
- una funzione per costruire nuovi casi da quelli esistenti (passo ricorsivo).

Esempio: definizione numeri naturali N

- 0 è un numero naturale;
- se n è un numero naturale, allora anche s(n) è un numero naturale (n+1).

→ Ordine naturale

I numeri naturali hanno un ordine totale che è possibile definire ricorsivamente:

- per ogni $n \in N$, $n < n+1$
- se $n < m$ e $m < l$, allora $n < l$.

→ Buon ordinamento

In un poset $(S, \leq)$, $\leq$ è un buon ordine solo se ogni sottoinsieme non vuoto $X \subseteq S$ ha un elemento $\leq$-minimo. In questo caso si dice che S è ben ordinato o ben fondato (N è ben ordinato).

→ Principio di Induzione

Il principio di induzione afferma che una proprietà P è vera in N se e solo se:

1) è vera in 0;
2) se è vera in n, allora sarà vera anche in s(n).

Una dimostrazione per induzione in N si svolge in tre passi:

1) dimostrare il caso base ($n = 0$);
2) supporre che la proprietà sia vera per una n (ipotesi di induzione);
3) dimostrare (sotto l'II) che è anche vera per $n + 1$ (passo induttivo).

Esempio 1: Somma dei primi pari

Dimostrare che per ogni $n \in \mathbb{N}$

$$\sum_{k=0}^{n} 2k = n(n+1)$$

Caso base: $n = 0$

Per $n = 0$ abbiamo $\sum_{k=0}^{0} 2k = 0 = 0(0+1)$
quindi, la proprietà si verifica

Dimostrare che per ogni $n \in \mathbb{N}$

$$\sum_{k=0}^{n} 2k = n(n+1)$$

Ipotesi di induzione: supponiamo per vero che

$$\sum_{k=0}^{n} 2k = n(n+1)$$

Passo induttivo: dimostriamo per $n+1$

$$\left(\sum_{k=0}^{n+1} 2k = (n+1)(n+2)\right)$$

$$\sum_{k=0}^{n+1} 2k = \sum_{k=0}^{n} 2k + 2(n+1) = n(n+1) + 2(n+1)$$
$$= (n+2)(n+1)$$

Esempio 2: Somma dei primi dispari

$$\sum_{k=0}^{n}(2k+1) = (n+1)^2$$

Base: $n = 0$ $\sum_{k=0}^{0}(2k+1) = 2 \cdot 0 + 1 = 1 = (0+1)^2$

Ipotesi: supponiamo che $\sum_{k=0}^{n}(2k+1) = (n+1)^2$

Passo: dimostriamo che $\sum_{k=0}^{n+1}(2k+1) = (n+2)^2$

$$\sum_{k=0}^{n+1}(2k+1) = \sum_{k=0}^{n}(2k+1) + 2(n+1) + 1$$
$$= (n+1)^2 + 2(n+1) + 1$$
$$= ((n+1)+1)^2 = (n+2)^2$$

→ Altri esempi: il fattoriale

Il fattoriale di un numero naturale è il prodotto di tutti i numeri naturali fino a n.

$$n! = 1 \cdot 2 \cdot \dots \cdot (n-1) \cdot n = \prod_{k=1}^{n} k$$

Definizione ricorsiva di fattoriale:

```
(n+1)! = n! * (n+1)
```

Codice Ricorsivo (C++)

```
int fatt(int n) {
    if (n==0)
        return 1;
    else
        return n*fatt(n-1);
}
```

```
Inserire il valore: 5
==============================
Fattoriale = 120
---------------------------------
```

Altri esempi: calcolo lunghezza stringa

Codice ricorsivo

```
string queue(string w) {
    int j=1;
    string newstring=w.substr(j,w.length()-1);
    return newstring;
}
int lunghStringa(string w) {
    if (w=="")
        return 0;
    else
        return 1+lunghStringa(queue(w));
}
```

```
Inserire una stringa = ciao
=====================================
Lunghezza della stringa = 4
---------------------------------
```

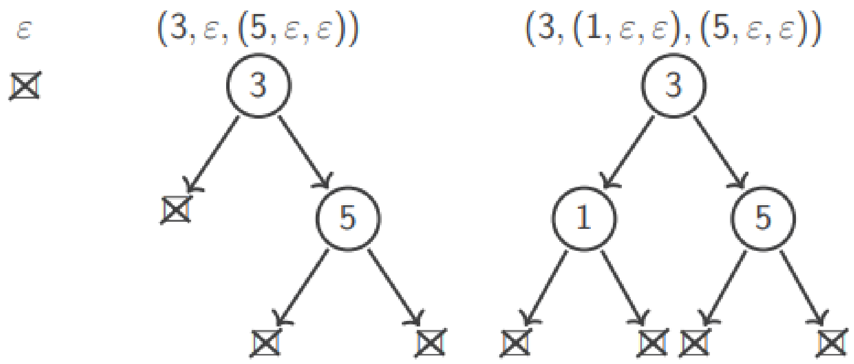
→ Alberi binari

Un albero binario è un grafo dove ogni nodo ha al più due successori e un predecessore e sono utilizzati come strutture dati.

