

Corso di Laurea Triennale in Informatica

Università di Pisa

FONDAMENTI DELL'INFORMATICA

Dispense per il corso di Laurea in Informatica

**Filippo Bonchi, Alessio Conte, Andrea Corradini,
Roberto Grossi**

[Anno accademico 2023-24]

11 settembre 2023



Indice

Indice	i
1 Insiemi	1
1.1 Rappresentare gli insiemi	1
1.2 Confrontare gli insiemi	4
1.3 Operazioni su insiemi	6
1.4 Leggi	9
1.5 Insiemi di insiemi	16
1.6 Prodotto cartesiano	19
1.7 Cardinalità	21
1.8 Esercizi	22
2 Relazioni	25
2.1 Nozioni di Base	25
2.2 Operazioni su relazioni	27
2.3 Leggi	33
2.4 Proprietà di relazioni	35
2.5 Funzioni	41
2.6 Biiezioni	48
2.7 n -uple, sequenze di lunghezza fissata e arbitraria	53
3 Induzione Matematica	57
3.1 I naturali, i numeri triangolari e la formula di Gauss	57
3.2 Sommatorie, Produttorie, Unioni e Intersezioni n -arie.	63
3.3 Numeri di Fibonacci	67
3.4 Princípio di Induzione Forte sui naturali	68
3.5 Esercizi	69
4 Relazioni su un insieme	71
4.1 Nozioni di Base	71
4.2 Proprietà di relazioni su un insieme	72
4.3 Chiusure	75
4.4 Relazioni di equivalenza	83
4.5 Relazioni di Ordinamento	87
5 Grafi	91
5.1 Grafi orientati	91
5.2 Cammini, cicli e connettività nei grafi orientati	95
5.3 Grafi orientati aciclici (DAG)	100
5.4 Grafi non orientati	102
5.5 Cammini, cicli e connettività nei grafi non orientati	104
5.6 Alberi	107
5.7 Cammini euleriani e cammini hamiltoniani	110

5.8 Distanza su grafi	113
5.9 Isomorfismo	115
5.10 Altri grafi noti	117
5.11 Esercizi	117
6 Calcolo Combinatorio	119
6.1 Operazioni su insiemi e cardinalità	119
6.2 Relazioni e cardinalità	123
6.3 Permutazioni, disposizioni e combinazioni	126
6.4 Contare nei grafi	132
7 Induzione Strutturale e Ricorsione	141
7.1 Liste	141
7.2 Alberi binari	148
7.3 Alberi Binari Etichettati	151
7.4 Induzione Strutturale	155
7.5 Funzioni ricorsive	162
7.6 Tipologie di Ricorsione	167
8 Linguaggi Formali	169
8.1 Alfabeti, Parole e Linguaggi	170
8.2 Automi	174
8.3 Grammatiche Libere da Contesto	183
8.4 Uno studio comparativo di automi e grammatiche	192
8.5 Espressioni Regolari	194
9 Logica Matematica	199
9.1 Il Calcolo Proposizionale: sintassi e semantica	200
9.2 Tavole di verità e tautologie	205
9.3 Dimostrazioni, nel calcolo proposizionale e oltre	209
9.4 Esercizi su calcolo proposizionale	218
9.5 Cenni di logica dei predicati: motivazioni e sintassi	222
9.6 Formalizzazione di frasi	226
9.7 Interpretazioni e semantica delle formule predicative	228
9.8 Dimostrazioni per sostituzione di formule valide	234
Indice analitico	239

CAPITOLO 1

Insiemi

In questo capitolo esamineremo la nozione di *insieme*, richiamando brevemente le principali operazioni di confronto e di composizione tra insiemi e le leggi che le regolano. Inoltre vedremo come dimostrare l'uguaglianza di insiemi usando tre tecniche diverse: le dimostrazioni basate sui diagrammi di Eulero-Venn, intuitive e che non richiedono particolare creatività, ma di applicabilità limitata; le dimostrazioni per sostituzione, che sfruttano le leggi sugli insiemi e alcune semplici tecniche di manipolazione algebrica; e le dimostrazioni discorsive, basate direttamente sulla definizione di uguaglianza tra insiemi. Nel presentare le tecniche di dimostrazione daremo enfasi anche alla costruzione di controesempi per confutare enunciati erronei.

Presenteremo inoltre due costruzioni di importanza fondamentale per il resto del corso: l'insieme delle parti e il prodotto cartesiano tra due insiemi. Il capitolo si concluderà con il concetto di cardinalità.

1.1 Rappresentare gli insiemi

Definizione 1.1.1 (Insieme). *Un INSIEME è una collezione di oggetti, detti ELEMENTI.*

Dati un oggetto a e un insieme A , scriviamo $a \in A$ per dire che a è un elemento di A (oppure a appartiene ad A), mentre scriviamo $a \notin A$ per dire che a non è un elemento di A (oppure a non appartiene ad A). Il simbolo “ \in ” è chiamato simbolo di appartenenza.

Come vedremo più avanti, l'ordine in cui sono presentati gli elementi di un insieme non è importante, come non lo è il numero di ripetizioni di un elemento.

Gli insiemi sono quindi usati per raggruppare oggetti. Tipicamente gli oggetti vengono raggruppati perché posseggono qualche caratteristica comune, ma non necessariamente.

Esempio 1.1.2 (Alcuni insiemi).

- *I numeri interi / naturali / reali.*
- *I numeri pari / dispari / primi.*
- *Le lettere dell'alfabeto italiano / inglese.*
- *Le lettere dell'alfabeto che compaiono anche nei cognomi degli studenti.*
- *Le lettere dell'alfabeto che non compaiono come iniziali degli studenti.*
- *I giocatori di una squadra di calcio.*
- *Gli studenti dell'Università di Pisa.*
- *Gli studenti che seguono il corso di Fondamenti dell'Informatica.*
- *Gli utenti di Facebook / Twitter / Instagram.*
- *Le pagine web.*

Dagli esempi si capisce che gli elementi di un insieme possono essere di qualsiasi natura, non solo astrazioni matematiche. Normalmente useremo lettere maiuscole come A, B, C, \dots per denotare gli insiemi, e lettere minuscole come a, b, c, \dots per denotarne gli elementi.

Insiemi definiti per enumerazione

Si può definire un insieme *per enumerazione* (oppure *in modo estensionale*) elencandone tutti gli elementi tra parentesi graffe e separati da virgole. Vediamo alcuni esempi:

- I giorni della settimana: { lunedì, martedì, mercoledì, giovedì, venerdì, sabato, domenica }.
- Le ore in un giorno: { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 }.
- Le vocali $V = \{ a, e, i, o, u \}$.
- I valori di verità {t, f}. Qui il simbolo t sta per “true” (“vero” in inglese) e f sta per “false” (“falso” in inglese). Questo insieme è particolarmente importante in Informatica e viene chiamato anche l’insieme dei valori booleani:¹

$$Bool = \{ t, f \}$$

E se gli insiemi sono molto grandi, o infiniti? Si usano i puntini (...) per sottintendere una regola di enumerazione:

- I minuti in un’ora { 1, 2, ..., 60 }
- $K = \{ 1, 2, \dots, 1000 \}$ (i numeri da 1 a mille)
- $\mathbb{N} = \{ 0, 1, 2, \dots \}$ (i **naturali**)
- $\mathbb{N}^+ = \{ 1, 2, \dots \}$ (i **naturali positivi**)
- $\mathbb{Z} = \{ \dots, -2, -1, 0, 1, 2, \dots \}$ (gli **interi**)

Un insieme può anche contenere elementi di natura diversa:

- $AN = \{ a, b, \dots, z, A, B, \dots, Z, 0, 1, \dots, 9 \}$ (i caratteri alfanumerici)
- {42, mercoledì, a, 10:10, Roma}

L’insieme che non contiene nessun elemento è chiamato l’**INSIEME VUOTO** e viene rappresentato con il simbolo \emptyset :

- $\emptyset = \{ \}$

Insiemi definiti per proprietà

In alternativa si può definire un insieme *per proprietà* (oppure *in modo intensionale*) descrivendo una proprietà soddisfatta da tutti e soli i suoi elementi. Di seguito indichiamo con P una generica proprietà e con la notazione $P(a)$ il fatto che la proprietà P vale per l’elemento a (diciamo anche che a *soddisfa* P). Assumiamo inoltre che presi un elemento a e una proprietà P , o a soddisfa P , oppure no. Una proprietà definita sui numeri è per esempio “essere pari”: se la indichiamo come *Pari* allora $Pari(a)$ indica se a è un numero pari.

Dato un insieme A , l’insieme di tutti gli elementi di A che godono della proprietà P è denotato da

$$X = \{ x \mid x \in A \text{ e } P(x) \}$$

O più semplicemente:

$$X = \{ x \in A \mid P(x) \}$$

O ancora più semplicemente (quando A è ovvio dal contesto):

¹Dal nome di George Boole (1815–1864), importante logico e matematico inglese.

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

Il simbolo $|$ si legge “*tale che*” e serve a specificare una condizione, riportata subito dopo; quindi scrivere $K = \{ x \in A \mid P(x) \}$ significa definire K come l’insieme di tutti gli elementi x appartenenti ad A tali che x soddisfa P .

Vediamo alcuni esempi, usando il linguaggio naturale per specificare le proprietà (più avanti nel corso vedremo come essere più rigorosi):

- $\mathbb{N}^p = \{ n \mid n \in \mathbb{N} \text{ e } n \text{ è divisibile per } 2 \} = \{ n \in \mathbb{N} \mid n \text{ è divisibile per } 2 \} = \{0, 2, 4, 6, \dots\}$ (i numeri **naturali pari**)
- $\mathbb{N}^d = \{ n+1 \mid n \in \mathbb{N}^p \} = \{1, 3, 5, \dots\}$ (i numeri **naturali dispari**)
- $\mathbb{Q} = \{ n/m \mid n \in \mathbb{Z}, m \in \mathbb{N}^+ \}$ (i **razionali**)
- $\mathbb{R} = \{ x \mid x \text{ è un numero reale} \}$
- $\mathbb{P} = \{ p \in \mathbb{N}^+ \mid p \text{ ha esattamente due divisori distinti} \}$ (i **numeri primi**)²
- $AB = \{ a \mid a \text{ è un anno bisestile} \}$ ³

Esempio 1.1.3. Un altro esempio interessante è fornito dall’insieme delle pagine web che d’ora in avanti denoteremo con W . L’insieme

$$PippoW = \{x \in W \mid \text{la parola "Pippo" occorre in } x\}$$

rappresenta l’insieme di tutte le pagine web in cui appare la parola “Pippo”. Cercando su un motore di ricerca (ad esempio Google) la parola “Pippo”, si ottiene come risultato l’insieme $PippoW$. L’aspetto più interessante dei motori di ricerca è l’ordine in cui vengono visualizzati i risultati (nel nostro caso, gli elementi di $PippoW$), ma questo affascinante argomento esula dagli obiettivi di queste note.

I paradossi

La definizione di insieme che abbiamo dato è piuttosto intuitiva, ma poco precisa (per esempio siamo stati vaghi sul concetto di elemento... e lo resteremo). In particolare, un insieme potrebbe a sua volta essere visto come elemento di un altro insieme (un anno è un insieme di mesi, ogni mese un insieme di giorni, ogni giorno un insieme di ore,...).

Vedremo gli insiemi di insiemi più in dettaglio nella Sezione 1.5 e, per il momento, ci limitiamo ad osservare che la nostra definizione di insieme conduce a *paradossi* (inconsistenze logiche), come evidenziato da Bertrand Russell nel 1902.

Esempio 1.1.4 (il paradosso di Russel). Esistono alcuni insiemi che contengono se stessi: ad esempio, l’insieme degli insiemi non vuoti

$$NV = \{X \mid X \neq \emptyset\}$$

non è vuoto e quindi

$$NV \in NV.$$

È dunque naturale chiedersi se un certo insieme X appartiene a se stesso ($X \in X$) oppure no ($X \notin X$). Si può quindi definire l’insieme di tutti gli insiemi che contengono se stessi:

$$CS = \{X \mid X \in X\}$$

²Si noti che $1 \notin \mathbb{P}$: 1 non è un numero primo perché ha un solo divisore, se stesso. Il “primo” numero primo è 2.

³Una definizione precisa di *anno bisestile* è la seguente: a è un anno bisestile se $a > 1584$ e a è divisibile per 4 ma non per 100, oppure è divisibile per 400. Quindi in particolare $2000 \in AB$, ma $1400, 1900 \notin AB$.

1. Insiemi

In modo del tutto analogo si può definire l'insieme di tutti gli insiemi che non contengono se stessi:

$$NCS = \{X \mid X \notin X\}$$

Il paradosso arriva nel momento in cui ci chiediamo: NCS appartiene a se stesso?

$$NCS \in NCS ?$$

- se $NCS \notin NCS$ allora l'insieme NCS gode della proprietà che caratterizza tutti gli elementi di NCS , e quindi $NCS \in NCS$, in contrasto con l'ipotesi!
- se $NCS \in NCS$ allora per definizione si ha $NCS \notin NCS$ (perché tutti e soli gli elementi che soddisfano quella proprietà appartengono a NCS), di nuovo in contrasto con l'ipotesi!

Una trattazione approfondita che eviti tali paradossi esula dagli obiettivi di queste note: accenniamo solo che esistono teorie assiomatiche degli insiemi che scongiurano la presenza di paradossi. Per semplicità in seguito faremo sempre riferimento alla definizione *intuitiva* di insieme.

1.2 Confrontare gli insiemi

Spesso incontreremo durante il corso la necessità di dover dimostrare che due insiemi $S_1 = \{x \mid P_1(x)\}$ e $S_2 = \{x \mid P_2(x)\}$ definiti in maniera diversa in realtà coincidono.

Definizione 1.2.1 (Uguaglianza di insiemi). Due insiemi A e B sono UGUALI (scritto $A = B$) se hanno gli stessi elementi, cioè ogni elemento di A appartiene anche a B e ogni elemento di B appartiene anche ad A . Due insiemi A e B sono DIVERSI (scritto $A \neq B$) se non sono uguali, cioè se uno dei due insiemi contiene almeno un elemento che non appartiene all'altro.

Una conseguenza importante di questa definizione è che due insiemi che differiscono solo per l'ordine in cui vengono elencati i loro elementi sono uguali, cioè sono lo stesso insieme. Lo stesso vale se i due insiemi differiscono solo per il numero di volte in cui un elemento compare nella loro descrizione. Per questo motivo, descrivendo gli elementi di un insieme spesso si dice che “l'ordine e la molteplicità non contano”.

Esempio 1.2.2. Si considerino gli insiemi $A = \{a, e, i, o, u\}$, $B = \{o, i, a, o, i, e, u\}$, e $C = \{x \mid x \text{ è una vocale della lingua italiana}\}$. Allora abbiamo $A = B$, $B = C$ e $A = C$, cioè sono tre descrizioni dello stesso insieme.

Esempio 1.2.3.

- L'insieme $\{n \in \mathbb{N} \mid n > n^2\}$ è uguale all'insieme vuoto \emptyset .
- $\{n \in \mathbb{N}^+ \mid n \geq n^2\} = \{1\}$.
- $\{n + m \mid n \in \mathbb{N}^p, m \in \mathbb{N}^d\} = \mathbb{N}^d$
- $\{n + m \mid n \in \mathbb{N}^d, m \in \mathbb{N}^d\} = \{k \in \mathbb{N}^p \mid k > 0\}$
- $\{1, 2, 1, 2, 1, \dots, 2\} = \{1, 2\}$.

Esempio 1.2.4.

- $\{1, 2, 3\} \neq \{4, 3, 2\}$ perché, per esempio $1 \in \{1, 2, 3\}$ ma $1 \notin \{4, 3, 2\}$.
- $\mathbb{R} \neq \mathbb{Q}$ perché, per esempio, $\sqrt{2} \in \mathbb{R}$ ma $\sqrt{2} \notin \mathbb{Q}$.
- $\mathbb{Q} \neq \mathbb{Z}$ perché, per esempio, $\frac{1}{3} \in \mathbb{Q}$ ma $\frac{1}{3} \notin \mathbb{Z}$.
- $\mathbb{Z} \neq \mathbb{N}$ perché, per esempio, $-35 \in \mathbb{Z}$ ma $-35 \notin \mathbb{N}$.
- $\mathbb{N} \neq \mathbb{N}^+$ perché $0 \in \mathbb{N}$ ma $0 \notin \mathbb{N}^+$.
- $\mathbb{N}^+ \neq \emptyset$ perché, per esempio, $1 \in \mathbb{N}^+$ ma $1 \notin \emptyset$.

- $\mathbb{N}^p \neq \mathbb{N}^d$ (perché?).

Definizione 1.2.5 (Inclusione). *A è SOTTOINSIEME di B, scritto $A \subseteq B$, se ogni elemento di A è anche elemento di B. A è SOTTOINSIEME PROPRIO di B, scritto $A \subset B$, se $A \subseteq B$ e $A \neq B$. Due insiemi A e B sono DISGIUNTI se non hanno elementi in comune.*

Dalle definizioni appena introdotte deriva che possiamo sfruttare la relazione di appartenenza (\in) nel modo seguente per confrontare due insiemi:

1. per dimostrare che $A \subseteq B$ basta mostrare che ogni elemento a che appartiene ad A appartiene anche a B (si dimostra che se $a \in A$ allora $a \in B$)
2. per dimostrare che $A = B$ si può mostrare che ogni elemento di A appartiene a B e viceversa, ogni elemento di B appartiene ad A (si dimostrano separatamente le due inclusioni $A \subseteq B$ e $B \subseteq A$ come al punto 1.)
3. per dimostrare che $A \neq B$ basta esibire un elemento di A che non appartiene a B o, viceversa, un elemento di B che non appartiene ad A .
4. per dimostrare che $A \subset B$ si può mostrare che ogni elemento di A appartiene a B , ma che esiste un elemento di B che non appartiene ad A (si dimostra che $A \subseteq B$, come al punto 1. e $A \neq B$ come al punto 2.)
5. per dimostrare che A e B sono disgiunti si deve mostrare che ogni elemento di A non appartiene a B , e che ogni elemento di B non appartiene ad A

Esempio 1.2.6 (uguaglianze e inclusioni tra insiemi, motivate).

- $\mathbb{N}^+ \subseteq \mathbb{N}$.

Ogni naturale positivo $n \in \mathbb{N}^+$ è chiaramente un naturale, e quindi $n \in \mathbb{N}$.

- $\{n + m \mid n \in \mathbb{N}^d, m \in \mathbb{N}^p\} \subset \mathbb{N}^p$.

L'insieme a sinistra nell'uguaglianza è incluso in quello a destra: se prendiamo due numeri naturali dispari n e m , la loro somma è un numero naturale pari, e quindi $n + m \in \mathbb{N}^p$. Inoltre c'è un elemento di \mathbb{N}^p che non è la somma di due numeri naturali dispari, e precisamente lo zero.

- *Dato un qualsiasi insieme A , vale $\emptyset \subseteq A$.*

Infatti dovremmo mostrare che per ogni elemento a tale che $a \in \emptyset$, vale $a \in A$. Ma poiché non ci sono elementi a in \emptyset , non bisogna dimostrare nulla e l'inclusione vale in modo banale.

Questa argomentazione potrebbe sembrare poco convincente a una prima lettura. Facciamo vedere, in modo equivalente, che il contrario non è vero, cioè non può essere che $\emptyset \not\subseteq A$. Infatti, per mostrare che $\emptyset \not\subseteq A$ dovremmo trovare un elemento di \emptyset che non è in A . Ma dato che non ci sono elementi in \emptyset , questo è necessariamente falso, e quindi $\emptyset \subseteq A$.

- *Dati tre insiemi A , B e C , se $A \subset B$ e $B \subseteq C$, allora $A \subset C$.*

Sotto le ipotesi $A \subset B$ e $B \subseteq C$, dobbiamo mostrare che (1) per ogni $a \in A$ vale $a \in C$, e che (2) esiste $c \in C$ tale che $c \notin A$. Infatti, per (1), se $a \in A$ allora $a \in B$ (visto che vale $A \subset B$), e quindi $a \in C$ (visto che vale $B \subseteq C$). D'altra parte, per (2), visto che vale $A \subset B$, sappiamo che esiste un elemento $b \in B$ tale che $b \notin A$; ma poiché vale $B \subseteq C$ sappiamo che $b \in C$ e quindi scegliamo $c = b$.

Diagrammi di Eulero-Venn

I diagrammi di Eulero-Venn sono un utile strumento per facilitare il ragionamento sulle relazioni tra insiemi e sulle operazioni tra insiemi, con una notazione grafica e intuitiva. In generale, se ne consiglia l'uso per verificare rapidamente se una certa relazione tra insiemi sia da ritenersi valida oppure no prima di procedere con una dimostrazione formale. L'uso di questi diagrammi è anche utile per identificare opportuni controesempi quando una presunta relazione tra insiemi risulti falsa.

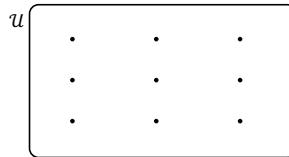
La notazione grafica di Eulero-Venn si basa sui seguenti principi:

1. Insiemi

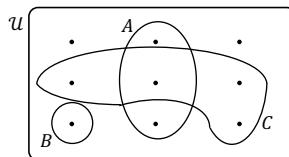
- L'insieme universo \mathcal{U} che contiene tutti gli elementi che intendiamo considerare viene rappresentato da un rettangolo.



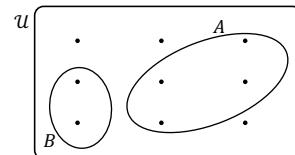
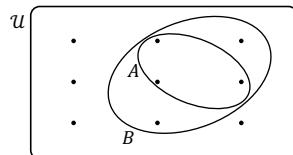
- Gli elementi vengono rappresentati come punti.



- Dentro il rettangolo usiamo circonferenze, ellissi e altre forme per rappresentare gli insiemi.



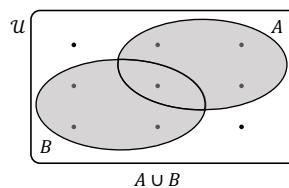
- Se A è incluso in B allora la forma che rappresenta l'insieme A viene disegnata all'interno della forma che rappresenta B , come nel prossimo diagramma a sinistra. Se A e B sono disgiunti, le due forme corrispondenti non si intersecano, come nel diagramma a destra.



1.3 Operazioni su insiemi

A partire da alcuni insiemi possiamo definirne altri selezionando elementi comuni oppure che compaiono in un insieme ma non nell'altro, oppure in uno qualsiasi degli insiemi.

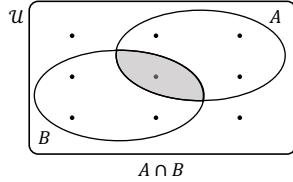
Definizione 1.3.1 (Unione). *L'UNIONE di due insiemi A e B , denotata $A \cup B$, è l'insieme che contiene (tutti e soli) gli elementi che sono elementi di A oppure di B (o anche di entrambi). In formule, $A \cup B = \{x \mid x \in A \text{ oppure } x \in B\}$.*



Esempio 1.3.2.

- $\{1, 2, 3\} \cup \{1, 3, 5\} = \{1, 2, 3, 5\}$
- $\mathbb{N}^p \cup \mathbb{N}^d = \mathbb{N}$

Definizione 1.3.3 (Intersezione). L'INTERSEZIONE di due insiemi A e B , denotata $A \cap B$, è l'insieme che contiene (tutti e soli) gli elementi che appartengono sia ad A che a B . In formule, $A \cap B = \{x \mid x \in A \text{ e } x \in B\}$.



Esempio 1.3.4.

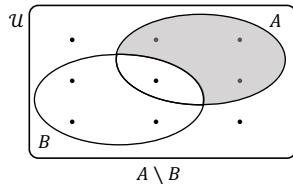
- $\{1, 2, 3\} \cap \{1, 3, 5\} = \{1, 3\}$
- $\mathbb{N}^p \cap \mathbb{N}^d = \emptyset$

Esempio 1.3.5.

- $\mathbb{N} \cup \mathbb{Z} = \mathbb{Z}$
- $\mathbb{N} \cap \mathbb{Z} = \mathbb{N}$

Esercizio 1.3.6. Se sappiamo che $A \subseteq B$, cosa possiamo dire circa $A \cup B$ e $A \cap B$? E perché?

Definizione 1.3.7 (Differenza). La differenza di un insieme A con un insieme B , scritta $A \setminus B$, è l'insieme che contiene (tutti e soli) gli elementi di A che non stanno in B . In formule, $A \setminus B = \{x \mid x \in A \text{ e } x \notin B\}$.



Esempio 1.3.8.

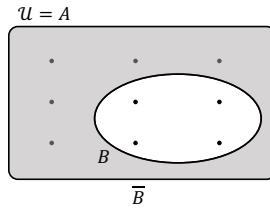
- $\mathbb{Z} \setminus \mathbb{N} = \{-n \mid n \in \mathbb{N}^+\}$
- $\mathbb{N} \setminus \mathbb{Z} = \emptyset$ (si noti che in generale $A \setminus B \neq B \setminus A$)
- $\mathbb{N} \setminus \mathbb{N}^+ = \{0\}$
- $\mathbb{N}^p \setminus \mathbb{N}^d = \mathbb{N}^p$

Definizione 1.3.9 (Complemento). Nel caso particolare in cui $B \subseteq A$ si dice che $A \setminus B$ è il COMPLEMENTO di B rispetto a A .

Se l'insieme di riferimento A è chiaro dal contesto (es. $A = \mathcal{U}$) allora si scrive semplicemente

$$\overline{B} = \{x \mid x \notin B\}$$

invece di $A \setminus B$.



Esempio 1.3.10. Assumendo come universo di riferimento l'insieme \mathbb{N} :

- $\overline{\mathbb{N}^+} = \{ 0 \}$
- $\overline{\mathbb{N}^p} = \mathbb{N}^d$
- $\overline{\mathbb{N}^d} = \mathbb{N}^p$
- $\overline{\mathbb{P}} = \{ 0, 1 \} \cup \{ n \mid n \text{ è un numero COMPOSTO} \}$
Quindi i numeri composti (o compositi) sono tutti i numeri non primi maggiori di 1.

Operatori booleani

Nella definizione di unione, abbiamo utilizzato il termine italiano *oppure*:

$$A \cup B = \{ x \mid x \in A \text{ oppure } x \in B \}$$

Nella lingua italiana, a questo termine viene spesso associato un significato esclusivo: piove *oppure* c'è il sole, rosso *oppure* verde, questo *oppure* quello... Nella teoria degli insiemi, e più in generale in Matematica e in Informatica, *oppure* ha un significato non esclusivo: quando si scrive P *oppure* Q (dove P e Q sono due proprietà) significa che può valere P , oppure Q , oppure entrambe. Per denotare questo significato, in matematica si usa spesso il simbolo \vee che talvolta si legge *or* (la traduzione dell'inglese di *oppure*). Se due insiemi A e B sono definiti come $A = \{x \mid P(x)\}$ e $B = \{x \mid Q(x)\}$ la loro unione può essere definita come

$$A \cup B = \{ x \mid P(x) \vee Q(x) \}.$$

In modo analogo, la definizione dell'intersezione fa uso della parola italiana *e*. Anche questa parola ha un significato ben preciso nella teoria degli insiemi: P e Q significa che le proprietà P e Q devono valere entrambe. Per denotare questo significato, in matematica si usa spesso il simbolo \wedge che talvolta si legge *and* (la traduzione dell'inglese di *e*). Utilizzando questo simbolo, l'intersezione di $A = \{x \mid P(x)\}$ e $B = \{x \mid Q(x)\}$ può essere definita come

$$A \cap B = \{ x \mid P(x) \wedge Q(x) \}.$$

Infine la parola italiana *non* viene denotata dal simbolo \neg che talvolta si legge *not* (traduzione dell'inglese di *no*). Anche il significato di questa parola si discosta un po' dal suo uso comune in italiano. In particolare scrivere $\neg\neg P$, per una qualche proprietà P , equivale a scrivere P , mentre in italiano la doppia negazione non sempre viene interpretata come una affermazione. Il simbolo \neg può essere utilizzato per definire il complemento di un insieme $B = \{x \mid Q(x)\}$:

$$\overline{B} = \{ x \mid \neg Q(x) \}.$$

Talvolta per indicare $\neg P$, si utilizza il simbolo $\not P$: abbiamo già visto un esempio: $x \notin X$ significa esattamente $\neg(x \in X)$.

Gli operatori \vee , \wedge and \neg , spesso chiamati *disgiunzione*, *congiunzione* e *negazione*, formano la base del Calcolo Proposizionale, un argomento che tratteremo verso la fine del corso (Capitolo 9). Fino ad allora, utilizzeremo spesso questi simboli con il significato che abbiamo appena descritto.

Gli operatori booleani sono alla base di tutte le architetture hardware e sono esprimibili in tutti i linguaggi di programmazione. Chiaramente sono utilizzati anche nella ricerca di pagine web.

Esempio 1.3.11. Anche i motori di ricerca permettono di esprimere gli operatori booleani. Ad esempio in Google, \vee è esprimibile con il termine *OR*, \wedge con il termine *AND* e \neg con il termine $-$.

Nell'Esempio 1.1.3, abbiamo visto l'insieme W delle pagine web e l'insieme $PippoW$ delle pagine web in cui occorre la parola "Pippo". Definiamo $PlutoW$ come l'insieme $\{x \in W \mid \text{la parola "Pluto" occorre in } x\}$. Adesso l'unione

$$PippoW \cup PlutoW = \{x \in W \mid \text{le parole "Pippo" o "Pluto" occorrono in } x\}$$

è il risultato che si ottiene cercando su Google

"Pippo OR Pluto".

Similmente l'intersezione

$$PippoW \cap PlutoW = \{x \in W \mid \text{le parole "Pippo" e "Pluto" occorrono in } x\}$$

è il risultato che si ottiene cercando su Google

"Pippo AND Pluto".

Infine il complemento

$$\overline{PippoW} = \{x \in W \mid \text{la parola "Pippo" non occorre in } x\}$$

è il risultato che si ottiene cercando su Google

"- Pippo".

Esempio 1.3.12. Alcuni esempi che usano gli operatori booleani per definire gli insiemi:

- $\mathbb{N} = \{n \mid n \in \mathbb{N}^+ \vee n = 0\}$ (naturali),
- $\mathbb{N}^p = \{n \mid n \in \mathbb{N} \wedge n \text{ è divisibile per } 2\}$ (naturali pari),
- $\mathbb{N}^d = \{n \in \mathbb{N} \mid \neg n \in \mathbb{N}^p\}$ (naturali dispari).

1.4 Leggi

Componendo insiemi tramite le operazioni viste è possibile ottenere lo stesso risultato in modi diversi. Si considerino per esempio gli insiemi

$$(A \cup B) \cup C \quad \text{e} \quad (A \cup C) \cup B.$$

L'insieme di sinistra è ottenuto unendo prima A con B e poi l'insieme risultante da questa unione con C ; l'insieme di destra invece, unendo prima A con C e poi con B . È facile convincersi che i due insiemi così ottenuti sono uguali, e cioè che la seguente uguaglianza è vera.

$$(A \cup B) \cup C = (A \cup C) \cup B \tag{1.1}$$

Quando ci interroghiamo sulla verità di uguaglianze come quella di sopra è solitamente opportuno specificare cosa sono gli insiemi A , B e C . Consideriamo per esempio la domanda: è vero che

$$A \cup B = A \cap B? \tag{1.2}$$

Formulata in questo modo la domanda non ha molto senso: quali insiemi A e B stiamo considerando? Due insiemi fissati oppure ci stiamo chiedendo se l'uguaglianza vale indipendentemente da quali siano i particolari insiemi A e B (cioè deve valere *per tutti* i possibili A e B)? L'uguaglianza (1.2) risulta vera se prendiamo per esempio $A = B = \{1\}$, ma risulta falsa se invece prendiamo $A = \{1, 2\}$ e $B = \{1\}$.

Ci sono uguaglianze che valgono solo per particolari insiemi, come l'uguaglianza (1.2), altre che invece valgono per qualunque insieme si consideri, come l'uguaglianza (1.1). Queste ultime sono ovviamente di maggiore interesse e sono dette *leggi*.

1. Insiemi

Alcune leggi sono ovvie, altre “credibili”, ma non sempre è così... Consideriamo per esempio la seguente domanda: è vero che le seguenti uguaglianze valgono per ogni scelta di insiemi A, B, C, D ?

$$A \cap (B \setminus C) = (A \cap B) \setminus C \quad (1.3)$$

$$(A \cup B) \setminus (B \cap A) = (A \setminus B) \cup (B \setminus A) \quad (1.4)$$

$$(A \cap B) \cup (C \cap D) = (A \cup B) \cap (C \cup D) \quad (1.5)$$

Naturalmente non possiamo verificare le uguaglianze per *tutte* le scelte degli insiemi, perché sono infinite. Quindi per rispondere positivamente a questa domanda bisogna fornirne una *prova o dimostrazione*, come vedremo tra poco.

Invece, per rispondere negativamente è sufficiente fornire un *controesempio*: essendo una legge valida per tutte le possibili scelte, basta trovare una scelta che la confuti. Nel nostro caso dobbiamo trovare degli insiemi per cui l'uguaglianza non vale.

Per esempio, l'uguaglianza (1.5) non vale scegliendo gli insiemi A, B, C, D come segue.

$$A = \{1, 2\}, B = \{2, 3\}, C = \{3, 4\}, D = \{4, 5\}$$

Infatti $A \cap B = \{2\}$ e $C \cap D = \{4\}$, mentre $A \cup B = \{1, 2, 3\}$ e $C \cup D = \{3, 4, 5\}$, da cui

$$\{2\} \cup \{4\} = \{2, 4\} \neq \{1, 2, 3\} \cap \{3, 4, 5\} = \{3\}$$

È buona prassi fornire un controesempio che sia il più “semplice” possibile. Per esempio, la seguente scelta di insiemi fornisce un controesempio più piccolo all'uguaglianza (1.5).

$$A = \{1\}, B = \emptyset, C = \{1\}, D = \emptyset$$

Possiamo adesso elencare alcune leggi fondamentali per gli insiemi.

Teorema 1.4.1. *Per tutti gli insiemi A, B, C (nell'universo \mathcal{U}) valgono le uguaglianze nelle Tabelle 1.1, 1.2 e 1.3.*

associatività	$A \cup (B \cup C) = (A \cup B) \cup C$	$A \cap (B \cap C) = (A \cap B) \cap C$
unità	$A \cup \emptyset = A$	$A \cap \mathcal{U} = A$
commutatività	$A \cup B = B \cup A$	$A \cap B = B \cap A$
idempotenza	$A \cup A = A$	$A \cap A = A$
assorbimento	$A \cup \mathcal{U} = \mathcal{U}$	$A \cap \emptyset = \emptyset$

Tabella 1.1: Leggi per \cup e \cap

distributività di \cup su \cap	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
distributività di \cap su \cup	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
assorbimento di \cup su \cap	$A \cup (A \cap B) = A$
assorbimento di \cap su \cup	$A \cap (A \cup B) = A$
complemento per \cup	$A \cup \bar{A} = \mathcal{U}$
complemento per \cap	$A \cap \bar{A} = \emptyset$

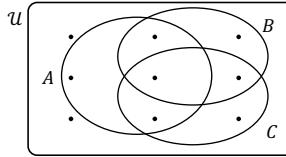
Tabella 1.2: Leggi che collegano \cup , \cap e $(\bar{\ })$

Per ognuna di queste leggi è possibile fornire una dimostrazione basata sulla definizione delle operazioni su insiemi presentate nella Sezione 1.3, o anche usando i diagrammi di Eulero-Venn. In queste note dimostriamo solamente la distributività di \cup su \cap (Tabella 1.2) e lasciamo la dimostrazione delle altre leggi come esercizio.

$$\text{differenza } A \setminus B = A \cap \overline{B}$$

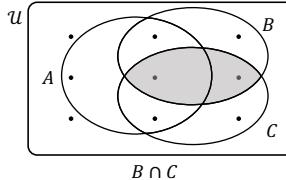
Tabella 1.3: Legge per \setminus

Dimostrazione di $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$. Vediamo una dimostrazione grafica utilizzando i diagrammi di Eulero-Venn. Abbiamo tre insiemi qualsiasi A , B e C .

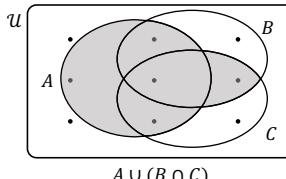


Consideriamo il membro sinistro dell'uguaglianza: $A \cup (B \cap C)$. Questo insieme è ottenuto facendo prima l'intersezione di B con C e poi unendo l'insieme risultante con A . Procediamo con cautela, per piccoli passi.

Prima coloriamo l'insieme $B \cap C$:



Poi uniamo A all'insieme ottenuto:

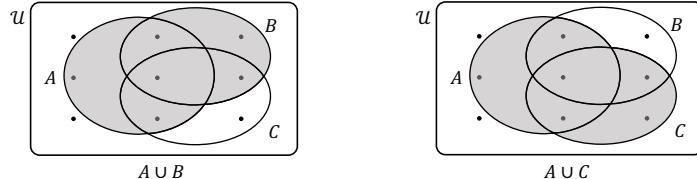


(1.6)

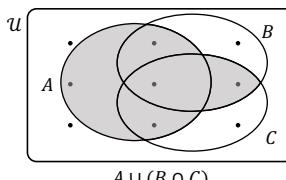
La colorazione del diagramma di sopra rappresenta esattamente l'insieme $A \cup (B \cap C)$.

Consideriamo adesso il membro destro dell'uguaglianza: $(A \cup B) \cap (A \cup C)$. Questo insieme è ottenuto facendo prima l'unione di A con B e di A con C e poi prendendo l'intersezione degli insiemi risultanti.

Prima coloriamo gli insiemi $A \cup B$ (sinistra) e $A \cup C$ (destra):



Poi prendiamo la loro intersezione:



(1.7)

La colorazione del diagramma di sopra rappresenta esattamente l'insieme $(A \cup B) \cap (A \cup C)$.

Osservando che la colorazione dei diagrammi (1.6) e (1.7) è la stessa, possiamo concludere che $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$. ■

Esercizio 1.4.2. Dimostrare le uguaglianze nelle Tabelle 1.1, 1.2 e 1.3 utilizzando i diagrammi di Eulero-Venn.

Osservazione 1.4.3 (Parentesi). Le leggi nelle Tabelle 1.1 e 1.2, così come molte altre espressioni che vedremo durante il corso, fanno uso delle parentesi tonde (e). Spieghiamo adesso, una volta per tutte, il loro significato esatto, che è stato già anticipato in modo approssimativo nella dimostrazione di $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ e all'inizio di questa sezione, prima dell'equazione (1.1).

Le parentesi tonde specificano l'ordine delle operazioni all'interno di una espressione: le operazioni racchiuse dentro le parentesi tonde vengono eseguite prima delle operazioni all'esterno. Per esempio l'espressione $A \cup (B \cup C)$ indica l'insieme ottenuto facendo prima l'unione di B con C e poi unendo il risultato con A ; invece l'espressione $(A \cup B) \cup C$ indica l'insieme ottenuto facendo prima l'unione di A con B e poi con C .

Si noti che questo uso delle parentesi è lo stesso di quello che si fa con le comuni espressioni aritmetiche. Per esempio, $x \times (y + z)$ indica che prima si deve effettuare la somma di y con z e poi moltiplicare il risultato per x .

Osservazione 1.4.4 (Efficienza). Quando gli insiemi sono molto grandi, come ad esempio PippoW e PlutoW, fare le operazioni di unione e intersezione può essere molto costoso in termini di tempo e risorse di calcolo (ad esempio memoria ed energia). È quindi conveniente fare il minor numero di operazioni possibile.

Alcune delle leggi che abbiamo introdotto in questa lezione sono molto utili a questo scopo. Ad esempio, la distributività ci permette di ridurre un calcolo che coinvolge tre operazioni (nel membro destro dell'uguaglianza) ad un calcolo che coinvolge solo due operazioni (membro sinistro).

Esercizio 1.4.5. Alcune delle leggi nelle Tabelle 1.1 e 1.2 valgono anche per gli operatori aritmetici. Se sostituiamo $+$ al posto di \cup , \times al posto di \cap , 0 al posto di \emptyset e 1 al posto di \mathcal{U} , quali uguaglianze della Tabella 1.1 valgono per tutti i numeri naturali A , B e C ?

Esercizio 1.4.6. È vero che per tutti gli insiemi A , B , C vale che $(A \cup B) \cap C = A \cup (B \cap C)$? In caso affermativo fornire una dimostrazione; in caso negativo fornire un controesempio.

Dimostrazioni per sostituzione

Per dimostrare il Teorema 1.4.1, abbiamo presentato una dimostrazione grafica utilizzando i diagrammi di Eulero-Venn. In molti casi questi diagrammi permettono di presentare prove visive, facili da seguire e convincenti anche per chi non abbia basi matematiche. Però quando le leggi coinvolgono più di tre insiemi, le dimostrazioni con i diagrammi diventano più complicate e meno leggibili.

Un'alternativa per dimostrare la validità di una legge, consiste nell'effettuare una *dimostrazione per sostituzione* utilizzando altre leggi dimostrate in precedenza. Vediamo subito come dimostrare per sostituzione l'uguaglianza in (1.1).

Esempio 1.4.7. Dimostriamo per sostituzione che per ogni A , B , C , vale che $(A \cup B) \cup C = A \cup (C \cup B)$. A tal fine è sufficiente osservare la seguente derivazione (una derivazione è una sequenza di uguaglianze giustificate da una legge dimostrata in precedenza).

$$\begin{aligned} (A \cup B) \cup C &= A \cup (B \cup C) && \text{(associatività)} \\ &= A \cup (C \cup B) && \text{(commutatività)} \end{aligned}$$

Utilizzando le leggi nelle Tabelle 1.1, 1.2 e 1.3, è possibile dimostrare per sostituzione alcune leggi molto utili.

Teorema 1.4.8. Per tutti gli insiemi A , B , C (nell'universo \mathcal{U}) valgono le uguaglianze nelle Tabella 1.4.

Come per il Teorema 1.4.1 mostriamo la prova solamente di alcune leggi e lasciamo le rimanenti per esercizio

complemento-1	$A \cup (\overline{A} \cap B) = A \cup B$	$A \cap (\overline{A} \cup B) = A \cap B$
complemento-2	$\overline{A} \cup (A \cap B) = \overline{A} \cup B$	$\overline{A} \cap (A \cup B) = \overline{A} \cap B$
convoluzione	$\overline{\overline{A}} = A$	
De Morgan	$\overline{A \cup B} = \overline{A} \cap \overline{B}$	$\overline{A \cap B} = \overline{A} \cup \overline{B}$
$\mathcal{U} : \emptyset$	$\overline{\emptyset} = \mathcal{U}$	$\overline{\mathcal{U}} = \emptyset$

Tabella 1.4: Altre leggi importanti

Dimostrazione di $A \cup (\overline{A} \cap B) = A \cup B$.

$$\begin{aligned} A \cup (\overline{A} \cap B) &= (A \cup \overline{A}) \cap (A \cup B) && \text{(distributività)} \\ &= \mathcal{U} \cap (A \cup B) && \text{(complemento per } \cup\text{)} \\ &= A \cup B && \text{(unità)} \end{aligned}$$

■

Esercizio 1.4.9. Dimostrare per sostituzione che per tutti gli insiemi A, B, C vale che $\overline{A} \cup (A \cap B) = \overline{A} \cup B$ (complemento-2, nella Tabella 1.4).

Dimostrazione di $\overline{\overline{A}} = A$.

$$\begin{aligned} A &= A \cup \emptyset && \text{(unità)} \\ &= A \cup (\overline{A} \cap \overline{\overline{A}}) && \text{(complemento per } \cap\text{)} \\ &= A \cup \overline{\overline{A}} && \text{(complemento-1)} \\ &= \overline{\overline{A}} \cup A && \text{(commutatività)} \\ &= \overline{\overline{A}} \cup (A \cap \overline{A}) && \text{(complemento-2)} \\ &= \overline{\overline{A}} \cup \emptyset && \text{(complemento per } \cap\text{)} \\ &= \overline{\overline{A}} && \text{(unità)} \end{aligned}$$

■

Esercizio 1.4.10. Dimostrare per sostituzione le leggi di De Morgan e $\mathcal{U} : \emptyset$ nella Tabella 1.4.

Proprietà di uguaglianza e inclusione

Nelle dimostrazioni per sostituzione abbiamo utilizzato tacitamente alcune banali proprietà dell'uguaglianza: per esempio, nella derivazione dell'Esempio 1.4.7, abbiamo scritto l'uguaglianza

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

giustificata dalla legge di associatività, che è

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

Si noti che le espressioni a destra ed a sinistra dell'uguaglianza sono invertite. Questo non comporta alcun problema perché ogni volta che vale $S = D$ allora vale anche $D = S$. Questa proprietà è detta *simmetria* di $=$. È importante osservare che \subseteq non gode di questa proprietà: se vale che $S \subseteq D$ non necessariamente vale $D \subseteq S$.

Esercizio 1.4.11. Per quali insiemi A e B valgono contemporaneamente $A \subseteq B$ e $B \subseteq A$?

1. Insiemi

Un'altra proprietà dell'uguaglianza che utilizziamo sempre, ma implicitamente, nelle dimostrazioni per sostituzione è la *transitività*: sapendo che $A = B$ e che $B = C$ si può concludere che $A = C$. Si consideri per esempio una generica derivazione che utilizza tre leggi:

$$\begin{aligned} E_0 &= E_1 && \text{(legge-1)} \\ &= E_2 && \text{(legge-2)} \\ &= E_3 && \text{(legge-3)} \end{aligned}$$

Questa derivazione mostra tre uguaglianze: $E_0 = E_1$, $E_1 = E_2$ ed $E_2 = E_3$. Per concludere che $E_0 = E_3$ si deve ricorrere alla transitività due volte: (1) sapendo che $E_0 = E_1$ e $E_1 = E_2$ si conclude che $E_0 = E_2$; (2) sapendo che $E_0 = E_2$ e $E_2 = E_3$ si conclude che $E_0 = E_3$.

Il seguente risultato elenca le ovvie proprietà dell'uguaglianza.

Proposizione 1.4.12. *Per tutti gli insiemi A, B, C vale che:*

- $A = A$ (*riflessività*);
- Se $A = B$ e $B = C$, allora $A = C$ (*transitività*);
- Se $A = B$, allora $B = A$ (*simmetria*).

Queste proprietà sono ovvie, nel senso che seguono immediatamente dalla definizione di uguaglianza e quindi non necessitano di una dimostrazione.

Il seguente risultato elenca le ovvie proprietà dell'inclusione.

Proposizione 1.4.13. *Per tutti gli insiemi A, B, C vale che:*

- $A \subseteq A$ (*riflessività*);
- Se $A \subseteq B$ e $B \subseteq C$, allora $A \subseteq C$ (*transitività*);
- Se $A \subseteq B$ e $B \subseteq A$, allora $A = B$ (*anti-simmetria*);

Anche in questo caso, le proprietà seguono immediatamente dalla definizione di inclusione.

È importante notare che mentre per l'uguaglianza vale la simmetria, per l'inclusione vale l'anti-simmetria. Utilizzeremo frequentemente questa proprietà nelle dimostrazioni discorsive.

Dimostrazioni discorsive

Le dimostrazioni per sostituzione sono estremamente formali e convincenti, ma hanno lo svantaggio di essere talvolta molto lunghe e difficili da scoprire. Per esempio la dimostrazione per sostituzione di $\overline{\overline{A}} = A$ che abbiamo visto prima è molto sofisticata, ma dimostrare questa legge con i digrammi di Eulero-Venn è quasi immediato.

Esercizio 1.4.14. *Dimostrare con i diagrammi di Eulero-Venn che per tutti gli insiemi A vale che $\overline{\overline{A}} = A$.*

Le dimostrazioni che probabilmente siete abituati a conoscere alternano la notazione matematica (non ambigua, universale) con frasi in linguaggio naturale (ambiguo, deve essere tradotto a seconda della nazionalità del lettore).

Sono quelle più comuni nei libri e cercano di rendere leggibili e meno tediose le prove puramente formali, guidando il lettore ma saltando alcuni passaggi logici. Sono comunque fondate su prove puramente formali, non ambigue, universalmente leggibili e verificabili.

Uno degli scopi del corso sarà quello di conciliare queste due visioni per poter costruire argomentazioni solide, per presentarle ad altri e per verificare quelle di altri.

Adesso mostriamo una prova discorsiva della legge di distributività di \cup su \cap (che abbiamo già provato con i diagrammi di Eulero-Venn).

Esempio 1.4.15. Vogliamo dimostrare che per tutti gli insiemi A, B, C vale che:

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

Anziché dimostrare l'uguaglianza direttamente possiamo dimostrare separatamente le inclusioni

$$A \cup (B \cap C) \subseteq (A \cup B) \cap (A \cup C) \quad e \quad A \cup (B \cap C) \supseteq (A \cup B) \cap (A \cup C)$$

e poi concludere l'uguaglianza utilizzando l'antisimmetria di \subseteq (Proposizione 1.4.13).

- **Prima inclusione** ($A \cup (B \cap C) \subseteq (A \cup B) \cap (A \cup C)$)

Prendiamo un qualsiasi elemento $x \in A \cup (B \cap C)$.

Vogliamo fare vedere che $x \in (A \cup B) \cap (A \cup C)$.

Dalla definizione di unione possiamo distinguere due possibilità, che consideriamo separatamente:

$$(i) x \in A \quad (ii) x \in B \cap C$$

(Si noti che questi due casi non sono necessariamente disgiunti tra loro.)

In entrambi i casi bisogna arrivare a dimostrare che $x \in (A \cup B) \cap (A \cup C)$.

– caso (i)

Siccome $x \in A$, per definizione di unione si ha che $x \in A \cup B$ e che $x \in A \cup C$.

Ma allora, visto che x appartiene a entrambi questi insiemi deve appartenere anche alla loro intersezione, ovvero $x \in (A \cup B) \cap (A \cup C)$ (per definizione di intersezione).

– caso (ii)

Siccome $x \in B \cap C$, per definizione di intersezione si ha che $x \in B$ e che $x \in C$.

Dato che $x \in B$, per definizione di unione si ha che $x \in A \cup B$.

Analogamente, dato che $x \in C$, per definizione di unione si ha che $x \in A \cup C$.

Ma allora, visto che x appartiene a entrambi questi insiemi deve appartenere anche alla loro intersezione, ovvero $x \in (A \cup B) \cap (A \cup C)$ (per definizione di intersezione).

Non ci restano altri casi da considerare e quindi la prima inclusione è dimostrata.

- **Seconda inclusione** ($(A \cup B) \cap (A \cup C) \subseteq A \cup (B \cap C)$)

Prendiamo un qualsiasi elemento $y \in (A \cup B) \cap (A \cup C)$.

Vogliamo fare vedere che $y \in A \cup (B \cap C)$.

Dato che $y \in (A \cup B) \cap (A \cup C)$, dalla definizione di intersezione sappiamo che $y \in A \cup B$ e $y \in A \cup C$.

A questo punto possiamo distinguere due possibilità, che consideriamo separatamente:⁴

$$(a) y \in A \quad (b) y \notin A$$

(Si noti che questi due casi sono disgiunti tra loro.)

In entrambi i casi bisogna arrivare a dimostrare che $y \in A \cup (B \cap C)$.

– caso (a)

Se $y \in A$ allora per definizione di unione $y \in A \cup (B \cap C)$ e abbiamo finito.

– caso (b)

Dato che $y \in A \cup B$, se $y \notin A$, allora necessariamente $y \in B$.

Analogamente, dato che $y \in A \cup C$, se $y \notin A$, allora $y \in C$.

Ma allora, visto che y appartiene a entrambi questi insiemi deve appartenere anche alla loro intersezione, ovvero $y \in B \cap C$ (per definizione di intersezione).

Per definizione di unione, $y \in A \cup (B \cap C)$ e abbiamo finito.

⁴Sapendo che $y \in A \cup B$ e $y \in A \cup C$ ci sono quattro combinazioni possibili: (1) $y \in A$ e $y \in A$, (2) $y \in A$ e $y \in B$, (3) $y \in C$ e $y \in A$, (4) $y \in B$ e $y \in C$. Le prime tre sono sussunte da $y \in A$, la quarta da $y \notin A$.

Non ci restano altri casi da considerare e quindi anche la seconda inclusione è dimostrata.

Esercizio 1.4.16. Dimostrare in modo discorsivo che per tutti gli insiemi A, B , vale che $A \cup (A \cap B) = A$ (legge di assorbimento).

Esempio 1.4.17. Dimostriamo in maniera discorsiva che per tutti gli insiemi $\overline{(A \cup B)} = \overline{A} \cap \overline{B}$ (legge di De Morgan nella Tabella 1.4).

Come prima dimostriamo separatamente le inclusioni

$$\overline{(A \cup B)} \subseteq \overline{A} \cap \overline{B} \quad \text{e} \quad \overline{A} \cap \overline{B} \subseteq \overline{(A \cup B)}$$

e poi concludiamo l'uguaglianza utilizzando l'antisimmetria di \subseteq (Proposizione 1.4.13).

- Per prima cosa facciamo vedere che $\overline{(A \cup B)} \subseteq \overline{A} \cap \overline{B}$.

Preso un qualsiasi $x \in \overline{(A \cup B)}$, dobbiamo mostrare che $x \in \overline{A} \cap \overline{B}$.

Per definizione di complemento, se $x \in \overline{(A \cup B)}$ vuol dire che $x \notin A \cup B$. Quindi, per definizione di unione, si ha che $x \notin A$ e $x \notin B$.

Dato che $x \notin A$, per definizione di complemento si ha $x \in \overline{A}$.

Analogamente, dato che $x \notin B$, per definizione di complemento si ha $x \in \overline{B}$.

Ma allora $x \in \overline{A} \cap \overline{B}$ e abbiamo finito.

- Resta da far vedere che $\overline{A} \cap \overline{B} \subseteq \overline{(A \cup B)}$.

Prendiamo un qualsiasi elemento $y \in \overline{A} \cap \overline{B}$. Dobbiamo far vedere che $y \in \overline{(A \cup B)}$.

Dato che $y \in \overline{A} \cap \overline{B}$, per definizione di intersezione si ha che $y \in \overline{A}$ e $y \in \overline{B}$.

Dunque, per definizione di complemento, si ha $y \notin A$ e $y \notin B$.

Ma se y non appartiene a A e non appartiene a B allora non può certo appartenere alla loro unione, quindi $y \notin A \cup B$.

Per definizione di complemento, questo vuol dire che $y \in \overline{(A \cup B)}$ e abbiamo finito.

Si noti che le dimostrazioni dei due casi sono composte dagli stessi passaggi ma in ordine rovesciato: in questi casi non occorre dimostrare separatamente le due inclusioni, ma basta una sequenza di “se-e-solo-se”.

1.5 Insiemi di insiemi

Alcuni problemi richiedono di investigare tutte le possibili combinazioni degli elementi di un insieme. In questi casi è utile trattare insiemi i cui elementi siano a loro volta altri insiemi, come per esempio:

$$\{\{2, 4, 6, 8\}, \{1, 3, 5, 7, 9\}\} \quad \text{e} \quad \{\{a\}, \{a, b\}, \{a, b, c\}\}$$

È importante notare che

$$\{a\} \in \{\{a\}, \{a, b\}, \{a, b, c\}\}$$

mentre invece

$$a \notin \{\{a\}, \{a, b\}, \{a, b, c\}\}.$$

Infatti a è un elemento, mentre $\{a\}$ è l'insieme che contiene solamente l'elemento a . Similmente $\{a\} \neq \{\{a\}\}$: il primo è l'insieme che contiene l'elemento a , mentre il secondo è l'insieme che contiene come elemento l'insieme $\{a\}$.

Osservazione 1.5.1. Si deve fare estrema attenzione a usare la notazione in modo corretto, specialmente per quanto riguarda l'uso delle parentesi graffe.

Possiamo adesso introdurre una costruzione che giocherà un ruolo fondamentale durante tutto il corso.

Definizione 1.5.2 (Insieme delle parti). *Dato un insieme A , il suo INSIEME DELLE PARTI $\mathcal{P}(A)$ è quell'insieme che ha come elementi tutti e soli i sottoinsiemi di A . In formule, $\mathcal{P}(A) = \{ X \mid X \subseteq A \}$.*

Esempio 1.5.3. *Dato $A = \{0, 1, 2\}$ definire $\mathcal{P}(A)$ per enumerazione.*

$$\mathcal{P}(A) = \{\emptyset, \{0\}, \{1\}, \{2\}, \{0, 1\}, \{0, 2\}, \{1, 2\}, A\}$$

Vediamo un semplice fatto che riguarda l'insieme delle parti.

Proposizione 1.5.4. *Per tutti gli insiemi A vale che:*

$$\emptyset \in \mathcal{P}(A) \quad e \quad A \in \mathcal{P}(A)$$

Dimostrazione. Dimostrare che $\emptyset \in \mathcal{P}(A)$ è, per definizione di $\mathcal{P}(A)$, equivalente a dimostrare che $\emptyset \subseteq A$. Per definizione di \subseteq , si deve dimostrare che tutti gli elementi di \emptyset sono anche elementi di A . Ma questo è banalmente vero dal momento che \emptyset non contiene alcun elemento (si veda l'Esempio 1.2.6).

Dimostrare che $A \in \mathcal{P}(A)$ è, per definizione di $\mathcal{P}(A)$, equivalente a dimostrare che $A \subseteq A$, e questo è vero per la Proposizione 1.4.13. ■

Esercizio 1.5.5. *È vero che per tutti gli insiemi A vale che $\emptyset \subset A$? In caso affermativo dare una dimostrazione. In caso negativo fornire un controesempio.*

Esempio 1.5.6.

- Definire $\mathcal{P}(\emptyset)$ per enumerazione:

$$\mathcal{P}(\emptyset) = \{\emptyset\}$$

(l'unico sottoinsieme dell'insieme vuoto è... l'insieme vuoto!)

- Definire $\mathcal{P}(\mathcal{P}(\emptyset))$ per enumerazione:

$$\mathcal{P}(\mathcal{P}(\emptyset)) = \mathcal{P}(\{\emptyset\}) = \{\emptyset, \{\emptyset\}\}$$

($\mathcal{P}(\emptyset)$ contiene un solo elemento, quindi ha due possibili sottoinsiemi.)

Famiglie di insiemi

Durante il corso troveremo spesso utile considerare famiglie di insiemi.

Definizione 1.5.7 (famiglia di insiemi). *Sia I un insieme tale che per ogni $i \in I$ sia definito un certo insieme A_i . L'insieme \mathcal{F} che ha come elementi tutti gli insiemi A_i viene detta FAMIGLIA INDICIZZATA DA I . In formule, $\mathcal{F} = \{A_i \mid i \in I\} = \{A_i\}_{i \in I}$.*

Le operazioni di unione e intersezione sono generalizzate a famiglie di insiemi come segue:

- $\bigcup \mathcal{F} = \bigcup_{i \in I} A_i$ è l'insieme degli elementi che appartengono a A_i per un qualche indice $i \in I$.
- $\bigcap \mathcal{F} = \bigcap_{i \in I} A_i$ è l'insieme degli elementi che appartengono a A_i per tutti gli indici $i \in I$.

Esercizio 1.5.8. *Sia $I = \{1, 2, \dots, n\}$, cioè l'insieme dei naturali maggiori di 0 e minori o uguali a n , e*

- $A_i = \{i\}$
- $B_i = \{1, 2, \dots, i\}$
- $C_i = \{i, i+1, i+2, \dots\}$

Determinare:

- $\bigcup_{i \in I} A_i$ e $\bigcap_{i \in I} A_i$

1. Insiemi

- $\bigcup_{i \in I} B_i$ e $\bigcap_{i \in I} B_i$
- $\bigcup_{i \in I} C_i$ e $\bigcap_{i \in I} C_i$

E se invece I fosse uguale a \mathbb{N}^+ ?

Si noti che quando $I = \{1, 2, \dots, n\}$ possiamo usare alternativamente la notazione $\bigcup_{i=1}^n A_i$ invece di $\bigcup_{i \in I} A_i$, e analogamente per l'intersezione.

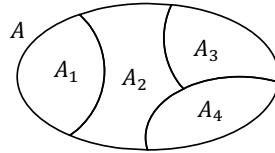
Partizioni

Quando parleremo di relazioni di equivalenza (nel Capitolo 4), ci interesseremo di particolari famiglie che *partizionano* gli elementi di un certo insieme A in sottoinsiemi separati.

Definizione 1.5.9 (partizione). *Dato un insieme A , una PARTIZIONE di A è una famiglia di insiemi $\mathcal{F} = \{A_i\}_{i \in I}$ tali che:*

- *Ogni insieme A_i è diverso da \emptyset (insiemi non vuoti).*
- $\bigcup_{i \in I} A_i = A$ (*copertura di A*);
- *Presi due indici qualsiasi i e j con $i \neq j$ si ha $A_i \cap A_j = \emptyset$ (insiemi disgiunti);*

Nel seguente diagramma $\{A_1, A_2, A_3, A_4\}$ è una partizione dell'insieme A .



