Appunti di Fondamenti

Fondamenti dell'Informatica (prof. Peñaloza) - CdL Informatica Unimib - 23/24

Federico Zotti

2023-12-13

Indice

1 Matematica discreta

> **Discreto**: composto di elementi distinti, separati tra di loro.

Un sistema è:

- **Discreto** se è costituito da elementi isolati
- **Continuo** se non ci sono *vuoti* tra gli elementi

I sistemi informatici si basano su un sistema *binario*, perciò discreto.

Possiamo approssimare un sistema continuo dividendolo in piccole parti (*discretizzazione* o *digitalizzazione*).

1.1 Fasi della matematica discreta

- **Classificazione**: individuare le caratteristiche comuni di entità diverse (*teoria degli insiemi*)
- **Enumerazione**: assegnare ad ogni oggetto un numero naturale (*contare*)
- **Combinazione**: permutarne e combinarne gli elementi (*grafi*)

Queste fasi guidano un **algoritmo**.

1.2 Logica

In filosofia, la **logica** è lo studio del ragionamento, dell'argomentazione, e dei procedimenti **inferenziali** per distinguere quelli *validi* da quelli *non validi*.

La **logica matematica** vede questi procedimenti come calcoli formali, con una struttura algoritmica.

Infatti, è tutto basato sull'**algebra di Boole**.

1.2.1 Algebra astratta

L'algebra astratta studia le **strutture algebriche**, ovvero insiemi muniti di operazioni.

2 Insiemi e Operazioni

2.1 Numeri

2.1.1 Numeri naturali

I numeri **naturali** sono i primi che impariamo, e nascono dall'attività di contare.

Essi formano un **insieme**, chiamato *insieme dei numeri naturali* ($\mathbb{N}$).

$$\mathbb{N} = \{ 0, 1, 2, 3, 4, \dots, n, n+1, \dots \}$$

Contare non è altro che assegnare ad ogni oggetto un numero naturale (in ordine).

$\mathbb{N}$ ha un *limite inferiore* (0), ma non ha un *limite superiore*, quindi $\mathbb{N}$ è infinito.

2.1.1.1 Definizione semiformale

- I numeri naturali hanno l'elemento 0
- Ogni elemento n ha (**esattamente**) un successore $s(n)$
- 0 non è un successore di nessun elemento
- Due elementi diversi hanno successori diversi

Questa definizione è la base del **processo di induzione**.

Una proprietà è vera in tutto $\mathbb{N}$ se e solo se:

- È vera in 0
- Se è vera in n allora è vera in $s(n)$

È possibile anche iniziare da un numero arbitrario.

2.1.2 Numeri interi

I numeri **interi** (relativi) è l'insieme dei numeri naturali preceduti da un segno "+" o "−".
Questo insieme si denota con il simbolo $\mathbb{Z}$.

$$\mathbb{Z} = \{ \dots, -(n+1), -n, \dots, -2, -1, 0, 1, 2, \dots, n, n+1, \dots \}$$

Ogni intero ha un successore, ma anche un **predecessore** (non c'è un *minimo*).

I numeri interi positivi (più 0) formano $\mathbb{N}$.

$$\mathbb{N} \subset \mathbb{Z}$$

$$\mathbb{N} = \mathbb{Z}^+ \cup \{ 0 \}$$

2.1.2.1 Valore assoluto

Il **valore assoluto** di un numero intero è il numero privo di segno.

$$|-n| = n$$

$$|n| = n$$

L'**opposto** di un numero si ottiene cambiandogli il segno.

2.1.3 Numeri razionali

Razionale in questo caso si riferisce a **ratio** ossia **proporzione**. Indicano dunque una proporzione risultante da una divisione.

Si esprimono come rapporto di due numeri interi (*frazioni*).

$$\frac{m}{n}$$

Si indicano con il simbolo $\mathbb{Q}$.

2.1.3.1 Rappresentazioni e Relazioni

Ogni numero razionale può essere rappresentato da un numero decimale finito o periodico.

$$\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q}$$

2.1.3.2 Densità

I numeri razionali sono **densi**: fra due razionali c'è sempre un altro numero.

Sono comunque **discreti**.

2.1.4 Numeri reali

I **numeri irrazionali** ($\mathbb{I}$) sono quelli che non si possono esprimere tramite frazioni: hanno un'espansione decimale infinita e non periodica.

L'insieme dei **numeri reali** ($\mathbb{R}$) contiene tutti i numeri che ammettono una rappresentazione decimale.

$$\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R}$$

$$\mathbb{R} = \mathbb{Q} \cup \mathbb{I}$$

2.1.4.1 La Retta reale

L'insieme dei numeri reali spesso viene rappresentato su una **retta** (ordine implicito).

A ogni punto della retta è associato un numero reale e viceversa (*corrispondenza biunivoca*).

2.1.5 Numeri complessi

I **numeri complessi** ($\mathbb{C}$) estendono i reali per eseguire operazioni che non sono ben definite altrimenti.

Nascono dalla necessità di estrarre radici a numeri negativi.

Definiscono l'**unità immaginaria** $i = \sqrt{-1}$. Un numero complesso è $a + bi$, con $a, b \in \mathbb{R}$

$$\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \subset \mathbb{C}$$

2.1.6 Numeri booleani

L'insieme dei **numeri booleani** è

$$\mathbb{B} = \{ 0, 1 \}$$

2.2 Insiemi

Gli **insiemi**, le loro proprietà e le loro **operazioni** sono alla base della matematica moderna e dell'informatica.

Un sistema è **discreto** se costituito da elementi isolati e **continuo** se non vi sono spazi vuoti. In matematica, discreto si basa sul concetto di **cardinalità** (il "numero" di elementi che contiene).

Un insieme è discreto se (e solo se) i suoi elementi si possono **numerare**.

> Un insieme è un raggruppamento di oggetti distinti e ben definiti.

Gli oggetti che formano l'insieme sono i suoi **elementi**. In un insieme, tutti gli elementi sono **distinti** e l'ordine non è rilevante.

> Gli elementi di un insieme possono essere anch'essi insiemi.

Un tempo si pensava che la **teoria degli insiemi** poteva dare una base solida alla matematica. Esistono paradossi però che dicono il contrario.

Per esempio il paradosso del barbiere

> In un villaggio vi è un solo barbiere, che rade tutti e soli gli uomini del villaggio che non si radono da soli. *Chi rade il barbiere?*

o il paradosso eterologico

> Una parola è **autologica** se descrive se stessa (*"polisillabica"*, *"corta"*, *"leggibile"*).
> Una parola è **eterologica** se non è autologica (*"polisillabica"*, *"lunga"*, *"illeggibile"*).
> "*Eterologica*" è eterologica?

Il più famoso di essi è il paradosso degli insiemi (*Bertrand Russel*)

> Considerate l'insieme N di tutti gli insiemi che non appartengono a se stessi. N appartiene a se stesso?

Per costruire questo tipo di paradossi è necessario usare un'**autoreferenza** e una **negazione**.

Questa idea torna in diversi contesti per dimostrare l'impossibilità o inesistenza di certe strutture.

2.2.1 Notazione

Gli insiemi generici saranno denotati da lettere latine maiuscole

$$A, B, C, \dots$$

e i loro elementi con lettere latine minuscole

$$a, b, c, \dots$$

L'insieme senza elementi si chiama **vuoto** e si denota con $\varnothing$.

L'**uguaglianza** fra oggetti (elementi, insiemi, entità, ecc.) si denota con "=". La **disuguaglianza** si denota con "$\neq$".

L'uguaglianza ha tre importanti proprietà:

- **Riflessività**: $A = A$
- **Simmetria**: $A = B \iff B = A$
- **Transitività**: se $A = B$ e $B = C$ allora $A = C$

Un insieme può avere diverse rappresentazioni:

- **Diagramma Eulero-Venn**
- **Rappresentazione estensionale**: elenco di tutti gli elementi ($\{ x, y, z \}$)
  - $\{$ rosso, giallo, arancio $\}$: insieme con tre elementi

  - $\{$ rosso, giallo, rosso $\}$: insieme con due elementi
  - $\{ \varnothing \}$: insieme con un elemento
  - $\{ 0, 1, 2, 3, \dots \}$: insieme dei numeri naturali
  - $\{ \varnothing, 1, 2, \{ 3 \} \}$

- **Rappresentazione intensionale**: consiste nel formulare una proprietà $\mathcal{P}$ caratteristica che distingue precisamente gli elementi dell'insieme ($S = \{ x \mid \mathcal{P}(x) \}$)
  - $\{ x \mid x \in \mathbb{Z}, x > 0 \}$: insieme dei numeri interi positivi
  - $\{ x \mid x$ è un colore dell'arcobaleno $\}$
  - $\{ x \mid x \in \mathbb{Z}, x > 3, x \leq 100 \} = \{ 4, 5, \dots, 99, 100 \}$
  - $\{ x \mid x$ è un numero primo $\}$

Per ogni elemento x esiste l'insieme **singoletto** $\{ x \}$.

Proprietà complesse si possono costruire combinando proprietà più semplici mediante operazioni **vero-funzionali**.

Un **sottoinsieme** di A è un insieme formato unicamente per (alcuni) elementi di A. Un sottoinsieme B di A è **proprio** se è diverso da A e da $\varnothing$.

L'insieme vuoto ammette esattamente un sottoinsieme: $\varnothing$ (*sottoinsieme non proprio*). Un singoletto $\{ a \}$ ammette due sottoinsiemi: $\varnothing$ e $\{ a \}$ (*sottoinsiemi non propri*).

Se A e B hanno gli stessi elementi, sono mutuamente sottoinsiemi

$$A = B \text{ se } A \subseteq B, B \subseteq A$$

L'inclusione soddisfa le proprietà:

- **Riflessività**: $A \subseteq A$
- **Antisimmetria**: $A \subseteq B \land B \subseteq A \iff A = B$
- **Transitività**: $A \subseteq B \land B \subseteq C \iff A \subseteq C$

L'insieme potenza (o insieme delle parti) di un insieme S, scritto $\mathscr{P}(S)$ è l'insieme formato da tutti i sottoinsiemi di S.

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

Esempi:

- $\mathscr{P}(\varnothing) = \{ \varnothing \}$
- $\mathscr{P}(\{ \varnothing \}) = \{ \varnothing, \{ \varnothing \} \}$
- $\mathscr{P}(\{ x, y \}) = ?$

Se S ha n elementi ($n \geq 0$) allora $\mathscr{P}(S)$ ha 2^n elementi.

2.2.2 Operazioni

2.2.2.1 Unione

L'**unione** di due insiemi A e B si denota

$$A \cup B$$

ed è definita come

$$A \cup B = \{ x \mid x \in A \vee x \in B \}$$

Le proprietà dell'unione sono:

- **Idempotenza**: $A \cup A = A$
- **Commutatività**: $A \cup B = B \cup A$
- **Associatività**: $A \cup (B \cup C) = (A \cup B) \cup C$
- **Esistenza del neutro**: $A \cup \varnothing = A$
- **Assorbimento**: $A \cup B = B$ se $A \subseteq B$
- **Monotonicità**: $A \subseteq A \cup B$ e $B \subseteq B \cup A$

2.2.2.2 Intersezione

L'**intersezione** di due insiemi A e B si denota

$$A \cap B$$

ed è definita come

$$A \cap B = \{ x \mid x \in A \wedge x \in B \}$$

Le proprietà dell'intersezione sono:

- **Idempotenza**: $A \cap A = A$
- **Commutatività**: $A \cap B = B \cap A$
- **Associatività**: $A \cap (B \cap C) = (A \cap B) \cap C$
- **Annichilazione**: $A \cap \varnothing = \varnothing$
- **Assorbimento**: $A \cap B = B$ se $A \subseteq B$
- **Monotonicità**: $A \cap B \subseteq A$ e $A \cap B \subseteq B$

L'unione e l'intersezione distribuiscono una sull'altra

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

2.2.2.3 Sottrazione

La **sottrazione** tra due insiemi A e B è definita come

$$A \setminus B = \{\, x \mid x \in A \wedge x \notin B \,\}$$

Le proprietà della sottrazione sono:

- $A \setminus A = \varnothing$
- $A \setminus \varnothing = A$
- $\varnothing \setminus A = \varnothing$
- $A \setminus B = A \cap \overline{B}$
- $(A \setminus B) \setminus C = A \setminus (B \cup C) = (A \setminus C) \setminus B$
- $A \setminus B \neq B \setminus A$

2.2.2.4 Differenza simmetrica

La differenza simmetrica tra A e B è

$$A \triangle B = (A \setminus B) \cup (B \setminus A) = (A \cup B) \setminus (A \cap B)$$

Proprietà:

- $A \triangle A = \varnothing$

- $A \triangle \varnothing = A$
- $A \triangle B = B \triangle A$

2.2.2.5 Complementazione

Dato un insieme di riferimento U (chiamato **Universo**), il **complemento** assoluto di A è definito come:

$$\overline{A} = \{ x \mid x \in U, x \notin A \} = U \setminus A$$

Le proprietà della complementazione sono:

- $\overline{U} = \varnothing$
- $\overline{\varnothing} = U$
- $\overline{\overline{A}} = A$
- $A \cap \overline{A} = \varnothing$ (*terzo escluso*)
- $A \cup \overline{A} = U$
- $\overline{(A \cup B)} = \overline{A} \cap \overline{B}$ (*legge di De Morgan*)
- $\overline{(A \cap B)} = \overline{A} \cup \overline{B}$ (*legge di De Morgan*)
- $A \subseteq B \iff \overline{B} \subseteq \overline{A}$

2.2.3 Famiglie di insiemi

Un insieme i cui elementi sono tutti insiemi viene chiamato **famiglia di insiemi** ($\mathcal{F}$).

Le operazioni su una famiglia di insiemi sono:

$$\cup \mathcal{F} = \{ x \mid x \in A \text{ per almeno un insieme } A \in \mathcal{F} \}$$

$$\cap \mathcal{F} = \{ x \mid x \in A \ \forall \ A \in \mathcal{F} \}$$

Dunque

$$\cup \mathcal{P}(A) = A \ \forall \ A$$

2.2.4 Partizioni

Una partizione di un insieme $A \neq \varnothing$ è una famiglia $\mathcal{F}$ di sottoinsiemi di A tale che:

- $\forall\, c \in \mathcal{F}, c \neq \varnothing$ (*non trivialità*)
- $\cup\mathcal{F} = A$ (*copertura*)
- se $c \in \mathcal{F}$, $D \in \mathcal{F}$ e $C \neq D$, allora $C \cap D = \varnothing$ (*disgiunzione*)

2.3 Relazioni

2.3.1 Ordinamenti negli insiemi

Ricordate che gli insiemi **non** sono ordinati

$$\{ x, y \} = \{ y, x \}$$

A volte è utile poter ordinare i loro elementi in modo chiaro.

2.3.1.1 Coppia ordinata

Una **coppia ordinata** è una collezione di due elementi, dove si può distinguere il **primo** e il **secondo** elemento

$$\langle x, y \rangle$$

Il primo elemento è x e il secondo è y. Notare che esiste la coppia ordinata $\langle x, x \rangle$.

2.3.1.1.1 Formulazione Insiemistica

La coppia ordinata $\langle x, y \rangle$ non è altro che l'insieme

$$\{ \{ x \}, \{ x, y \} \}$$

Sia $\mathcal{F} = \{ \{ x \}, \{ x, y \} \}$. x è il **primo elemento** $\iff$ $x \in \cap\mathcal{F}$ (*appartiene a tutti gli insiemi*). y è il **secondo elemento** $\iff$ $y \in \cup\mathcal{F} \setminus \cap\mathcal{F}$ (*non appartiene a tutti gli insiemi*) oppure $\{ y \} = \cup\mathcal{F}$ ($\mathcal{F} = \{\{y\}\}$).

Notare che $\langle x, x\rangle = \{\{x\}, \{x, x\}\}$.

2.3.1.1.2 Definizione giusta Vogliamo vedere che questa definizione **caratterizza** le coppie ordinate. Cioè, che

$$\langle a, b\rangle = \langle x, y\rangle \iff \{\{a\}, \{a, b\}\} = \{\{x\}, \{x, y\}\}$$

Le coppie ordinate sono **ben definite**.

2.3.1.1.3 Generalizzazione Possiamo generalizzare le coppie ordinate a **tuple ordinate** di lunghezza $n \geq 2$ (n-tuple ordinate) definendo

$$\langle x_1, x_2, \dots, x_n, x_{n+1}\rangle = \langle\langle x_1, x_2, \dots, x_n\rangle, x_{n+1}\rangle$$

2.3.1.2 Prodotto cartesiano Dati due insiemi A e B, definiamo il prodotto cartesiano come

$$A \times B = \{\, \langle x, y\rangle \mid x \in A, y \in B \,\}$$

$A \times B$ è l'insieme di tutte le coppie ordinate dove:

- il primo elemento appartiene ad A
- il secondo elemento appartiene a B

Notare che:

- $A \times B \neq B \times A$
- $A \times \varnothing = \varnothing = \varnothing \times A$

$A \times A$ è a volte denotato con A^2.

2.3.1.3 Sequenze S^n è l'insieme di tutte le n-tuple di elementi di S definito tramite prodotti cartesiani di S. Una **sequenza finita** di elementi di S è un elemento di S^n per qualche $n \in \mathbb{N}$.

In altre parole, una sequenza è una tupla ordinata

$$\langle s_1, \dots s_n \rangle$$

dove $n \in \mathbb{N}$ e ogni $s_i \in S$.

2.3.1.4 Segmento

Data una sequenza finita $\sigma = \langle s_1, \dots, s_n \rangle$, una sequenza $\sigma' = \langle s_k, s_{k+1}, \dots, s_\ell \rangle$ dove $1 \leq k \leq \ell \leq n$ è chiamata un **segmento** di σ.

Il segmento è **iniziale** sse $k = 1$.