La definizione di albero binario è ricorsiva, infatti:

1) L'albero vuoto ε è un albero binario;

2) se t1, t2 sono alberi, e n ∈ Z, allora (n, t1, t2) è un albero binario.



Altezza di un albero

- alt(ε) = 0
- alt((n,t1,t2)) = 1 + max(alt(t1), alt(t2))

Codifica in C++

```
int altezzaAlbero(tree* root) {
    if (!root)
        return 0;
    else
        return 1 + max(altezzaAlbero(root->left),altezzaAlbero(root->right));
}
```

Numero nodi albero

- nodi(ε) = 0
- nodi((n,t1,t2)) = 1 + nodi(t1) + nodi(t2)

Codifica in C++

```
int numeroNodiAlbero(tree* root) {
    if (!root)
        return 0;
    else
        return 1 + numeroNodiAlbero(root->left) + numeroNodiAlbero(root->right);
}
```

Dimostrazione per induzione del numero di nodi di un albero

Se alt(t) = n allora nodi(t) $\leq 2^n - 1$

Base: nodi(ε) = 0 = $2^0 - 1$

Ipotesi Induttiva: Supponiamo vero per ogni k, $0 \leq k \leq n$

Passo: Si dimostra che se alt(t) = n+1 allora nodi(t) $\leq 2^{n+1} - 1$.

Se alt(t) = n + 1, allora t = (z,t1,t2) e alt(t1), alt(t2) $\leq$ n

nodi(t) = 1 + nodi(t1) + nodi(t2) $\leq 1 + 2^n - 1 + 2^n - 1 =$

$2 \cdot 2^n - 1 =$

$2^{n+1} - 1$

Capitolo 14 – Logica Proposizionale

→ Generalità: Algebra booleana

Sappiamo che l'algebra di Boole più semplice (B = ({0,1}, <=)) è un reticolo complementato. Infatti si nota che:

```
• ¬0 = 1, ¬1 = 0
• 1 ⊓ 0 = 0 ⊓ 0 = 0, 1 ⊓ 1 = 1
• 0 ⊔ 1 = 1 ⊔ 1 = 1, 0 ⊔ 0 = 0
```

Si usa 0 e 1 per rappresentare i valori di verità:

```
0 -> false
1 -> true
```

Questi vengono denominati anche bit.

Le operazioni del reticolo per la manipolazione sono:
1) congiunzione (⊓): il bit risultante è pari a 1 se entrambi i bit sono pari a 1;
2) disgiunzione (⊔): il bit risultante è pari a 1 se almeno un bit è pari a 1;
3) negazione (¬): il bit risultante è 1 se il bit è 0 e viceversa.

→ Proposizioni

Data la seguente espressione booleana

$(\neg(1 \vee 0)) \wedge (1 \vee (1 \wedge 0))$

è possibile ricavere il risultato in maniera immediata (risultato è 0).
Limitarsi ad eseguire operazioni con i bit 0 e 1 può risultare piuttosto inutile e triviale. Tuttavia è possibile sostituire i bit 0 e 1 con altre variabili in B, denominate proposizioni atomiche.
Una proposizione non è altro che un'affermazione che può essere vera o falsa.
Esempio:
• gli elefanti sono mammiferi
• i rettili hanno le squame
• le biciclette portano l'ombrello
• A

→ Formula

Una formula è un'espressione booleana che combina diverse proposizioni. Una formula può anch'essa assumere valori di vero e falso in base ai valori di verità delle proposizioni che la compongono.
La logica proposizionale è una branca della logica matematica che si occupa di studiare le formule e i valori di verità.

→ Logica

La logica si riferisce ad un lignuaggio che ha ben due caratteristiche:

- preciso;
- senza ambiguità.

Un linguaggio è composto dalla sintassi (classe di espressioni permesse nel linguaggio) e semantica (significato).

→ Definizione generale di formula

Siano:

- A un insieme non vuoto di proposizioni atomiche;
- Op1 un insieme di operatori (connettivi) unari (¬);
- Op2 un insieme di operatori binari (∧, ∨).