Esempio 1.5.10. Un insieme di persone può essere partizionato in accordo all'anno di nascita di ciascuna persona.

Esercizio 1.5.11. È vero che gli insiemi \mathbb{N}^p (naturali pari), e \mathbb{N}^d (naturali dispari) formano una partizione di \mathbb{N} ?

Esercizio 1.5.12. Sia $I = \mathbb{N}^+$ e

- $A_i = \{i\}$
- $B_i = \{1, 2, \dots, i\}$
- $C_i = \{i, i+1, i+2, \dots\}$

Dire se è vero che le seguenti famiglie di insiemi sono partizioni di \mathbb{N}^+ :

- $\{B_5, A_6, C_7\}$
- $\{A_i\}_{i \in I}$
- $\{B_i\}_{i \in I}$
- $\{C_1\}$
- $\{(B_i \cap C_i)\}_{i \in I}$
- $\{(B_i \cap C_{i+1})\}_{i \in I}$

Esercizio 1.5.13. È possibile trovare un insieme A tale che $\mathcal{P}(A)$ sia una partizione di A ? In caso positivo fornire un tale insieme, altrimenti dimostrare che non è possibile.

Numeri naturali come insiemi

Spesso utilizzeremo i numeri naturali per denotare insiemi.

Definizione 1.5.14 (naturali come insiemi). *Per ogni $n \in \mathbb{N}$, denotiamo con n l'insieme $\{m \in \mathbb{N} \mid m < n\}$. Equivalentemente, per enumerazione, definiamo $n = \{0, 1, \dots, n - 1\}$.*

Per esempio,abbiamo che

$$\begin{aligned} 0 &= \{\} \text{ (l'insieme vuoto),} \\ 1 &= \{0\}, \\ 2 &= \{0, 1\}, \\ 3 &= \{0, 1, 2\}, \\ \dots &= \dots \end{aligned}$$

Segue immediatamente dalla definizione che per ogni $n \in N$ l'insieme denotato da n ha proprio cardinalità n , cioè $|n| = n$. Si noti inoltre che espandendo ulteriormente i numeri elencati sopra otteniamo che l'insieme denotato da ogni naturale può essere descritto usando solo parentesi graffe, senza altri elementi. Per esempio,abbiamo che

$$\begin{aligned} 0 &= \{\} \text{ (l'insieme vuoto),} \\ 1 &= \{0\} = \{\{\}\}, \\ 2 &= \{0, 1\} = \{\{\}, \{\{\}\}\}, \\ 3 &= \{0, 1, 2\} = \{\{\}, \{\{\}\}, \{\{\}, \{\{\}\}\}\}, \\ \dots &= \dots \end{aligned}$$

1.6 Prodotto cartesiano

Come detto, gli insiemi collezionano elementi in maniera *non ordinata*, però spesso è utile rappresentare collezioni ordinate del tipo:

$$(a_1, a_2, a_3, \dots, a_n)$$

Per esempio per rappresentare delle stringhe o dei vettori.⁵

Ovviamente vorremmo che le coppie ordinate (a, b) e (b, a) siano ritenute diverse, mentre sappiamo che $\{a, b\} = \{b, a\}$.

Definizione 1.6.1 (Prodotto cartesiano). *Siano A e B due insiemi. Il PRODOTTO CARTESIANO di A per B , scritto $A \times B$, è l'insieme formato da tutte e sole le coppie ordinate (a, b) tali che $a \in A$ e $b \in B$. In formule, $A \times B = \{(a, b) \mid a \in A, b \in B\}$.*

Osservazione 1.6.2. La terminologia deriva da René Descartes (1596–1650) che dopo una vita avventurosa (giocatore, soldato, girovago, matematico-filosofo) ricevette un invito dalla Regina Christina di Svezia per divenire il suo tutore personale in filosofia, ma l'inverno del 1649–1650 fu particolarmente rigido e Descartes morì di polmonite a metà Febbraio.

Esempio 1.6.3. Dati $V = \{A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K\}$ e $S = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$ il prodotto $V \times S$ definisce un classico mazzo di carte da gioco.

Esempio 1.6.4. L'insieme $\mathbb{R} \times \mathbb{R}$ è l'insieme delle coppie (x, y) tali che $x, y \in \mathbb{R}$. Molti studenti sono abituati a rappresentare gli elementi di questo insieme come i punti di un piano (solitamente chiamato piano cartesiano) in cui la posizione sull'asse delle x denota il primo elemento della coppia (in Figura 1.1, x_P) e la posizione sull'asse delle y denota il secondo elemento della coppia (in Figura 1.1, y_P).

⁵Talvolta si usano le parentesi angolari $\langle a, b \rangle$ per rappresentare delle collezioni ordinate, invece delle tonde... ma noi non lo faremo.

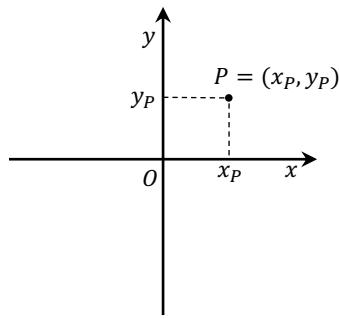


Figura 1.1: Il piano Cartesiano ($\mathbb{R} \times \mathbb{R}$) e l'elemento $P = (x_P, y_P) \in \mathbb{R} \times \mathbb{R}$.

Esempio 1.6.5. Dati $A = \{a, b, c\}$, $B = \{1, 2\}$ e $C = \{a\}$ mostriamo gli insiemi

$$A \times (B \times C) \text{ e } (A \times B) \times C.$$

L'insieme di sinistra è ottenuto calcolando prima

$$B \times C = \{(1, a), (2, a)\}$$

e poi facendo il prodotto di A per tale insieme:

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

L'insieme di destra è ottenuto calcolando prima

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

e poi facendo il prodotto di tale insieme per C :

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

È importante notare che i due insiemi sono molto simili, ma diversi: gli elementi dell'insieme di sinistra (ad esempio $(a, (1, a))$) sono coppie in cui il secondo elemento è esso stesso una coppia (in $B \times C$). Invece, gli elementi dell'insieme di destra (ad esempio $((a, 1), a)$) sono coppie in cui il primo elemento è esso stesso una coppia (in $A \times B$). Si ha pertanto che il prodotto cartesiano non è associativo, cioè

$$A \times (B \times C) \neq (A \times B) \times C$$

Nel prossimo capitolo vedremo che tali insiemi sono in biiezione.

Esercizio 1.6.6. Dati $A = \{a, b, c\}$ e $B = \{1, 2\}$ calcolare $A \times B$ e $B \times A$. (Notare che il prodotto non è commutativo, ovvero $A \times B \neq B \times A$).

Esercizio 1.6.7. Esistono due insiemi A e B tali che $A \times B = B \times A$?

Esercizio 1.6.8. Esiste un insieme A tale che $A \times A = A$?

Esercizio 1.6.9. Calcolare $A \times \emptyset$.

Puzzle 1.6.10. Dire quali altre coppie comprende l'insieme

$$\{(2, 1), (4, 2), (6, 3), (8, 4), (10, 5), (12, 6), \dots\}$$

E se invece consideriamo l'insieme riportato sotto?

$$\{(2, 3), (4, 7), (6, 3), (8, 4), (10, 5), (12, 6), \dots\}$$

1.7 Cardinalità

Molti dei problemi che vedrete durante il corso dei vostri studi (e che dovete affrontare nella vostra vita di Informatici) possono, in linea del tutto generale, essere ridotti alla seguente domanda:

quanti elementi ha un dato insieme?

Questa quantità è chiamata “cardinalità”.

Definizione 1.7.1 (Cardinalità). *Sia A un insieme contenente esattamente n elementi distinti tra loro (con $n \in \mathbb{N}$). In questo caso diciamo che A è un insieme FINITO e che A ha CARDINALITÀ n . In questo caso si scrive $|A| = n$.*

Esempio 1.7.2.

- l’insieme delle vocali $V = \{a, e, i, o, u\}$ ha cardinalità 5, cioè $|V| = 5$.
- $|AN| = 62$ (solo 52 caratteri alfanumerici se consideriamo l’alfabeto italiano invece di quello inglese).
- $|\emptyset| = 0$.
- Ricordiamoci che ogni numero naturale $n \in \mathbb{N}$, può essere visto come l’insieme $\{0, 1, \dots, n-1\}$. È facile da vedere che, per tutti gli $n \in \mathbb{N}$, $|n| = n$.

E $|\mathbb{N}|$? Oppure $|\mathbb{R}|$? Sono esempi di insiemi *infiniti* (ovvero contenenti un numero di elementi non finito). Ci sono molti tipi diversi di infiniti (ad esempio $|\mathbb{N}| < |\mathbb{R}|$) che sono studiati nella teoria dei numeri ordinali e cardinali. Questa teoria, per quanto affascinante, sarà utile (e comprensibile) solo a pochi di voi. Per questa ragione, durante il corso studieremo solo insiemi di cardinalità finita.

Osservazione 1.7.3. *Nel resto di queste note, ogni volta che ci riferiremo alla cardinalità di un insieme, staremo assumendo implicitamente che l’insieme è finito. Pertanto, eviteremo di specificare ogni volta insieme finito.*

Esercizio 1.7.4. Determinare la cardinalità di ciascuno dei seguenti insiemi: $\{a\}$, $\{\{a\}\}$, $\{a, \{a\}\}$, $\{a, \{a\}, \{a, \{a\}\}\}$, \emptyset , $\{\emptyset\}$, $\{\emptyset, \{\emptyset\}\}$, $\{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$

Proposizione 1.7.5. Per tutti gli insiemi A, B vale che se $A \subseteq B$, allora $|A| \leq |B|$.

Dimostrazione. Assumendo che $A \subseteq B$, dobbiamo dimostrare che $|A| \leq |B|$. Diciamo che l’insieme A contiene esattamente n elementi, cioè $|A| = n$, e che l’insieme B contiene esattamente m elementi, cioè $|B| = m$. Visto che $A \subseteq B$, sappiamo che tutti gli elementi di A , sono anche in B . Quindi gli n elementi di A devono essere anche in B , quindi $m \geq n$. ■

Utilizzando il concetto di cardinalità possiamo adesso introdurre l’ultima definizione di questo capitolo che giocherà un ruolo importante nei Grafi (Capitolo 5) e nel Calcolo Combinatorio (Capitolo 6).

Definizione 1.7.6 (k -insiemi). *Sia A un insieme di cardinalità n e sia k un numero naturale. Un insieme X è detto un k -INSIEME DI A se $X \subseteq A$ e $|X| = k$. Indichiamo con $\mathcal{P}_k(A)$ l’insieme di tutti i k -insiemi di A , cioè*

$$\mathcal{P}_k(A) = \{X \subseteq A \mid k = |X|\}.$$

Tale insieme è talvolta anche chiamato insieme delle parti di A di cardinalità k .

Esempio 1.7.7. Sia $A = \{a, b, c\}$. L’insieme $\mathcal{P}_1(A)$ è l’insieme $\{\{a\}, \{b\}, \{c\}\}$, l’insieme $\mathcal{P}_2(A)$ è $\{\{a, b\}, \{b, c\}, \{a, c\}\}$ e l’insieme $\mathcal{P}_3(A)$ è $\{\{a, b, c\}\}$.

In Tabella 1.5, illustriamo alcune importanti leggi che permettono di calcolare la cardinalità di insiemi composti attraverso le operazioni viste sino ad ora. Le prime due leggi sono state discusse nell’Esempio 1.7.2, le altre verranno dimostrate nel Capitolo 6. In particolare, nel membro destro della quinta uguaglianza si utilizza il simbolo $\binom{n}{k}$ che denota il coefficiente binomiale, un’operazione che verrà introdotta nel Capitolo 6.

(1)	$ \emptyset $	=	0
(2)	$ n $	=	n
(3)	$ A \setminus B $	=	$ A - A \cap B $
(4)	$ A \cup B $	=	$ A + B - A \cap B $
(5)	$ A \times B $	=	$ A \cdot B $
(6)	$ \mathcal{P}(A) $	=	$2^{ A }$
(7)	$ \mathcal{P}_k(A) $	=	$\binom{ A }{k}$

Tabella 1.5: Cardinalità di alcuni insiemi notevoli

1.8 Esercizi

Esercizio 1.8.1. Elencare gli elementi che appartengono ai seguenti insiemi:

1. $\{x \in \mathbb{R} \mid x^2 = 1\}$
2. $\{x \in \mathbb{N} \mid x^2 = 2\}$
3. $\{x \in \mathbb{N} \mid x < 100 \text{ e } x \text{ è il quadrato di un qualche numero intero}\}$

Esercizio 1.8.2. Per ognuna delle seguenti coppie di insiemi, dire se $A = B$ o se $A \neq B$:

1. $A = \{1, 3, 3, 3, 5, 5, 5, 5, 5\}$ e $B = \{5, 3, 1\}$
2. $A = \{\}$ e $B = \{\emptyset\}$
3. $A = \{1, \{1\}\}$ e $B = \mathcal{P}(\{1\})$

Esercizio 1.8.3. Dire quali delle seguenti affermazioni sono vere e quali false:

1. $a \in \{a\}$
2. $\{a\} \subseteq \{a\}$
3. $\{a\} \in \{a\}$
4. $\{a\} \in \{\{a\}\}$
5. $\emptyset \subseteq \{a\}$
6. $\emptyset \in \{a\}$

Esercizio 1.8.4. Determinare l'insieme delle parti di ciascuno dei seguenti insiemi:

1. $\{a\}$
2. $\{a, b\}$
3. $\{\emptyset, \{\emptyset\}\}$
4. $\{a, b, c, d\}$

Esercizio 1.8.5. Dati $A = \{a, b, c\}$ e $B = \{a, d\}$ calcolare i seguenti insiemi:

1. $A \times B$
2. $B \times A$
3. $A \cup B$
4. $A \cap B$
5. $A \setminus B$

$$6. B \setminus A$$

Esercizio 1.8.6. Trovare un insieme A tale che $A \cup \mathbb{N}^+ = \mathbb{N}$, $\mathbb{N}^p \cap A = \{0\}$ e $A \cap \mathbb{P} = \{11\}$. Quanti insiemi esistono che soddisfano queste condizioni?

Esercizio 1.8.7. Tre categorie di utenti sono soci di un centro sportivo. Sapreste dire quanti sono complessivamente i soci se vi dico che:

- 44 giocano a Tennis (T)
- 26 Nuotano (N)
- 31 giocano a Golf (G)

La risposta non è 101! Non avete abbastanza elementi per rispondere perché le tre categorie non sono necessariamente disgiunte. Fatemi aggiungere che:

- 12 si dedicano al Tennis e Nuotano
- 5 al Tennis e al Golf
- 6 al Nuoto e al Golf
- 4 a tutti e tre gli sport

Quanti sono in tutto i soci? (risolvere usando diagrammi di Eulero-Venn)

Esercizio 1.8.8. Dimostrare (per sostituzione) che per ogni A, B, C :

1. $A \setminus \overline{A} = A$
2. $(A \setminus B) \setminus C = (A \setminus C) \setminus B$
3. $\overline{A \setminus B} = \overline{A} \cup B$
4. $\overline{A \cup (\overline{B} \cap C)} = (\overline{C} \cap \overline{A}) \cup (\overline{A} \cap B)$
5. $(A \cup \overline{B}) \cap C = (C \cap A) \cup (C \setminus B)$

Esercizio 1.8.9. È vero che per tutti gli insiemi A, B, C, D , vale che $(A \cap D) \setminus (B \cup C) = \emptyset$? In caso affermativo fornire una dimostrazione; in caso negativo un controesempio.

Esercizio 1.8.10. Dimostrare il Teorema 1.4.8 usando i diagrammi di Eulero-Venn.

Esercizio 1.8.11. Elencare tutte le possibili partizioni dell'insieme $A = \{a, b, c\}$.

CAPITOLO 2

Relazioni

In questo capitolo esaminiamo la nozione di *relazione*. Mostreremo svariate operazioni su relazioni e le leggi algebriche che le regolano. Considereremo poi le seguenti quattro proprietà

TOTALE	SURGETTIVA
UNIVALENTE	INIETTIVA

che sono necessarie per definire i concetti di *funzione* e *biezione*. Alla fine del capitolo, utilizzeremo questi concetti per introdurre alcune strutture dati fondamentali in Informatica: le *n-uple*, le sequenze di lunghezza fissata (array) e le sequenze di lunghezza arbitraria (stringhe).

2.1 Nozioni di Base

Definizione 2.1.1 (Relazione). *Una RELAZIONE R tra l'insieme A e l'insieme B è un sottoinsieme del prodotto cartesiano $A \times B$, quindi $R \subseteq A \times B$. Indicheremo l'insieme di tutte le relazioni tra A e B con la notazione $\text{Rel}(A, B)$. Per indicare che R è una relazione tra A e B , cioè $R \in \text{Rel}(A, B)$, scriveremo spesso $R: A \leftrightarrow B$*

Si noti che dalla definizione segue che $\text{Rel}(A, B) = \mathcal{P}(A \times B)$.

Data una relazione $R: A \leftrightarrow B$ e due elementi $a \in A$ e $b \in B$, se $(a, b) \in R$ diremo che a è in relazione R con b .

Esempio 2.1.2.

- Se $A = \{x, y\}$ e $B = \{a, b, c\}$, il sottoinsieme

$$\{(x, a), (x, c)\} \subseteq A \times B \quad (2.1)$$

è una relazione in $\text{Rel}(A, B)$.

- Per tutti gli insiemi A e B , il prodotto cartesiano $A \times B$ è una relazione in $\text{Rel}(A, B)$. Questa viene chiamata RELAZIONE COMPLETA.
- Per tutti gli insiemi A e B , $\emptyset \subseteq A \times B$ è una relazione in $\text{Rel}(A, B)$. Questa viene chiamata RELAZIONE VUOTA, e viene denotata con $\emptyset_{A,B}$.

In una relazione $R: A \leftrightarrow B$, A è detto INSIEME DI PARTENZA e B è detto INSIEME DI ARRIVO. Talvolta, l'insieme di partenza e l'insieme di arrivo possono coincidere: durante il corso vedremo molte relazioni di questo tipo, che sono anche argomento del Capitolo 4. Illustriamo come esempi prima le relazioni di parentela tra esseri umani e poi la relazione identità su un qualsiasi insieme A .

Esempio 2.1.3. Sia U l'insieme di tutti gli esseri umani, e consideriamo le seguenti relazioni di parentela:

- $\text{Madre} = \{(x, y) \in U \times U \mid x \text{ è madre di } y\}$,
- $\text{Padre} = \{(x, y) \in U \times U \mid x \text{ è padre di } y\}$,
- $\text{Figlia} = \{(x, y) \in U \times U \mid x \text{ è figlia di } y\}$,

2. Relazioni

- $\text{Figlio} = \{(x, y) \in U \times U \mid x \text{ è figlio di } y\}$.

A partire da queste quattro relazioni costruiremo nel seguito tutte le comuni relazioni di parentela, tipo sorella, fratello, zia, ...

Osservazione 2.1.4. Gli autori di queste note credono nella libertà di ogni essere umano di determinare e definire la propria sessualità, i propri affetti e le proprie relazioni familiari. Rigettano quindi ogni forma di discriminazione sessista ed etero-normativa. Tuttavia in queste note le relazioni familiari considerate saranno spesso riferite a quelle della famiglia così detta tradizionale. Tali relazioni infatti sono degli esempi molto convenienti delle varie proprietà che desideriamo illustrare.

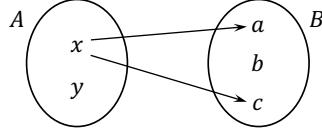
Esempio 2.1.5. Per tutti gli insiemi A ,

$$\{(x, x) \mid x \in A\} \subseteq A \times A$$

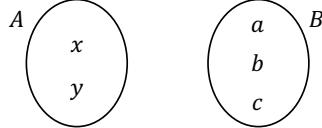
è una relazione. Questa viene chiamata RELAZIONE IDENTITÀ su A , e viene denotata $Id_A: A \leftrightarrow A$.

Come gli insiemi, anche le relazioni possono essere comodamente visualizzate attraverso dei diagrammi: data una relazione $R: A \leftrightarrow B$ questa viene visualizzata disegnando l'insieme A sulla sinistra, l'insieme B sulla destra e inserendo una freccia dall'elemento a all'elemento b per ogni coppia $(a, b) \in R$.

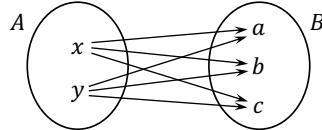
Esempio 2.1.6. Prendiamo $A = \{x, y\}$ e $B = \{a, b, c\}$ come nell'Esempio 2.1.2. La relazione in (2.1) viene disegnata come



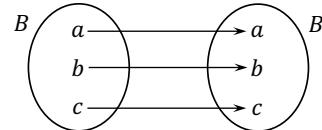
mentre la relazione vuota $\emptyset: A \leftrightarrow B$ come



e la relazione completa $A \times B: A \leftrightarrow B$ come



Invece, la relazione identità su B , $Id_B: B \leftrightarrow B$ è disegnata come



Esercizio 2.1.7. Sia $A = \{x, y, z\}$ e $B = \{x, y, z, u\}$. Disegnare la relazione $\{(x, x), (y, y), (z, z)\}: A \leftrightarrow B$.

Quando le relazioni sono molto grandi, o infinite, diventa complicato rappresentarle attraverso diagrammi. Per esempio, per disegnare la relazione *Madre* su tutti gli esseri umani (Esempio 2.1.3) non basterebbero tutte le pagine di queste dispense. Quando però nelle relazioni c'è una sorta di regola, come per esempio in molte relazioni matematiche si possono usare i punti (...) per sottintendere la regola.

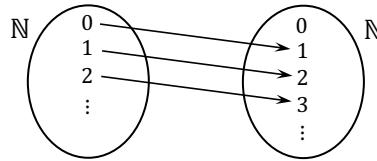
Esempio 2.1.8. Si consideri la relazione *successore*

$$\text{Succ} = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid y = x + 1\}: \mathbb{N} \leftrightarrow \mathbb{N}$$

dove + denota l'usuale operazione di somma tra numeri. Questa relazione si può anche definire per enumerazione come

$$\text{Succ} = \{(0, 1), (1, 2), (2, 3), \dots\}: \mathbb{N} \leftrightarrow \mathbb{N}$$

e rappresentare con il seguente diagramma.



Molto spesso l'insieme di partenza è il prodotto cartesiano di due insiemi, cioè $A = A_1 \times A_2$ per qualche insieme A_1, A_2 .

Esempio 2.1.9. Si consideri la relazione

$$\text{Plus} = \{((x, y), z) \mid z = x + y\}: \mathbb{N} \times \mathbb{N} \leftrightarrow \mathbb{N}$$

dove + denota l'usuale operazione di somma tra numeri. In questo caso l'insieme di partenza è $\mathbb{N} \times \mathbb{N}$, mentre l'insieme di arrivo è \mathbb{N} .

Questa relazione si può anche definire per enumerazione come

$$\begin{aligned} \text{Plus} = & \{((0, 0), 0), ((1, 0), 1), ((2, 0), 2), \dots \\ & ((0, 1), 1), ((1, 1), 2), ((2, 1), 3), \dots \\ & \dots \\ & \} \end{aligned}$$

2.2 Operazioni su relazioni

Come gli insiemi, anche le relazioni possono essere combinate in vari modi per ottenere nuove relazioni. In questa sezione illustriamo le principali operazioni su relazioni.

Operazioni insiemistiche su relazioni

Dal momento che ogni relazione è essa stessa un insieme, le relazioni possono essere combinate con tutti gli operatori insiemistici.

Esempio 2.2.1. Si ricordino le relazioni *Madre*: $U \leftrightarrow U$ e *Padre*: $U \leftrightarrow U$ dell'Esempio 2.1.3. La relazione *Genitore*: $U \leftrightarrow U$ può essere definita come

$$\text{Genitore} = \text{Madre} \cup \text{Padre}. \quad (2.2)$$

Infatti

$$\begin{aligned} & \text{Madre} \cup \text{Padre} \\ & = \{(x, y) \in U \times U \mid x \text{ è madre di } y\} \cup \{(x, y) \in U \times U \mid x \text{ è padre di } y\} \\ & = \{(x, y) \in U \times U \mid x \text{ è madre di } y \vee x \text{ è padre di } y\} \\ & = \{(x, y) \in U \times U \mid x \text{ è madre di } y \text{ oppure } x \text{ è padre di } y\} \\ & = \{(x, y) \in U \times U \mid x \text{ è genitore di } y\} \end{aligned}$$

Esercizio 2.2.2. Si definisca attraverso unione, la relazione

$$Figli* = \{(x, y) \in U \times U \mid x \text{ è figlio o figlia di } y\}.$$

Quando si compongono relazioni attraverso le operazioni insiemistiche si deve però fare sempre attenzione agli insiemi di partenza e di arrivo. Per evitare ogni tipo di complicazioni, nel nostro corso consideriamo queste operazioni solo su relazioni che hanno uno stesso insieme di partenza (A) e uno stesso insieme di arrivo (B).

Definizione 2.2.3 (operazioni insiemistiche su relazioni). *Siano date due relazioni $R: A \leftrightarrow B$ e $S: A \leftrightarrow B$.*

- La relazione $R \cup S: A \leftrightarrow B$ è detta UNIONE di R e S ;
- La relazione $R \cap S: A \leftrightarrow B$ è detta INTERSEZIONE di R e S ;
- La relazione $R \setminus S: A \leftrightarrow B$ è detta DIFFERENZA di R con S ;
- La relazione $(A \times B) \setminus R: A \leftrightarrow B$ è detta il COMPLEMENTO di R . Il complemento di una relazione $R: A \leftrightarrow B$ è denotato da \overline{R} .

Si noti che il complemento di R è definito in maniera apparentemente diversa dal complemento insiemistico: come insieme il complemento di R è $\mathcal{U} \setminus R$, mentre come relazione è $A \times B \setminus R$. L'idea è che quando si considerano relazioni tra A e B si fissa sempre come universo \mathcal{U} l'insieme $A \times B$.

Tutte le leggi nelle Tabelle 1.1, 1.2, 1.3 e 1.4 valgono chiaramente anche per le relazioni tra A e B ma, ancora una volta, si deve prendere $\mathcal{U} = A \times B$.

Composizione

Abbiamo appena visto che quando si combinano relazioni attraverso le operazioni insiemistiche è opportuno restringersi a relazioni con gli stessi insiemi di partenza e di arrivo.

Adesso consideriamo un'operazione che permette di combinare due relazioni, quando l'insieme di arrivo della prima relazione è lo stesso dell'insieme di partenza della seconda.

Definizione 2.2.4. [composizione di relazioni] *Siano $R: A \leftrightarrow B$ e $S: B \leftrightarrow C$. La COMPOSIZIONE di R con S è la relazione $R; S: A \leftrightarrow C$ così definita:*

$$R; S = \{(x, z) \in A \times C \mid \text{esiste almeno un } y \in B \text{ tale che } (x, y) \in R \text{ e } (y, z) \in S\}.$$

Esempio 2.2.5. Siano dati $A = \{x, y, z\}$, $B = \{a, b, c, d\}$ e $C = \{1, 2\}$. Consideriamo le relazioni

$$R = \{(x, a), (y, b), (z, c), (z, d)\}: A \leftrightarrow B$$

e

$$S = \{(a, 1), (a, 2), (c, 2), (d, 2)\}: B \leftrightarrow C.$$

Le due relazioni possono essere composte in quanto l'insieme di arrivo di R è uguale all'insieme di partenza di S . La loro composizione è la relazione

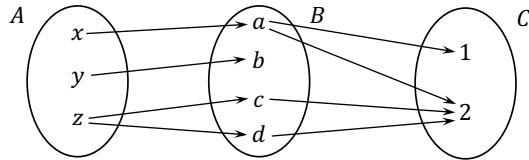
$$R; S = \{(x, 1), (x, 2), (z, 2)\}: A \leftrightarrow C.$$

Infatti $(x, 1) \in R; S$ perché per l'elemento $a \in B$ si ha che $(x, a) \in R$ e $(a, 1) \in S$. Similmente, $(x, 2) \in R; S$ perché per l'elemento $a \in B$ valgono sia $(x, a) \in R$ che $(a, 2) \in S$.

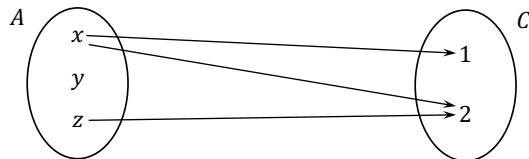
Per vedere che $(z, 2) \in R; S$ abbiamo due possibili giustificazioni: (1) per l'elemento $c \in B$ si ha che $(z, c) \in R$ e $(c, 2) \in S$; (2) per l'elemento $d \in B$ si ha che $(z, d) \in R$ e $(d, 2) \in S$.

Per vedere che nessun'altra coppia $(u, w) \in A \times C$ appartiene a $R; S$ si deve mostrare che non esiste alcun elemento $v \in B$ tale che $(u, v) \in R$ e $(v, w) \in S$. Per esempio, per vedere che $(y, 1) \notin R; S$ si può osservare che: per $a \in B$, $(y, a) \notin R$; per $b \in B$, $(b, 1) \notin S$; per $c \in B$, $(y, c) \notin R$; e per $d \in B$, $(y, d) \notin R$.

L'operazione di composizione di relazioni può sembrare un po' complicata, ma in realtà risulta estremamente naturale quando si visualizza attraverso diagrammi. Per visualizzare la composizione di $R: A \leftrightarrow B$ e $S: B \leftrightarrow C$, prima di tutto si disegnano gli insiemi A (a sinistra), B (al centro) e C (a destra), nonché le relazioni R e S , utilizzando la solita modalità delle frecce. Per esempio il seguente diagramma



visualizza le relazioni R e S dell'Esempio 2.2.5. Poi per ottenere il diagramma della relazione composta è sufficiente seguire le frecce da sinistra a destra e vedere per ogni elemento a di A e ogni elemento c di C se c'è un "percorso" da a a c . In tal caso, si aggiunge una freccia da a a c . Una volta fatto ciò per tutte le coppie $(a, c) \in A \times C$ si cancella l'insieme B dal centro e tutte le frecce ad esso collegate. Il seguente diagramma visualizza la relazione $R; S$ dell'Esempio 2.2.5.



Esercizio 2.2.6. Si consideri la relazione $\text{Succ}: \mathbb{N} \leftrightarrow \mathbb{N}$. Si disegni il diagramma della relazione $\text{Succ}; \text{Succ}: \mathbb{N} \leftrightarrow \mathbb{N}$. Definire prima per enumerazione e poi per proprietà tale relazione.

Abbiamo finora mostrato degli esempi di composizione su entità piuttosto astratte. In realtà l'operazione di composizione di relazioni ci permette di formalizzare concetti della vita quotidiana.

Esempio 2.2.7. Si ricordi la relazione $\text{Padre}: U \leftrightarrow U$ dell'Esempio 2.1.3 e la relazione $\text{Genitore}: U \leftrightarrow U$ dell'Esempio 2.2.1. La relazione $\text{Nonno}: U \leftrightarrow U$ può essere definita come

$$\text{Nonno} = \text{Padre}; \text{Genitore}.$$

Esercizio 2.2.8. Definire per proprietà la relazione $\text{Genitore}; \text{Genitore}$.

Esercizio 2.2.9. Definire attraverso composizione la relazione $\text{Nonna} = \{(x, y) \in U \times U \mid x \text{ è nonna di } y\}$.

Esercizio 2.2.10. Definire utilizzando solamente la composizione la relazione $\text{Nonn} * \text{Patern}* = \{(x, y) \in U \times U \mid x \text{ è nonno paterno o nonna paterna di } y\}$.

I quantificatori

Nella definizione dell'operazione di composizione (Definizione 2.2.4) abbiamo utilizzato l'espressione italiana *esiste almeno*:

$$\{(x, z) \in A \times C \mid \text{esiste almeno un } y \in B \text{ tale che } (x, y) \in R \text{ e } (y, z) \in S\}$$

Questa espressione riveste un ruolo speciale nel linguaggio matematico e viene denotata dal simbolo \exists , chiamato QUANTIFICATORE ESISTENZIALE. La formula

$$(\exists a \in A . P(a))$$

si legge

2. Relazioni

esiste almeno un elemento di A tale che la proprietà P è vera.

Pertanto la composizione di relazioni può essere scritta in formule come

$$R; S = \{(x, z) \in A \times C \mid (\exists y \in B . (x, y) \in R \wedge (y, z) \in S)\}.$$

Un altro simbolo di fondamentale importanza è \forall , chiamato QUANTIFICATORE UNIVERSALE, che si legge in italiano *per ogni*. La formula

$$(\forall a \in A . P(a))$$

si legge

per tutti gli elementi di A vale che la proprietà P è vera.

Per dimostrare che la formula $(\exists a \in A . P(a))$ è vera è sufficiente esibire un elemento $a \in A$ per cui vale la proprietà P . Dimostrare che la formula $(\forall a \in A . P(a))$ è vera è molto più laborioso in generale: infatti si deve dimostrare che P vale per tutti gli $a \in A$. È importante notare che in questo caso, quando A è vuoto, non c'è niente da dimostrare: quindi la formula $(\forall a \in A . P(a))$ è banalmente vera.

I quantificatori \exists e \forall saranno studiati in profondità nel Capitolo 9 dove presenteremo la logica del primo ordine. Fino ad allora, utilizzeremo di tanto in tanto questi simboli con il significato che abbiamo appena descritto.

Relazione opposta

Definizione 2.2.11 (relazione opposta). *Sia $R: A \leftrightarrow B$ una relazione. La RELAZIONE OPPOSTA di R è la relazione $R^{op}: B \leftrightarrow A$ definita come*

$$R^{op} = \{(y, x) \in B \times A \mid (x, y) \in R\}.$$

Esempio 2.2.12. Si considerino le relazioni R e S dell'Esempio 2.2.5. Si ha che

$$R^{op} = \{(a, x), (b, y), (c, z), (d, z)\}: B \leftrightarrow A$$

e

$$S^{op} = \{(1, a), (2, a), (2, c), (2, d)\}: C \leftrightarrow B.$$

È importante notare che R^{op} non può essere composta con S^{op} perché l'insieme di arrivo di R^{op} è A , mentre quello di partenza di S^{op} è C . Comunque si possono comporre le relazioni nel senso inverso, cioè come $S^{op}; R^{op}$ perché sia l'insieme di arrivo di S^{op} che quello di partenza di R^{op} sono l'insieme B .

Esercizio 2.2.13. Si considerino le relazioni R , S ed $R; S$ nell'Esempio 2.2.5 e le relazioni R^{op} e S^{op} nell'Esempio 2.2.12. È vera la seguente uguaglianza?

$$(R; S)^{op} = S^{op}; R^{op}$$

Esempio 2.2.14. Si consideri la relazione Genitore: $U \leftrightarrow U$ introdotta nell'Esempio 2.2.1. Si ha che

$$\text{Genitore}^{op} = \text{Figlio*}$$

dove Figlio* è la relazione dell'Esempio 2.2.2. Infatti si ha che

$$\begin{aligned} \text{Genitore}^{op} &= \{(y, x) \in U \times U \mid (x, y) \in \text{Genitore}\} \\ &= \{(y, x) \in U \times U \mid x \text{ è madre o padre di } y\} \\ &= \{(y, x) \in U \times U \mid y \text{ è figlio o figlia di } x\} \\ &= \{(x, y) \in U \times U \mid x \text{ è figlio o figlia di } y\} \\ &= \text{Figlio*} \end{aligned}$$

Esercizio 2.2.15. È vero che

$$Figlio^{\text{op}} = \text{Genitore}?$$

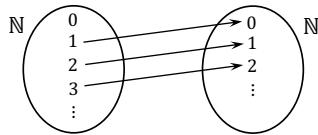
Esempio 2.2.16. Si consideri la relazione

$$\text{Succ}: \mathbb{N} \leftrightarrow \mathbb{N} = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid y = x + 1\}$$

introdotta nell'Esempio 2.1.8. Si ha che

$$\begin{aligned} \text{Succ}^{\text{op}} &= \{(y, x) \in \mathbb{N} \times \mathbb{N} \mid (x, y) \in \text{Succ}\} \\ &= \{(y, x) \in \mathbb{N} \times \mathbb{N} \mid y = x + 1\} \\ &= \{(y, x) \in \mathbb{N} \times \mathbb{N} \mid y - 1 = x\} \\ &= \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid x - 1 = y\} \\ &= \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid y = x - 1\} \end{aligned}$$

Chiamiamo questa relazione *Pred* che sta per “predecessore”.



È importante osservare come si può ottenere il diagramma per Succ^{op} a partire da quello di Succ : dato il diagramma di una qualsiasi relazione R , il diagramma di R^{op} è ottenuto come l'immagine allo specchio (cioè girandolo di 180 gradi) e invertendo la direzione delle frecce.

Mostriamo adesso un po' di esempi ben noti dalla matematica.

Esempio 2.2.17.

$$\text{Double}: \mathbb{N} \leftrightarrow \mathbb{N} = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid y = 2 \times x\}$$

$$\text{Double}^{\text{op}} = \text{Half}: \mathbb{N} \leftrightarrow \mathbb{N} = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid y = x/2\}$$

Esercizio 2.2.18. Disegnare il diagramma delle relazioni $\text{Double}: \mathbb{N} \leftrightarrow \mathbb{N}$ e $\text{Double}^{\text{op}} = \text{Half}: \mathbb{N} \leftrightarrow \mathbb{N}$

Esempio 2.2.19.

$$\text{Square}: \mathbb{N} \leftrightarrow \mathbb{N} = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid y = x^2\}$$

$$\text{Square}^{\text{op}} = \text{Sqrt}: \mathbb{N} \leftrightarrow \mathbb{N} = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid y = \sqrt{x}\}$$

Esercizio 2.2.20. Disegnare il diagramma delle relazioni $\text{Square}: \mathbb{N} \leftrightarrow \mathbb{N}$ e $\text{Square}^{\text{op}} = \text{Sqrt}: \mathbb{N} \leftrightarrow \mathbb{N}$

Esempio 2.2.21.

$$\text{Exp}: \mathbb{N} \leftrightarrow \mathbb{N} = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid y = 2^x\}$$

$$\text{Exp}^{\text{op}} = \text{Log}: \mathbb{N} \leftrightarrow \mathbb{N} = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid y = \log_2(x)\}$$

Esercizio 2.2.22. Disegnare il diagramma delle relazioni $\text{Exp}: \mathbb{N} \leftrightarrow \mathbb{N}$ e $\text{Exp}^{\text{op}} = \text{Log}: \mathbb{N} \leftrightarrow \mathbb{N}$

Esercizio 2.2.23. Definire attraverso le relazioni e le operazioni viste finora, la relazione *Fratello* = $\{(x, y) \in U \times U \mid x \text{ è fratello di } y\}$. Per “fratello” si intende il figlio di almeno uno dei genitori. Per esempio, due uomini sono considerati fratelli, anche se hanno la stessa madre ma non lo stesso padre.

Esercizio 2.2.24. Definire attraverso le relazioni e le operazioni viste finora, la relazione *Sorella* = $\{(x, y) \in U \times U \mid x \text{ è sorella di } y\}$. Anche qui per “sorella” si intende la figlia di almeno uno dei genitori.

Esercizio 2.2.25. Definire attraverso le relazioni e le operazioni viste finora, le relazioni *Zia* = $\{(x, y) \in U \times U \mid x \text{ è zia di } y\}$ e *Zio* = $\{(x, y) \in U \times U \mid x \text{ è zio di } y\}$.

Concludiamo mostrando un'applicazione immediata delle operazioni su relazioni all'Informatica.

2. Relazioni

Esempio 2.2.26. Uno tra i più noti linguaggi di interrogazione di database, SQL (Structured Query Language), è basato sulle operazioni tra relazioni: un database è pensato come una collezione di relazioni e le interrogazioni (queries) al database come operazioni su tali relazioni. Questi argomenti verranno trattati in modo esauritivo nel corso di Basi di Dati; in questo corso ci limitiamo a mostrare un esempio giocattolo per dare un'intuizione più concreta.

Immaginiamoci una piccola Università con pochi studenti, corsi e professori.

- $\text{Studenti} = \{\text{Lucia, Mario, Franca, Luisa, Gino, Marco, Marco, Gaia, Carlo}\};$
- $\text{Corsi} = \{\text{PA, LAB, FDI, ANA}\};$
- $\text{Professori} = \{\text{Paperino, Pippo, Pluto, Paperone}\}.$

Il database dell'Università consiste delle seguenti due tabelle.

Segue

Studenti	Corsi
Lucia	FDI
Lucia	PA
Mario	ANA
Franca	FDI
Franca	LAB
Luisa	PA
Luisa	ANA
Luisa	LAB
Gino	LAB
Marco	FDI
Marco	LAB
Gaia	FDI
Gaia	ANA

Insegna

Professori	Corsi
Paperino	PA
Paperino	LAB
Pippo	FDI
Pluto	ANA
Paperone	FDI

La tabella di sinistra rappresenta la relazione Segue: $\text{Studenti} \leftrightarrow \text{Corsi}$: $(x, y) \in \text{Segue}$ se e solo se lo studente x segue il corso y . Ad esempio Lucia segue FDI e PA. In modo analogo la tabella di destra rappresenta la relazione Insegna: $\text{Professori} \leftrightarrow \text{Corsi}$.

Una interrogazione tipica per questo database potrebbe essere:

Quali professori ha ciascun studente?

La risposta a tale interrogazione è data calcolando la relazione Segue; $\text{Insegna}^{\text{op}}$

Segue; Insegna^{op}

Studenti	Professori
Lucia	Pippo
Lucia	Paperone
Lucia	Paperino
Mario	Pluto
Franca	Pippo
Franca	Paperone
Franca	Paperino
Luisa	Paperino
Luisa	Pluto
Gino	Paperino
Marco	Pippo
Marco	Paperone
Marco	Paperino
Gaia	Pippo
Gaia	Pluto
Gaia	Paperone

2.3 Leggi

Come per gli insiemi, anche per le relazioni esistono delle utili leggi che regolano il comportamento delle varie operazioni. Abbiamo già anticipato che per le operazioni insiemistiche valgono le stesse leggi studiate nella Sezione 1.4, con l'accorgimento di prendere, per una relazione $R: A \leftrightarrow B$, l'insieme $\mathcal{U} = A \times B$ come insieme universo.

Teorema 2.3.1. *Per tutti gli insiemi A, B e relazioni $R, S, T: A \leftrightarrow B$ valgono le uguaglianze nelle Tabelle 2.1, 2.2 e 2.3.*

associatività	$R \cup (S \cup T) = (R \cup S) \cup T$	$R \cap (S \cap T) = (R \cap S) \cap T$
unità	$R \cup \emptyset = R$	$R \cap (A \times B) = R$
commutatività	$R \cup S = S \cup R$	$R \cap S = S \cap R$
idempotenza	$R \cup R = R$	$R \cap R = R$
assorbimento	$R \cup (A \times B) = (A \times B)$	$R \cap \emptyset = \emptyset$

Tabella 2.1: Leggi per le operazioni insiemistiche

distributività di \cup su \cap	$R \cup (S \cap T) = (R \cup S) \cap (R \cup T)$
distributività di \cap su \cup	$R \cap (S \cup T) = (R \cap S) \cup (R \cap T)$
assorbimento di \cup su \cap	$R \cup (R \cap S) = R$
assorbimento di \cap su \cup	$R \cap (R \cup S) = R$
complemento per \cup	$R \cup \bar{R} = (A \times B)$
complemento per \cap	$R \cap \bar{R} = \emptyset$

Tabella 2.2: Leggi per le operazioni insiemistiche (2)

$$\boxed{\text{differenza } R \setminus S = R \cap \bar{S}}$$

Tabella 2.3: Leggi per le operazioni insiemistiche (3)

La validità del Teorema 2.3.1 è garantita dai risultati della Sezione 1.4, che valgono per tutti gli insiemi e quindi, in particolare, anche per tutte le relazioni.

Il seguente teorema illustra invece alcune utili leggi per la composizione e la relazione opposta.

Teorema 2.3.2. *Per tutti gli insiemi A, B, C, D e relazioni $R: A \leftrightarrow B$, $S: B \leftrightarrow C$ e $T: C \leftrightarrow D$ valgono le uguaglianze nelle Tabelle 2.4 e 2.5.*

associatività	$R; (S; T) = (R; S); T$
unità	$Id_A; R = R = R; Id_B$
assorbimento	$R; \emptyset_{B,C} = \emptyset_{A,C} = \emptyset_{A,B}; S$

Tabella 2.4: Leggi per composizione di relazioni

La notazione grafica che abbiamo introdotto per le relazioni può fornire un'intuizione chiara riguardo alla validità di tali leggi. In queste note forniamo una dimostrazione discorsiva per l'associatività della composizione e lasciamo le dimostrazioni delle altre leggi come esercizio.

Dimostrazione di $R; (S; T) = (R; S); T$. Anziché dimostrare l'uguaglianza direttamente possiamo dimostrare separatamente le inclusioni

$$R; (S; T) \subseteq (R; S); T \quad \text{e} \quad R; (S; T) \supseteq (R; S); T$$

convoluzione	$(R^{op})^{op} = R$
op-id	$Id_A^{op} = Id_A$
op-compl	$(A \times B)^{op} = B \times A$
op-vuoto	$\emptyset_{A,B}^{op} = \emptyset_{B,A}$

Tabella 2.5: Leggi per relazione opposta

e poi concludere l'uguaglianza utilizzando l'antisimmetria di \subseteq (Proposizione 1.4.13).

- $R; (S; T) \subseteq (R; S); T$. Per dimostrare questa inclusione, dobbiamo dimostrare che preso un generico elemento di $R; (S; T)$, tale elemento appartiene anche a $(R; S); T$. Sia $(a, d) \in R; (S; T)$: per definizione della composizione (\cdot) esiste almeno un $b \in B$ tale che (1) $(a, b) \in R$ e (2) $(b, d) \in S; T$. Da (2) e dalla definizione di composizione si ha che esiste almeno un $c \in C$ tale che (2.1) $(b, c) \in S$ e (2.2) $(c, d) \in T$. Da (1) e (2.1) segue che (3) $(a, c) \in R; S$. Da (3) e (2.2) segue che $(a, d) \in (R; S); T$.
- $R; (S; T) \supseteq (R; S); T$. Per dimostrare questa inclusione, dobbiamo dimostrare che preso un generico elemento di $(R; S); T$, tale elemento appartiene anche a $R; (S; T)$. La dimostrazione è del tutto analoga a quella dell'inclusione precedente.

■

Esercizio 2.3.3. Dimostrare in modo discorsivo le leggi nelle Tabelle 2.4 e 2.5.

L'ultimo teorema illustra le leggi che regolano le interazioni fra le operazioni insiemistiche, la composizione e la relazione opposta.

Teorema 2.3.4. Per tutti gli insiemi A, B, C, D e relazioni $R: A \leftrightarrow B$, $S, T: B \leftrightarrow C$ e $U: C \leftrightarrow D$ valgono le uguaglianze nelle Tabelle 2.6.

distributività di ; su \cup (sinistra)	$R; (S \cup T) = (R; S) \cup (R; T)$
distributività di ; su \cup (destra)	$(S \cup T); U = (S; U) \cup (T; U)$
distributività di \cdot^{op} su ;	$(R; S)^{op} = S^{op}; R^{op}$
distributività di \cdot^{op} su \cup	$(S \cup T)^{op} = S^{op} \cup T^{op}$
distributività di \cdot^{op} su \cap	$(S \cap T)^{op} = S^{op} \cap T^{op}$
distributività di \cdot^{op} su $\bar{\cdot}$	$(\bar{R})^{op} = \overline{(R^{op})}$

Tabella 2.6: Leggi di distributività

Ancora una volta, forniamo la dimostrazione di una sola delle leggi e lasciamo le altre come esercizio.

Dimostrazione di $(R; S)^{op} = S^{op}; R^{op}$. Anziché dimostrare l'uguaglianza direttamente possiamo dimostrare separatamente le inclusioni

$$(R; S)^{op} \subseteq S^{op}; R^{op} \quad \text{e} \quad (R; S)^{op} \supseteq S^{op}; R^{op}$$

e poi concludere l'uguaglianza utilizzando l'antisimmetria di \subseteq (Proposizione 1.4.13).

- $(R; S)^{op} \subseteq S^{op}; R^{op}$. Per dimostrare questa inclusione, dobbiamo dimostrare che preso un generico elemento di $(R; S)^{op}$, tale elemento appartiene anche a $S^{op}; R^{op}$. Sia $(c, a) \in (R; S)^{op}$: per definizione della relazione opposta si ha che $(a, c) \in R; S$. Quindi, Per definizione di composizione, esiste almeno un $b \in B$ tale che (1) $(a, b) \in R$ e (2) $(b, c) \in S$.

Utilizzando la definizione di relazione opposta, da (1) si deduce che $(b, a) \in R^{op}$ e, da (2) che $(c, b) \in S^{op}$. Per definizione di composizione, $(c, a) \in S^{op}; R^{op}$.

- $(R; S)^{op} \supseteq S^{op}; R^{op}$. Per dimostrare questa inclusione, dobbiamo dimostrare che preso un generico elemento di $S^{op}; R^{op}$, tale elemento appartiene anche a $(R; S)^{op}$. Sia $(c, a) \in S^{op}; R^{op}$. Per definizione di composizione si ha che esiste almeno un $b \in B$ tale che (1) $(c, b) \in S^{op}$ e (2) $(b, a) \in R^{op}$. Utilizzando la definizione di relazione opposta da (1) si deduce che $(b, c) \in S$ e da (2) che $(a, b) \in R$.

Pertanto, per definizione di composizione, $(a, c) \in R; S$ e quindi $(c, a) \in (R; S)^{op}$. ■

Esercizio 2.3.5. Dimostrare in modo discorsivo le leggi nelle Tabella 2.6.

Esercizio 2.3.6. Dimostrare per sostituzione che

$$\begin{aligned} \text{Genitore;} \text{Genitore} = \\ (\text{Madre;} \text{Madre}) \cup (\text{Madre;} \text{Padre}) \cup (\text{Padre;} \text{Madre}) \cup (\text{Padre;} \text{Padre}) \end{aligned}$$

Esercizio 2.3.7. Dimostrare in modo discorsivo che per tutti gli insiemi A, B, C e relazioni $R: A \leftrightarrow B$ e $S: B \leftrightarrow C$ e $T: B \leftrightarrow C$ vale che

$$\text{se } S \subseteq T, \text{ allora } R; S \subseteq R; T.$$

Esercizio 2.3.8. Considera la seguente affermazione: per tutti gli insiemi A, B, C e relazioni $R: A \leftrightarrow B$ e $S: B \leftrightarrow C$ e $T: B \leftrightarrow C$ vale che

$$\text{se } R; S \subseteq R; T \text{ allora } S \subseteq T.$$

Se è vera fornire una dimostrazione, se è falsa fornire un controesempio.

Esercizio 2.3.9. Dimostrare in modo discorsivo che per tutti gli insiemi A, B e relazioni $R: A \leftrightarrow B$ e $S: A \leftrightarrow B$ vale che

$$R \subseteq S \text{ se e solo se } R^{op} \subseteq S^{op}.$$

2.4 Proprietà di relazioni

In questa sezione illustriamo quattro fondamentali proprietà di relazioni. Queste proprietà appaiono ovunque, sia in Informatica che in Matematica e, nel resto del nostro corso, ci permetteranno di definire importanti concetti, quali funzioni e biezioni.

Definizione 2.4.1 (relazione totale). *Siano A e B due insiemi e R una relazione tra A e B . $R: A \leftrightarrow B$ si dice TOTALE se per tutti gli $a \in A$, esiste almeno un $b \in B$ tale che $(a, b) \in R$.*

Esempio 2.4.2. Siano dati $A = \{x, y, z\}$ e $B = \{a, b, c, d\}$. La relazione

$$R = \{(x, a), (x, b), (y, a), (z, d)\}: A \leftrightarrow B$$

è totale, mentre la relazione

$$S = \{(x, c), (x, d), (z, d)\}: A \leftrightarrow B$$

non è totale perché per l'elemento $y \in A$ non esiste alcun elemento $u \in B$ tale che $(y, u) \in S$.

Esempio 2.4.3. Per tutti gli insiemi A e B

- la relazione identità $Id_A: A \leftrightarrow A$ è totale;
- la relazione completa $A \times B: A \leftrightarrow B$ è totale.

Esempio 2.4.4. Per gli insiemi $A = \{x, y, z\}$ e $B = \{a, b, c, d\}$, la relazione vuota $\emptyset: A \leftrightarrow B$ non è totale. Per vederlo, basta osservare che per $x \in A$ non esiste alcun $u \in B$ tale che $(x, u) \in \emptyset$.

Esercizio 2.4.5. Esiste un insieme A per il quale la relazione vuota $\emptyset: A \leftrightarrow B$ è sicuramente totale. Quale insieme è?

Esercizio 2.4.6. La relazione $\text{Succ}: \mathbb{N} \leftrightarrow \mathbb{N}$ è totale? La relazione $\text{Pred}: \mathbb{N} \leftrightarrow \mathbb{N}$? La relazione $\text{Plus}: \mathbb{N} \times \mathbb{N} \leftrightarrow \mathbb{N}$?

Esercizio 2.4.7. La relazione $\text{Madre}: U \leftrightarrow U$ è totale? La relazione $\text{Genitore}: U \leftrightarrow U$? La relazione $\text{Figlia}: U \leftrightarrow U$? La relazione $\text{Figli*}: U \leftrightarrow U$?

Definizione 2.4.8 (relazione univalente). Siano A e B due insiemi e R una relazione tra A e B . $R: A \leftrightarrow B$ si dice UNIVALENTE se per tutti gli $a \in A$, esiste al più un $b \in B$ tale che $(a, b) \in R$.

Esempio 2.4.9. Sia $A = \{x, y, z\}$ e $B = \{a, b, c, d\}$. La relazione

$$R = \{(x, a), (x, b), (y, a), (z, d)\}: A \leftrightarrow B$$

è totale ma non è univalente perché per l'elemento $x \in A$ esistono due elementi in B che sono in relazione R con x : infatti $(x, a) \in R$ e $(x, b) \in R$. Invece, la relazione

$$S = \{(x, c), (z, d)\}: A \leftrightarrow B$$

è univalente ma non è totale.

Esempio 2.4.10. Per tutti gli insiemi A e B

- la relazione identità $\text{Id}_A: A \leftrightarrow A$ è univalente;
- la relazione vuota $\emptyset: A \leftrightarrow B$ è univalente.

Esempio 2.4.11. Per gli insiemi $A = \{x, y, z\}$ e $B = \{a, b, c, d\}$, la relazione completa $A \times B: A \leftrightarrow B$ non è univalente. Per vederlo, basta osservare che per $x \in A$ si ha sia che $(x, a) \in A \times B$ che $(x, b) \in A \times B$.

Esercizio 2.4.12. Esiste un insieme A per il quale la relazione completa $A \times B: A \leftrightarrow B$ è sicuramente univalente. Quale insieme è?

Esercizio 2.4.13. La relazione $\text{Succ}: \mathbb{N} \leftrightarrow \mathbb{N}$ è univalente? La relazione $\text{Pred}: \mathbb{N} \leftrightarrow \mathbb{N}$? La relazione $\text{Plus}: \mathbb{N} \times \mathbb{N} \leftrightarrow \mathbb{N}$?

Esercizio 2.4.14. La relazione $\text{Madre}: U \leftrightarrow U$ è univalente? La relazione $(\text{Madre})^{\text{op}}$?

È interessante notare che l'unica differenza nelle definizioni di totale e univalente consiste nel *esiste almeno* (totale) e *esiste al più* (univalente). Per una qualche proprietà P , “esiste almeno un elemento x tale che $P(x)$ ” significa che il numero di elementi che soddisfano P deve essere *maggior o uguale* ad 1. Invece “esiste al più un elemento x tale che $P(x)$ ” significa che il numero di elementi che soddisfano P deve essere *minore o uguale* ad 1.

È importante capire questa differenza, e cosa significa in termini di dimostrazioni: quando si desidera dimostrare che “esiste almeno un elemento x tale che $P(x)$ ” si deve esibire un tale x ; invece quando si desidera dimostrare che “esiste al più un elemento x tale che $P(x)$ ” si deve mostrare che, se ci sono due elementi x e y per cui vale sia $P(x)$ che $P(y)$, allora i due elementi sono lo stesso.

Osservazione 2.4.15. Abbiamo visto che l'espressione italiana esiste almeno viene denotata nel linguaggio matematico dal simbolo \exists . Per l'espressione esiste al più non c'è un simbolo corrispondente. La ragione è che esiste al più può essere scritta in formule utilizzando \forall e $=$. Più precisamente,

esiste al più un $a \in A$ tale che $P(a)$

si può scrivere come

$$(\forall a \in A . (\forall b \in A . \text{ se } P(a) \wedge P(b), \text{ allora } a = b)).$$

Adesso consideriamo altre due proprietà. L'unica differenza tra le due consiste ancora nell'*almeno* e nell'*al più*, ma diversamente dalle due proprietà discusse prima, riguardano l'insieme di arrivo e non quello di partenza.

Definizione 2.4.16 (relazione surgettiva). Siano A e B due insiemi e R una relazione tra A e B . $R: A \leftrightarrow B$ si dice SURGETTIVA se per tutti i $b \in B$, esiste almeno un $a \in A$ tale che $(a, b) \in R$.

Esempio 2.4.17. Sia $A = \{x, y, z\}$ e $B = \{a, b, c, d\}$. La relazione

$$R: A \leftrightarrow B = \{(x, a), (x, b), (y, c), (z, d)\}$$

è surgettiva, mentre la relazione

$$S: A \leftrightarrow B = \{(x, c), (x, b), (z, d)\}$$

non è surgettiva perché per l'elemento $a \in B$ non esiste alcun elemento $u \in A$ tale che $(u, a) \in S$.

Esempio 2.4.18. Per tutti gli insiemi A e B

- la relazione identità $Id_A: A \leftrightarrow A$ è surgettiva;
- la relazione completa $A \times B: A \leftrightarrow B$ è surgettiva.

Esempio 2.4.19. Per gli insiemi $A = \{x, y, z\}$ e $B = \{a, b, c, d\}$, la relazione vuota $\emptyset: A \leftrightarrow B$ non è surgettiva. Per vederlo, basta osservare che per $a \in B$ non esiste alcun $u \in A$ tale che $(u, a) \in \emptyset$.

Esercizio 2.4.20. Esiste un insieme B per il quale la relazione vuota $\emptyset: A \leftrightarrow B$ è surgettiva. Quale insieme è?

Esercizio 2.4.21. La relazione $Succ: \mathbb{N} \leftrightarrow \mathbb{N}$ è surgettiva? La relazione $Pred: \mathbb{N} \leftrightarrow \mathbb{N}$? La relazione $Plus: \mathbb{N} \times \mathbb{N} \leftrightarrow \mathbb{N}$?

Esercizio 2.4.22. La relazione $Madre: U \leftrightarrow U$ è surgettiva? La relazione $Genitore: U \leftrightarrow U$? La relazione $Figlia: U \leftrightarrow U$? La relazione $Figlio*: U \leftrightarrow U$?

Definizione 2.4.23 (relazione iniettiva). Siano A e B due insiemi e R una relazione tra A e B . $R: A \leftrightarrow B$ si dice INIETTIVA se per tutti $i b \in B$, esiste al più un $a \in A$ tale che $(a, b) \in R$.

Esempio 2.4.24. Sia $A = \{x, y, z\}$ e $B = \{a, b, c, d\}$. La relazione

$$R: A \leftrightarrow B = \{(x, a), (x, b), (y, c), (z, d)\}$$

è iniettiva, mentre la relazione

$$S: A \leftrightarrow B = \{(x, c), (x, b), (z, b)\}$$

non è iniettiva perché per l'elemento $b \in B$ esistono due elementi in A in relazione S con b : $(x, b) \in S$ e $(z, b) \in S$.

Esempio 2.4.25. Per tutti gli insiemi A e B

- la relazione identità $Id_A: A \leftrightarrow A$ è iniettiva;
- la relazione vuota $\emptyset: A \leftrightarrow B$ è iniettiva.

Esempio 2.4.26. Per gli insiemi $A = \{x, y, z\}$ e $B = \{a, b, c, d\}$, la relazione completa $A \times B: A \leftrightarrow B$ non è iniettiva. Per vederlo, basta osservare che per $a \in B$, si ha che $(x, a) \in A \times B$ e $(y, a) \in A \times B$.

Esercizio 2.4.27. Esiste un insieme B per il quale la relazione completa $A \times B: A \leftrightarrow B$ è iniettiva. Quale insieme è?