2.3.2 Relazioni

Una **relazione** tra gli elementi di due insiemi A e B non è altro che un sottoinsieme di $A \times B$.

Una relazione rappresenta un **collegamento** tra gli elementi di A e quelli di B.

2.3.3 Relazioni tra oggetti

Se la coppia ordinata $\langle x, y \rangle$ appartiene a una relazione $R \subseteq A \times B$, si dice che $x \in A$ ha come **corrispondente** $y \in B$ nella relazione R oppure che x è *in relazione con* y.

2.3.4 Rappresentazione tabulare

Ogni relazione si può rappresentare graficamente tramite una tabella.

2.3.5 Rappresentazione matriciale

R si può anche rappresentare tramite una **matrice booleana**.

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

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

2.3.6 Elementi di una relazione

Sia $R \subseteq A \times B$ una relazione

- Il **dominio** di R ($\text{dom}(R)$) è l'insieme di tutti gli oggetti $x \in A$ tali che $\langle x, y \rangle \in R$ per qualche $y \in B$.

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

- Il **codominio** è l'insieme di tutti gli oggetti $y \in B$ tali che $\langle x, y \rangle \in R$ per qualche $x \in A$.

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

- Il **campo** o **estensione** di R è $\text{dom}(R) \cup \text{codom}(R)$.

2.3.7 Relazioni n-arie

Il concetto di relazione può estendersi a tuple ordinate con **più di due** elementi.

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

Esempi:

- $\{\, \langle x, x \rangle \mid x \in A \,\}$ è una relazione binaria su A
- $\{\, \langle x, y \rangle \mid x, y \in \mathbb{N}, x \leq y \,\}$ è la relazione d'ordine naturale su $\mathbb{N}$
- $\{\, \langle x, y, z \rangle \mid x, y, z \in \mathbb{R}, x^2 + y^2 = z^2 \,\}$ è un'area geometrica

2.3.8 Operazioni su relazioni

Siano $R, S \subseteq A \times B$ due relazioni

- $R \cup S$ ha tutte le coppie che appartengono a R o a S
- $R \cap S$ ha tutte le coppie che appartengono ad entrambi R e S
- $\overline{R} = \{ \langle x, y \rangle \mid \langle x, y \rangle \notin R \} \subseteq A \times B$ è il **complemento** di R
- $R^{-1} = \{ \langle y, x \rangle \mid \langle x, y \rangle \in R \} \subseteq A \times B$ è la **relazione inversa** di R

2.3.9 Proprietà delle relazioni

Siano $R, S \subseteq A \times B$ due relazioni

- Se $R \subseteq S$ allora $\overline{S} \subseteq \overline{R}$
- $\overline{(R \cap S)} = \overline{R} \cup \overline{S}$
- $\overline{(R \cup S)} = \overline{R} \cap \overline{S}$
- se $R \subseteq S$ allora $R^{-1} \subseteq S^{-1}$
- $(R \cap S)^{-1} = R^{-1} \cap S^{-1}$
- $(R \cup S)^{-1} = R^{-1} \cup S^{-1}$

2.3.9.1 Esempi Siano $A = \{ a, b \}, R = \{ \langle a, b \rangle, \langle b, a \rangle \}, S = \{ \langle a, b \rangle, \langle a, a \rangle \}$ ($R \subseteq A^2; S \subseteq A^2$).

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

2.3.10 Identità

Dato un insieme A, la relazione

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

dove ogni elemento è in relazione con se stesso è chiamata l'**identità** su A.

2.3.11 Proprietà delle relazioni binarie

Una relazione $R \subseteq A^2$ è

- **Riflessiva** se $\langle x, x \rangle \in R \; \forall \, x \in A \; (I_A \subseteq R)$
- **Simmetrica** se $\langle x, y \rangle \in R \implies \langle y, x \rangle \in R \; (R = R^{-1})$
- **Antisimmetrica** se $\langle x, y \rangle, \langle y, x \rangle \in R \implies x = y \; (R \cap R^{-1} \subseteq I_A)$
- **Antisimmetrica** (*def alternativa*) se $x \neq y \wedge \langle x, y \rangle \in R \implies \langle y, x \rangle \notin R \; (R \cap R^{-1} \subseteq I_A)$
- **Transitiva** se $\langle x, y \rangle, \langle y, z \rangle \in R \implies \langle x, z \rangle \in R$

2.4 Funzioni

Una classe di relazioni binarie di particolare importanza sono le **funzioni** (o **applicazioni**).

Una funzione è una relazione $R \subseteq A \times B$ tale che ad ogni $a \in A$ corrisponde **al più** un elemento $b \in B$.

Formalmente: se $\langle a, b \rangle, \langle a, c \rangle \in R$ allora $b = c$.

Notazione: $f : A \to B$

Se per ogni $a \in A$ esiste **esattamente un** $b \in B$ tale che $\langle a, b \rangle \in R$, allora f è una **funzione totale**.

Riformulazione: una relazione $f \subseteq A \times B$ è una funzione se per ogni $x \in \text{dom}(f)$ esiste un unico $y \in B$ tale che $\langle x, y \rangle \in f$. $f(x)$ denota tale elemento y.

Se $x \in \text{dom}(f)$, allora si dice che f è **definita** in x. Se $A = \text{dom}(f)$ allora f è una funzione **totale**.

2.4.1 Funzione iniettiva

Una funzione f è **iniettiva** se porta elementi distinti del dominio in elementi distinti del codominio (immagine).

> $f : A \to B$ è iniettiva sse per ogni $x, y \in A, x \neq y \implies f(x) \neq f(y)$.

2.4.2 Funzione suriettiva

Una funzione f è **suriettiva** quando ogni elemento di B è immagine di almeno un elemento di A ossia, quando $B = \text{codom}(f)$.

> $f : A \to B$ è suriettiva sse per ogni $y \in B$ esiste un $x \in A$ tale che $f(x) = y$.

2.4.3 Funzione biiettiva

Una funzione $f : A \to B$ è **biettiva** sse è iniettiva e suriettiva.

Attenzione: f può non essere totale.

- Ad ogni $x \in \text{dom}(f)$ corrisponde esattamente un $y \in B$
- Ad ogni $y \in B$ corrisponde esattamente un $x \in \text{dom}(f)$

2.4.4 Corrispondenza biunivoca

Una **corrispondenza biunivoca** tra A e B è una relazione binaria $R \subseteq A \times B$ tale che ad ogni elemento di A corrisponde uno ed un solo elemento di B e viceversa, ad ogni elemento di B corrisponde uno ed un solo elemento di A.

Tale R deve essere una funzione *totale*, *iniettiva* e *suriettiva*.

2.4.5 Formalizzazione

$f \subseteq A \times B$

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

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

Funzione (parziale)

$$\forall a \in A. \forall x, y \in B. (\langle a, x \rangle \in f \land \langle a, y \rangle \in f) \implies x = y$$

Funzione totale

$$\forall a \in A. \exists! x \in B. \langle a, x \rangle \in f$$

Funzione iniettiva

$$\forall a \in A. \forall x, y \in B. (\langle a, x \rangle \in f \land \langle a, y \rangle \in f) \implies x = y \land$$
$$\forall a, b \in A. \forall x \in B. (\langle a, x \rangle \in f \land \langle b, x \rangle \in f) \implies a = b$$

Funzione suriettiva

$$\forall a \in A. \forall x, y \in B. (\langle a, x \rangle \in f \land \langle a, y \rangle \in f) \implies x = y \land$$
$$\forall x \in B. \exists a \in A. \langle a, x \rangle \in f$$

Funzione biiettiva

$$\forall a \in A. \forall x, y \in B. (\langle a, x \rangle \in f \land \langle a, y \rangle \in f) \implies x = y \land$$
$$\forall a, b \in A. \forall x \in B. (\langle a, x \rangle \in f \land \langle b, x \rangle \in f) \implies a = b \land$$
$$\forall x \in B. \exists a \in A. \langle a, x \rangle \in f$$

2.4.6 Punto fisso

Sia A un insieme e $f : A \to A$ una funzione.

Un **punto fisso** di f è un elemento di A che coincide con la sua immagine

$$x = f(x)$$

2.4.7 Operazioni

Sia A un insieme.

Un'**operazione** (n-aria) su A è una funzione $A^n \to A$.

L'operazione è totale sse la funzione è totale.

2.4.8 Immagine inversa

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

$$f^{-1} : B \to \mathcal{P}(A)$$
$$f^{-1}(y) = \{ x \in A \mid f(x) = y \}$$

Nota: f è iniettiva sse per ogni $y \in B$, $f^{-1}(y)$ ha al più un elemento.

2.4.9 Funzione inversa

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 $y \in B$o

$$g(f(x)) = x$$
$$f(g(y)) = y$$

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

Una funzione f è invertibile sse è iniettiva. f_{-1} è totale sse f è suriettiva.

2.4.10 Composizione di Funzioni

La **composizione** di due funzioni si riferisce all'applicazione di una funzione al risultato di un'altra.

Siano $f : A \to B$ e $g : B \to C$ due funzioni. La funzione composta $g \circ f : A \to C$ è definita per ogni $x \in A$ da

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

$(g \circ f)(x)$ è definita sse $f(x)$ e $g(f(x))$ sono definite.

Se $f : A \to B$ e $g : C \to D$ sono due funzioni, allora la composizione $g \circ f$ è solo definibile se codom$(f) \subseteq C$.

Le proprietà della composizione:

- **Associativa:** $f \circ (g \circ h) = (f \circ g) \circ h$
- Se f e g sono entrambe iniettive, allora $f \circ g$ è **iniettiva**
- Se f e g sono entrambe suriettive, allora $f \circ g$ è **suriettiva**
- Se f e g sono entrambe invertibili, allora $f \circ g$ è **invertibile** $((g \circ f)^{-1} = f^{-1} \circ g^{-1})$

2.4.11 Funzione caratteristica

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

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

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

Per ogni $x \in A$

- $f_{S \cap T}(x) = f_S(x) \cdot f_T(x)$
- $f_{S \cup T}(x) = f_S(x) + f_T(x) - f_S(x) \cdot f_T(x)$
- $f_{S \triangle T}(x) = f_S(x) + f_T(x) - 2 \cdot f_S(x) \cdot f_T(x)$

2.4.12 Multinsiemi

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

$$\{\!\{ a, a, b, c, c, c \}\!\} \neq \{\!\{ a, b, c \}\!\}$$

Formalmente un multinsieme è una funzione da un insieme a $\mathbb{N}$

$$f : A \to \mathbb{N}$$

che esprime quante volte si ripete ogni elemento nel multinsieme ($A = \{ a, b, c, d \}$)

$$\{ \langle a, 2 \rangle, \langle b, 1 \rangle, \langle c, 3 \rangle, \langle d, 0 \rangle \}$$

2.5 Cardinalità

I **numeri cardinali** si utilizzano per misurare gli insiemi (indicare la loro *grandezza*). Se un insieme è **finito**, la sua cardinalità è un numero naturale (il numero di elementi). Con i numeri cardinali, possiamo anche misurare e classificare insiemi **infiniti**.

2.5.1 Cardinalità tramite funzioni

Georg Cantor utilizzò le proprietà delle funzioni per paragonare la cardinalità degli insiemi.

Sia f una funzione $f : A \to B$

- Se f è *suriettiva* allora B non è "più grande" di A
- Se f è *totale* e *iniettiva* allora A non è "più grande" di B

Due insiemi sono **equipotenti** (hanno la stessa cardinalità) sse esiste una funzione **biunivoca** fra di loro.

$$A \sim B$$

2.5.2 Cardinalità finite

Se A ha n elementi, allora $A \sim \{ 1, \dots, n \}$. In questo caso si dice che A è **finito** e ha **cardinalità** (o potenza) n.

Utilizziamo la notazione

$$|A| = n$$

I numeri naturali si utilizzano come cardinali finiti.

Se$|A| = n$ allora $|\mathscr{P}(A)| = 2^n$.

2.5.3 Numerabili

Basati su questa definizione, chiamiamo **numerabili** tutti gli insiemi che hanno la cardinalità di $\mathbb{N}$. I suoi elementi possono essere posti in corrispondenza biunivoca con i naturali.

$$A \sim \mathbb{N} \sim \mathbb{N}^+$$

La cardinalità di $\mathbb{N}$ è chiamata $\aleph_0$.

$$|\mathbb{N}| = \aleph_0$$

$\aleph_0$ è il più piccolo dei numeri cardinali **transfiniti** (i cardinali per misurare insiemi infiniti). Ovviamente $\aleph_0$ non è un numero naturale.

I seguenti insiemi sono numerabili:

- L'insieme dei numeri pari
- L'insieme dei numeri primi
- L'insieme dei numeri interi $\mathbb{Z}$

$$f : \mathbb{N} \to \mathbb{Z}$$

$$f(x) = \begin{cases} -\frac{x}{2} & \text{se } x \text{ pari} \\ \left\lceil \frac{x}{2} \right\rceil & \text{se } x \text{ dispari} \end{cases}$$

- Il prodotto cartesiano $\mathbb{N} \times \mathbb{N}$

- I numeri razionali $\mathbb{Q}$ ($\subset \mathbb{N} \times \mathbb{N}$)

2.5.4 Il continuo

$$[0, 1] = \{ x \in \mathbb{R} \mid 0 \leq x \leq 1 \} \sim \mathscr{P}(\mathbb{N})$$

Denotiamo per convenzione $|\mathscr{P}(\mathbb{N})| = 2^{\aleph_0}$. Allora $|\mathbb{R}| \geq 2^{\aleph_0}$.

Cantor dimostro che $\aleph_0 < 2^{\aleph_0}$ (in realtà che $|A| < |\mathscr{P}(A)|$). Dunque $\mathbb{R}$ non è numerabile.

2.5.4.1 Teorema di Cantor

$$\aleph_0 < 2^{\aleph_0}$$

Dobbiamo dimostrare che *non esiste* una funzione biunivoca $f : \mathbb{N} \to \mathscr{P}(\mathbb{N})$.

Supponiamo che esiste una tale funzione f. Definiamo

$$Z = \{ z \in \mathbb{N} \mid n \notin f(n) \} \subseteq \mathbb{N}$$

Siccome f è biunivoca (quindi suriettiva), esiste $k \in \mathbb{N}$ tale che $f(k) = Z$.

Domanda: $k \in Z$?

Se $k \in Z$, allora per definizione $k \notin f(k) = Z$. Se $k \notin Z$, allora $k \notin f(x)$ e quindi per definizione $k \in Z$.

Conclusione: la funzione f non può esistere.

2.5.5 Gerarchia transfinita

Cantor definì la gerarchia dei numeri transfiniti

$$\aleph_0 < \aleph_1 < \aleph_2 < \dots$$

L'**ipotesi del continuo** dice che $\aleph_1 = 2^{\aleph_0}$. Non ci sono insiemi di cardinalità intermedia fra $\mathbb{N}$ e $\mathbb{R}$.

3 Strutture relazionali, Grafi e Ordinamenti

3.1 Rappresentazioni

Le relazioni possono essere rappresentate da diverse forme:

- **Rappresentazione per elencazione:** descrivere l'insieme di coppie ordinate ($R = \{ \langle 1, 2\rangle, \langle 2, 4\rangle, \langle 3, 6\rangle \}$)
- **Rappresentazione sagittale:** collegare con delle frecce gli elementi che verificano la relazione
- **Rappresentazione tramite diagramma cartesiano:** se S e T sono sottoinsiemi di $\mathbb{R}$, rappresentare le coppie come coordinate sul piano cartesiano
- **Rappresentazione tramite tabella:** una matrice booleana con per colonne gli elementi dell'insieme di arrivo e per righe l'insieme di partenza.

3.1.1 Relazioni in un insieme

Una relazione $R \subseteq S \times S$ è detta **relazione in S**. In una relazione in S, la rappresentazione sagittale collassa in un **grafo**. Usiamo lo stesso insieme per l'origine e la destinazione di ogni freccia. Formalmente un grafo è costituito da **nodi** collegati fra loro da frecce (o **spigoli**). Se $\langle x, y\rangle \in \mathbb{R}$, disegnamo uno spigolo da x a y.

Le proprietà di una relazione sono (*again*):

- **Riflessiva** se: $\langle x, x\rangle \in R \; \forall \, x \in S$ (*ogni nodo ha un cappio*)
- **Irriflessiva** se: $\langle x, x\rangle \notin R \; \forall \, x \in S$ (*nessun nodo ha un cappio*)
- **Simmetrica** se: $\langle x, y\rangle \in R \implies \langle y, x\rangle \in R$ (*ogni spigolo ha il suo inverso*)
- **Asimmetrica** se: $\langle x, y\rangle \in R \implies \langle y, x\rangle \notin R$ (*nessuno spigolo ha il suo inverso e nessun nodo ha un cappio*)
- **Antisimmetrica** se: $\langle x, y\rangle \in R \land \langle y, x\rangle \in R \implies x = y$ (*nessuno spigolo ha il suo inverso (escluso il cappio)*)
- **Transitiva** se: $\langle x, y\rangle \in R \land \langle y, z\rangle \in R \implies \langle x, z\rangle \in R$

Una relazione $R \subseteq S \times S$ in S è

- **Connessa** se ogni due elementi sono collegati. $\forall\, x, y \in S$x se $x \neq y$ allora $\langle x, y\rangle \in R$ oppure $\langle y, x\rangle \in R$
- **Relazione di equivalenza** se è riflessiva, transitiva e simmetrica

La relazione vuota $\varnothing \subseteq S \times S$ è irriflessiva, simmetrica, asimmetrica, antisimmetrica e transitiva. L'identità I_S è riflessiva, simmetrica e transitiva (è una relazione di equivalenza).

