

F

D

T

Insiemi

\emptyset = insieme vuoto

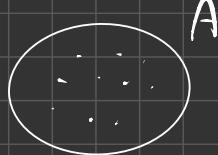
Si rappresentano in due modi

1) estensionalmente

$\{x, y, z\}$ elementi distinti e
ordine irrilevante

$\{x\}$ = singupletto, unico
elemento è x

a) Euler-Venn



\cup = unione

\cap = intersezione

Un insieme può avere anche altri insiemi come elementi

$\{\emptyset, 2, \{3\}\}$

2) intensionalmente

formula una proprietà caratteristica che distingue precisamente gli elementi dell'insieme

$$S = \{x \mid P(x)\}$$

insieme di tutti x elementi tali che
 $P(x)$ è vera

Insieme potenza

$P(x)$ = insieme che contiene tutta i possibili sottinsiemi

$$|P(x)| = 2^{|x|} \quad \text{cardinalità (quanti elementi)}$$

RELAZIONI

RST

equivalenza: +; flessiva, transitiva e simmetrica

parziale: riflessiva, antisimmetrica e transitiva

PROPRIETÀ

Riflessiva: ogni elemento dev'essere in relazione con se stesso $\langle x, x \rangle$

↓
ogni elemento ha un cappio

Irriflessiva: nessun elemento è in relazione con sé stesso \rightarrow non ci sono cappi

Simmetrica

$\langle \text{luca}, \text{giorgio} \rangle$

$\langle \text{giorgio}, \text{luca} \rangle$

$\{\langle \text{luca}, \text{giorgio} \rangle, \langle \text{giorgio}, \text{luca} \rangle\}$

POSET:
rifless - antisimm - transitivo

Asimmetrica: tutte le tuple non sono simmetriche

Antisimmetrica: tutte le coppie sono dissimmetriche tranne al più quelle in relazione con sé stesse (i cappi sono concessi)

↓ $\langle x, x \rangle$

$\nabla x R x \wedge x R x \Rightarrow x = x$

Transitiva

$\langle -x, x \rangle \langle x, x \rangle$

↓

$\langle -x, x \rangle$

$\langle x, x \rangle \langle x, x \rangle$

↓

$\langle x, x \rangle$

devo accodare le tuple: guardo la fine

della prima e associo la seconda con stesso valore

$\langle -x, x \rangle \langle x, x \rangle$

↓
devo fare il domino

INVERTIBILITÀ RELAZIONE

Una relazione è invertibile quando è iniettiva (al più 1 elemento)

↓ In parole povere

Se un elemento x punta a più elementi non è una funzione

IN CASO POSITIVO

→ Scambio l'ordine delle tuple

$\langle -x, x \rangle \quad \langle x, x \rangle$
diventa

$\langle x, x \rangle \quad \langle -x, x \rangle$

ORDINAMENTO DI RELAZIONI

* Stretto: IRRIFFLESSIVA, TRANSITIVA, ASIMMETRICA (RTA)

* equivalenza: dev'essere RIFLESSIVA, TRANSITIVA e SIMMETRICA (RTS)
↓
(totale)
è un'uguaglianza tra oggetti

* parziale: RIFLESSIVA, ANTISIMMETRICA, TRANSITIVA (PAT)

* preordine: RIFLESSIVA, TRANSITIVA (PT)

GRAFI

Un grafo ha diversi modi:

- Sorgente: solo uscite, niente entrate
- pozzo: solo entrate, niente uscite
- isolato: non ha né entrate né uscite
- adiacenti: ?

I modi possono avere un GRADO

* uscita: numero di uscite dal nodo

* ingresso: numero di entrate nel nodo

* adiacenti: ?