L'insieme F di formule ben formate su (A,Op1, Op2) è definito ricorsivamente:

- $\mathcal{A} \subseteq \mathcal{F}$ (ogni proposizione atomica è una fbf)
- se $F \in \mathcal{F}$ e $* \in \mathrm{Op}_1$ allora $(*F) \in \mathcal{F}$
- se $F, G \in \mathcal{F}$ e $\circ \in \mathrm{Op}_2$ allora $(F \circ G) \in \mathcal{F}$

→ Logica proposizionale base

Sia Op1 = {¬} e Op2 = {∨,∧,→,←→}

L'insieme F di formule ben formate è definito da:

- $\mathcal{A} \subseteq \mathcal{F}$
- se $F \in \mathcal{F}$ allora $(\neg F) \in \mathcal{F}$
- se $F, G \in \mathcal{F}$ allora
  $$\{(F \wedge G),\ (F \vee G),\ (F \to G),\ (F \leftrightarrow G)\} \subseteq \mathcal{F}$$

Esempio

- A
- $((\neg A) \to (B \wedge C))$
- $(\wedge B)$
- $B \wedge C$
- $(B \wedge AC)$
- $B \wedge C \vee A$

Attenzione: Le parentesi che circondano ogni formula servono per evitare ambiguità!

Esempio: l'espressione booleana $B \wedge C \vee A$ può riferirsi a:

- $((B \wedge C) \vee A)$ oppure
- $(B \wedge (C \vee A))$

Dall'altra parte, le parentesi aumentano la difficoltà di lettura:
$(((\neg A) \to (B \vee (\neg(A \wedge (\neg C))))) \vee (\neg(\neg((\neg C) \wedge (\neg A)))))$

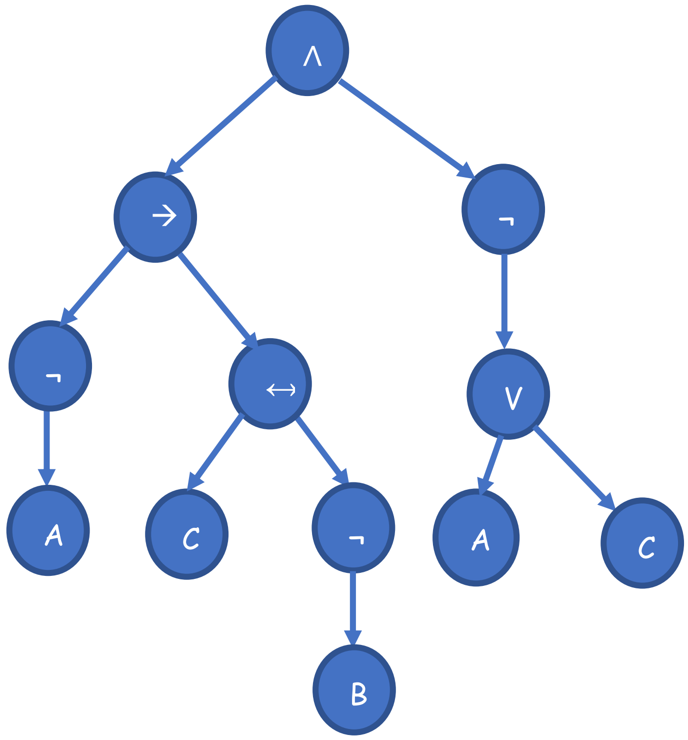
→ Precedenza tra connettivi
L'ordine con cui viene svolta un'espressione è la seguente:
$\neg \quad \wedge \quad \vee \quad \to \quad \leftrightarrow$
Esempio:

- $A \wedge B \vee C \quad \rightsquigarrow \quad ((A \wedge B) \vee C)$
- $A \to B \to C \quad \rightsquigarrow \quad (A \to (B \to C))$
- $\neg A \wedge B \to C \wedge D \quad \rightsquigarrow \quad (((\neg A) \wedge B) \to (C \wedge D))$
- $\neg A \wedge (B \to C) \wedge D \quad \rightsquigarrow \quad ((\neg A) \wedge ((B \to C) \wedge D))$

→ Terminologia
Si chiamano:
1) atomi (o variabili): gli elementi di A;
2) literali (A e ¬A) dove A è un elemento di A
- A è un literale positivo;
- ¬A è un literale negativo
3) formule gli elementi di F.

→ Unicità della scomposizione
Per ogni formula ben formata non atomica F esiste esattamente un connettivo principale $\odot$. F è formato dall'applicazione di $\odot$ a una o due formule ben formate.
Un albero sintattico rappresenta una formula ben formata.

→ Albero sintattico
$(\neg A \to (C \leftrightarrow \neg B)) \wedge \neg(A \vee C)$



Proprietà

- l'albero sintattico è sempre un albero non vuoto;
- le foglie corrispondono sempre a proposizioni atomiche;
- tutti gli altri nodi corrispondono a connettivi;
- stessa sotto formula può apparire più volte in un albero.