3.1.2 Riflessività ed operazioni

Siano R ed R' due relazioni su S

1. Se R è riflessiva, R^{-1} è riflessiva (*stesso per irriflessibilità*)
2. R è riflessiva sse $\overline{R}$ è irriflessiva
3. Se R ed R' sono riflessive, allora anche $R \cup R'$ e $R \cap R'$ sono riflessive (*stesso per irriflessibilità*)

3.1.3 Simmetria ed operazioni

Siano R ed R' due relazioni su S

1. R è simmetrica sse $R = R^{-1}$
2. Se R è simmetrica, allora R^{-1} e $\overline{R}$ sono simmetriche
3. R è antisimmetrica sse $R \cap R^{-1} \subseteq I_S$
4. R è asimmetrica sse $R \cap R^{-1} = \varnothing$
5. Se R ed R' sono simmetriche, allora anche $R \cup R'$ e $R \cap R'$ sono simmetriche

3.1.4 Transitività ed operazioni

Se R ed R' sono transitive allora $R \cap R'$ è transitiva. $R \cup R'$ non è necessariamente transitiva.

3.1.5 Matrici booleane

Una matrice booleana è una matrice a valori $\{ 0, 1 \}$. La matrice booleana associata a $R \subseteq S \times T$ si denota M_R. Se $|S| = n$ e $|T| = m$, M_R ha n righe e m colonne.

La riga i corrisponde all'elemento $s_i \in S$, la colonna j corrisponde all'elemento $t_j \in T$ ed è tale che

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

3.1.5.1 Proprietà di una matrice booleana

Se R è una relazione su S, M_R ha le stesse proprietà della visualizzazione tabulare.

- R è **riflessiva** sse M_R ha tutti 1 sulla diagonale principale
- R è **irriflessiva** sse M_R ha tutti 0 sulla diagonale principale
- R è **simmetrica** sse M_R è simmetrica
- R è **asimmetrica** sse per ogni i, j, se $m_{ij} = 1$, allora $m_{ji} = 0$
- R è **antisimmetrica** sse per ogni $i \neq j$, se $m_{ij} = 1$, allora $m_{ji} = 0$
- $M_{R^{-1}}$ è la trasposta di M_R
- $M_{\overline{R}}$ si ottiene scambiando 0 e 1 in M_R

$$R = \{ \langle 0, 0 \rangle, \langle 1, 2 \rangle \}$$

$$M_R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \longrightarrow M_{R^{-1}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

3.1.6 Operazioni su matrici booleane

Se M e N sono due matrici booleane di dimensioni $n \times m$, $M \sqcup N$ (il **join** di M e N) è la matrice booleana L didimensionee $n \times m$ i cui elementi sono

$$\ell_{ij} = \begin{cases} 1 & m_{ij} = 1 \vee n_{ij} = 1 \\ 0 & \text{altrimenti} \end{cases}$$

$M \sqcap N$ (il **meet** di M e N) è la matrice booleana L di dimensione $n \times m$ i cui elementi sono

$$\ell_{ij} = \begin{cases} 1 & m_{ij} = 1 \wedge n_{ij} = 1 \\ 0 & \text{altrimenti} \end{cases}$$

$\sqcup$ e $\sqcap$ sono commutative, associative e distributive fra di loro.

3.1.7 Prodotto booleano

Siano M e N matrici booleane di dimensioni $n \times m$ e $m \times p$ rispettivamente. Il loro **prodotto booleano** è la matrice $L = M \odot N$ di dimensioni $n \times p$ dove

$$\ell_{ij} = \begin{cases} 1 & \exists\, k, 1 \leq k \leq m \text{ t.c. } m_{ik} = 1 \wedge n_{kj} = 1 \\ 0 & \text{altrimenti} \end{cases}$$

Esempio:

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \odot \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Questa operazione è associativa ma non commutativa.

YT Link con spiegazione[1].

3.2 Composizione di relazioni

Dati $R_1 \subseteq S \times T$, $R_2 \subseteq T \times Q$:

$$R_2 \circ R_1 = \{ \langle x, y \rangle \in S \times Q \mid \exists \in T . \langle x, z \rangle \in R_1, \langle z, y \rangle \in R_2 \}$$

$R_2 \circ R_1$ è la **composizione** di R_1 e R_2.

La composizione si può calcolare tramite il prodotto di matrici booleane.

$$M_{R_2 \circ R_1} = M_{R_1} \odot M_{R_2}$$

3.3 Relazioni di Equivalenza

Una **relazione di equivalenza** ci aiuta a creare blocchi di elementi che hanno *qualcosa* in comune. Sono relazioni che si comportano "come l'uguaglianza" tra oggetti. Dal punto di vista di una proprietà data, **non** esistono differenze tra due elementi in una relazione di equivalenza.

Def: una relazione riflessiva, simmetrica e transitiva è detta **relazione di equivalenza**.

Esempio:

- Appartenere alla stessa classe
- Essere nati nello stesso anno
- Essere parallele nell'insieme delle rette
- …

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

$$R := \{ \langle x, y \rangle \in A \times A \mid f(x) = f(y) \}$$

[1] https://youtu.be/BjTeDlpj-ts?si=snvhzdZvQByBGinl

è una relazione di equivalenza.

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

3.3.1 Partizioni e classi di equivalenza

Dividendo S in gruppi i cui elementi sono "uguali", possiamo studiare insiemi grandi osservando soltanto pochi elementi. Questi gruppi sono chiamati **classi di equivalenza**.

Sia S un insieme. Una partizione di S è una famiglia di insiemi $\mathcal{P} = \{ T_1, \dots, T_n \}, T_i \subseteq S, 1 \leq i \leq n$ tali che:

- $T_i \neq \varnothing$ per ogni $i, 1 \leq i \leq n$
- $T_i \cap T_j \neq \varnothing$ per ogni $i, j, 1 \leq i \leq j \leq n$
- $\cup \mathcal{P} = S$

Se R è una **relazione di equivalenza** su S allora $T \neq \varnothing \subseteq S$ è una classe di equivalenza se per ogni $x \in S$:

$$x \in T \iff \{ y \in S \mid \langle x, y \rangle \in R \} = T$$

Cioè, x è in relazione con tutti e soltanto quegli elementi di T.

Sia S un insieme e R una relazione di equivalenza su S. Ogni elemento $x \in S$ definisce una classe di equivalenza

$$[x]_R = \{ y \in S \mid \langle x, y \rangle \in R \}$$

La famiglia di insiemi $\{ [x]_R \mid x \in S \}$ (gli elementi sono le classi di equivalenza di S) è chiamato l'**insieme quoziente** di S rispetto a R (indicato con S/R). L'insieme quoziente è una partizione di S.

Esempio: Sia $n \in \mathbb{N}$. La relazione $\simeq_n \subseteq \mathbb{N} \times \mathbb{N}$ definita come

$$x \simeq_n y \iff x \equiv y \mod n \leftrightarrow (\text{ossia } (x \mod n) = (y \mod n))$$

è una relazione di equivalenza.

Per $n = 4$, $\simeq_4$ definisce 4 classi di equivalenza.

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

$$[0] = \{\, 0, 4, 8, 12, \dots \,\}$$

$$[1] = \{\, 1, 5, 9, 13, \dots \,\}$$

$$[2] = \{\, 2, 6, 10, 14, \dots \,\}$$

$$[3] = \{\, 3, 7, 11, 15, \dots \,\}$$

L'insieme quoziente $\mathbb{N}/\simeq_4 = \{\, [0], [1], [2], [3] \,\}$ è spesso indicato con $\mathbb{N}_4$.

3.4 Grafi

Un grafo è definito da

- Un insieme di **nodi** (chiamati anche *vertici*)
- Collegamenti tra vertici che possono essere:
  - Orientati (**archi**)
  - Non orientati (**spigoli**)
- (eventualmente) Dati associati ai nodi e collegamenti (**etichette**)

I grafi possono rappresentare **relazioni binarie**.

3.4.1 Gradi

Un arco che va da v a w è **uscente** da v ed entrante in w. Il numero di archi uscenti dal nodo v è il **grado di uscita** di v. Il numero di archi entranti in v è il **grado in ingresso** di v.

Un nodo è chiamato:

- **Sorgente** se non ha archi entranti (*grado di entrata* 0)

- **Pozzo** se non ha archi uscenti (*grado di uscita* 0)
- **Isolato** se non ha archi né uscenti né entranti

I nodi v e w sono **adiacenti** se c'è un arco tra v e w (in qualunque direzione). Questo arco è **incidente** su v e w. Il grado di v è il numero di nodi adiacenti a v.

3.4.2 Cammino

Un **cammino** è una sequenza **finita** di nodi

$$\langle v_1, v_2, \dots, v_n \rangle$$

tali che per ogni i, $1 \leq i < n$, esiste un arco uscente da v_i ed entrante in v_{i+1}. Questo cammino va da v a w se $v_1 = v$ e $v_n = w$.

3.4.3 Semicammino

Un **semicammino** è una sequenza finita di nodi

$$\langle v_1, v_2, \dots, v_n \rangle$$

tali che per ogni i, $1 \leq i < n$, 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 (anche se $v_1 = v_n$).

Un grafo è **connesso** se esiste sempre un semicammino tra due nodi qualsiasi.

3.4.4 Ciclo

Un **ciclo** intorno al nodo v è un cammino tra v e v. Un **semiciclo** intorno al nodo v è un semicammino tra v e v. Un **cappio** intorno a v è un ciclo di lunghezza 1.

3.4.5 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 c'è nessun cammino da v a w allora la distanza è infinita (∞)

In un grafo ordinato, la distanza da v a w **non** è sempre uguale alla distanza da w a v.

3.4.6 Trovare le distanze: Algoritmo

Ricerca in **ampiezza** delle distanze da v ad ogni nodo.

Inizializzazione:

- Segnare v come **visitato** con distanza $d(v) = 0$
- Segnare altri nodi come **non visitato**

Ciclo:

- Trovare un nodo w **visitato** con distanza *minima* $d(w) = n$
- Segnare w come **esplorato**
- 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 $d(w) = \infty$.

3.4.7 Definizione formale di grafo

Un **grafo orientato** è una coppia $G = (V, E)$ dove

- V è un insieme di **nodi**
- $E \subseteq V \times V$ è una relazione binaria in V (**archi**)

Un **grafo non orientato** è un grafo orientato dove E è una relazione **simmetrica**. In questo caso gli archi sono rappresentati come **coppie non ordinate** (v, w) ($(v, w) = (w, v)$). Graficamente togliamo le frecce (l'ordine) agli archi.

3.4.8 Sottografo

Il grafo $G_1 = (V_1, E_1)$ è un **sottografo** di $G_2 = (V_2, E_2)$ sse $V_1 \subseteq V_2$ e $E_1 \subseteq E_2$. Un sottografo si ottiene togliendo nodi e/o archi dal grafo.

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

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

3.4.9 Grafo aciclico orientato (DAG)

Un grafo orientato senza cicli si chiama **grafo aciclico orientato**.

In un DAG non esiste nessun cammino da un nodo a se stesso

3.4.10 Grafi etichettati

Un **grafo etichettato** è una tripla $G = (V, E, \ell)$ dove

- (V, E) è un grafo
- $\ell : E \to L$ è una funzione totale che associa ad ogni arco $e \in E$ un'**etichetta** da un insieme L

Diamo un'etichetta ad ogni arco del grafo.

Un grafo etichettato può rappresentare una **relazione ternaria** (e viceversa).

I nomi e le etichette sono spesso irrilevanti.

3.4.11 Matrice di adiacenza

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

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

3.4.12 Grafo completo

Un **grafo completo** collega ogni nodo con tutti gli altri nodi (ma non con se stesso).

La sua matrice di adiacenza ha 0 su tutta la diagonale ed 1 sulle altre posizioni.

3.4.13 Connettività

Ricordiamo che $G = (V, E)$ è **connesso** se per ogni $v, w \in V$ esiste un **semicammino** da v a w. G è **fortemente connesso** se per ogni due nodi $v, w \in V$ esiste un **cammino** da v a w.

In un grafo fortemente connesso:

- Esiste sempre un ciclo che visita **ogni** nodo (non necessariamente semplice)
- Non ci sono né sorgenti né pozzi

3.4.14 Isomorfismi tra grafi

Due grafi $G_1 = (V_1, E_1)$ e $G_2 = (V_2, E_2)$ sono **isomorfi** se esiste una funzione biunivoca $f : V_1 \to V_2$ tale che

$$\langle v, w \rangle \in E_1 \iff \langle f(v), f(w) \rangle \in E_2$$

L'isomorfismo f mantiene la struttura del grafo G_1, ma sostituisce i nomi dei vertici con quelli di G_2. Due grafi isomorfi sono in realtà lo **stesso grafo** con i nodi rinominati.

3.4.15 Chiusure

3.4.15.1 Chiusura riflessiva

La **chiusura riflessiva** di $R \subseteq S^2$ è la più piccola relazione riflessiva R^{refl} su S che contiene R.

$$R \subseteq R^{\text{refl}} = R \cup I_S$$

3.4.15.2 Chiusura transitiva

La **chiusura transitiva** di $R \subseteq S^2$ è la più piccola relazione transitiva R^{trans} su S che contiene R.

$$R \subseteq R^{\text{trans}} \subseteq S$$

3.4.15.3 Chiusura simmetrica

La **chiusura simmetrica** di $R \subseteq S^2$ è la più piccola relazione transitiva R^{simm} su S che contiene R.

$$R \subseteq R^{\text{trans}} = R \cup R^{-1}$$

3.5 Alberi

Un'**albero** è un DAG *connesso* tale che

- Esiste *esattamente un* nodo sorgente (*radice dell'albero*)
- Ogni nodo diverso dalla radice ha **un solo** arco entrante

I nodi pozzo di un albero sono chiamati **foglie** o **nodi esterni**. Tutti gli altri nodi sono chiamati **interni**. Per analogia con gli **alberi genealogici**, le relazioni tra i nodi usano nomi come *padre*, *figlio*, *discendente*, …

3.5.1 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 dalla radice a v.

Un albero non può essere mai vuoto (la radice esiste sempre).

Se un albero è finito, allora esiste *almeno* una foglia (che può essere anche la radice).

I nodi **intermedi** sono contemporaneamente padre e figlio.

3.5.2 Rappresentazione gerarchica

Gli alberi spesso rappresentano **strutture gerarchiche**. In questo caso, l'ordine è **implicito** (gli archi si disegnano **senza frecce**).

3.5.3 Cammini in un albero

In un albero c'è *esattamente un* **cammino** dalla radice a qualunque nodo v diverso dalla radice. Ogni nodo w in questo cammino è un **ascendente** di v (oppure *avo*) e v è un **discendente** di w (la radice è l'unico nodo senza discendenti). Se il cammino da w a v ha lunghezza 1, allora w è il *padre* di v e v è un figlio di w.

3.5.4 Profondità

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

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

3.5.5 Alberi binari

Un **albero binario** è un albero dove ogni nodo ha al massimo due figli. I figli di un nodo in un albero binario sono **ordinati** (*figlio sinistro* e *figlio destro*).

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.

Un albero binario è una **struttura ricorsiva** composta da

- Un nodo (*radice*)
- Un albero binario sinistro (*eventualmente vuoto*)
- Un albero binario destro (*eventualmente vuoto*)

Possiamo rappresentare un albero binario sia

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

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

Un albero binario è **completo** se

- Ha altezza n
- Ad ogni profondità $i, 0 \leq i < n$ ci sono 2^i nodi
- L'ultimo livello è riempito da sinistra a destra

In rappresentazione tabulare, i nodi vuoti sono soltando sulle ultime righe.

Un albero binario è **bilanciato** se per ogni nodo v la differenza fra

- Il numero di nodi nell'albero sinistro di v
- Il numero di nodi nell'albero destro di v

è *al massimo 1*.

3.5.5.1 Albero binario di ricerca

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

- $z \in \mathbb{Z}$
- Ogni nodo dell'albero sinistro di z è minore di z
- Ogni nodo dell'albero destro di z è maggiore di z

Essi sono utili per rappresentare liste ordinate dinamiche.

3.5.5.2 Attraversamento di un albero binario

Un **attraversamento** è un processo che visita tutti i nodi di un albero. Solitamente in un ordine particolare. Un attraversamento che elenca ogni nodo *esattamente una volta* è un'**enumerazione** (dei nodi).

Distinguiamo fra due tipi di attraversamento:

- In **profondità** esplora ogni ramo dell'albero fino in fondo (*figli prima dei fratelli*)
- In **ampiezza** esplora prima i nodi più vicini alla radice (*fratelli prima dei figli*)

Ci sono tre tipi diversi di ordini in profondità, basati su *quando* enumeriamo un elemento. Si usa la notazione:

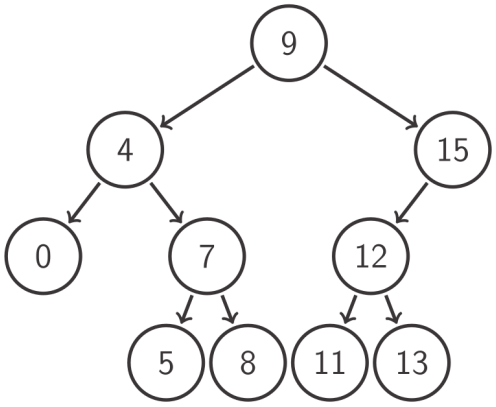
- **L** per sinistra

- **R** per destra
- **V** per enumerazione (*visit*)

I tre ordini di enumerazione in profondità sono:

- In **preordine**: si visita un nodo prima di visitare i figli (*VLR*)
- In **ordine**: si visita l'albero sinistro, poi il nodo, poi l'albero destro (*LVR*)
- In **postordine**: si visitano prima i figli e poi il nodo (*LRV*)

Viene implementata come una **pila** che contiene gli elementi da esplorare (*LIFO*).