Esercizio 2.4.28. La relazione $Succ: \mathbb{N} \leftrightarrow \mathbb{N}$ è iniettiva? La relazione $Pred: \mathbb{N} \leftrightarrow \mathbb{N}$? La relazione $Plus: \mathbb{N} \times \mathbb{N} \leftrightarrow \mathbb{N}$?

Esercizio 2.4.29. La relazione $Madre: U \leftrightarrow U$ è iniettiva? La relazione $Genitore: U \leftrightarrow U$? La relazione $Figlia: U \leftrightarrow U$? La relazione $Figlio*: U \leftrightarrow U$?

La Tabella 2.7 fornisce un modo efficace per riassumere e, soprattutto, capire le quattro proprietà totale e univalente riguardano l'insieme di partenza, mentre surgettiva e iniettiva l'insieme di arrivo. Le proprietà totale e surgettiva richiedono l'esistenza di almeno un elemento, mentre univalente e iniettiva richiedono l'esistenza di al più un elemento.

	partenza	arrivo
almeno	TOTALE	SURGETTIVA
al più	UNIVALENTE	INIETTIVA

Tabella 2.7: Tabella riassuntiva delle quattro principali proprietà

Risultati di dualità

Confrontando le definizioni di relazione totale e surgettiva è facile osservare che queste impongono lo stesso vincolo ma la prima sull'insieme di partenza, mentre la seconda sull'insieme di arrivo. Abbiamo visto che l'operazione \cdot^{op} inverte l'insieme di partenza e di arrivo. Il seguente risultato collega le due proprietà attraverso la relazione opposta.

Proposizione 2.4.30. *Per tutti gli insiemi A, B e per tutte le relazioni $R: A \leftrightarrow B$ vale che:*

1. *$R: A \leftrightarrow B$ è totale se e solo se $R^{op}: B \leftrightarrow A$ è surgettiva.*
2. *$R: A \leftrightarrow B$ è univalente se e solo se $R^{op}: B \leftrightarrow A$ è iniettiva.*

Dimostrazione. Dimostriamo il primo punto e lasciamo il secondo come esercizio. Trattandosi di un se e solo se, possiamo dividere la dimostrazione in due parti:

- Se $R: A \leftrightarrow B$ è totale, allora $R^{op}: B \leftrightarrow A$ è surgettiva. Visto che R è totale allora

per ogni $x \in A$ esiste almeno un $y \in B$ tale che $(x, y) \in R$.

Per definizione di R^{op} questo significa che

per ogni $x \in A$ esiste almeno un $y \in B$ tale che $(y, x) \in R^{op}$.

Questo significa, per definizione, che $R^{op}: B \leftrightarrow A$ è surgettiva.

- Se $R^{op}: B \leftrightarrow A$ è surgettiva, allora $R: A \leftrightarrow B$ è totale. Visto che $R^{op}: B \leftrightarrow A$ è surgettiva allora

per ogni $x \in A$ esiste almeno un $y \in B$ tale che $(y, x) \in R^{op}$.

Per definizione di R^{op} questo significa che

per ogni $x \in A$ esiste almeno un $y \in B$ tale che $(x, y) \in R$.

Questo significa, per definizione, che $R: A \leftrightarrow B$ è totale.

■

Esercizio 2.4.31. *Dimostrare la Proposizione 2.4.30.2.*

Proposizione 2.4.32. *Per tutti gli insiemi A, B e per tutte le relazioni $R: A \leftrightarrow B$ vale che:*

1. *$R: A \leftrightarrow B$ è surgettiva se e solo se $R^{op}: B \leftrightarrow A$ è totale.*
2. *$R: A \leftrightarrow B$ è iniettiva se e solo se $R^{op}: B \leftrightarrow A$ è univalente.*

Dimostrazione. Dimostriamo il primo punto, il secondo lo lasciamo come esercizio. Dalla Proposizione 2.4.30.1 R^{op} è totale se e solo se $(R^{op})^{op}$ è surgettiva. Ma, visto che $(R^{op})^{op} = R$, R^{op} è totale se e solo se R è surgettiva. ■

Esercizio 2.4.33. *Dimostrare la Proposizione 2.4.32.2.*

Teorema di caratterizzazione

Le quattro proprietà fondamentali che abbiamo studiato in questa sezione possono essere caratterizzate attraverso le operazioni su relazioni introdotte nella Sezione 2.2.

Teorema 2.4.34. *Per tutti gli insiemi A, B e per tutte le relazioni $R: A \leftrightarrow B$ vale che:*

1. R è totale se e solo se $Id_A \subseteq R; R^{op}$;
2. R è univalente se e solo se $R^{op}; R \subseteq Id_B$;
3. R è surgettiva se e solo se $Id_B \subseteq R^{op}; R$;
4. R è iniettiva se e solo se $R; R^{op} \subseteq Id_A$.

Dimostrazione. Dimostriamo il primo punto. Trattandosi di un se e solo se, possiamo dividere la dimostrazione in due parti:

- Se R è totale, allora $Id_A \subseteq R; R^{op}$. Assumendo che R sia totale, dobbiamo dimostrare che $Id_A \subseteq R; R^{op}$, cioè che per tutti gli $a \in A$, $(a, a) \in R; R^{op}$.

Prendiamo un qualsiasi elemento $a \in A$. Visto che R è totale, allora sappiamo che esiste almeno un $b \in B$ tale che $(a, b) \in R$. Per definizione di R^{op} abbiamo inoltre che $(b, a) \in R^{op}$. Pertanto, per definizione della composizione, si ha che $(a, a) \in R; R^{op}$.

- Se $Id_A \subseteq R; R^{op}$, allora R è totale. Assumendo che $Id_A \subseteq R; R^{op}$, dobbiamo dimostrare che R è totale, cioè che per tutti gli $a \in A$, esiste almeno un $b \in B$ tale che $(a, b) \in R$.

Prendiamo un qualsiasi elemento $a \in A$. Visto che $Id_A \subseteq R; R^{op}$, si ha che $(a, a) \in R; R^{op}$. Per definizione di composizione, questo significa che esiste almeno un $b \in B$ tale che $(a, b) \in R$ e $(b, a) \in R^{op}$.

Dimostriamo ora il secondo punto. Trattandosi di un se e solo se, possiamo dividere la dimostrazione in due parti:

- Se R è univalente, allora $R^{op}; R \subseteq Id_B$. Assumendo che R sia univalente, dobbiamo dimostrare che $R^{op}; R \subseteq Id_B$, cioè che per tutti gli $(x, y) \in R; R^{op}$, si ha che $(x, y) \in Id_B$, vale a dire che $x = y$.

Prendiamo una qualsiasi coppia $(x, y) \in R^{op}; R$. Per definizione di composizione, esiste un $a \in A$ tale che $(x, a) \in R^{op}$ e $(a, y) \in R$. Visto che $(x, a) \in R^{op}$, si ha che $(a, x) \in R$. Essendo R univalente, da $(a, x) \in R$ e $(a, y) \in R$ si può concludere che $x = y$.

- Se $R^{op}; R \subseteq Id_B$, allora R è univalente. Assumendo che $R^{op}; R \subseteq Id_B$, dobbiamo dimostrare che R è univalente, cioè che per tutti gli $a \in A$, esiste al più un $b \in B$ tale che $(a, b) \in R$.

Prendiamo un qualsiasi elemento $a \in A$ ed assumiamo che $(a, x) \in R$ e $(a, y) \in R$ per qualche $x, y \in B$. Per definizione di R^{op} , si ha che $(x, a) \in R^{op}$ e, per definizione di composizione, che $(x, y) \in R^{op}; R$. Visto che $R^{op}; R \subseteq Id_B$, si ha che $x = y$. Questo prova che R è univalente.

Dimostriamo il terzo punto, utilizzando il primo punto ed i risultati di dualità.

R è surgettiva se e solo se	$R^{op}: B \leftrightarrow A$ è totale	(Proposizione 2.4.32.1)
se e solo se	$Id_B \subseteq R^{op}; (R^{op})^{op}$	(primo punto)
se e solo se	$Id_B \subseteq R^{op}; R$	(convoluzione)

Lasciamo la dimostrazione del quarto punto come esercizio. ■

Esercizio 2.4.35. *Dimostrare il Teorema 2.4.34.4.*

Chiusura per composizione

Il precedente teorema ci fornisce una caratterizzazione che ci sarà utile in più occasioni. Adesso mostriamo come il teorema può essere utile per dimostrare che prese due relazioni che soddisfano una delle quattro proprietà, la loro composizione soddisfa ancora tale proprietà.

Proposizione 2.4.36. *Per tutti gli insiemi A, B, C e per tutte le relazioni $R: A \leftrightarrow B$, $S: B \leftrightarrow C$ vale che:*

1. *Se R e S sono totali, allora $R; S$ è totale.*
2. *Se R e S sono univalenti, allora $R; S$ è univalente.*
3. *Se R e S sono surgettive, allora $R; S$ è surgettiva.*
4. *Se R e S sono iniettive, allora $R; S$ è iniettiva.*

Mostriamo prima una dimostrazione per sostituzione che utilizza la caratterizzazione fornita dal Teorema 2.4.34.

Dimostrazione per sostituzione. Dimostriamo solo il primo punto e lasciamo gli altri per esercizio.

1. Se $Id_A \subseteq R; R^{op}$ e $Id_B \subseteq S; S^{op}$, allora $Id_A \subseteq (R; S); (R; S)^{op}$.

$$\begin{aligned}
 Id_A &\subseteq R; R^{op} && (R \text{ totale}) \\
 &\subseteq R; Id_B; R^{op} && (\text{unità}) \\
 &\subseteq R; (S; S^{op}); R^{op} && (S \text{ totale}) \\
 &\subseteq (R; S); (S^{op}; R^{op}) && (\text{associatività}) \\
 &\subseteq (R; S); (R; S)^{op} && (\text{distributività})
 \end{aligned}$$

■

Esercizio 2.4.37. Dimostrare per sostituzione le Proposizioni 2.4.36.2, 2.4.36.3 e 2.4.36.4.

Esercizio 2.4.38. Dimostrare per sostituzione che per tutti gli insiemi A , la relazione identità $Id_A: A \leftrightarrow A$ è totale, univalente, surgettiva e iniettiva.

Esercizio 2.4.39. Dimostrare per sostituzione che per tutti gli insiemi A, B , la relazione vuota $\emptyset: A \leftrightarrow B$ è univalente e iniettiva.

Esercizio 2.4.40. Dimostrare per sostituzione che per tutti gli insiemi A, B , la relazione completa $A \times B: A \leftrightarrow B$ è totale e surgettiva.

Mostriamo adesso una dimostrazione discorsiva del Teorema 2.4.36.

Dimostrazione discorsiva. Dimostriamo i primi tre punti. Il quarto lo lasciamo come esercizio.

1. Se R e S sono totali, allora $R; S$ è totale. Assumendo che R e S siano totali, dobbiamo dimostrare che $R; S$ è totale, cioè che per tutti gli $a \in A$, esiste almeno un $c \in C$ tale che $(a, c) \in R; S$.

Prendiamo un qualsiasi elemento $a \in A$. Visto che R è totale, allora esiste almeno un $b \in B$ tale che $(a, b) \in R$. Visto che S è totale allora esiste almeno un elemento $c \in C$ tale che $(b, c) \in S$. Per definizione di composizione abbiamo che $(a, c) \in R; S$.

2. Se R e S sono univalenti, allora $R; S$ è univalente. Assumendo che R e S siano univalenti, dobbiamo dimostrare che $R; S$ è univalente, cioè che per tutti gli $a \in A$, esiste al più un $c \in C$ tale che $(a, c) \in R; S$.

Prendiamo un qualsiasi elemento $a \in A$ ed assumiamo che $(a, x) \in R; S$ e $(a, y) \in R; S$ per qualche $x, y \in C$. Per la definizione di composizione, si ha che devono esistere $u, v \in B$ tali che: (1) $(a, u) \in R$, (2) $(u, x) \in S$, (3) $(a, v) \in R$ e (4) $(v, y) \in S$. Visto che R è univalente, da (1) e (3) si può dedurre che $u = v$. Visto che S è univalente, da (2) e (4) e dal fatto che $u = v$ si può dedurre che $x = y$. Pertanto $R; S$ è univalente.

3. Se R e S sono surgettive, allora $R; S$ è surgettiva. Si dimostra questo risultato utilizzando quello di sopra e i risultati di dualità. Se R e S sono surgettive, allora $R^{op}: B \leftrightarrow A$ e $S^{op}: C \leftrightarrow B$ sono totali. Pertanto, per il risultato appena dimostrato, $S^{op}; R^{op}$ è totale. Quindi, per la Proposizione 2.4.30.1 $(S^{op}; R^{op})^{op}$ è surgettiva. Si osservi che

$$(S^{op}; R^{op})^{op} = (R^{op})^{op}; (S^{op})^{op} \quad (\text{distributività di } .^{op} \text{ su } ;) \\ = R; S \quad (\text{convoluzione})$$

Pertanto $R; S$ è surgettiva. ■

Esercizio 2.4.41. Dimostrare la Proposizione 2.4.36.4.

Esercizio 2.4.42. È vero che per tutti gli insiemi A, B e per tutte le relazioni $R, S: A \leftrightarrow B$ vale che:

$$\text{Se } R \text{ e } S \text{ sono totali, allora } R \cup S \text{ è totale?}$$

In caso affermativo fornire una prova. In caso negativo fornire un controesempio.

Esercizio 2.4.43. È vero che per tutti gli insiemi A, B e per tutte le relazioni $R, S: A \leftrightarrow B$ vale che:

$$\text{Se } R \text{ e } S \text{ sono totali, allora } R \cap S \text{ è totale?}$$

In caso affermativo fornire una prova. In caso negativo fornire un controesempio.

Esercizio 2.4.44. È vero che per tutti gli insiemi A, B e per tutte le relazioni $R: A \leftrightarrow B, S: B \leftrightarrow C$ vale che:

$$\text{Se } R; S \text{ è totale, allora anche } R \text{ e } S \text{ sono totali?}$$

In caso affermativo fornire una prova. In caso negativo fornire un controesempio.

2.5 Funzioni

Definizione 2.5.1. [funzione] Siano A e B due insiemi. Una relazione $R \in \text{Rel}(A, B)$ è una FUNZIONE se è totale e univalente.

Esempio 2.5.2. Sia $A = \{x, y, z\}$ e $B = \{a, b, c, d\}$. La relazione

$$R = \{(x, a), (y, a), (z, d)\}: A \leftrightarrow B$$

è una funzione, mentre la relazione

$$S = \{(x, c), (x, d), (y, d), (z, d)\}: A \leftrightarrow B$$

non è una funzione perché non è univalente.

È fondamentale osservare che per definizione $R: A \leftrightarrow B$ è totale se e solo se

per ogni $a \in A$, esiste almeno un $b \in B$ tale che $(a, b) \in R$

mentre $R: A \leftrightarrow B$ è univalente se e solo se

per ogni $a \in A$, esiste al più un $b \in B$ tale che $(a, b) \in R$.

Pertanto $R: A \leftrightarrow B$ è una funzione se e solo se

per ogni $a \in A$, esiste esattamente un $b \in B$ tale che $(a, b) \in R$.

Notazione 2.5.3. Solitamente le funzioni vengono denotate da lettere minuscole, tipicamente f, g, h, \dots . Anziché dire una funzione tra A e B , come nelle relazioni, si dice una funzione da A a B . Anziché scrivere $f: A \leftrightarrow B$, si scrive $f: A \rightarrow B$. Inoltre si scrive $f(a)$ per denotare l'unico elemento $b \in B$ tale che $(a, b) \in f$. Per enfatizzare che tale elemento è b , si scrive $f(a) = b$. L'insieme di tutte le funzioni da A a B è denotato da $\text{Fun}(A, B)$, quindi $\text{Fun}(A, B) = \{f: A \rightarrow B\}$.

Esempio 2.5.4. La relazione successore $\text{Succ}: \mathbb{N} \leftrightarrow \mathbb{N}$ è una funzione, così come la relazione $\text{Plus}: \mathbb{N} \times \mathbb{N} \leftrightarrow \mathbb{N}$ (Esempio 2.1.9). In virtù di questo, d'ora in avanti, scriveremo $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}$ e $\text{plus}: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$. Invece la relazione $\text{Pred}: \mathbb{N} \leftrightarrow \mathbb{N}$ non è una funzione perché non è totale.

Esempio 2.5.5. Per tutti gli insiemi A , la relazione identità $\text{Id}_A: A \leftrightarrow A$ è una funzione. In virtù di questo, d'ora in avanti, scriveremo $\text{id}_A: A \rightarrow A$. La relazione vuota $\emptyset: A \leftrightarrow B$ e la relazione completa $A \times B$ solitamente non sono funzioni.

Esercizio 2.5.6. Per quale insieme A , la relazione vuota $\emptyset: A \leftrightarrow B$ e la relazione completa $A \times B$ sono sicuramente funzioni?

La notazione $f(a) = b$ può essere usata anche per definire funzioni: specificando $f(a) = b$ per ogni elemento a dell'insieme di partenza si definisce infatti una funzione.

Esempio 2.5.7. La funzione $R: A \rightarrow B$ dell'Esempio 2.5.2 può essere definita come

$$R(x) = a, \quad R(y) = a, \quad R(z) = d.$$

Quando la funzione è chiara dal contesto, piuttosto che scrivere $f(a) = b$, si può scrivere $a \mapsto b$.

Esempio 2.5.8. La funzione $R: A \rightarrow B$ dell'Esempio 2.5.2 può essere definita come

$$x \mapsto a, \quad y \mapsto a, \quad z \mapsto d.$$

Nel caso si considerino insiemi di partenza non finiti, questa notazione è abbastanza sconveniente perché necessita la specifica di $f(a) = b$ per gli infiniti elementi a dell'insieme di partenza (o i soliti puntini sospensivi \dots).

Esempio 2.5.9. La funzione $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}$ può essere definita come

$$\text{succ}(0) = 1, \quad \text{succ}(1) = 2, \quad \text{succ}(2) = 3, \quad \dots$$

In questi casi un modo conveniente di specificare funzioni è quello di usare delle espressioni con variabili:

$$f(x) = \langle \text{espressione il cui valore può dipendere da } x \rangle$$

L'idea è che per calcolare l'elemento associato ad un certo valore v dalla funzione f basta sostituire v al posto di x nell'espressione e valutare il risultato.

Esempio 2.5.10.

- $f_1(x) = x$
- $f_2(x) = x + 1$
- $f_3(x) = -x$
- $f_4(x) = x/2$
- $f_5(x) = \sqrt{x}$

Ma attenzione: di quale insieme di partenza e di arrivo stiamo parlando? Bisogna sempre chiarire quali siano gli insiemi di riferimento!

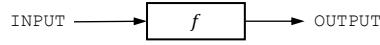
Per esempio la funzione f_1 può essere definita per ogni insieme A : questa è infatti la funzione identità $\text{id}_A: A \rightarrow A$. Invece la funzione f_2 non può essere definita per ogni insieme, perché l'espressione $x + 1$ ha significato solamente negli insiemi di numeri (come $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$). La funzione f_3 necessita dell'operazione di sottrazione – e quindi non può essere definita sui numeri naturali,

ma può essere definita sugli interi \mathbb{Z} . Similmente, per f_4 non sono sufficienti gli interi ma può essere definita sui razionali \mathbb{Q} . Ed infine per f_5 i numeri razionali non sono sufficienti ma questa funzione può essere definita sui reali positivi \mathbb{R}^+ .

Nel capitolo 3 vedremo come si può definire in modo rigoroso, usando l'*induzione*, una funzione avente come insieme di partenza i numeri naturali. In seguito nel capitolo 7 vedremo come questa tecnica può essere sfuttata per definire funzioni su di un qualunque insieme di partenza che sia stato definito induttivamente.

Le funzioni che solitamente sono studiate nei corsi di Analisi Matematica sono funzioni $f: \mathbb{R} \rightarrow \mathbb{R}$ dove l'insieme di partenza e quello di arrivo sono quello dei numeri reali \mathbb{R} (o dei loro sottinsiemi). Per questa ragione, spesso tali insiemi sono sottointesi e non vengono specificati.

In questo corso invece, e più in generale in tutta l'Informatica, consideriamo spesso funzioni tra insiemi che non sono numerici ed è quindi fondamentale **specificare sempre l'insieme di partenza e di arrivo**. Infatti, i programmi che scriviamo con i linguaggi di programmazione spesso sono funzioni dove l'insieme di partenza rappresenta l'insieme dei possibili valori di input del programma e l'insieme di arrivo l'insieme dei valori di output.



Di seguito mostriamo un esempio fondamentale nell'informatica.

Esempio 2.5.11. Si consideri l'insieme dei valori booleani $Bool = \{t, f\}$. La funzione $\neg: Bool \rightarrow Bool$, chiamata **NEGAZIONE**, è definita come:

$$\neg(t) = f \quad \neg(f) = t$$

In una funzione $f: A \rightarrow B$, l'insieme di partenza A può essere anche il prodotto cartesiano di due insiemi, vale a dire $A = A_1 \times A_2$ per qualche insieme A_1 e A_2 . In tal caso, anziché scrivere $f((a_1, a_2))$ (che, formalmente denota l'elemento associato da f alla coppia $(a_1, a_2) \in A_1 \times A_2$) scriveremo più semplicemente $f(a_1, a_2)$.

Esempio 2.5.12. Si consideri l'insieme dei valori booleani $Bool = \{t, f\}$. La funzione $\wedge: Bool \times Bool \rightarrow Bool$, chiamata **CONGIUNZIONE**, è definita come:

$$\begin{aligned} \wedge(t, t) &= t \\ \wedge(f, t) &= f \\ \wedge(t, f) &= f \\ \wedge(f, f) &= f \end{aligned}$$

La funzione $\vee: Bool \times Bool \rightarrow Bool$, chiamata **DISGIUNZIONE**, è definita come:

$$\begin{aligned} \vee(t, t) &= t \\ \vee(f, t) &= t \\ \vee(t, f) &= t \\ \vee(f, f) &= f \end{aligned}$$

Esempio 2.5.13. Un altro esempio di funzione in cui l'insieme di partenza è il prodotto cartesiano è la funzione $plus: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$. Con la notazione descritta sopra si ha, ad esempio che

$$plus(2, 1) = 3$$

Le funzioni che hanno come insieme di partenza il prodotto cartesiano di un insieme sono dette **BINARIE**. Per le funzioni binarie, è comune utilizzare la notazione infissa.

Notazione 2.5.14. Per le funzioni binarie anziché utilizzare la notazione $f(x, y)$, chiamata **NOTAZIONE PREFISSA**, si utilizza spesso la **NOTAZIONE INFISSA** $x \text{ } f \text{ } y$. Per esempio, $plus(2, 1)$ può essere scritto $2 \text{ } plus \text{ } 1$ (o più comunemente $2 + 1$) e $\wedge(t, t)$ come $t \wedge t$.

2. Relazioni

Diversamente dalla Matematica, in Informatica capita spesso di definire funzioni in cui l'insieme di partenza è il prodotto cartesiano di insiemi diversi.

Esempio 2.5.15. Si consideri la funzione $f: \text{Bool} \times \mathbb{N} \rightarrow \mathbb{N}$ che, ad ogni coppia $(x, y) \in \text{Bool} \times \mathbb{N}$, assegna 0 se $x = \text{f}$ e y se $x = \text{t}$. Come sottoinsieme di $((\text{Bool} \times \mathbb{N}) \times \mathbb{N})$ questa funzione è

$$f = \{((\text{f}, x), 0) \in (\text{Bool} \times \mathbb{N}) \times \mathbb{N} \mid x \in \mathbb{N}\} \cup \{((\text{t}, x), x) \in (\text{Bool} \times \mathbb{N}) \times \mathbb{N} \mid x \in \mathbb{N}\}.$$

Altrimenti si può definire con la notazione

$$f(x, y) = \begin{cases} 0 & \text{se } x = \text{f} \\ y & \text{se } x = \text{t} \end{cases}$$

Durante il corso utilizzeremo questa notazione di tanto in tanto.

Proprietà e implicazione

La nozione di funzione ci permette di rendere formale il concetto di *proprietà* che abbiamo utilizzato informalmente nella Sezione 1.1.

Per introdurre questo concetto è conveniente utilizzare degli esempi: *essere più alto di un 1 metro*, *essere nato prima del 1990*, *essere biondo*, o *aver superato l'esame di Fondamenti* sono proprietà sull'insieme degli esseri umani. Intuitivamente questo significa che ogni essere umano può soddisfare oppure non soddisfare una qualunque di queste proprietà.

In modo del tutto analogo, *essere multiplo di 3* o *essere primo* sono proprietà sull'insieme \mathbb{N} dei numeri naturali. Nuovamente, ogni numero naturale o è un multiplo di 3 o non lo è; o è primo oppure non lo è.

A livello intuitivo, una proprietà P su un insieme A è un'entità che, per ogni elemento $a \in A$, ci dice se a soddisfa la proprietà oppure no. Più formalmente una proprietà P su A è una funzione $P: A \rightarrow \text{Bool}$.

Definizione 2.5.16 (proprietà). *Sia A un insieme. Una PROPRIETÀ SU A è una funzione che ha come insieme di partenza A e come insieme di arrivo Bool . Per ogni elemento $a \in A$, si dice che a soddisfa la proprietà P se $P(a) = \text{t}$, mentre si dice che a non soddisfa la proprietà P , se $P(a) = \text{f}$.*

Nel seguito di questo testo, come avviene in generale in Matematica e in Informatica, per una proprietà P spesso scriveremo $P(a)$ al posto di $P(a) = \text{t}$ e $\neg P(a)$ per $P(a) = \text{f}$.

Date due proprietà P e Q definite su uno stesso insieme A , queste possono essere combinate utilizzando le funzioni booleane introdotte negli Esempi 2.5.11 e 2.5.12. Ad esempio la proprietà $P \wedge Q: A \rightarrow \text{Bool}$ è definita per ogni $a \in A$ come

$$P(a) \wedge Q(a)$$

dove $\wedge: \text{Bool} \times \text{Bool} \rightarrow \text{Bool}$ è la funzione su Booleani definita nell'Esempio 2.5.12. Si noti che $P \wedge Q(a) = \text{t}$ se e solo se entrambi le proprietà sono vere su a , cioè $P(a) = \text{t}$ e $Q(a) = \text{t}$.

In modo del tutto analogo $P \vee Q: A \rightarrow \text{Bool}$ è definita per ogni $a \in A$ come

$$P(a) \vee Q(a)$$

dove $\vee: \text{Bool} \times \text{Bool} \rightarrow \text{Bool}$ è la funzione su Booleani definita nell'Esempio 2.5.12. In altre parole $P \vee Q(a) = \text{t}$ se e solo se almeno uno tra $P(a)$ e $Q(a)$ vale t .

Infine la proprietà $\neg P: A \rightarrow \text{Bool}$ è definita similmente utilizzando la funzione $\neg: \text{Bool} \rightarrow \text{Bool}$ dell'Esempio 2.5.11: $\neg P(a) = \text{t}$ se e solo se $P(a) = \text{f}$.

Esiste un altro modo di combinare proprietà la cui comprensione è di importanza fondamentale. Come nel caso di \neg , \vee e \wedge si può definire a partire da una funzione binaria su booleani.

Definizione 2.5.17 (Implicazione). *Si consideri l'insieme dei valori booleani $\text{Bool} = \{\text{t}, \text{f}\}$. La funzione $\Rightarrow: \text{Bool} \times \text{Bool} \rightarrow \text{Bool}$, chiamata IMPLICAZIONE, è definita come:*

$$\Rightarrow(\text{t}, \text{t}) = \text{t}$$

$$\begin{aligned}\Rightarrow(f, t) &= t \\ \Rightarrow(t, f) &= f \\ \Rightarrow(f, f) &= t\end{aligned}$$

Come nel caso di \vee e \wedge , si preferisce spesso utilizzare la notazione infissa. Ad esempio, piuttosto che scrivere $\Rightarrow(t, t)$ si scrive $t \Rightarrow t$.

Si noti come la funzione \Rightarrow restituisca f in uno solo dei quattro casi possibili: quando il primo argomento è vero (t) ed il secondo è falso (f). Infatti, \Rightarrow cattura il significato intuitivo dell'implicazione: il primo argomento è la *premessa* e il secondo argomento è la *conseguenza*. Un'implicazione è falsa solamente quando la premessa è vera e la conseguenza è falsa.

Il significato dell'implicazione risulta ancora più chiaro quando si utilizza per comporre proprietà: la proprietà $P \Rightarrow Q: A \rightarrow \text{Bool}$ è definita per ogni $a \in A$ come

$$P(a) \Rightarrow Q(a)$$

dove $\Rightarrow: \text{Bool} \times \text{Bool} \rightarrow \text{Bool}$ è la funzione su Booleani della Definizione 2.5.17. Quando la premessa $P(a)$ è falsa, l'implicazione $P \Rightarrow Q$ assegna automaticamente t ad a . Quando $P(a)$ è vera, per valutare l'implicazione è necessario conoscere il valore di verità della conseguenza $Q(a)$: quando $Q(a) = t$, l'implicazione è vera; quando $Q(a) = f$, l'implicazione è falsa.

Abbiamo detto che $\neg P$, $P \vee Q$ e $P \wedge Q$ in italiano si leggono come *non P*, *P oppure Q* e *P e Q*. L'implicazione $P \Rightarrow Q$ si legge come

$$\text{se } P \text{ allora } Q.$$

Alcuni risultati incontrati fino ad ora (ad esempio nelle Proposizioni 1.4.12, 1.4.13, 1.7.5 2.4.36) utilizzano *se ... allora ...*. Formalmente questi risultati sono delle implicazioni $P \Rightarrow Q$. Il loro significato intuitivo è che *se* per un elemento a vale la premessa $P(a)$ *allora* deve valere anche la conseguenza $Q(a)$. Altrimenti se $P(a)$ non vale, non sappiamo niente del valore di verità di $Q(a)$.

Lo studente potrà riconoscere le implicazioni tra i risultati che presenteremo d'ora in avanti. Inoltre nel Capitolo 9 torneremo ad approfondire le proprietà e le loro combinazioni.

Composizione di funzioni

In prima approssimazione, dal momento che sono relazioni, le funzioni potrebbero essere combinate con le operazioni su relazioni che abbiamo visto nella Sezione 2.2. Sfortunatamente questo in molti casi non è possibile perché il risultato di tale operazione è una relazione ma non una funzione.

Prendiamo per esempio l'operazione di relazione opposta: la relazione opposta di una funzione è sicuramente una relazione surgettiva e iniettiva (grazie alla Proposizione 2.4.30), ma solitamente questa non è una funzione. Per un esempio concreto, il lettore può osservare che $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}$ è una funzione, mentre $\text{Pred} = \text{succ}^{\text{op}}: \mathbb{N} \leftrightarrow \mathbb{N}$ non è una funzione.

In modo del tutto analogo, le operazioni insiemistiche falliscono nel restituire una funzione: se $f: A \rightarrow B$ e $g: A \rightarrow B$ sono due funzioni, allora $f \cap g: A \leftrightarrow B$ sarà ancora univale, ma in generale non sarà totale. Similmente $f \cup g: A \leftrightarrow B$ sarà totale, ma non univale.

L'unica operazione che preserva la proprietà di restituire funzioni è l'operazione di composizione.

Proposizione 2.5.18. *Per tutti gli insiemi A, B, C e per tutte le funzioni $f: A \rightarrow B$ e $g: B \rightarrow C$, la relazione $f; g$ è una funzione.*

Dimostrazione. Assumendo che f e g siano totali e univalenti, dobbiamo dimostrare che $f; g$ è totale e univale.

Dalla Proposizione 2.4.36.1, si ha che $f; g$ è totale, visto che f e g sono totali. Dalla Proposizione 2.4.36.2, si ha che $f; g$ è univale, visto che f e g sono univalenti. ■

Visto che le leggi illustrate nella Sezione 2.3 valgono per tutte le relazioni, valgono in particolare anche per le funzioni.

Proposizione 2.5.19. *Per tutti gli insiemi A, B, C, D e per tutte le funzioni $f: A \rightarrow B$, $g: B \rightarrow C$, $h: C \rightarrow D$ vale che:*

2. Relazioni

1. $f; (g; h) = (f; g); h$; (associatività)
2. $id_A; f = f = f; id_B$. (unità)

È importante notare che, come per le relazioni, le funzioni possono essere composte solo se l'insieme di arrivo della prima coincide con l'insieme di partenza della seconda.

Esempio 2.5.20. La funzione $f: \text{Bool} \times \mathbb{N} \rightarrow \mathbb{N}$ dell'Esempio 2.5.15 può essere composta con la funzione $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}$. La funzione risultante $f; \text{succ}: \text{Bool} \times \mathbb{N} \rightarrow \mathbb{N}$ è definita come

$$(x, y) \mapsto \begin{cases} 1 & \text{se } x = \text{f} \\ y + 1 & \text{se } x = \text{t} \end{cases}$$

In Informatica si usa spesso la composizione di programmi visti come funzioni: $f; g$ significa intuitivamente che prima viene eseguito il programma f e poi l'output di questo viene passato come input al programma g .



La composizione di funzioni è molto importante anche in Matematica (per esempio, per calcolare le derivate) ma viene spesso utilizzata una notazione diversa.

Notazione 2.5.21. La composizione $f; g$ viene talvolta indicata come $g \circ f$, oppure, omettendo \circ , da gf . Inoltre l'elemento $f; g(a)$ viene denotato come $g(f(a))$ oppure, omettendo le parentesi più esterne, come $gf(a)$.

Per esempio per dire che f è la composizione della funzione $\text{seno}: \mathbb{R} \rightarrow \mathbb{R}$ e della funzione $\text{coseno} \cos: \mathbb{R} \rightarrow \mathbb{R}$, si scrive $f(x) = \cos(\text{sen}(x))$.

Teorema di caratterizzazione

Il Teorema 2.4.34 fornisce una caratterizzazione per le quattro proprietà fondamentali. Grazie a tale teorema, si ottiene banalmente una caratterizzazione delle funzioni.

Teorema 2.5.22. Per tutti gli insiemi A, B e per tutte le relazioni $R: A \leftrightarrow B$ vale che:

R è una funzione se e solo se $id_A \subseteq R; R^{op}$ e $R^{op}; R \subseteq id_B$.

Esercizio 2.5.23. Dimostrare il Teorema 2.5.22 utilizzando il Teorema 2.4.34.

Il seguente risultato ci sarà utile in seguito per dimostrare il Teorema 2.6.16.

Proposizione 2.5.24. Per tutti gli insiemi A, B e per tutte le relazioni $R: A \leftrightarrow B$ e $S: B \leftrightarrow A$ vale che se $id_A \subseteq R; S$ e $S; R \subseteq id_B$, allora $S = R^{op}$.

Dimostrazione. Dimostriamo separatamente $S \subseteq R^{op}$ e $R^{op} \subseteq S$ e concludiamo con l'antisimmetria di \subseteq (Proposizione 1.4.13).

- $S \subseteq R^{op}$. Dobbiamo dimostrare che, presa una generica coppia $(x, y) \in S$, questa appartiene anche a R^{op} .

Prendiamo $(x, y) \in S$. Si ha che $(y, y) \in id_A$ e, utilizzando l'ipotesi $id_A \subseteq R; S$, possiamo concludere che $(y, y) \in R; S$. Per la definizione di composizione, esiste almeno un $z \in B$ tale che $(y, z) \in R$ e $(z, y) \in S$.

Utilizzando la definizione di composizione, da $(x, y) \in S$ e $(y, z) \in R$, possiamo concludere che $(x, z) \in S; R$. Utilizzando l'ipotesi $S; R \subseteq id_B$, da $(x, z) \in S; R$ possiamo concludere che $x = z$. Da $x = z$ e $(y, z) \in R$, possiamo concludere che $(y, x) \in R$. Per definizione di relazione opposta, $(x, y) \in R^{op}$.

- $R^{op} \subseteq S$. Dobbiamo dimostrare che, presa una generica coppia $(x, y) \in R^{op}$, questa appartiene anche ad S .

Prendiamo $(x, y) \in R^{op}$. Per definizione di relazione opposta, si ha che $(y, x) \in R$. Si ha che $(y, y) \in id_A$ e, utilizzando l'ipotesi $id_A \subseteq R; S$, possiamo concludere che $(y, y) \in R; S$. Per la definizione di composizione, esiste almeno un $z \in B$ tale che $(y, z) \in R$ e $(z, y) \in S$.

Utilizzando la definizione di composizione, da $(z, y) \in S$ e $(y, x) \in R$, possiamo concludere che $(z, x) \in S; R$. Utilizzando l'ipotesi $S; R \subseteq id_B$, da $(z, x) \in S; R$ possiamo concludere che $x = z$. Pertanto $(x, y) \in S$. ■

Funzioni Parziali

Abbiamo spiegato a livello intuitivo che un programma calcola una funzione. Talvolta il calcolo può non andare a buon fine: il programma può per esempio restituire un messaggio di errore o addirittura non terminare. In questo caso non è vero che per tutti i valori di input il programma restituisce un output, cioè la funzione calcolata da un programma potrebbe non essere una relazione totale, ma solamente univale.

Definizione 2.5.25 (funzione parziale). *Siano A e B due insiemi ed R una relazione tra di loro. $R: A \leftrightarrow B$ si dice una FUNZIONE PARZIALE se R è univale.*

Se $R: A \leftrightarrow B$ è una funzione parziale e $a \in A$, diciamo che R È DEFINITA SU a se esiste un $b \in B$ tale che $(a, b) \in R$, altrimenti diciamo che R NON È DEFINITA SU a .

Per le funzioni parziali, si utilizza la stessa notazione che abbiamo descritto per le funzioni.

Esempio 2.5.26. *Si consideri $f: \mathbb{R} \rightarrow \mathbb{R}$ definita come*

$$f(x) = 1/x$$

La divisione di qualunque numero n per 0 non è definita. Per questa ragione f è una funzione parziale.

Come per le funzioni, le funzioni parziali possono essere composte e la loro composizione è associativa.

Proposizione 2.5.27. *Per tutti gli insiemi A, B, C e per tutte le funzioni parziali $f: A \rightarrow B$ e $g: B \rightarrow C$, la relazione $f; g$ è una funzione parziale.*

Proposizione 2.5.28. *Per tutti gli insiemi A, B, C, D e per tutte le funzioni parziali $f: A \rightarrow B$, $g: B \rightarrow C$, $h: C \rightarrow D$ vale che:*

1. $f; (g; h) = (f; g); h$; (associatività)
2. $id_A; f = f = f; id_B$. (unità)

Esercizio 2.5.29. Dimostrare le Proposizioni 2.5.27 e 2.5.28.

Funzioni surgettive ed iniettive

Il lettore avrà probabilmente sentito parlare di funzioni surgettive ed iniettive: queste sono relazioni che sono funzioni (quindi totali e univalenti) ed in aggiunta godono della proprietà surgettiva o di quella iniettiva.

Definizione 2.5.30 (funzione surgettiva). *Siano A e B due insiemi ed R una relazione tra di loro. $R: A \leftrightarrow B$ si dice una FUNZIONE SURGETTIVA se R è totale, univale e surgettiva.*

Definizione 2.5.31 (funzione iniettiva). *Siano A e B due insiemi ed R una relazione tra di loro. $R: A \leftrightarrow B$ si dice una FUNZIONE INIETTIVA se R è totale, univale e iniettiva.*

Come per le funzioni e le funzioni parziali, le funzioni surgettive ed iniettive possono essere composte

Proposizione 2.5.32. Per tutti gli insiemi A, B, C e per tutte le funzioni surgettive $f: A \rightarrow B$ e $g: B \rightarrow C$, la relazione $f; g$ è una funzione surgettiva.

Proposizione 2.5.33. Per tutti gli insiemi A, B, C e per tutte le funzioni iniettive $f: A \rightarrow B$ e $g: B \rightarrow C$, la relazione $f; g$ è una funzione iniettiva.

Esercizio 2.5.34. Dimostrare le Proposizioni 2.5.32 e 2.5.33.

2.6 Biiezioni

Le funzioni che sono allo stesso tempo surgettive ed iniettive sono particolarmente importanti e vengono dette biiezioni.

Definizione 2.6.1 (Biiezione). Siano A e B due insiemi ed R una relazione tra di loro. $R: A \leftrightarrow B$ si dice una BIIEZIONE se R è totale, univalente, surgettiva ed iniettiva. L'insieme delle biiezioni da A a B è denotato $Bii(A, B)$.

È importante notare che ogni biiezione è anche una funzione. Per questo utilizzeremo la stessa notazione che abbiamo usato per le funzioni.

Esempio 2.6.2. Per ogni insieme A , $id_A: A \rightarrow A$ è una biiezione.

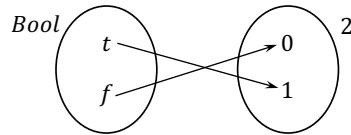
Esempio 2.6.3. Si consideri l'insieme $Bool = \{t, f\}$ e l'insieme $2 = \{0, 1\}$. La relazione $i: Bool \rightarrow 2$

$$\{(f, 0), (t, 1)\}$$

è una biiezione. Questa può anche essere specificata come

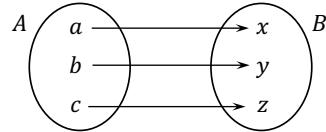
$$i(f) = 0, \quad i(t) = 1$$

e illustrata attraverso il seguente diagramma.



Esercizio 2.6.4. Esiste un'altra biiezione da $Bool$ a 2 ?

Esempio 2.6.5. Si consideri l'insieme $A = \{a, b, c\}$ e l'insieme $B = \{x, y, z\}$. La relazione



è una biiezione da A a B .

Esercizio 2.6.6. Siano A e B gli insiemi dell'Esempio 2.6.5. Esistono altre biiezioni da A a B ? In caso positivo, quante sono?

Come per le funzioni, le biiezioni possono essere composte.

Proposizione 2.6.7. Per tutti gli insiemi A, B, C e per tutte le biiezioni $f: A \rightarrow B$ e $g: B \rightarrow C$, la relazione $f; g$ è una biiezione.

Esercizio 2.6.8. Dimostrare la Proposizione 2.6.7.

Esiste però una differenza fondamentale tra funzioni e biiezioni: data una funzione $f: A \rightarrow B$ non necessariamente $f^{op}: B \leftrightarrow A$ è una funzione, invece se f è una biiezione, allora anche f^{op} lo è.

Proposizione 2.6.9. Per tutti gli insiemi A, B e per tutte le biiezioni $f: A \rightarrow B$, la relazione f^{op} è una biiezione.

Dimostrazione. Assumendo che f sia totale, univalente, surgettiva ed iniettiva, dobbiamo dimostrare che f^{op} ha le stesse proprietà.

- Grazie alla Proposizione 2.4.30.1, f^{op} è surgettiva perché f è totale.
- Grazie alla Proposizione 2.4.30.2, f^{op} è iniettiva perché f è univalente.
- Grazie alla Proposizione 2.4.32.1, f^{op} è totale perché f è surgettiva.
- Grazie alla Proposizione 2.4.32.2, f^{op} è univalente perché f è iniettiva.

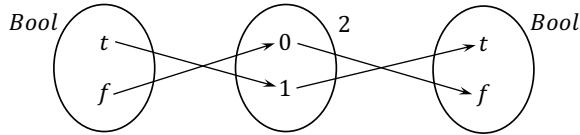
■

Esercizio 2.6.10. Dimostrare che se f^{op} è una biiezione allora anche f lo è.

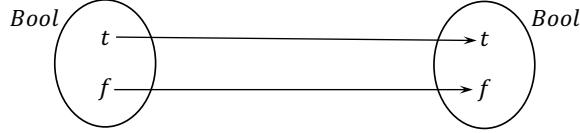
Esercizio 2.6.11. Si ricordi la biiezione i dell’Esempio 2.6.3. Illustrare la sua biiezione opposta attraverso un diagramma.

Esercizio 2.6.12. Si ricordi la biiezione dell’Esempio 2.6.5. Illustrare la sua biiezione opposta attraverso un diagramma.

Componendo una biiezione $f: A \rightarrow B$ con la sua biezione opposta $f^{op}: B \rightarrow A$ si ha che $f; f^{op} = id_A$. Per esempio, componendo la biiezione $i: Bool \rightarrow 2$ dell’Esempio 2.6.3 con $i^{op}: 2 \rightarrow Bool$



si ottiene $i; i^{op}: Bool \rightarrow Bool$.



Similmente componendo $f^{op}: B \rightarrow A$ con $f: A \rightarrow B$ si ottiene che $f^{op}; f = id_B$.

Questa proprietà fornisce una caratterizzazione delle biiezioni tra tutte le relazioni.

Teorema 2.6.13. Per tutti gli insiemi A, B e per tutte le relazioni $R: A \leftrightarrow B$ vale che:

R è una biiezione se e solo se $R; R^{op} = id_A$ e $R^{op}; R = id_B$.

Esercizio 2.6.14. Dimostrare il Teorema 2.6.13. Suggerimento: si utilizzi il Teorema 2.4.34.

Teorema di caratterizzazione attraverso relazioni invertibili

La caratterizzazione fornita dal Teorema 2.6.13 può essere resa ancora più forte utilizzando la nozione di relazione inversa.

Definizione 2.6.15 (Relazione inversa). Siano $R: A \leftrightarrow B$ e $S: B \leftrightarrow A$ due relazioni. Si dice che S è l’INVERSA di R se $R; S = id_A$ e $S; R = id_B$. Si dice che R è INVERTIBILE se esiste almeno una relazione inversa di R .

Le trasformazioni invertibili giocano un ruolo chiave in moltissimi campi. Per esempio in Informatica, sono molto studiate per la Crittografia (la materia che studia come cifrare i messaggi in modo sicuro) e nelle computazioni quantistiche. Purtroppo, questi argomenti non possono essere trattati in questo corso introduttivo e ci limiteremo quindi a fare funzioni invertibili il loro uso più classico, cioè quello di mettere in biiezione insiemi apparentemente distinti.

Teorema 2.6.16. Per tutti gli insiemi A, B e per tutte le relazioni $R: A \leftrightarrow B$ vale che:

R è una biiezione se e solo se R è invertibile.

Dimostrazione. Trattandosi di un se e solo se possiamo dimostrare separatamente che

- Se R è una biiezione, allora R è invertibile;
- Se R è invertibile, allora R è una biiezione.

Partiamo dal primo punto. Se R è una biiezione allora, grazie al Teorema 2.6.13, $R; R^{op} = id_A$ e $R^{op}; R = id_B$. Pertanto, prendendo $S = R^{op}$ si ha che R è invertibile.

Dimostriamo adesso il secondo punto. Se esiste un S tale che $id_A = R; S$ e $S; R = id_B$, allora per la Proposizione 2.5.24, si ha che $S = R^{op}$. Pertanto, per il Teorema 2.6.13, R è una biiezione. ■

Nel resto delle note scriveremo R^{-1} al posto di R^{op} quando R è una biiezione, per enfatizzare che R^{op} è l'inversa di R .

Insiemi in biiezione

La caratterizzazione delle biiezioni come relazioni invertibili è molto interessante: dopo aver applicato una relazione invertibile $i: A \rightarrow B$ ad un qualsiasi elemento in A , è sempre possibile tornare indietro all'elemento originale utilizzando $i^{-1}: B \rightarrow A$. Viceversa, trasformando prima con $i^{-1}: B \rightarrow A$ è sempre possibile tornare all'elemento originale utilizzando i .

$$a \quad \xrightarrow{i} \quad b \quad \xrightarrow{i^{-1}} \quad a$$

$$b \quad \xrightarrow{i^{-1}} \quad a \quad \xrightarrow{i} \quad b$$

Intuitivamente l'esistenza della biiezione $i: A \rightarrow B$ garantisce che ci possiamo spostare avanti e indietro tra gli elementi di A e gli elementi di B senza perdere alcuna "informazione". Questo permette quasi di identificare gli insiemi A e B come se fossero lo stesso insieme.

Definizione 2.6.17 (Insiemi in biiezione). Due insiemi A e B si dicono IN BIIEZIONE (o in corrispondenza uno a uno) se esiste una biiezione $i: A \rightarrow B$. In questo caso scriviamo $A \cong B$.

Esempio 2.6.18. Abbiamo mostrato che esiste una biiezione dall'insieme $Bool = \{\text{t}, \text{f}\}$ all'insieme $2 = \{0, 1\}$. I due insiemi sono quindi in biiezione: $Bool \cong 2$. È molto comune, in Informatica e in Logica identificare questi due insiemi, identificando t con 1 e f con 0.

Esercizio 2.6.19. Dimostrare che per tutti gli insiemi A , se $A \cong \emptyset$, allora $A = \emptyset$.

Nella Definizione 2.5.16 abbiamo introdotto le proprietà su A come funzioni da A in $Bool$. Il seguente risultato ci dice un fatto abbastanza intuitivo: le proprietà su A sono in biiezione con i sottoinsiemi di A .

Proposizione 2.6.20. Per tutti gli insiemi A , vale che:

$$\mathcal{P}(A) \cong Fun(A, Bool)$$

Dimostrazione. Ricordiamo che $\mathcal{P}(A)$ è l'insieme delle parti di A (Definizione 1.5.2), mentre $Fun(A, Bool)$ è l'insieme di tutte le funzioni $f: A \rightarrow Bool = \{\text{f}, \text{t}\}$ (Notazione 2.5.3). Mostriamo per prima cosa una funzione $i: \mathcal{P}(A) \rightarrow Fun(A, Bool)$. Per ogni insieme $B \subseteq A$, definiamo la sua FUNZIONE CARATTERISTICA $\chi_B: A \rightarrow Bool$ come

$$\chi_B(a) = \begin{cases} \text{t} & \text{se } a \in B \\ \text{f} & \text{se } a \notin B \end{cases}$$

Equivalentemente, possiamo definire questa funzione come una relazione, cioè un sottoinsieme di $A \times Bool$

$$\chi_B = \{(a, \text{t}) \in A \times Bool \mid a \in B\} \cup \{(a, \text{f}) \in A \times Bool \mid a \notin B\}$$

La funzione $i: \mathcal{P}(A) \rightarrow \text{Fun}(A, \text{Bool})$ mappa ogni $B \subseteq A$ nella sua funzione caratteristica χ_B .

$$\begin{array}{rcl} i: & \mathcal{P}(A) & \rightarrow \text{Fun}(A, \text{Bool}) \\ & B \subseteq A & \mapsto \quad \chi_B \end{array}$$

Mostriamo ora una funzione $j: \text{Fun}(A, \text{Bool}) \rightarrow \mathcal{P}(A)$. Per ogni funzione $f: A \rightarrow \text{Bool}$, definiamo $\text{Sub}(f) \subseteq A$ come

$$\text{Sub}(f) = \{x \in A \mid f(x) = \text{t}\}$$

La funzione j associa ad ogni $f: A \rightarrow \text{Bool}$ il sottoinsieme $\text{Sub}(f)$.

$$\begin{array}{rcl} j: & \text{Fun}(A, \text{Bool}) & \rightarrow \mathcal{P}(A) \\ & f: A \rightarrow \text{Bool} & \mapsto \text{Sub}(f) \end{array}$$

Per mostrare che i è una biiezione, mostriamo che j è la sua funzione inversa, cioè che

$$i; j = id_{\mathcal{P}(A)} \quad \text{e} \quad j; i = id_{\text{Fun}(A, \text{Bool})}$$

- $i; j = id_{\mathcal{P}(A)}$ Dobbiamo dimostrare che per tutti $B \in \mathcal{P}(A)$ vale che $j(i(B)) = id_{\mathcal{P}(A)}(B)$, cioè che $\text{Sub}(\chi_B) = B$. Ma questo è immediato perché per tutti gli $a \in A$

$$a \in \text{Sub}(\chi_B) \text{ se e solo se } \chi_B(a) = \text{t} \text{ se e solo se } a \in B.$$

- $j; i = id_{\text{Fun}(A, \text{Bool})}$ Dobbiamo dimostrare che per tutte le $f \in \text{Fun}(A, \text{Bool})$ vale che $i(j(f)) = id_{\text{Fun}(A, \text{Bool})}(f)$, cioè che $\chi_{\text{Sub}(f)} = f$. Ma questo è immediato perché per tutti gli $a \in A$, vale che:

$$\chi_{\text{Sub}(f)}(a) = \text{t} \text{ se e solo se } a \in \text{Sub}(f) \text{ se e solo se } f(a) = \text{t}$$

e

$$\chi_{\text{Sub}(f)}(a) = \text{f} \text{ se e solo se } a \notin \text{Sub}(f) \text{ se e solo se } f(a) = \text{f}.$$

■

Esempio 2.6.21. Siano $A = \{a, b, c\}$ e $B = \{a, c\}$. La funzione caratteristica $\chi_B: A \rightarrow \text{Bool}$ è

$$\begin{array}{rcl} \chi_B: & A & \rightarrow \text{Bool} \\ & a & \mapsto \quad \text{t} \\ & b & \mapsto \quad \text{f} \\ & c & \mapsto \quad \text{t} \end{array}$$

Esercizio 2.6.22. Sia $A = \{a, b, c, d\}$. Illustare la funzione caratteristica dei sottoinsiemi $\emptyset \subseteq A$, $A \subseteq A$ e $\{b, d\} \subseteq A$.

Nell'Esercizio 1.6.5 abbiamo visto che il prodotto cartesiano non è associativo, cioè che in generale

$$A \times (B \times C) \neq (A \times B) \times C$$

Comunque l'associatività vale *a meno di biiezione*, cioè i due insiemi anche se diversi sono in biiezione.

Proposizione 2.6.23. Per tutti gli insiemi A, B, C vale che

$$A \times (B \times C) \cong (A \times B) \times C$$

2. Relazioni

Dimostrazione. La funzione $i: A \times (B \times C) \rightarrow (A \times B) \times C$ è definita come

$$(a, (b, c)) \mapsto ((a, b), c)$$

mentre la funzione $j: (A \times B) \times C \rightarrow A \times (B \times C)$ come

$$((a, b), c) \mapsto (a, (b, c)).$$

È immediato vedere che j è la funzione inversa di i . ■

Il seguente risultato illustra un'importante biiezione che sta alla base della programmazione funzionale. Lasciamo la sua dimostrazione come esercizio.

Esercizio 2.6.24. *Dimostrare che per tutti gli insiemi A, B, C , vale che*

$$\text{Fun}(A \times B, C) \cong \text{Fun}(A, \text{Fun}(B, C))$$

Per \cong valgono le stesse proprietà dell'uguaglianza insiemistica.

Proposizione 2.6.25. *Per tutti gli insiemi A, B, C vale che:*

1. $A \cong A$ (riflessività);
2. Se $A \cong B$ e $B \cong C$, allora $A \cong C$ (transitività);
3. Se $A \cong B$, allora $B \cong A$ (simmetria).

Dimostrazione. Dimostriamo i primi due punti e lasciamo il terzo come esercizio.

1. $A \cong A$. Per dimostrare che $A \cong A$ dobbiamo esibire una biiezione $i: A \rightarrow A$. Visto che l'insieme di arrivo e di partenza sono lo stesso insieme, possiamo prendere $\text{id}_A: A \rightarrow A$ che, come illustrato nell'Esempio 2.6.2, è una biiezione.
2. Se $A \cong B$ e $B \cong C$, allora $A \cong C$; Per dimostrare che $A \cong C$ dobbiamo esibire una biiezione $i: A \rightarrow C$. Per ipotesi abbiamo una biiezione $f: A \rightarrow B$ ed una biiezione $g: B \rightarrow C$. La loro composizione $f; g: A \rightarrow C$ è una biiezione per la Proposizione 2.6.7. Possiamo quindi prendere $i = f; g$. ■

Esercizio 2.6.26. *Dimostrare la Proposizione 2.6.25.3*

Esercizio 2.6.27. *Dimostrare che per tutti gli insiemi A, A', B vale che*

$$\text{se } A \cong A', \text{ allora } \text{Fun}(A, B) \cong \text{Fun}(A', B).$$

Esercizio 2.6.28. *Dimostrare che per tutti gli insiemi A, B, B' vale che*

$$\text{se } B \cong B', \text{ allora } \text{Fun}(A, B) \cong \text{Fun}(A, B').$$

Esercizio 2.6.29. *Si ricordi che $1 = \{0\}$. Dimostrare che per tutti gli insiemi A , vale che*

$$A \times 1 \cong A$$

Esercizio 2.6.30. *Dimostrare che per tutti gli insiemi A, B , vale che*

$$A \times B \cong B \times A$$

2.7 *n*-uple, sequenze di lunghezza fissata e arbitraria

La Proposizione 2.6.23 ci dice che gli insiemi $A \times (B \times C)$ e $(A \times B) \times C$ anche se non sono uguali sono comunque in biezione. Per questa ragione, prenderemo la libertà di scrivere

$$A \times B \times C$$

senza dover specificare le parentesi. Inoltre, denoteremo gli elementi di questo insieme come delle TRIPLE

$$(a, b, c)$$

dove $a \in A$, $b \in B$ e $c \in C$. Formalmente, $A \times B \times C = \{(a, b, c) \mid a \in A, b \in B, c \in C\}$. Quando il prodotto cartesiano è fra 4 insiemi A, B, C, D scriveremo

$$A \times B \times C \times D$$

e denoteremo gli elementi di questo insieme come delle QUADRUPLE

$$(a, b, c, d).$$

Per il prodotto cartesiano per 5 insiemi, faremo la stessa cosa e chiameremo i suoi elementi quintupple. Lo stesso procedimento si applica ad un qualsiasi numero $n \in \mathbb{N}$ di insiemi e chiamiamo gli elementi n -uple. Per $n = 0$, esiste una sola 0-upla che denotiamo con ().

Le n -uple sono omnipresenti in Informatica. Infatti, mentre in matematica la maggior parte delle funzioni sono unarie o binarie, le funzioni che rappresentano programmi hanno spesso insiemi di partenza che sono il prodotto cartesiano di più di due insiemi.

Esempio 2.7.1. Mostriamo come esempio una semplice variante dell'Esempio 2.5.15. Consideriamo la funzione $f: \text{Bool} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ definita come

$$f(x, y, z) = \begin{cases} 0 & \text{se } x = \text{f} \\ y + z & \text{se } x = \text{t} \end{cases}$$

Questa funzione assegna ad ogni tripla $(x, y, z) \in \text{Bool} \times \mathbb{N} \times \mathbb{N}$, $y + z$ se $x = \text{t}$ e 0 altrimenti. Si noti che il primo elemento della tripla deve essere un elemento di Bool, mentre il secondo e il terzo sono numeri naturali in \mathbb{N} .

Un caso di n -upla particolarmente interessante è quello in cui tutti gli insiemi coinvolti nel prodotto cartesiano sono uguali.

Definizione 2.7.2 (sequenze). Sia A un insieme. Una SEQUENZA SU A DI LUNGHEZZA n è una n -upla $(a_0, a_1, \dots, a_{n-1})$ dove per tutti gli indici $i \in \{0, \dots, n-1\}$, $a_i \in A$. L'insieme A^n di tutte le sequenze su A di lunghezza n è definito come:

$$A^n = \{(a_0, a_1, \dots, a_{n-1}) \mid (\forall i \in \{0, \dots, n-1\}. a_i \in A)\}$$

Esempio 2.7.3. Sia $A = \{a, b\}$. Allora

$$\begin{aligned} A^0 &= \{\emptyset\} \\ A^1 &= \{(a), (b)\} \\ A^2 &= \{(a, a), (a, b), (b, a), (b, b)\} \\ A^3 &= \{(a, a, a), (a, a, b), (a, b, a), (a, b, b), (b, a, a), (b, a, b), (b, b, a), (b, b, b)\} \\ \dots &= \dots \end{aligned}$$

Osservazione 2.7.4. È interessante osservare che $A^0 \cong 1$, $A^1 \cong A$, $A^2 \cong A \times A$, $A^3 \cong A \times (A \times A)$ e, più in generale $A^n \cong A \times (\dots \times (A \times A) \dots)$. Questo può essere espresso attraverso l'induzione, un argomento che affronteremo in modo approfondito nel Capitolo 3 (si veda la Proposizione 3.1.4):

$$\begin{aligned} A^0 &\cong 1 \\ A^{n+1} &\cong A \times A^n \end{aligned}$$

Queste strutture sono molto usate in Matematica e Fisica: quando l'insieme A è l'insieme dei numeri reali \mathbb{R} , una sequenza su \mathbb{R} di lunghezza n è chiamata un *vettore* di \mathbb{R}^n e viene solitamente rappresentata senza virgole in riga $(r_0 \ r_1, \dots \ r_n)$ o in colonna.

$$\begin{pmatrix} r_0 \\ r_1 \\ \vdots \\ r_{n-1} \end{pmatrix}$$

L'insieme di tutte le sequenze su \mathbb{R} di lunghezza n , \mathbb{R}^n , forma uno *spazio vettoriale*: nel corso di Algebra Lineare studierete approfonditamente questo concetto.