→ Cammino: percorso da un nodo ad un altro, seguendo l'ordine delle frecce } lunghezza = numero Archi - 1
→ Semicammino: non si segue l'ordine

→ Ciclo: un cammino da un punto a quello stesso punto

Semi-ciclo: un semicammino da un punto a quello stesso punto

→ DISTANZA: il cammino più corto (ottimale) tra un punto x e uno y

□ Se non esiste un cammino tra x e y allora la distanza è infinita

Tipi di grafo:

* orientato: relazioni binarie, coppie ordinate (ci sono le frecce di orientamento)

* non orientato: coppie non ordinate, relazioni simmetriche (non ci sono frecce)

* sottografo: grafo ricavato da un grafo maggiore

* DAG: orientato senza cicli, non si può tornare ad uno stesso modo

esempio: ALBERO

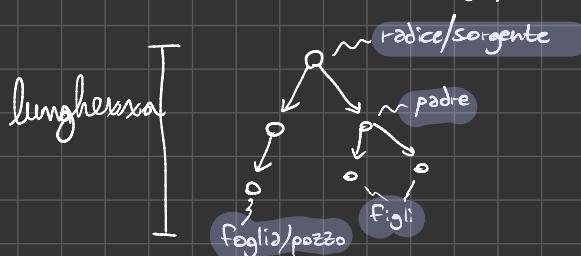


* etichettati: ogni arco (frecchia) ha un'etichetta

* Completo: irriflessivo (nessun coppia), collega ogni modo con tutti gli altri modi, ma non con sé stesso



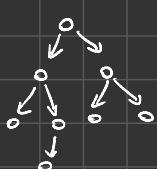
Un albero è un DAG, un grafo orientato senza cicli



grado: 0 se radice, tutti ingresso gli altri 1

padre/figlio = ascendente/descendente

ALBERO BINARIO: massimo due figli

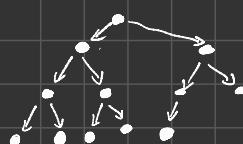


struttura ricorsiva: ogni nodo padre funge da radice di un sotto albero

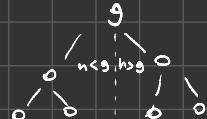
ALBERO BINARIO PIENO: solo 2 figli, non di meno

COMPLETO: ci sono sempre 2^i nodi

BILANCIATO: albero destro e sinistro differiscono di massimo 1



RICERCA: alberi binari con percorsi ottimizzati: la radice divide l'albero per $n < x$ e $x > n$ a destra e sinistra



FUNZIONI

→ N.B.: 0 è un numero naturale pari

QUANDO è TOTALE

Se non ha problemi di continuità nel dominio (~~magari~~ sempre nel dominio)

QUANDO è INIETTIVA

Ao uno e un solo valore corrisponde un valore (faccio le prove)

QUANDO È SURGETTIVA

Copre tutto il codominio

QUANDO È INVERTIBILE

→ Se è iniettiva

IMMAGINE

Specifico l'immagine e la sua caratteristica (es: Numeri dispari)

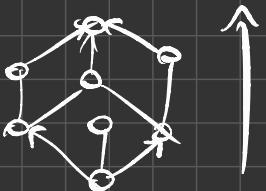
COMPOSTA

$$f \circ g = f(g(x))$$

Controllo compatibilità codominio g e dominio f (cod. dev'essere compreso nel dom.)

HASSE

RIFLESSIVITÀ E TRANSITIVITÀ IMPLICITE



si legge così

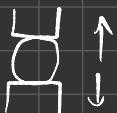
RAPPRESENTAZIONE DI UN POSET
↳ insieme parziale

TROVARE max/min (disegno le frecce per capire meglio)

- max = il punto del diagramma, quello più in alto dove finiscono tutte le frecce
- min: la sorgente più bassa, parte tutto e non arriva niente

OPERAZIONI

\sqcup = join



\sqcap = meet

join: $g \sqcup e$

- 1) trovo tutti i nodi raggiungibili da g ed e (verso l'alto)
- 2) trovo i nodi in comune e stabilisco il min (più piccolo, sorgente)

meet: $b \sqcap e$

- 1) trovo tutti i nodi raggiungibili (verso il basso)

- 2) trovo i nodi in comune e stabilisco il massimo (pozzo)

E' UN RETICOLO? \rightarrow poset (RAT) con MAX e MIN

- 1) Trovo le coppie non collegate: Vado sempre verso l'alto o verso il basso
- 2) Faccio i vari meet e join
- 3) Se dimostro che per le coppie non in relazione esistono sempre meet e join