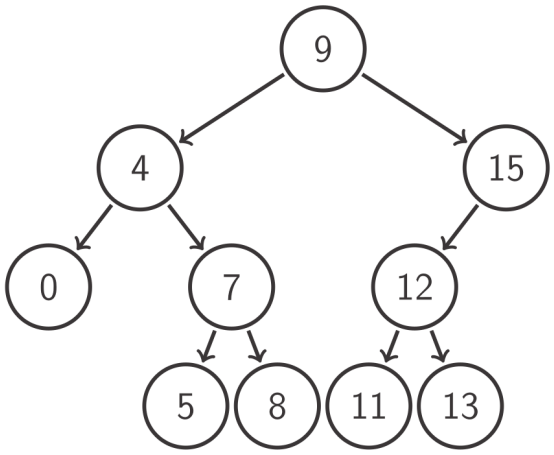


VLR
9,4,0,7,5,8,15,12,11,13

LVR
0,4,5,7,8,9,11,12,13,15

LRV
0,5,8,7,4,11,13,12,15,9

L'enumerazione in ampiezza visita *tutti* i nodi ad una profondità prima di esplorare altri livelli dell'albero. Viene implementata tramite una **coda** di elementi da esplorare (*FIFO*).



9,4,15,0,7,12,5,8,11,13

3.5.5.3 Numero di foglie in un albero

Un albero finito ha sempre al meno una foglia. Per *massimizzare* il numero di foglie dobbiamo avere un albero **pieno**. Un albero pieno

con n nodi interni ha $n+1$ **foglie** (*dimostrazione per induzione*). Il numero di **puntatori nulli** in un albero binario con n nodi è $n+1$. Basta sostituire i puntatori vuoti per foglie speciali, formando un albero pieno.

> Per la dimostrazione per induzione consultare le slide #view-slide

3.6 Ordinamenti

Molto spesso, gli elementi in un insieme hanno una struttura d'**ordine**. Per esempio, $\mathbb{N}$ e $\mathbb{R}$. Anche se ha volte non è possibile paragonare tutti gli oggetti (è più grande $\langle 0, 1\rangle$ o $\langle 1, 0\rangle$?).

Un'**ordinamento** è un tipo particolare di **relazione** fra elementi.

Una relazione R su un insieme S è un:

- **Preordine** sse R è riflessiva e transitiva
- **Ordine parziale** sse R è un preordine antisimmetrico (*riflessiva, antisimm. e transitiva*)
- **Ordine stretto** sse R è irriflessiva e transitiva (e quindi anche asimmetrica)

Un ordine parziale si rappresenta con $\leq$; uno stretto con $<$. La coppia $(S, \leq)$ si chiama **insieme parzialmente ordinato** (*poset*).

Un **ordine totale** è un ordine parziale *fortemente connesso*:

$$\forall x, y \in S \qquad x \leq y \ \vee \ y \leq x$$

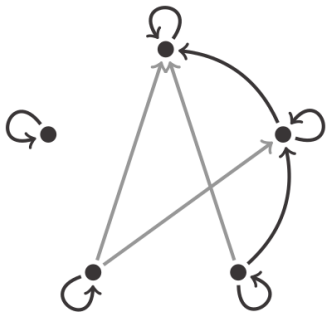
Un **ordine totale stretto** è un ordine stretto *connesso*:

$$\forall x, y \in S \text{ t.c. } x \neq y \qquad x < y \ \vee \ y < x$$

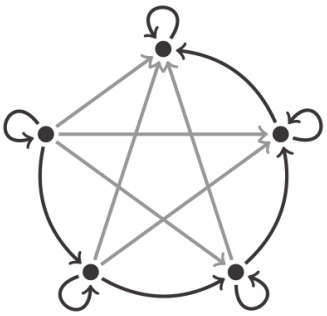
Ordini totali stretti e non stretti sono molto vicini:

- Se Ro 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

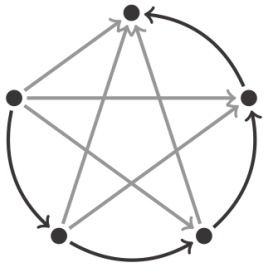
Uno è riflessivo, l'altro irriflessivo.



ordine parziale



ordine totale



ordine totale stretto

3.6.1 Tricotomia

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. $\langle y, x \rangle \in R$

3.6.2 Prodotto di ordinamenti

Siano $(S, \leq_S)$ e $(T, \leq_T)$ due *poset*. Definiamo la relazione $\leq_{S \times T}$ su $S \times T$ come

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

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

3.6.3 Ordinamento lessicografico

L'ordine **lessicografico** paragona tuple di elementi posizione per posizione. La relazione $\leq_{\text{lex}}$ su $S \times T$ è definita da

$$\langle s, t \rangle \leq_{\text{lex}} \langle s', t' \rangle \iff \text{(i) } s <_S s' \text{ oppure (ii) } s = s', t \leq_T t'$$

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

Generalizza l'ordine alfabetico usuale e si può estendere a tuple di lunghezza arbitraria.

3.6.4 Copertura

In un poset $(S, \leq)$, una **copertura** di $x \in S$ è un elemento **minimo più grande** di x.

$y \in S$ è una copertura di $x \in S$ sse

- $x \leq y, x \neq y$
- $\nexists z, x \neq z \neq y$ tale che $x \leq z, z \leq y$

3.6.5 Elementi estremali

In un poset $(S, \leq)$, un elemento $s \in S$ è

- **Minimale** se non esiste un elemento $s' \neq s$ tale che $s' \leq s$
- **Massimale** se non esiste un elemento $s' \neq s$ tale che $s \leq s'$

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

3.6.6 Minoranti e maggioranti

Dato un poset $(S, \leq)$ e un insieme $X \subseteq S$, un elemento $s \in S$ è

- **Minorante** di X sse $s \leq x$ per ogni $x \in X$
- **Massimo minorante** di X ($\sqcap X$) sse $s' \leq s$ per ogni minorante s' di X e se s è un minorante

- **Maggiorante** di X sse $x \leq s$ per ogni $x \in X$
- **Minimo maggiorante** di X ($\sqcup X$) sse $s \leq s'$ per ogni maggiorante s' di X e se s è un maggiorante
- **Minimo** di X sse $s = \sqcap X \in X$
- **Massimo** di X sse $s = \sqcup X \in X$

3.6.7 Proprietà

Ogni $X \subseteq S$ ha **al più** un massimo minorante e un minimo maggiorante.

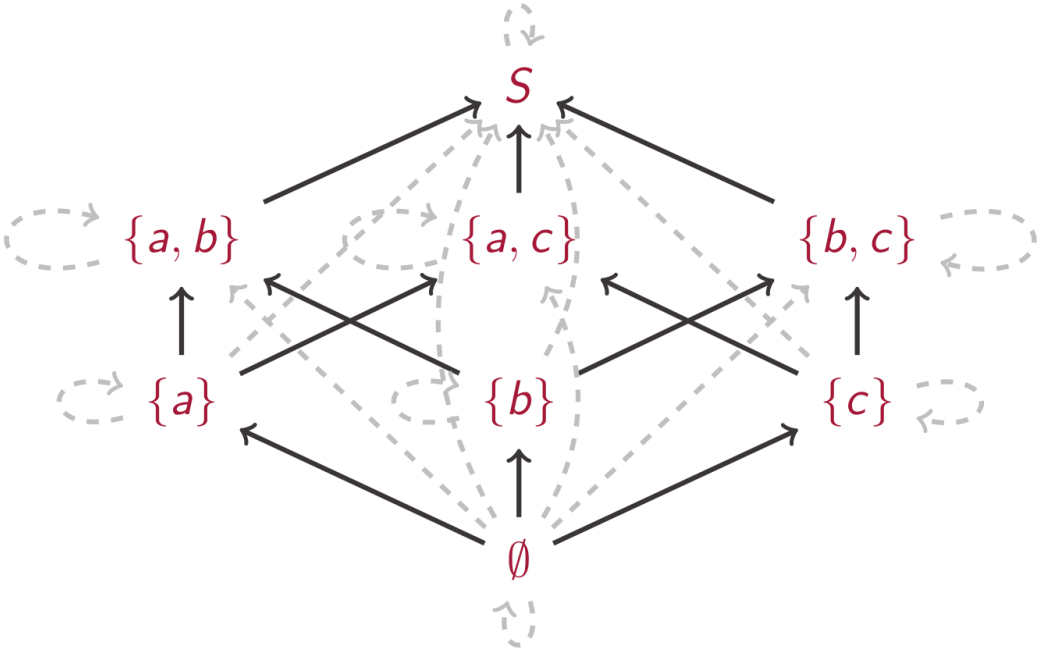
Se ogni $X \subseteq S$ ha un minimo, allora $(S, \leq)$ è un insieme **ben ordinato** (o ben fondato).

Se esiste, $\sqcap S$ è il minimo di S, denotato da $\underline{0}$. Se esiste, $\sqcup S$ è il massimo di S, denotato da $\underline{1}$.

3.6.8 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**.

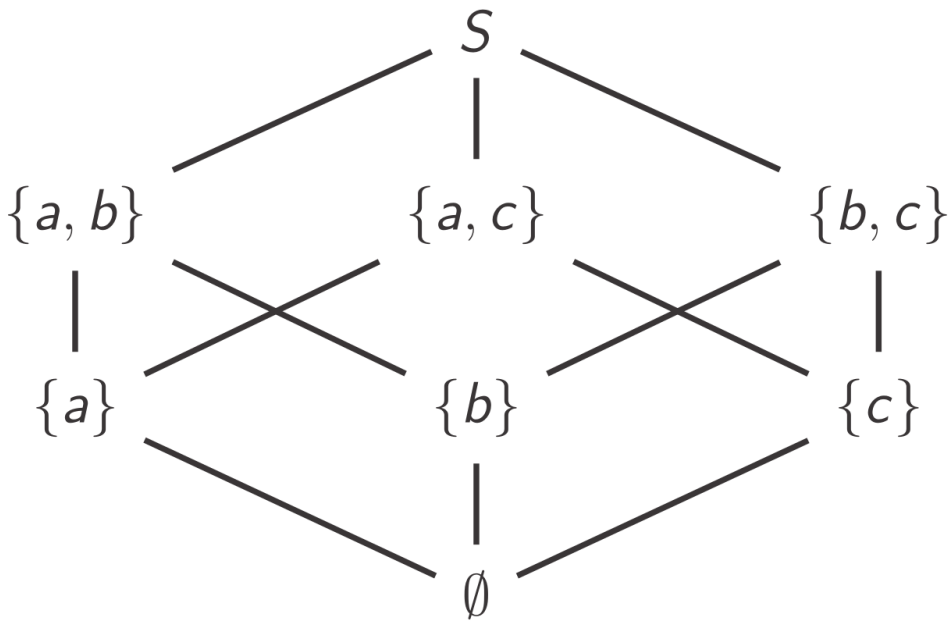
Sia $S = \{ a, b, c \}$. Consideriamo il poset $(\mathscr{P}(S), \subseteq)$



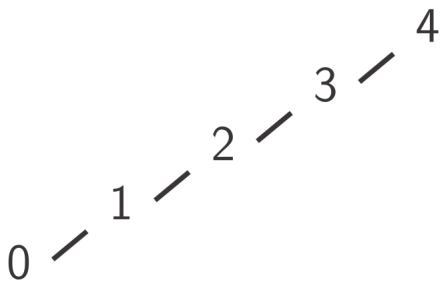
Un diagramma di Hasse è un grafo non orientato tale che per ogni x, y:

- Se $x \leq y$ allora x appare **sotto** y
- x e y sono collegati sse y è una **copertura** di x

L'ordine è la chiusura riflessiva e transitiva del grafo ordientato da giù verso su.



Il diagramma di Hasse di un ordinamento **totale** formerà sempre una **catena**. Per esempio, $(\{ 0, 1, 2, 3, 4 \}, \leq)$:

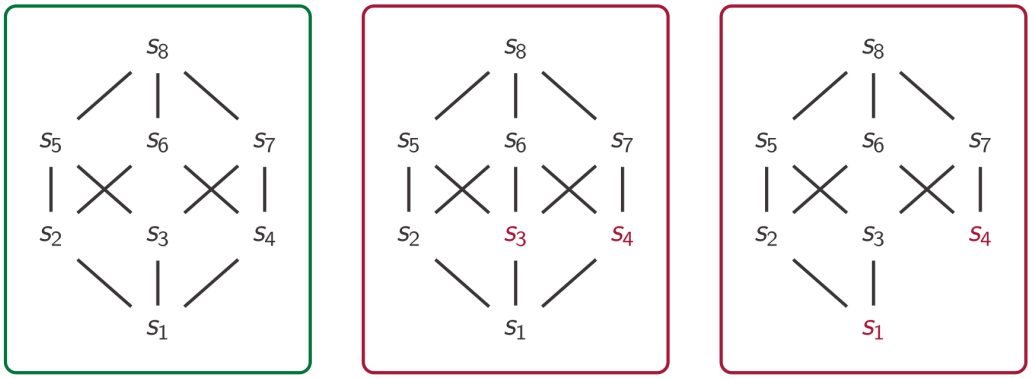


3.7 Reticoli

Un **reticolo** è un poset $(S, \leq)$ tale che per ogni $x, y \in S$:

- Esiste un **minimo maggiorante** $x \sqcup y$ (*join*)

- Esiste un **massimo minorante** $x \sqcap y$ (*meet*)



reticolo | no reticolo | no reticolo

3.7.1 Proprietà

- $a \sqcup a = a = a \sqcap a$ (*idempotenza*)
- $a \sqcup b = b \sqcup a$ (*commutatività*)
- $a \sqcup (b \sqcup c) = (a \sqcup b) \sqcup c$ (*associatività*)
- $a \sqcap (b \sqcap c) = (a \sqcap b) \sqcap c$ (*associatività*)
- $a \sqcup (a \sqcap b) = a = a \sqcap (a \sqcup b)$ (*assorbimento*)

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 $c \leq a \sqcap b$
- $a \sqcup b = b$ sse $a \leq b$
- $a \sqcap b = a$ sse $a \leq b$

3.7.2 Monotonicità

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

- $a \sqcup b \leq c \sqcup d$
- $a \sqcap b \leq c \sqcap d$

3.7.3 Tipi di reticoli

Un reticolo $(L, \leq)$ è

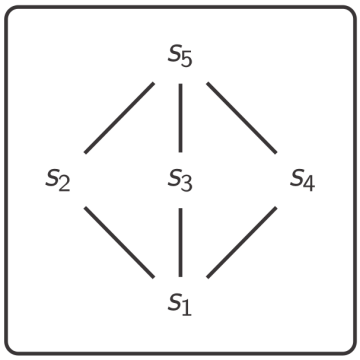
- **Completo** sse per ogni $M \subseteq L$, $\sqcup M$ e $\sqcap M$ esistono
- **Limitato** sse $\underline{1} = \sqcup L$ e $\underline{0} = \sqcap L$ esistono
- **Distributivo** sse meet e join distribuiscono fra di loro:

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

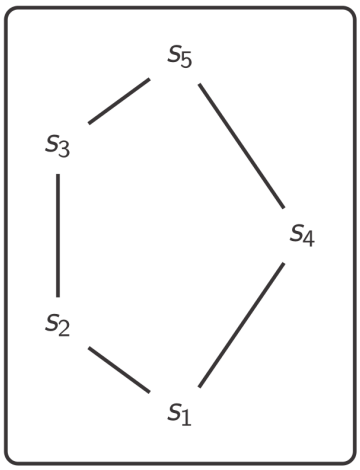
$$a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$$

Ogni reticolo completo è limitato. Ogni reticolo finito è completo e limitato.

I due reticoli *non* distributivi prototipici sono



M_3



N_5

Per sapere se un reticolo è distributivo basta controllare che non sia presente una di queste due strutture.

3.7.4 Complemento

Siano $(L, \leq)$ un reticolo **distributivo limitato** e $a \in L$. Un elemento $b \in L$ è il **complemento** di a sse

$$a \sqcap b = \underline{0} \qquad \wedge \qquad a \sqcup b = \underline{1}$$

Se $a \in L$ ha un complemento, allora questo è **unico**.

$(L, \leq)$ è un **reticolo (distributivo) complementato** sse ogni $a \in L$ ha un complemento.

Il complemento del minimo ($\underline{0}$) è sempre il massimo ($\underline{1}$), e viceversa.

3.8 Algebra di Boole

Boole provò a formalizzare le regole di **ragionamento** combinando proposizioni in base al loro **valore di verità**.

Corrispondono alla prima **formalizzazione** delle **operazioni logiche**:

- **Congiunzione** (*"e"*)
- **Disgiunzione** (*"oppure inclusivo"*)
- **Negazione** (*"non"*)

Boole prima considerò due valori di verità:

- **Vero**: 1
- **Falso**: 0

Ma presto generalizzò a strutture più complesse chiamate **algebre di Boole**.

3.8.1 Reticolo booleano

Un **reticolo booleano** è un reticolo:

- Limitato
- Distributivo
- Complementato

Un reticolo booleano $(L, \leq)$ definisce l'**algebra di Boole**

$$(L, \sqcup, \sqcap, \bar{\cdot}, \underline{0}, \underline{1})$$

Con operazioni per disgiunzione, congiunzione e negazione.

3.8.1.1 Esempio tipico

Per ogni insieme S, il reticolo $(\mathscr{P}(S), \subseteq)$ è un reticolo booleano.

Per ogni $T, T' \subseteq S$:

- $T \sqcup T' = T \cup T'$
- $T \sqcap T' = T \cap T'$
- $\overline{T} = S \setminus T$

La struttura dipende soltanto dalla **cardinalità** di S.

> Aggiungere diagrammi di Hasse #todo-uni

3.8.2 Algebra di Boole tradizionale

L'algebra di Boole "tradizionale" è definita dal reticolo $(\mathscr{P}(\{ a \}), \subseteq)$. Invece di chiamare gli elementi $\varnothing$ e $\{ a \}$, saranno 0 e 1 ("falso" e "vero").