In Informatica, le sequenze su A di lunghezza n sono tipicamente chiamate *array*, una struttura dati di base nella maggior parte dei linguaggi di programmazione. La sequenza

$$a = (a_0, a_1, \dots, a_{n-1})$$

viene solitamente rappresentata come

$$a = \boxed{a_0 \ | \ a_1 \ | \ \dots \ | \ a_{n-1}}$$

Esempio 2.7.5. Una sequenza su $2 = \{0, 1\}$ di lunghezza n è un array di n bits. Per esempio,

$$a = \boxed{0 \ | \ 1 \ | \ \dots \ | \ 1}$$

Una sequenza su \mathbb{Z} di lunghezza n è un array di n interi. Per esempio,

$$a = \boxed{3 \ | \ -1 \ | \ \dots \ | \ 26}$$

Ricordandosi che ogni numero naturale $n \in \mathbb{N}$ può essere visto come l'insieme $n = \{0, \dots, n-1\}$, ogni sequenza su A di lunghezza n può essere vista come una funzione dall'insieme n ad A .

$$\begin{aligned} (a_0, a_1, \dots, a_{n-1}): \quad n &\rightarrow A \\ i \in n &\mapsto a_i \in A \end{aligned}$$

Viceversa, ogni funzione da n ad A può essere vista come una sequenza su A di lunghezza n .

Osservazione 2.7.6. Per ogni $n \in \mathbb{N}$ esiste una biiezione tra A^n e $\text{Fun}(n, A)$, cioè $A^n \cong \text{Fun}(n, A)$.

Osservazione 2.7.7. Quando si pensa a un array come una funzione $a: n \rightarrow A$, $a(i)$ denota l' i -esimo elemento di a (a_i). Nella maggior parte dei linguaggi di programmazione, anziché scrivere $a(i)$ si scrive $a[i]$. Comunque molto spesso, gli array non sono visti come funzioni di tipo $n \rightarrow A$, ma come funzioni parziali di tipo $\mathbb{Z} \rightarrow A$. Infatti, è possibile passare ad un array un qualsiasi indice intero $i \in \mathbb{Z}$ e quando $i \notin n$ si ottiene un errore.

Concludiamo questa sezione con una nozione che ritroveremo frequentemente durante tutto il corso e in particolare nel Capitolo 8.

Definizione 2.7.8 (sequenze di lunghezza arbitraria). Una SEQUENZA SU A DI LUNGHEZZA ARBITRARIA è una sequenza su A di lunghezza n per un qualsiasi numero naturale $n \in \mathbb{N}$. L'insieme A^* di tutte le sequenze su A di lunghezza arbitraria è definito come

$$A^* = \bigcup_{n \in \mathbb{N}} A^n$$

Si noti che se l'insieme A ha un numero finito di elementi, allora per ogni $n \in \mathbb{N}$ l'insieme A^n è finito, ma in generale l'insieme A^* non lo è. Infatti

$$A^* = A^0 \cup A^1 \cup A^2 \cup \dots \cup A^n \cup \dots$$

Osservazione 2.7.9 (La stella di Kleene). *L'operazione che assegna ad ogni insieme A , l'insieme A^* è chiamata per ragioni storiche la stella di Kleene. Vedremo, nel Capitolo 4 che tale operazione può essere definita anche su certe relazioni.*

Esercizio 2.7.10. *Prendiamo l'insieme A come $1 = \{0\}$, dimostrare che $1^* \cong \mathbb{N}$.*

Esercizio 2.7.11. *Esiste un insieme A per il quale A^* è finito? Se sì, quale?*

Molto spesso le sequenze su A di lunghezza arbitraria sono chiamate *stringhe* sull'alfabeto A : anziché scrivere

$$(a_0, a_1, \dots, a_n)$$

si scrive semplicemente

$$a_0 a_1 \dots a_n$$

omettendo parentesi e virgolette. In questa prospettiva, l'unica sequenza di lunghezza 0, prima denotata da $()$, viene scritta come la *stringa vuota* ε .

Esempio 2.7.12. *Prendiamo come A l'insieme AN dei caratteri alfanumerici. L'insieme AN^* contiene tutte le stringhe composte da caratteri alfanumerici, come per esempio `si`, `123`, `anna`, `pippo`, `Pluto` e `topolino3`.*

Concludiamo questo capitolo con una generalizzazione del concetto di relazione che è usata molto comunemente. Dati tre insiemi A , B e C un sottoinsieme $R \subseteq A \times B \times C$ è chiamato, in molti libri di testo, una relazione su $A \times B \times C$. La differenza sostanziale con la nostra definizione di relazione è che in $R \subseteq A \times B \times C$ non è chiaro quale sia l'insieme di partenza (è A ? Oppure è $A \times B$?) e quale quello di arrivo. Utilizzando questa nozione più generale diviene quindi impossibile studiare le quattro proprietà illustrate in questo capitolo.

Il concetto si estende in modo del tutto ovvio ad un numero $n > 3$ di insiemi. Se tutti gli n insiemi coincidono si ha allora che $R \subseteq A^n$. In tal caso si parla di una relazione n -aria su A .

CAPITOLO 3

Induzione Matematica

L'*induzione* è un potente metodo formale che permette di definire in modo rigoroso non solo insiemi con un numero potenzialmente infinito di elementi, ma anche funzioni aventi tali insiemi come insieme di partenza, nonché per dimostrare che una proprietà è vera per tutti gli elementi di un tale insieme. In questo capitolo presentiamo l'*induzione matematica*, ampiamente utilizzata in matematica discreta e nell'informatica, che si basa su una definizione induttiva dei numeri naturali e può essere usata sia per introdurre funzioni definite sui naturali, sia per dimostrare che certe proprietà valgono per tutti i naturali. Nel Capitolo 7 vedremo che l'induzione matematica è un caso particolare dell'*induzione strutturale*, che permette di generalizzare queste tecniche a strutture dati di primaria importanza per l'informatica, come le liste e gli alberi binari.

3.1 I naturali, i numeri triangolari e la formula di Gauss

Cominciamo a esplorare questi concetti presentando un semplice esempio (probabilmente già noto al lettore) che comprende (1) la definizione induttiva dell'insieme dei naturali, (2) la definizione induttiva di una funzione sui naturali e (3) una dimostrazione per induzione di una semplice proprietà di tale funzione. L'esempio ha sia lo scopo di introdurre la terminologia rilevante che quello di mostrare come questi tre aspetti dell'induzione si complementano.

Cominciamo col chiederci: ma perché abbiamo bisogno di una nuova definizione dell'insieme \mathbb{N} dei naturali? Il lettore attento avrà osservato che nel Capitolo 1 abbiamo introdotto questo insieme come

- $\mathbb{N} = \{0, 1, 2, \dots\}$ (i **naturali**)

Questa definizione si basa sulla nostra capacità di capire la regola suggerita dai puntini sospensivi, che ci dice che aggiungendo 1 a qualunque elemento di questo insieme otteniamo il prossimo elemento dell'insieme. Ma questo tipo di definizione presenta diversi problemi,

- Come possiamo supporre che un'entità non intelligente (come un computer) possa capire questa definizione? Quanto meno dovremmo rendere esplicita la regola che genera gli elementi di questo insieme.
- Come possiamo essere sicuri che la regola che abbiamo inferito sia corretta? E se avessimo voluto definire l'insieme delle cifre decimali $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$? Oppure l'insieme delle radici dell'equazione $x^3 - 3x^2 + 2x = 0$, che sono proprio 0, 1 e 2?
- Abbiamo detto più volte che l'ordine in cui si elencano gli elementi di un insieme è irrilevante, ma allora che senso ha parlare del “prossimo elemento” di un insieme?
- Come possiamo determinare se un elemento appartiene oppure no all'insieme? Per esempio, come sappiamo che $\sqrt{9}$ appartiene ad \mathbb{N} mentre $\sqrt{10}$ no?

Un modo semplice di definire una collezione infinita di oggetti consiste nel fornire una regola per generare nuovi elementi a partire da quelli esistenti, con la tecnica descritta di seguito.

Definizione induttiva di un insieme

In generale, la DEFINIZIONE INDUTTIVA DI UN INSIEME è costituita da tre componenti:

1. La CLAUSOLA BASE, che stabilisce che certi oggetti appartengono all'insieme. Questi elementi costituiscono i mattoncini per costruire altri elementi dell'insieme.
2. La CLAUSOLA INDUTTIVA, che descrive in che modo gli elementi dell'insieme possono essere usati per produrre altri elementi dell'insieme.
3. La CLAUSOLA TERMINALE, che stabilisce che l'insieme che si sta definendo non contiene altri elementi oltre a quelli ottenuti dalle due clausole precedenti. Quindi l'insieme definito è il *più piccolo* insieme che soddisfa la clausola base e quella induttiva.

Definizione 3.1.1 (Definizione induttiva di \mathbb{N}). *L'insieme \mathbb{N} dei numeri naturali è l'insieme di numeri che soddisfa le seguenti clausole:*

1. $0 \in \mathbb{N}$.
2. Se $n \in \mathbb{N}$ allora $(n + 1) \in \mathbb{N}$.
3. Nessun altro elemento appartiene a \mathbb{N} .

Si noti che nella definizione consideriamo il concetto di “numero” e l’operazione di addizione come già conosciuti: implicitamente stiamo definendo i naturali come un sottoinsieme di un insieme di numeri più grande, come i reali. Questa definizione ci permette di mostrare per esempio che $\sqrt{9} = 3$ è un elemento di \mathbb{N} . Infatti dalla clausola base sappiamo che $0 \in \mathbb{N}$; quindi con la clausola induttiva otteniamo che $(0 + 1) = 1 \in \mathbb{N}$, e applicando ancora due volte quest’ultima otteniamo che $(1 + 1) = 2 \in \mathbb{N}$ e $(2 + 1) = 3 \in \mathbb{N}$, come desiderato. E cosa possiamo dire a proposito del numero 2.7, per esempio? Si osservi che l’insieme $\{0, 1, 1.7, 2, 2.7, 3, 3.7, 4, 4.7, \dots\}$ soddisfa le prime due clausole della definizione e contiene il numero 2.7. Ma questo insieme non soddisfa la clausola terminale: infatti 2.7 è diverso da 0, e non può essere ottenuto da 0 aggiungendo 1 un numero qualunque di volte. In altre parole, questo insieme non è il più piccolo che soddisfa le prime due clausole: anche $\{0, 1, 2, 3, 4, \dots\}$ le soddisfa, e non contiene 2.7. Quindi 2.7 non appartiene a \mathbb{N} , come sappiamo.

In modo più conciso ma del tutto equivalente, possiamo definire \mathbb{N} come *il più piccolo insieme* che soddisfa:

1. $0 \in \mathbb{N}$.
2. Se $n \in \mathbb{N}$ allora $(n + 1) \in \mathbb{N}$.

In questo caso abbiamo incorporato la clausola terminale nella premessa della definizione, dicendo che \mathbb{N} è *il più piccolo insieme* che soddisfa le due clausole. Molto spesso le definizioni induttive vengono presentate in questo modo, senza rendere esplicita la clausola terminale.

Definizione induttiva di una funzione

La definizione induttiva dell’insieme \mathbb{N} può essere sfruttata per definire in modo conciso funzioni $f : \mathbb{N} \rightarrow A$, per un qualche insieme A . Infatti per presentare la DEFINIZIONE INDUTTIVA DI UNA FUNZIONE su un insieme definito a sua volta induttivamente è sufficiente fornire (1) il valore della funzione sugli elementi che appartengono all’insieme per la clausola base, e (2) una regola per calcolare il valore della funzione sugli elementi che vi appartengono in base alla clausola induttiva, a partire dal suo valore sugli elementi che sono assunti appartenere all’insieme. Grazie alla clausola terminale della definizione induttiva siamo sicuri che i punti (1) e (2) sono sufficienti a definire la funzione per tutti gli elementi dell’insieme. Nel caso dei numeri naturali, per definire una funzione $f : \mathbb{N} \rightarrow A$ è sufficiente fornire (1) il valore di $f(0)$ e (2) un’espressione che definisce $f(n + 1)$ in termini di $f(n)$.

Per introdurre il prossimo esempio, ricordiamo che una successione di elementi di un insieme A ,

$$a_0, a_1, a_2, \dots, a_n, \dots$$

non è altro che una funzione $a : \mathbb{N} \rightarrow A$ che presentiamo scrivendo a_n invece di $a(n)$ per ogni $n \in \mathbb{N}$. Vediamo un esempio di funzione sui naturali definita induttivamente.

Definizione 3.1.2 (Definizione induttiva dei numeri triangolari). *Per ogni $n \in \mathbb{N}$, il NUMERO TRIANGOLARE T_n è uguale alla somma di tutti i numeri naturali da 1 a n :*

$$T_n = 1 + 2 + \cdots + n$$

In modo del tutto equivalente possiamo definire induttivamente la successione dei numeri triangolari (visti come una funzione $T : \mathbb{N} \rightarrow \mathbb{N}$) nel seguente modo:

1. $T_0 = 0$
2. $T_{n+1} = T_n + (n + 1)$

La clausola base non dovrebbe sorprendere: infatti, per definizione, T_0 è uguale alla somma di tutti i numeri naturali da 1 a 0, ma non ci sono numeri naturali che sono al tempo stesso maggiori o uguali a 1, e minori o uguali a 0. Quindi T_0 è la somma di un insieme vuoto di numeri, che vale ragionevolmente 0 (l'elemento neutro dell'addizione).

Sfruttando la definizione induttiva, abbiamo per esempio che $T_3 = T_2 + 3 = T_1 + 2 + 3 = T_0 + 1 + 2 + 3 = 0 + 1 + 2 + 3 = 6$. Quindi $T_3 = 6$, ma quanto vale T_n per un generico n ? A questo risponde quella che è conosciuta, per motivi storici, come la *formula di Gauss*:

$$\forall n \in \mathbb{N}. \left(T_n = \frac{n \cdot (n + 1)}{2} \right) \quad (3.1)$$

Come si può verificare la validità di questa formula, cioè che l'uguaglianza è vera per ogni numero $n \in \mathbb{N}$? Possiamo provare a vedere cosa succede in alcuni casi:

- Per $n = 0$ abbiamo $0 = \frac{0 \cdot 1}{2}$? OK!
- Per $n = 1$ abbiamo $0 + 1 = \frac{1 \cdot 2}{2}$? OK!
- Per $n = 2$ abbiamo $0 + 1 + 2 = \frac{2 \cdot 3}{2}$? OK!
- Per $n = 3$ abbiamo $0 + 1 + 2 + 3 = \frac{3 \cdot 4}{2}$? OK!
- ...

Ma questo non dimostra niente! In qualunque momento ci fermassimo, non avremmo dimostrato che la proprietà è valida, ma solo che l'uguaglianza è vera per un numero finito di valori (quelli esplicitamente considerati).

Il Principio di Induzione sui naturali

Per dimostrare la validità della formula di Gauss utilizziamo invece il PRINCIPIO DI INDUZIONE sui naturali. Data una generica proprietà $P(n)$ sui naturali, cioè un'asserzione che può essere vera o falsa al variare di $n \in \mathbb{N}$, il Princípio di Induzione stabilisce che:

Se (CASO BASE) $P(0)$ è vera, e se (PASSO INDUTTIVO) per ogni $n \in \mathbb{N}$ vale che se $P(n)$ è vera allora anche $P(n + 1)$ lo è, allora $P(m)$ è vera per ogni $m \in \mathbb{N}$.

In modo più compatto possiamo scrivere il Princípio di Induzione come una *regola di inferenza*:

$$\frac{P(0) \quad \forall n \in \mathbb{N}. (P(n) \Rightarrow P(n + 1))}{\forall m \in \mathbb{N}. P(m)} \text{ Princípio di Induzione}$$

Introdurremo formalmente le regole di inferenza nel Capitolo 9. Informalmente, la regola dice che per dimostrare la formula sotto la linea è sufficiente dimostrare le formule elencate sopra la linea.

Proposizione 3.1.3 (Correttezza della formula di Gauss). *La formula (3.1) è valida, cioè per ogni $n \in \mathbb{N}$ vale che*

$$T_n = \frac{n \cdot (n + 1)}{2}. \quad (3.2)$$

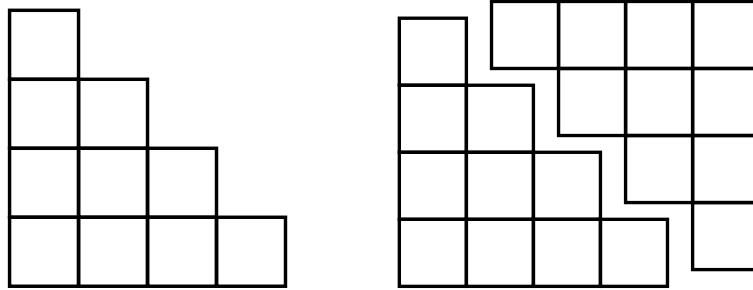


Figura 3.1: Rappresentazione grafica di T_4 e del suo doppio.

Dimostrazione. Utilizziamo il Principio di Induzione prendendo come proprietà $P(n)$ l'uguaglianza (3.2). In altre parole $P(n)$ è vera se e solo se l'uguaglianza (3.2) vale.

Si vuole dimostrare che $P(n)$ è vera per tutti i numeri naturali $n \in \mathbb{N}$. Grazie al Principio di Induzione è sufficiente dimostrare i seguenti casi:

1. [CASO BASE] Dobbiamo dimostrare che $P(n)$ è vera per $n = 0$, vale a dire si deve dimostrare che vale l'uguaglianza $T_0 = \frac{0 \cdot (0+1)}{2}$. Questo segue immediatamente dalla definizione di T_n e da evidenti calcoli aritmetici:

$$\begin{aligned} T_0 &= 0 && (\text{Punto 1 della Definizione 3.1.2}) \\ &= \frac{0 \cdot (0+1)}{2} && (\text{Aritmetica}) \end{aligned}$$

2. [PASSO INDUTTIVO] Dobbiamo dimostrare che se $P(n)$ è vera, allora anche $P(n + 1)$ è vera. In altre parole si deve dimostrare $P(n + 1)$, cioè che vale l'uguaglianza

$$T_{n+1} = \frac{n + 1 \cdot ((n + 1) + 1)}{2}$$

assumendo come ipotesi $P(n)$, cioè che vale l'uguaglianza (3.2). L'utilizzo dell'ipotesi $P(n)$, chiamata l'*ipotesi induttiva*, è fondamentale nelle dimostrazioni per induzione. Procediamo come segue:

$$\begin{aligned} T_{n+1} &= T_n + (n + 1) && (\text{Punto 2 della Definizione 3.1.2}) \\ &= \frac{n \cdot (n+1)}{2} + (n + 1) && (\text{Ipotesi induttiva}) \\ &= \frac{n \cdot (n+1) + 2 \cdot (n+1)}{2} && (\text{Aritmetica}) \\ &= \frac{(n+2) \cdot (n+1)}{2} && (\text{Aritmetica}) \\ &= \frac{(n+1) \cdot ((n+1)+1)}{2} && (\text{Aritmetica}) \end{aligned}$$

■

Possiamo verificare intuitivamente la correttezza della formula di Gauss anche in modo grafico. Nella parte sinistra della Figura 3.1 vediamo una rappresentazione grafica del numero triangolare T_4 , che spiega anche la scelta del nome. Nella parte destra abbiamo accostato in modo opportuno due copie di T_4 ottenendo un rettangolo di dimensione 4×5 , per cui vediamo subito che $T_4 = \frac{4 \cdot 5}{2}$.

Un altro esempio di dimostrazione per induzione sui naturali

Nella Definizione 2.7.2 abbiamo introdotto l'insieme A^n di tutte le sequenze di lunghezza n di elementi di un dato insieme A , definito come:

$$A^n = \{(a_0, a_1, \dots, a_{n-1}) \mid \forall i \in \{0, \dots, n-1\}. a_i \in A\}$$

Inoltre nell'Osservazione 2.7.4 abbiamo affermato che $A^0 \cong 1$ e che in generale

$$A^n \cong \underbrace{A \times (\cdots \times (A \times A) \cdots)}_{n \text{ volte}}$$

cioè che esiste una biiezione tra questi due insiemi. Dimostriamo per induzione che questa affermazione è vera.

Proposizione 3.1.4 (Sequenze e prodotto cartesiano). *Per ogni $n \in \mathbb{N}$ vale che*

$$A^n \cong \underbrace{A \times (\cdots \times (A \times A) \cdots)}_{n \text{ volte}}$$

Dimostrazione.

1. [CASO BASE] Il fatto che $A^0 \cong 1$ segue immediatamente dalle definizioni. Infatti $A^0 = \{()\}$ contiene un solo elemento, la sequenza vuota. Inoltre dall'enunciato è chiaro che stiamo considerando 1 come un insieme e quindi (si veda la fine della Sezione 1.5) anche $1 = \{0\}$ contiene un solo elemento. Quindi abbiamo una ovvia biiezione $i_0 : A^0 \rightarrow 1$ definita da $i_0(()) = 0$.

2. [PASSO INDUTTIVO] Assumiamo come ipotesi induttiva di avere una biiezione

$$i_n : A^n \rightarrow \underbrace{A \times (\cdots \times (A \times A) \cdots)}_{n \text{ volte}}$$

e mostriamo che esiste una biiezione

$$i_{n+1} : A^{n+1} \rightarrow \underbrace{A \times (\cdots \times (A \times A) \cdots)}_{n+1 \text{ volte}}$$

Infatti poiché ogni sequenza di $n + 1$ elementi può essere vista come un coppia composta di un elemento e di una sequenza di n elementi, abbiamo una ovvia biiezione $j_{n+1} : A^{n+1} \rightarrow A \times A^n$ definita da $(a_0, a_1, \dots, a_n) \mapsto (a_0, (a_1, \dots, a_n))$. Sfruttando la biiezione i_n che esiste per ipotesi induttiva possiamo definire la biiezione desiderata come

$$i_{n+1}((a_0, a_1, \dots, a_n)) = (a_0, i_n(a_1, \dots, a_n))$$

■

Sottoinsiemi di \mathbb{N} definiti induttivamente

La definizione induttiva dei naturali (Definizione 3.1.1) è paradigmatica. Con piccole variazioni possiamo definire opportuni sottoinsiemi di \mathbb{N} . Per esempio, sostituendo nella clausola base 0 con 1 si ottiene, ovviamente, una definizione induttiva di \mathbb{N}^+ .

Definizione 3.1.5 (Naturali positivi). *L'insieme \mathbb{N}^+ dei numeri naturali positivi è il più piccolo insieme che soddisfa:*

1. CLAUSOLA BASE: $1 \in \mathbb{N}^+$
2. CLAUSOLA INDUTTIVA: *Se $n \in \mathbb{N}^+$ allora $(n + 1) \in \mathbb{N}^+$.*

L'insieme dei numeri dispari (e analogamente quello dei numeri pari) può essere definito induttivamente come segue.

Definizione 3.1.6 (Numeri dispari). *L'insieme \mathbb{N}^d dei numeri dispari è il più piccolo insieme che soddisfa:*

1. CLAUSOLA BASE: $1 \in \mathbb{N}^d$

2. CLAUSOLA INDUTTIVA: Se $n \in \mathbb{N}^d$ allora $(n + 2) \in \mathbb{N}^d$.

Esercizio 3.1.7. L'insieme \mathbb{N} soddisfa sia la clausola base ($1 \in \mathbb{N}$) che la clausola induttiva (se $n \in \mathbb{N}$ allora $(n + 2) \in \mathbb{N}$) della definizione precedente. Perché questo non implica che $\mathbb{N} = \mathbb{N}^d$?

Esercizio 3.1.8. Fornire una definizione induttiva dell'insieme \mathbb{N}^p dei numeri pari.

Quando si intende dimostrare che una certa proprietà $P(n)$ vale per tutti gli elementi n di uno di questi sottoinsiemi (come per esempio \mathbb{N}^+ , \mathbb{N}^d o \mathbb{N}^p) è possibile utilizzare il principio di induzione, ma sono necessarie delle semplici modifiche. Di seguito enunciamo il principio di induzione per \mathbb{N}^+ che riutilizzeremo più volte in questo capitolo ma lasciamo come esercizi al lettore i principi per \mathbb{N}^p e \mathbb{N}^d . Una formulazione più generale del principio di induzione verrà fornita nel Capitolo 7.

Se (CASO BASE) $P(1)$ è vera, e se (PASSO INDUTTIVO) per ogni $n \in \mathbb{N}^+$ vale che se $P(n)$ è vera allora anche $P(n + 1)$ lo è, allora $P(m)$ è vera per ogni $m \in \mathbb{N}^+$.

In modo più compatto possiamo scrivere il Principio di Induzione per \mathbb{N}^+ come una regola di inferenza:

$$\frac{P(1) \quad \forall n \in \mathbb{N}^+ . (P(n) \Rightarrow P(n + 1))}{\forall m \in \mathbb{N}^+ . P(m)} \text{Principio di Induzione su } \mathbb{N}^+$$

Si osservi che l'unica differenza con il principio di induzione su \mathbb{N} consiste nel caso base: anziché dimostrare $P(0)$ è necessario dimostrare $P(1)$.

Nella prossima sezione, mostreremo un utilizzo di tale principio.

Esercizio 3.1.9. Enunciare il principio di induzione su \mathbb{N}^d

Esercizio 3.1.10. Enunciare il principio di induzione su \mathbb{N}^p

Esercizio 3.1.11. Fornire una definizione induttiva dell'insieme Pow-2 delle potenze di 2:

$$\text{Pow-2} = \{1, 2, 4, 8, 16, 32, 64, 128, \dots\}$$

Esercizio 3.1.12. Enunciare il principio di induzione su Pow-2

Fattoriale

Un esempio paradigmatico di funzione definita induttivamente sui naturali è il fattoriale. Il *fattoriale* di un numero $n \in \mathbb{N}$, scritto $n!$, è il prodotto di tutti i naturali da 1 a n . In simboli

$$n! = 1 \cdot 2 \cdot \dots \cdot n$$

Induttivamente è definito come segue.

Definizione 3.1.13 (Fattoriale). Il fattoriale di un numero $n \in \mathbb{N}$, scritto $n!$, è definito come:

1. $0! = 1$
2. $(n + 1)! = (n + 1) \cdot n!$

Ad esempio $4! = 4 \cdot 3! = 4 \cdot 3 \cdot 2! = 4 \cdot 3 \cdot 2 \cdot 1! = 4 \cdot 3 \cdot 2 \cdot 1 \cdot 0! = 4 \cdot 3 \cdot 2 \cdot 1 \cdot 1 = 24$.

Si noti la somiglianza con la definizione della successione di numeri triangolari (Definizione 3.1.2): nella clausola induttiva la somma è rimpiazzata con il prodotto, e nella clausola base 0 è rimpiazzato con 1. Quest'ultima potrebbe sembrare una scelta arbitraria, ma non lo è: in entrambe le definizioni induttive il primo elemento della successione è l'elemento neutro dell'operazione usata nella clausola induttiva. Infatti 0 è l'elemento neutro della somma, mentre 1 è quello del prodotto.

Vediamo un esempio di dimostrazione per induzione di una proprietà del fattoriale.

Proposizione 3.1.14 (Il fattoriale cresce più rapidamente dell'esponenziale). Per ogni $n \in \mathbb{N}^+$ vale che

$$n! \geq 2^{n-1}. \tag{3.3}$$

Dimostrazione. Utilizziamo il principio di induzione su \mathbb{N}^+ e quindi il caso base dovrà considerare 1 e non 0. Prendiamo come proprietà $P(n)$ la disuguaglianza (3.3). In altre parole $P(n)$ è vera se e solo se la disuguaglianza (3.3) vale.

Si vuole dimostrare che $P(n)$ è vera per tutti i numeri naturali $n \in \mathbb{N}^+$. Grazie al Principio di Induzione su \mathbb{N}^+ è sufficiente dimostrare i seguenti casi:

1. [CASO BASE] Dobbiamo dimostrare che $P(n)$ è vera per $n = 1$, vale a dire che $1! \geq 2^{1-1}$. Ma questo è quasi banale:

$$\begin{aligned} 1! &= 1 \cdot 0! && (\text{Punto 2 della Definizione 3.1.13}) \\ &= 1 \cdot 1 && (\text{Punto 1 della Definizione 3.1.13}) \\ &= 1 && (\text{Aritmetica}) \\ &= 2^0 && (\text{Aritmetica}) \\ &= 2^{1-1} && (\text{Aritmetica}) \end{aligned}$$

2. [PASSO INDUTTIVO] Dobbiamo dimostrare che se $P(n)$ è vera, allora anche $P(n + 1)$ è vera. In altre parole si deve dimostrare $P(n + 1)$, cioè che vale la disuguaglianza

$$(n + 1)! \geq 2^{(n+1)-1}$$

assumendo come ipotesi $P(n)$, cioè che vale la disuguaglianza (3.3).

$$\begin{aligned} (n + 1)! &= (n + 1) \cdot n! && (\text{Punto 2 della Definizione 3.1.13}) \\ &\geq (n + 1) \cdot 2^{n-1} && (\text{Ipotesi induttiva}) \\ &\geq 2 \cdot 2^{n-1} && (n \in \mathbb{N}^+, \text{ quindi } n + 1 \geq 2) \\ &= 2^n && (\text{Aritmetica}) \\ &= 2^{(n+1)-1} && (\text{Aritmetica}) \end{aligned}$$

■

3.2 Sommatorie, Produttorie, Unioni e Intersezioni n -arie.

Nei precedenti capitoli abbiamo incontrato diversi operatori binari che soddisfano le leggi di associatività ed unità (talvolta detta *elemento neutro*). Nel capitolo sugli insiemi abbiamo studiato l'unione \cup e l'intersezione \cap ; per i valori booleani abbiamo introdotto nell'Esempio 2.5.12 la disgiunzione \vee e la congiunzione \wedge ; e sui naturali conosciamo la somma $+$ e il prodotto \cdot . Tutti questi sono operatori binari, cioè hanno 2 argomenti, ma possono essere estesi ad un numero arbitrario $n \in \mathbb{N}$ di argomenti. In questa sezione mostriamo come l'induzione può aiutarci a questo scopo.

Iniziamo illustrando le sommatorie. Ricordiamo che una successione di elementi di un insieme A è una funzione $a: \mathbb{N} \rightarrow A$ e come al solito scriviamo a_i per denotare l'elemento $a(i)$ dell'insieme A . Qualche volta, per motivi tecnici, considereremo anche successioni definite come funzioni $a: \mathbb{N}^+ \rightarrow A$, in cui il primo elemento della successione è a_1 , e non a_0 .¹ In questo esempio consideriamo successioni di elementi di \mathbb{N} , ma tutte le definizioni si estendono banalmente ad insiemi, come \mathbb{Z} , \mathbb{Q} , \mathbb{R} , in cui sono definite le operazioni di somma (di seguito denotata con $+$) e prodotto (denotata con \cdot).

Dato un numero naturale $n \in \mathbb{N}$ e una successione di numeri naturali $a: \mathbb{N}^+ \rightarrow \mathbb{N}$, scriviamo

$$\sum_{i=1}^n a_i$$

per denotare la somma $a_1 + a_2 + \dots + a_n$. Questa notazione viene comunemente chiamata in matematica la *sommatoria degli a_i per i che va da 1 ad n* .

¹Lasciamo al lettore la dimostrazione che esiste una biiezione tra $\text{Fun}(\mathbb{N}^+, A)$ e $\text{Fun}(\mathbb{N}, A)$.

Grazie all'induzione è possibile dare una definizione più rigorosa di questa notazione, che evita l'utilizzo dei soliti fastidiosi puntini di sospensione “...”. Per ogni $n \in \mathbb{N}$, $\sum_{i=1}^n a_i$ è definita induttivamente come

1. $\sum_{i=1}^0 a_i = 0$
2. $\sum_{i=1}^{n+1} a_i = (\sum_{i=1}^n a_i) + a_{n+1}$

Per esempio, $\sum_{i=1}^3 a_i = (\sum_{i=1}^2 a_i) + a_3 = ((\sum_{i=1}^1 a_i) + a_2) + a_3 = (((\sum_{i=1}^0 a_i) + a_1) + a_2) + a_3 = ((0 + a_1) + a_2) + a_3 = a_1 + a_2 + a_3$. Si noti che nell'ultima uguaglianza abbiamo utilizzato la proprietà associativa della somma + e il fatto che 0 è l'elemento neutro.

Più in generale, dato un $k \in \mathbb{N}^+$, è comune scrivere

$$\sum_{i=k}^n a_i$$

per denotare la somma $a_k + a_{k+1} + \dots + a_n$. Questa notazione viene comunemente chiamata la *sommatoria degli a_i per i che va da k ad n* . Si noti che questa notazione generalizza la somma binaria + a una sequenza di $n - k + 1$ numeri, e che se $k > n$ allora la sequenza è vuota. Questo giustifica i casi base nella seguente definizione induttiva.

Definizione 3.2.1 (Sommatoria). *Per ogni $n \in \mathbb{N}$ e $k \in \mathbb{N}^+$, $\sum_{i=k}^n a_i$ è definita induttivamente come*

1. $\sum_{i=k}^0 a_i = 0$
2. $\sum_{i=k}^{n+1} a_i = \begin{cases} 0 & \text{se } k > n + 1 \\ (\sum_{i=k}^n a_i) + a_{n+1} & \text{se } k \leq n + 1 \end{cases}$

Per esempio, $\sum_{i=2}^4 a_i = (\sum_{i=2}^3 a_i) + a_4 = ((\sum_{i=2}^2 a_i) + a_3) + a_4 = (((\sum_{i=2}^1 a_i) + a_2) + a_3) + a_4 = (((0) + a_2) + a_3) + a_4 = a_2 + a_3 + a_4$.

Esercizio 3.2.2. *Si mostri che la definizione induttiva di $\sum_{i=1}^n a_i$ mostrata sopra è un caso particolare della Definizione 3.2.1.*

Un'altra notazione frequentemente utilizzata è quella della *produttoria*:

$$\prod_{i=k}^n a_i$$

denota il prodotto $a_k \cdot a_{k+1} \cdot \dots \cdot a_n$. La produttoria può essere definita induttivamente in modo analogo alla sommatoria: è sufficiente rimpiazzare + con · e 0 con 1.

Definizione 3.2.3 (Produttoria). *Per ogni $n \in \mathbb{N}$ e $k \in \mathbb{N}^+$, $\prod_{i=k}^n a_i$ è definita induttivamente come*

1. $\prod_{i=k}^0 a_i = 1$
2. $\prod_{i=k}^{n+1} a_i = \begin{cases} 1 & \text{se } k > n + 1 \\ (\prod_{i=k}^n a_i) \cdot a_{n+1} & \text{se } k \leq n + 1 \end{cases}$

Osservazione 3.2.4 (Indice). *Nelle sommatorie $\sum_{i=k}^n a_i$ e produttorie $\prod_{i=k}^n a_i$, la lettera i è detta l'indice perché intuitivamente scorre da k a n tutti i valori di a_i . Molto spesso la lettera i può essere rimpiazzata da altre lettere, ma è importante che sia rimpiazzata ovunque: ad esempio $\sum_{j=k}^n a_j$ è esattamente $\sum_{i=k}^n a_i$, ma $\sum_{j=k}^n a_i$ e $\sum_{i=k}^n a_j$ sono diversi.*

Esercizio 3.2.5. *Valutare $\sum_{j=2}^3 (j \cdot 3)$ e $\sum_{i=0}^3 (2 + \sum_{j=2}^3 (j \cdot i))$.*

Esempio 3.2.6 (numeri triangolari e fattoriale). *Avendo introdotto le sommatorie e le produttorie, è possibile esplicitare meglio la corrispondenza tra i numeri triangolari e i fattoriali. Infatti abbiamo, per definizione:*

$$T_n = \sum_{i=1}^n i \quad n! = \prod_{i=1}^n i$$

Si noti che le definizioni si applicano anche per $n = 0$: per la Definizione 3.2.1 abbiamo che $T_0 = 0$, mentre per la Definizione 3.2.3 abbiamo che $0! = 1$.

Esercizio 3.2.7 (*). Si dimostri per induzione la validità della formula di Nicomaco:

$$\forall n \in \mathbb{N}. \left(\sum_{i=0}^n i^3 = T_n^2 \right)$$

Quindi la somma dei cubi dei numeri naturali minori o uguali a n è uguale al quadrato dell' n -simo numero triangolare (Definizione 3.1.2).

In modo del tutto analogo a sommatorie e produttorie, possiamo definire unioni e intersezioni per famiglie di insiemi indicizzate da \mathbb{N} . Data una famiglia di insiemi $\{A_i\}_{i \in \mathbb{N}}$ e due numeri naturali $n, k \in \mathbb{N}$ tali che $k \leq n$ si scrive

$$\bigcup_{i=k}^n A_i \quad \text{e} \quad \bigcap_{i=k}^n A_i$$

per denotare, rispettivamente, l'unione $A_k \cup A_{k+1} \cup \dots \cup A_n$ e l'intersezione $A_k \cap A_{k+1} \cap \dots \cap A_n$. Queste notazioni vengono spesso chiamata unione n -aria ed intersezione n -aria. Si noti che l'unione n -aria è simile nello spirito a quella utilizzata nella Definizione 2.7.8, ma sostanzialmente diversa: nell'unione n -aria si fa l'unione di un numero *finito* di insiemi, mentre nella Definizione 2.7.8 di un numero *infinito* di insiemi.

Le definizioni induttive ricordano molto quelle di sommatoria e produttoria.

Definizione 3.2.8 (Unione n -aria). Per ogni $n \in \mathbb{N}$ e $k \in \mathbb{N}^+$, $\bigcup_{i=k}^n A_i$ è definita induttivamente come

1. $\bigcup_{i=k}^0 A_i = \emptyset$
2. $\bigcup_{i=k}^{n+1} A_i = \begin{cases} \emptyset & \text{se } k > n + 1 \\ (\bigcup_{i=k}^n A_i) \cup A_{n+1} & \text{se } k \leq n + 1 \end{cases}$

Definizione 3.2.9 (Intersezione n -aria). Per ogni $n \in \mathbb{N}$ e $k \in \mathbb{N}^+$, $\bigcap_{i=k}^n A_i$ è definita induttivamente come

1. $\bigcap_{i=k}^0 A_i = \mathcal{U}$
2. $\bigcap_{i=k}^{n+1} A_i = \begin{cases} \mathcal{U} & \text{se } k > n + 1 \\ (\bigcap_{i=k}^n A_i) \cap A_{n+1} & \text{se } k \leq n + 1 \end{cases}$

È interessante notare che nella definizione di unione n -aria si utilizza l'insieme vuoto \emptyset mentre nella definizione di intersezione si utilizza l'insieme universo \mathcal{U} . In modo sostanzialmente analogo, nella definizione di sommatoria si utilizza il numero naturale 0 mentre nella definizione di produttoria il numero 1. Questa analogia può essere spiegata in termini algebrici: \emptyset è l'elemento neutro di \cup , \mathcal{U} di \cap , 0 di $+$ e 1 di \cdot .

Esercizio 3.2.10. Definire la disgiunzione n -aria $\bigvee_{i=k}^n P_i$ e la congiunzione n -aria $\bigwedge_{i=k}^n P_i$ per una successione di booleani $P: \mathbb{N} \rightarrow \text{Bool}$.

Gli operatori n -ari definiti in questo modo, soddisfano leggi simili alle loro versioni binarie. Concludiamo questa sezione mostrando un esempio per unioni e intersezioni n -arie.

Abbiamo visto nell'Esempio 1.4.17 la dimostrazione della legge di De Morgan

$$\overline{(A \cup B)} = \overline{A} \cap \overline{B}$$

Ma possiamo generalizzare questa legge a più insiemi? Per esempio, è vero che

$$\overline{(A \cup B \cup C)} = \overline{A} \cap \overline{B} \cap \overline{C} ?$$

Usando l'induzione facciamo vedere come questa legge possa essere generalizzata a un numero finito qualunque di insiemi. (Naturalmente questo vale anche per l'altra legge di De Morgan riportata nella Tabella 1.4.)

Per una arbitraria famiglia di insiemi $\{A_i \mid i \in \mathbb{N}^+\}$, definiamo De Morgan n -ario, in simboli $DM(n)$, come la proprietà

$$\overline{\left(\bigcup_{i=1}^n A_i\right)} = \bigcap_{i=1}^n \overline{A_i}$$

Proposizione 3.2.11 (De Morgan n -ario). *Per ogni $n \in \mathbb{N}$, vale $DM(n)$.*

Dimostrazione. Utilizziamo il principio di induzione, prendendo come proprietà $P(n)$ esattamente $DM(n)$.

1. [CASO BASE] Dobbiamo dimostrare $DM(0)$:

$$\begin{aligned} \overline{\left(\bigcup_{i=1}^0 A_i\right)} &= \emptyset && (\text{Punto 1 Definizione 3.2.8}) \\ &= \mathcal{U} && (\text{Legge } \mathcal{U} : \emptyset \text{ in Tabella 1.4}) \\ &= \bigcap_{i=1}^0 \overline{A_i} && (\text{Punto 1 Definizione 3.2.9}) \end{aligned}$$

2. [PASSO INDUTTIVO] Prendiamo un generico n . Assumiamo come ipotesi induttiva che $DM(n)$ sia vero e dimostriamo che vale anche $DM(n+1)$.

Se vale $DM(n)$ vuol dire che

$$\overline{\left(\bigcup_{i=1}^n A_i\right)} = \bigcap_{i=1}^n \overline{A_i}$$

Vogliamo dimostrare $DM(n+1)$, cioè che

$$\overline{\left(\bigcup_{i=1}^{n+1} A_i\right)} = \bigcap_{i=1}^{n+1} \overline{A_i}$$

Notiamo che:

$$\begin{aligned} \overline{\left(\bigcup_{i=1}^{n+1} A_i\right)} &= \overline{\left(\bigcup_{i=1}^n A_i\right) \cup A_{n+1}} && (\text{Punto 2 Definizione 3.2.8}) \\ &= \overline{\left(\bigcup_{i=1}^n A_i\right)} \cap \overline{A_{n+1}} && (\text{De Morgan}) \\ &= \bigcap_{i=1}^n \overline{A_i} \cap \overline{A_{n+1}} && (\text{Ipotesi induttiva}) \\ &= \bigcap_{i=1}^{n+1} \overline{A_i} && (\text{Punto 2 Definizione 3.2.9}) \end{aligned}$$

■

3.3 Numeri di Fibonacci

Una successione numerica molto famosa è quella dei NUMERI DI FIBONACCI, che prendono il nome dal matematico pisano che li introdusse per risolvere il famoso *problema dei conigli*.

Il problema consiste nello studiare la crescita di una popolazione di conigli su un'isola, assumendo che:

- Al mese 0 viene posta una sola coppia di conigli (maschio e femmina) appena nati;
- Una coppia di conigli si riproduce solo quando entrambi hanno più di due mesi;
- Passati due mesi di età, ogni coppia produce un'altra coppia di conigli ogni mese;
- I conigli non muoiono mai, hanno tutto il cibo necessario e sull'isola non ci sono predatori;
- *Qual è il numero f_n di coppie di conigli dopo n mesi?*

Ragioniamo sul problema:

- Al mese 1 c'è una sola coppia ($f_1 = 1$);
- Al mese 2 c'è ancora una sola coppia ($f_2 = 1$);
- Al mese 3 l'unica coppia presente sull'isola ne produce un'altra ($f_3 = 2$);
- Al mese 4 la coppia più vecchia ne produce un'altra, mentre l'altra coppia è ancora troppo giovane ($f_4 = 3$);
- Al mese n ci sono ancora tutte le coppie che esistevano già al mese $n - 1$ e di queste, tutte (e sole) le coppie che esistevano al mese $n - 2$ adesso possono generare figli ($f_n = f_{n-1} + f_{n-2}$);

Quindi la successione di Fibonacci è definita induttivamente come segue.

Definizione 3.3.1. [Numeri di Fibonacci] Per ogni $n \in \mathbb{N}^+$, l' n -esimo numero di Fibonacci, scritto f_n , è definito induttivamente come

1. $f_1 = 1$
2. $f_2 = 1$
3. $f_n = f_{n-1} + f_{n-2}$ se $n > 2$

Si osservi che nella definizione induttiva della successione di Fibonacci la clausola base determina due valori, f_1 e f_2 , e la clausola induttiva definisce un elemento usando i due elementi precedenti. Questo mostra che le definizioni induttive di funzioni possono assumere una varietà di forme. Ma chi ci garantisce che una definizione così data sia corretta? Torneremo su questo problema nella Sezione 7.5: per il caso in esame ci convinciamo facilmente che la definizione è ben data perché f_n è definita in termini dei due valori *precedenti* della sequenza, e la clausola viene applicata solo per $n > 2$, quindi tali valori esistono.

La successione di Fibonacci gode di moltissime proprietà interessanti ed è argomento di studi approfonditi nella teoria dei numeri. Vediamo un esempio.

Proposizione 3.3.2. Per ogni $n \in \mathbb{N}^+$, vale che

$$\sum_{i=1}^n f_i^2 = f_n \cdot f_{n+1} \quad (3.4)$$

Dimostrazione. Per dimostrare che l'uguaglianza (3.4) vale per tutti gli $n \in \mathbb{N}^+$, utilizziamo il principio di induzione su \mathbb{N}^+ . Prendiamo come proprietà $P(n)$ l'uguaglianza (3.4).

3. Induzione Matematica

1. [CASO BASE] Dobbiamo dimostrare $P(1)$, cioè che

$$\sum_{i=1}^1 f_i^2 = f_1 \cdot f_2$$

Ma questo segue immediatamente dalla definizione di f_1 e f_2 :

$$\begin{aligned} \sum_{i=1}^1 f_i^2 &= (\sum_{i=1}^0 f_i^2) + f_1^2 && (\text{Punto 2 Definizione 3.2.1}) \\ &= 0 + f_1^2 && (\text{Punto 2 Definizione 3.2.1}) \\ &= 1 && (\text{Punto 1 Definizione 3.3.1, Aritmetica}) \\ &= 1 \cdot 1 && (\text{Aritmetica}) \\ &= f_1 \cdot f_2 && (\text{Punti 1 e 2 Definizione 3.3.1}) \end{aligned}$$

2. [PASSO INDUTTIVO] Dobbiamo dimostrare $P(n+1)$, cioè che

$$\sum_{i=1}^{n+1} f_i^2 = f_{n+1} \cdot f_{n+2}$$

assumendo come ipotesi $P(n)$, cioè che $\sum_{i=1}^n f_i^2 = f_n \cdot f_{n+1}$. Allora

$$\begin{aligned} \sum_{i=1}^{n+1} f_i^2 &= \sum_{i=1}^n f_i^2 + f_{n+1}^2 && (\text{Punto 2 Definizione 3.2.1}) \\ &= f_n \cdot f_{n+1} + f_{n+1}^2 && (\text{Ipotesi induttiva}) \\ &= f_n \cdot f_{n+1} + f_{n+1} \cdot f_{n+1} && (\text{Aritmetica}) \\ &= (f_n + f_{n+1}) \cdot f_{n+1} && (\text{Distributività di } \cdot \text{ rispetto a } +, \text{ al contrario}) \\ &= f_{n+2} \cdot f_{n+1} && (\text{Punto 3 della Definizione 3.3.1}) \end{aligned}$$

■

3.4 Principio di Induzione Forte sui naturali

Qualche volta il Principio di Induzione sui naturali non è abbastanza “potente” per completare la dimostrazione di una certa proprietà $P(n)$ in maniera agevole, perché per dimostrare che $P(n+1)$ è vera non basta l’ipotesi induttiva che $P(n)$ è vera, ma è conveniente assumere che $P(m)$ valga anche per altri valori minori di n .

Il PRINCIPIO DI INDUZIONE FORTE sui naturali ci permette di rafforzare le ipotesi del passo induttivo e portare avanti la dimostrazione in modo più semplice. Esso stabilisce che:

Se per ogni $n \in \mathbb{N}$ vale che se $P(0), P(1), \dots, P(n-1)$ sono vere allora anche $P(n)$ lo è, allora $P(m)$ è vera per ogni $m \in \mathbb{N}$.

In modo più compatto, come regola di inferenza:

$$\frac{\forall n . (P(0) \wedge P(1) \wedge \dots \wedge P(n-1) \Rightarrow P(n))}{\forall m . P(m)} \text{ Induzione Forte}$$

Si noti che il caso base non è scomparso, ma è inglobato nell’unica premessa della regola. Infatti per $n = 0$ il fatto che $P(0)$ valga deve essere dimostrato senza poter assumere alcuna premessa. Come si vede, per dimostrare che $P(n)$ valga, nel passo induttivo possiamo assumere non solo che $P(n-1)$ sia vero, ma che lo siano anche $P(0), P(1), P(2), \dots, P(n-2)$, cioè preso un n generico abbiamo più ipotesi a disposizione per dimostrare $P(n)$. Nonostante l’apparenza, non è difficile mostrare che il Principio di Induzione Forte e quello ordinario sono equivalenti, nel senso che permettono di dimostrare le stesse proprietà, ma questo va oltre gli obiettivi di queste note.

Esempio 3.4.1 (Teorema fondamentale dell’Aritmetica). *Usando il Principio di Induzione Forte sui naturali dimostriamo che ogni intero n maggiore di 1 o è un numero primo oppure può essere scritto come prodotto di numeri primi. Questo enunciato è parte del Teorema fondamentale dell’Aritmetica che stabilisce anche l’unicità di tale decomposizione.*

Procediamo quindi per induzione forte su n . Assumiamo che $n > 1$ e che ogni intero k con $1 < k < n$ o è primo oppure è esprimibile come prodotto di numeri primi. Se n stesso è primo non c’è niente da dimostrare, quindi supponiamo che $n = a \cdot b$, con $1 < a < n$ e $1 < b < n$. Per l’ipotesi induttiva ognuno tra a e b o è primo oppure è esprimibile come prodotto di primi. Ma allora poiché $n = a \cdot b$ anche n è esprimibile come un prodotto di numeri primi.

Esercizio 3.4.2. Si consideri la funzione

$$f(n) = \begin{cases} 0, & \text{se } n = 0 \\ 2 \cdot f\left(\frac{n}{2}\right), & \text{se } n \text{ è pari e maggiore di 0} \\ f(n-1) + 1, & \text{se } n \text{ è dispari.} \end{cases}$$

Si dimostri per induzione forte che $f(n) = n$ per ogni $n \in \mathbb{N}$.

Esercizio 3.4.3. Sia f_i l’ i -esimo numero di Fibonacci. Dimostrare per induzione forte che la seguente formula di Binet è corretta per ogni $n \in \mathbb{N}^+$:

$$f_n = \frac{1}{\sqrt{5}} \cdot \left(\frac{1+\sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \cdot \left(\frac{1-\sqrt{5}}{2} \right)^n$$

3.5 Esercizi

Esercizio 3.5.1. Dimostrare per induzione che per ogni $n \in \mathbb{N}$ il valore $n^3 - n$ è divisibile per 3.

Esercizio 3.5.2. Dimostrare per induzione che per ogni $n \in \mathbb{N}$ il valore $2^n \cdot 2^n - 1$ è divisibile per 3.

Esercizio 3.5.3. Dimostrare per induzione che per ogni $n \in \mathbb{N}$ il valore $n^2 + n$ è divisibile per 2.

Esercizio 3.5.4. Dimostrare per induzione che la somma dei primi n numeri dispari è uguale a n^2 . In formule:

$$\forall n \in \mathbb{N}^+ . \left(\sum_{i=1}^n (2 \cdot i - 1) = n^2 \right)$$

Esercizio 3.5.5. Si consideri la successione

1. $S_0 = 0$
2. $S_{n+1} = S_n + 1 + 2 \cdot n$

Si dimostri per induzione che $\forall n \in \mathbb{N} . S_n = n^2$.

Esercizio 3.5.6. Sia A un insieme e $\{A_i \mid i \in \mathbb{N}\}$ una famiglia di insiemi. Consideriamo la generalizzazione della proprietà distributiva:

$$D(n) = A \cap \left(\bigcup_{i=1}^n A_i \right) = \bigcup_{i=1}^n (A \cap A_i)$$

Dimostrare per induzione che:

$$\forall n \in \mathbb{N} . D(n)$$

Esercizio 3.5.7. Per concludere questa sezione, vediamo se riusciamo a convincervi che tutti i gatti hanno lo stesso colore con la seguente dimostrazione induttiva.

Sia $C(n) =$ “preso un qualsiasi insieme di n gatti, questi hanno tutti lo stesso colore” Vogliamo dimostrare che $\forall n \in \mathbb{N}^+ . C(n)$. Procediamo per induzione su n :

3. Induzione Matematica

- [CASO BASE] *Dobbiamo dimostrare che vale $C(1)$. Ma chiaramente, se l'insieme è composto di un solo gatto allora la proprietà è banalmente soddisfatta.*
- [PASSO INDUTTIVO] *Prendiamo un generico n . Assumiamo che $C(n)$ sia vero e dimostriamo che vale anche $C(n + 1)$.*

Allineiamo gli $n + 1$ gatti dell'insieme e consideriamo i due sottoinsiemi contenenti i primi n gatti e gli ultimi n gatti

Per ipotesi induttiva, i primi n gatti hanno lo stesso colore. Per ipotesi induttiva, gli ultimi n gatti hanno lo stesso colore.

Ma allora per transitività tutti gli $n + 1$ gatti hanno lo stesso colore!

Cosa c'è di sbagliato in questa dimostrazione?

CAPITOLO 4

Relazioni su un insieme

Nel Capitolo 2 abbiamo affrontato il concetto di relazione, al più alto livello di generalità. In Informatica, ma anche in Matematica e Fisica, ricoprono un ruolo di particolare interesse quelle relazioni in cui l'insieme di partenza e l'insieme di arrivo sono lo stesso insieme. In questo capitolo, ci concentreremo su questo tipo di relazioni: mostreremo la loro rappresentazione diagrammatica attraverso dei grafi e illustreremo quattro proprietà di comune interesse: riflessività, transitività, simmetria ed anti-simmetria. Vedremo poi il concetto di chiusura, con particolare enfasi sulla stella di Kleene (chiusura riflessiva e transitiva). Infine studieremo le relazioni di equivalenza e di ordinamento.

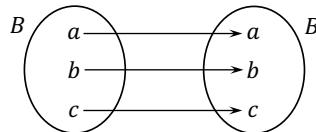
4.1 Nozioni di Base

Fino ad ora abbiamo studiato relazioni R tra un insieme A e un insieme B . In questo capitolo, ci concentriamo su relazioni in cui l'insieme di partenza e quello di arrivo sono lo stesso.

Definizione 4.1.1 (relazione su un insieme). *Una RELAZIONE R SULL'INSIEME A è un sottoinsieme del prodotto cartesiano $A \times A$, quindi $R \subseteq A \times A$. Indicheremo l'insieme di tutte le relazioni su A con la notazione $\text{Rel}(A, A)$, e una relazione $R \in \text{Rel}(A, A)$ come $R: A \leftrightarrow A$.*

Abbiamo già visto molti esempi di relazioni di questo tipo nel Capitolo 2: le relazioni di parentela sull'insieme degli esseri umani, la relazione identità su A (per un qualsiasi insieme A), le relazioni *succ* e *Pred* sull'insieme dei numeri naturali.

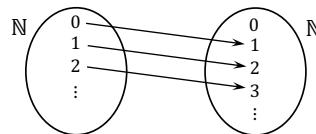
La prima peculiarità di questo tipo di relazioni consiste nella loro rappresentazione diagrammatica: anziché rappresentare due insiemi, quello di partenza sulla sinistra e quello di arrivo sulla destra, visto che i due insiemi sono lo stesso, si disegnano solamente gli elementi dell'insieme e, per ogni $(a, b) \in R$, si inserisce una freccia da a a b . Per esempio, la relazione id_B (per $B = \{a, b, c\}$) oltre ad essere disegnata come



può essere disegnata come



Similmente, la relazione *succ*: $\mathbb{N} \leftrightarrow \mathbb{N}$ oltre ad essere disegnata come



può essere rappresentata come

$$0 \rightsquigarrow 1 \rightsquigarrow 2 \rightsquigarrow \dots$$

Questo tipo di diagrammi vengono chiamati *grafi*: gli elementi di A vengono chiamati *nodi*, mentre gli elementi di R , rappresentati come frecce, vengono chiamate *archi*. I grafi rivestono un ruolo chiave in tutta l'Informatica per la loro importanza in molteplici applicazioni (per esempio nelle reti sociali) e la Teoria dei Grafi è considerata da molti una delle aree teoriche più interessanti. Affronteremo con maggiore dettaglio i grafi nel Capitolo 5 ma, per il momento, ci limiteremo ad utilizzarli come diagrammi per rappresentare relazioni su un insieme.

4.2 Proprietà di relazioni su un insieme

Tutto ciò che abbiamo visto sino ad adesso per le relazioni tra due insiemi vale chiaramente anche per le relazioni su un insieme. Ma ci sono diverse proprietà che ha senso considerare solo quando l'insieme di partenza e quello di arrivo coincidono.

Definizione 4.2.1 (relazione riflessiva). *Sia $R: A \leftrightarrow A$ una relazione su A . Si dice che R è RIFLESSIVA se per tutti gli elementi $a \in A$, vale che*

$$(a, a) \in R.$$

Esempio 4.2.2. *Per ogni insieme A , la relazione identità $id_A: A \leftrightarrow A$ e la relazione completa $A \times A: A \leftrightarrow A$ sono riflessive. La relazione vuota $\emptyset: A \leftrightarrow A$ solitamente non è riflessiva.*

Esercizio 4.2.3. *Per quale insieme A , la relazione vuota $\emptyset: A \leftrightarrow A$ è riflessiva?*

Esempio 4.2.4. *La relazione minore ($<$) sui naturali, cioè*

$$< = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid x < y\},$$

non è riflessiva. La relazione minore o uguale (\leq) sui naturali, cioè

$$\leq = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid x \leq y\},$$

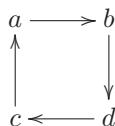
è riflessiva perché, per ogni $x \in \mathbb{N}$, $(x, x) \in \leq$.

Esempio 4.2.5. *Per $A = \{a, b, c, d\}$, la relazione $R = \{(a, a), (b, b), (a, d)\}$ non è riflessiva, mentre la relazione $S = \{(a, a), (b, b), (c, c), (d, d), (a, d)\}$ è riflessiva.*

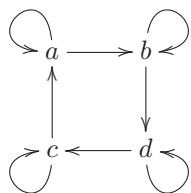
È molto facile identificare le relazioni riflessive attraverso i diagrammi: $R: A \leftrightarrow A$ è riflessiva se e solo se

per ogni nodo a c'è un arco 

Per esempio la relazione



non è riflessiva, mentre la relazione



è riflessiva.

Consideriamo adesso una seconda proprietà.

Definizione 4.2.6 (relazione transitiva). *Sia $R: A \leftrightarrow A$ una relazione su A . Si dice che R è TRANSITIVA se per tutti gli elementi $a, b, c \in A$, vale che*

$$\text{se } (a, b) \in R \text{ e } (b, c) \in R, \text{ allora } (a, c) \in R.$$

Esempio 4.2.7. Per ogni insieme A , la relazione identità $\text{id}_A: A \leftrightarrow A$, la relazione completa $A \times A: A \leftrightarrow A$ e la relazione vuota $\emptyset: A \leftrightarrow A$ sono transitive.

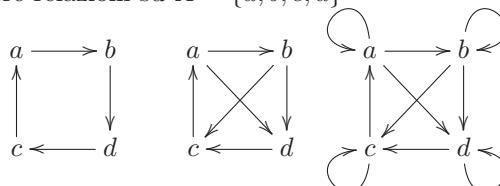
Esempio 4.2.8. Le relazioni $<: \mathbb{N} \leftrightarrow \mathbb{N}$ e $\leq: \mathbb{N} \leftrightarrow \mathbb{N}$ sono transitive.

Esempio 4.2.9. Per $A = \{a, b, c, d\}$, la relazione $R = \{(a, b), (b, c), (a, d)\}$ non è transitiva, mentre la relazione $S = \{(a, b), (b, c), (a, c), (a, d)\}$ lo è.

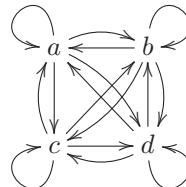
È molto facile identificare le relazioni transitive attraverso i diagrammi: $R: A \leftrightarrow A$ è transitiva se e solo se ogni volta che ci sono due archi



Per esempio le seguenti tre relazioni su $A = \{a, b, c, d\}$



non sono transitive, mentre la relazione



è transitiva.

Continuiamo la nostra esposizione con una terza proprietà.

Definizione 4.2.10 (relazione simmetrica). *Sia $R: A \leftrightarrow A$ una relazione su A . Si dice che R è SIMMETRICA se per tutti gli elementi $a, b \in A$, vale che*

$$\text{se } (a, b) \in R, \text{ allora } (b, a) \in R.$$

Esempio 4.2.11. Per ogni insieme A , la relazione identità $\text{id}_A: A \leftrightarrow A$, la relazione completa $A \times A: A \leftrightarrow A$ e la relazione vuota $\emptyset: A \leftrightarrow A$ sono simmetriche.