→ Assegnazioni e valutazioni

Un' assegnazione booleana è una funzione totale V: A → {0,1}.
Una valutazione stabilisce quali proposizioni atomiche sono vere e quali false.

→ Connettive

Negazione

A	¬A
0	1
1	0

Disgiunzione

A	B	A V B
0	0	0
0	1	1
1	0	1
1	1	1

Gianluca Rota 1 L-31 FONDAMENTI INFORMATICA 19/11/2022

Congiunzione

A	B	A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

Implicazione

A	B	A -> B
0	0	1
0	1	1
1	0	0
1	1	1

Doppia Implicazione

A	B	A <-> B
0	0	1
0	1	0
1	0	0
1	1	1

Riassunto Tavole Verità

A	B	A ∧ B	A V B	A -> B	A <-> B
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

A	¬A
0	1
1	0

Valutazioni

A	B	A ∧ B	A V B	A -> B	A <-> B
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

A	¬A
0	1
1	0

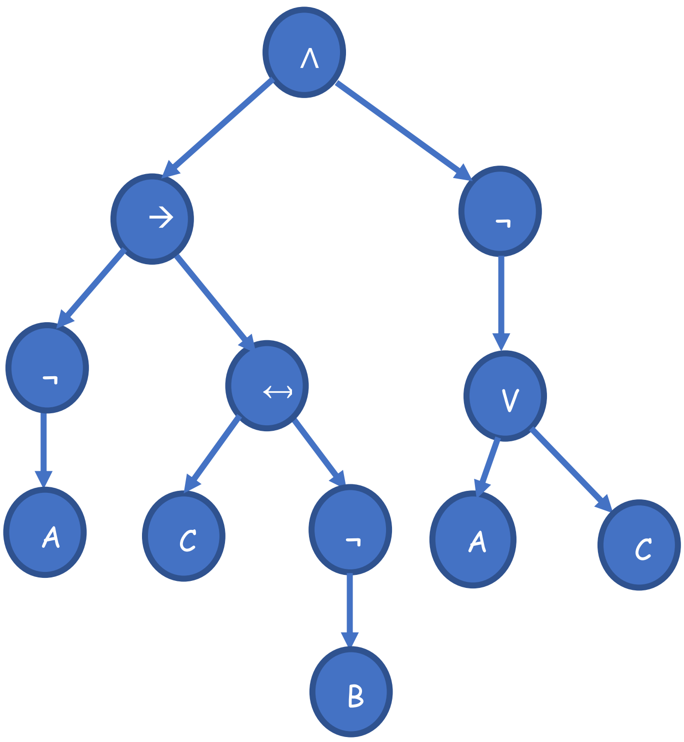
- IV(¬A) = IV(¬B) = 1
- IV(C ⇔ ¬B) = 0
- IV(¬A → (C ⇔ ¬B)) = 0
- IV(A ∨ C) = 0
- IV(¬(A ∨ C)) = 1
- IV((¬A → (C ⇔ ¬B)) ∧ ¬(A ∨ C)) = 0

V(A) = V(B) = V(C) = 0

→ Propagazione in albero sintattico

A	B	A ∧ B	A V B	A -> B	A <-> B
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

A	¬A
0	1
1	0



→ Composizionalità

Essendo che gli operatori vengono applicati nelle sotto formule, non ci si preoccupa di sapere se le formule sono atomiche o meno.

→ Equivalenze

Due formule sono equivalenti solo se non sono distinguibili tramite assegnazioni.

F e G sono equivalenti per ogni assegnazione V se IV(F) = IV(G).

F e G sono equivalenti se entrambe definiscono la stessa tavola di verità.

Esempio

A	¬A	A V ¬A	¬¬A	A ∧ A	A V A	A V ¬A -> A
0	1	1	0	0	0	0
1	0	1	1	1	1	1

$$A \equiv \neg\neg A \equiv A \wedge A \equiv A \vee A \equiv A \vee \neg A \rightarrow A$$

→ Proprietà Equivalenze

- $A \equiv \neg\neg A$ (involuzione)
- $A \equiv A \wedge A$ (idempotenza)
- $A \equiv A \vee A$ (idempotenza)
- $A \wedge B \equiv B \wedge A$ (commutatività)
- $A \vee B \equiv B \vee A$ (commutatività)
- $A \wedge B \vee C \equiv (A \vee C) \wedge (B \vee C)$ (distributività)
- $(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$ (distributività)
- $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ (associatività)
- $(A \vee B) \vee C \equiv A \vee (B \vee C)$ (associatività)

→ Legge di De Morgan

- $\neg (A \wedge B) \equiv \neg A \vee \neg B$
- $\neg (A \vee B) \equiv \neg A \wedge \neg B$

A	B	$\neg A$	$\neg B$	$A \wedge B$	$\neg(A \wedge B)$	$\neg A \vee \neg B$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