Operazioni:

- La **disgiunzione** è data dal **join** ($\vee$)
- La **congiunzione** è data dal **meet** ($\wedge$)
- La **negazione** è data dal **complemento** ($\neg$)

Non è casuale che le operazioni su insiemi e su valori logici si somiglino. Infatti le operazioni su insiemi definiscono un reticolo di Boole. Di conseguenza, le proprietà delle operazioni si mantengono.

3.8.3 Proprietà delle operazioni logiche

- $\wedge$ e $\vee$ sono **idempotenti**, **commutative** e **associative**
- $\neg$ è **involutivo** ($\neg\neg x = x$)
- $\wedge$ e $\vee$ **distribuiscono** fra di loro

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

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

- si soddisfano le **leggi di De Morgan**

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

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

4 Automi a stati finiti e Linguaggi regolari

4.1 Automi

Gli automi sono una rappresentazione formale di un modello di calcolo. Dispositivi che **leggono** una sequenza di simboli, ed **eseguono** istruzioni basate su di essa. È un tipo particolare di una macchina di Turing.

In particolare, gli automi a stati finiti hanno tre proprietà:

- Memoria finita
- Leggono senza scrivere
- Leggono in ordine, senza tornare indietro

Vengono utilizzati per:

- Progettare circuiti digitali
- Analizzare espressioni lessicali
- Cercare parole in un file
- Verificare sistemi temporali
- …

4.1.1 Elementi di un automa

Un automa è composto da:

- 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**

L'automa comincia in uno stato iniziale e legge **un'istruzione alla volta**. 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*).

4.1.2 Definizione formale

Un **automa a stati finiti** è una quintupla $\mathcal{A} = \langle Q, \Sigma, \Delta, I, F \rangle$:

- 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 degli **stati iniziali**
- $F \subseteq Q$ è l'insieme degli **stati finali**

Esempio:

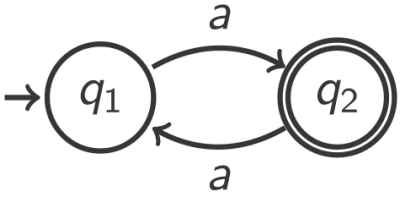
$$\langle \{ q_1, q_2 \}, \{ a \}, \{ \langle q_1, a, q_2 \rangle, \langle q_2, aq_1 \rangle \}, \{ q_1 \}, \{ q_2 \} \rangle$$

Accetta solo le sequenze dispari.

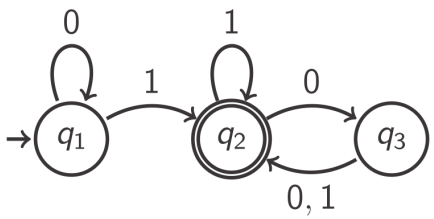
4.1.3 Rappresentazione grafica

Un automa si può rappresentare come un **grafo etichettato** $\langle Q, E, \ell \rangle$ con $\ell : E \to \mathcal{P}(\Sigma)$.

I **nodi** del grafo rappresentano gli stati e gli **archi** le transizioni. Gli stati iniziali si rappresentano con un **semiarco** (freccia senza "partenza") e gli stati finali con un **doppio bordo**.



Esempio:



È l'automa $\langle Q, \Sigma, \Delta, I, F \rangle$ dove

- $Q = \{q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- $I = \{q_1\}$
- $F = \{q_2\}$

$$\Delta = \{ \langle q_1, 0, q_1 \rangle, \quad \langle q_1, 1, q_2 \rangle, \\ \langle q_2, 0, q_3 \rangle, \quad \langle q_2, 1, q_2 \rangle, \\ \langle q_3, 0, q_2 \rangle, \quad \langle q_3, 1, q_2 \rangle \quad \}$$

4.1.4 Linguaggi

Un **alfabeto** è un insieme finito non-vuoto Σ e i suoi elementi si chiamano **simboli**. Una **stringa** (o **parola**) è una sequenza finita di simboli. Essa può essere anche **vuota** (ε). Un **linguaggio** è un insieme di parole. Può essere anche vuoto o infinito. Il linguaggio vuoto $\varnothing$ è diverso dal linguaggio composto solo dalla parola vuota $\{ \varepsilon \}$.

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

$$\text{ab} \cdot \text{aaab} = \text{abaaab}$$

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

$$\{ \varepsilon, \text{ab}, \text{aaa} \} \cdot \{ \text{bb}, \text{ba} \} = \{ \text{bb}, \text{ba}, \text{abbb}, \text{abba}, \text{aaabb}, \text{aaaba} \}$$

La concatenazione non è commutativa.

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

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

In M^* è garantita la presenza di ε.

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 $\{ 1 \}$
- $\{ \mathbf{a} \} \cdot \{ \mathbf{a}, \mathbf{b} \}^*$ è il linguaggio delle parole che iniziano con a su $\{ a, b \}$

4.2 Linguaggi regolari

La famiglia dei **linguaggi regolari** è definita ricorsivamente

- Tutti i linguaggi **finiti** sono regolari
- Se L, M sono linguaggi regolari, allora sono regolari anche
  - $L \cup M$
  - $L \cdot M$
  - L^*
  - L^+

Esempio:

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

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

4.3 Teorema di equivalenza

Il **linguaggio riconosciuto** da un automa $\mathcal{A}$ è l'insieme delle parole accettate da $\mathcal{A}$.

Teo: gli automi di stati finiti riconoscono *esattamente* i linguaggi regolari.

4.4 Costruzione di automi

Per vedere che ogni linguaggio regolare è **riconosciuto** da un automa, vediamo che

- Ogni **parola** è riconosciuta; $\varnothing$ e $\{ \varepsilon \}$ sono riconosciuti
- Se L, M sono riconosciuti, allora $L \cup M$, $L \cdot M$ e L^+ sono riconosciuti

4.4.1 Unione

Se $\mathcal{A} = (Q_1, \Sigma, \Delta_1, I_1, F_1)$ e $\mathcal{B} = (Q_2, \Sigma, \Delta_2, I_2, F_2)$ riconsocono L e M ($Q_1 \cap Q_2 = \varnothing$), allora

$$(Q_1 \cup Q_2, \Sigma, \Delta_1 \cup \Delta_2, I_1 \cup I_2, F_1 \cup F_2)$$

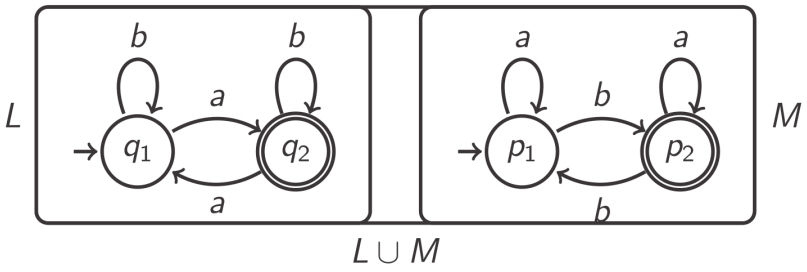
riconosce $L \cup M$.

Proviamo ad accettare con ogni automa *indipendentemente*.

Esempio:

- Alfabeto $\{ \mathrm{a}, \mathrm{b} \}$
- L sono le parole che hanno un numero dispari di a
- M sono le parole che hanno un numero dispari di b

Creare un automa per $L \cup M$



4.4.2 Concatenazione

Se $\mathcal{A} = (Q_1, \Sigma, \Delta_1, I_1, F_1)$ e $\mathcal{B} = (Q_2, \Sigma, \Delta_2, I_2, F_2)$ riconsocono L e M ($Q_1 \cap Q_2 = \varnothing$), allora

$$(Q_1 \cup Q_2, \Sigma, \Delta, I_1, F_2)$$

dove

$$\Delta := \Delta_1 \cup \Delta_2 \cup \{\, \langle q, \sigma, p\rangle \mid \langle q, \sigma, q'\rangle \in \Delta_1, q' \in F_1, p \in I_2 \,\}$$

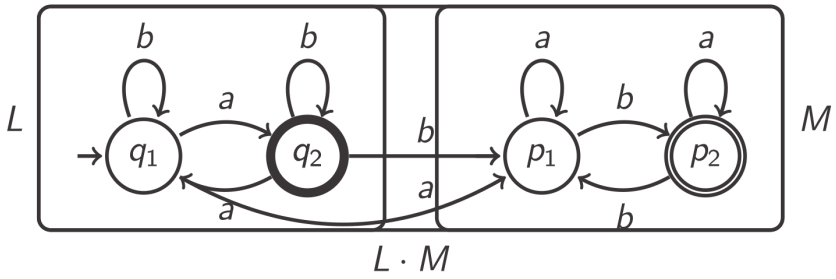
riconosce $L \cdot M$.

Accettiamo la parola in L e poi quella in M.

Esempio:

- Alfabeto { a, b }
- L sono le parole che hanno un numero dispari di a
- M sono le parole che hanno un numero dispari di b

Creare un automa per $L \cdot M$



4.4.3 Iterazione

Se $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ riconosce L, allora

$$(Q, \Sigma, \Delta', I, F)$$

dove

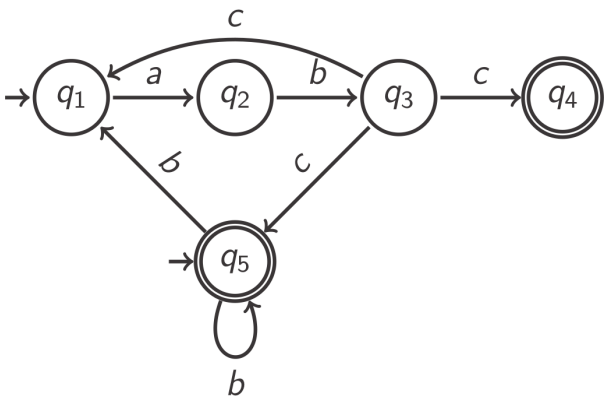
$$\Delta' := \Delta \cup \{ \langle q, \sigma, p \rangle \mid \langle q, \sigma, q' \rangle \in \Delta, q' \in F, p \in I \}$$

riconosce L^+.

Accettiamo una parola in L e ricominciamo.

Esempio:

Sia $L = \{$ abc, b $\}$. Costruire un automa che riconosce L^+



4.5 Determinismo

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

In un automa **determinista** il processo di lettura è **prefissato**: c'è un solo cammino da seguire per parola.

Formalmente, in un automa determinista:

- Δ è una funzione $Q \times \Sigma \to Q$
- I è un singoletto $I = \{ q \}$

4.6 Linguaggi

Gli automi nondeterministi sono **più generali** dei deterministi. Eppure accettano esattamente la stessa classe di linguaggi: quelli **regolari**.

Gli automi nondeterministi si possono trasformare in automi deterministi che accettano lo stesso linguaggio. Ma l'automa determinista ha bisogno di molti più stati (*costruzione dell'insieme potenza*).

5 Ricorsione e Induzione

Ricorsione e induzione sono due **principi matematici** per definire, studiare e manipolare oggetti e strutture complesse a partire da elementi semplici.

La loro caratteristica principale è l'autoreferenza, che però deve essere *ben fondata*.

In matematica e informatica il termine **ricorsione** si riferisce alla **definizione** di strutture basate su sé stesse.

Induzione si refirisce invece al processo di derivare una proprietà generale a partire da casi particolari. In matematica è un metodo di dimostrazione per gestire le strutture definite ricorsivamente.

L'uso autoreferenziale in ricorsione e induzione è utile per

- **Definire** insiemi, strutture dati, … (*definizioni ricorsive*)
- **Verificare** proprietà di questi insiemi, … (*dimostrazioni per induzione*)
- **Descrivere** metodi di calcolo e programmi su di essi (*definizioni ricorsi e algoritmi*)

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

5.1 Assiomi

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

La scelta degli assiomi può avere ripercussioni importanti. Per esempio la geometria euclidea si basa su cinque assiomi, l'ultimo dei quali è: *data una retta e un putno fuori da essa, esiste soltanto una parallela*. Diverse geometrie sono state "create" variando quest'ultimo assioma.

5.2 Ipotesi

Un'**ipotesi** è una proposizione considerata **temporaneamente vera** durante il processo di dimostrazione.

È fondamentale per l'induzione, ma anche utile in dimostrazioni dove ci sono diversi casi da analizzare.

5.3 Teorema

Un **teorema** è una conseguenza logica degli assiomi. Una proposizione che è **sempre vera** nella teoria definita da essi.

Per essere sicuri che siano teoremi, abbiamo bisogno di una **dimostrazione**.

A volte, un teorema è anche chiamato

- Lemma

- Corollario
- Proposizione

Questa scelta è guidata da una questione stilistica per distinguere la loro importanza o funzionalità.

5.4 Definizioni ricorsive

In generale, una definizione ricorsiva ha bisogno di

- Uno o più **casi base** (*base della ricorsione*)
- Una **funzione** per costruire nuovi casi da quelli esistenti (*passo ricorsivo*)

L'esempio più semplice è la definizione dei numeri naturali:

- 0 è un numero naturale
- Se n è un numero naturale, allora $s(n)$ (*il successivo di n*) è un numero naturale

5.5 Ordine naturale

I numeri naturali hanno un **ordine totale** che possiamo anche definire ricorsivamente:

- $\forall\, n \in \mathbb{N}, n < n + 1$
- Se $n < m$ e $m < l$, allora $n < l$

5.6 Buon ordinamento

In un poset $(S, \leq)$, $\leq$ è un **buon ordine** sse 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**.

Teorema: $\mathbb{N}$ è ben ordinato.

Ogni definizione per ricorsione stabilisce un ordine naturale che è un buon ordine.

5.7 Principio di induzione (generale)

Sia S un insieme definito ricorsivamente e P una proprietà. Se:

1. Dimostriamo che P è vero in ogni caso base
2. Supponiamo che P è vero per elementi generici $T \subseteq S$
3. Dimostriamo che P è vero per elementi costruiti da T tramite il passo ricorsivo

allora P è vero per **tutti** gli elementi di S.

5.8 Principio di induzione in $\mathbb{N}$

Per dimostrare che una proprietà P è vera **per ogni** $n \in \mathbb{N}$ sfruttiamo la definizione ricorsiva di $\mathbb{N}$:

- P deve essere vera per 0
- **Se** P è vera per un generico $n \in \mathbb{N}$, allora P **deve** essere vera per $n + 1$

Ovvero se:

- $P(0)$ è vero
- $P(n)$ implica $P(n + 1)$ per qualunque generico $n \in \mathbb{N}$

allora $P(n)$ è vera per ogni $n \in \mathbb{N}$.

Nota: cambiando il caso base si possono dimostrare proprietà per ogni $n \geq k$.

Una dimostrazione per induzione in $\mathbb{N}$ si svolge in **tre passi**:

1. Dimostrare il **caso base** ($n = 0$)
2. Supporre che la proprietà sia vera per un n (*ipotesi di induzione*)
3. Dimostrare che è vera anche per $n + 1$ (*passo induttivo*)

5.9 Induzione Completa

L'**induzione completa** generalizza il principio di induzione. Permette un'ipotesi di induzione più "forte" quando non basta dimostrare per il successore.

5.9.1 Principio di induzione completa in $\mathbb{N}$

Sia P una proprietà dei numeri naturali.

Se

1. Dimostriamo che $P(0)$ è vera
2. Dato un generico $n \in \mathbb{N}$ supponiamo che $P(k)$ è vero $\forall\, 0 \leq k \leq n$
3. Dimostriamo che $P(n+1)$ è vera.

Esempio: ogni numero naturale maggiore a 1 si può esprimere come il prodotto di numeri primi.

- **Base:** $(n = 2)$ 2 è un numero primo, e quindi trivialmente il prodotto di un numero primo.
- **IIG:** supponiamo che la proprietà è vera per ogni k $(2 \leq k \leq n)$.
- **Passo:** se $n+1$ è primo, allora la proprietà è trivialmente vera. Se non è primo, allora è divisibile per un numero che non è 1 o $n+1$. Cioè esistono $2 \leq \ell, m \leq n$ tali che $n+1 = \ell \cdot m$. Per l'ipotesi di induzione, ℓ e m si possono esprimere come prodotti di primi, e quindi anche $n+1$.

5.10 Definizione di insiemi

La ricorsione ci permette di costruire insiemi basati su qualche proprietà specifica. La definizione richiede i **casi base** e il **passo ricorsivo** che costruisce nuovi elementi da quelli già presenti. Tipicamente, il passo ricorsivo è basato su una **funzione**.

5.11 Funzioni e Procedure

Spesso serve la ricorsione per definire funzioni su $\mathbb{N}$ e procedure di calcolo con una struttura **ricorrente** (per esempio la funzione che calcola la sequenza di Fibonacci).

> Qui manca molta roba con gli esempi. #view-slide #todo-uni

6 Logica proposizionale: Sintassi e Semantica

6.1 Algebra booleana

Consideriamo l'algebra di Boole **più semplice**:

$$\mathbb{B} = (\{ 0, 1 \}, \leq)$$

È un reticolo complementato:

- $\neg 0 = 1, \qquad \neg 1 = 0$
- $1 \sqcap 0 = 0 \sqcap 0 = 0, \qquad 1 \sqcap 1 = 1$
- $1 \sqcup 0 = 1 \sqcup 1 = 1, \qquad 0 \sqcup 0 = 0$

Usiamo 0 e 1 per rappresentare **valori di verità**, è le operazioni del reticolo per **manipolarle**.

Il meet ($\sqcap$) in logica proposizionale viene indicato come una congiunzione ($\wedge$); il join ($\sqcup$) viene indicato come una disgiunzione ($\vee$).

6.2 Proposizioni

Le variabili in $\mathbb{B}$ si chiamano **proposizioni atomiche**. Una proposizione non è altro che un'**affermazione** che può essere vera o falsa.

6.3 Formule

Una **formula** è un'espressione (complessa) su $\mathbb{B}$ che **combina** diverse proposizioni. Anche le formule sono proposizioni (complesse) e quindi possono, a sua volta, essere vere o false. Questo dipende dalle proposizione atomiche che la compongono.