Esempio 4.2.12. Le relazioni $<: \mathbb{N} \leftrightarrow \mathbb{N}$ e $\leq: \mathbb{N} \leftrightarrow \mathbb{N}$ non sono simmetriche.

Esempio 4.2.13. Per $A = \{a, b, c, d\}$, la relazione $R = \{(a, b), (b, c), (a, d)\}$ non è simmetrica, mentre la relazione $S = \{(a, b), (b, c), (a, d), (b, a), (c, b), (d, a)\}$ lo è.

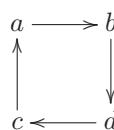
Come per le relazioni riflessive e transitive, è conveniente identificare le relazioni simmetriche attraverso i diagrammi: $R: A \leftrightarrow A$ è simmetrica se e solo se ogni volta che c'è un arco da a a b (per tutti i nodi a, b)



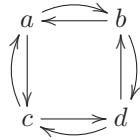
c'è anche un arco da b ad a



Per esempio, la relazione



non è simmetrica, mentre la relazione



è simmetrica.

Consideriamo adesso l'ultima proprietà.

Definizione 4.2.14 (relazione anti-simmetrica). *Sia $R: A \leftrightarrow A$ una relazione su A . Si dice che R è ANTI-SIMMETRICA se per tutti gli elementi $a, b \in A$, vale che*

$$\text{se } (a, b) \in R \text{ e } (b, a) \in R, \text{ allora } a = b.$$

Esempio 4.2.15. Per ogni insieme A , la relazione identità $\text{id}_A: A \leftrightarrow A$ e la relazione vuota $\emptyset: A \leftrightarrow A$ sono anti-simmetriche. La relazione completa $A \times A: A \leftrightarrow A$ solitamente non è anti-simmetrica.

Esercizio 4.2.16. Per quali insiemi A la relazione completa $A \times A: A \leftrightarrow A$ è anti-simmetrica?

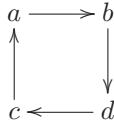
Esempio 4.2.17. Le relazioni $<: \mathbb{N} \leftrightarrow \mathbb{N}$ e $\leq: \mathbb{N} \leftrightarrow \mathbb{N}$ sono anti-simmetriche.

Esempio 4.2.18. Per $A = \{a, b, c, d\}$, la relazione $R = \{(a, b), (b, a), (a, d)\}$ non è anti-simmetrica, mentre la relazione $S = \{(a, b), (a, d)\}$ lo è.

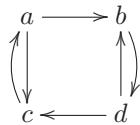
È molto facile identificare le relazioni anti-simmetriche attraverso i diagrammi: $R: A \leftrightarrow A$ è anti-simmetrica se e solo se non ci sono coppie di archi



Per esempio, la relazione



è anti-simmetrica, mentre la relazione



non lo è.

Teorema di caratterizzazione

Come per le proprietà viste nella Sezione 2.4, anche le relazioni riflessive, transitive, simmetriche e anti-simmetriche possono essere caratterizzate attraverso le operazioni su relazioni.

Teorema 4.2.19. Per tutti gli insiemi A e per tutte le relazioni $R: A \leftrightarrow A$ su A vale che:

1. R è riflessiva se e solo se $\text{id}_A \subseteq R$;
2. R è transitiva se e solo se $R; R \subseteq R$;
3. R è simmetrica se e solo se $R^{\text{op}} \subseteq R$;
4. R è anti-simmetrica se e solo se $R \cap R^{\text{op}} \subseteq \text{id}_A$;

Dimostrazione. Proviamo solo il punto 2. e lasciamo gli altri per esercizio.

$R; R \subseteq R$
 se e solo se
 per tutti gli $(a, c) \in A \times A$, se $(a, c) \in R; R$ allora $(a, c) \in R$
 se e solo se
 per tutti gli $(a, c) \in A \times A$, se $\exists b \in A . (a, b) \in R \wedge (b, c) \in R$ allora $(a, c) \in R$
 se e solo se
 R è transitiva.

■

Esercizio 4.2.20. Dimostrare i punti 1, 3 e 4 del Teorema 4.2.19.

Esercizio 4.2.21. Usando il Teorema 4.2.19, dimostrare che per tutti gli insiemi A e per tutte le relazioni $R: A \leftrightarrow A$ vale che:

$$R \text{ è simmetrica se e solo se } R \subseteq R^{op}$$

Esercizio 4.2.22. Usando il Teorema 4.2.19 e il risultato dell'Esercizio 4.2.21, dimostrare che per tutti gli insiemi A e per tutte le relazioni $R: A \leftrightarrow A$ vale che:

$$R \text{ è simmetrica se e solo se } R = R^{op}$$

4.3 Chiusure

Chiusura riflessiva

Data una relazione $R: A \leftrightarrow A$ è sempre possibile trasformarla in una relazione riflessiva seguendo una ricetta standard.

Definizione 4.3.1 (chiusura riflessiva). Sia $R: A \leftrightarrow A$ una relazione su A . La CHIUSURA RIFLESSIVA DI R è la relazione $R \cup id_A$.

Esempio 4.3.2. Per $A = \{a, b, c, d\}$, la relazione $R = \{(a, a), (b, b), (a, d)\}$ non è riflessiva. La chiusura riflessiva di R è la relazione $R \cup id_A = \{(a, a), (b, b), (c, c), (d, d), (a, d)\}$.

Esempio 4.3.3. La relazione minore $<$ sui naturali, non è riflessiva. La chiusura riflessiva di $<$ è la relazione $< \cup id_{\mathbb{N}}$ che è esattamente la relazione \leq .

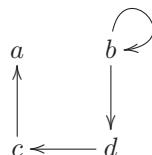
Esempio 4.3.4. La relazione vuota $\emptyset: A \leftrightarrow A$ non è riflessiva (se $A \neq \emptyset$). La chiusura riflessiva di \emptyset è la relazione id_A .

La chiusura riflessiva può essere visualizzata facilmente in modo grafico: è sufficiente inserire un arco

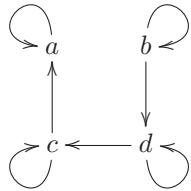


per ogni nodo x .

Per esempio la chiusura riflessiva della relazione



è rappresentata dal seguente grafo.



Proposizione 4.3.5. Per tutti gli insiemi A e per tutte le relazioni $R: A \leftrightarrow A$, vale che

- (1) $R \cup id_A$ è riflessiva;
- (2) $R \subseteq R \cup id_A$;
- (3) per tutte le relazioni $S: A \leftrightarrow A$, se $R \subseteq S$ e S è riflessiva, allora $R \cup id_A \subseteq S$.

Tale proposizione ci spiega il significato della parola *chiusura*: la chiusura riflessiva di R (1) è riflessiva, (2) contiene R e (3) è la più “piccola” tra tutte le relazioni che soddisfano (1) e (2).

Esercizio 4.3.6. Dimostrare la Proposizione 4.3.5.

Chiusura simmetrica

Per le relazioni simmetriche si può procedere in modo del tutto analogo.

Definizione 4.3.7 (chiusura simmetrica). Sia $R: A \leftrightarrow A$ una relazione su A . La CHIUSURA SIMMETRICA DI R è la relazione $R \cup R^{op}$.

Esempio 4.3.8. Per $A = \{a, b, c, d\}$, la relazione $R = \{(a, b), (b, c), (a, d)\}$ non è simmetrica. La sua chiusura simmetrica è $R \cup R^{op} = \{(a, b), (b, c), (a, d), (b, a), (c, b), (d, a)\}$.

Esempio 4.3.9. La relazione minore o uguale \leq sui naturali non è simmetrica. La chiusura simmetrica di \leq è la relazione $\leq \cup \leq^{op}$ che è esattamente la relazione completa $\mathbb{N} \times \mathbb{N}: \mathbb{N} \leftrightarrow \mathbb{N}$.

La chiusura simmetrica può essere costruita facilmente in modo grafico: è sufficiente aggiungere, per ogni arco da a a b ,

$$a \overbrace{\hspace{1cm}}^{\curvearrowright} b$$

un arco da b ad a

$$a \overbrace{\hspace{1cm}}^{\curvearrowright \curvearrowleft} b$$

Proposizione 4.3.10. Per tutti gli insiemi A e per tutte le relazioni $R: A \leftrightarrow A$, vale che

- (1) $R \cup R^{op}$ è simmetrica;
- (2) $R \subseteq R \cup R^{op}$;
- (3) per tutte le relazioni $S: A \leftrightarrow A$, se $R \subseteq S$ e S è simmetrica, allora $R \cup R^{op} \subseteq S$.

Quindi la chiusura simmetrica di R (1) è simmetrica, (2) contiene R e (3) è la più “piccola” tra tutte le relazioni che soddisfano (1) e (2).

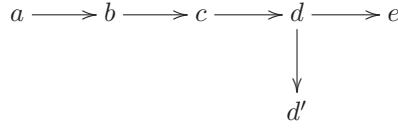
Esercizio 4.3.11. Dimostrare la Proposizione 4.3.10.

Chiusura transitiva

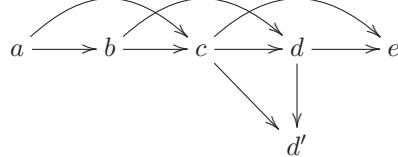
Per ottenere relazioni transitive, l’idea è di procedere in modo analogo alla chiusura simmetrica ed a quella riflessiva, ma alcune peculiarità rendono la procedura un po’ più complessa.

Data una relazione $R: A \leftrightarrow A$, per trasformarla in una relazione transitiva, il primo tentativo consiste nell’aggiungere tutte le coppie $(a, c) \in A \times A$ tali che $\exists b \in A . (a, b) \in R \wedge (b, c) \in R$. In altre parole questo significa unire ad R la relazione $R; R$.

Per esempio, se R è la relazione

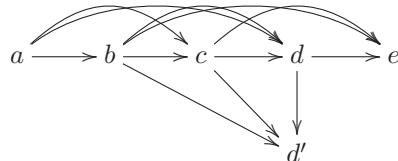


allora $R \cup (R; R)$ è la relazione



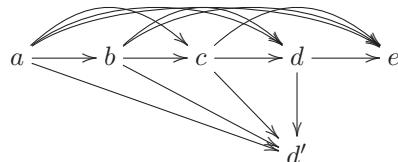
Si noti che tale relazione non è, come nei nostri desideri, transitiva: infatti $(b, c) \in R \cup (R; R)$ e $(c, d') \in R \cup (R; R)$, ma $(b, d') \notin R \cup (R; R)$.

Il secondo tentativo consiste nell'unire a $R \cup (R; R)$ la relazione $R; R; R$. La relazione $R \cup (R; R) \cup (R; R; R)$ è la seguente.



Ancora una volta la relazione ottenuta non è transitiva: $(a, b) \in R \cup (R; R) \cup (R; R; R)$ e $(b, d') \in R \cup (R; R) \cup (R; R; R)$ ma $(a, d') \notin R \cup (R; R) \cup (R; R; R)$.

Nel terzo tentativo si unisce a $R \cup (R; R) \cup (R; R; R)$ la relazione $R; R; R; R$. La relazione risultante $R \cup (R; R) \cup (R; R; R) \cup (R; R; R; R)$



è transitiva.

Osservazione 4.3.12. È interessante notare che, facendo un quarto tentativo, vale a dire aggiungendo alla relazione ottenuta la relazione $R; R; R; R$, la relazione risultante resta la stessa.

Si osservi che per la relazione R illustrata sopra sono stati necessari tre tentativi, ma è facile immaginare una relazione dove sono necessari più tentativi. Per esempio, per la relazione

$$a \longrightarrow b \longrightarrow c \longrightarrow d \longrightarrow e \longrightarrow f$$

occorrono quattro tentativi.

Come possiamo dare una ricetta generale per rendere una qualsiasi relazione R transitiva?

L'idea è di prendere una relazione ottenuta come unione infinita di tante relazioni:

$$R \cup (R; R) \cup (R; R; R) \cup (R; R; R; R) \cup \dots \quad (4.1)$$

Per essere del tutto formali, procediamo in due passi. Innanzitutto forniamo una definizione induttiva:

Definizione 4.3.13 (composizione n -aria di relazione). *Sia A un insieme e $R: A \leftrightarrow A$ una relazione su A . Per ogni numero naturale $n \in \mathbb{N}$ definiamo R^n induttivamente:*

1. CLAUSOLA BASE: $R^0 = id_A$
2. CLAUSOLA INDUTTIVA: $R^{n+1} = R; R^n$

4. Relazioni su un insieme

Cerchiamo di capire la definizione di sopra: la relazione R^n è la composizione di R con se stessa n volte. Per esempio R^4 può essere calcolata come

$$\begin{aligned} R^4 &= R; R^3 \\ &= R; (R; R^2) \\ &= R; (R; (R; R^1)) \\ &= R; (R; (R; (R; R^0))) \\ &= R; (R; (R; (R; id_A))) \end{aligned}$$

che, per le leggi di associatività ed unità, è esattamente $R; R; R; R$.

Adesso per ottenere la relazione in (4.1) è sufficiente fare l'unione infinita degli R^n per tutti gli $n = 1, 2, \dots$

$$\bigcup_{n \in \mathbb{N}^+} R^n$$

Definizione 4.3.14 (chiusura transitiva). *Sia $R: A \leftrightarrow A$ una relazione su A . La CHIUSURA TRANSITIVA DI R , denotata R^+ , è la relazione definita come*

$$R^+ = \bigcup_{n \in \mathbb{N}^+} R^n$$

Esempio 4.3.15. Calcoliamo la chiusura transitiva di id_A , $\emptyset: A \leftrightarrow A$ e $A \times A: A \leftrightarrow A$ per un qualsiasi insieme A .

Per id_A si ha che

$$\begin{aligned} id_A^1 &= id_A; id_A^0 \\ &= id_A; id_A \\ &= id_A. \end{aligned}$$

Inoltre

$$\begin{aligned} id_A^2 &= id_A; id_A^1 \\ &= id_A; id_A \\ &= id_A. \end{aligned}$$

Similmente, $id_A^3 = id_A; id_A^2 = id_A; id_A = id_A$. Utilizzando il Principio di Induzione è facile dimostrare che $id_A^n = id_A$ per ogni $n \in \mathbb{N}^+$. Pertanto

$$id_A^+ = \bigcup_{n \in \mathbb{N}^+} id_A^n = id_A.$$

Per $\emptyset: A \leftrightarrow A$ e $A \times A: A \leftrightarrow A$, possiamo convincerci attraverso un simile ragionamento che $\emptyset^n = \emptyset$ e $(A \times A)^n = A \times A$ per tutti gli $n \in \mathbb{N}^+$. Pertanto

$$\emptyset^+ = \bigcup_{n \in \mathbb{N}^+} \emptyset^n = \emptyset$$

$$(A \times A)^+ = \bigcup_{n \in \mathbb{N}^+} (A \times A)^n = A \times A.$$

Osservazione 4.3.16. È importante fare attenzione a non confondere la notazione R^n per una qualche relazione R con A^n per un insieme A . La prima, data nella Definizione 4.3.13, denota la relazione ottenuta componendo R con se stessa n volte. La seconda, data nella Definizione 2.7.2, denota l'insieme delle sequenze su A di lunghezza n .

Esempio 4.3.17. Si consideri la relazione $R = \{(a, b), (b, c), (a, d)\}$ sull'insieme $A = \{a, b, c, d\}$. Per calcolare R^+ è conveniente analizzare le relazioni R^n per $n \in \mathbb{N}^+$. Si ha che:

$$\begin{array}{lll} R^1 &= R; R^0 & R^2 = R; R^1 \\ &= R; id_A & = R; R \\ &= R & = \{(a, c)\} \\ & & = \emptyset \end{array}$$

Visto che $R^3 = \emptyset$, si ha che $R^4 = R; R^3 = R; \emptyset = \emptyset$ e, più in generale, $R^n = \emptyset$ per tutti gli $n \geq 3$. Pertanto

$$R^+ = \bigcup_{n \in \mathbb{N}^+} R^n = R^1 \cup R^2 \cup \emptyset \cup \emptyset \cup \dots \cup \emptyset \cup \dots$$

e quindi

$$R^+ = \{(a, b), (b, c), (a, d), (a, c)\}.$$

Aggiungendo una sola coppia alla relazione R il calcolo di R^+ può essere molto più complicato, come vediamo nel seguente esempio.

Esempio 4.3.18. Si consideri la relazione $R = \{(a, b), (b, c), (a, d), (c, a)\}$ sull'insieme $A = \{a, b, c, d\}$. Per calcolare R^+ , è conveniente analizzare le relazioni R^n per $n \in \mathbb{N}^+$. Si ha che $R^1 = R; R^0 = R; id_A = R$. Inoltre,

$$\begin{aligned} R^2 &= R; R^1 \\ &= R; R \\ &= \{(a, c), (b, a), (c, b), (c, d)\} \\ \\ R^3 &= R; R^2 \\ &= R; \{(a, c), (b, a), (c, b), (c, d)\} \\ &= \{(a, a), (b, b), (c, c), (b, d)\} \end{aligned}$$

È importante osservare che R^4 risulta essere inclusa nella relazione di partenza R .

$$\begin{aligned} R^4 &= R; R^3 \\ &= R; \{(a, a), (b, b), (c, c), (b, d)\} \\ &\subseteq R; (id_A \cup \{(b, d)\}) \\ &= (R; id_A) \cup (R; \{(b, d)\}) \\ &= R \cup \{(a, d)\} \\ &= R \end{aligned}$$

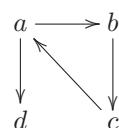
A questo punto non dobbiamo calcolare nuove relazioni perché ognuno degli R^n risulterà essere incluso o in R o in R^2 o in R^3 . Per esempio, $R^5 = R; R^4 \subseteq R; R = R^2, R^6 = R; R^5 \subseteq R; R^2 = R^3$ e $R^7 = R; R^6 \subseteq R; R^3 = R$. Pertanto

$$\begin{aligned} R^+ &= \bigcup_{n \in \mathbb{N}^+} R^n = R^1 \cup R^2 \cup R^3 \cup R^4 \cup R^5 \cup R^6 \cup R^7 \cup \dots \\ &\subseteq R^1 \cup R^2 \cup R^3 \cup R^1 \cup R^2 \cup R^3 \cup R^1 \cup \dots \\ &= R_1 \cup R_2 \cup R_3 \end{aligned}$$

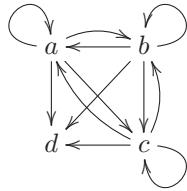
e quindi $R^+ =$

$$\{(a, b), (b, c), (a, d), (c, a), (a, c), (b, a), (c, b), (c, d), (a, a), (b, b), (c, c), (b, d)\}$$

Un modo efficace per semplificare il calcolo di R^+ consiste nel visualizzare la relazione R come un grafo ed aggiungere un arco da un nodo x ad un nodo y ogni volta che c'è un "percorso" da x a y . Qua, per "percorso", si intende una sequenza (di lunghezza arbitraria) di archi. Per esempio la relazione R dell'Esempio 4.3.18 è disegnata come il seguente grafo.



Aggiungendo un arco per ogni percorso, si ottiene la relazione R^+ .



Proposizione 4.3.19. Per tutti gli insiemi A e per tutte le relazioni $R: A \leftrightarrow A$, vale che

- (1) R^+ è transitiva;
- (2) $R \subseteq R^+$;
- (3) per tutte le relazioni $S: A \leftrightarrow A$, se $R \subseteq S$ e S è transitiva, allora $R^+ \subseteq S$.

Analogamente alla chiusura riflessiva e simmetrica, la chiusura transitiva di R è la più “piccola” relazione transitiva che contiene R .

Esercizio 4.3.20. Dimostrare la Proposizione 4.3.19.

Esercizio 4.3.21. Si ricordi la relazione Genitore: $U \leftrightarrow U$. Definire per proprietà (in modo intensionale) la relazione Genitore⁺.

Esercizio 4.3.22. Si ricordi la relazione succ: $\mathbb{N} \rightarrow \mathbb{N}$. Definire per proprietà (in modo intensionale) la relazione succ⁺.

Esercizio 4.3.23. Dimostrare che per tutti gli insiemi A e per tutte le relazioni $R: A \leftrightarrow A$, vale che

$$\text{se } R \text{ è transitiva, allora } R^+ = R.$$

Suggerimento: utilizzare la Proposizione 4.3.19.

Esercizio 4.3.24. Si ricordi la relazione minore $<: \mathbb{N} \leftrightarrow \mathbb{N}$. Definire per proprietà (in modo intensionale) la relazione $<^+$.

Stella di Kleene

Nella definizione di R^+ si prende l'unione di tutti gli R^n per $n \in \mathbb{N}^+$. Cambiando la definizione in modo da prendere $n \in \mathbb{N}$, si ottiene la relazione $R^0 \cup R^+ = id_A \cup R^+$ che chiamiamo chiusura riflessiva e transitiva.

Definizione 4.3.25 (chiusura riflessiva e transitiva). Sia $R: A \leftrightarrow A$ una relazione su A . La CHIUSURA RIFLESSIVA E TRANSITIVA DI R , denotata R^* , è la relazione definita come

$$R^* = \bigcup_{n \in \mathbb{N}} R^n$$

Osservazione 4.3.26. È interessante notare che la notazione \cdot^* era già stata utilizzata nella Definizione 2.7.8 per A^* , l'insieme di sequenze di lunghezza arbitraria su un insieme A . Chiaramente A^* , per un insieme A , e R^* , per una relazione R , sono due concetti ben distinti. Ma allora perché utilizzare la stessa notazione? La ragione è che le due costruzioni sono molto simili: in entrambe, si fa l'unione per tutti gli $n \in \mathbb{N}$ di un qualche esponente n -esimo.

Proposizione 4.3.27. Per tutti gli insiemi A e per tutte le relazioni $R: A \leftrightarrow A$, vale che

- R^* è riflessiva e transitiva;
- $R \subseteq R^*$;
- per tutte le relazioni $S: A \leftrightarrow A$, se $R \subseteq S$ e S è riflessiva e transitiva, allora $R^* \subseteq S$.

In altre parole, la chiusura riflessiva e transitiva di R è la più “piccola” relazione riflessiva e transitiva che contiene R .

Esercizio 4.3.28. Si ricordi la relazione $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}$. Definire per proprietà (in modo intensionale) la relazione succ^* .

Esercizio 4.3.29. Si ricordi la relazione minore $<: \mathbb{N} \leftrightarrow \mathbb{N}$. Definire per proprietà (in modo intensionale) la relazione $<^*$.

La stella di Kleene ricopre un ruolo importante in Informatica in quanto R^* è spesso pensata come una sorta di *iterazione illimitata* di R . Per rendere questa intuizione più chiara è conveniente mostrare un esempio

Esempio 4.3.30. Consideriamo un banalissimo programma while:

`while (x ≥ 2) do {x := x - 2}`

Tale programma prende in input un qualsiasi numero intero $x \in \mathbb{Z}$ e restituisce x se $x < 2$, 0 se x è pari e 1 se x è dispari. Ad esempio, partendo con $x = 6$, dopo una prima iterazione x varrà 4, poi 2 e poi 0. Con $n = 0$, il corpo del ciclo non viene eseguito perché la condizione della guardia $x \geq 2$ non è soddisfatta. Partendo con $x = 7$, nelle iterazioni successive x varrà 5, poi 3 e poi infine 1. Partendo con $x = -1$, la guardia del while è falsa e quindi non viene eseguita nessuna iterazione del ciclo while.

Il comportamento di tale programma può essere espresso utilizzando la stella di Kleene. Per rappresentare il corpo del ciclo while (il comando $x := x - 2$) definiamo la relazione $C = \{(x, x - 2) \mid x \in \mathbb{Z}\}: \mathbb{Z} \leftrightarrow \mathbb{Z}$. Per rappresentare la condizione di guardia ($x \geq 2$), definiamo la relazione $G = \{(x, x) \mid x \in \mathbb{Z} \text{ e } x \geq 2\}: \mathbb{Z} \leftrightarrow \mathbb{Z}$ (si osservi che $G \subseteq \text{Id}_{\mathbb{Z}}$, quindi G è la relazione di identità parziale definita solo per i valori che soddisfano la guardia). Inoltre definiamo \tilde{G} come $\tilde{G} = \text{Id}_{\mathbb{Z}} \setminus G$, quindi $\tilde{G} = \{(x, x) \mid x \in \mathbb{Z} \text{ e } x < 2\}$.

Adesso il “comportamento” del banale programma di sopra può essere espresso come la relazione

$$W = (G; C)^*; \tilde{G}. \quad (4.2)$$

Mostriamo che per ogni coppia $(a, b) \in \mathbb{Z} \times \mathbb{Z}$, $(a, b) \in W$ se e solo se il programma while eseguito a partire da uno stato in cui $x = a$ termina in uno stato in cui $x = b$.

Dalla definizione della stella di Kleene e usando la distributività della composizione rispetto all'unione (Tabella 2.6) abbiamo che

$$W = (\text{Id}_{\mathbb{Z}} \cup (G; C)^+); \tilde{G} = (\text{Id}_{\mathbb{Z}}; \tilde{G}) \cup ((G; C)^+; \tilde{G}) = \tilde{G} \cup ((G; C)^+; \tilde{G})$$

Quindi per ogni valore $a < 2$ vale $(a, a) \in \tilde{G} \subseteq W$, come desiderato, perché per tali valori il ciclo non viene mai eseguito e x mantiene il suo valore iniziale.

Adesso per ogni $n \in \mathbb{N}^+$ è facile convincersi che $(G; C)^n$ conterrà tutte e sole le coppie (a, b) dove a è il valore iniziale di x e b è il valore di x dopo l' n -esima iterazione. Ad esempio

$$\begin{array}{ll} (6, 4) \in (G; C)^1 & (7, 5) \in (G; C)^1 \\ (6, 2) \in (G; C)^2 & (7, 3) \in (G; C)^2 \\ (6, 0) \in (G; C)^3 & (7, 1) \in (G; C)^3 \end{array}$$

Si osservi che $(6, -2) \notin (G; C)^4$ in quanto $(0, -2) \notin G; C$. Di conseguenza per $n > 3$, non esiste alcuna coppia $(a, b) \in (G; C)^n$ dove il primo elemento a è 6. Pertanto le uniche coppie in $(G; C)^+$ aventi come primo elemento 6 sono esattamente quelle elencate sopra. Lo stesso vale per 7.

Per concludere, ricordando che $W = (G; C)^*; \tilde{G}$, osserviamo che di tutte le coppie in $(G; C)^+$ aventi 6 come primo elemento, l'unica che contribuisce alla relazione W è $(6, 0)$, perché $(0, 0) \in \tilde{G}$ mentre $(b, b) \notin \tilde{G}$ per $b \in \{2, 4\}$. Analogamente $(7, 1) \in W$ e $(7, b) \notin W$ per $b \neq 1$. Poiché lo stesso ragionamento può essere fatto per ogni numero pari o dispari maggiore o uguale a 2, questo conclude la dimostrazione.

Osservazione 4.3.31. Il lettore potrebbe adesso pensare che ogni programma while può essere espresso come la relazione in (4.2). È quindi opportuno precisare che l'espressione in (4.2) è solo un esempio ad hoc, e non vuole rappresentare una ricetta generale per esprimere programmi while utilizzando la stella di Kleene e le operazioni su relazioni. Questo infatti è un problema abbastanza complesso che non può essere affrontato in questo corso introduttivo. Lo studente interessato potrà affrontare questo tipo di problematiche in corsi avanzati.

4. Relazioni su un insieme

Vista l'importanza della stella di Kleene, è opportuno studiare le leggi che la regolano.

Teorema 4.3.32. *Per tutti gli insiemi A e relazioni $R: A \leftrightarrow A$ e $S: A \leftrightarrow A$ valgono le uguaglianze nella Tabella 4.1.*

riflessività	$id_A \subseteq R^*$
transitività	$R^*; R^* \subseteq R^*$
chiusura	$R \subseteq R^*$
idempotenza	$(R^*)^* = R^*$
\star -id	$id_A^* = id_A$
\star -compl	$(A \times A)^* = A \times A$
\star -vuoto	$\emptyset_{A,A}^* = id_A$
distributività di \star su \cup	$R^* \cup S^* \subseteq (R \cup S)^*$
distributività di \star su \cap	$(R \cap S)^* \subseteq R^* \cap S^*$
distributività di \star su \cdot^{op}	$(R^*)^{op} = (R^{op})^*$

Tabella 4.1: Leggi della stella di Kleene.

Le prime tre leggi (riflessività, transitività e chiusura) seguono immediatamente dalla Proposizione 4.3.27. Anche l'idempotenza segue dalla Proposizione 4.3.27 ma la dimostrazione richiede qualche passaggio.

Esercizio 4.3.33. *Dimostrare in maniera discorsiva la legge idempotenza nella Tabella 4.1.*

Per le leggi \star -id, \star -compl e \star -vuoto è sufficiente calcolare la chiusura riflessiva e transitiva di id_A , \emptyset e $A \times A$.

Esercizio 4.3.34. *Calcolare id_A^* , \emptyset^* e $(A \times A)^*$, per un arbitrario insieme A .*

Le tre leggi di distributività alla fine della Tabella 4.1 richiedono invece l'utilizzo del Principio di Induzione sui naturali che abbiamo introdotto nel Capitolo 3. Vediamo in particolare la distributività di \star su \cup e la distributività di \star su \cdot^{op} .

Esempio 4.3.35 (Distributività della stella di Kleene sull'unione di relazioni). *Dimostriamo che $R^* \cup S^* \subseteq (R \cup S)^*$.*

Per cominciare, dimostriamo per induzione che

$$\text{Se } R \subseteq S, \text{ allora } R^n \subseteq S^n \text{ per ogni } n \in \mathbb{N} \quad (4.3)$$

Infatti per $n = 0$ (caso base) abbiamo $R^0 = Id_A = S^0$. Invece per $n = m + 1$ (passo induttivo) abbiamo

$$\begin{aligned} R^n &= R; R^m \\ &\subseteq S; R^m && (\text{Ipotesi: } R \subseteq S) \\ &\subseteq S; S^m && (\text{Ipotesi induttiva: } R^m \subseteq S^m) \\ &= S^n \end{aligned}$$

Mostriamo ora che

$$\text{Se } R \subseteq S, \text{ allora } R^* \subseteq S^* \quad (4.4)$$

Infatti abbiamo:

$$R^* = \bigcup_{n \in \mathbb{N}} R^n \quad (\text{Definizione 4.3.25}) \subseteq \bigcup_{n \in \mathbb{N}} S^n \quad (\text{per (4.3)}) = S^*$$

Infine dimostriamo che $R^* \cup S^* \subseteq (R \cup S)^*$. Infatti abbiamo

- $R \subseteq (R \cup S)$, e quindi $R^* \subseteq (R \cup S)^*$ (per (4.4))

- $S \subseteq (R \cup S)$, e quindi $S^* \subseteq (R \cup S)^*$ (per (4.4))
- e quindi $R^* \cup S^* \subseteq (R \cup S)^*$ (per proprietà dell'unione)

Esempio 4.3.36 (Distributività della stella di Kleene sull'opposto di una relazione). *Dimostriamo che $(R^*)^{op} = (R^{op})^*$*

Vediamo prima che

$$R; R^n = R^n; R \quad \text{per ogni } n \in \mathbb{N} \quad (4.5)$$

Infatti se $n = 0$ (caso base) abbiamo $R; R^0 = R; Id_A = R = Id_A; R = R^0; R$.

Per $n = m + 1$ (passo induttivo) abbiamo

$$\begin{aligned} R; R^{m+1} &= R; (R; R^m) \\ &= R; (R^m; R) \quad (\text{Ipotesi induttiva}) \\ &= (R; R^m); R \quad (\text{associatività}) \\ &= R^{m+1}; R \end{aligned}$$

Dimostriamo ora per induzione che

$$(R^n)^{op} = (R^{op})^n \quad \text{per ogni } n \in \mathbb{N} \quad (4.6)$$

Infatti se $n = 0$ (caso base) abbiamo $(R^0)^{op} = (Id_A)^{op} = Id_A = (R^{op})^0$.

Per $n = m + 1$ (passo induttivo) abbiamo

$$\begin{aligned} (R^n)^{op} &= (R^{m+1})^{op} \\ &= (R; R^m)^{op} \\ &= (R^m)^{op}; R^{op} \quad (\text{Distrib. di } .^{op}) \\ &= (R^{op})^m; R^{op} \quad (\text{Ipotesi induttiva}) \\ &= R^{op}; (R^{op})^m \quad (\text{per (4.5)}) \\ &= R^{op}; (R^{op})^{m+1} = (R^{op})^n \end{aligned}$$

Infine possiamo concludere osservando che

$$(R^*)^{op} = \left(\bigcup_{n \in \mathbb{N}} R^n \right)^{op} \quad (\text{Def. 4.3.25}) = \bigcup_{n \in \mathbb{N}} (R^n)^{op} \quad (\text{Dist. di } .^{op} \text{ su } \cup) = \bigcup_{n \in \mathbb{N}} (R^{op})^n \quad (\text{per (4.6)}) = (R^{op})^*$$

4.4 Relazioni di equivalenza

Definizione 4.4.1 (relazione di equivalenza). *Sia $R: A \leftrightarrow A$ una relazione su A . Si dice che R è una RELAZIONE DI EQUIVALENZA se è riflessiva, transitiva e simmetrica.*

Esempio 4.4.2. Per ogni insieme A , l'identità $id_A: A \leftrightarrow A$ e la relazione completa $A \times A: A \leftrightarrow A$ sono relazioni di equivalenza. La relazione vuota $\emptyset: A \leftrightarrow A$ in generale non è una relazione di equivalenza perché non è riflessiva.

Esempio 4.4.3. Le relazioni $<: \mathbb{Z} \leftrightarrow \mathbb{Z}$ e $\leq: \mathbb{Z} \leftrightarrow \mathbb{Z}$ non sono relazioni di equivalenza, perché non sono simmetriche.

Esempio 4.4.4. Per $A = \{a, b, c, d\}$, la relazione $R = \{(a, a), (a, b), (b, a), (c, c)\}$ non è una relazione di equivalenza perché non è riflessiva mentre la relazione $S = \{(a, a), (b, b), (a, b), (b, a), (c, c), (d, d)\}$ è una relazione di equivalenza.

Le relazioni di parentela sugli esseri umani solitamente non sono relazioni di equivalenza: queste relazioni molto spesso non sono né simmetriche né transitive (per esempio *Madre*, *Padre*, *Zia*). La relazione *Fratello*: $U \leftrightarrow U$ è sia simmetrica che transitiva, ma non è una relazione di equivalenza perché non è riflessiva: un essere umano non è solitamente considerato fratello di se stesso. Si può comunque definire una relazione di equivalenza molto simile.

Esempio 4.4.5. La relazione

$$StessaMadre = \{(x, y) \in U \times U \mid x \text{ e } y \text{ hanno la stessa madre}\}$$

è una relazione di equivalenza poiché (a) x e x hanno la stessa madre; (b) se x e y hanno la stessa madre e y e z hanno la stessa madre, allora anche x e z hanno la stessa madre; (c) se la madre di x è la madre di y , allora anche la madre di y è la madre di x .

Esercizio 4.4.6. Dimostrare che le seguenti relazioni sono equivalenze.

$$\begin{aligned} StessoCogn &= \{(x, y) \in U \times U \mid x \text{ e } y \text{ hanno lo stesso cognome}\} \\ StessoCompl &= \{(x, y) \in U \times U \mid x \text{ e } y \text{ hanno lo stesso compleanno}\} \\ StessiCapelli &= \{(x, y) \in U \times U \mid x \text{ e } y \text{ hanno lo stesso numero di capelli}\} \end{aligned}$$

Teorema di caratterizzazione

Le relazioni nell’Esempio 4.4.5 e nell’Esercizio 4.4.6 seguono uno schema comune:

Definizione 4.4.7 (kernel di una funzione). Sia $f: A \rightarrow B$ una funzione. La relazione KERNEL DI f è definita come

$$Ker(f) = \{(x, y) \in A \times A \mid f(x) = f(y)\}.$$

Esempio 4.4.8. Si consideri la funzione capelli: $U \rightarrow \mathbb{N}$ che associa ad ogni essere umano il suo numero di capelli. $Ker(\text{capelli})$ è la relazione StessiCapelli dell’Esercizio 4.4.6.

Proposizione 4.4.9. Per tutti gli insiemi A, B e per tutte le funzioni $f: A \rightarrow B$ vale che

1. $Ker(f) = f; f^{op}$;
2. $Ker(f)$ è una relazione di equivalenza.

Dimostrazione. Per il primo punto si osservi che:

$$\begin{aligned} &(x, y) \in f; f^{op} \\ \text{se e solo se } &\exists b \in B . (x, b) \in f \wedge (b, y) \in f^{op} && (\text{def di } ;) \\ \text{se e solo se } &\exists b \in B . (x, b) \in f \wedge (y, b) \in f && (\text{def di } f^{op}) \\ \text{se e solo se } &f(x) = f(y) && (f \text{ funzione}) \\ \text{se e solo se } &(x, y) \in Ker(f) && (\text{def. } Ker(f)) \end{aligned}$$

Per il secondo punto, si dimostrano separatamente le tre proprietà per $f; f^{op}$.

- Riflessiva: $id_A \subseteq f; f^{op}$ segue subito dal fatto che f è totale e dal Teorema 2.4.34.1.
- Transitiva: $(f; f^{op}); (f; f^{op}) = f; (f^{op}; f); f^{op} \subseteq f; f^{op}$ segue subito dal fatto che f è univalente e dal Teorema 2.4.34.2.
- Simmetrica: $f; f^{op} = (f; f^{op})^{op}$ segue subito dalle leggi di distributività e convoluzione di $.^{op}$.

■

La Proposizione 4.4.9 può essere trasformata in un teorema di caratterizzazione: tutte le relazioni di equivalenza sono il kernel di una qualche funzione. Per dimostrare questo fatto usiamo la nozione di classe di equivalenza.

Definizione 4.4.10 (classe di equivalenza). Sia $R: A \leftrightarrow A$ una relazione di equivalenza ed $a \in A$ un elemento di A . La CLASSE DI R -EQUIVALENZA DI a è definita come

$$[a]_R = \{b \in A \mid (a, b) \in R\}$$

Esempio 4.4.11. Sia S la relazione di equivalenza dell’Esempio 4.4.4. Si ha che $[a]_S = \{a, b\}$; $[b]_S = \{a, b\}$; $[c]_S = \{c\}$ e $[d]_S = \{d\}$.

Esempio 4.4.12. Per ogni insieme A si considerino le relazioni di equivalenza $\text{id}_A: A \leftrightarrow A$ e $A \times A: A \leftrightarrow A$. Per tutti gli $a \in A$, vale che

$$\begin{aligned} [a]_{\text{id}_A} &= \{a\} \\ [a]_{A \times A} &= A \end{aligned}$$

Proposizione 4.4.13. Per tutti gli insiemi A , per tutte le relazioni di equivalenza $R: A \leftrightarrow A$, e per tutti gli elementi $a, b \in A$, le seguenti affermazioni sono equivalenti:

1. $(a, b) \in R$
2. $[a]_R = [b]_R$
3. $[a]_R \cap [b]_R \neq \emptyset$

Dimostrazione. La dimostrazione procede dimostrando che:

- L'affermazione 1 implica la 2:

Assumiamo che $(a, b) \in R$ e dimostriamo le due inclusioni $[a]_R \subseteq [b]_R$ e $[a]_R \supseteq [b]_R$. Facciamo vedere solo la prima, la seconda è del tutto analoga. Preso un qualsiasi elemento $c \in [a]_R$ dobbiamo far vedere che $c \in [b]_R$. Per definizione di classe di equivalenza sappiamo che $b \in [a]_R$. Dato che $c \in [a]_R$ si ha $(a, c) \in R$. Per la proprietà simmetrica delle relazioni di equivalenza si ha $(b, a) \in R$. Ma allora per la transitività deve valere $(b, c) \in R$ e quindi $c \in [b]_R$.

- La 2 implica la 3:

Assumendo che $[a]_R = [b]_R$, per dimostrare che $[a]_R \cap [b]_R \neq \emptyset$ basta far vedere che $[a]_R \neq \emptyset$. Infatti, la proprietà riflessiva garantisce che $a \in [a]_R$.

- La 3 implica la 1:

Assumiamo che $[a]_R \cap [b]_R \neq \emptyset$. Quindi esiste almeno un elemento $c \in [a]_R \cap [b]_R$, cioè tale che $(a, c) \in R$ e $(b, c) \in R$. Ma per la proprietà simmetrica si ha $(c, b) \in R$ e dunque per la transitività si ha che $(a, b) \in R$.

■

Teorema 4.4.14. Per tutti gli insiemi A e per tutte le relazioni $R: A \leftrightarrow A$ vale che R è una relazione di equivalenza se e solo se esiste un insieme B ed una funzione $f: A \rightarrow B$ tale che $R = \text{Ker}(f)$.

Dimostrazione. Trattandosi di un se e solo se, possiamo dividere la dimostrazione in due parti.

- Se R è una relazione di equivalenza, allora esiste un insieme B ed una funzione $f: A \rightarrow B$ tale che $R = \text{Ker}(f)$.

Prendiamo come B l'insieme di tutte le classi di R -equivalenza e come funzione $f: A \rightarrow B$ la funzione che associa ad ogni elemento $a \in A$ la sua classe di R -equivalenza, cioè

$$a \mapsto [a]_R$$

Dobbiamo dimostrare adesso che $R = \text{Ker}(f)$: $(x, y) \in \text{Ker}(f)$ se e solo se $f(x) = f(y)$ (per definizione di Ker) se e solo se $[x]_R = [y]_R$ (per definizione di f) se e solo se $(x, y) \in R$ (per la Proposizione 4.4.13).

- Se esiste un insieme B ed una funzione $f: A \rightarrow B$ (per un qualche insieme B) tale che $R = \text{Ker}(f)$, allora R è una relazione di equivalenza.

Questo segue immediatamente dalla Proposizione 4.4.9.2.

■

Relazioni di equivalenza e partizioni

Data una relazione di equivalenza $R: A \leftrightarrow A$ si può considerare l'insieme delle classi di R -equivalenza

$$EC_R = \{[a]_R \mid a \in A\}.$$

Essendo ogni $[a]_R$ un insieme EC_R è un insieme di insiemi. Visto che gli elementi non vengono ripetuti negli insiemi, se $[a]_R = [b]_R$ allora $[a]_R$ e $[b]_R$ appariranno una volta sola in EC_R .

Esempio 4.4.15. Sia S la relazione di equivalenza dell'Esempio 4.4.4. Si ha che $EC_S = \{\{a, b\}, \{c\}, \{d\}\}$.

È importante notare che EC_R forma una *partizione* dell'insieme A (Definizione 1.5.9).

Proposizione 4.4.16. Per tutti gli insiemi A e per tutte le relazioni d'equivalenza $R: A \leftrightarrow A$, EC_R è una partizione.

Dimostrazione. Per dimostrare che EC_R forma una partizione si deve dimostrare che

1. Ogni insieme $X \in EC_R$ è diverso da \emptyset (insiemi non vuoti);
2. $\bigcup_{X \in EC_R} X = A$ (copertura di A);
3. Per tutti gli $X, Y \in EC_R$, se $X \neq Y$, allora $X \cap Y = \emptyset$ (insiemi disgiunti);

Per dimostrare il primo punto, si osservi che per ogni $[a]_R \in EC_R$, vale che $a \in [a]_R$ e quindi $[a]_R \neq \emptyset$.

Per dimostrare il secondo punto, si osservi che per ogni $a \in A$, vale che $a \in [a]_R$ e $[a]_R \in EC_R$. Pertanto $a \in \bigcup_{X \in EC_R} X$.

Per il terzo punto, dalla Proposizione 4.4.13, si ha che $[a]_R \neq [b]_R$ se e solo se $[a]_R \cap [b]_R = \emptyset$. ■

Viceversa, data una partizione dell'insieme A si può definire una relazione di equivalenza su A . Per mostrare questa costruzione procediamo per passi: prima si definisce una funzione e poi prendiamo il suo kernel.

Data una partizione $\mathcal{F} = \{X_i\}_{i \in I}$ dell'insieme A , si definisce la relazione $f_{\mathcal{F}}: A \leftrightarrow I$ come

$$f_{\mathcal{F}} = \{(a, i) \in A \times I \mid a \in X_i\}$$

Proposizione 4.4.17. Per tutti gli insiemi A e per tutte le partizioni $\mathcal{F} = \{X_i\}_{i \in I}$ di A , la relazione $f_{\mathcal{F}}$ è una funzione.

Dimostrazione. Bisogna dimostrare che $f_{\mathcal{F}}$ è totale e univalente.

- totale: visto che $\mathcal{F} = \{X_i\}_{i \in I}$ è una partizione di A , vale che $A \subseteq \bigcup_{X \in EC_R} X$ e quindi per ogni $a \in A$ esiste almeno un X_i tale che $a \in X_i$ e quindi, per definizione di $f_{\mathcal{F}}$, si ha che $(a, i) \in f_{\mathcal{F}}$.
- univalente: visto che $\mathcal{F} = \{X_i\}_{i \in I}$ è una partizione di A , vale che se $i \neq j$, allora $X_i \cap X_j = \emptyset$. Pertanto per ogni $a \in A$, esiste al più un $i \in I$, tale che $a \in X_i$, cioè esiste al più un $i \in I$, tale che $(a, i) \in f_{\mathcal{F}}$.

■

Adesso, la relazione di equivalenza corrispondente ad \mathcal{F} è esattamente il kernel di $f_{\mathcal{F}}$.

Le corrispondenze che abbiamo illustrato definiscono una biiezione tra l'insieme delle relazioni di equivalenza su A (denotato da $ERel(A)$) e l'insieme delle partizioni su A (denotato da $Part(A)$).

Teorema 4.4.18. Per tutti gli insiemi A , vale che

$$ERel(A) \cong Part(A).$$

Dimostrazione. Definiamo $i: ERel(A) \rightarrow Part(A)$ come la funzione che mappa una relazione di equivalenza R nella partizione EC_R

$$\begin{aligned} i: & ERel(A) \rightarrow Part(A) \\ R & \mapsto EC_R \end{aligned}$$

Definiamo $j: Part(A) \rightarrow ERel(A)$ come la funzione che mappa una partizione \mathcal{F} nella relazione di equivalenza $Ker(f_{\mathcal{F}})$

$$\begin{aligned} j: & Part(A) \rightarrow ERel(A) \\ \mathcal{F} & \mapsto Ker(f_{\mathcal{F}}) \end{aligned}$$

Dobbiamo dimostrare che $i; j = id_{ERel(A)}$ e $j; i = id_{Part(A)}$.

- $i; j = id_{ERel(A)}$. Dobbiamo dimostrare che per ogni relazione di equivalenza $R: A \leftrightarrow A$, $j(i(R)) = R$.

Si osserva che $(x, y) \in j(i(R))$ se e solo se $(x, y) \in Ker(f_{EC_R})$ se e solo se $f_{EC_R}(x) = f_{EC_R}(y)$ se e solo se $[x]_R = [y]_R$ se e solo se $(x, y) \in R$.

- $j; i = id_{Part(A)}$. Dobbiamo dimostrare che per ogni partizione \mathcal{F} , $i(j(\mathcal{F})) = \mathcal{F}$.

Si osserva che $X \in i(j(\mathcal{F}))$ se e solo se $X \in EC_{Ker(f_{\mathcal{F}})}$ se e solo se esiste un $a \in A$, tale che $X = [a]_{Ker(f_{\mathcal{F}})}$, se e solo se esiste un $a \in A$ tale che $X = [a]_{Ker(f_{\mathcal{F}})}$, se e solo se esiste un $a \in A$ tale che $X = \{b \in A \mid f_{\mathcal{F}}(b) = f_{\mathcal{F}}(a)\}$, se e solo se esiste un $X_i \in \mathcal{F}$ tale che $X = X_i$.

■

4.5 Relazioni di Ordinamento

Definizione 4.5.1 (ordinamento parziale). *Sia $R: A \leftrightarrow A$ una relazione su A . Si dice che R è una RELAZIONE DI ORDINAMENTO PARZIALE se è riflessiva, transitiva ed anti-simmetrica.*

Esempio 4.5.2. *Per ogni insieme A , la relazione identità $id_A: A \leftrightarrow A$ è un ordinamento parziale. La relazione vuota $\emptyset: A \leftrightarrow A$ non è un ordinamento parziale perché non è riflessiva, e neanche la relazione completa $A \times A: A \leftrightarrow A$, perché non è anti-simmetrica*

Esempio 4.5.3. *La relazione $<: \mathbb{N} \leftrightarrow \mathbb{N}$ non è un ordinamento parziale perché non è riflessiva mentre la relazione $\leq: \mathbb{N} \leftrightarrow \mathbb{N}$ è un ordinamento parziale.*

Esempio 4.5.4. *Per $A = \{a, b, c, d\}$, la relazione $R = \{(a, b), (b, c), (a, d)\}$ non è un ordinamento parziale perché non è riflessiva e neppure transitiva, mentre la relazione $S = \{(a, b), (b, c), (a, c), (a, d), (a, a), (b, b), (c, c), (d, d)\}$ è un ordinamento parziale.*

Proposizione 4.5.5. *Per ogni insieme A , l'inclusione su $\mathcal{P}(A)$*

$$\{(X, Y) \in \mathcal{P}(A) \times \mathcal{P}(A) \mid X \subseteq Y\}$$

è una relazione di ordinamento parziale.

Esercizio 4.5.6. *Dimostrare la Proposizione 4.5.5.*

Per le relazioni di ordinamento parziale spesso si usa la notazione infissa:

si scrive $a R b$ per $(a, b) \in R$.

Inoltre, le relazioni di ordinamento sono genericamente denotate dal simbolo \sqsubseteq . Molto spesso si utilizza il simbolo \sqsubset per denotare la relazione

$$\sqsubset = \{(x, y) \mid x \sqsubseteq y \text{ e } x \neq y\}.$$

Si noti che questo è del tutto analogo alla notazione che si usa per le relazioni sui numeri *minore o uguale* (\leq) e *minore* ($<$).

C'è una differenza sostanziale tra l'ordinamento parziale \leq sui numeri e gli altri ordinamenti parziali che abbiamo visto fin'ora. In \leq vale che per ogni coppia di numeri $(n, m) \in \mathbb{N} \times \mathbb{N}$ o vale $n \leq m$ oppure vale $m \leq n$. Questa proprietà è importante in molte occasioni.

Definizione 4.5.7 (ordinamento). *Sia $R: A \leftrightarrow A$ una relazione di ordinamento parziale su A . Si dice che R è un ORDINAMENTO se*

per tutti gli $(a, b) \in A \times A$, vale che $(a, b) \in R$ oppure $(b, a) \in R$.

Esercizio 4.5.8. *Mostrare che la relazione di ordinamento parziale della Proposizione 4.5.5 non è un ordinamento per tutti gli insiemi A .*

Esercizio 4.5.9 (Teorema di Caratterizzazione). *Dimostrare che per tutti gli insiemi A e per tutte le relazioni $R: A \leftrightarrow A$, vale che*

$$\begin{aligned} R \text{ è un ordinamento se e solo se } id_A \subseteq R, \\ R; R \subseteq R, R \cap R^{op} \subseteq id_A \text{ e } A \times A \subseteq R \cup R^{op}. \end{aligned}$$

Esercizio 4.5.10. *Siano A e B due insiemi e $\sqsubseteq_A: A \leftrightarrow A$ e $\sqsubseteq_B: B \leftrightarrow B$ due relazioni di ordinamento parziale. Si consideri ora la relazione $\sqsubseteq_{A \times B}: A \times B \leftrightarrow A \times B$ definita come*

$$(a_1, b_1) \sqsubseteq_{A \times B} (a_2, b_2) \text{ se e solo se } a_1 \sqsubseteq_A a_2 \text{ e } b_1 \sqsubseteq_B b_2.$$

Dimostrare che $\sqsubseteq_{A \times B}$ è una relazione di ordinamento parziale. Questa relazione è un ordinamento? In caso affermativo, dare una dimostrazione; in caso negativo fornire un controesempio.

Esercizio 4.5.11. *Siano A e B due insiemi e $\sqsubseteq_A: A \leftrightarrow A$ e $\sqsubseteq_B: B \leftrightarrow B$ due ordinamenti. Si consideri ora la relazione $\sqsubseteq_{A \times B}: A \times B \leftrightarrow A \times B$ definita come*

$$(a_1, b_1) \sqsubseteq_{A \times B} (a_2, b_2) \text{ se e solo se } a_1 \sqsubseteq_A a_2 \text{ oppure } a_1 = a_2 \wedge b_1 \sqsubseteq_B b_2.$$

Dimostrare che $\sqsubseteq_{A \times B}$ è un ordinamento.

Ordinamento Lessicografico

Un esempio di ordinamento particolarmente importante è l'ordinamento lessicografico usato per ordinare le parole nei dizionari o negli elenchi.

Dato un insieme A ed un ordinamento $\sqsubseteq_A: A \leftrightarrow A$ si vuole definire un ordinamento $\sqsubseteq_{A^*}: A^* \leftrightarrow A^*$ sull'insieme delle stringhe su A .

L'idea è che l'ordinamento sul prodotto cartesiano visto nell'Esercizio 4.5.11 può essere esteso facilmente ad arbitrari insiemi A^n (l'insieme di sequenze su A di lunghezza n): date le sequenze $s = a_0a_1\dots a_n$ e $t = a'_0a'_1\dots a'_n$ in A^n si ha che

$$\begin{aligned} s \sqsubseteq_{A^n} t \text{ se e solo se esiste un } i \in \{0, \dots, n\} \text{ tale che} \\ \text{per tutti gli indici } j < i \text{ vale che } a_j = a'_j, \text{ ed } a_i \sqsubset a'_i. \end{aligned}$$

Per avere un ordinamento sull'insieme di sequenze di lunghezza arbitraria (A^*), si deve solamente tenere conto della possibilità che le sequenze abbiano lunghezza diversa.

Definizione 4.5.12 (ordinamento lessicografico). *Sia $\sqsubseteq_A: A \leftrightarrow A$ un ordinamento sull'insieme A . L'ORDINAMENTO LESSICOGRAFICO $\sqsubseteq_{A^*}: A^* \leftrightarrow A^*$ è definito come segue.*

Per tutte le stringhe $s = a_0a_1\dots a_n$ e $t = a'_0a'_1\dots a'_m$ in A^ si ha che $s \sqsubseteq_{A^*} t$ se e solo se $s = t$ oppure esiste un $i \in \mathbb{N}$ tale che per tutti $j < i$, vale che $a_j = a'_j$ ed almeno una delle due seguenti condizioni è vera:*

- $a_i \sqsubset a'_i$;
- $i = n + 1$ e $n < m$.

Osservazione 4.5.13. *Per chiarire la definizione può essere conveniente esprimerla attraverso la notazione matematica: $s \sqsubseteq_{A^*} t$ se e solo se*

$$s = t \vee (\exists i \in \mathbb{N}. (\forall j < i. a_j = a'_j) \wedge (a_i \sqsubset a'_i \vee (i = n + 1 \wedge n < m))).$$

Esempio 4.5.14. Si consideri l'insieme delle lettere dell'alfabeto italiano, ordinate nel modo classico:

$$a \leq b \leq \cdots \leq z$$

Vale che $\text{anna} \sqsubseteq \text{zio}$: nella definizione si prenda $i = 0$ e si osservi che $a \sqsubset z$; $\text{anna} \sqsubseteq \text{antonio}$: nella definizione si prenda $i = 2$ e si osservi che $\text{an} = \text{an} \sqsubset \text{t}$; $\text{anna} \sqsubseteq \text{annarella}$: nella definizione si prenda $i = 4$ e si osservi che $4 = 3 + 1$ e $3 < 8$.

Esercizio 4.5.15. Considerare l'insieme delle cifre decimali ordinate nel modo ovvio

$$0 \leq 1 \leq 2 \leq \cdots \leq 9.$$

È vero che l'ordinamento lessicografico su questo insieme coincide con il consueto \leq sui naturali? In caso affermativo, dare una dimostrazione; in caso negativo, fornire un controesempio.

CAPITOLO 5

Grafi

In questo capitolo presentiamo i grafi e molti concetti ad essi collegati. Introdotti nel XVIII secolo dal famoso matematico Eulero, l'importanza dei grafi nell'informatica (e non solo) deriva dal fatto che essi permettono di modellare in modo preciso e visualmente intuitivo relazioni tra elementi di un insieme, come rotte tra aeroporti, relazioni sociali, alberi genealogici e così via. Inoltre un gran numero di problemi computazionali possono essere formulati in modo preciso sfruttando una rappresentazione del dominio di interesse come grafo: si pensi per esempio al problema di trovare il percorso più breve tra due città, rappresentando la mappa stradale come un grafo.

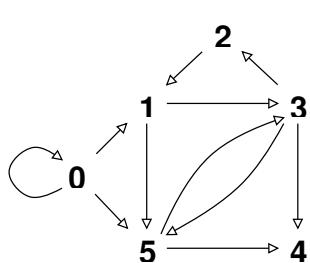
Prima di iniziare è bene precisare che esistono due importanti tipologie di grafi: i grafi *orientati* e i grafi *non orientati*. Al fine di enfatizzare il collegamento con le relazioni (viste nei capitoli precedenti), la nostra esposizione comincerà con i grafi orientati, per poi spostarsi sui grafi non orientati. Come vedremo molti concetti sono simili nei due casi, ma bisognerà stare attenti ad alcune sottili differenze.

5.1 Grafi orientati

Definizione 5.1.1 (grafo orientato). Un GRAFO ORIENTATO è una relazione $E: V \leftrightarrow V$ su un insieme finito V . Gli elementi di V vengono detti NODI o VERTICI e gli elementi di E vengono detti ARCHI o LATI. Di solito denoteremo un grafo con la lettera maiuscola G o sue varianti come G' , G_1 , G_2 , ... Per enfatizzare l'insieme dei nodi V e l'insieme degli archi E , solitamente si scrive $G = (V, E)$.

I grafi così definiti si chiamano *orientati* per enfatizzare il fatto che dato un arco $(x, y) \in E$, dove $x, y \in V$ sono due nodi distinti, possiamo assegnargli un verso o orientamento, dicendo che “parte da x ” e “arriva a y ”. Un arco del tipo $(x, x) \in E$, che parte e arriva nello stesso nodo, è chiamato CAPPIO o LOOP.

Dato un grafo $G = (V, E)$, da ora in avanti indichiamo con n il numero di nodi nel grafo (cioè $n = |V|$) e con m il suo numero di archi (cioè $m = |E|$). La DIMENSIONE di G è data dalla somma $|V| + |E|$ cioè $n + m$. Inoltre, salvo indicazioni contrarie, assumiamo che l'insieme dei nodi V sia esattamente l'insieme $V = \{0, 1, \dots, |V| - 1\}$.



(a)

	0	1	2	3	4	5
0	1	1				1
1				1		1
2		1				
3			1		1	1
4						
5				1	1	

(b)

0	→	0	1	5
1	→	3	5	
2	→	1		
3	→	2	4	5
4	→			
5	→	3	4	

(c)

Figura 5.1: (a) Un grafo orientato; (b) La sua matrice di adiacenza; (c) Le sue liste di adiacenza.

Esempio 5.1.2. Si consideri il grafo orientato $G = (V, E)$ disegnato nella Figura 5.1(a). Abbiamo $V = \{0, 1, 2, 3, 4, 5\}$, quindi ci sono $n = 6$ nodi (rappresentati dai numeri) e $m = 11$ archi, rappresentati da frecce. L'insieme degli archi è $E = \{(0, 0), (0, 1), (0, 5), (1, 3), (1, 5), (2, 1), (3, 2), (3, 4), (3, 5), (5, 3), (5, 4)\}$. Si noti che ci sono due archi diversi, con verso opposto, tra gli stessi nodi 3 e 5, e che il nodo 0 ha un cappio.

Osservazione 5.1.3. Il termine nodo deriva dall'idea che un grafo possa essere visto anche come una rete di corde, dove i nodi—ovvero i punti dove le corde si incontrano—rappresentano gli elementi dell'insieme, e gli archi sono segmenti di corda che rappresentano le relazioni tra nodi. Di fatti il termine rete viene a volte utilizzato come sinonimo di grafo: la rete (the web) è il nome dato all'enorme grafo di pagine multimediali a cui possiamo accedere tramite internet, e i cosiddetti social sono noti come reti sociali, enfatizzando l'interconnessione sociale tra le persone.

Vicinato e grado dei nodi

Nel seguito, quando parliamo di nodi e di archi assumiamo che essi appartengano a un grafo orientato $G = (V, E)$ arbitrario ma fissato.