→ Assorbimento

- $A \wedge (A \vee B) \equiv A$
- $A \vee (A \wedge B) \equiv A$

A	B	$A \vee B$	$A \wedge B$	$A \wedge (A \vee B)$	$A \vee (A \wedge B)$
0	0	0	0	0	0
0	1	1	0	0	0
1	0	1	0	1	1
1	1	1	1	1	1

- $A \rightarrow B \equiv \neg A \vee B$
- $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

A	B	$\neg A$	$A \rightarrow B$	$B \rightarrow A$	$\neg A \vee B$	$A \leftrightarrow B$
0	0	1	1	1	1	1
0	1	1	1	0	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

$A \rightarrow B \equiv \neg B \rightarrow \neg A$ (contrapposizione)

→ Visione funzionale delle formule

Un altro modo di vedere le formule è in termini di funzioni. Una formula F è una funzione che associa ad ogni assegnazione V un valore $IV(F) \in \{0, 1\}$.

La tavola di verità è la descrizione estensionale di questa funzione.

Esempio:

A	B	C	F	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\neg A \wedge B \wedge C$
1	0	0	0	
1	0	1	1	$A \wedge \neg B \wedge C$
1	1	0	1	$A \wedge B \wedge \neg C$
1	1	1	0	

$F \equiv (\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C)$

→ Modello e Contro modello

Data una formula F e un'assegnazione V, si dice che V è un modello di F solo se IV(F) = 1, altrimenti si dice che V è un contro modello.

→ Tipi di Formule

Una formula viene detta:

- tautologia: se non ha contro modelli (sempre vera sotto ogni valutazione);
- contraddizione: se non ha modelli (sempre falsa sotto ogni valutazione);
- soddisfacibile non tautologica: se non né tautologica né una contraddizione.

→ Proprietà ed esempi
- $A \vee \neg A$ è una tautologia
- se $F \equiv G$, allora $F \leftrightarrow G$ è una tautologia
- $A \wedge \neg A$ è una contraddizione
- F è una tautologia solo se ¬F è una contraddizione
- $F \rightarrow G$ è una tautologia solo se $F \wedge \neg G$ è una contraddizione

Capitolo 15 – Tableaux

→ Assegnazione

Un'assegnazione è una funzione $V \rightarrow \{0, 1\}$ che definisce un valore di verità per ogni proposizione atomica.

- $\mathcal{V}(A) = \mathcal{V}(C) = 1, \ \mathcal{V}(B) = \mathcal{V}(D) = 0$
- $\mathcal{V}(A) = \mathcal{V}(B) = \mathcal{V}(C) = \mathcal{V}(D) = 0$

V è la funzione caratteristica dell'insieme.

In generale una proposizione può essere vera o falsa (dipende dall'assegnazione scelta). L'obiettivo da raggiungere consiste nel ricavare la verità di altre formule, dato un insieme (finito) di formule considerate vere.

Esempio:
Data la conoscenza:
- se Zeus è umano, allora (Zeus) è mortale;
- non è vero che: Zeus è una divinità e (Zeus) mortale;
- Zeus è una divinità;

Si conclude che Zeus non è umano.

La conoscenza si traduce nei seguenti termini logici:
p = umano;
q = mortale;
r = divinità.
- $p \rightarrow q$
- $\neg (r \wedge q)$
- r

si deduce che ¬p.

→ Conseguenze generali

La relazione di conseguenza logica può essere scritta nel seguente modo: $A \Rightarrow B$

se e soltanto se M (A) ⊆ M (B) Quando M (A) è vero allora anche M (B) è vero. Ad esempio, la formula x=0(A) consegue logicamente la formula x · y = 0(B).

Una relazione di conseguenza è una relazione $\models \subseteq \mathscr{P}(\mathcal{F}) \times \mathcal{F}$.
Si legge "F è una conseguenza di Γ" (Γ |= F).

La relazione di conseguenza è definita da una:
- classe M di interpretazioni;
- relazione di soddisfacibilità |= ⊆ M × F.

Dato M ∈ M e F ∈ F, se M |= F si dice che
- M soddisfa F;
- M è un modello di F.

In altre parole: F è una conseguenza di Γ se $\mathcal{I}_{\mathcal{V}}(F) = 1$.

Esempio precedente:
- p → q
- ¬ (r ∧ q)
- r

p	q	r	p → q	¬(r ∧ q)	r
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	1	1	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	1	1	0
1	1	1	1	0	1

→ Terminologia
Una formula è valida solo se ∅ |= F. Da un altro punto di vista è possibile affermare che se una formula F è valida solo se ogni interpretazione soddisfa F (F è una tautologia).
Cosa significa F |= G?
Significa che se F è vera, G dovrà essere necessariamente vera.