Il formalismo per studiare formule e valori di verità è la **logica proposizionale**.

6.3.1 Definizione generale di formula

Siano

- $\mathcal{A}$ un insieme **non vuoto** di proposizioni atomiche
- Op_1 un insieme di operatori (**connettivi**) unari ($\neg$)
- Op_2 un insieme di operatori binari ($\wedge, \vee$)

L'insieme $\mathcal{F}$ di **formule ben formate** su $\left(\mathcal{A}, \mathrm{Op}_1, \mathrm{Op}_2\right)$ è definito **ricorsivamente** da

- $\mathcal{A} \subseteq \mathcal{F}$ (*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 $(F \circ G) \in \mathcal{F}$

Grazie alla definizione ricorsiva di formule ben formate, le proprietà delle formule si possono dimostrare per **induzione**.

Prendiamo la logica con $\mathrm{Op}_1 = \{ \neg \}$ e $\mathrm{Op}_2 = \{ \wedge, \vee, \to, \leftrightarrow \}$.

L'insieme $\mathcal{F}$ di fbf è quindi 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 $\{ (F \wedge G), (F \vee G), (F \to G), (F \leftrightarrow G) \} \subseteq \mathcal{F}$

> ATTENZIONE ALLE PARENTESI!

6.3.2 Precedenza tra connettivi

Possiamo eliminare qualche parentesi tramite una **precedenza** tra i connettivi. In logica proposizionale, la convenzione è (in ordine di precedenza *decrescente*):

$$\neg \qquad \wedge \qquad \vee \qquad \to \qquad \leftrightarrow$$

Se ci sono più occorrenze dello stesso connettivo, si associa a **destra**.

$$\begin{aligned}
A \wedge B \vee C &\rightsquigarrow ((A \wedge B) \vee C) \\
A \to B \to C &\rightsquigarrow (A \to (B \to C)) \\
\neg A \wedge B \to C \wedge D &\rightsquigarrow (((\neg A) \wedge B) \to (C \wedge D)) \\
\neg A \wedge (B \to C) \wedge D &\rightsquigarrow ((\neg A) \wedge ((B \to C) \wedge D))
\end{aligned}$$

6.3.3 Terminologia

Da ora in poi chiameremo

- **Atomi**, variabili proposizionali o **variabili** gli elementi di $\mathcal{A}$
- **Letterali** le formule A e $\neg A$ dove $A \in \mathcal{A}$
  - A è un letterale **positivo**
  - $\neg A$ è un letterale **negativo**
- **Formule** gli elementi di $\mathcal{F}$

6.4 Unicità della scomposizione

Per ogni fbf **non atomica** F esiste **esattamente un connettivo** principale $\odot$. F è formato dall'applicazione di $\odot$ a una o due fbf. Quindi possiamo descrivere ogni fbf come un **albero sintattico** e viceversa.

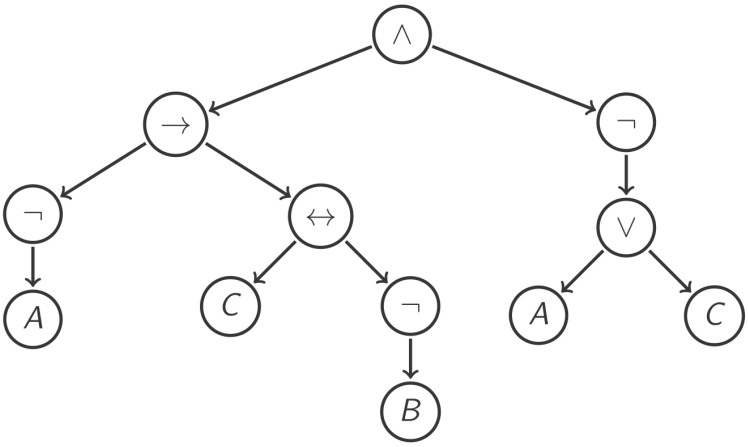
6.4.1 Albero sintattico generale

Per ogni formula $F \in \mathcal{F}$, l'albero sintattico T_F di F è un albero binario definito come

- Se $F \in \mathcal{A}$ allora T_F è un albero con un solo nodo etichettato F
- Se $F = (*G)$ con $* \in \text{Op}_1$, allora T_F ha un nodo radice etichettato $*$ e un **unico** successore T_G
- Se $F = (G_1 \circ G_2)$ con $\circ \in \text{Op}_2$, allora T_F ha
  - la radice etichettata $\circ$
  - due successori: il successore sinistro T_{G_1} e il destro T_{G_2}

Esempio:

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



Proprietà:

- L'albero sintattico è sempre un albero non vuoto
- Le foglie corrispondono sempre a proposizioni atomiche
- Tutti gli altri nodi corrispondono a connettivi
- Una stessa sottoformula può apparire più volte in un albero

6.5 Semantica

Come si **interpretano** delle formule proposizionali ben formate? Una proposizione atomica può essere vera o falsa.

6.5.1 Assegnazione booleana

Un'**assegnazione booleana** è una funzione totale

$$\mathcal{V} : \mathcal{A} \to \{ 0, 1 \}$$

$\mathcal{V}$ dice quali proposizioni atomiche sono vere e quali false.

Data questa "interpretazione", possiamo calcolare il valore di verità di ogni fbf.

Vogliamo estendere $\mathcal{V}$ a una funzione $\mathcal{F} \to \{ 0, 1 \}$.

6.5.2 I connettivi

I connettivi logici rappresentano operatori (funzioni)

- $\mathbb{B} \to \mathbb{B}$ (*operatori unari*)
- $\mathbb{B} \times \mathbb{B} \to \mathbb{B}$ (*operatori binari*)

6.5.2.1 Negazione

La formula $\neg F$ è vera sse F è falsa.

L'interpretazione di $\neg$ complementa il valore di verità.

Descrizione come **tavole di verità**

F	$\neg F$
0	1
1	0

6.5.2.2 Congiunzione

La formula $F \wedge G$ è vera sse F e G sono **entrambe** vere.

L'interpretazione di $\wedge$ calcola il **minimo** fra i valori di verità.

Descrizione come tavole di verità

F	G	$F \wedge G$
0	0	0
0	1	0
1	0	0
1	1	1

6.5.2.3 Disgiunzione

La formula $F \vee G$ è vera sse **almeno** una fra F e G è vera.

L'interpretazione di $\vee$ calcola il **massimo** fra i valori di verità.

Descrizione come tavole di verità

F	G	$F \vee G$
0	0	0
0	1	1
1	0	1
1	1	1

6.5.2.4 Implicazione

L'**implicazione** (materiale) è l'operazione logica per descrivere situazioni **condizionali**.

La formula $F \to G$ esprime "**se** F è vera **allora** G è vera". Quindi per fare $F \to G$ vera, G deve essere vera **sempre** quando F lo è. Se F è falsa, $F \to G$ è vera indipendentemente dal valore di G. Per avere un'implicazione falsa, F deve essere vera e G falsa.

La tavola di verità per l'implicazione è

F	G	$F \to G$
0	0	1
0	1	1
1	0	0
1	1	1

Questo operatore non è commutativo.

6.5.2.5 Doppia implicazione

La formula $F \leftrightarrow G$ è vera sse i valori di verità di F e G **coincidono**.

La tavola di verità per la doppia implicazione è

F	G	$F \leftrightarrow G$
0	0	1
0	1	0
1	0	0
1	1	1

6.5.3 Valutazioni

Sia $\mathcal{V} : \mathcal{A} \to \{0, 1\}$ un'assegnazione booleana. La **valutazione booleana** $\mathcal{I}_{\mathcal{V}} : \mathcal{F} \to \{0, 1\}$ è l'estensione di $\mathcal{V}$ che soddisfa la semantica dei connettivi.

Esempio: $\mathcal{V}(A) = \mathcal{V}(B) = \mathcal{V}(C) = 0$

- $\mathcal{I}_{\mathcal{V}}(\neg A) = \mathcal{I}_{\mathcal{V}}(\neg B) = 1$
- $\mathcal{I}_{\mathcal{V}}(C \leftrightarrow \neg B) = 0$
- $\mathcal{I}_{\mathcal{V}}(\neg A \to (C \leftrightarrow \neg B)) = 0$
- $\mathcal{I}_{\mathcal{V}}(A \vee C) = 0$
- $\mathcal{I}_{\mathcal{V}}(\neg(A \vee C)) = 1$
- $\mathcal{I}_{\mathcal{V}}((\neg A \to (C \leftrightarrow \neg B)) \wedge \neg(A \vee C)) = 0$

6.5.4 Propagazione in albero sintattico

La costruzione della valutazione di una formula si può dedurre dalla **propagazione** sull'albero sintattico.

Cominciando dalle **foglie** (assegnate da $\mathcal{V}$) si traversa l'albero verso la radice, applicando la definizione degli operatori.

6.6 Analisi di formule

Un metodo per saper il "comportamento" di una formula sotto **tutte** le assegnazioni possibili è costruire la sua **tavola di verità**.

Per esempio la tavola di verità di

$$F = (\neg A \to (C \leftrightarrow \neg B)) \wedge \neg(A \vee C) = G_1 \wedge G_2$$

					$C \leftrightarrow$				
A	B	C	$\neg A$	$\neg B$	$\neg B$	G_1	$A \vee C$	G_2	F
0	0	0	1	1	0	0	0	1	0
0	0	1	1	1	1	1	1	0	0
0	1	0	1	0	1	1	0	1	**1**
0	1	1	1	0	0	0	1	0	0
1	0	0	0	1	0	1	1	0	0
1	0	1	0	1	1	1	1	0	0
1	1	0	0	0	1	1	1	0	0
1	1	1	0	0	0	1	1	0	0

6.7 Equivalenze

Due formule sono **equivalenti** sse non sono distinguibili tramite assegnazioni.

F e G sono equivalenti ($F \equiv G$) sse **per ogni** assegnazione $\mathcal{V}$, $\mathcal{I}_{\mathcal{V}}(F) = \mathcal{I}_{\mathcal{V}}(G)$.

In altre parole, $F \equiv G$ sse F e Go definiscono la stessa tavola di verità.

Esempio:

$$A \equiv \neg\neg A \equiv A \wedge A \equiv A \vee A \equiv A \vee \neg A \to A$$

Grazie alla composizionabilità, sappiamo che se $F_1 \equiv F_2$ allora

- $*F_1 \equiv *F_2$
- $F_1 \circ G \equiv F_2 \circ G; G \circ F_1 \equiv G \circ F_2 \; \forall G \in \mathcal{F}$

Cioè possiamo sostituire (sotto)formule con altre equivalenti senza modificare la semantica.

In più, generalizzano alla sostituzione di atomi con formule:

$$F \equiv \neg\neg F \equiv F \wedge F \equiv \dots$$

Alcune equivalenze le conosciamo dal fatto che $\mathbb{B}$ è un'algebra booleana:

- $A \equiv \neg\neg A$ (*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*)

Inoltre ci sono le leggi di De Morgan:

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

Assorbimento:

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

- $A \to B \equiv \neg A \vee B$
- $A \leftrightarrow B \equiv (A \to B) \wedge (B \to A)$
- $A \to B \equiv \neg B \to \neg A$ (*contrapposizione*)

6.7.1 Operatori superflui

Abbiamo visto che $\to$ e $\leftrightarrow$ si possono esprimere utilizzando soltanto $\neg$, $\wedge$, $\vee$.

Grazie alle leggi di De Morgan inoltre possiamo esprimere tutta la logica proposizionale utilizzando $\neg$ e uno fra $\vee$ e $\wedge$.

6.8 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 $\mathcal{V}$ un valore $\mathcal{I}_{\mathcal{V}}(F) \in \{0, 1\}$.

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

6.9 Completezza

Data una tavola di verità, possiamo costruire una formula F che la descrive come segue:

1. Per ogni assegnazione $\mathcal{V}$ associata ad 1, costruire la formula $F_{\mathcal{V}}$ della congiunzione di tutti i letterali in $\mathcal{V}$
2. Costruire la disgiunzione delle formule $F_{\mathcal{V}}$

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	

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

6.10 Modelli e contromodelli

Ogni assegnazione $\mathcal{V}$ è estesa di **forma unica** ad una valutazione $\mathcal{I}_{\mathcal{V}} : \mathcal{F} \to \{ 0, 1 \}$.

Data una formula $F \in \mathcal{F}$ e un'assegnazione $\mathcal{V}$:

- $\mathcal{V}$ è un **modello** di F sse $\mathcal{I}_{\mathcal{V}}(F) = 1$
- $\mathcal{V}$ è un **contromodello** di F sse $\mathcal{I}_{\mathcal{V}}(F) = 0$

Una formula può avere 0, 1 o più modelli e contromodelli.

6.11 Tipi di formule

Una **tautologia** è una formula che non ha contromodelli (è sempre **vera** sotto ogni valutazione).

Una **contraddizione** è una formula che non ha modelli (è sempre **falsa** sotto ogni valutazione).

Una formula che non è né una tautologia né una contraddizione è **soddisfacibile non tautologica**.

7 Apparati deduttivi e Tableaux

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

Per rappresentarle, spesso viene utilizzato un insieme con tutti gli atomi veri.

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

$\mathcal{V}$ è la **funzione caratteristica** dell'insieme.

7.1 Conseguenze generali

Sia $\mathcal{F}$ l'insieme di tutte le **formule** di una logica.

Una relazione di conseguenza è una relazione

$$\vDash \subseteq \mathscr{P}(\mathcal{F}) \times \mathcal{F}$$

(viene utilizzata la notazione infissa $\Gamma \vDash F$ dove $\Gamma \subseteq \mathcal{F}, F \in \mathcal{F}$).

Si legge: "F è una conseguenza di Γ".

7.1.1 Conseguenze classiche

In logica classica, la relazione di conseguenza è definita da:

- una classe $\mathcal{M}$ di **interpretazioni**
- una **relazione di soddisfacibilità** $\vDash \subseteq \mathcal{M} \times \mathcal{F}$

Dati $M \in \mathcal{M}$ e $F \in \mathcal{F}$, se $M \vDash F$ diciamo che

- M **soddisfa** F
- M è un **modello** di F

$\Gamma \vDash F$ sse **ogni** modello di Γ soddisfa anche F.

7.1.2 Logica proposizionale

In logica proposizionale:

- $\mathcal{M}$ è l'insieme di tutte le **assegnazioni**
- per $\mathcal{V} \in \mathcal{M}$, $\mathcal{V} \vDash F$ sse $\mathcal{I}_{\mathcal{V}}(F) = 1$

Quindi, F è una **conseguenza** di Γ se per ogni assegnazione $\mathcal{V}$ che soddisfa **tutte** le formule in Γ

$$\mathcal{I}_{\mathcal{V}}(F) = 1$$

7.2 Terminologia

Una formula è **valida** sse $\varnothing \vDash F$ ($\vDash F$). Cioè se è una conseguenza dall'insieme vuoto di assiomi.

Dal punto di vista classico, F è valida sse **ogni** interpretazione soddisfa F, quindi sse F è una **tautologia**.

D'ora in poi parleremo principalmente di relazioni di conseguenza **classiche**.

7.3 Spostamenti

Cosa vuol dire $F \vDash G$?

Formalmente, per ogni interpretazione M, se $M \vDash F$ allora $M \vDash G$. In altre parole, sempre che F sia vera, G dovrà **necessariamente** essere vera

$$F \vDash G \quad \text{sse} \quad \vDash F \to G$$

7.3.1 Dimostrazione

Vediamo il caso **proposizionale**

$$F \vDash G \quad \text{sse} \quad \vDash F \to G$$

Sia $\mathcal{V}$ un'assegnazione.

- Se $\mathcal{I}_{\mathcal{V}}(F) = 0$ allora per la semantica $\mathcal{I}_{\mathcal{V}}(F \to G) = 1$
- Se $\mathcal{I}_{\mathcal{V}}(F) = 1$ allora $\mathcal{V} \vDash F$ e quindi $\mathcal{V} \vDash G$; cioè $\mathcal{I}_{\mathcal{V}}(G) = 1$. Questo significa che $\mathcal{I}_{\mathcal{V}}(F \to G) = 1$

Per il converso, se $\mathcal{V} \vDash F$, come $\mathcal{I}_{\mathcal{V}}(F \to G) = 1$, sappiamo che $\mathcal{I}_{\mathcal{V}}(G) = 1$. E quindi $\mathcal{V} \vDash G$.

7.4 Tautologie

In generale, abbiamo che

$$\Gamma, F \vDash G \quad \text{sse} \quad \Gamma \vDash F \to G$$

Cioè possiamo "spostare" una formula (F) dall'insieme di assiomi alla conseguenza (tramite un'implicazione).

Ripetendo quest'argomento per ogni assioma, abbiamo

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

7.5 Sistemi deduttivi

Un **sistema deduttivo** non è altro che un insieme di regole per **manipolare** formule. Si chiamano **regole di inferenza**.

Le regole di inferenza hanno di solito la forma

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

dove

- $F_1, \dots, F_n$ sono le **premesse** della regola
- F è la sua **conclusione**

Vuol dire che se **tutte** le premesse sono **vere** allora la conclusione è anche vera.

In ogni sistema deduttivo ci sono **assiomi** base che descrivono la semantica e altre proprietà della logica. Essendo **validi** non hanno bisogni di premesse per derivarli.

$$\frac{}{F}$$

7.6 Dimostrazioni

Dato un sistema deduttivo (insieme di regole di inferenza) una **dimostrazione** è una sequenza $F_1, F_2, \dots, F_n$ 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

$$\frac{G_1, \dots G_n}{F_i}$$

tale che $\{ G_1, \dots, G_n \} \subseteq \{ F_1, \dots, F_{i-1} \}$.

7.7 Proprietà e terminologia

Sia $F_1, \dots, F_n$ una dimostrazione.

Questa è chiamata una **dimostrazione di** F_n e diciamo che F_n ha una dimostrazione.