Definizione 5.1.4 (vicinato, grado). Due nodi $x, y \in V$ sono ADIACENTI o VICINI se c'è un arco che li collega (quindi $(x, y) \in E$ o $(y, x) \in E$). Il VICINATO IN USCITA di un nodo $x \in V$ è l'insieme $N^+(x) = \{y \mid (x, y) \in E\}$, ed è chiamato anche STELLA USCENTE da x , perché formato dai suoi archi uscenti. Analogamente, il VICINATO IN INGRESSO di un nodo $x \in V$ è l'insieme $N^-(x) = \{y \mid (y, x) \in E\}$, ed è chiamato anche STELLA ENTRANTE in x . Il GRADO DI USCITA di x è definito come la cardinalità del suo vicinato di uscita, $d_x^+ = |N^+(x)|$, mentre il suo GRADO DI INGRESSO è definito come $d_x^- = |N^-(x)|$.¹

Proposizione 5.1.5. Per ogni grafo orientato $G = (V, E)$, vale che

$$\sum_{x \in V} d_x^- = \sum_{x \in V} d_x^+ = |E|$$

Dimostrazione. Intuitivamente, preso un arco arbitrario $(i, j) \in E$, questo contribuisce per una unità sia a d_j^- che a d_i^+ . Quindi sommando tutti i gradi in ingresso si ottiene esattamente $|E|$, e lo stesso sommando i gradi di uscita.

Una dimostrazione discorsiva di questo tipo in generale è più che convincente, ma per gli obiettivi di questo corso è utile presentare anche una dimostrazione per induzione. Procediamo per induzione su $|E|$ e dimostriamo che $\sum_{x \in V} d_x^+ = |E|$ (l'altra uguaglianza è analoga).

[Caso Base] Se $|E| = 0$ non ci sono archi, quindi il grado di uscita di ogni nodo è 0, e quindi $\sum_{x \in V} d_x^+ = 0 = |E|$.

[Passo induttivo] Supponiamo che la proprietà valga per tutti i grafi con m archi. Sia $G = (V, E)$ un grafo orientato con $|E| = m + 1$. Essendo $m + 1 > 0$, esiste almeno un arco: scegliamone uno in modo arbitrario, e sia esso $(i, j) \in E$. Consideriamo ora il grafo G' ottenuto rimuovendo l'arco (i, j) da G , quindi $G' = (V, E')$ con $E' = E \setminus \{(i, j)\}$. Poiché $|E'| = m$, per ipotesi induttiva abbiamo che $\sum_{x \in V} d_x^+ = |E'|$, dove d_x^+ è il grado di uscita del nodo x in G' . Per come abbiamo definito G' abbiamo che $d_x^+ = d'_x^+$ per ogni $x \in V \setminus \{i\}$, mentre $d_i^+ = d'_i^+ + 1$. Quindi sommando i gradi di uscita di G otteniamo:

$$\sum_{x \in V} d_x^+ = \sum_{x \in V} d'_x^+ + 1 = [\text{per ipotesi induttiva}] \quad |E'| + 1 = |E|$$

■

Esempio 5.1.6. Consideriamo il grafo orientato nella Figura 5.1 (a), dove $|V| = 6$ ed $|E| = 11$. Si noti che i vicinati in ingresso sono $N^-(0) = \{0\}$, $N^-(1) = \{0, 2\}$, $N^-(2) = \{3\}$, $N^-(3) = \{1, 5\}$, $N^-(4) = \{3, 5\}$, $N^-(5) = \{0, 1, 3\}$, e i vicinati in uscita sono $N^+(0) = \{0, 1, 5\}$, $N^+(1) = \{3, 5\}$, $N^+(2) = \{1\}$, $N^+(3) = \{2, 4, 5\}$, $N^+(4) = \{\}$, $N^+(5) = \{3, 4\}$. Come si può facilmente verificare, la somma dei gradi in ingresso è uguale alla somma dei gradi di uscita, entrambi uguali a $|E|$.

¹La lettera N sta per *neighborhood* e la lettera d per *degree*, i corrispondenti termini inglesi.

È interessante notare che le quattro principali proprietà delle relazioni studiate nel Capitolo 2 corrispondono esattamente a condizioni sui gradi d'ingresso e d'uscita di tutti i nodi, come elencato sotto e mostrato nella Tabella 5.1:

1. $E: V \leftrightarrow V$ è totale se e solo se per ogni nodo $x \in V$ vale $d_x^+ \geq 1$;
2. $E: V \leftrightarrow V$ è univalente se e solo se per ogni nodo $x \in V$ vale $d_x^+ \leq 1$;
3. $E: V \leftrightarrow V$ è iniettiva se e solo se per ogni nodo $x \in V$ vale $d_x^- \leq 1$;
4. $E: V \leftrightarrow V$ è surgettiva se e solo se per ogni nodo $x \in V$ vale $d_x^- \geq 1$.

	$\forall x \in V. d_x^+ \dots$	$\forall x \in V. d_x^- \dots$
$\dots \geq 1$	TOTALE	SURGETTIVA
$\dots \leq 1$	UNIVALENTE	INIETTIVA

Tabella 5.1: Tabella riassuntiva per collegare le quattro proprietà della relazione E su V con i gradi d'ingresso e d'uscita del grafo orientato $G = (V, E)$

Rappresentazione dei grafi orientati

Disegnare un grafo di dimensioni contenute come in Figura 5.1 (a) su di un foglio o sullo schermo del computer è efficace e intuitivo nella comunicazione tra esseri umani, ma per supportare la manipolazione di grafi, anche di grandi dimensioni, da parte di programmi software si utilizzano due modi principali per rappresentarli nella memoria del computer, come discusso di seguito. Per una trattazione sugli algoritmi che utilizzano queste rappresentazioni si rimanda al corso di Programmazione e Algoritmi.

Matrice di adiacenza

La rappresentazione più immediata per un grafo orientato $G = (V, E)$ con $n = |V|$ nodi è quella tabellare.

Definizione 5.1.7 (matrice di adiacenza). *La MATRICE DI ADIACENZA di G è una matrice quadrata A con n righe e n colonne, numerate da 0 a $n - 1$, dove l'elemento A_{ij} (in riga i e colonna j) assume un valore in $\{0, 1\}$ con il seguente significato:*

$$A_{ij} = \begin{cases} 1 & \text{se l'arco } (i, j) \in E; \\ 0 & \text{se l'arco } (i, j) \notin E; \end{cases}$$

Esempio 5.1.8. In Figura 5.1 (b) è mostrata la matrice di adiacenza del grafo orientato di Figura 5.1 (a).

Esercizio 5.1.9. Dato un grafo orientato $G = (V, E)$ con $n = |V|$, quanti archi può avere al massimo G ?

Esercizio 5.1.10. Si dimostri che l'insieme di tutti i grafi orientati con insieme di nodi n è in biiezione con l'insieme delle funzioni $\text{Fun}(n \times n, 2)$.²

Per quanto riguarda l'occupazione di memoria la matrice di adiacenza richiede $n \times n$ celle, indipendentemente dal numero $m = |E|$ di archi.

Osservazione 5.1.11. La rappresentazione mediante matrici di adiacenza è utile in quei contesti dove si possono usare tecniche provenienti dall'algebra lineare e dagli algoritmi numerici, che saranno discusse nei corsi successivi. Nel nostro contesto, discutiamo un esempio famoso che

²Si ricordi che per la Definizione 1.5.14 ogni naturale è un insieme, e si veda la Notazione 2.5.3 per il significato di $\text{Fun}(_, _)$.

riguarda le pagine web. Il grafo $G = (V, E)$ per il web è orientato, dove V rappresenta l'insieme di tutte le pagine web navigabili con i nostri "browser". Nel navigare passiamo da una pagina i a una pagina j facendo click in un punto opportuno della pagina i (in termini tecnici, seguiamo un "hyperlink"): questo stabilisce una relazione implicita tra le due pagine che viene modellata con l'arco $(i, j) \in E$.

Questa struttura a grafo implicita tra le pagine web, unitamente a ulteriori tecniche, consente di realizzare una importante funzionalità dei motori di ricerca: classificare le pagine web in base alla loro importanza. Questa informazione viene utilizzata nel momento di presentare i risultati all'utente, ordinandoli per importanza decrescente. Una delle tecniche più popolari è quella di fornire un valore di significatività alle pagine, chiamato rango, e sostanzialmente procedere con il seguente meccanismo di votazione: l'arco (i, j) viene interpretato come un voto che i effettua nei confronti di j e il contributo di tale voto viene pesato in base al rango corrente di i . Attraverso un meccanismo iterativo di voto, il rango di ogni pagina j viene aggiornato in base ai ranghi dei suoi vicini entranti in $N^-(j)$: una pagina è tanto più significativa quanto più lo sono le pagine che puntano a lei. Se certe condizioni del grafo sono soddisfatte, il meccanismo si stabilizza: a quel punto un rango maggiore denota una significatività maggiore. Il meccanismo sfrutta la rappresentazione del web come matrice di adiacenza e l'iterazione è ottenuta come moltiplicazione tra opportune matrici.

Liste di adiacenza

Per le reti sociali e per molti grafi che modellano strutture dal mondo reale (internet, reti neuronali, alberi genealogici, ...), spesso si utilizza una rappresentazione diversa che permette di risparmiare spazio se il grafo è sparso, cioè se il numero di archi è proporzionale al numero di nodi. A tal fine si utilizzano le *liste di adiacenza*.

Definizione 5.1.12 (liste di adiacenza). *Le rappresentazione con LISTE DI ADIACENZA di un grafo orientato $G = (V, E)$ è costituita da un array A di $n = |V|$ insiemi in cui l'elemento i -esimo è il vicinato in uscita del nodo $i \in V$, cioè $A[i] = N^+(i)$.*

Esempio 5.1.13. La Figura 5.1(c) mostra la rappresentazione con liste di adiacenza del grafo di Figura 5.1(a). Si noti che il nodo 4 non ha vicini in uscita e quindi la sua lista è vuota.

È facile convincersi che i vicinati in uscita sono sufficienti per rappresentare tutti gli archi in E e che da questi è possibile ricavare i vicinati in ingresso. Questa è solo una delle possibili rappresentazioni con liste di adiacenza: per esempio per specifiche applicazioni può essere conveniente memorizzare per ogni nodo il vicinato in uscita, invece di quello in ingresso, oppure entrambi. Inoltre per la rappresentazione dell'insieme $N^+(i)$ può essere scelta una qualunque tecnica di rappresentazione di insiemi (come liste, array, alberi binari, ...), un argomento che verrà affrontato nel corso di Programmazione e Algoritmica.

Astraendo dalla rappresentazione concreta scelta, possiamo comunque osservare che usando le liste di adiacenza si richiedono n celle di memoria, una per nodo del grafo, e poi tante celle quanti sono gli elementi nelle liste di adiacenza. Poiché la lista per il nodo i contiene un numero di elementi pari al suo grado ($d_i = |N^+(i)|$), per la Proposizione 5.1.5 avremo che $\sum_{i \in V} d_i^+ = |E| = m$ celle sono occupate in questo modo. Quindi in totale abbiamo bisogno di $n + m$ celle di memoria, e questo rende conveniente questa rappresentazione rispetto alla matrice di adiacenza quando $n + m$ è molto minore di n^2 (come spesso succede nei grafi reali).

Osservazione 5.1.14. Le liste di adiacenza sono utilizzate in modo proficuo per esempio dai motori di ricerca, nella fase di raccolta delle pagine navigabili. Il meccanismo usato per tale raccolta sfrutta specifici programmi chiamati crawler o spider. Questi partono da un insieme predeterminato S di pagine web a cui aggiungono le pagine via via esplorate: presa una pagina $i \in S$, ne viene esaminato il contenuto per determinare il suo vicinato in uscita $N^+(i)$; per ogni pagina $j \in N^+(i)$, se la pagina j non appare in S , viene aggiunta a esso. Questo procedimento si ripete fino a che non è più possibile estendere S in questo modo. La scelta iniziale di S è determinante per raggiungere il maggior numero di pagine e non tutte le pagine sono comunque raggiunte in questo modo: si pensi per esempio a quelle del dark web oppure alle pagine generate dinamicamente.

Grafi etichettati e pesati

La struttura di base di un grafo $G = (V, E)$ può essere arricchita con ulteriori informazioni, aggiungendo delle *etichette* sugli archi e/o sui nodi. Per esempio, se il grafo rappresenta una rete sociale, possiamo etichettare ogni nodo con l'identità dell'utente corrispondente e l'arco con la data in cui è stata accettata l'amicizia. In una mappa stradale, possiamo associare ai nodi le località e agli archi la distanza in chilometri tra di esse.

Definizione 5.1.15 (grafo etichettato, grafo pesato). *Un GRAFO ORIENTATO ETICHETTATO è una tripla $G = (V, E, L)$ dove L è una funzione $L : (V \cup E) \rightarrow D$ che associa ad ogni nodo e arco una ETICHETTA presa da un certo dominio D di valori (identità, date, località, distanze, ecc.). Nel caso che D sia un valore numerico (solitamente un numero reale), il grafo etichettato si chiama PESATO e ciascuna etichetta viene indicata come PESO (dell'arco o del nodo).*³

Le possibili rappresentazioni di un grafo orientato etichettato (o pesato) sono un'immediata estensione di quanto visto sopra.

- Per i nodi, si usa un array di n etichette, dove l'elemento i -esimo contiene l'etichetta del nodo i .
- Nella matrice di adiacenza, l'elemento $A_{i,j}$ contiene l'etichetta $L((i, j))$ se $(i, j) \in E$, mentre contiene un valore speciale (per esempio `null`) al posto del valore 0 se l'arco non esiste.
- Nelle liste di adiacenza, $A[i]$ contiene tutte le coppie del tipo $(j, L(i, j))$ dove $j \in N^+(i)$ è un vicino in uscita di i .

Un esempio di grafo pesato con le sue rappresentazioni è illustrato in Figura 5.2.

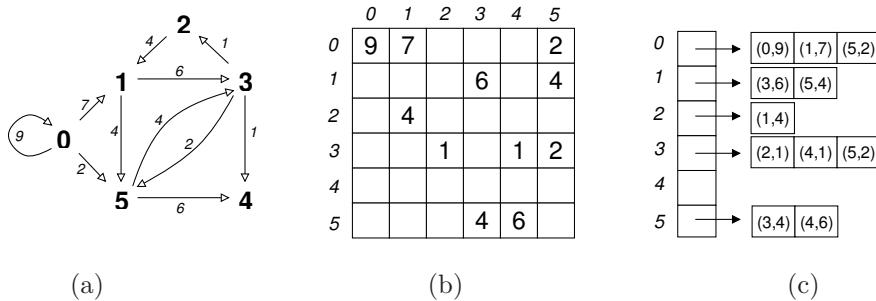


Figura 5.2: (a) Un grafo orientato pesato; (b) La sua matrice di adiacenza; (c) Le sue liste di adiacenza.

5.2 Cammini, cicli e connettività nei grafi orientati

In un grafo orientato (eventualmente pesato) la presenza di un arco (i, j) può essere interpretato come il fatto che il nodo j è raggiungibile direttamente dal nodo i (eventualmente con un certo costo, determinato dall'etichetta). Molti problemi computazionali definiti su strutture modellabili come grafi sono formulabili come problemi di raggiungibilità tra nodi. Per esempio, in un albero genealogico possiamo chiederci se due persone sono una discendente dell'altra; in una rete sociale possiamo controllare se due membri hanno conoscenti in comune.

Tutti questi problemi possono essere formulati in modo preciso su grafi partendo dalla nozione fondamentale di *cammino*, cioè una sequenza di nodi ognuno dei quali è collegato al successivo con un arco. Diremo per esempio che un nodo è raggiungibile da un altro se c'è un cammino che li collega. Un cammino chiuso, che inizia e termina nello stesso nodo, si chiama *ciclo*. Alcuni grafi non possono contenere cicli (per esempio una albero genealogico), mentre altri ne possono contenere (per esempio le reti sociali).

³La lettera L sta per *labelling*, inglese per *etichettatura*.

In questa sezione introduciamo formalmente questi concetti per i grafi orientati, ma attenzione: cammini e cicli come introdotti informalmente sopra sono dei concetti astratti che corrispondono ognuno a più definizioni formali, ognuna con le sue caratteristiche. Iniziamo definendo tre tipi di “cammini” usando, per evitare ambiguità, i loro nomi in inglese (*walk*, *trail* e *path*), mentre useremo solo sporadicamente il termine “cammino” in modo informale. Successivamente introdurremo tre tipi di “cicli” (*walk chiuso*, *circuito* e *ciclo*): come il lettore attento avrà capito, useremo a volte “ciclo” in modo informale per “cammino chiuso”, e più spesso nel modo formale che definiremo più avanti.

Definizione 5.2.1 (walk). *Sia $G = (V, E)$ un grafo orientato. Un WALK P in G è una sequenza di nodi $P = v_0, \dots, v_k$ di lunghezza arbitraria (con $k \in \mathbb{N}$), tali che $(v_{i-1}, v_i) \in E$ per $i \in \{1, \dots, k\}$. In questo caso P è un walk di LUNGHEZZA k . Le coppie $(v_{i-1}, v_i) \in E$ sono detti gli ARCHI ATTRaversati da P ed i nodi v_0, \dots, v_k sono detti i NODI ATTRaversati da P . Si dice inoltre che P è un WALK DA v_0 A v_k , o che P INIZIA con il nodo v_0 e TERMINA nel nodo v_k . I nodi v_0 e v_k sono chiamati gli ESTREMI di P . Si noti che se $k = 0$, il walk ha lunghezza 0 ed è costituito dal solo nodo v_0 .*

Esempio 5.2.2. Nel grafo orientato in Figura 5.1(a), la sequenza $0, 1, 3, 4$ è un walk: attraversa gli archi $(0, 1)$, $(1, 3)$ e $(3, 4)$; attraversa i nodi $0, 1, 3, 4$; i suoi estremi sono i nodi 0 e 4 . Questo walk ha lunghezza 3.

Invece la sequenza $5, 1, 3$ non è un walk perché la coppia $(5, 1)$ non è un arco in E (c’è un arco da 1 a 5 ma non vice versa).

La sequenza 0 è un walk di lunghezza 0 che, come tutti i walk di lunghezza 0, non attraversa alcun arco. La sequenza $0, 0$ è un walk di lunghezza 1 che attraversa l’arco $(0, 0) \in E$ (si noti che questo è un cappio). La sequenza $0, 0, 0$ è un walk di lunghezza 2 che attraversa (due volte) l’arco $(0, 0) \in E$.

In molte occasioni siamo interessati a vedere se esiste un walk da un nodo x a un nodo y , cioè se y è raggiungibile da x . A volte siamo interessati a walk di lunghezza fissata. Per esempio, se un grafo rappresenta con le rotte delle compagnie aeree che collegano delle città, possiamo chiederci se esistono voli da Pisa a Honolulu che fanno esattamente due scali. La seguente proposizione collega walk di lunghezza n con la relazione E^n definita nella Sezione 4.3.

Proposizione 5.2.3. Sia $G = (V, E)$ un grafo orientato e siano $x, y \in V$ due nodi. Allora vale che:

$$\text{Esiste un walk di lunghezza } n \in \mathbb{N} \text{ da } x \text{ a } y \text{ se e solo se } (x, y) \in E^n.$$

Dimostrazione. Dimostriamo la proposizione per induzione.

[Caso Base] Per ogni nodo $x \in V$ esiste uno e un solo walk di lunghezza 0 che parte da x , cioè il walk x (da x a x). D’altra parte $(x, x) \in id_V$ e per la Definizione 4.3.13 abbiamo che $E^0 = id_V$, quindi $(x, x) \in E^0$.

[Passo induttivo] Per ipotesi induttiva assumiamo che la proposizione valga per i walk di lunghezza n e dimostriamo che allora vale anche per quelli di lunghezza $n + 1$.

Esiste un walk di lunghezza $n + 1$ da x a y		
se e solo se	esiste un walk v_0, v_1, \dots, v_{n+1} con $v_0 = x$ e $v_{n+1} = y$	
se e solo se	$(x, v_1) \in E$ ed esiste un walk v_1, \dots, v_{n+1} con $v_{n+1} = y$	
se e solo se	$(x, v_1) \in E$ ed esiste un walk di lunghezza n da v_1 a y	
se e solo se	$(x, v_1) \in E$ e $(v_1, y) \in E^n$	(Ip. Induttiva)
se e solo se	$(x, y) \in E^{n+1}$	(Def. 4.3.13)

■

Definizione 5.2.4 (trail, path). *Un walk P è detto un TRAIL se attraversa ogni arco in E al più una volta. Un trail è detto un PATH se attraversa ogni nodo in V al più una volta.*

Esempio 5.2.5. Nel grafo orientato in Figura 5.1, il walk $0, 1, 3, 4$ è un path (e quindi anche un trail). Il walk $0, 1, 3, 2, 1, 3, 4$ non è un trail (e quindi neanche un path): l’arco $(1, 3)$ viene attraversato due volte. Il walk $5, 3, 2, 1, 3, 5, 4$ è un trail ma non un path: i nodi 3 e 5 vengono attraversati due volte.

Il walk 0 è, come tutti i walk di lunghezza 0, un path (e quindi anche un trail). Il walk 0,0 non è un path, ma è un trail: il nodo 0 viene attraversato due volte, mentre l'arco (0,0) una volta sola. Il walk 0,0,0 non è un trail (e quindi neanche un path): l'arco (0,0) viene attraversato due volte.

Chiaramente tutti i path sono trail e tutti i trail sono walk, ma l'esempio ci mostra che non è vero il contrario: esistono walk che non sono trail ed esistono trail che non sono path. Comunque nei grafi orientati, se esiste un walk da un nodo x ad un nodo y , allora esiste sicuramente anche un path (e quindi un trail). Questo fatto è enunciato nella Proposizione 5.2.9 che illustriamo dopo aver dimostrato un paio di lemmi accessori.

Lemma 5.2.6. *Sia $G = (V, E)$ un grafo orientato e siano $x, y \in V$ due nodi. Allora vale che:*

1. *Se esiste un walk da x a y , allora esiste un trail da x a y .*
2. *Inoltre se il walk ha lunghezza > 0 , allora anche il trail ha lunghezza > 0 .*

Dimostrazione. Sia $P = v_0, \dots, v_k$ un walk in G con $v_0 = x$ e $v_k = y$. Se questo walk è un trail, allora non abbiamo niente da dimostrare. Se non è un trail allora esiste almeno un $j \in \{1, \dots, k-1\}$ ed un $i \in \{j+1, \dots, k\}$ tale che $v_{j-1} = v_{i-1}$ e $v_j = v_i$. Cioè

$$P = v_0, \dots, v_{j-1}, v_j, \dots, v_{i-1}, v_i, \dots, v_k$$

e l'arco $(v_{j-1}, v_j) = (v_{i-1}, v_i)$ è attraversato due volte.

Possiamo quindi costruire il walk

$$v_0, \dots, v_{j-1}, v_i, \dots, v_k \tag{5.1}$$

da v_0 a v_k . Si noti che tale walk non è necessariamente un trail, ma contiene sicuramente un numero minore di ripetizioni di archi. Iterando la costruzione appena illustrata, si possono rimuovere tutte le ripetizioni di archi ed ottenere quindi un trail.

Adesso dimostriamo che se $k > 0$, allora esiste anche un trail da v_0 a v_k di lunghezza strettamente maggiore di 0. Ma questo è immediato perché, nella procedura illustrata sopra, viene comunque lasciato almeno un arco: (v_{j-1}, v_i) in (5.1). ■

Lemma 5.2.7. *Sia $G = (V, E)$ un grafo orientato e siano $x, y \in V$ due nodi. Allora vale che:*

Se esiste un trail da x a y , allora esiste un path da x a y .

Dimostrazione. Sia $P = v_0, \dots, v_k$ un trail in G con $v_0 = x$ e $v_k = y$. Se questo trail è un path, allora non abbiamo niente da dimostrare. Se non è un path allora esiste almeno un $j \in \{0, \dots, k-1\}$ ed un $i \in \{j+1, \dots, k\}$ tale che $v_j = v_i$. Cioè

$$v_0, \dots, v_j, \dots, v_i, \dots, v_k$$

In tal caso possiamo costruire una trail

$$v_0, \dots, v_j = v_i, \dots, v_k$$

da v_0 a v_k . Si noti che tale trail non è necessariamente un path, ma contiene sicuramente un numero minore di ripetizioni di nodi. Iterando la costruzione appena illustrata, si può sempre rimuovere tutte le ripetizioni di nodi ed ottenere quindi un path. ■

Esercizio 5.2.8.

1. *Fornire una dimostrazione per induzione del Lemma 5.2.7.*
2. *Fornire una dimostrazione per induzione del Lemma 5.2.6.*

Proposizione 5.2.9. *Sia $G = (V, E)$ un grafo orientato e siano $x, y \in V$ due nodi. Le seguenti affermazioni sono equivalenti:*

- (a) *esiste un walk da x a y ,*

- (b) esiste un trail da x a y ,
- (c) esiste un path da x a y ,
- (d) $(x, y) \in E^*$.

Dimostrazione. Dobbiamo dimostrare che (a) se e solo se (b) se e solo se (c) se e solo se (d).

Si osservi che per definizione di walk, path e trail si ha immediatamente che se (c) allora (b) e se (b) allora (a). Il Lemma 5.2.6 ci da se (a) allora (b) ed il Lemma 5.2.7 ci da se (b) allora (c). Pertanto (a) se e solo se (b) se e solo se (c).

Dalla Proposizione 5.2.3 segue immediatamente che (a) se e solo se (d). ■

Cicli nei grafi orientati

Definizione 5.2.10 (walk chiuso, circuito, ciclo). Un walk P è detto CHIUSO se i suoi estremi sono uguali (cioè $v_0 = v_k$) e se ha lunghezza > 0 . Un walk chiuso che è un trail è detto CIRCUITO. Un circuito che è anche un path è detto un CICLO, facendo ovviamente eccezione per gli estremi (cioè gli estremi non vengono contati due volte). In altre parole un ciclo è un circuito v_0, \dots, v_k (con $v_0 = v_k$) tale che v_0, \dots, v_{k-1} è un path.

Esempio 5.2.11. Il walk 1, 3, 2, 1 del grafo di Figura 5.1(1) è un ciclo (e quindi anche un circuito e un walk chiuso). Invece 1, 3, 5, 3, 2, 1 è un circuito poiché ogni arco è attraversato al più una volta, ma non un ciclo: passa per il nodo 3 due volte (chiaramente $v_0 = v_k = 1$ non conta).

Il walk 0 non è chiuso perché ha lunghezza 0. Il walk 0, 0 è un ciclo (e quindi anche un circuito ed un walk chiuso). Il walk 0, 0, 0 è chiuso ma non è un circuito (e quindi neanche un ciclo).

Definizione 5.2.12 (grafo ciclico, aciclico). Un grafo G si dice CICLICO se esiste almeno un ciclo in G . Altrimenti, si dice ACICLICO.

Ad esempio, il grafo orientato in Figura 5.1(a) è ovviamente ciclico.

La condizione di ciclicità può essere caratterizzata attraverso l'algebra delle relazioni come illustrato nella Proposizione 5.2.14. Prima però è opportuno chiarire i rapporti tra walk chiusi, circuiti e cicli.

Proposizione 5.2.13. Sia $G = (V, E)$ un grafo orientato e x un nodo in V . Le seguenti affermazioni sono equivalenti:

- (a) esiste un walk chiuso che inizia e termina in x ,
- (b) esiste un circuito che inizia e termina in x ,
- (c) esiste un ciclo che inizia e termina in x ,
- (d) $(x, x) \in E^+$.

Dimostrazione. Iniziamo dimostrando se (a) allora (b). Se esiste un walk chiuso da x a x , allora esiste un walk v_0, \dots, v_k di lunghezza $k > 0$ tale che $v_0 = v_k = x$. Dalla Proposizione 5.2.6, esiste un quindi un trail da x a x di lunghezza $k > 0$. Per definizione di circuito, questo è un circuito.

Dimostriamo se (b) allora (c). Se esiste un circuito da x a x , allora esiste un trail v_0, \dots, v_k di lunghezza $k > 0$ tale che $v_0 = v_k = x$. Si noti che, essendo $k > 0$, allora esiste anche un trail v_0, \dots, v_{k-1} (possibilmente di lunghezza 0). Per il Lemma 5.2.7, esiste un path (possibilmente di lunghezza 0) da v_0 a v_{k-1} . Aggiungendo a questo path l'arco $(v_{k-1}, v_k) \in E$ si ottiene un ciclo da x a x .

Si osservi che se (c) allora (b) e se (b) allora (a) seguono immediatamente dalla Definizione 5.2.10.

Abbiamo quindi dimostrato che (a) se e solo se (b) se e solo se (c). Dalla Proposizione 5.2.3 segue immediatamente che (a) se e solo se (d). ■

Proposizione 5.2.14. Per tutti i grafi orientati $G = (V, E)$ vale che:

$$G \text{ è aciclico se e solo se } E^+ \cap id_V = \emptyset_{V,V}$$

Dimostrazione. Dalla Proposizione 5.2.13 segue che G è aciclico se e solo se per tutti gli $x \in V$ $(x, x) \notin E^+$. Questo vale se e solo se $E^+ \cap id_V = \emptyset_{V,V}$. ■

Connettività

Una nozione intimamente collegata a quella di walk è la connettività tra nodi.

Definizione 5.2.15 (grafo fortemente connesso). *Un grafo orientato $G = (V, E)$ è FORTEMENTE CONNESSO se per ogni coppia di nodi $(u, v) \in V \times V$ esiste un walk da u a v .*

Definizione 5.2.16 (componente fortemente connessa). *Sia $G = (V, E)$ un grafo orientato. Una COMPONENTE FORTEMENTE CONNESSA di G è un sottoinsieme non vuoto di nodi $U \subseteq V$ tale che:*

1. *Per ogni coppia di nodi $(x, y) \in U \times U$ esiste un walk da x a y ;*
2. *Se $U' \subseteq V$ soddisfa la proprietà 1. e $U \subseteq U'$, allora $U = U'$.*

La prima condizione richiede che in una componente connessa, presi due nodi arbitrari x e y , ci sia sempre un walk da x a y e viceversa. La seconda condizione invece richiede che ogni componente fortemente connessa sia *massimale*, cioè aggiungendo un nodo esterno ad essa la prima condizione deve essere violata.

Esempio 5.2.17. *Per esempio, il grafo in Figura 5.1 non è fortemente connesso perché non esiste un cammino, per esempio, dal nodo 4 al nodo 0. Ci sono tre componenti fortemente connesse: due sono formate da singoli nodi, cioè $\{0\}$ e $\{4\}$, l'altra è formata dai restanti nodi, cioè $\{1, 2, 3, 5\}$.*

Si noti che, $\{1, 2, 3\}$ non è una componente fortemente connessa in quanto non soddisfa la seconda proprietà della Definizione 5.2.16, cioè non è massimale. Infatti $\{1, 2, 3\} \subsetneq \{1, 2, 3, 5\}$ che è una componente fortemente connessa.

Proposizione 5.2.18. *Se $G = (V, E)$ è fortemente connesso, allora G ha una sola componente fortemente connessa (che è esattamente l'intero insieme di nodi V).*

Dimostrazione. Si osservi che V rispetta sia la proprietà 1. della Definizione 5.2.16 che, ovviamente, la proprietà 2. Pertanto V è una componente fortemente connessa. Si osservi che è l'unica componente fortemente connessa: infatti un qualsiasi insieme $U \subseteq V$ che soddisfa le proprietà 1. e 2. è, a causa della proprietà 2., necessariamente uguale a V (si prenda $U' = V$). ■

Come si può osservare, prese due qualunque componenti fortemente connesse, esiste almeno un nodo di una delle due che non è collegato da un arco ad alcun nodo dell'altra componente, altrimenti le due componenti potrebbero fondersi in un'unica componente fortemente connessa. Per esempio, nel grafo in Figura 5.1, dal nodo 2 non si può andare al nodo 0; dal nodo 4 non si può andare né al nodo 0 né al nodo 2. Questa intuizione è resa formale dalla seguente proposizione.

Proposizione 5.2.19. *Sia $G = (V, E)$ un grafo orientato. L'insieme delle componenti fortemente connesse di G , cioè l'insieme*

$$\{U \subseteq V \mid U \text{ componente fortemente connessa di } G\} \quad (5.2)$$

forma una partizione di V .

Dimostrazione. Si ricordi la definizione di partizione (Definizione 1.5.9). Dobbiamo dimostrare che:

- Ogni componente fortemente connessa è non vuota: segue immediatamente dalla Definizione 5.2.16.
- Copertura: l'unione di tutte le componenti fortemente connesse è uguale a V . Infatti per ogni $v \in V$ abbiamo che $\{v\}$ rispetta la condizione 1. della Definizione 5.2.16. Quindi o $\{v\}$ è una componente fortemente connessa oppure $\{v\}$ è contenuto in una componente fortemente connessa.
- Disgiunzione: Se U_1 e U_2 sono due componenti fortemente connesse distinte allora sono disgiunte. Mostriamo, per assurdo, che se esiste un $x \in U_1 \cap U_2$, allora $U_1 \subseteq U_2$, e quindi per la condizione 2. della Definizione 5.2.16 $U_1 = U_2$, il che contraddice l'ipotesi.

Dato un qualunque nodo $y \in U_1$ esiste un walk da x a y e viceversa, perché U_1 è una componente fortemente connessa. Visto che anche U_2 è una componente fortemente connessa (e quindi soddisfa la proprietà di massimalità) e $x \in U_2$ allora anche $y \in U_2$.

■

Abbiamo visto nella Sezione 4.4 che le partizioni sono in corrispondenza uno a uno con le relazioni di equivalenza. È quindi naturale chiedersi a quale relazione di equivalenza corrisponda la partizione (5.2). Il seguente risultato fornisce una risposta a tale quesito, oltre a provvedere una caratterizzazione attraverso l'algebra delle relazioni della nozione di grafo fortemente connesso.

Proposizione 5.2.20. *Per tutti i grafi orientati $G = (V, E)$ e tutti gli $x, y \in V$, vale che:*

1. *G è fortemente connesso se e solo se $V \times V \subseteq E^*$;*
2. *$(x, y) \in E^* \cap (E^*)^{op}$ se e solo se x ed y appartengono alla stessa componente fortemente connessa.*

Dimostrazione.

1. G è fortemente connesso se e solo se (per la Definizione 5.2.15) per ogni $(x, y) \in V \times V$ esiste un walk da x a y , se e solo se (per la Proposizione 5.2.9) per ogni $(x, y) \in V \times V$ vale $(x, y) \in E^*$, se e solo se $V \times V \subseteq E^*$.
2. Abbiamo che $(x, y) \in E^* \cap (E^*)^{op}$ se e solo se esiste un walk da x a y e viceversa. Quindi x e y fanno parte della stessa componente connessa. D'altronde, se x e y fanno parte della stessa componente connessa, $(x, y) \in E^*$ e $(x, y) \in (E^*)^{op}$; quindi $(x, y) \in E^* \cap (E^*)^{op}$.

■

All'interno di ogni componente fortemente connessa si nota un'interessante proprietà: presi due qualunque nodi in essa, esiste sempre un walk chiuso che li attraversa entrambi. Per esempio, nel grafo in Figura 5.1, per i nodi 2 e 5 un walk chiuso è formato dalla sequenza 1, 5, 3, 2, 1; per i nodi 1 e 3 un walk chiuso corrisponde a 2, 1, 3, 2; per i nodi 3 e 5, un walk chiuso è dato da 3, 5, 3.

Proposizione 5.2.21. *Un grafo $G = (V, E)$ è fortemente connesso se e solo se per ogni coppia di nodi $x, y \in V$ distinti ($x \neq y$) esiste un walk chiuso che attraversa x e y .*

Dimostrazione. (solo se) Sia G fortemente connesso. Presi due nodi distinti arbitrari x e y in U , per definizione esiste un walk da x a y e un walk da y a x . Se prendiamo la concatenazione dei due walk otteniamo il walk

$$x, \dots, y, \dots, x$$

che è chiuso. Si noti che l'assunzione $x \neq y$ è necessaria per garantire che il walk chiuso risultante abbia lunghezza maggiore di 0.

(se) Se per ogni coppia di nodi distinti $x, y \in V$ esiste un walk chiuso che li attraversa, allora esiste un walk da x a y e uno da y a x . Inoltre ovviamente per ogni nodo $x \in V$ c'è un walk da x a x (quello di lunghezza 0). Quindi G è fortemente connesso perché soddisfa la condizione della Definizione 5.2.15. ■

5.3 Grafi orientati aciclici (DAG)

Definizione 5.3.1 (grafo orientato aciclico, DAG). *Un grafo orientato aciclico è detto DAG (dall'inglese directed acyclic graph). In un DAG i nodi con grado d'ingresso zero sono detti SORGENTI e quelli con grado d'uscita zero sono detti POZZI.*

Proposizione 5.3.2. *Se $G = (V, E)$ è un DAG, allora E^* è una relazione d'ordinamento parziale.*

Dimostrazione. E^* è riflessiva e transitiva. Dobbiamo dimostrare che è antisimmetrica: per il Teorema 4.2.19(4.) basta mostrare che $E^* \cap (E^*)^{op} \subseteq id_V$. Infatti se $(x, y) \in E^* \cap (E^*)^{op}$ allora per la Proposizione 5.2.9 esiste un walk da x a y e viceversa. Se valesse $x \neq y$, allora esisterebbe un walk chiuso che attraversa entrambi, il che è impossibile perché G è aciclico per ipotesi. Quindi $x = y$. ■

Osservazione 5.3.3. I DAG emergono in modo naturale in quei contesti in cui tra gli elementi di un insieme esiste una relazione che per sua natura è aciclica, come per esempio la relazione di propedeuticità tra gli esami di un corso di studi, la relazione di ereditarietà tra classi nella programmazione a oggetti, la relazione tra ingressi e uscite delle porte in un circuito logico, oppure l'ordine di valutazione delle formule in un foglio elettronico. In generale, supponiamo di avere decomposto un'attività complessa in un insieme di attività elementari e di avere individuato le relative dipendenze. Un esempio concreto potrebbe essere la costruzione di un'automobile: volendo eseguire sequenzialmente le attività elementari (per esempio, in una catena di montaggio), bisogna soddisfare il vincolo che, se l'attività B dipende dall'attività A , allora A va eseguita prima di B . Usando i DAG per modellare tale situazione, poniamo i vertici in corrispondenza biunivoca con le attività elementari, e introduciamo un arco (A, B) per indicare che l'attività A va eseguita prima dell'attività B .

Un'esecuzione in sequenza delle attività che soddisfi i vincoli di precedenza tra esse, corrisponde a un ordinamento topologico del DAG costruito su tali attività.

Definizione 5.3.4 (ordinamento topologico). *Dato un DAG $G = (V, E)$, un ordinamento topologico di G è una biiezione $\eta : V \rightarrow n = \{0, 1, \dots, n - 1\}$ ⁴ tale che*

$$\text{per ogni arco } (u, v) \in E \text{ vale } \eta(u) < \eta(v). \quad (5.3)$$

In altre parole, se disponiamo i vertici lungo una linea orizzontale in base alla loro numerazione η , in ordine crescente, otteniamo che gli archi risultano tutti orientati da sinistra verso destra.

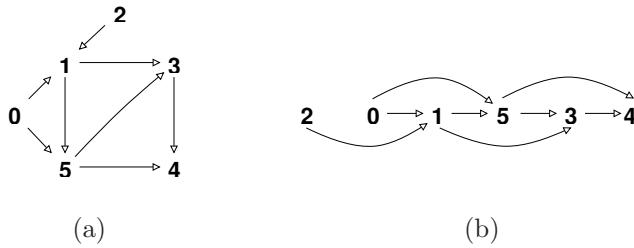


Figura 5.3: (a) Un DAG con due sorgenti (0 e 2) e un pozzo (4). (b) Rappresentazione grafica di un suo ordinamento topologico.

Proposizione 5.3.5. Ogni DAG $G = (V, E)$ ha almeno un ordinamento topologico.

Dimostrazione. Dimostriamo l'enunciato per induzione su $|V|$.

[Caso base] Se $|V| = 0$ allora $V = \emptyset$, ed esiste un'unica funzione $\eta : V \rightarrow 0 = \emptyset$ che è una biiezione e soddisfa banalmente la condizione (5.3).

[Passo induttivo] Se $|V| = n + 1$ il DAG non è vuoto ed esiste almeno un pozzo $x \in V$, altrimenti G sarebbe ciclico. Sia G' il grafo ottenuto da G togliendo il nodo x e tutti gli archi entranti in x . Chiaramente G' è un DAG (eliminando un nodo e degli archi non si può creare un ciclo) e ha n nodi, quindi per ipotesi induttiva esiste una biiezione $\eta' : V \setminus \{x\} \rightarrow n$ che soddisfa la condizione (5.3). Definiamo $\eta : V \rightarrow n + 1 = \{0, \dots, n\}$ come $\eta(v) = \eta'(v)$ per ogni $v \in V \setminus \{x\}$ e $\eta(x) = n$. Chiaramente η è una biiezione, e soddisfa la condizione (5.3) perché ogni arco $(u, v) \in E$ o appartiene a G' , oppure è un arco entrante in x . ■

L'ordinamento topologico può anche essere definito come un ordinamento totale compatibile con l'ordinamento parziale rappresentato dal DAG. Osserviamo che ci possono essere più ordinamenti topologici per lo stesso DAG.

Esercizio 5.3.6. Dato un grafo orientato $G = (V, E)$, si dimostri che le componenti fortemente connesse di G formano un DAG.

⁴Si ricordi che per la Definizione 1.5.14 ogni numero naturale n denota un insieme.

5.4 Grafi non orientati

Fino ad adesso abbiamo visto grafi orientati, cioè grafi in cui gli archi hanno una direzione prestabilita, e sono rappresentati da frecce da un nodo ad un altro. Esiste un altro tipo di grafi di fondamentale importanza: i *grafo non orientati*. In questi grafi, gli archi non hanno una direzione e, anzichè essere rappresentati da frecce, vengono rappresentati da linee che collegano due nodi.

I grafi non orientati si prestano bene a modellare per esempio delle mappe stradali: i nodi rappresentano delle località e gli archi le strade che li collegano. Tali archi non hanno una direzione e quindi possono essere percorsi in entrambi i sensi (non ci sono sensi unici!). Un altro esempio, forse ancora più calzante, è la mappa della metropolitana di una città: le fermate della metro sono i nodi e gli archi sono i collegamenti forniti dalla metropolitana. Un esempio di grafo non orientato è mostrato in Figura 5.4(a).

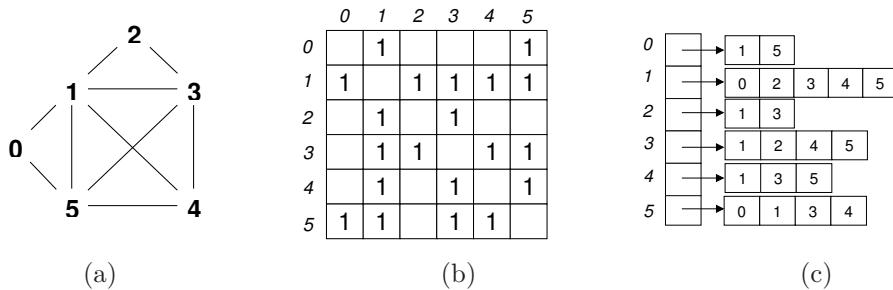


Figura 5.4: (a) Un grafo non orientato. (b) La sua matrice di adiacenza. (c) Le sue liste di adiacenza.

Per i grafi non orientati si possono introdurre molti concetti già presentati per quelli orientati, e in generale tali concetti risultano semplificati perché non occorre tener conto della direzione degli archi. Tuttavia in alcuni casi delle proprietà che valevano per i grafi orientati non valgono più. In generale, entrambe queste classi di grafi hanno una propria teoria e applicazioni rilevanti, e abbiamo presentato prima i grafi orientati solo perché sono definibili facilmente con le relazioni viste precedentemente.

Per definire formalmente i grafi non orientati utilizziamo il concetto di k -insieme introdotto nella Definizione 1.7.6, e in particolare l'insieme $\mathcal{P}_2(V)$ che contiene tutti i sottoinsiemi di V di cardinalità 2. Infatti identifichiamo un *arco non orientato* con l'insieme che contiene i suoi estremi, cioè un elemento di $\mathcal{P}_2(V)$. Poiché l'ordine di presentazione degli elementi di un insieme non è rilevante, gli elementi di $\mathcal{P}_2(V)$ catturano esattamente l'intuizione di *coppia non ordinata* e quindi di arco non orientato.

Definizione 5.4.1 (grafo non orientato). *Un GRAFO NON ORIENTATO $G = (V, E)$ consiste di un insieme finito V e di un insieme $E \subseteq \mathcal{P}_2(V)$. Gli elementi di V vengono detti NODI o VERTICI e gli elementi di E vengono detti ARCHI o LATI.*

Come per i grafi orientati, utilizzeremo n ed m per indicare, rispettivamente il numero di nodi e di archi, e inoltre assumeremo spesso V essere l'insieme $V = \{0, \dots, n-1\}$. Per alleggerire la notazione, talvolta scriveremo x, y per indicare l'arco non orientato $\{x, y\}$.

È importante osservare che, per definizione, nei grafi non orientati *non ci possono essere cappi*: l'insieme $\{x, x\}$ è esattamente l'insieme $\{x\}$ ed quindi ha cardinalità 1, non 2.

Esempio 5.4.2. Nell'esempio di Figura 5.4(a), il grafo non orientato ha $V = \{0, 1, 2, 3, 4, 5\}$, quindi ci sono $n = 6$ nodi e $m = 10$ archi: l'insieme degli archi è $E = \{01, 05, 12, 13, 14, 15, 23, 34, 35, 45\}$. Si noti che $\{1, 5\}$, $\{5, 1\}$, 15, 51 sono tutte notazioni alternative per indicare il medesimo arco tra 1 e 5.

Il lettore avrà probabilmente pensato che ogni grafo non orientato può essere visto come un grafo orientato, rappresentando ogni arco non orientato $\{x, y\} \in E$ con due archi orientati (x, y) e (y, x) . Questa idea è resa formale dalla seguente definizione.

Definizione 5.4.3 (grafo orientato associato). *Dato un grafo non orientato $G = (V, E)$, quindi $E \subseteq \mathcal{P}_2(V)$, il GRAFO ORIENTATO ASSOCIATO a G è il grafo orientato $\tilde{G} = (V, \tilde{E})$ dove la relazione \tilde{E} è definita come*

$$\tilde{E} = \{(x, y) \in V \times V \mid \{x, y\} \in E\} : V \leftrightarrow V.$$

Esempio 5.4.4. *Sia $G = (V, E)$ il grafo non orientato della Figura 5.5(a). Il corrispondente grafo orientato $\tilde{G} = (V, \tilde{E})$ è disegnato in Figura 5.5(b). Si noti come ad ogni arco non orientato corrispondano due archi orientati: ad esempio all'arco non orientato $\{0, 1\}$ corrispondono gli archi orientati $(0, 1)$ e $(1, 0)$. Quindi per ogni grafo non orientato G il numero di archi di \tilde{G} è il doppio del numero degli archi di G , e la relazione $\tilde{E} : V \leftrightarrow V$ è una relazione simmetrica.*

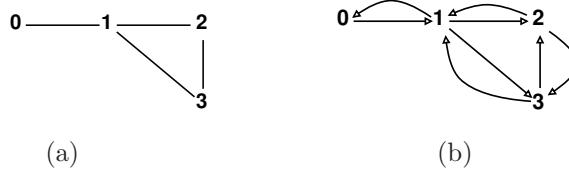


Figura 5.5: (a) Un grafo non orientato G . (b) Il grafo orientato associato \tilde{G} .

Purtroppo però guardare i grafi non orientati come grafi orientati simmetrici risulta essere fuorviante in diverse occasioni. Per questa ragione è opportuno re-introdurre molte di quelle nozioni che abbiamo già visto per i grafi orientati.

Vicinato e grado dei nodi

Le nozioni di vicinato e grado dei nodi risultano semplificate nel caso di grafi non orientati perché non si deve tener conto della direzione degli archi.

Definizione 5.4.5 (vicinato, grado). *Sia $G = (V, E)$ un grafo non orientato, quindi $E \subseteq \mathcal{P}_2(V)$. Due nodi $x, y \in V$ sono VICINI o ADIACENTI se c'è un arco $xy \in E$: in tal caso tale arco si dice INCIDENTE a x e y , i quali sono chiamati ESTREMI dell'arco. Il VICINATO di un nodo $x \in V$ è l'insieme $N(x) = \{y \mid xy \in E\}$. Il nodo x è UNIVERSALE se è vicino a tutti i nodi (cioè $N(x) \cup \{x\} = V$), mentre è ISOLATO se $N(x)$ è vuoto. Il GRADO di x è definito come il numero di nodi a lui vicini, $d_x = |N(x)|$; quindi $d_x = 0$ se il nodo è isolato. Con Δ si rappresenta il grado massimo in G (cioè $\Delta = \max\{d_x \mid x \in V\}$).*

Proposizione 5.4.6 (Hand-shaking lemma (stretta di mano)). *Per ogni grafo non orientato $G = (V, E)$, vale che*

$$\sum_{x \in V} d_x = 2|E|$$

ovvero la somma dei gradi dei nodi è il doppio del numero degli archi. Inoltre, G contiene un numero pari di nodi che hanno grado dispari.

Dimostrazione. Si osservi che per ogni nodo $x \in V$ il grado d_x nel grafo non orientato G è uguale al grado di uscita \tilde{d}_x^+ dello stesso nodo nel grafo orientato \tilde{G} associato a G . Quindi abbiamo

$$\sum_{x \in V} d_x = \sum_{x \in V} \tilde{d}_x^+ = [Prop. 5.1.5] \quad |\tilde{E}| = 2|E|$$

Per la seconda parte dell'enunciato, poiché la somma dei gradi è pari ($2|E|$), essa può essere ottenuta solo sommando un qualunque numero di gradi pari e un numero *pari* di gradi dispari, altrimenti la somma risultante sarebbe dispari. ■

Esempio 5.4.7. Consideriamo il grafo a sinistra nella Figura 5.4(a), dove $|V| = 6$ ed $|E| = 10$. Si noti che i vicinati sono $N(0) = \{1, 5\}$, $N(1) = \{0, 2, 3, 4, 5\}$, $N(2) = \{1, 3\}$, $N(3) = \{1, 2, 4, 5\}$, $N(4) = \{1, 3, 5\}$, $N(5) = \{0, 1, 3, 4\}$, e il nodo 1 è universale. La somma dei gradi è $d_0 + d_1 + d_2 + d_3 + d_4 + d_5 = 2 + 5 + 2 + 4 + 3 + 4 = 20 = 2|E|$; inoltre ci sono 2 nodi con gradi dispari (d_1 e d_4).

Esercizio 5.4.8. Si dimostri l'Hand-shaking lemma (Proposizione 5.4.6) per induzione, seguendo lo schema della dimostrazione per induzione della Proposizione 5.1.5.

È naturale chiedersi come i concetti appena illustrati per i grafi non orientati siano collegati alle rispettive nozioni sui grafi orientati che abbiamo introdotto nella Sezione 5.1.

Proposizione 5.4.9. Sia $G = (V, E)$ un grafo non orientato e $\tilde{G} = (V, \tilde{E})$ il grafo orientato associato della Definizione 5.4.3. Per tutti i nodi $x \in V$ vale che

- Il vicinato di x in G coincide con il vicinato di ingresso e di uscita di x in \tilde{G} , cioè $N(x) = N^+(x) = N^-(x)$.
- Il grado di x in G coincide con il grado di ingresso e di uscita di x in \tilde{G} , cioè $d(x) = d^+(x) = d^-(x)$.

La dimostrazione di questo facile risultato è lasciato come esercizio.

Esercizio 5.4.10. Dimostrare la Proposizione 5.4.9.

Rappresentazione dei grafi non orientati

La rappresentazione di grafi non orientati mediante matrici e liste di adiacenza è del tutto simile a quella vista per i grafi orientati. Nella matrice di adiacenza di un grafo non orientato, l'elemento $A_{i,j}$ vale 1 se $\{i, j\} \in E$ e 0 altrimenti. Si noti che la matrice risultante è *simmetrica*: $A_{i,j} = 1$ se e solo $A_{j,i} = 1$; inoltre la diagonale è nulla: $A_{i,i} = 0$ per ogni $i \in V$, perché non ci possono essere cappi. Nelle liste di adiacenza, per ogni nodo $i \in V$ l'elemento i -esimo dell'array A è il vicinato di i , $A[i] = N(x)$.

Un esempio di grafo non orientato con la sua rappresentazione con matrice di adiacenza e con liste di adiacenza è illustrato in Figura 5.4.

Concludiamo questa sezione osservando che le definizioni di *grafo etichettato* e *grafo pesato* che abbiamo visto nel caso dei grafi orientati si possono adattare facilmente al caso di grafi non orientati.

5.5 Cammini, cicli e connettività nei grafi non orientati

I grafi non orientati si prestano a modellare molti problemi computazionali e, come vedremo in seguito, Eulero li usò nel XVIII secolo per risolvere un “puzzle” della sua epoca che riguardava i percorsi nella città di Königsberg. Per adesso immaginiamo la mappa di una città come un grafo non orientato G , dove i nodi corrispondono a incroci e gli archi corrispondono a segmenti di strada.

Definizione 5.5.1 (walk). Sia $G = (V, E)$ un grafo non orientato. Un walk in G è una sequenza (di lunghezza arbitraria) di nodi v_0, \dots, v_k , tali che $\{v_{i-1}, v_i\} \in E$ per ogni $i \in \{1, \dots, k\}$.

La lunghezza di un walk, gli estremi, i nodi attraversati e gli archi attraversati sono definiti come per i grafi orientati.

Esempio 5.5.2. Nel grafo non orientato di Figura 5.4(a), la sequenza $0, 1, 3, 4$ è un walk dal nodo 0 al nodo 4 di lunghezza 3. Attraversa gli archi $0, 1, 1, 3, 3, 4$ ed i nodi $0, 1, 3, 4$.

Esercizio 5.5.3. Sia $G = (V, E)$ un grafo non orientato e $\tilde{G} = (V, \tilde{E})$ il grafo orientato associato. Dimostrare che per tutte le sequenze di nodi v_0, \dots, v_k vale che: v_0, \dots, v_k è un walk in G se e solo se è un walk in \tilde{G} .

Combinando l'esercizio precedente con la Proposizione 5.2.3, si ottiene immediatamente il seguente risultato.

Proposizione 5.5.4. Per tutti i grafi non orientati $G = (V, E)$ e tutti i nodi $x, y \in V$ vale che:

esiste un walk di lunghezza $n \in \mathbb{N}$ da x a y se e solo se $(x, y) \in \tilde{E}^n$.

I concetti di *path* e *trail* sono definiti come per i grafi orientati (Definizione 5.2.4).

Esempio 5.5.5. Nel grafo di Figura 5.4(a), il walk $5, 1, 2, 3, 5, 0$ è un trail perché non attraversa nessun arco più di una volta, ma non è un path perché attraversa due volte il nodo 5. Il walk $3, 5, 1, 4, 5, 3, 2$ non è un trail (e quindi neanche un path), dato che l'arco non orientato 35 viene attraversato più di una volta.

È importante osservare che ci sono delle sottili differenze tra i concetti di path e trail su un grafo non orientato G e i corrispondenti concetti sul grafo orientato associato \tilde{G} (mentre i concetti di walk coincidono: vedi Esercizio 5.5.3). Infatti presi due qualsiasi nodi adiacenti x e y in un grafo non orientato G , il walk x, y, x non è un trail in quanto l'arco $\{x, y\}$ viene attraversato due volte, mentre nel grafo orientato associato \tilde{G} tale walk è un trail perché prima si attraversa l'arco orientato (x, y) e poi l'arco orientato (y, x) .

Esempio 5.5.6. Si consideri il grafo non orientato G e il grafo orientato associato \tilde{G} in Figura 5.5. Il walk $0, 1, 0$ è un trail in \tilde{G} ma non in G : in quest'ultimo l'arco non orientato $\{0, 1\}$ viene attraversato due volte.

Lemma 5.5.7. Sia $G = (V, E)$ un grafo non orientato e siano $x, y \in V$ due nodi. Allora vale che:

Se esiste un walk da x a y , allora esiste un trail da x a y .

Dimostrazione. Sia $P = v_0, \dots, v_k$ un walk in G con $v_0 = x$ e $v_k = y$. Se questo walk è un trail, allora non abbiamo niente da dimostrare. Se non è un trail allora esiste almeno un arco che viene attraversato due volte. Ci sono due casi: o l'arco è attraversato due volte nella stessa direzione o in direzione opposta.

- L'arco è attraversato due volte nella stessa direzione. Allora esistono un $j \in \{1, \dots, k-1\}$ ed un $i \in \{j+1, \dots, k\}$ tali che $v_{j-1} = v_{i-1}$ e $v_j = v_i$. In questo caso possiamo procedere come nella dimostrazione del Lemma 5.2.6.
- L'arco è attraversato in direzione opposta. Allora esistono un $j \in \{1, \dots, k-1\}$ ed un $i \in \{j+1, \dots, k\}$ tali che $v_{j-1} = v_i$ e $v_j = v_{i-1}$. Cioè

$$P = v_0, \dots, v_{j-1}, v_j, \dots, v_{i-1}, v_i, \dots, v_k$$

e l'arco $\{v_{j-1}, v_j\} = \{v_{i-1}, v_i\}$ è attraversato due volte.

Possiamo quindi costruire il walk

$$v_0, \dots, v_{j-1} = v_i, \dots, v_k \tag{5.4}$$

da v_0 a v_k . Si noti che tale walk non è necessariamente un trail, ma contiene sicuramente un numero minore di ripetizioni di archi. Iterando la costruzione appena illustrata, si possono rimuovere tutte le ripetizioni di archi ed ottenere quindi un trail. ■

È importante osservare che l'analogo del punto 2. del Lemma 5.2.6 non vale: se esiste un walk di lunghezza maggiore di zero, il trail ottenuto come nella dimostrazione di sopra potrebbe avere lunghezza zero.

Esempio 5.5.8. Si consideri il grafo non orientato G di Figura 5.5(a). Il walk $0, 1, 0$ ha lunghezza 3 e va dal nodo 0 al nodo 0. Invece si noti che l'unico trail in G che va dal nodo 0 al nodo 0 è il walk 0 che ha lunghezza 0.

L'analogo del Lemma 5.2.7 vale anche per i grafi non orientati (la dimostrazione, che non riportiamo, è identica).

Lemma 5.5.9. Per tutti i grafi non orientati $G = (V, E)$ e tutti i nodi x, y in V vale che:

Se esiste un trail da x a y , allora esiste un path da x a y .

Proposizione 5.5.10. Sia $G = (V, E)$ un grafo non orientato e x, y due nodi in V . Le seguenti affermazioni sono equivalenti:

- (a) esiste un walk da x a y ,
- (b) esiste un trail da x a y ,
- (c) esiste un path da x a y ,
- (d) $(x, y) \in \tilde{E}^*$.

La dimostrazione è analoga a quella della Proposizione 5.2.9.

Cicli nei grafi non orientati

I concetti di *walk chiuso*, *circuito* e *ciclo* sono definiti come per i grafi orientati (Definizione 5.2.10), così come anche le nozioni di grafo ciclico e aciclico (Definizione 5.2.12). Però questi concetti acquisiscono un valore leggermente diverso.

In particolare, non è vero che l'esistenza di un walk chiuso implica l'esistenza di un circuito (e quindi di un ciclo), perché il trail corrispondente a tale walk (per il Lemma 5.5.7) potrebbe avere lunghezza 0 e quindi non essere un circuito.

Esempio 5.5.11. Si consideri il seguente grafo non orientato, costituito da due soli nodi ed un arco che li collega. La sequenza 0, 1, 0 è un walk chiuso, ma non un circuito (e quindi neppure un ciclo). Inoltre, in questo grafo ci sono solo due possibili trail con gli estremi uguali, 0 e 1, entrambi di lunghezza 0. Pertanto questo grafo è aciclico, benché ci sia un walk chiuso.



Vale invece che l'esistenza di un circuito, implica l'esistenza di un ciclo.

Proposizione 5.5.12. Sia $G = (V, E)$ un grafo non orientato e x un nodo in V . Le seguenti affermazioni sono equivalenti:

- (a) esiste un circuito che inizia e termina in x ,
- (b) esiste un ciclo che inizia e termina in x ,

Dimostrazione. Dimostriamo se (a) allora (b). Se esiste un circuito da x a x , allora esiste un trail v_0, \dots, v_k di lunghezza $k > 0$ tale che $v_0 = v_k = x$. Si noti che, essendo $k > 0$, allora esiste anche un trail v_0, \dots, v_{k-1} (eventualmente di lunghezza 0). Per il Lemma 5.5.9, esiste un path (eventualmente di lunghezza 0) da v_0 a v_{k-1} . Aggiungendo a questo path l'arco $\{v_{k-1}, v_k\} \in E$ si ottiene un ciclo da x a x .

Si osservi che se (b) allora (a) segue immediatamente dalla definizione di ciclo e circuito. ■

Si osservi che mentre in un grafo orientato esiste un ciclo da x a x se e solo se $(x, x) \in E^+$ (Proposizione 5.2.13(d)), nei grafi non orientati questo non vale poiché l'esistenza di walk chiusi non implica l'esistenza di cicli. In particolare, anche la caratterizzazione dei grafi orientati aciclici fornita dalla Proposizione 5.2.14 non vale più.