$F \models G$ sse $\models F \rightarrow G$

Dimostrazione

F |= G solo se |= F → G

Sia V un'assegnazione:

1) Se IV(F) = 0 allora per la semantica IV (F → G) = 1;

2) Se IV(F) = 1 allora V |= F e quindi V |= G; cioè IV(G) = 1. Questo significa che IV (F → G) = 1.

Se V|= F e IV (F → G) = 1, allora IV(G) = 1. Quindi V |= G.

Rappresentazione

$$\Gamma \models G \quad \text{sse} \quad \models \left(\bigwedge_{F \in \Gamma} F \right) \to G$$

→ Sistemi deduttivi

Un sistema deduttivo non è altro che un insieme di regole per manipolare formule, dette regole di inferenza.

Le regole di inferenza hanno la seguente forma:

$$\frac{F_1, \ldots, F_n}{F}$$

dove:

$F_1, \ldots, F_n$ → premesse della regola;

F → conclusione

→ Interpretazione delle regole

$$\frac{F_1, \ldots, F_n}{F}$$

Se tutte le premesse sono vere, allora la conclusione è anche vera.

Esempio:

$$\frac{F \wedge G}{F}$$

$$\frac{F \to G, F}{G}$$

$$\frac{}{F \vee \neg F}$$

→ Proprietà e terminologia

Sia $F_1, \ldots, F_n$ una dimostrazione. Questa viene chiamata dimostrazione di F_n e F_n ha una dimostrazione.

Dato un sistema deduttivo (insieme di regole di inferenza) una dimostrazione è:

una sequenza $F_1, F_2, \ldots, F_n$ di formule tale che ogni formula è la conclusione di una regola applicata a formule precedenti

Formalmente, per ogni $1 \leq i \leq n$ esiste una regola di inferenza

$$\frac{G_1, \ldots, G_m}{F_i}$$

tale che $\{G_1, \ldots, G_m\} \subseteq \{F_1, \ldots, F_{i-1}\}$

→ Teorema

Un teorema è una formula che ha una dimostrazione. F è un teorema se è derivabile nel sistema deduttivo. Ogni formula che ha una dimostrazione è un teorema.

→ Conseguenze deduttive

Sia Γ un insieme di formule e F una formula, F si deriva da Γ solo se F è un teorema del sistema deduttivo ottenuto da aggiungere tutte le formule di Γ come assiomi (in simboli $\Gamma \vdash F$). Gli elementi di Γ si chiamano premesse o ipotesi.

→ Chiusura e consistenza

La chiusura deduttiva di un insieme Γ di formule è l'insieme $Cn(\Gamma) := \{F \in \mathcal{F} \mid \Gamma \vdash F\}$ di tutte le formule derivabili da Γ.

L'insieme Γ è consistente solo se $Cn(\Gamma) \subsetneq \mathcal{F}$.

→ Inclusione e monotonia

Un sistema deduttivo è:

- inclusivo: $\Gamma \subseteq Cn(\Gamma)$ (F è una dimostrazione di F da Γ, quindi $\Gamma \vdash F$)
- monotono: se $\Gamma \subseteq \Delta$ allora $Cn(\Gamma) \subseteq Cn(\Delta)$ (una dimostrazione di F da Γ è anche una dimostrazione di F da qualunque insieme contenente Γ)
- compattezza: $\Gamma \vdash F$ solo se esiste $\Delta \subseteq \Gamma$ finito tale che $\Delta \vdash F$;
- taglio di premesse: se $\Delta \vdash F$ e $\Gamma \vdash G$ per ogni $G \in \Delta$, allora $\Gamma \vdash F$;

• deduzione: Γ, F ⊢ G solo se Γ ⊢ F → G.

→ Collegamento tra sistemi

Un sistema può essere logico con una relazione di conseguenza |= o deduttivo con una relazione di derivabilità |-.
Γ |= F: da Γ si deduce F;
Γ |- F: da Γ si dimostra F;

→ Correttezza e completezza

Un sistema deduttivo è corretto solo se per ogni F ∈ F, |- F implica |= F.
Un sistema corretto è in grado di dimostrare unicamente formule valide.
Un sistema deduttivo è completo solo se per ogni F ∈ F, |= F implica ⊢ F.
Un sistema completo è capace di dimostrare ogni formula valida.

→ Decidibilità

In un sistema deduttivo corretto e completo ogni tautologia può essere dimostrata. Tuttavia è necessario costruire un algoritmo per sviluppare dimostrazioni. Se tale algoritmo termina sempre, allora la logica è decidibile.

→ Algoritmi

Esistono diversi algoritmi utilizzati per decidere la validità di una formula.
Un altro algoritmo, finora non trattato, assume l'idea dalle regole di inferenza per decomporre una formula in pezzi più semplici. Questo algoritmo viene chiamato tableaux.