Notare che in questo caso F_1 è derivato da una regola senza premesse.

7.8 Teorema

Un **teorema** è una formula che ha una dimostrazione.

La notazione $\vdash F$ esprime che F è un teorema; cioè F è **derivabile** nel sistema deduttivo.

Notare che ogni formula che appare nella dimostrazione di un teorema è anch'essa un teorema.

7.9 Conseguenze deduttive

Sia Γ un insieme di formule e F una formula.

F **si deriva da** Γ sse F è un teorema del sistema deduttivo ottenuto aggiungendo tutte le formule di Γ **come assiomi** (in simboli $\Gamma \vdash F$).

In altre parole, nella dimostrazione di F, possiamo aggiungere formule di Γ.

7.9.1 Formalizzazione delle conseguenze deduttive

$\Gamma \vdash F$ sse esiste una sequenza $F_1, \dots, F_n$ di formule tale che

- $F_n = F$
- Per ogni $1 \leq i \leq n$:
  - $F_i \in \Gamma$ oppure
  - Esiste una regola $\frac{\Delta}{F_i}$ con $\Delta \subseteq \{ F_1, \dots, F_{i-1} \}$

Questa sequenza si chiama **dimostrazione** di F da Γ.

Gli elementi di Γ si chiamano **premesse** o **ipotesi**.

7.10 Chiusura e consistenza

La **chiusura deduttiva** di un insieme Γ di formule è l'insieme

$$\mathrm{Cn}(\Gamma) := \{ F \in \mathcal{F} \mid \Gamma \vdash F \}$$

di tutte le formule derivabili da Γ.

L'insieme Γ è consistente sse $\mathrm{Cn}(\Gamma) \subsetneq \mathcal{F}$. In un sistema classico, questo equivale all'impossibilità di derivare una formula F e la sua negazione $\neg F$.

7.11 Inclusione e monotonia

Un sistema deduttivo, come definito, è sempre

- **Inclusivo**: $\Gamma \subseteq \mathrm{Cn}(\Gamma)$
- **Monotono**: se $\Gamma \subseteq \Delta$ allora $\mathrm{Cn}(\Gamma) \subseteq \mathrm{Cn}(\Delta)$

Ovvero:

- **Inclusione**: se $F \in \Gamma$, allora "F" è una dimostrazione di F da Γ, quindi $\Gamma \vdash F$
- **Monotonia**: per definizione, una dimostrazione di F da Γ è anche una dimostrazione di F da qualunque insieme contenente Γ

7.11.1 Altre proprietà

Un sistema deduttivo può avere (o no) altre proprietà:

- **Compattezza**: $\Gamma \vdash F$ sse 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**: $\Gamma, F \vdash G$ sse $\Gamma \vdash F \to G$

In realtà le prime due proprietà si soddisfano sempre

> Perchè?

7.12 Collegamento tra sistemi

Abbiamo definito due sistemi:

- Un sistema **logico** (*semantico*) con una relazione di conseguenza $\vDash$
- Un sistema **deduttivo** (*sintattico*) con una relazione di derivabilità $\vdash$

- $\Gamma \vDash F$: da Γ possiamo **dedurre** F
- $\Gamma \vdash F$: da Γ possiamo **dimostrare** F

Vogliamo collegarli tali che si comportino allo stesso modo. In particolare identificare formule valide e teoremi.

7.13 Correttezza e completezza

Un sistema deduttivo è **corretto** sse per ogni $F \in \mathcal{F}$,

$$\vdash F \quad \text{implica} \quad \vDash F$$

Un sistema **corretto** è capace di dimostrare **unicamente** formule valide. Quindi ogni teorema è "vero" nel sistema logico.

Un sistema deduttivo è **completo** sse per ogni $F \in \mathcal{F}$,

$$\vDash F \quad \text{implica} \quad \vdash F$$

Un sistema **completo** è capace di dimostrare ogni formula valida.

In generale, dato un sistema logico, vogliamo un sistema deduttivo che sia **corretto e completo**. Così ogni formula valida è un teorema e viceversa. In questo modo possiamo dimenticarci della **semantica** e sviluppare metodi **sintattici** per derivare tautologie.

7.14 Decidibilità

In un sistema deduttivo corretto e completo, ogni tautologia può essere **dimostrata**. Ma non necessariamente sappiamo **come** trovare la dimostrazione. E neanche quanto sarà **lunga**.

Associato al sistema deduttivo, è utile costruire un'**algoritmo** per sviluppare dimostrazioni. Se tale algoritmo **termina** sempre, allora diciamo che la logica è **decidibile**.

7.14.1 Algoritmi

Esistono diversi algoritmi per decidere la validità di una formula. Oltre ai sistemi deduttivi, per la logica proposizionale abbiamo già visto il metodo delle tavole di verità. Un'altro metodo prende l'idea dalle regole di inferenza per **decomporre** una formula in pezzi più semplici.

7.15 Tableaux

Tableaux si riferisce a una classe di metodi di decisione sviluppati per diverse logiche.

Qui vedremo il metodo per la logica proposizionale.

La caratteristica principale del tableau è che prova a costruire **modelli** di una o più formule.

Decompone formule in sottoformule fino a trovare un modello o caiper che non ci possono essere modelli.

7.15.1 Da modelli a tautologie

In logica proposizionale, vogliamo capire se una formula è **tautologica**. Cioè se **ogni** assegnazione è un modello.

F è una tautologia

- sse $\neg F$ è una contraddizione
- sse $\neg F$ **non** ha modelli

7.15.2 Idea base

Per decidere se F è una tautologia:

- Negare F
- Provare a costruire un modello di $\neg F$
- Se esiste un modello, F **non** è tautologica; altrimenti lo è

Per decidere se F è una contraddizione:

- Provare a costruire un modello di F
- Se esiste un modello, F **non** è una contraddizione; altrimenti lo è

7.15.3 Decomposizione

Per costruire un modello, un tableau assegna valori di verità alle **sottoformule** mantenendo la semantica. Dal valore di una formula, deduce quello delle sottoformule fino ad arrivare ad una assegnazione (modello) o una contraddizione.

Per rappresentare i valori di verità, utilizziamo $T : \varphi$ e $F : \varphi$ dove φ è una formula.

7.15.3.1 Esempio

Se vogliamo un **modello** per $\neg(p \to q)$

- $T : \neg(p \to q)$
- $F : p \to q$
- $T : p, F : q$

Se vogliamo un **modello** per $p \wedge \neg p$

- $T : p \wedge \neg p$
- $T : p, T : \neg P$
- $T : p, F : p$

7.15.4 Scelte

Purtroppo la semantica a volte richiede una scelta ($\vee$). In quel caso basta che **una** possibilità produca un modello.

7.15.4.1 Esempio

Se vogliamo un **modello** per $p \wedge (\neg p \vee q)$

- $T : p \wedge (\neg p \vee q)$
- $T : p, T : \neg \vee q$
- $T : p, T : \neg p \mid T : p, T : q$
- $T : p, F : p \mid$

7.15.5 Regole dei tableaux

In ogni passo, scegliamo una formula da decomporre tramite una **regola del tableaux**.

Per ogni connettiva logica, ci servono **due** regole: una per **ogni** valore di verità.

7.15.5.1 Negazione

$$\frac{T : \neg\varphi}{F : \varphi} \qquad \frac{F : \neg\varphi}{T : \varphi}$$

7.15.5.2 Congiunzione e disgiunzione

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

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

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

7.15.5.3 Implicazioni

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

7.15.6 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

Un ramo del tableau è **aperto** sse non ha una **contraddizione atomica**. I rami aperti rappresentano assegnazioni che garantiscono il valore di verità richiesto.

7.15.7 Terminazione

Il metodo del tableau **termina** dopo un numero **finito** di passi. La **profondità** della struttura è limitata dal numero di connettivi nella formula. Ad ogni livello, al **massimo** si duplica il numero di rami. Quindi il tableau è un algoritmo di decisione.

8 Logica dei Predicati

8.1 Sintassi e Semantica

La logica proposizionale parla di proprietà **generali** ma è incapace di parlare di oggetti **specifici**.

Quindi, è anche impossibile sviluppare argomenti logici che **dipendono** da questi oggetti.

Per esempio, non possiamo fare affermazioni **esistenziali**.

- *Uno zio ha un fratello che è un genitore*
- *Esistono mammiferi carnivori e tutti i carnivori sono predatori*
- *La relazione "essere avo di" è transitiva*
- *Siccome Luca è più alto di Andrea, Andrea non può essere il più alto della classe*

Vogliamo un linguaggio logico capace di:

- Riferirsi a oggetti, concetti, proprietà e relazioni
- Fare affermazioni particolari o universali potenzialmente con pronomi e aggettivi **indefiniti**

Utilizziamo **costanti**, **variabili** e **quantificatori**.

Introdurremo una **logica di primo ordine** (FOL) dove le variabili si riferiscono a **individui** (oggetti).

Come **quantificatori** useremo

- Esiste (∃) (*un, alcuni*)
- Per ogni (∀) (*ogni, tutti*)

che fanno referenza alle variabili.

8.1.1 Simboli

Il linguaggio della logica dei predicati è definito da insiemi potenzialmente infiniti di

- **Variabili** $\mathcal{V} = x_1, x_2, \dots, y_1, \dots$
- **Simboli di costanti** $\mathcal{C} = a_1, a_2, \dots, b_1, \dots$
- **Simboli predicativi** $\mathcal{P} = P_1, Q, \dots$ associati ad un'**arietà** (*un'espressione vera o falsa*)
- **Simboli funzionali** $\mathcal{F} = f_1, \dots, g_1, \dots$ associati ad un'**arietà** (*funzioni che ritornano un altro oggetto*)

Questi insiemi formano la **segnatura** del linguaggio.

Le **formule** sono costruite tramite i costruttori logici ($\neg, \vee$) e i quantificatori ($\exists, \forall$).

- Gli insiemi $\mathcal{C}$ e $\mathcal{F}$ possono essere vuoti
- $\mathcal{P}$ non è mai vuoto
- $\mathcal{V}$ è sempre infinito

Spesso (ma non sempre) pensiamo che abbiamo il predicato di uguaglianza $=$ in $\mathcal{P}$ (l'identità).

In realtà le costanti sono un tipo particolare di funzioni (più chiaro con la semantica).

8.1.2 Arietà

I simboli perdicativi e funzionali sono associati ad un'**arietà**: il numero di **parametri** oppure **oggetti** che manipolano.

L'arietà si rappresenta con:

- un'apice (P^n: P è un predicato n-ario) o
- un quoziente (P/n)

8.1.3 Espressioni formali

Questi ingredienti insieme producono espressioni formali che ci aiutano a rappresentare la conoscenza

- Esiste un numero maggiore di 0

$$\exists\, x\,.\,(0 \leq x \land \neg(0 = x))$$

- Il successore di qualunque numero naturale è maggiore di 0

$$\forall\, x\,.\,(0 \leq s(x) \land \neg(0 = s(x)))$$

- La somma di due numeri è sempre maggiore o uguale agli addendi

$$\forall\, x\,.\,\forall\, y\,.\,(x \leq x + y \land y \leq x + y)$$

8.1.4 Quantificazione

I quantificatori si riferiscono a

- Un oggetto potenzialmente sconosciuto (∃) o
- A ogni oggetto di un dominio (∀)

8.1.4.1 Quantificazione esistenziale

- Qualche messicano vive in Italia
- Esiste una stella più grande del Sole
- Ci sono pianeti abitati
- Lorenzo viaggia con qualcuno

8.1.4.2 Quantificazione universale

- Tutti i pianeti orbitano intorno al Sole
- Ogni animale è mortale
- Non esistono creature aliene
- Tutti gli studenti hanno una matricola

8.1.5 Formalizzazione

Ora introduciamo formalmente l'insieme di **formule** della logica dei predicati.

Ci serve introdurre prima:

- I **termini**
- Gli **atomi**

8.1.5.1 Termini Data una segnatura $\mathcal{L}$, l'insieme dei termini di $\mathcal{L}$ (Term) è definito induttivamente da:

- Ogni simbolo di costante e ogni variabile è un **termine** ($\mathcal{C} \cup \mathcal{V} \subseteq$ Term)
- Se $t_1, \dots, t_n \in$ Term e f è un simbolo di funzione n-ario, allora $f(t_1, \dots, t_n)$ è un termine (**termine funzionale**)

I termini si riferiscono sempre a **oggetti**.

8.1.5.2 Atomi

L'insieme Atom degli **atomi** (o formule atomiche) è definito induttivamente da:

- $\top$ e $\bot$ sono **atomi** (tautologia e contraddizione)
- Se $t_1, \dots, t_n \in$ Term e P è un simbolo di **predicato** n-ario allora $P(t_1, \dots, t_n)$ è un atomo

Gli atomi parlano di **proprietà**; possono essere veri o falsi.

Quindi si possono combinare in formule complesse con attenzione alle variabili.

8.1.6 Formule

Le **formule** sulla segnatura $\mathcal{L}$ sono definite da:

- Ogni atomo è una formula
- Se φ è una formula, allora anche $\neg\varphi$ lo è
- Se φ, ψ sono formule, allora lo sono anche $\varphi \wedge \psi, \varphi \vee \psi, \varphi \to \psi, \varphi \leftrightarrow \psi$
- Se φ è una formula e $x \in \mathcal{V}$ allora $\exists\, x\,.\,\varphi$ e $\forall\, x\,.\,\varphi$ sono formule

Le lettere greche minuscole denotano **formule**. Quelle maiuscole denotano **insiemi di formule**.

8.1.7 Precedenza tra gli operatori

Consideriamo la **precedenza** tra gli operatori stabilita da

$$\forall, \quad \exists, \quad \neg, \quad \wedge, \quad \vee, \quad \to, \quad \leftrightarrow$$

Gli operatori, come nel caso proposizionale, associano a destra.

Per brevità, accumuliamo sequenze di quantificatori uguali in uno solo:

$$\exists x.\exists y. \rightsquigarrow \exists xy. \qquad \forall x.\forall y. \rightsquigarrow \forall xy.$$

Esempio:

$$\forall x.P(x) \to \exists yz.Q(y,z) \land \neg \forall x.R(x)$$

descrive la formula

$$\big(\forall x.P(x)\big) \to \Big(\big(\exists yz.Q(y,z)\big) \land \Big(\neg\big(\forall x.R(x)\big)\Big)\Big)$$

8.1.8 Campo d'azione

Il **campo d'azione**

- del quantificatore $\forall x$ nella formula $\forall x.\varphi$ è φ
- del quantificatore $\exists x$ nella formula $\exists x.\varphi$ è φ

8.1.9 Variabili e quantificatori

Una formula può avere variabili **senza** quantificatori:

$$\text{Alto}(x) \land \text{Rosso}(x)$$

In questo caso la variabile x è **libera** (senza quantificazione). Non possiamo assegnare un **valore di verità** alla formula.

Consideriamo invece la formula

$$\exists x.(\text{Alto}(x) \land \text{Rosso}(x))$$

che esprime che "*esiste un oggetto alto e rosso*".

Qui la variabile x è **quantificata** e la formula può acquisire un valore di verità.

8.1.9.1 Variabili di un termine

L'insieme $\text{var}(t)$ delle variabili di un termine t è definito da:

- $\text{var}(t) = \{ t \}$ se $t \in \mathcal{V}$ (t è una variabile)
- $\text{var}(t) = \varnothing$ se $t \in \mathcal{C}$ (t è una costante)
- $\text{var}(f(t_1, \dots, t_n)) = \bigcup_{i=1}^{n} \text{var}(t_i)$

Un termine t è chiuso se $\text{var}(t) = \varnothing$ (se non contiene variabili).

8.1.9.2 Variabili libere

Le variabili di un atomo sono:

- $\text{var}(\top) = \text{var}(\bot) = \varnothing$
- $\text{var}(P(t_1, \dots, t_n)) = \bigcup_{i=1}^{n} \text{var}(t_i)$

L'**occorrenza libera** di x in una formula è definita da:

- x occorre libera in un atomo φ sse x occorre in φ
- x occorre libera in $\neg\varphi$ sse x occorre libera in φ
- x occorre libera in $(\varphi \circ \psi)$ sse x occorre libera in φ o x occorre libera in ψ
- x occorre libera in $\forall z.\varphi$ sse $x \neq z$ e x occorre libera in φ
- x occorre libera in $\exists z.\varphi$ sse $x \neq z$ e x occorre libera in φ

8.1.9.3 Variabili legate

I quantificatori $\exists x$ e $\forall x$ **legano** le occorrenze libere di x nel proprio campo d'azione. Un'occorrenza di una variabile x è **legata** se non è libera. Le occorrenze legate sono esattamente quelle nel campo d'azione di un quantificatore.

8.1.10 Formule chiuse

Una formula φ è **chiusa** sse nessuna variabile occorre libera in φ.

Le formule chiuse si chiamano anche **enunciati**.

La semantica è propriamente definita unicamente per formule chiuse.

8.1.11 Semantica della logica dei predicati

Ogni formula **chiusa** riceve un valore di verità che dipende dagli elementi che la compongono.

Nella logica proposizionale si assegna un valore di verità ad ogni variabile proposizionale.
Il valore di una **formula** dipende soltanto da questa assegnazione.

Vogliamo estendere questa idea alla logica dei predicati:

- Assegnare un valore di verità ad ogni atomo
- Propagarlo a formule complesse

Dobbiamo però specificare anche gli oggetti di interesse.

8.1.12 Atomi chiusi e aperti

Gli atomi **chiusi** ricevono un valore di verità

$$\text{Professore}(\text{anna}) \qquad \text{Amico}(\text{anna}, \text{bob}) \qquad \text{Pari}(\text{matricola}(\text{bob}))$$

Ma come gestire l'uso di variabili?

$$\text{Professore}(x) \qquad \text{Amico}(y, \text{bob}) \qquad \text{Pari}(\text{matricola}(x))$$

Dipende da come **interpretiamo** questi oggetti.