Connettività

Le nozioni di connettività per i grafi non orientati coincidono con quelle per i grafi orientati: (a) un grafo non orientato $G = (V, E)$ è *fortemente connesso* se il grafo orientato corrispondente $\tilde{G} = (V, \tilde{E})$ è fortemente connesso e (b), una *componente fortemente connessa* di G è la stessa cosa di una componente fortemente connessa di \tilde{G} . Con queste definizioni, tutti i risultati di connettività che valgono per i grafi orientati valgono anche per i grafi non orientati.

Però, essendo la relazione \tilde{E} simmetrica, definizioni e risultati possono essere notevolmente semplificati. È quindi opportuno mostrare, seppur brevemente, tale semplificazione.

Definizione 5.5.13 (grafo connesso). Un grafo non orientato $G = (V, E)$ si dice CONNESSO se per ogni coppia di nodi $(u, v) \in V \times V$ esiste un walk da u a v .

Definizione 5.5.14 (componente连通). Sia $G = (V, E)$ un grafo non orientato. Un sottoinsieme non vuoto di nodi $U \subseteq V$ si dice una **COMPONENTE CONNESSA** se:

1. Per ogni coppia di nodi $(x, y) \in U \times U$ esiste un walk da x a y ;
2. Se $U' \subseteq V$ soddisfa la proprietà 1. e $U \subseteq U'$, allora $U = U'$.

Esempio 5.5.15. Il grafo di Figura 5.4(a) è connesso, mentre quello di Figura 5.6 non lo è: infatti non esiste un cammino, per esempio, dal nodo 2 al nodo 8.

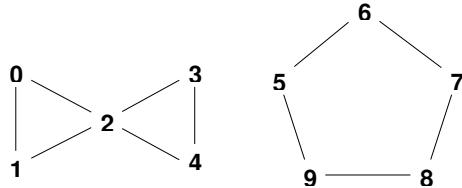


Figura 5.6: Un grafo non connesso, contenente due componenti connesse $\{0, 1, 2, 3, 4\}$ e $\{5, 6, 7, 8, 9\}$

Dal momento che \tilde{E} è una relazione simmetrica, \tilde{E}^* è una relazione di equivalenza. Si ricorda che $(x, y) \in \tilde{E}^*$ se e solo se esiste un walk da x a y (Proposizione 5.5.10). Pertanto x e y appartengono alla stessa componente connessa se e solo se appartengono a \tilde{E}^* . Pertanto le classi di equivalenza di \tilde{E}^* sono esattamente le componenti connesse di G .

Proposizione 5.5.16. Sia $G = (V, E)$ un grafo non orientato e siano $x, y \in V$ due nodi. Allora vale che:

1. G è connesso se e solo se $V \times V \subseteq \tilde{E}^*$;
2. $(x, y) \in \tilde{E}^*$ se e solo se x ed y appartengono alla stessa componente connessa.

5.6 Alberi

Definizione 5.6.1 (albero). Un ALBERO è un grafo non orientato connesso, aciclico e non vuoto. I nodi alle estremità, ovvero i nodi di grado 1, vengono detti FOGLIE mentre gli altri nodi sono talora indicati come INTERNI.

Una FORESTA è un grafo non orientato e aciclico (eventualmente non connesso). Ogni componente connessa di una foresta è un albero.

Esempio 5.6.2. Il grafo in Figura 5.7 è un albero. Le foglie sono i nodi 1, 3, 4, 5, 7, 9 e i nodi interni sono 0, 2, 6, 8.

Facciamo alcune osservazioni che esemplificano delle proprietà che dimostreremo essere valide per tutti gli alberi. Per prima cosa si osservi che l'albero ha $n = 10$ nodi e $m = 9$ archi, quindi vale $m = n - 1$. Vediamo anche che, presi due nodi x e y , c'è un solo path che li collega: per esempio, $x = 0$ e $y = 8$ hanno 0, 2, 6, 8 in quanto gli altri walk passano necessariamente due volte per lo stesso nodo. Proviamo adesso a togliere un qualunque arco, per esempio 26: il risultato è che l'albero non è più connesso, e questo vale per ogni arco. Se invece proviamo ad aggiungere un arco tra due nodi che non sono vicini, il grafo diventa ciclico: per esempio, collegando 0 e 8, chiudiamo il path 0, 2, 6, 8 che li collega.

Osservazione 5.6.3. Gli alberi sono strutture fondamentali in informatica: essi consentono di rappresentare relazioni gerarchiche e sono alla base di alcune delle più importanti strutture di dati. Il nome deriva dall'idea che la struttura di questo tipo di grafo possa essere immaginata simile alle diramazioni di un albero, come mostrato in Figura 5.7: tali diramazioni si originano in un nodo e non vanno mai a chiudersi su un altro nodo, perché altrimenti formerebbero un ciclo. Un esempio di albero nell'accezione informatica che tutti conosciamo è l'albero genealogico: se immaginiamo un capostipite della famiglia, come un bis-nonno, e di tracciare linee verso i figli, poi da loro ai nipoti, e così via. Otteniamo (tipicamente) un grafo senza cicli, dove le foglie sono le ultime generazioni

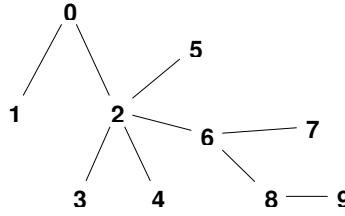


Figura 5.7: Un albero, ovvero un grafo senza cicli.

che ancora non hanno figli. Inoltre, come vedremo in seguito, gli alberi sono implicitamente alla base di alcune fondamentali tecniche di ragionamento.

Dimostriamo ora le proprietà osservate empiricamente nell'esempio precedente.

Proposizione 5.6.4. *Dato un albero $G = (V, E)$ con $n = |V|$ nodi, valgono le seguenti proprietà:*

- (a) *Se $n \geq 2$, allora G ha almeno una foglia (ovvero un nodo di grado 1).*
- (b) *G ha esattamente $n - 1$ archi, cioè $|E| = n - 1$.*
- (c) *Per ogni coppia di nodi distinti $x, y \in V$, esiste un unico path da x a y .*
- (d) *Per ogni arco $x y \in E$, la rimozione di $x y$ rende il grafo non connesso.*
- (e) *Per ogni coppia di nodi distinti $x, y \in V$ tale che $x y \notin E$, l'aggiunta dell'arco $x y$ crea un ciclo, cioè il grafo $G' = (V, E \cup \{x y\})$ è ciclico.*

Dimostrazione.

- (a) Procediamo per induzione su $|V|$. [Caso base] Se $|V| = 2$, poiché G è connesso i due nodi sono connessi da un arco, e non ci possono essere altri archi. Quindi entrambi i nodi sono foglie. [Passo induttivo] Come ipotesi induttiva assumiamo che ogni albero con n nodi abbia una foglia, per un certo $n \geq 2$, e mostriamo che lo stesso vale per qualunque albero con $n + 1$ nodi. Sia $G = (V, E)$ un albero con $n + 1$ nodi, e sia $x \in V$ un nodo arbitrario di G . Chiaramente $d_x > 0$, perché ogni albero è connesso. Se $d_x = 1$ allora G ha una foglia e concludiamo. Se $d_x > 1$, sia v_1, \dots, v_k un qualunque ordinamento di $N(x)$, il vicinato di x . Consideriamo il grafo $G' = (V', E')$ ottenuto da G togliendo il nodo x e tutti gli archi adiacenti ad esso, e aggiungendo gli archi $v_1 v_2, v_2 v_3, \dots, v_{k-1} v_k$. È facile mostrare che G' è un albero (in particolare è connesso e non ha cicli), e che il grado di ogni nodo in G' è maggiore o uguale del grado dello stesso nodo in G . Per ipotesi induttiva, poiché G' ha n nodi allora ha una foglia y , ma allora y è una foglia anche in G .
- (b) Procediamo per induzione su n . [Caso base] Se $n = 1$ l'albero è costituito da un solo nodo e nessun arco, quindi il numero di archi è $0 = n - 1$. [Passo induttivo] Come ipotesi induttiva assumiamo che la proprietà valga per tutti gli alberi con al massimo n nodi, e consideriamo un generico albero G con $n + 1$ nodi. Poichè $n \geq 1$ per ipotesi, abbiamo che $n + 1 > 1$, e quindi esiste almeno una foglia per il punto (a). Se rimuoviamo da G tale foglia e il suo unico arco incidente, otteniamo un nuovo albero G' con n nodi. Per ipotesi induttiva G' ha $n - 1$ archi, e quindi G ne ha esattamente n , cioè il suo numero di nodi ($n + 1$) meno 1.
- (c) Essendo G connesso, esiste un walk da x a y e quindi un path P , per i Lemmi 5.5.7 e 5.5.9. Per dimostrare che P è unico, procediamo per assurdo. Supponiamo che P non sia unico, ma esista un altro path $P' \neq P$. Siccome entrambi partono da x e arrivano in y , sia x' l'ultimo nodo in comune tra P e P' prima che divergano partendo da x . Inoltre, sia y' il primo nodo in P , successivo a x' , che viene attraversato anche da P' . Notiamo che x' e y' esistono sempre (ma che vada $x' = x$ e $y' = y$). Ora possiamo andare da x' a y' usando parte di P e tornare indietro da y' a x' usando parte di P' a ritroso. Abbiamo cioè un ciclo, contraddicendo l'ipotesi che G sia aciclico.

- (d) Anche qui procediamo per assurdo, ipotizzando che la rimozione di un arco $xy \in E$ da G lasci $G' = (V, E \setminus \{xy\})$ connesso: vuol dire che esiste un walk da x a y in G' , e quindi un path P da x a y . Poiché ogni arco di G' appartiene anche a G , deduciamo che P esiste anche in G . Ma poiché G contiene anche xy abbiamo un ciclo formato da P e xy , che contraddice l'ipotesi che G è aciclico.
- (e) Questa proprietà deriva immediatamente dalle precedenti. Sappiamo che esiste un path P da x a y : se aggiungiamo l'arco xy , otteniamo $G' = (V, E \cup \{xy\})$. Poiché tutti gli archi di G appaiono anche in G' , osserviamo che P esiste anche in G' : l'aggiunta di xy chiude P e crea un ciclo.

■

Quando rappresentiamo una gerarchia tramite un albero, solitamente identifichiamo un nodo di ingresso tra quelli dell'albero.

Definizione 5.6.5 (albero radicato). *Un ALBERO RADICATO $G = (V, E, r)$ è costituito da un albero $G = (V, E)$ e da un suo nodo $r \in V$ chiamato RADICE r . Dato un nodo $y \neq r$, i nodi lungo l'unico cammino che collega y a r vengono chiamati gli ANTENATI di y , e il primo (quello adiacente a y) è detto il GENITORE di y . Simmetricamente, y viene detto DISCENDENTE dei suoi antenati, e FIGLIO del suo nodo genitore.*

Dato un albero radicato $G = (V, E, r)$ e un nodo $r' \in V$, il SOTTOALBERO di G con radice r' è l'albero radicato $G' = (V', E', r')$, in cui $V' \subseteq V$ contiene r' e tutti i suoi discendenti in G , e $E' \subseteq E$ contiene tutti gli archi di G tra nodi in V' (formalmente, $E' = E \cap \mathcal{P}_2(V')$).

Esempio 5.6.6. Consideriamo l'albero in Figura 5.7, e scegliamo il nodo $r = 0$ come radice. Possiamo osservare che la foglia 9 è un discendente di 2, ma non di 5, in quanto solamente 2 si trova sul cammino da 9 verso la radice 0. Gli antenati di 4 sono 2 e 0, e il suo genitore è 2.

Il sottoalbero con radice 6 contiene i nodi 6, 7, 8 e 9, e gli archi con entrambi gli estremi in questo insieme.

Osservazione 5.6.7. Si noti come l'inclusione tra insiemi sia collegata all'ordine parziale rappresentato da un albero. Associamo a ogni nodo il suo sottoalbero, visto come insieme di nodi. Per esempio, al nodo 2 associamo l'insieme $\{2, 3, 4, 5, 6, 7, 8, 9\}$ in Figura 5.7. Vale la relazione che u è antenato di v se e solo se l'insieme associato a u contiene quello associato a v . Il nodo 2 è antenato del nodo 8 perché $\{2, 3, 4, 5, 6, 7, 8, 9\}$ contiene $\{8, 9\}$; vice versa, i nodi 8 e 4 non sono antenati l'uno dell'altro perché i loro insiemi sono disgiunti. In generale, notiamo che gli insiemi associati a due nodi arbitrari o sono uno contenuto nell'altro oppure sono disgiunti.

Nella definizione di albero radicato che abbiamo presentato, fissato un qualunque nodo $x \in V$ possiamo determinare l'insieme dei suoi figli come $N(x)$ se x è la radice, e come $N(x) \setminus \{y\}$ se x non è radice e y è il suo genitore. In ogni caso i figli costituiscono un insieme, quindi non sono ordinati.

Spesso risulta utile, nella modellazione di problemi, avere un ordinamento definito sui figli di ogni nodo. Questo si può ottenere con delle varianti della definizione di albero, chiamati *ordinali* e *cardinali*.

Definizione 5.6.8 (albero ordinale e cardinale). *Un albero radicato si dice ORDINALE se per ciascun nodo interno è definito un ordinamento totale tra i suoi figli. Un albero radicato si dice CARDINALE o k-ario se ogni nodo interno ha esattamente k figli, alcuni dei quali possono essere vuoti (indicati con null). I figli sono numerati e chiamati figlio 0, figlio 1, ..., figlio $k - 1$. L'albero è PIENO se ogni nodo interno ha tutti e k i figli non vuoti.*

Un caso speciale e molto importante è quando $k = 2$: in tal caso l'albero viene detto BINARIO, dove il primo figlio viene chiamato FIGLIO SINISTRO e il secondo figlio viene chiamato FIGLIO DESTRO.

Esempio 5.6.9. Possiamo interpretare l'albero radicato di Figura 5.7 (con radice 0) come un albero ordinale, usando la convenzione che i figli di ogni nodo sono ordinati in senso anti-orario. Quindi per esempio il nodo 2 ha come primo figlio 3, secondo figlio 4, terzo figlio 6 e quarto figlio 5.

Si noti che gli alberi cardinali e gli alberi ordinali sono strutture dati differenti, nonostante l'apparente somiglianza. Per esempio, i due alberi radicati (con radice A) nella Figura 5.8 sono

diversi se considerati come alberi cardinali, in quanto il nodo D è il figlio sinistro del nodo B nel primo caso ed è il figlio destro nel secondo caso, mentre tali alberi sono indistinguibili come alberi ordinali in quanto D è in entrambi il primo e unico figlio di B.

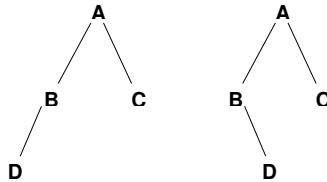


Figura 5.8: Due alberi cardinali diversi che sono indistinguibili come alberi ordinali.

5.7 Cammini euleriani e cammini hamiltoniani

Sin dagli inizi dello sviluppo della teoria dei grafi i ricercatori si sono posti il problema di determinare in quali casi in un grafo possono esistere dei cammini con delle proprietà interessanti. Consideriamo qui due esempi specifici che hanno anche una rilevanza storica: i cammini che attraversano una e una sola volta tutti gli archi di un grafo, studiati per la prima volta nel 1736 da Eulero, e i cammini che attraversano una e una sola volta tutti i nodi di un grafo, introdotti nel XIX secolo dal matematico William R. Hamilton.

È opportuno sottolineare che i concetti di cammino euleriano e cammino hamiltoniano possono essere definiti sia per grafi orientati che non orientati, anche se nell'esposizione che segue li abbiamo definiti solo per i grafi non orientati.

Cammini euleriani

Definizione 5.7.1 (circuito e trail euleriano). *Dato un grafo non orientato连通 G = (V, E), un CIRCUITO EULERIANO per G è un circuito che attraversa tutti gli archi in E una e una sola volta. Analogamente, un TRAIL EULERIANO (più comunemente chiamato PERCORSO EULERIANO) è un trail per G che attraversa tutti gli archi in E una e una sola volta.*

Ricordiamo che per le definizioni viste un trail può avere il nodo di partenza diverso da quello di destinazione, mentre un circuito no.

Esempio 5.7.2. *Un esempio famoso è quello di disegnare una cassetta su un foglio di carta senza mai sollevare la penna. Nella Figura 5.4(a), un trail euleriano dal nodo 4 al nodo 1 è dato dalla sequenza di nodi 4, 1, 2, 3, 5, 0, 1, 3, 4, 5, 1 che attraversa tutti gli archi esattamente una volta.*

Eulero si pose il problema di trovare un circuito che soddisfacesse tale proprietà perché nella città di Königsberg c'era un parco naturale ricco di torrenti che formavano un'isola A e dividevano il resto del parco in tre zone B, C, D, come illustrato nel disegno originario di Eulero e riportato in Figura 5.9(a), collegandole con sette ponti a, b, c, d, e, f. Le persone si divertivano a cercare di attraversare tutti i ponti una e una sola volta, tornando al loro punto di partenza.

Eulero formulò il problema come un grafo connesso $G = (V, E)$ in cui trovare quello che oggi chiamiamo appunto un *circuito euleriano*. Considerando il disegno annotato nella Figura 5.9(b), le zone e i ponti diventano gli 11 nodi del grafo. Per ogni zona $Z \in \{A, B, C, D\}$, creiamo un arco Zp per ogni ponte $p \in \{a, b, c, d, e, f\}$ che congiunge Z alle altre zone. Per esempio, abbiamo gli archi Cc, Cd, Cg mentre non possiamo avere Ca o CA per ovvi motivi geografici e non possiamo avere ca perché occorre passare per una zona tra i due punti. Gli archi sono non orientati perché ciascun collegamento può essere percorso in entrambe le direzioni. È quindi possibile attraversare tutti i ponti una e una sola volta come richiesto se e solo se G ammette un circuito euleriano.

Eulero dimostrò che una condizione necessaria e sufficiente affinché ciò sia possibile è che G sia connesso e i suoi nodi abbiano tutti grado pari: pertanto il grafo G nella Figura 5.9 non contiene un circuito euleriano, in quanto presenta quattro nodi di grado dispari. Eulero dimostrò inoltre che un grafo contiene un percorso (trail) euleriano con estremi diversi se e solo se esattamente due

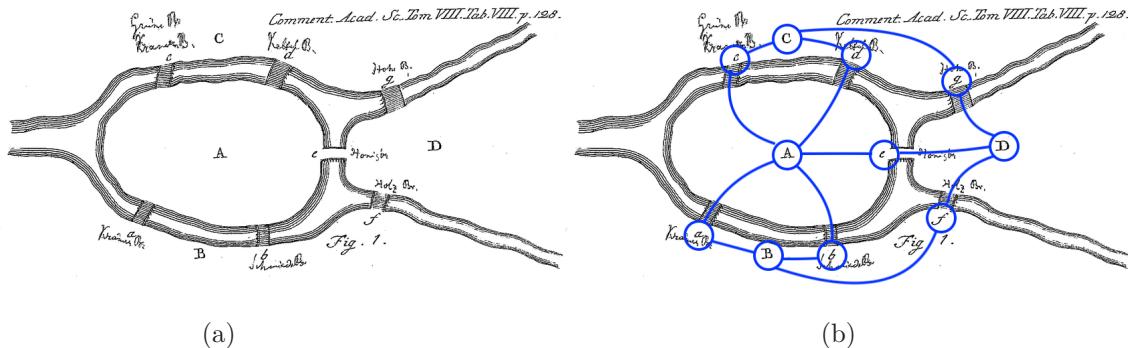


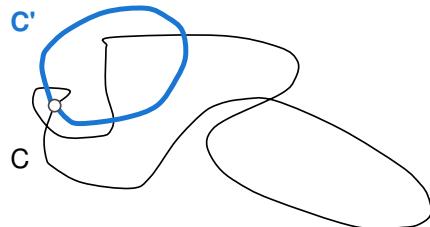
Figura 5.9: (a) Il problema dei ponti di Königsberg nella figura originale di Eulero. (b) La sua rappresentazione mediante un grafo.

nodi hanno grado dispari: quindi purtroppo il grafo G nella Figura 5.9 non contiene neanche un percorso euleriano.

Teorema 5.7.3. *Dato un grafo non orientato connesso G , esiste un circuito euleriano se e solo se ogni nodo ha grado pari.*

Dimostrazione. (*Solo se*) Se esiste un circuito euleriano, vuol dire che esiste un circuito che utilizza tutti gli archi di G esattamente una volta. Ogni volta che il circuito attraversa un nodo utilizza due archi mai traversati prima, ovvero il numero di archi incidenti in un nodo (cioè il suo grado) è esattamente il doppio del numero di volte che il circuito passa per il nodo, quindi il grado è pari.

(*Se*) Se ogni nodo ha grado pari, dimostriamo che esiste un circuito euleriano in G . Attraversiamo G a partire da un nodo r qualsiasi, marcando di volta in volta l'arco utilizzato per spostarsi nel nodo successivo: possiamo utilizzare solo archi non marcati (cioè mai traversati prima) e quando attraversiamo un arco lo marchiamo per evitare di attraversarlo nuovamente. Osserviamo che in questo modo attraversiamo un trail (non ci sono archi traversati più volte) che prima o poi ritorna in r , creando un circuito. Infatti, se così non fosse, terminerebbe in un nodo $u \neq r$ che non ha altri archi incidenti: ma u non può essere un nodo diverso da r in quanto, se fosse stato attraversato già $k \geq 0$ volte, il suo numero di archi incidenti sarebbe $2k + 1$, quindi u avrebbe grado dispari, contraddicendo l'ipotesi. Possiamo quindi dedurre che abbiamo trovato un circuito C che passa per r . Se questo circuito attraversa tutti gli archi di G allora C è euleriano e possiamo concludere la nostra argomentazione. Altrimenti utilizziamo l'ipotesi che G è connesso: in tal caso, deve esserci per forza un nodo r' in C che ha un arco incidente non marcato (altrimenti G non sarebbe connesso). L'idea è ripetere per r' quanto fatto per r , trovando così un altro circuito C' che passa per r' , utilizzando la stessa argomentazione esposta poco fa. Notiamo che C e C' sono disgiunti sugli archi, cioè non possono attraversare uno stesso arco, perché marchiamo gli archi man mano che li attraversiamo. Inoltre C e C' si intersecano sicuramente in r' per costruzione (magari anche in ulteriori nodi, ma questo non è rilevante). Utilizziamo allora la seguente osservazione: se due circuiti disgiunti si intersecano in almeno un nodo, allora formano un unico circuito, come illustrato in figura:



Indichiamo con $C + C'$ il circuito risultante: se attraversa tutti gli archi, abbiamo finito; altrimenti troviamo un altro circuito C'' come sopra e lo componiamo con $C + C'$ ottenendo il circuito

$C + C' + C''$. Procedendo in questo modo, continuiamo a comporre circuiti per ottere un unico circuito, e prima o poi attraversiamo tutti gli archi. Quindi esiste un circuito euleriano in G . ■

Lasciamo come esercizio per il lettore la dimostrazione del seguente corollario, che segue facilmente dal teorema precedente.

Corollario 5.7.4. *Dato un grafo non orientato connesso G e due nodi distinti $x, y \in V_G$, esiste un trail euleriano dal nodo x al nodo y se e solo se d_x e d_y sono gli unici gradi dispari.*

Esercizio 5.7.5. *Dimostrare il Corollario 5.7.4.*

Cammini hamiltoniani

Consideriamo adesso una nozione apparentemente simile e di grande importanza per l'informatica, il cui nome deriva dal matematico William R. Hamilton che la introdusse nel XIX secolo.

Definizione 5.7.6 (ciclo e path hamiltoniano). *Dato un grafo non orientato e connesso $G = (V, E)$, un CICLO HAMILTONIANO è un ciclo che attraversa tutti i nodi in V una e una sola volta. Un PATH HAMILTONIANO (più comunemente chiamato CAMMINO HAMILTONIANO) è un path che attraversa tutti i nodi in V una e una sola volta.*

Esempio 5.7.7. *Nel grafo non orientato di Figura 5.4(a), un ciclo hamiltoniano è dato dalla sequenza di nodi $0, 1, 2, 3, 4, 5, 0$ che attraversa tutti i nodi esattamente una volta (chiaramente il nodo 0 non conta due volte perché è sia di partenza che di arrivo).*

Si noti che un grafo può avere in generale più cicli hamiltoniani. Trovare un path hamiltoniano in sostanza consiste nel trovare una permutazione dei nodi in V che dia luogo a un path.

L'apparente analogia con i circuiti euleriano purtroppo termina qui: attraversare tutti gli *archi* una sola volta nel circuito euleriano, di contro ad attraversare tutti i *nodi* una sola volta nel ciclo hamiltoniano. Non esiste al momento una caratterizzazione per garantire l'esistenza o meno di un ciclo hamiltoniano in G , analoga a quanto enunciato nel Teorema 5.7.3. Il ciclo hamiltoniano ha struttura combinatoria più complessa e sfuggente: trovare un ciclo hamiltoniano può risultare più arduo come compito e fa parte di una famiglia importante di problemi, chiamati *NP-completi*, la cui trattazione sarà argomento di un corso successivo.

Il problema del commesso viaggiatore

Il famoso *problema del commesso viaggiatore* consiste nell'individuare, su di una mappa stradale in cui ogni strada è etichettata con la lunghezza in chilometri, un cammino che permetta al commesso viaggiatore di attraversare tutte le città e di tornare a casa percorrendo il minor numero possibile di chilometri.

Il lettore può convincersi facilmente che questo problema può essere formulato come quello di trovare un *ciclo hamiltoniano di peso minimo* in un grafo pesato.

Definizione 5.7.8 (peso di ciclo hamiltoniano). *Dato un grafo pesato $G = (V, E, L)$, il peso di un ciclo hamiltoniano $H = v_0, v_1, \dots, v_k$ (quindi con $v_0 = v_k$) è la somma dei pesi degli archi attraversati da H :*

$$\text{peso}(H) = \sum_{i=1}^k L(v_{i-1}, v_i)$$

Esempio 5.7.9. *Consideriamo il grafo pesato non orientato mostrato in Figura 5.10(a), dove i pesi sono interi positivi. Prendendo il ciclo hamiltoniano $H = 0, 1, 2, 3, 4, 5, 0$, mostrato in Figura 5.10(b), otteniamo $\text{peso}(H) = 7 + 4 + 1 + 1 + 6 + 3 = 22$ ma non è minimo. Al commesso viaggiatore conviene infatti percorrere quello di Figura 5.10(c) $H^* = 0, 2, 3, 4, 5, 1, 0$ con $\text{peso}(H^*) = 2 + 1 + 1 + 6 + 4 + 7 = 21$.*

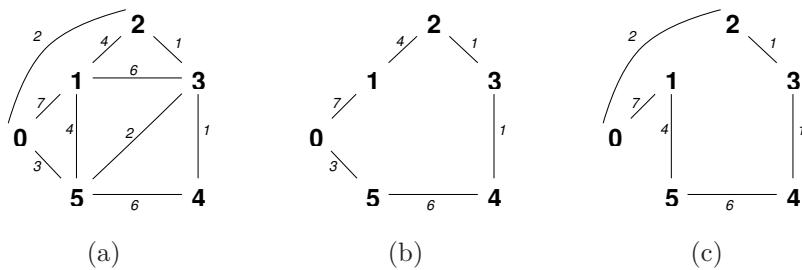


Figura 5.10: (a) Un grafo pesato. (b) Un ciclo hamiltoniano. (c) Un ciclo hamiltoniano di peso minimo.

5.8 Distanza su grafi

La definizione matematica di *distanza* caratterizza con opportune proprietà il concetto intuitivo di “distanza” che usiamo nel linguaggio quotidiano. In questa sezione prima introduciamo la definizione generale di distanza, poi vediamo come essa si può applicare ai grafi.

Il concetto di distanza che normalmente usiamo è la cosiddetta *distanza euclidea*: intuitivamente, la distanza tra due oggetti (che possiamo pensare puntiformi) è la lunghezza del segmento di retta che li unisce. La definizione matematica di distanza fissa alcune proprietà intuitive che valgono per la distanza euclidea, ma non prescrive come una distanza sia effettivamente calcolata. Quindi si possono avere una varietà di distanze definite in modo diverso, di cui vedremo solo alcuni semplici esempi.

Definizione 5.8.1 (distanza). Una DISTANZA (METRICA) su un insieme A è una funzione $d : A \times A \rightarrow \mathbb{R}$ che soddisfa le seguenti proprietà per ogni $x, y, z \in A$:

1. $d(x, y) \geq 0$
 2. $d(x, y) = 0$ se e solo se $x = y$;
 3. $d(x, y) = d(y, x)$ (simmetria);
 4. $d(x, y) \leq d(x, z) + d(z, y)$ (disuguaglianza triangolare).

Una funzione $d : A \times A \rightarrow \mathbb{R}$ che soddisfa tutte queste proprietà tranne la simmetria è chiamata DISTANZA QUASI-METRICA.

Si noti come siano naturali queste proprietà, che sono ovviamente soddisfatte dalla distanza euclidea: 1) la distanza tra due elementi è un reale positivo o nullo; 2) un elemento ha distanza nulla da se stesso, e solo da se stesso; 3) la distanza tra due elementi non cambia a seconda dell'ordine in cui li consideriamo; e 4) in un triangolo la somma delle lunghezze di due lati è maggiore o uguale a quella del terzo lato.

Nel caso di grafi non orientati, possiamo definire una distanza sui nodi in V .

Definizione 5.8.2 (Distanza su grafo). *Sia $G = (V, E)$ un grafo non orientato connesso. Dati due nodi $x, y \in V$, la loro DISTANZA $d(x, y)$ è la lunghezza del walk più breve tra x e y , chiamato WALK MINIMO (o più comunemente cammino minimo).*

Esempio 5.8.3. Nel grafo in Figura 5.4 la distanza tra i nodi 0 e 4 è 2, in quanto esistono i walk minimi $0, 1, 4$ e $0, 5, 4$.

Nella Definizione 5.8.2 abbiamo introdotto una definizione esplicita di distanza, ma questo ha senso solo se tale definizione soddisfa le condizioni generali della Definizione 5.8.1. Questo è quanto dimostriamo nella prossima proposizione.

Proposizione 5.8.4. La distanza su grafi della Definizione 5.8.2 è una distanza metrica, cioè soddisfa le condizioni della Definizione 5.8.1.

Dimostrazione. Iniziamo a dimostrare che $d : V \times V \rightarrow \mathbb{R}$ è una funzione ben definita. Infatti sia $W(x, y)$ l'insieme dei walk in G da x a y , e $\hat{W}(x, y) \subseteq \mathbb{N}$ l'insieme delle loro lunghezze, cioè $\hat{W}(x, y) = \{|w| \mid w \in W(x, y)\}$. Poiché G è connesso, esiste almeno un walk da x a y e quindi $\hat{W}(x, y)$ è un insieme non vuoto di naturali, e per il Principio del Buon Ordinamento ha un minimo.⁵ Quindi $d(x, y)$ è ben definito, e questo vale per tutte le coppie di nodi.

Vediamo ora che per tutti i nodi $x, y, z \in V$ valgono le seguenti proprietà:

1. $d(x, y) \geq 0$, infatti la lunghezza di qualunque walk è un numero naturale;
2. $d(x, y) = 0$ se e solo se $x = y$: infatti tra x e y esiste un walk di lunghezza 0 se e solo se x e y sono lo stesso nodo;
3. $d(x, y) = d(y, x)$ (simmetria): infatti la lunghezza di un walk non dipende dalla direzione in cui lo percorriamo;
4. $d(x, y) \leq d(x, z) + d(z, y)$ (disuguaglianza triangolare): se per assurdo fosse $d(x, y) > d(x, z) + d(z, y)$, avremmo trovato da x a y un walk più breve del minimo, che passa per z , ma allora $d(x, y)$ non è la lunghezza del walk minimo.

■

Presentiamo ora una definizione *ricorsiva* di distanza, equivalente alla Definizione 5.8.2.⁶ L'idea alla base è che dato un grafo connesso $G = (V, E)$, un walk minimo che parte da x verso $y \neq x$, deve necessariamente passare da un suo vicino in $N(x)$. Non sapendo quale sia questo vicino, consideriamo tutti i candidati $z \in N(x)$. Essendo il grafo connesso, z avrà un walk verso y : se prendiamo il più breve tra tali walk, otteniamo il walk minimo da x a y aggiungendo l'arco xz a tale walk da z a y . In altre parole, se togliamo il primo arco di un walk minimo, otteniamo ancora un walk minimo. Queste osservazioni garantiscono che la definizione seguente di distanza è equivalente a quella della Definizione 5.8.2

Definizione 5.8.5 (Distanza su grafo, ricorsivamente). *Sia $G = (V, E)$ un grafo non orientato connesso. Dati due nodi $x, y \in V$, la loro DISTANZA $d(x, y)$ è definita ricorsivamente come:*

1. [Clausola base] $d(x, y) = 0$, se $x = y$;
2. [Clausola ricorsiva] $d(x, y) = 1 + \min\{d(z, y) \mid z \in N(x)\}$, altrimenti.

La Definizione 5.8.2, basata sulla lunghezza del cammino minimo, si può applicare anche a *grafoi orientati fortemente connessi*. Tuttavia in questo caso la funzione $d : V \times V \rightarrow \mathbb{R}$ non soddisfa la proprietà di simmetria della Definizione 5.8.1 perché il walk minimo da x a y può avere lunghezza diversa da quello da y o x . Quindi nel caso di grafi orientati questa funzione è una distanza quasi-metrica.

Basandosi sul concetto di distanza, la seguente nozione è molto usata nello studio delle reti di calcolatori e quelle sociali.

Definizione 5.8.6 (diametro di grafo). *Il diametro di un grafo G (orientato o meno) è la massima distanza tra coppie di nodi:*

$$\text{diam}(G) = \max_{x, y \in V} d(x, y)$$

Nel caso degli alberi, si eredita la distanza dei grafi non orientati.

Definizione 5.8.7 (profondità e altezza di nodi in alberi). *Dato un albero radicato, la PROFONDITÀ di un nodo x è la sua distanza $d(x, r)$ dalla radice r , e l'ALTEZZA di x è la massima distanza tra x e le sue foglie discendenti. L'ALTEZZA di un albero radicato è l'altezza della sua radice. Un albero cardinale si dice COMPLETO se è pieno e le foglie sono tutte alla stessa distanza dalla radice.*

⁵ Il Principio del Buon Ordinamento, che assumiamo essere valido, stabilisce appunto che ogni insieme non vuoto di naturali ha un minimo.

⁶ Parleremo più a fondo di funzioni ricorsive nella Sezione 7.5.

La radice r ha profondità zero e quella degli altri nodi è sempre pari a uno più della profondità del genitore. Analogamente, ogni foglia ha altezza zero e ogni nodo interno ha altezza pari a uno più il massimo tra le altezze dei figli.

Infine, per i grafi pesati aventi pesi *non negativi*, si considera la *somma* dei pesi lungo ciascun cammino, piuttosto che la loro lunghezza. Per cammino minimo si intende quindi il *cammino pesato avente somma minima*: è importante osservare che un cammino minimo in un grafo pesato non è necessariamente quello col minor numero di archi. Per esempio in una rete stradale, è naturale pensare al peso di un arco come la lunghezza del tratto di strada corrispondente, e la lunghezza di un cammino è la somma dei pesi di tutti gli archi su di esso. Si può immediatamente verificare che la distanza pesata è una distanza metrica nei grafi non orientati e una distanza quasi-metrica nei grafi orientati. È naturalmente definito il diametro, essendo la distanza massima tra coppie di nodi. Invitiamo il lettore a pensare cosa può succedere con i pesi negativi: per esempio, se siamo in presenza di un ciclo in cui tutti gli archi hanno peso negativo.

5.9 Isomorfismo

Nelle rappresentazioni discusse finora abbiamo assegnato una numerazione agli n nodi del grafo G . È possibile che G stesso venga rappresentato altrove con un'altra numerazione di tali n nodi. Per esempio, potrebbero esserci due applicazioni che costruiscono lo *stesso* grafo ma con numerazione *diversa* dei nodi: potremmo ricostruire una rete sociale a partire da certi individui oppure da altri, producendo due grafi che differiscono solo per come sono numerati i nodi. Come possiamo renderci conto che i due grafi ottenuti hanno la stessa forma?⁷

Supponiamo per esempio di avere i due grafi illustrati in Figura 5.11; ai fini della nostra discussione, chiamiamo $G_1 = (V_1, E_1)$ quello a sinistra e $G_2 = (V_2, E_2)$ quello a destra. Se guardiamo le loro matrici d'adiacenza e le loro liste d'adiacenza, queste risultano essere differenti. Possiamo però osservare che G_1 e G_2 hanno alcune similità: hanno lo stesso numero di nodi e archi ($|V_1| = |V_2|$ e $|E_1| = |E_2|$); anche i gradi dei nodi sono uguali.

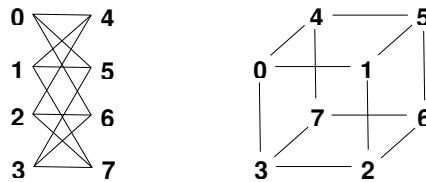


Figura 5.11: Due grafi isomorfi.

Va però detto che G_1 e G_2 sono apparentemente distinti. Per accertarci che abbiano la stessa forma, non basta la nozione di uguaglianza, ma ne occorre una più robusta.

Diremo quindi che due grafi sono *isomorfi* (dal greco, appunto “uguale forma”) quando differiscono non nella sostanza, ma solo nella numerazione dei nodi.

Nel nostro caso, i due grafi in Figura 5.11 sono in effetti isomorfi. Non è intuitivo guardando l’immagine, ma un modo per accorgersi di questo è proprio quello di rinumerare i nodi: in Figura 5.12 sono stati rinumerati i nodi di G_1 in modo adeguato, e risulta evidente che adesso i due grafi sono uguali perché i nodi hanno le stesse liste di adiacenza (il nodo 0 è adiacente a 4, 1, 3 in entrambi i disegni, il nodo 1 a 0, 2, 5, e così via...). Questa rinumerazione è possibile se e solo se i due grafi di partenza sono isomorfi, anche se non è sempre facile da trovare.

Formalmente, quello che abbiamo fatto è trovare una biiezione tra i nodi di G_1 e G_2 che mette in evidenza il loro essere isomorfi. Chiamiamo questa biiezione un *isomorfismo*:

Definizione 5.9.1 (isomorfismo di grafi). *Dati due grafi $G_1 = (V_1, E_1)$ e $G_2 = (V_2, E_2)$, dove $|V_1| = |V_2|$ e $|E_1| = |E_2|$, un isomorfismo tra G_1 e G_2 è una biiezione $f : V_1 \rightarrow V_2$ tra i loro nodi tale che, per ogni coppia di nodi $u, v \in V_1$, vale che $uv \in E_1$ se e solo se $f(u)f(v) \in E_2$ (cioè esiste*

⁷Per “forma” non intendiamo il disegno, ma proprio l’essenza del grafo. Spostare la posizione di uno o più nodi in un disegno cambia solo la rappresentazione grafica e non le proprietà del grafo.

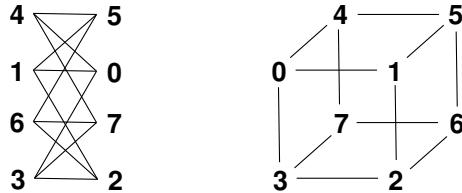


Figura 5.12: Rinumerando opportunamente i nodi di G_1 , le due figure ora rappresentano lo stesso grafo.

il corrispondente arco in entrambi i grafi, oppure non esiste in entrambi). Due grafi sono detti isomorfi se esiste un isomorfismo tra di essi.

Considerando i grafi G_1 e G_2 della Figura 5.11, è facile verificare che la seguente biiezione $f : V_1 \rightarrow V_2$ è un isomorfismo:

$$f : 0 \mapsto 4, \quad 1 \mapsto 1, \quad 2 \mapsto 6, \quad 3 \mapsto 3, \quad 4 \mapsto 5, \quad 5 \mapsto 0, \quad 6 \mapsto 7, \quad 7 \mapsto 2.$$

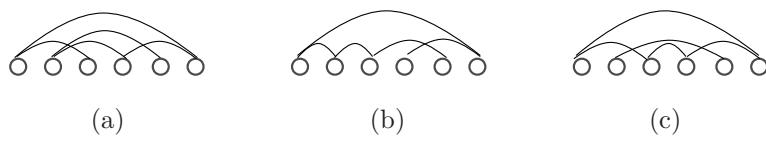
Osserviamo che $24 \in E_1$ e $f(2)f(4) = 65 \in E_2$, $25 \notin E_1$ e $f(2)f(5) = 60 \notin E_2$, e così via, per tutte le coppie di nodi in V_1 e le loro corrispondenti coppie in V_2 .

In realtà, esiste un gran numero di modi per numerare i nodi con valori distinti in $V = \{0, 1, \dots, n-1\}$ e, quindi, altrettanti modi per rappresentare lo stesso grafo (esistono cioè più biiezioni tra V_1 e V_2 che sono isomorfismi). Per esempio, anche questa biiezione \tilde{f} fornisce un isomorfismo per i grafi G_1 e G_2 della Figura 5.11.

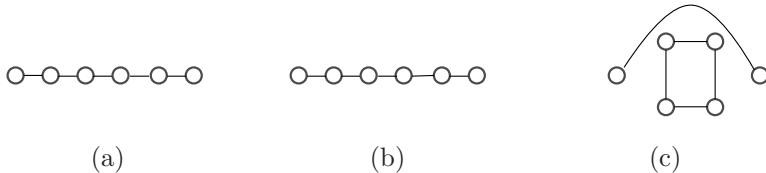
$$\tilde{f} : 0 \mapsto 5, \quad 1 \mapsto 7, \quad 2 \mapsto 2, \quad 3 \mapsto 0, \quad 4 \mapsto 6, \quad 5 \mapsto 4, \quad 6 \mapsto 1, \quad 7 \mapsto 3.$$

Una volta fissate le numerazioni dei nodi in V_1 e V_2 , non tutte le biiezioni da V_1 a V_2 sono isomorfismi: per esempio, la biiezione \hat{f} tale che $\hat{f}(0) = 3$ e $\hat{f}(4) = 5$ non è un isomorfismo poiché $04 \in E_1$ ma $\hat{f}(0)\hat{f}(4) = 35 \notin E_2$ in Figura 5.11.

L'apparente distinzione nella Figura 5.11 nasce dall'artificio di enumerare differentemente gli stessi nodi ma è chiaro che la relazione tra i nodi è la medesima se ignoriamo la loro numerazione. Purtroppo non basta verificare che i gradi dei nodi corrispondano. Nell'esempio seguente, dove la numerazione dei nodi non è riportata per brevità, osserviamo che i tre grafi hanno lo stesso numero di nodi e archi, e i gradi dei nodi possono essere messi in corrispondenza. Tuttavia, solo i primi due grafi (a) e (b) sono isomorfi, mentre il terzo non lo è con i primi due.



Quest'osservazione si evidenzia meglio se disponiamo diversamente i nodi dei tre grafi e disegniamo i corrispondenti archi.



Notiamo che l'isomorfismo è una relazione di equivalenza e che il problema di decidere se due grafi arbitrari G_1 e G_2 di n nodi sono isomorfi equivale a trovare una loro biiezione che preserva gli archi, se esiste, ed è uno dei problemi algoritmici fondamentali con molte implicazioni (per esempio, stabilire se due grafi arbitrari non sono isomorfi ha delle importanti implicazioni nella crittografia). Chiaramente il problema è interessante quando $|V_1| = |V_2|$ e $|E_1| = |E_2|$, altrimenti è facile rispondere (NO).

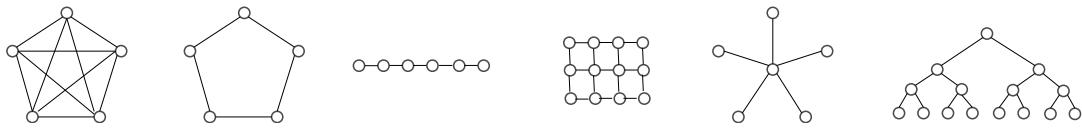


Figura 5.13: Esempi di grafi, da sinistra a destra: clique, ciclo, lineare, regolare (griglia), stella, albero completo

5.10 Altri grafi noti

Terminiamo il capitolo con una breve rassegna informale di alcune classi di grafi famosi, illustrati in Figura 5.13. Una *clique* è un grafo in cui ogni coppia di nodi è collegata da un arco. Un grafo *d-regolare* ha ogni nodo di grado al più d , ed è *completo* se ogni nodo ha grado esattamente d (la clique è un caso particolare: d -regolare completo con $d = n - 1$). Un *ciclo* è un grafo ciclico composto da un solo ciclo. Un grafo *lineare* è un grafo aciclico composto da un solo cammino semplice. Una *stella* ha un nodo universale e gli altri nodi sono foglie.

5.11 Esercizi

Esercizio 5.11.1. *Dato un grafo non orientato G con n nodi, determinare il massimo numero di archi.*

Esercizio 5.11.2. *Dato un grafo orientato $G = (V, E)$, si consideri E come una relazione su V . Mostrare che valgono le seguenti proprietà per i suoi gradi d'ingresso e d'uscita:*

1. *E è totale se e solo se $d_x^+ \geq 1$ per ogni $x \in V$;*
2. *E è univalente se e solo se $d_x^+ \leq 1$ per ogni $x \in V$;*
3. *E è iniettiva se e solo se $d_x^- \leq 1$ per ogni $x \in V$;*
4. *E è surgettiva se e solo se $d_x^- \geq 1$ per ogni $x \in V$.*

Esercizio 5.11.3. *Data la matrice di adiacenza di un grafo non orientato G , indicare cosa si ottiene sommando i valori 1 su ogni riga e colonna. E nel caso di grafo orientato?*

Esercizio 5.11.4. *Mostrare che l'inverso di un isomorfismo (da G_1 a G_2) è ancora un isomorfismo (da G_2 a G_1).*

Esercizio 5.11.5. *Dimostrare che l'isomorfismo tra grafi è una relazione di equivalenza.*

Esercizio 5.11.6. *Dimostrare che due componenti connesse distinte non possono avere un nodo in comune.*

Esercizio 5.11.7. *Escogitare un modo per rappresentare in maniera univoca e non ambigua un albero utilizzando solo delle sequenze di parentesi.*

Esercizio 5.11.8. *Dimostrare che ogni DAG ha almeno una sorgente e almeno un pozzo.*

CAPITOLO 6

Calcolo Combinatorio

Il *calcolo combinatorio* è la branca della matematica che studia i modi per raggruppare e/o ordinare secondo date regole gli elementi di un insieme finito di oggetti. Il calcolo combinatorio si interessa soprattutto di *contare* tali modi, ossia le configurazioni e solitamente risponde a domande quali “Quanti sono..”, “In quanti modi..”, “Quante possibili combinazioni..” e così via.

Nella Sezione 1.7 abbiamo introdotto il concetto di *cardinalità* di un insieme finito, cioè il numero dei suoi elementi, e nella Tabella 1.5 abbiamo anticipato alcune leggi per determinare la cardinalità di alcuni insiemi notevoli. Di seguito riportiamo una versione estesa della tabella che illustra anche le cardinalità degli insiemi $Rel(A, B)$, $Fun(A, B)$ e $Bii(A, B)$.

Esempio 1.7.2	$ \emptyset = 0$
Esempio 1.7.2	$ n = n$
Corollario 6.1.4.1	$ A \setminus B = A - A \cap B $
Corollario 6.1.4.2	$ A \cup B = A + B - A \cap B $
Proposizione 6.1.10	$ A \times B = A \cdot B $
Proposizione 6.2.9	$ \mathcal{P}(A) = 2^{ A }$
Corollario 6.3.15	$ \mathcal{P}_k(A) = \binom{ A }{k}$
Esercizio 6.2.10	$ Rel(A, B) = 2^{ A \cdot B }$
Proposizione 6.2.7	$ Fun(A, B) = B ^{ A }$
Esercizio 6.3.6	$ Bii(A, B) = \begin{cases} 0 & \text{se } A \neq B \\ A ! & \text{se } A = B \end{cases}$

Tabella 6.1: Cardinalità di alcuni insiemi notevoli

Le prime due leggi sono state discusse nell’Esempio 1.7.2, le altre verranno dimostrate in questo capitolo. Per dimostrare alcune di queste leggi faremo frequentemente appello alla *regola di biiezione*, un principio matematico che si basa sulla nozione di cardinalità di relazioni. Utilizzeremo inoltre dei concetti fondamentali del calcolo combinatorio: le *permutazioni*, le *disposizioni* e le *combinazioni*.

Infine sfrutteremo queste nozioni per “contare sui grafi”, cioè per determinare la cardinalità di nodi / archi / cammini su grafi che soddisfano certe proprietà.

In tutto il capitolo quando introdurremo un insieme assumeremo implicitamente che sia *finito*.

6.1 Operazioni su insiemi e cardinalità

Una tecnica intuitiva ed efficace per contare gli elementi di un insieme A consiste nel “partizionare” A in una famiglia di sottoinsiemi disgiunti, nel contare gli elementi di ognuno di tali sottoinsiemi, e nel sommarne le cardinalità ottenendo la cardinalità di A . Questo procedimento è formalizzato dalla seguente semplice proposizione.

Proposizione 6.1.1. *Dato un insieme A , sia $\mathcal{F} = \{A_i \mid i \in I\}$ una famiglia di sottoinsiemi di A tale che*

6. Calcolo Combinatorio

- $\bigcup_{i \in I} A_i = A$ (copertura di A);
- Per ogni coppia di indici $i, j \in I$ con $i \neq j$ si ha $A_i \cap A_j = \emptyset$ (insiemi disgiunti);

Allora allora $|A| = \sum_{i \in I} |A_i|$.

Dimostrazione. Supponiamo che A abbia cardinalità n , e chiamiamo a_1, a_2, \dots, a_n gli elementi di A , tutti distinti e in un ordine qualunque. Supponiamo per assurdo che $\sum_{i \in I} |A_i| = m$ e che $m \neq n$. Abbiamo due casi:

1. $m < n$: in questo caso deve esistere almeno un elemento $a_k \in A$ tale che a_k non appartiene a nessuno dei sottoinsiemi in \mathcal{F} , ma questo è impossibile perché \mathcal{F} è una copertura di A ;
2. $m > n$: poiché tutti gli insiemi in \mathcal{F} sono sottoinsiemi di A , deve esserci almeno un elemento di A che è stato contato due volte in $\sum_{i \in I} |A_i|$, ma questo è impossibile perché gli insiemi in \mathcal{F} sono mutuamente disgiunti.

■

Si noti che non si richiede che la famiglia \mathcal{F} sia una *partizione* di A : infatti potrebbe non soddisfare la prima condizione della Definizione 1.5.9 (\mathcal{F} potrebbe contenere degli insiemi vuoti), ma deve soddisfare le altre due condizioni. Vediamo subito un'applicazione di questo risultato.

Proposizione 6.1.2. *Per tutti gli insiemi A e B valgono i seguenti fatti:*

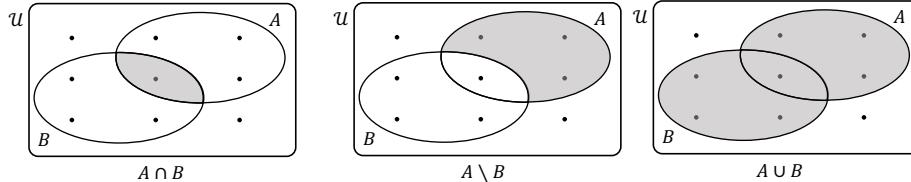
1. $|A| = |A \setminus B| + |A \cap B|$
2. $|A \cup B| = |A \setminus B| + |A \cap B| + |B \setminus A|$

Dimostrazione. Per la Proposizione 6.1.1 è sufficiente dimostrare che:

1. $A = (A \setminus B) \cup (A \cap B)$ e $(A \setminus B) \cap (A \cap B) = \emptyset$: entrambe queste proprietà sono conseguenza immediata delle Definizioni 1.3.3 e 1.3.7.
2. $A \cup B = (A \setminus B) \cup (A \cap B) \cup (B \setminus A)$; $(A \setminus B) \cap (A \cap B) = \emptyset$; $(A \setminus B) \cap (B \setminus A) = \emptyset$ e $(B \setminus A) \cap (A \cap B) = \emptyset$: anche queste proprietà sono conseguenza immediata delle Definizioni 1.3.1, 1.3.3 e 1.3.7.

■

Osservazione 6.1.3. *Le proprietà descritte nella dimostrazione precedente possono essere verificate, almeno intuitivamente, analizzando i diagrammi di Eulero-Venn mostrati nel Capitolo 1 per le tre operazioni su insiemi, e in particolare osservando che quando l'area che rappresenta un insieme viene suddivisa in più regioni, queste rappresentano sottoinsiemi mutuamente disgiunti.*



Dal risultato precedente otteniamo immediatamente le leggi (3) e (4) della Tabella 6.1.

Corollario 6.1.4. *Per tutti gli insiemi A e B valgono i seguenti fatti:*

1. $|A \setminus B| = |A| - |A \cap B|$: immediato dal primo punto della Proposizione 6.1.2;
2. $|A \cup B| = |A| + |B| - |A \cap B|$: infatti abbiamo

$$\begin{aligned}
 |A \cup B| &= |A \setminus B| + |A \cap B| + |B \setminus A| && (\text{Punto 2. della Proposizione 6.1.2}) \\
 &= |A \setminus B| + |B| && (\text{Punto 1. della Prop. 6.1.2, applicato a } B) \\
 &= |A| - |A \cap B| + |B| && (\text{Punto 1.})
 \end{aligned}$$

Intuitivamente, il punto essenziale della legge (4) della Tabella 6.1 è che nel contare gli elementi dell'unione $A \cup B$ occorre prestare attenzione a non contare *due* volte gli elementi comuni ad A e B .

Vediamo tre semplici conseguenze della legge appena vista, l'ultima della quale era stata presentata come Proposizione 1.7.5.

Corollario 6.1.5.

- $|A \cup B| \leq |A| + |B|$: infatti $|A \cup B| = |A| + |B| - |A \cap B| \leq |A| + |B|$, poiché $|A \cap B| \geq 0$;
- $|A \cup B| = |A| + |B|$ se e solo se A e B sono disgiunti: infatti A e B sono disgiunti se e solo se $|A \cap B| = 0$;
- se $B \subseteq A$, allora $|B| \leq |A|$: infatti se $B \subseteq A$ abbiamo ovviamente che $A = A \cup B$, quindi:

$$\begin{aligned} |A| &= |A \cup B| && (A = A \cup B) \\ &= |A| - |A \cap B| + |B| && (\text{Tabella 6.1, legge (4)}) \\ &= |A \setminus B| + |B| && (\text{Punto 1. del Corollario 6.1.4}) \\ &\geq |B| && (|A \setminus B| \geq 0) \end{aligned}$$

La legge (4) della Tabella 6.1, che permette di calcolare la cardinalità dell'unione di due insiemi, può essere generalizzata al caso di più di due insiemi. Nel caso di tre insiemi la formula diventa

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C| \quad (6.1)$$

Si osservi che in questa formula oltre a sottrarre la cardinalità di tutte le intersezioni di coppie di insiemi, si *aggiunge* la cardinalità dell'intersezione dei tre insiemi. Il caso generale, per un numero arbitrario ma finito di insiemi, è catturato dalla seguente proposizione di cui non mostriamo la dimostrazione (ma è importante ricordarsi l'enunciato).

Proposizione 6.1.6 (Principio di inclusione-esclusione). *Presi r insiemi S_1, S_2, \dots, S_r , abbiamo la seguente uguaglianza, dove $(-1)^i$ vale 1 se i è un numero pari e vale -1 se i è dispari:*

$$\left| \bigcup_{j=1}^r S_j \right| = \sum_{I \subseteq \{1, 2, \dots, r\}, I \neq \emptyset} (-1)^{|I|+1} \left| \bigcap_{i \in I} S_i \right| \quad (6.2)$$

In parole, possiamo ottenere la cardinalità dell'unione di r insiemi S_1, \dots, S_r nel seguente modo:

1. per ogni possibile sottoinsieme $I \subseteq \{1, 2, \dots, r\}$ consideriamo tutti gli insiemi S_i tali che $i \in I$, e calcoliamo la cardinalità n_I della loro intersezione;
2. sommiamo tutti i valori n_I così calcolati per cui la cardinalità di I è un numero dispari, e sottraiamo tutti i valori n_I per cui la cardinalità di I è un numero pari.

Esercizio 6.1.7.

1. Mostrare che la formula (6.2) è equivalente alla legge (4) della Tabella 6.1 se $r = 2$.
2. Mostrare che la formula (6.2) è equivalente alla formula (6.1) se $r = 3$.

Esercizio 6.1.8. Dimostrare la Proposizione 6.1.1 sfruttando il Principio di inclusione-esclusione.

Osservazione 6.1.9. Diverse applicazioni nell'information retrieval (e non solo) richiedono di determinare la similarità di documenti testuali, per esempio, le pagine di Wikipedia. Viene adottato il modello "bags of words", dove l'universo U è l'insieme di tutte le parole usate nel mondo, dove le parole sono lemmatizzate: in parole semplici, i verbi sono portati tutti all'infinito e le parole al singolare maschile. Ogni documento D viene rappresentato come una "bag of words", cioè un sottoinsieme $S_D \subseteq U$: semplicemente si prendono le parole distinte in D , ignorando il loro ordine di apparizione dentro D . La similarità tra due documenti viene quindi stabilita mediante i loro corrispettivi insiemi $A, B \subseteq U$. Tra gli indici adottati, c'è l'indice di Jaccard, definito come

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Se $A = B$, vale che $J(A, B) = 1$. Se $A \cap B = \emptyset$, vale che $J(A, B) = 0$. In generale, vale che $0 \leq J(A, B) \leq 1$, e quanto più alto è $J(A, B)$, tanto più simili sono gli insiemi A e B (e quindi i documenti che rappresentano).

Procedendo con l'analisi della Tabella 6.1, consideriamo ora il prodotto cartesiano $A \times B = \{(a, b) \mid a \in A, b \in B\}$ (Definizione 1.6.1).

Proposizione 6.1.10. Per tutti gli insiemi A, B vale che $|A \times B| = |A| \cdot |B|$.

Dimostrazione. Definiamo $S_a = \{(a, b) \mid b \in B\}$ per un elemento fissato $a \in A$. Valgono le seguenti osservazioni:

- (1) $|S_a| = |B|$ per costruzione perché ci sono tante coppie in S_a quanti sono gli elementi b in B .
- (2) $a \neq a'$ implica che $S_a \cap S_{a'} = \emptyset$: presa una qualunque coppia $(a, b) \in S_a$ e una qualunque coppia $(a', b') \in S_{a'}$, poiché $a \neq a'$ per ipotesi, abbiamo che vale sempre $(a, b) \neq (a', b')$ indipendentemente da b e b' .
- (3) $\cup_{a \in A} S_a = A \times B$ perché tutte le possibili coppie (a, b) , con $a \in A, b \in B$, sono ottenute in tal modo.

Da queste osservazioni è facile ottenere il risultato atteso, sfruttando la Proposizione 6.1.1. Infatti per (2) e (3) abbiamo che $\{S_a\}_{a \in A}$ è una famiglia di sottoinsiemi di $A \times B$ che soddisfa entrambe le condizioni della Proposizione 6.1.1, e quindi $|A \times B| = \sum_{a \in A} |S_a|$. Inoltre per l'osservazione (1) abbiamo $|A \times B| = \sum_{a \in A} |B| = |A| \cdot |B|$. ■

Se passiamo al prodotto cartesiano di più insiemi, caratterizziamo le n-ple (anche note come tuple, Sezione 2.7). Ci aspettiamo che valgano relazioni come $|A \times B \times C| = |A| \cdot |B| \cdot |C|$ e $|A \times B \times C \times D| = |A| \cdot |B| \cdot |C| \cdot |D|$ per esempio. Proviamo che uguaglianze di questo tipo valgono, usando l'induzione.

Proposizione 6.1.11. Per ogni $n \geq 1$, per tutti gli insiemi $A_1, A_2, \dots, A_{n-1}, A_n$, vale che

$$|A_1 \times A_2 \times \cdots \times A_{n-1} \times A_n| = |A_1| \cdot |A_2| \cdots |A_{n-1}| \cdot |A_n|$$

Dimostrazione. Procediamo per induzione su $n \geq 1$.

[Caso base] $n = 1$. Ovvio: l'asserto si riduce a $|A_1| = |A_1|$.