→ Tableaux

Il tableau è una classe di metodi di decisione sviluppati per diverse logiche.
La caratteristica principale del tableau è il fatto che prova a costruire modelli di una o più formule e decompone le formule in sotto formule per trovare un modello oppure per rilevare dei contro modelli.
Il tableau permette di stabilire se una formula è tautologica.
F è una tautologia solo se:
¬F è una contraddizione
¬F non ha modelli

→ Idea di base del tableau

Per decidere se F è una tautologia si esegue il procedimento indicato sotto:
• negare F;
• provare a costruire un modello di ¬F;
• se esiste un modello, F non è tautologica; altrimenti lo è.

Per decidere se F è una contraddizione si esegue il procedimento indicato sotto:

- provare a costruire un modello di F;
- se esiste un modello, F non è una contraddizione; altrimenti lo è.

→ Decomposizione

Per costruire un modello, un tableau assegna valori di verità alle sotto formule mantenendo la semantica. Dal valore di una formula, deduce quello delle sotto formule fino ad arrivare ad una assegnazione (modello) o una contraddizione. Per rappresentare i valori di verità, si utilizza T: φ e F: φ dove φ è una formula.

Esempio:

1) Si vuole un modello per ¬ (p → q)

T: ¬(p->q)
F: p->q
T:p, F:q

2) Si vuole un modello per p ∧ ¬p

T: p ∧ ¬p
T: p, T:¬p
T:p, F:p

3) Si vuole un modello per p ∧ (¬p ∨ q)

T: p ∧ (¬p V q)	
T: p, T: ¬p V q	
T:p, T:¬p	T:p, T:q
T:p, F:p	

→ Regole Tableaux

Negazione

T: ¬p		F: ¬p
F: p		T: p

Congiunzione e Disgiunzione

$$\frac{T : \varphi \wedge \psi}{T : \varphi, T : \psi} \qquad \frac{F : \varphi \wedge \psi}{F : \varphi \mid F : \psi}$$

$$\frac{T : \varphi \vee \psi}{T : \varphi \mid T : \psi} \qquad \frac{F : \varphi \vee \psi}{F : \varphi, F : \psi}$$

Implicazioni

$$\frac{T : \varphi \rightarrow \psi}{F : \varphi \mid T : \psi} \qquad \frac{F : \varphi \rightarrow \psi}{T : \varphi, F : \psi}$$

$$\frac{T : \varphi \leftrightarrow \psi}{T : \varphi, T : \psi \mid F : \varphi, F : \psi} \qquad \frac{F : \varphi \leftrightarrow \psi}{T : \varphi, F : \psi \mid F : \varphi, T : \psi}$$

→ Descrizione algoritmica

Il metodo di tableau:

• comincia con una formula e un valore di verità associato;

• ad ogni passo, sostituisce una formula con una o due formule, formando uno o due rami;

• si ferma quando tutte le formule sono atomiche.

Il ramo del tableau è aperto solo se non ha una contraddizione atomica. I rami aperti rappresentano assegnazioni che garantiscono il valore di verità richiesto.

→ Terminazione

Il metodo del tableau termina dopo un numero finito di passi.
La profondità è limitata dal numero di connettive nella formula e ad ogni livello, al massimo si duplica il numero di rami. Il tableau è un algoritmo di decisione.

Capitolo 16 – Logica Predicativa

→ Definizione

La logica dei predicati è un ramo della logica matematica che si occupa di introdurre nomi per individui e per predicati e le variabili sono solo quelle individuali.
Gli elementi della logica predicativa sono costanti, simboli relazionali, simboli funzionali e variabili.

→ Semantica

La semantica è basata da interpretazione che è un "mondo possibile" che esprime un potenziale per tutti i simboli.

→ Interpretazione

Una interpretazione ha un dominio Δ (non vuoto) e una funzione di interpretazione ^{I} che legge ogni simbolo:

- una costante in un elemento di Δ;
- un simbolo relazionale n-ario in una relazione in Δ^n;
- un simbolo funzionale n-ario in una funzione $\Delta^n \rightarrow \Delta$;
- variabili non sono interpretate: dipendono dal quantificatore.

→ Valori di verità

La formula è un predicato (una frase) che può essere vero o falso. La verità di un predicato dipende dall'interpretazione che viene considerata per la logica. Come nella logica proposizionale, esistono tautologie e contraddizioni.
La logica non legge il linguaggio naturale (italiano).

→ Rappresentazione della conoscenza

La logica predicativa viene utilizzata per descrivere il mondo di interesse e le formule vengono utilizzate come assiomi che devono essere veri. Nel mondo della logica predicativa, le interpretazioni che le falsificano cono irrilevanti.