8.1.13 Interpretazione

Un'**interpretazione** (o **struttura del primo ordine**) è una coppia $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ tale che

- $\Delta^{\mathcal{I}}$ è un insieme non vuoto chiamato **dominio** di $\mathcal{I}$
- $\cdot^{\mathcal{I}}$ è una **funzione di interpretazione** che associa:
  - A ogni $c \in \mathcal{C}$ un **elemento** $c^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
  - A ogni $f/n \in \mathcal{F}$ una **funzione** n-aria $f^{\mathcal{I}} : (\Delta^{\mathcal{I}})^n \to \Delta^{\mathcal{I}}$
  - A ogni $P/n \in \mathcal{P}$ una **relazione** n-aria $P^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^n$

Da notare la differenza fra i simboli e la loro interpretazione.

Esempio: consideriamo l'interpretazione $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ con

- $\Delta^{\mathcal{I}} := \{ \alpha, \beta, \gamma, \sigma \}$
- $\text{anna}^{\mathcal{I}} := \alpha, \qquad \text{bob}^{\mathcal{I}} := \beta$
- $\text{matricola}^{\mathcal{I}} := \{ \langle \beta, \sigma \rangle \}$
- $\text{Professore}^{\mathcal{I}} := \{ \gamma \}, \qquad \text{Pari}^{\mathcal{I}} := \{ \sigma \}$
- $\text{Amico}^{\mathcal{I}} := \{ \langle \alpha, \beta \rangle, \langle \beta, \gamma \rangle, \langle \gamma, \beta \rangle \}$

$$? \quad \text{Professore(anna)} \quad \text{Amico(anna, bob)} \quad \text{Pari(matricola(bob))}$$

8.1.14 Assegnazione

Data un'interpretazione $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, un'**assegnazione** (in $\mathcal{I}$) è una funzione $\eta : \mathcal{V} \to \Delta^{\mathcal{I}}$. L'assegnazione η associa un **elemento del dominio** alle variabili in $\mathcal{V}$.

Consideriamo η come una funzione **totale**.

8.1.15 Assegnazione di termini

Data un'interpretazione $\mathcal{I}$ e un'assegnazione $\eta : \mathcal{V} \to \Delta^{\mathcal{I}}$, l'assegnazione sui termini $\overline{\eta}$ è definita da:

- Per $x \in \mathcal{V}, \overline{\eta}(x) = \eta(x)$ (*variabili come in η*)
- Per $c \in \mathcal{C}, \overline{\eta}(x) = c^{\mathcal{I}}$
- Se $f/n \in \mathcal{F}$ e $t_1, \dots, t_n$ sono termini, allora $\overline{\eta}(f(t_1, \dots, t_n)) = f^{\mathcal{I}}(\overline{\eta}(t_1), \dots, \overline{\eta}(t_n))$

Sostituiamo ogni variabile x nel termine per $\eta(x)$. Per abbreviare, si scrive $t^{\mathcal{I},\eta}$ invece di $\overline{\eta}(t)$.

8.1.16 Soddisfacibilità atomica

Un'interpretazione $\mathcal{I}$ e un'assegnazione η insieme

- Associano ogni termine ad un elemento del dominio ($t^{\mathcal{I},\eta} \in \Delta^{\mathcal{I}}$)
- Determinano un valore di verità per ogni atomo:

$$\mathcal{I}, \eta \vDash P(t_1, \dots, t_n) \text{ sse } \left\langle t_1^{\mathcal{I},\eta}, \dots, t_n^{\mathcal{I},\eta} \right\rangle \in P^{\mathcal{I}}$$

$\mathcal{I}, \eta \vDash P(t_1, \dots, t_n)$ si legge "$\mathcal{I}, \eta$ soddisfano $P(t_1, \dots, t_n)$". L'atomo è vero nell'interpretazione $\mathcal{I}$ sotto l'assegnazione η.

8.1.17 Sostituzioni

Siano $\mathcal{I}$ un'interpretazione e $\eta : \mathcal{V} \to \Delta^{\mathcal{I}}$ un'assegnazione. Date $x \in \mathcal{V}$ e $d \in \Delta^{\mathcal{I}}$, l'assegnazione $\eta[x/d]$ è la funzione

$$\eta[x/d](y) := \begin{cases} \eta(y) & \text{se } x \neq y \\ d & \text{se } x = y \end{cases}$$

Modifica l'assegnazione per la variabile x.

8.1.18 Soddisfacibilità delle formule

La **soddisfacibilità** di una formula si definisce ricorsivamente (nell'interpretazione $\mathcal{I}$ rispetto all'assegnazione η).

Atomi:

- $\mathcal{I}, \eta \vDash \top$; $\quad \mathcal{I}, \eta \nvDash \bot$
- $\mathcal{I}, \eta \vDash P(t_1, \dots, t_n)$ sse $\left\langle t_1^{\mathcal{I},\eta}, \dots, t_n^{\mathcal{I},\eta} \right\rangle \in P^{\mathcal{I}}$

Operatori booleani:

- $\mathcal{I}, \eta \vDash \neg\varphi$ sse $\mathcal{I}, \eta \nvDash \varphi$
- $\mathcal{I}, \eta \vDash \varphi \wedge \psi$ sse $\mathcal{I}, \eta \vDash \varphi$ e $\mathcal{I}, \eta \vDash \psi$
- $\mathcal{I}, \eta \vDash \varphi \vee \psi$ sse $\mathcal{I}, \eta \vDash \varphi$ oppure $\mathcal{I}, \eta \vDash \psi$
- $\mathcal{I}, \eta \vDash \varphi \to \psi$ sse $\mathcal{I}, \eta \vDash \neg\varphi \vee \psi$
- $\mathcal{I}, \eta \vDash \varphi \leftrightarrow \psi$ sse $\mathcal{I}, \eta \vDash (\varphi \to \psi) \wedge (\psi \to \varphi)$

Quantificatori:

- $\mathcal{I}, \eta \vDash \exists x.\varphi$ sse *esiste* un $d \in \Delta^{\mathcal{I}}$ tale che $\mathcal{I}, \eta[x/d] \vDash \varphi$
- $\mathcal{I}, \eta \vDash \exists x.\varphi$ sse *per ogni* $d \in \Delta^{\mathcal{I}}$ si verifica $\mathcal{I}, \eta[x/d] \vDash \varphi$

Come **intuizione**, basta pensare che $\mathcal{I}, \eta[x/d] \vDash \varphi$ sostituisce tutte le occorrenze **libere** di x in φ per d.

8.1.19 Modelli

L'interpretazione $\mathcal{I}$ è un **modello** della formula φ sse **per ogni** assegnazione η si verifica che $\mathcal{I}, \eta \vDash \varphi$. In questo caso scriviamo $\mathcal{I} \vDash \varphi$ e diciamo che φ è **vera** in $\mathcal{I}$.

La formula φ è **valida** (o tautologica) sse φ è vera in ogni interpretazione $\mathcal{I}$. In questo caso scriviamo $\vDash \varphi$.

8.1.20 Formule chiuse e assegnazioni

Nelle formule chiuse, le variabili hanno un significato nella formula indipendente dall'assegnazione.

$$\mathcal{I}, \eta \vDash \exists x.\varphi \qquad \rightsquigarrow \qquad \mathcal{I}, \eta[x/d] \vDash \varphi$$

L'assegnazione **originale** di x in η diventa **irrilevante**.

In una formula **chiusa** tutte le variabili sono legate da quantificatori. Possiamo costruire le assegnazioni finali procedendo dall'esterno verso l'interno della formula.

8.1.21 Equivalenze

- $\forall x.\varphi \equiv \neg\exists x.\neg\varphi$
- $\exists x.\varphi \equiv \neg\forall x.\neg\varphi$
- $\exists x.\exists y.\varphi \equiv \exists y.\exists x.\varphi$
- $\forall x.\forall y.\varphi \equiv \forall y.\forall x.\varphi$
- Se x non occorre in φ, allora $\varphi \equiv \exists x.\varphi \equiv \forall x.\varphi$
- $\forall x.(\varphi \wedge \psi) \equiv \forall x.\varphi \wedge \forall x.\psi$

- $\exists x.(\varphi \vee \psi) \equiv \exists x.\varphi \vee \exists x.\psi$

Se x non occorre libera in ψ, allora:

- $\forall x.(\varphi \vee \psi) \equiv \forall x.\varphi \vee \forall x.\psi$
- $\exists x.(\varphi \wedge \psi) \equiv \exists x.\varphi \wedge \exists x.\psi$
- $\forall x.\varphi \to \psi \equiv \exists x.(\varphi \to \psi)$
- $\exists x.\varphi \to \psi \equiv \forall x.(\varphi \to \psi)$
- $\psi \to \forall x.\varphi \equiv \forall x.(\psi \to \varphi)$
- $\psi \to \exists x.\varphi \equiv \exists x.(\psi \to \varphi)$

8.1.22 Conseguenze logiche

Un insieme di formule Γ è **soddisfacibile** (o **consistente**) sse esiste un'interpretazione $\mathcal{I}$ tale che $\mathcal{I} \vDash \varphi$ per ogni $\varphi \in \Gamma$. Tale $\mathcal{I}$ è un **modello** di Γ ($\mathcal{I} \vDash \Gamma$).

φ è una **conseguenza** di Γ (scritto $\Gamma \vDash \varphi$) sse ogni modello di Γ è un modello di φ ($\mathcal{I} \vDash \Gamma \implies \mathcal{I} \vDash \varphi$).

8.1.23 Definizione di teoria

Una **teoria** è un insieme di formule Θ che è chiuso rispetto alle conseguenze.

$$\text{Se } \Theta \vDash \varphi \text{ allora } \varphi \in \Theta$$

Sia Γ un insieme di formule, chiamati **assiomi**. La **teoria generata da** Γ è l'insieme di tutte le conseguenze di Γ.

$$\Theta_\Gamma := \{ \varphi \mid \Gamma \vDash \varphi \}$$

8.1.24 Proprietà delle teorie

Una teoria del primo ordine Θ è:

- **Consistente** sse non esiste una formula φ tale che $\Theta \vDash \varphi$ e $\Theta \vDash \neg\varphi$ sse esiste φ tale che $\Theta \nvDash \varphi$
- **Completa** sse per ogni formula φ, $\Theta \vDash \varphi$ oppure $\Theta \vDash \neg\varphi$

8.1.25 Teoria dei grafi

L'insieme di tutti i grafi è trivialmente definito da una segnatura con un solo simbolo di relazione binaria "*Arco*".

Ogni interpretazione deve definire:

- Un dominio (*i nodi del grafo*)
- Una relazione binaria che interpreta Arco (*gli archi*)

Teoria dei grafi irriflessivi: $\forall x.\neg\text{Arco}(x, x)$.

Teoria dei grafi non-orientati: $\forall xy.(\text{Arco}(x, y) \rightarrow \text{Arco}(y, x))$

8.1.26 Teoria dei numeri

1. $\forall x.\neg(x < 0)$
2. $\forall xy.(x < \text{s}(y) \leftrightarrow (x < y \vee x = y))$
3. $\forall x.(x \neq 0 \rightarrow \exists y.(x = \text{s}(y)))$
4. $\forall xy.(x < y \rightarrow \neg(y < x))$
5. $\forall xyz.(x < y \wedge y < z \rightarrow x < z)$
6. $\forall xy.(x < y \vee x = y \vee y < x)$

8.2 Rappresentazione della conoscenza

Gli elementi principali della logica dei predicati sono:

- Costanti
- Simboli relazionali
- Simboli funzionali
- Variabili (soltanto quantificate)

La semantica è basata su **interpretazioni**. Un interpretazione ha un dominio Δ (non vuoto) e una funzione d'interpretazione $\cdot^{\mathcal{I}}$ che "definisce" ogni simbolo:

- Una costante in un elementi di Δ
- Un simbolo relazionale n-ario in una relazione in Δ^n
- Un simbolo funzionale n-ario in una funzione $\Delta^n \to \Delta$

Le variabili non sono interpretate: dipendono dalla quantificazione.

8.2.1 Valori di verità

Una formula è un **predicato** che può essere vero o falso. Dipende dall'interpretazione che si considera per la logica, il **nome** del simbolo è irrilevante. Anche se esistono delle **tautologie** e delle **contraddizioni**.

Vogliamo utilizzare la logica per **descrivere** un **dominio** (mondo) d'interesse. Utilizziamo formule come **assiomi** che devono essere veri. Le interpretazioni che le **falsificano** sono irrilevanti. Diamo un **significato** ai simboli, in relazione agli altri.

8.2.2 Dominio vs realtà

Le formule descrivono un **dominio** che può essere o meno collegato alla realtà (*è possibile descrivere un mondo diverso dal nostro*).

8.2.3 Conoscenza come restrizione

Quando rappresentiamo conoscenza, limitiamo la classe di interpretazioni d'interesse. Ogni assioma è una restrizione imposta alle interpretazioni. In generale, esistono molte interpretazioni che soddisfano questi assiomi. Si potrebbe sempre aggiungere conoscenza (assiomi) più e più dettagliata, ma non è sempre necessario.

8.2.4 Conoscenza incompleta

Una **base di conoscenza**e è sempre incompleta. Descriviamo **quanto basta** per l'applicazione.

Vogliamo rappresentare al meglio possibile un dominio. Allo stesso tempo, più dettagli producono una base di conoscenza più **complessa**. Si deve trovare il giusto punto medio.

8.2.5 Da formule a linguaggio naturale

Vogliamo trasformare la **conoscenza umana**e in un insieme di formule logiche. Il problema è che il linguaggio naturale è **ambiguo** mentre la logica non lo è.

8.2.6 Leggere 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 importante ricordarsi di costruire formule ben formate (e chiuse).

8.3 Tableaux

Dopo che abbiamo costruito una **base di conoscenza** vogliamo dedurre le **conseguenze** di essa. Il processo di **ragionamento** non è altro che la dimostrazione di una tautologia. Quindi dobbiamo sviluppare un metodo per dimostrare che una formula è una tautologia.

Adattiamo il metodo dei tableaux per il caso predicativo. Ricordiamo che il tableau è un metodo che genera **modelli**: interpretazioni che soddisfano la formula.

Nel caso predicativo, dobbiamo tenere in conto anche il **dominio** e le entità **anonime**.

8.3.1 Esempio 1

$$\exists x.P(x)$$

Definiamo un dominio $\Delta = \{ a \}$ e il predicato $P^{\mathcal{I}} = \{ a \}$.

Sostituisco gli elementi del dominio nelle variabili uno a uno.

$$P(a)$$

Questo è **vero**.

Un altro esempio utilizzando gli stessi dati:

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

Dividiamo la formula in due

$$\exists x.P(x), \quad \forall y.\neg P(y)$$

e continuiamo. Per il $\forall$ utilizziamo a perchè appare nella prima formula. Questo però rimane perchè potrebbe generare altri elementi più avanti.

$$P(a), \quad \neg P(a), \quad \forall y.\neg P(y)$$

Notiamo subito una **contraddizione**.

8.3.2 Indecidibilità

La logica dei predicati è **indecidibile**. Nessun metodo può decidere in **tempo finito** se una formula è tautologica o meno.

Diversamente al caso proposizionale, il tableaux predicativo può **non finire mai**.

8.3.3 Restrizioni

Per semplificare la descrizione, consideriamo formule **senza simboli funzionali**. In realtà per il tableau non sono diversi dalle costanti, quindi non aggiungono niente al metodo.

I predicati **senza variabili** sono essenzialmente proposizioni atomiche

$$P(a) \land R(b,a) \to P(c) \qquad \approx \qquad pa \land rba \to pc$$

Le regole per le connettive logiche proposizionali ($\land, \lor, \neg, \to, \leftrightarrow$) si comportano come nel caso proposizionale. Quindi manca soltanto capire come gestire i quantificatori e le loro variabili.

8.3.4 Esistenziali positivi

Se abbiamo una formula

$$T : \exists x.\phi(x)$$

il tableau la sostituisce per

$$T : \phi(a)$$

dove a è una **nuova** costante.

8.3.5 Universali negativi

Se abbiamo una formula

$$F : \forall x.\phi(x)$$

il tableau la sostituisce per

$$F : \phi(a)$$

dove a è una nuova costante.

8.3.6 Altri casi

Per i casi restanti, dobbiamo stare molto attenti: fanno referenza a tutti gli elementi del dominio.

8.3.7 Universali positivi

La formula

$$T : \forall x.\phi(x)$$

dice che **tutti** gli elementi del dominio devono soddisfare ϕ. Quindi dobbiamo introdurre $\phi(a)$ per **ogni** costante a che esiste nel tableau. Ma dobbiamo tenere conto che **nuovi oggetti** possono apparire più avanti nello sviluppo. E ricordare che il dominio **non può essere vuoto**.

La formula

$$T : \forall x.\phi(x)$$

è sostituita da

$$T : \phi(a), \quad T : \forall x.\phi(x)$$

dove a è una costante nel tableau. Se non esiste, introdurre una **nuova** costante.

Nota: la formula originale quantificata universalmente riappare nella conclusione.

8.3.8 Esistenziali negativi

La formula

$$F : \exists x.\phi(x)$$

dice che **nessun**e elemento del dominio soddisfa ϕ.

È sostituita da

$$F : \phi(a), \quad F : \exists x.\phi(x)$$

dove a è una costante nel tableau o una nuova costante, se non ne esiste alcuna.

8.3.9 Processo del tableau

Si applicano le regole del tableau finchè non si possono applicare più. Il tableau è **aperto** se non c'è una contraddizione ovvia (senza variabili). Ogni tableau completo aperto rappresenta l'esistenza di un **modello**.

Se da $F : \phi$ il tableau finisce con tutti i rami chiusi, ϕ è una **tautologia**.

Se da $T : \phi$ il tableau finisce con tutti i rami chiusi, ϕ è una **contraddizione**.

Altrimenti è **soddisfacibile non tautologica**.