è un reticolo

COME TROVARE IL COMPLEMENTO DI UN NODO DI UN RETICOLO

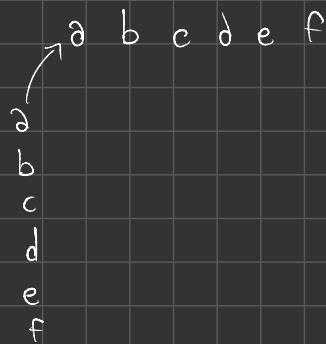
Il complemento y di un nodo x è un nodo tale che

- $x \sqcap y = \text{minimo} (\text{del reticolo})$
- $x \sqcup y = \text{massimo} (\text{del reticolo})$

→ Penso cerco i nodi con cui x non è connesso direttamente, faccio meet e join e verifico che sia rispettivamente min e max

GRAFI

MATRICE DI ADIACENZA



$0 = \text{non collegato}$
 $1 = \text{collegato}$

CAMMINO: Strada tra un nodo
e un altro (senza ripetizioni)

TAVOLA DI VERITÀ

$$\xrightarrow{\text{implica}} = \leq$$

$0 \wedge 0 = 0$	$0 \vee 0 = 0$	$\neg 0 = 1$
$0 \wedge 1 = 0$	$0 \vee 1 = 1$	
$1 \wedge 0 = 0$	$1 \vee 1 = 1$	
$1 \wedge 1 = 1$	$1 \vee 0 = 1$	$\neg 1 = 0$

Ultima colonna

- { solo 0 CONTRADDIZIONE}
- { solo 1 TAUTOLOGIA}
- mix SODDISFACIBILE NON TAUTOLOGIA

Centro modelli:

- 1) guardo gli 0 dell'ultima colonna
 - 2) guardo gli 1 corrispondenti $\xrightarrow{\text{true}}$ e li segno come centro modelli
- ↪ se non ci sono allora segno \emptyset (insieme vuoto)

TABLEAUX

LEGENDA

$A \rightarrow B$: A implica B

$B \vee C$: B oppure C

\neg : not (negazione)

\wedge : and (congiunzione)

\vee : or (oppure)

ORDINE DI PRECEDENZA (TABLEAU)

1) implica

2) and / or

3) not

COME SI FA

→ CIRCOLA (si procede dall'esterno)

→ funzioni

$$T: A \wedge B \Rightarrow T:A, T:B$$

$$F: A \wedge B \Rightarrow F:A \mid F:B$$

$$T: A \vee B \Rightarrow T:A \mid T:B$$

$$F: A \vee B \Rightarrow F:A, F:B$$

$$T: \neg A \Rightarrow F:A$$

$$F: \neg A \Rightarrow T:A$$

$$T: A \rightarrow B \Rightarrow F:A \mid T:B$$

$$F: A \rightarrow B \Rightarrow T:A, F:B$$

$$T: A \leftrightarrow B \Rightarrow T:A, T:B \mid F:A, F:B$$

$$F: A \leftrightarrow B \Rightarrow T:A, F:B \mid F:A, T:B$$

Come procedere

In base al tableau, stabilire se partire con T o F.

Se, concluso i rami del tableau chiudono $\left\{ \begin{array}{l} F: \text{autologia} \\ V: \text{Contraddizione} \end{array} \right.$

(ovvero non ci sono contraddizioni)

Se invece alla fine resta aperto qualcosa allora ristudio il tableau nell'altro ramo. → Se anche l'altro ramo rimane aperto allora è SODDISFACIBILE NON TAUTOLOGICA

TABLEAUX - PREDICATIVI

$\forall x$ = per ogni x

$\exists y$ = esiste y

$T: \forall x \Rightarrow$ elimino e sostituisco con var. esistente

$T: \exists y \Rightarrow$ elimino e sostituisco con nuova var.

$F: \forall x \Rightarrow$ elimino e sostituisco con nuova var.

$F: \exists y \Rightarrow$ elimino e sostituisco con var. esistente

LOGICA PREDICATIVA

Costanti: nomi

Predicati: sindaco(x) : x è un sindaco
cucina(x, y) : la persona x cucina altra y)

} restituiscono T/F

Funzioni: restituiscono uno ed un solo valore