→ Dominio e Realtà

Le formule descrivono un dominio che può essere collegato o meno con la realtà.

→ Conoscenza incompleta

Solitamente, quando si rappresenta una conoscenza, si impongono limiti alla classe di interpretazioni di interesse e molte interpretazioni soddisferanno questi assiomi. Tuttavia si potrebbe aggiungere conoscenze più dettagliate e, per questo motivo che la base di conoscenza è sempre incompleta.

→ Compromessi

Se si aggiungono conoscenze per rappresentare al meglio un dominio si rischia di complicare la base di conoscenza. Per questo motivo che è necessario trovare il giusto punto intermedio.

→ Obiettivo: trasformare le formule in linguaggio naturale

Trasformare la conoscenza umana in un insieme di formule può presentare un problema: il linguaggio naturale è ambiguo, mentre la logica non lo è.

→ Leggiamo i simboli

I simboli hanno una funzionalità specifica:
• costanti parlano di entità;
• relazioni parlano di tuple con una proprietà;
• funzioni ci ritornano una nuova entità;
• variabili "unificano" con entità in base al bisogno.
È fondamentale comprendere la funzione di ogni elemento per costruire formule adatte.

Esempio: il Grande Puffo

- ha_barba(grande_puffo)
- $\exists x.(\text{Puffo}(x) \wedge \text{ha\_barba}(x))$
- $\exists x.(\text{Capello}(x) \wedge \text{indossa}(\text{grande\_puffo}, x) \wedge \text{colore}(x, \text{rosso}))$
- colore(capello_di(grande_puffo), rosso)
- $\forall x.(\text{Puffo}(x) \rightarrow \exists y.(\text{Capello}(y) \wedge \text{indossa}(x, y)))$
- $\forall x.(\text{colore}(\text{capello\_di}(x), \text{rosso}) \rightarrow x = \text{grande\_puffo})$

Una possibile interpretazione sarebbe:
- Puffetta indossa un vestito
- Soltanto Puffetta indossa un vestito
- Tutti i capelli sono bianchi o rossi
- Tutti i puffi indossano un capello
- Ogni puffo ha una casa propria
- Ogni puffo ha una sola casa
- Non ci sono puffi cattivi

→ Tableaux

Dopo che abbiamo costruita una base di conoscenza, si vogliono dedurre conseguenze su di essa. Il processo di ragionamento non è altro che la dimostrazione di una tautologia. È necessario sviluppare un metodo per dimostrare che una formula è una tautologia.
Anche per la logica predicativa è possibile adottare il tableau, che è un metodo che genera modelli (interpretazioni che soddisfano la formula). Nel caso

predicativo, però, si tengono conto del domino e delle entità anonime.

Esempio

$\exists x.P(x)$

$\exists x.P(x) \wedge \forall y.\neg P(y)$

→ Svantaggio della logica predicativa: indecidibilità!!

La logica dei predicati è indecidibile. Non esiste nessun metodo che garantisce trovare tutte (e solo) le tautologie. Il tableau può non terminare mai, perciò server un minimo di creatività e attenzione in più.

→ Restrizioni

È possibile semplificare la descrizione considerando formule senza simboli funzionali. I tableaux non sono diversi dalle costanti, quindi non aggiungono nulla al metodo.

→ Regole

1) È possibile considerare tutti i predicati senza variabili, come proposizioni atomiche

$P(a) \wedge R(b, a) \rightarrow P(c)$ $\qquad$ $pa \wedge rba \rightarrow pc$

2) Le regole per le connettive logiche predicative si comportano come nel caso tradizionale.

→ Esistenziali Positivi

Se si ha la formula T: ∃x. Φ(x), il tableau la sostituisce per T: Φ(a), con a nuova costante.

→ Esistenziali Negativi

La formula F: ∃x. Φ(x), dice che nessun elemento del domino soddisfa Φ. Essa è sostituita per F: Φ(a), F: ∃x. Φ(x), dove a è una costante.

→ Universali negativi

Se abbiamo una formula F: ∀x. Φ(x), il tableau la sostituisce per F: Φ(a), con a nuova costante.

→ Universali Positivi

La formula T: ∀x. Φ(x), dice che tutti gli elementi del dominio devono soddisfare Φ. Dobbiamo introdurre Φ(a) per ogni costante a nel tableau.

La formula T: ∀x. Φ(x) è sostituita per T: Φ(a), T: ∀x. Φ(x), dove a è una costante nel tableau.

→ Processo del Tableau

Le regole del tableau si applicano finché non possono essere più applicato. Il tableau p aperto se non c'è una contraddizione ovvia (senza variabili). Ogni tableau completo aperto rappresenta l'esistenza di un modello.