[Passo induttivo] $n > 1$. Usando l'associatività possiamo scrivere $A_1 \times A_2 \times \cdots \times A_{n-1} \times A_n = (A_1 \times A_2 \times \cdots \times A_{n-1}) \times A_n$. Applichiamo la Proposizione 6.1.10, ponendo $A = (A_1 \times A_2 \times \cdots \times A_{n-1})$ e $B = A_n$, ottenendo così

$$|A_1 \times A_2 \times \cdots \times A_{n-1} \times A_n| = |A_1 \times A_2 \times \cdots \times A_{n-1}| \cdot |A_n|$$

Per ipotesi induttiva, sappiamo che

$$|A_1 \times A_2 \times \cdots \times A_{n-1}| = |A_1| \cdot |A_2| \cdots |A_{n-1}|$$

e quindi, sostituendo, otteniamo

$$|A_1 \times A_2 \times \cdots \times A_{n-1} \times A_n| = |A_1 \times A_2 \times \cdots \times A_{n-1}| \cdot |A_n| = |A_1| \cdot |A_2| \cdots |A_{n-1}| \cdot |A_n|$$

■

Esempio 6.1.12. Le targhe automobilistiche italiane hanno il formato **AA000AA** con 22 lettere utilizzate (prese dalle 26 lettere dell'alfabeto inglese, tranne I, O, Q, U perché potrebbero generare confusione nella lettura). Ogni targa può essere quindi vista come una tupla di $r = 7$ insiemi, $A_1 \times A_2 \times A_3 \times A_4 \times A_5 \times A_6 \times A_7$, dove $|A_1| = |A_2| = |A_6| = |A_7| = 22$ e $|A_3| = |A_4| = |A_5| = 10$. Ci sono quindi $22^4 \cdot 10^3 = 234\,256\,000$ targhe distinte.

Le implicazioni della Proposizione 6.1.11 sono importanti. Per esempio una sequenza di lunghezza n su A può essere vista come una n -upla di $A \times A \times \cdots \times A = A^n$, dove $A = A_1 = A_2 = \cdots = A_n$.

Corollario 6.1.13. *Sia A^n l'insieme delle sequenze di lunghezza n su un insieme A . La sua cardinalità è $|A^n| = |A|^n$.*

Esempio 6.1.14. *Per esempio, nelle sequenze binarie, vale che $A = \{0, 1\} = 2$ e quindi ci sono $|2^n| = |2|^n = 2^n$ sequenze binarie di lunghezza n . Un altro esempio sono le stringhe in formato ASCII esteso, dove ogni carattere occupa un byte (8 bit). In questo caso $|A| = 256 = 2^8$ e ci sono 256^n sequenze di lunghezza n in ASCII esteso.*

Osservazione 6.1.15. *Esiste un modo alternativo per mostrare che ci sono 2^n stringhe binarie di lunghezza n che utilizza l'induzione. Questo ci tornerà utile in seguito per contare i sottoinsiemi di cardinalità prefissata.*

Indichiamo con $B(n)$ il numero di sequenze binarie di lunghezza n (ovviamente sappiamo già che $B(n) = 2^n$, ma qui vogliamo invece dedurlo).

Se $n = 0$ otteniamo la sequenza vuota e quindi $B(0) = 1$, che rappresenta la clausola base.

Per $n \geq 1$, osserviamo che vale uno schema ricorsivo: prendiamo tutte le $B(n-1)$ sequenze binarie di lunghezza $n-1$ e ci appendiamo un bit $b = 0$ in fondo; poi prendiamo di nuovo tutte le $B(n-1)$ sequenze binarie di lunghezza $n-1$ e questa volta ci appendiamo un bit $b = 1$ in fondo. I due gruppi di sequenze così ottenute sono disgiunti e la loro unione fornisce tutte le $B(n)$ sequenze binarie di lunghezza n : abbiamo pertanto ottenuto una partizione in due di queste ultime.

Per $n \geq 1$, la clausola induttiva è $B(n) = B(n-1) + B(n-1)$, dove il primo $B(n-1)$ corrisponde al caso $b = 0$ e il secondo al caso $b = 1$. Quindi $B(n) = 2B(n-1)$.

Mostriamo per induzione che $B(n) = 2^n$.

Caso base: $B(0) = 2^0 = 1$.

Passo induttivo: $B(n) = 2B(n-1) = 2 \cdot 2^{n-1} = 2^n$, applicando l'ipotesi induttiva che $B(n-1) = 2^{n-1}$.

6.2 Relazioni e cardinalità

Dal momento che ogni relazione $R: A \leftrightarrow B$ è essa stessa un insieme, visto che $R \subseteq A \times B$ (Definizione 2.1.1), possiamo parlare di cardinalità di una relazione. Per la Proposizione 1.7.5 sappiamo che $|R| \leq |A \times B|$. Possiamo dire qualcosa in più? Quando R gode delle proprietà viste nella Sezione 2.4 possiamo stabilire dei limiti più precisi.

Proposizione 6.2.1. *Per tutti gli insiemi A, B e per tutte le relazioni $R: A \leftrightarrow B$ vale che:*

1. *Se R è totale, allora $|A| \leq |R|$;*
2. *Se R è univalente, allora $|R| \leq |A|$;*
3. *Se R è surgettiva, allora $|B| \leq |R|$;*
4. *Se R è iniettiva, allora $|R| \leq |B|$;*

Dimostrazione. Dimostriamo i primi due punti: il terzo e quarto sono lasciati come esercizio.

1. Se R è totale, allora per tutti gli $a \in A$ c'è almeno una coppia $(a, b) \in R$ per un qualche $b \in B$. Pertanto $|A| \leq |R|$.
2. Se R è univalente, allora per tutti gli $a \in A$ c'è al più una coppia $(a, b) \in R$ per un qualche $b \in B$, quindi il numero di coppie in R è necessariamente minore o uguale del numero di elementi di A , cioè $|R| \leq |A|$.

■

Esercizio 6.2.2. *Dimostrare le Proposizioni 6.2.1.3 e 6.2.1.4.*

Qundi le quattro proprietà fondamentali delle relazioni giocano un ruolo chiave anche nelle cardinalità. Per evidenziare la dualità che emerge ancora una volta, abbiamo riassunto la Proposizione 6.2.1 nella Tabella 6.2.

La Proposizione 6.2.1 ha una conseguenza immediata per le funzioni.

totale $\Rightarrow A \leq R $	surgettiva $\Rightarrow B \leq R $
univalente $\Rightarrow R \leq A $	iniettiva $\Rightarrow R \leq B $

Tabella 6.2: Proprietà di relazioni e cardinalità

Corollario 6.2.3. Per tutti gli insiemi A, B vale che se $R: A \leftrightarrow B$ è una funzione, allora $|A| = |B|$. Infatti per Definizione 2.5.1 R è sia totale che univalente. Quindi entrambe le condizioni $|A| \leq |R|$ e $|R| \leq |A|$ sono verificate, da cui segue $|A| = |B|$ perché \leq è antisimmetrica.

Un classico risultato di combinatoria, che ha anche un nome suggestivo, è il cosiddetto *Pigeonhole principle* o *Principio delle buche e dei piccioni*. Esso afferma che se abbiamo n piccioni e li vogliamo collocare nelle m caselle di una piccionaia, se $n > m$ allora almeno una casella conterrà due piccioni. Possiamo facilmente formalizzare questo principio usando una terminologia matematica. Supponiamo di avere un insieme P (i piccioni) di cardinalità n e un insieme C (le caselle) di cardinalità m . “Collocare i piccioni nelle caselle” vuol dire semplicemente fornire una funzione $f: P \rightarrow C$. Quindi il principio asserisce che se $n > m$ allora nessuna funzione $f: P \rightarrow C$ può essere iniettiva. Possiamo derivare il principio (o meglio, una sua formulazione leggermente più forte) dalla Proposizione 6.2.1.

Corollario 6.2.4 (Pigeonhole principle). *Dati due insiemi P e C , se $|P| > |C|$ allora non esiste nessuna relazione $R: P \leftrightarrow C$ che sia totale e iniettiva. Infatti se esistesse una tale R , per la Proposizione 6.2.1 avremmo $|P| \leq |R|$ (R totale) e $|R| \leq |C|$ (R iniettiva), da cui $|P| \leq |C|$ per transitività, contraddicendo l’ipotesi $|P| > |C|$.*

Si osservi che il corollario è leggermente più generale del principio enunciato sopra, perché asserisce che nelle ipotesi date non ci possono essere *relazioni totali e iniettive* tra P e C , e queste contengono *strettamente* le funzioni iniettive.

Regola di biiezione

La Proposizione 6.2.1 ha una conseguenza immediata di fondamentale importanza: considerando relazioni totali, univalenti, surgettive ed iniettive, cioè le biiezioni (Definizione 2.6.1), si ottiene l’importante *Regola di biiezione*, che useremo spesso in seguito.

Corollario 6.2.5 (Regola di biiezione). *Per tutti gli insiemi A, B vale che se esiste una biiezione $R: A \leftrightarrow B$, allora $|A| = |B|$.*

Dimostrazione. Assumiamo che esista $R: A \leftrightarrow B$ totale, univalente, surgettiva ed iniettiva. Allora dalla Proposizione 6.2.1 si ha che

$$\begin{array}{ll} |A| \leq |R| & |B| \leq |R| \\ |R| \leq |A| & |R| \leq |B| \end{array}$$

Dall’antisimmetria di \leq , ne segue che

$$|A| = |R| \quad |R| = |B|$$

e quindi $|A| = |B|$. ■

Esercizio 6.2.6. Dimostrare che per ogni relazione $R: A \leftrightarrow B$ vale $|R^{op}| = |R|$.

La Regola di biiezione è molto utile nel seguente scenario per due insiemi A e B :

- dobbiamo contare A , cioè calcolare la sua cardinalità $|A|$;
- abbiamo già contato B , cioè calcolato la sua cardinalità $|B|$;
- se troviamo una biiezione R tra A e B (quindi $A \cong B$) allora possiamo dedurre il valore della cardinalità $|A|$, perché $|A| = |B|$.

Come primo esempio vediamo come possiamo calcolare la cardinalità dell'insieme $\text{Fun}(A, B)$ di tutte le funzioni con insieme di partenza A e insieme di arrivo B .

Proposizione 6.2.7 (Cardinalità di $\text{Fun}(A, B)$). *Per ogni coppia di insiemi A e B vale che $|\text{Fun}(A, B)| = |B|^{|A|}$.*

Dimostrazione. Mostriamo che esiste una biiezione tra $\text{Fun}(A, B)$ e l'insieme $B^{|A|}$ delle sequenze di elementi di B di lunghezza $|A|$. Infatti supponiamo che A abbia cardinalità n , e chiamiamo a_1, a_2, \dots, a_n gli elementi di A , tutti distinti e in un ordine qualunque. Consideriamo la funzione $\text{seq} : \text{Fun}(A, B) \rightarrow B^n$ definita come

$$\text{seq} : f \mapsto f(a_1), f(a_2), \dots, f(a_n)$$

e la funzione $\text{fun} : B^n \rightarrow \text{Fun}(A, B)$ definita come

$$\text{fun} : b_1, b_2, \dots, b_n \mapsto \{(a_i, b_i) \mid i \in \{1, \dots, n\}\}$$

È facile verificare (e lo lasciamo al lettore) che $\text{seq}; \text{fun} = \text{id}_{\text{Fun}(A, B)}$ e che $\text{fun}; \text{seq} = \text{id}_{B^n}$. Quindi fun è l'inversa di seq (Definizione 2.6.15), e seq è una biiezione (Teorema 2.6.16).

Quindi per la Regola di biiezione abbiamo che $|\text{Fun}(A, B)| = |B^{|A|}|$, e possiamo concludere osservando che per il Corollario 6.1.13 abbiamo che $|B^{|A|}| = |B|^{|A|}$. ■

Esempio 6.2.8. Supponiamo di dover fornire le previsioni metereologiche per la prossima settimana. Denotiamo con A i sette giorni della settimana e con B le possibili condizioni metereologiche: per esempio, $B = \{\text{sole, nebbia, pioggia, neve}\}$. Allora una previsione è una funzione che assegna ad ogni giorno in A una condizione metereologica in B . Esistono quindi $|B|^{|A|} = 4^7 = 16384$ possibili previsioni diverse.

Come secondo esempio di uso della Regola di biiezione dimostriamo la legge (6) della Tabella 6.1, ricordando che l'insieme delle parti di A è definito come $\mathcal{P}(A) = \{B \mid B \subseteq A\}$ (Definizione 1.5.2).

Proposizione 6.2.9 (Cardinalità dell'insieme delle parti). *Per ogni insieme A la cardinalità dell'insieme delle parti è $|\mathcal{P}(A)| = 2^{|A|}$.*

Dimostrazione. Abbiamo visto nella Proposizione 2.6.20 che vale

$$\mathcal{P}(A) \cong \text{Fun}(A, \text{Bool})$$

cioè che esiste una biiezione tra i sottoinsiemi di A e le funzioni da A in Bool . Quindi abbiamo che

$$\begin{aligned} |\mathcal{P}(A)| &= |\text{Fun}(A, \text{Bool})| && (\text{Regola di biiezione}) \\ &= |\text{Bool}^{|A|}| && (\text{Proposizione 6.2.7}) \\ &= 2^{|A|} && (|\text{Bool}| = 2) \end{aligned}$$

■

Esercizio 6.2.10. Dimostrare che $|\text{Rel}(A, B)| = 2^{|A| \cdot |B|}$.

A questo punto è naturale la seguente questione: la Regola di biiezione ci dice che se due insiemi sono in biiezione allora hanno la stessa cardinalità; è vero anche il contrario, cioè, se due insiemi hanno la stessa cardinalità, allora sono in biiezione? La risposta è positiva.

Per dimostrarlo ricordiamo che ogni numero naturale $n \in \mathbb{N}$, rappresenta anche l'insieme $n = \{0, 1, \dots, n - 1\}$ (Definizione 1.5.14) e osserviamo il seguente semplice fatto.

Proposizione 6.2.11. *Per tutti gli insiemi A e per tutti i numeri naturali $n \in \mathbb{N}$ vale che, se $|A| = n$, allora $A \cong n$.*

Dimostrazione. Sia A un insieme con cardinalità n . Chiamiamo a_1, a_2, \dots, a_n gli elementi di A , tutti distinti e in un ordine qualunque. Definiamo adesso la funzione $f : A \rightarrow n$ (per $n = \{0, 1, \dots, n - 1\}$) come

$$a_i \mapsto i - 1$$

e la funzione $g: n \rightarrow A$ come

$$i \mapsto a_{i+1}.$$

È immediato verificare che g è la funzione inversa di f e che quindi f è una biiezione (Teorema 2.6.16). Pertanto $A \cong n$. ■

Questo è sufficiente per fornire una risposta positiva alla domanda.

Teorema 6.2.12. *Per tutte le coppie di insiemi A e B vale che*

$$A \cong B \text{ se e solo se } |A| = |B|.$$

Dimostrazione. Trattandosi di un se e solo se possiamo spezzare la dimostrazione in due.

- Se $A \cong B$, allora $|A| = |B|$. Immediato dalla Regola di biiezione (Corollario 6.2.5).
- Se $|A| = |B|$, allora $A \cong B$. Assumiamo che $|A| = |B| = n$ per un qualche numero naturale n . Dalla Proposizione 6.2.11 si ha quindi che $A \cong n$ e $B \cong n$. Per simmetria e transitività di \cong (Proposizione 2.6.25) vale che $A \cong B$. ■

6.3 Permutazioni, disposizioni e combinazioni

Un problema classico del Calcolo Combinatorio consiste nel determinare quanti “raggruppamenti” diversi si possono avere degli elementi di un insieme di n oggetti disponendoli su un dato numero k di posti. Si ottengono risultati diversi a seconda che l’ordine in cui si raggruppano gli oggetti abbia importanza oppure no, e se i raggruppamenti possono o non possono avere ripetizioni degli elementi. In questa sezione introduciamo (e contiamo!) tre modi diversi di raggruppare elementi di un insieme: le permutazioni, le disposizioni e le combinazioni.

Permutazioni

Quando si va in pizzeria con gli amici, ci chiediamo sempre “chi si siede vicino a chi?”, o, in altre parole, “in che ordine ci sediamo intorno al tavolo?”. Dato un insieme A , una *permutazione* di A è un ordinamento dei suoi elementi. Le permutazioni ci consentono di studiare problemi dove ci chiediamo, ad esempio, quanti modi diversi ci sono di ordinare un determinato insieme, come appunto il gruppo di amici in pizzeria.

Definizione 6.3.1 (Permutazione). *Dato un insieme finito A con $|A| = n$, una PERMUTAZIONE di A è una sequenza ordinata a_0, \dots, a_{n-1} dove tutti e soli gli elementi di A appaiono esattamente una volta.*

Osservazione 6.3.2. *Si noti che ogni permutazione è una biiezione $\pi: A \rightarrow n$ (dove $n = |A|$): π mappa ogni elemento $x \in A$ nell’indice i per cui $x = a_i$. Tale funzione π è iniettiva e surgettiva perché tutti gli elementi appaiono esattamente una volta.*

Viceversa ogni biiezione $\pi: A \rightarrow n$ è una permutazione in cui la posizione di un arbitrario elemento $x \in A$ è esattamente $\pi(x)$.

Infatti permutazioni e biezioni $\pi: A \rightarrow |A|$ sono esattamente le stesse entità matematiche.

Una domanda naturale è: in quanti modi diversi ci possiamo sedere intorno al tavolo? Ovvero, quante permutazioni ha l’insieme A ?

Esempio 6.3.3. *Sia il nostro gruppo composto da Anna, Bruno, Ciro e Daniela, rappresentati dall’insieme $A = \{a, b, c, d\}$: le permutazioni di A sono 24. È tedioso ma facile verificare questo fatto andando ad enumerarle una ad una:*

$abcd$	$adbc$	$bcad$	$cabd$	$cdab$	$dbac$
$abdc$	$adcb$	$bcda$	$cada$	$cdba$	$dbca$
$acbd$	$bacd$	$bdac$	$cbad$	$dabc$	$dcab$
$acdb$	$badc$	$bdca$	$cbda$	$dacb$	$dcba$

Naturalmente non vorremmo procedere in questo modo per contare, per esempio, le permutazioni dell'insieme delle lettere dell'alfabeto inglese, o dei primi 100 numeri naturali. Cerchiamo quindi di individuare una formula che ci fornisca il numero di permutazioni $P(n)$ di un insieme di n elementi: infatti è evidente che il numero di permutazioni di un insieme dipende solo dalla sua cardinalità, e non dagli elementi che contiene.

Esercizio 6.3.4. Si dimostri quest'ultima affermazione sfruttando la Regola di biezione.

Pensando a n amici in pizzeria, supponiamo di fissare una sedia a capotavola come punto di partenza e assumiamo che il numero di sedie sia uguale al numero di persone. Chiaramente ci sono n possibili scelte per chi si siederà in quel posto. Una volta seduto il primo amico, chiunque esso sia, rimarranno $n - 1$ scelte per la seconda sedia, poi $n - 2$ per la terza, e così via, fino ad arrivare all'ultima sedia libera dove avremo 1 sola scelta, ovvero si dovrà sedere l'ultimo amico rimasto in piedi. Quindi possiamo concludere che $P(n) = n \cdot (n - 1) \cdot (n - 2) \cdots 1$, cioè $P(n) = n!$. E infatti, per l'Esempio 6.3.3 abbiamo $24 = 4! = P(4)$.

Per quanto corretto e convincente, questa argomentazione rimane a tratti informale quando diciamo *e così via*, che nella formula corrisponde ai puntini sospensivi. Abbiamo mostrato nel Capitolo 3 che in questi casi possiamo usare l'induzione matematica per ottenere una completa rigorosità. Cerchiamo quindi una descrizione induttiva di $P(n)$, per induzione su n .

Pensiamo di nuovo alla situazione di sopra, con n amici in pizzeria e un tavolo con n sedie di cui una è fissata. Chiaramente se $n = 1$, allora l'unico amico si dovrà sedere sull'unica sedia, quindi una possibile clausola base è $P(1) = 1$. Per la clausola induttiva, se abbiamo $n + 1$ amici, ci sono $n + 1$ possibili scelte per chi si siederà a capotavola. A questo punto saranno rimasti n amici che si devono sedere in modo arbitrario nelle n sedie rimanenti, e per ipotesi induttiva assumiamo che lo possano fare in esattamente $P(n)$ modi. La clausola induttiva è quindi:

$$P(n + 1) = (n + 1) \cdot P(n) \quad (6.3)$$

Si osservi inoltre che se applichiamo questa clausola induttiva con $n = 0$, otteniamo

$$P(0 + 1) = (0 + 1) \cdot P(0) = P(0) \quad (6.4)$$

cioè $P(1) = P(0)$. Questo ci suggerisce che possiamo prendere come nuovo caso base $P(0) = 1$ (intuitivamente: “se abbiamo un gruppo di 0 amici, allora c'è un unico modo di farlo sedere su 0 sedie”). Questa definizione induttiva ci permette di dimostrare in modo formale che $P(n) = n!$.

Proposizione 6.3.5. Sia A un insieme di cardinalità $n \geq 0$. Allora ci sono esattamente

$$P(n) = n!$$

permutazioni di A .

Dimostrazione. Per induzione su n , ricordando la definizione di fattoriale in Sezione 3.1.

1. [Caso base]: $P(0) = 1 = 0!$.

2. [Passo induttivo]:

$$\begin{aligned} P(n + 1) &= (n + 1) \cdot P(n) && (6.3) \\ &= (n + 1) \cdot n! && (\text{Ipotesi induttiva } (P(n) = n!)) \\ &= (n + 1)! && (\text{Clausola induttiva del fattoriale}) \end{aligned}$$

■

Esercizio 6.3.6. Utilizzando l'Osservazione 6.3.2, dimostrare che

$$|Bii(A, B)| = \begin{cases} 0 & \text{se } |A| \neq |B| \\ |A|! & \text{se } |A| = |B| \end{cases}$$

Anagrammi e permutazioni con ripetizioni

Un classico esempio di permutazioni che incontriamo sin dai giochi dell'infanzia sono gli *anagrammi*: data una parola, un suo anagramma è una parola ottenuta cambiando l'ordine delle lettere. Per esempio CERTOSA è un anagramma di COSTARE. Ignorando il fatto che certi anagrammi potrebbero non essere parole di senso compiuto, è legittimo chiedersi quanti siano i possibili anagrammi di una parola. Avendo appena introdotto le permutazioni, la risposta ovvia sarebbe $P(n) = n!$ se la parola ha n lettere, quindi per esempio COSTARE ha $P(7) = 7! = 5040$ anagrammi, come il lettore può verificare con qualche sforzo.

Ma se considero la parola ANNA, quanti anagrammi ha? Il numero di permutazioni di 4 elementi è $P(4) = 4! = 24$, ma permutando le lettere di ANNA otteniamo solo 6 parole *diverse*:

AANN	ANAN	ANNA	NAAN	NANA	NNAA
------	------	------	------	------	------

Questa apparente contraddizione è presto spiegata: le permutazioni e la loro cardinalità sono state introdotte per gli elementi *di un insieme*, che sono tutti distinti per definizione. Invece nella parola ANNA ci sono solo due lettere distinte: infatti una parola non è un *insieme* di lettere, ma una *sequenza* o *tupla* di lettere, in cui la stessa lettera può comparire più volte. In questo caso il numero di permutazioni distinte si può ottenere con una formula che considera il numero di occorrenze di ogni singola lettera.

Proposizione 6.3.7. *Sia $S = s_1, s_2, \dots, s_k$ una sequenza di elementi di un insieme $A = \{a_1, a_2, \dots, a_n\}$ di cardinalità n , in cui ogni elemento di A può comparire 0 o più volte. Inoltre per ogni $i \in \{1, 2, \dots, n\}$ sia c_i il numero di volte che l'elemento a_i compare nella sequenza S . Allora il numero di PERMUTAZIONI CON RIPETIZIONI della sequenza è dato dalla seguente formula:*

$$\frac{k!}{c_1! \cdot c_2! \cdot \dots \cdot c_n!}$$

Per esempio, nel caso della parola ANNA discusso sopra possiamo considerare come insieme A quello delle lettere dell'alfabeto italiano, $k = 4$, $c_1 = 2$ (ci sono due A), $c_{12} = 2$ (ci sono due N) e $c_i = 0$ per tutte le altre lettere. Quindi il numero di permutazioni con ripetizioni risulta essere il seguente, ricordando che $0! = 1$ e $2! = 2$:

$$\frac{k!}{c_1! \cdot c_2! \cdot \dots \cdot c_{21}!} = \frac{4!}{c_1! \cdot c_{12}!} = \frac{24}{2 \cdot 2} = 6$$

Disposizioni

Supponiamo ora che il ristorante non possa mettere tutti gli n amici allo stesso tavolo, ma solamente k di loro, facendo sedere i rimanenti in un'altra sala. In quanti modi diversi possiamo riempire il tavolo con k posti? Chiamiamo questo concetto le *disposizioni* di n elementi in k posti, note anche come k -permutazioni di n .

Definizione 6.3.8 (Disposizione). *Dato un insieme finito A con $|A| = n$ e un intero $k \leq n$, una DISPOSIZIONE degli elementi di A in k posti è una sequenza ordinata a_1, \dots, a_k di elementi di A .*

Esempio 6.3.9. *Sia il nostro gruppo composto da Anna, Bruno, Ciro, Daniela, ed Enrico, rappresentati dall'insieme $A = \{a, b, c, d, e\}$, ed ammettiamo di avere un tavolo da 2 posti: ci sono 20 disposizioni di $n = 5$ elementi di A in $k = 2$ posti, come possiamo verificare:*

ab	ae	bd	cb	da	de	ec
ac	ba	be	cd	db	ea	ed
ad	bc	ca	ce	dc	eb	

Anche in questo caso vogliamo trovare una formula esplicita per il numero $D(n, k)$ di disposizioni degli elementi di A , un insieme di cardinalità n , in $k \leq n$ posti. Intuitivamente, nel primo posto possiamo mettere uno qualunque degli elementi di A , quindi abbiamo n possibili scelte; nel secondo posto abbiamo $n - 1$ scelte; nel terzo $n - 2$ scelte e così via, e nel k -esimo posto abbiamo ancora

$n - (k - 1) = n - k + 1$ scelte. Quindi $D(n, k) = n \cdot (n - 1) \cdot (n - 2) \cdots \cdot (n - k + 1)$, che si può scrivere in modo equivalente come

$$D(n, k) = \frac{n!}{(n - k)!} \quad (6.5)$$

E infatti, per l'Esempio 6.3.9 abbiamo $20 = 5 \cdot 4 = \frac{5!}{(5-2)!} = D(5, 2)$. Come per le permutazioni anche per il numero $D(n, k)$ di disposizioni è interessante vederne una caratterizzazione induttiva. Supponiamo che $n \geq 0$ sia fissato, e analizziamo $D(n, k)$ per induzione su k , con $0 \leq k \leq n$. Come clausola base abbiamo (\dagger) $D(n, 0) = 1$: c'è un solo modo di prendere una sequenza vuota di elementi da un insieme A che ne contiene n . Come clausola induttiva, se assumiamo come ipotesi induttiva che ci sono $D(n, k)$ disposizioni di A su k posti, allora se aggiungiamo un posto abbiamo $n - k$ possibili scelte per riempirlo, perché questo è il numero di elementi non ancora estratti da A . Quindi la clausola ricorsiva è

$$D(n, k + 1) = D(n, k) \cdot (n - k) \quad (6.6)$$

Possiamo ora dimostrare per induzione la (6.5).

Proposizione 6.3.10. *Sia A un insieme di cardinalità $n \geq 0$ e k tale che $0 \leq k \leq n$. Allora ci sono esattamente*

$$D(n, k) = \frac{n!}{(n - k)!}$$

disposizioni degli elementi di A in k posti.

Dimostrazione. Procediamo per induzione su k .

1. [Caso base]: Per $k = 0$, abbiamo $D(n, 0) \stackrel{(\dagger)}{=} 1 = \frac{(n)!}{(n-0)!}$.
2. [Passo induttivo]: Assumiamo ora che $0 < k + 1 \leq n$ e che per ipotesi induttiva valga $D(n, k) = \frac{n!}{(n-k)!}$. Dobbiamo far vedere che

$$D(n, k + 1) = \frac{n!}{(n - (k + 1))!}$$

Infatti abbiamo

$$\begin{aligned} D(n, k + 1) &= D(n, k) \cdot (n - k) && (6.6) \\ &= \frac{n!}{(n-k)!} \cdot (n - k) && (\text{Ipotesi induttiva}) \\ &= \frac{n!}{(n-(k+1))!} \end{aligned}$$

■

Combinazioni

Supponiamo, come all'inizio della sottosezione precedente, che il ristorante non possa mettere tutti gli n amici allo stesso tavolo, ma solamente k di loro, facendo sedere i rimanenti in un'altra sala. Ci chiediamo ora: in quanti modi diversi possiamo scegliere k degli n amici da far accomodare al tavolo? Chiamiamo questo concetto una *combinazione* di k elementi di un insieme di cardinalità n , ma a dispetto del nome nuovo lo avevamo già introdotto nel Capitolo 1.

Definizione 6.3.11 (Combinazione). *Sia A un insieme di cardinalità n e sia $k \leq n$ un numero naturale. Una COMBINAZIONE di k elementi di A è un k -insieme di A , cioè un sottoinsieme di A di cardinalità k (Definizione 1.7.6). L'insieme di tutte le combinazioni di k elementi di A è quindi denotato con $\mathcal{P}_k(A)$.*

Il numero di combinazioni di k elementi di un insieme di cardinalità n è chiamato *coefficiente binomiale* e indicato come $\binom{n}{k}$. Il suo valore può essere ottenuto con la cosiddetta *formula dei tre fattoriali*:

$$\binom{n}{k} = \frac{n!}{k! (n - k)!} \quad (6.7)$$

Esempio 6.3.12. Sia il nostro gruppo composto da Anna, Bruno, Ciro, Daniela, ed Enrico, rappresentati dall'insieme $A = \{a, b, c, d, e\}$, ed ammettiamo di avere un tavolo da 3 posti: le possibili combinazioni di 3 elementi di A sono

$$\binom{5}{3} = \frac{5!}{3!(5-3)!} = \frac{5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}{(3 \cdot 2 \cdot 1)(2 \cdot 1)} = \frac{5 \cdot 4}{2} = 10$$

Anche qui possiamo verificare il calcolo a mano:

abc	abe	ace	bcd
abd	acd	ade	bce
			bde
			cde

Non è difficile giustificare la formula (6.7) a partire da quelle di disposizioni e permutazioni. Abbiamo visto che possiamo disporre k amici di un insieme di n attorno a un tavolo in $D(n, k) = \frac{n!}{(n-k)!}$ modi diversi. Ma fissato un gruppo di k amici, essi si possono sedere in $P(k) = k!$ modi diversi attorno al tavolo. Poiché nelle combinazioni non interessa l'ordine degli elementi, possiamo ottenerne il numero dividendo il numero di disposizioni per il numero di permutazioni di k elementi:

$$\binom{n}{k} = \frac{D(n, k)}{P(k)} = \frac{\frac{n!}{(n-k)!}}{k!} = \frac{n!}{k! (n-k)!}$$

Come per il caso di permutazioni e disposizioni, anche per le combinazioni è interessante trovare una definizione induttiva. Tuttavia invece di considerare i k -insiemi, consideriamo opportune sequenze di zeri e uni, grazie al seguente risultato.

Proposizione 6.3.13. Esiste una biiezione tra l'insieme $\mathcal{P}_k(A)$ dei k -insiemi di A e le sequenze binarie di lunghezza $n = |A|$ (quindi gli elementi di $\{0, 1\}^n$) aventi esattamente k bit a 1.

Dimostrazione. Sia A un insieme con cardinalità n . Per la Proposizione 2.6.20 sappiamo che $\mathcal{P}(A) \cong \text{Fun}(A, \text{Bool})$, e in particolare che a ogni sottoinsieme $B \subseteq A$ corrisponde una funzione caratteristica $\chi_B : A \rightarrow \text{Bool}$ definita come

$$\chi_B(a) = \begin{cases} \text{t} & \text{se } a \in B \\ \text{f} & \text{se } a \notin B \end{cases}$$

Sia $\chi'_B : A \rightarrow 2$ la funzione $\chi_B ; i$, dove $i : \text{Bool} \rightarrow 2$ è la biiezione dell'Esempio 2.6.3. Chiamiamo a_1, a_2, \dots, a_n gli elementi di A , tutti distinti e in un ordine qualunque ma fissato. Consideriamo ora la funzione $\chi^k : \mathcal{P}_k(A) \rightarrow \{0, 1\}^n$ definita come:

$$\chi^k(B) = \chi'_B(a_1), \chi'_B(a_2), \dots, \chi'_B(a_n)$$

Quindi $\chi^k(B)$ è una sequenza binaria che ha un 1 in tutte e sole le posizioni corrispondenti agli elementi di A che appartengono a B , e quindi ha esattamente k bit a 1. È facile mostrare che $\chi^k(B)$ è iniettiva e che per ogni sequenza binaria s di lunghezza n con esattamente k bit a 1 esiste un sottoinsieme $B_s \in \mathcal{P}_k(A)$ tale che $\chi^k(B_s) = s$, quindi χ^k è la biiezione cercata. ■

Quindi per la Regola di biezione abbiamo che il numero di sequenze binarie di lunghezza n con esattamente k bit a 1 è $\binom{n}{k}$. Vediamo come dimostrare la (6.7) sfruttando l'induzione. Per prima cosa osserviamo che

$$\binom{n}{0} = \binom{n}{n} = 1 \tag{6.8}$$

Infatti esiste una sola sequenza binaria di lunghezza n con 0 bit a 1 (quella costituita solo da zeri), e ovviamente anche una sola con n bit a 1. Consideriamo ora tutte le sequenze binarie di lunghezza n con k bit a 1, con $0 < k < n$, e chiamiamo b il loro ultimo bit, e con α la sequenza dei loro primi $n - 1$ bit. Possiamo partizionare tali sequenze (che sono $\binom{n}{k}$) in due gruppi:

- le sequenze che hanno $b = 0$: in tali sequenze α deve necessariamente avere k bit a 1, e quindi ce ne sono esattamente $\binom{n-1}{k}$, per ipotesi induttiva;

- le sequenze che hanno $b = 1$: qui α deve avere $k - 1$ bit a 1, poiché sappiamo che il k -esimo 1 è appunto b ; quindi ci sono esattamente $\binom{n-1}{k-1}$ sequenze di questo tipo, per ipotesi induttiva.

Mettendo insieme queste osservazioni, possiamo scrivere la nostra clausola induttiva come

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1} \quad (6.9)$$

Proposizione 6.3.14. *Dati $n, k \in \mathbb{N}$ con $k \leq n$, abbiamo che*

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Dimostrazione. [Caso base] Per la (6.8) abbiamo

$$\binom{n}{0} = 1 = \frac{n!}{0! n!} \quad \text{e} \quad \binom{n}{n} = 1 = \frac{n!}{n! 0!}$$

[Passo induttivo] Per la (6.9) abbiamo

$$\begin{aligned} \binom{n}{k} &= \binom{n-1}{k} + \binom{n-1}{k-1} = \frac{(n-1)!}{k!(n-1-k)!} + \frac{(n-1)!}{(k-1)!(n-k)!} = \\ &= \frac{(n-1)!}{(k-1)!(n-1-k)!} \cdot \left(\frac{1}{k} + \frac{1}{n-k} \right) = \frac{(n-1)!}{(k-1)!(n-1-k)!} \cdot \frac{n}{k(n-k)} = \\ &= \frac{n!}{k!(n-k)!} \end{aligned}$$

■

Corollario 6.3.15. $|\mathcal{P}_k(A)| = \binom{|A|}{k}$.

È interessante collegare quanto visto finora per il coefficiente binomiale con la definizione di *insieme delle parti* di A , usando le combinazioni: notiamo infatti che il numero di tutti i sottoinsiemi corrisponde a

- L'insieme vuoto
- Tutti i sottoinsiemi di cardinalità 1
- Tutti i sottoinsiemi di cardinalità 2
- ...
- Tutti i sottoinsiemi di cardinalità $n - 1$
- L'insieme stesso A

Sappiamo ora calcolare questo valore, in quanto il numero di k -insiemi è dato proprio dal numero di combinazioni di k elementi di A . Possiamo quindi dire che il numero di sottoinsiemi di A pari a:

$$\sum_{k=0}^n \binom{n}{k}$$

Questo ci consente di calcolare il numero, ma non è pienamente soddisfacente, in quanto la formula non è di immediata valutazione. Avendo visto la biezione dell'insieme delle parti di A con il numero di sequenze possibili con $|A|$ bit, possiamo concludere il seguente risultato.

Proposizione 6.3.16.

$$\sum_{k=0}^n \binom{n}{k} = 2^n$$

Tornando alla regola di biiezione, essa agisce come una lente d'ingrandimento sulla capacità di contare: se uno determina la dimensione di un insieme, allora può determinare immediatamente le dimensioni di molti altri insiemi attraverso biiezioni.

Esempio 6.3.17. Consideriamo i due seguenti insiemi:

- (A) tutti i modi di selezionare una dozzina di ciambelle tra 5 varietà disponibili;
- (B) tutte le sequenze di 16 bit con esattamente 4 uni.

Un esempio di elemento di A è

$$\begin{array}{ccccc} \overbrace{00} & \overbrace{\quad} & \overbrace{000000} & \overbrace{00} & \overbrace{00} \\ \text{cioccolato} & \text{limone} & \text{zucchero} & \text{fragola} & \text{semplice} \end{array}$$

Di sopra abbiamo disegnato ogni ciambella con uno 0 ed abbiamo lasciato uno spazio vuoto tra ogni diversa varietà. Quindi la selezione di sopra consiste di due ciambelle al cioccolato, nessuna ciambella al limone, sei ciambelle con lo zucchero, due alla fragola e due semplici. Adesso inseriamo un 1 in ognuno dei quattro spazi

$$\begin{array}{ccccc} \overbrace{00} & 1 & \overbrace{1} & \overbrace{000000} & \overbrace{1} \\ \text{cioccolato} & \text{limone} & \text{zucchero} & \text{fragola} & \text{semplice} \end{array}$$

ed otteniamo un elemento di B

$$0011000000100100$$

Infatti questa è una sequenza di 16 bit in cui 1 appare esattamente 4 volte. Questo esempio suggerisce una biiezione dall'insieme A all'insieme B: una dozzina di ciambelle che consiste di sequenza di

c cioccolato, l limone, z zucchero, f fragola, s semplici

viene mappata nella sequenza

$$\underbrace{0\dots0}_{c} \underbrace{10\dots0}_{l} \underbrace{10\dots0}_{z} \underbrace{10\dots0}_{f} \underbrace{10\dots0}_{s}$$

La sequenza risultante ha sempre 16 bits e esattamente 4 uni, ed è quindi un elemento di B. Inoltre questa funzione è una biiezione: ogni sequenza di bits in B viene esattamente da un solo ordine di dodici ciambelle. Quindi $A \cong B$ e, grazie alla regola di biezione, $|A| = |B|$.

Più in generale vale che il numero di modi di selezionare z ciambelle quando sono disponibili g gusti è lo stesso del numero di sequenze con esattamente z zeri e $g - 1$ uni. Ponendo $n = z + g - 1$ e $k = g - 1$, abbiamo così che tale numero è dato da $\binom{z+g-1}{g-1}$.

Osserviamo, tra l'altro, che tale formula si applica anche al problema di ripartire z elementi in g insiemi: ad esempio, tutti i modi possibili di riporre z magliette in g cassetti.

6.4 Contare nei grafi

In questa sezione vediamo come applicare i concetti presentati nelle sezioni precedenti per contare nodi, archi e/o cammini con certe proprietà in vari tipi di grafi, o come contare il numero di grafi che soddisfano certe proprietà.

Contare negli alberi

Nella Proposizione 5.6.4 abbiamo già presentato alcune proprietà combinatorie degli alberi: ogni albero ha $n - 1$ archi, dove n è il numero di nodi, e per ogni coppia di nodi esiste un unico path che li collega.

Consideriamo ora un albero binario T (che è quindi radicato). Della Definizione 5.6.8 sappiamo che è *pieno* se ogni nodo interno ha entrambi i figli non vuoti; inoltre dalla Definizione 5.8.7 sappiamo che è *completo* se è pieno e le foglie sono tutte alla stessa distanza dalla radice. Prendiamo la sua altezza h e osserviamo la sua relazione con il numero f di foglie in T e il numero i di nodi interni (inclusa la radice): osserviamo che il numero totale di nodi è quindi $n = f + i$.

- Se l'altezza è $h = 0$, abbiamo $f = 1$ e $i = 0$.
- Se l'altezza è $h = 1$, abbiamo $f = 2$ e $i = 1$.
- Se l'altezza è $h = 2$, abbiamo $f = 4$ e $i = 3$.
- Se l'altezza è $h = 3$, abbiamo $f = 8$ e $i = 7$.
- ...

Proposizione 6.4.1. *Un albero completo di altezza h ha 2^h foglie e $2^h - 1$ nodi interni, per un totale di $2^{h+1} - 1$ nodi.*

Dimostrazione. Per induzione sull'altezza h . Il caso base, per $h = 0$, è immediato, come visto sopra. Per il passo induttivo, assumiamo come ipotesi induttiva che l'albero con altezza $h - 1$ ha $f' = 2^{h-1}$ foglie e $i' = 2^{h-1} - 1$ nodi interni. Passando a livello h , osserviamo che il numero di foglie raddoppia $f = 2 \cdot f' = 2^h$, e il numero di nodi interni è il numero totale di nodi dell'albero completo di altezza $h - 1$, cioè $i = f' + i' = 2^h - 1$. ■

La Proposizione 6.4.1 indirettamente dice anche che vale la seguente uguaglianza, che poi non è altro che il numero naturale rappresentato in binario da $h + 1$ bit tutti a 1:

$$\sum_{i=0}^h 2^i = 2^{h+1} - 1$$

La Proposizione 6.4.1 ci dice anche che $h = \log_2 f$ nell'albero completo. Se prendiamo un albero T qualunque, non necessariamente completo, abbiamo visto che la sua altezza è $h \leq n - 1$, e l'altezza massima $h = n - 1$ si ha quando ogni nodo interno ha un solo figlio e c'è solo una foglia. Qual è l'altezza minima rispetto al suo numero f di foglie? Vogliamo compattare al massimo i nodi di T verso la radice: anche se f non è più necessariamente una potenza del 2, esse sono disposte a distanza h e $h - 1$ (se invece sono 2^h sono tutte a distanza h perché l'albero è completo, come abbiamo visto). Ne consegue che in generale

$$h \geq \log_2 f$$

Negli alberi k -ari, la base 2 viene sostituita dalla base k e quindi abbiamo $\log_k f$.

Come esempio di applicazione, negli algoritmi di ordinamento, verrà mostrato un albero di decisione che ha $f = n!$ foglie. L'altezza minima di tale albero è $\log_2 n!$ che cresce asintoticamente come $n \log n$ usando l'approssimazione di Stirling del fattoriale $n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$.

Da quanto visto finora, non c'è necessariamente un legame tra il numero di nodi interni e le foglie. Prendiamo un albero radicato T con f foglie. Un nodo interno u di T (inclusa la radice) si dice *unario* se u ha un solo figlio (si ricordi che le foglie hanno zero figli). Chiaramente un nodo interno *non unario* ha almeno due figli. Se prendiamo una catena di nodi unari con una sola foglia, emerge che non c'è una relazione diretta tra il numero di nodi interni e il numero di foglie. Invece quelli non unari sono strettamente correlati a f , come mostrato dalla seguente proposizione.

Proposizione 6.4.2. *Il numero di nodi non unari in T è al massimo $f - 1$.*

Dimostrazione. Osserviamo che esiste sempre un nodo interno x in T i cui figli sono tutte foglie: per assurdo, se così non fosse, potremmo sempre scegliere un figlio che non è una foglia, rendendo di fatto T di cardinalità infinita. Abbiamo due casi su x usando l'induzione su f (lasciamo al lettore il caso base):

- Se x è unario, allora possiamo cancellare il suo unico figlio, che è una foglia, e x diventa una foglia; in questo modo, il numero f di foglie rimane inalterato ma c'è un nodo unario in meno (che mostra quindi che il numero di nodi unari può essere arbitrario rispetto a f).

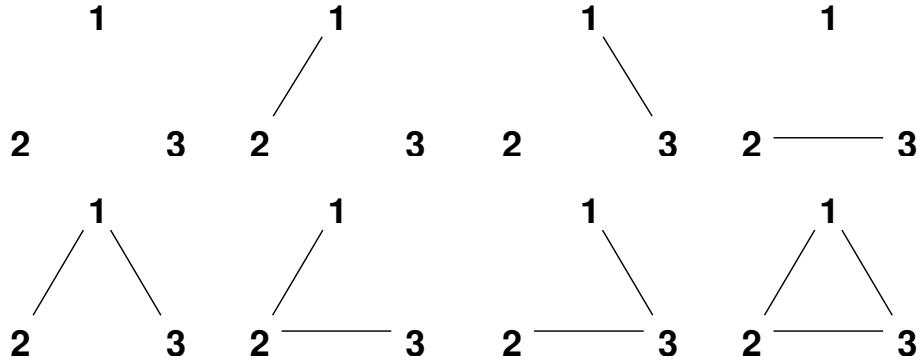


Figura 6.1: Grafi possibili su 3 nodi.

- Se x è non unario, allora ha almeno $c \geq 2$ figlie, che sono tutte foglie. Se le rimuoviamo, rendiamo x una foglia: nell'albero risultante T' , c'è un nodo non unario in meno e $f' = f - c + 1$ foglie (c sono rimosse e x diventa foglia). Poiché $c \geq 2$, abbiamo $f' \leq f - 1$ foglie e quindi c'è sicuramente una foglia in meno. Possiamo applicare l'ipotesi induttiva su f' e concludere che T' ha al massimo $f' - 1$ nodi non unari: di conseguenza T ne ha al massimo $f' - 1 + 1 = f'$ (cioè ha x in più rispetto a T') e il risultato segue dal fatto che $f' \leq f - 1$. ■

Contare i grafi

Quanti grafi *non orientati* esistono? Chiaramente un numero illimitato se non fissiamo la dimensione. Quindi proviamo a formulare una domanda a cui poter rispondere: quanti grafi (non orientati) esistono con n nodi, assumendo che l'insieme dei nodi sia proprio $n = \{1, 2, \dots, n\}$? Chiamiamo G_n l'insieme di grafi con n nodi, e quindi $|G_n|$ la sua cardinalità. Se ad esempio prendiamo $n = 3$, possiamo vedere concretamente che esistono 8 grafi diversi, mostrati in Figura 6.1.

L'intuizione è che ogni grafo consiste nel prendere un sottoinsieme degli archi possibili, ovvero un insieme $E \subseteq V \times V$. Nei grafi *non orientati*, abbiamo alcune restrizioni: non possiamo scegliere il cappio, e per ogni coppia di nodi x, y gli archi xy e yx sono in realtà lo stesso arco. Il numero di archi possibili è quindi $m_{max} = \frac{n(n-1)}{2}$.

Per ogni $i \in \{0, \dots, m_{max}\}$, sappiamo che il numero di grafi su n nodi con i archi corrisponde al numero di sottoinsiemi di archi di cardinalità i , ovvero $\binom{m_{max}}{i}$. Il numero totale di grafi che posso costruire è quindi:

$$|G_n| = \sum_{i \in \{0, \dots, m_{max}\}} \binom{m_{max}}{i}$$

Tuttavia esiste un metodo alternativo che ci consente di arrivare ad una soluzione più semplice:

Ricordiamo che l'insieme di tutti i sottoinsiemi di un insieme di cardinalità k è noto come l'*insieme delle parti* $\mathcal{P}(k)$. In questo caso ci stiamo chiedendo esattamente la cardinalità dell'insieme delle parti di m_{max} .

La cardinalità di $\mathcal{P}(k)$, ovvero il numero di sottoinsiemi, corrisponde a tutti i modi di scegliere quali elementi includere: abbiamo 2 scelte per il primo elemento (incluso/non incluso), 2 scelte per il secondo, 2 scelte per il terzo... Dato che queste scelte sono indipendenti, abbiamo 2^k possibili combinazioni.

Otteniamo quindi

$$\mathcal{P}(m_{max}) = \sum_{i \in \{0, \dots, m_{max}\}} \binom{m_{max}}{i} = 2^{m_{max}} = 2^{\frac{n(n-1)}{2}}$$

Come conseguenza, possiamo osservare anche le due seguenti proprietà del coefficiente binomiale:

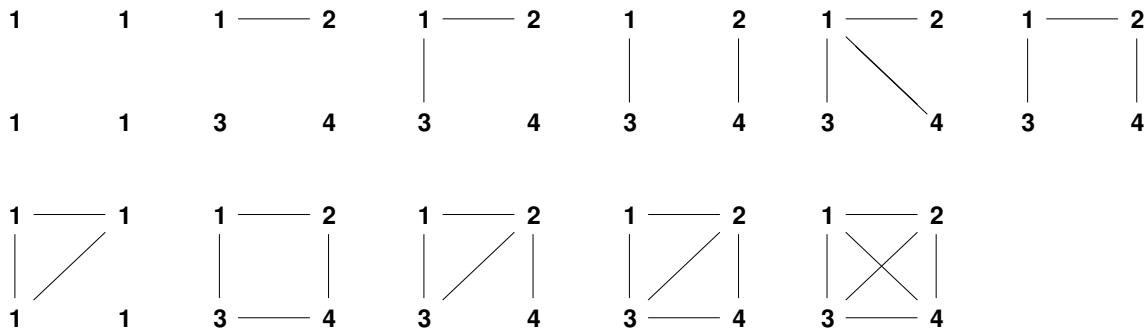


Figura 6.2: 11 tipi di grafi possibili su 4 nodi. Esistono $2^{\frac{n(n-1)}{2}} = 64$ grafi, ma ognuno è isomorfo a uno di questi 11.

$$\sum_{i \in \{0, \dots, k\}} \binom{k}{i} = 2^k, \text{ e quindi } \forall_{k > 0, i \geq 0} \binom{k}{i} < 2^k$$

Tuttavia, è bene osservare che alcuni dei grafi che generiamo in questo modo sono *isomorfi*, ovvero hanno “la stessa forma”: se non ci interessa la mappatura dei nodi, osserviamo in Figura 6.1 che esistono solo 4 “tipi” di grafi diversi con 3 nodi (il grafo vuoto, il grafo con 1 arco, il grafo con 2 archi adiacenti, e il triangolo). Se saliamo a $n = 4$, sappiamo immediatamente dire che esistono $2^{\frac{n(n-1)}{2}} = 64$ grafi, tuttavia disegnandoli possiamo scoprire che esistono solo 11 tipi diversi.

Quanti “tipi” di grafo esistono su n nodi è una domanda più complessa, alla quale non sappiamo rispondere con una formula chiusa. Tuttavia è possibile scrivere algoritmi per calcolare questi numeri, se siamo abbastanza pazienti da aspettare il risultato (e possiamo trovare i primi valori qui nella *Online Encyclopedia of Integer Sequences* <https://oeis.org/A000088>)

E quanti grafi *orientati* esistono? Il problema in questo caso diventa leggermente più semplice: qui posso tracciare un arco distinto da ogni nodo verso ogni nodo (incluso se stesso). Abbiamo quindi esattamente $|V \times V| = n^2$ possibili archi. Ne consegue che avremo esattamente 2^{n^2} possibili grafi orientati su n nodi.

Teorema 6.4.3. Esistono $2^{\frac{n(n-1)}{2}}$ grafi non orientati su n nodi, e 2^{n^2} grafi orientati su n nodi.

Osserviamo anche come lo stesso ragionamento si può ricavare dalle matrici di adiacenza:

Osservazione 6.4.4. Nella matrice di adiacenza di un grafo orientato ognuna delle n^2 celle può valere 0 o 1, abbiamo quindi 2^{n^2} possibili celle con cui generare 2^{n^2} possibili matrici di adiacenza.

In un grafo non orientato, invece, le n celle della diagonale valgono 0 in quanto non sono ammessi cappi, e delle rimanenti celle possiamo usarne solo metà, in quanto la matrice deve essere simmetrica rispetto alla diagonale principale. Abbiamo quindi $\frac{n^2 - n}{2} = \frac{n(n-1)}{2}$ celle utili, con cui possiamo generare $2^{\frac{n(n-1)}{2}}$ possibili matrici di adiacenza.

Complemento di un grafo

Dato un grafo $G = (V, E)$, possiamo definire il suo *complemento* come il grafo $H = (V, E')$ sugli stessi nodi, che ha tutti e soli gli archi che *mancano* in G .

Formalmente, $H = (V, E')$ è il complemento di $G = (V, E)$ se $E' = \{xy \in V \times V \mid xy \notin E\}$

Guardando la Figura 6.1, ad esempio, possiamo osservare come il primo grafo (con 0 archi) sia il complemento dell’ultimo (con 3 archi), mentre ognuno dei grafi con 1 arco è il complemento di uno dei grafi con 2 archi.

Osserviamo anche che la relazione è simmetrica: se H è il complemento di G , allora G è il complemento di H .

Questa osservazione ci porta a una regola più generale:

Osservazione 6.4.5. Ogni grafo $G \in G_n$ ha esattamente un complemento $H \in G_n$, e la relazione complemento $C \subseteq G_n \times G_n$ è una biiezione. Osserviamo inoltre che $C^{-1} = C$.

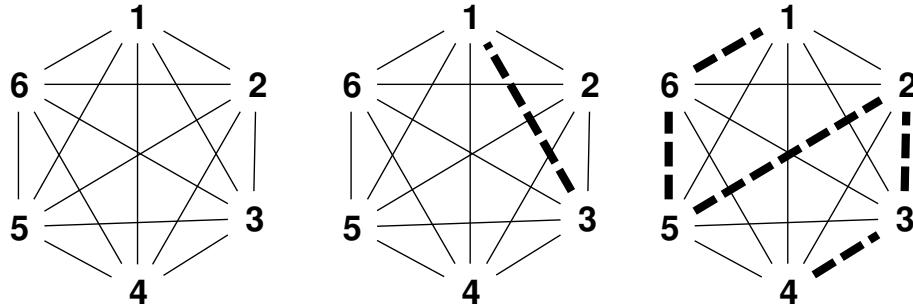


Figura 6.3: K_6 , ovvero una cricca di 6 nodi (sinistra), lo shortest path tra 1 e 3 (centro), un path hamiltoniano (destra)

Infine, se abbiamo $\frac{n(n-1)}{2}$ possibili archi, tutti i grafi complemento di un grafo G con m archi avranno esattamente $\frac{n(n-1)}{2} - m$ archi (tutti gli archi possibili, meno gli m di G). Per la stessa ragione, tutti i grafi con $\frac{n(n-1)}{2} - m$ archi hanno come complemento un grafo di m archi. Possiamo quindi concludere che il numero di grafi con n nodi e m archi è sempre uguale al numero di grafi con n nodi e $\frac{n(n-1)}{2} - m$ archi.

Contare cammini in grafi notevoli

Vediamo in questa sezione domande del tipo “quanti cammini di un certo tipo ci sono in un grafo?”, e come possiamo utilizzare le tecniche che abbiamo visto in precedenza per rispondervi.

Cammini in una cricca

Definizione 6.4.6 (cricca). *Una cricca di n nodi è un grafo dove ogni coppia di nodi è connessa. La cricca viene anche denotata come K_n , e detta grafo completo,*

Una prima domanda che possiamo chiederci è:

Esempio 6.4.7. *Quanti shortest paths ci sono tra due nodi distinti x e y in una cricca?*

La risposta è 1, dato che se esistono tutti gli archi esiste sempre il cammino xy di lunghezza 1. Qualsiasi altro cammino avrà lunghezza maggiore.

Ricordiamo adesso la definizione di path hamiltoniano: questo è un path che tocca *tutti* i nodi del grafo esattamente una volta. Sicuramente esiste almeno un path hamiltoniano in una cricca (se ne vede uno in figura), ma quanti ne esistono?

Esempio 6.4.8. *In una cricca, qualsiasi sequenza di nodi v_1, v_2, \dots senza ripetizioni corrisponde a un path, in quanto per ogni coppia v_i, v_j esiste sempre l'arco v_i, v_j .*

Ne consegue che qualsiasi permutazione dei nodi $1, 2, \dots, n$ corrisponde ad un path hamiltoniano, dato che ogni nodo appare esattamente una volta nella sequenza. Possiamo convincerci di questo scrivendo una permutazione qualsiasi, e provando a seguire il corrispondente cammino nel grafo in figura.

Quindi K_n ha esattamente $n!$ path hamiltoniani.

Un concetto simile al path hamiltoniano è quello di *ciclo hamiltoniano*: questo è un ciclo che tocca ogni nodo del grafo esattamente una volta.

Quanti cicli hamiltoniani esistono?

Esempio 6.4.9. *Se guardiamo un path hamiltoniano che va da i a j , notiamo che possiamo sempre aggiungere l'arco ij e trasformare il path in un ciclo hamiltoniano.*

Ne consegue che K_n ha $n!$ cicli hamiltoniani.

Tuttavia, osserviamo che alcuni cicli sono sostanzialmente equivalenti: il ciclo $1, 6, 5, 2, 3, 4, 1$ è composto dagli stessi archi del ciclo “opposto” $1, 4, 3, 2, 5, 6, 1$.

Poniamoci allora la domanda: quanti cicli hamiltoniani *non equivalenti* esistono in K_n ?

Esempio 6.4.10. Abbiamo osservato come un ciclo e il suo simmetrico ottenuto percorrendo gli archi in senso opposto sono composti dagli stessi archi, quindi equivalenti.

Proviamo ora a percorrere un ciclo due volte: 1, 6, 5, 2, 3, 4, 1, 6, 5, 2, 3, 4, 1 notiamo che se iniziamo da 1 otteniamo il ciclo 1, 6, 5, 2, 3, 4, 1, ma iniziando da un qualsiasi altro dei primi n nodi otteniamo uno nuovo ciclo hamiltoniano, che usa tuttavia gli stessi archi: ad esempio 2, 3, 4, 1, 6, 5, 2 partendo da 2 o 4, 1, 6, 5, 2, 3, 4, 1 partendo da 4. Possiamo chiamare questi cicli rotazioni del ciclo originale.

Per ogni ciclo, quindi, sappiamo che esistono un totale di $2n$ cicli hamiltoniani equivalenti ad esso: n rotazioni e per ognuna il suo simmetrico.

Ne concludiamo che K_n ha $\frac{n!}{2n}$ cicli hamiltoniani non equivalenti.

Quanti path di k nodi esistono in K_n ?

Esempio 6.4.11. Proviamo a costruire un cammino di k nodi: abbiamo n scelte per il primo nodo, poi $n - 1$ per chi sarà il secondo, $n - 2$ per il terzo, e così via fino al k -esimo per cui avremo $n - (k - 1)$ scelte.

Il numero di cammini di k nodi in K_n è quindi:

$$(n)(n - 1) \cdots (n - k + 1) = \frac{(n)(n - 1) \cdots (1)}{(n - k) \cdots (1)} = \frac{n!}{(n - k)!}$$

Notiamo anche che qualsiasi sequenza di k nodi distinti corrisponde a un valido cammino, e che quindi il numero di cammini di k nodi in K_n corrisponde al numero di possibili sequenze ordinate di lunghezza k (ovvero k -permutazioni) di n elementi.

Possiamo osservare che la formula qui sopra ci porta a una dimostrazione costruttiva della cosiddetta formula dei tre fattoriali: Se ci chiediamo quanti sono i sottoinsiemi di dimensione k di n elementi (denotato dal coefficiente binomiale $\binom{n}{k}$), notiamo che ogni sequenza di lunghezza k corrisponde ad un sottoinsieme, tuttavia, esistono esattamente $k!$ permutazioni di ogni sottoinsieme, corrispondenti a $k!$ sequenze che corrispondono allo stesso sottoinsieme.

Ne consegue che il numero di sequenze di lunghezza k è pari a $k!$ volte il numero di sottoinsiemi di dimensione k . Quindi che $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

Cammini nel grafo bipartito completo

Il grafo $K_{n,n}$ denota il grafo *bipartito completo* con n nodi su ogni partizione.

Definizione 6.4.12 (grafo bipartito). Un grafo $G = (V, E)$ è bipartito se sono vere le seguenti condizioni (tutte equivalenti tra loro):

- Esiste una partizione V_1, V_2 di V , i.e., con $V_1 \cap V_2 = \emptyset$ e $v_1 \cup v_2 = V$, tale che non esistono archi tra nodi della stessa partizione. Ovvero, per ogni arco $xy \in E$ vale $x \in V_1 \implies y \in V_2$.
- Dati due colori, è possibile colorare ogni nodo di G di un colore in modo che i due estremi di ogni arco abbiano sempre colore diverso (chiamata 2-colorazione).
- G non contiene cicli di lunghezza dispari.

Osservazione 6.4.13. Gli alberi sono grafi che non hanno cicli. Ne consegue che un albero non ha cicli dispari, e quindi tutti gli alberi sono grafi bipartiti. E' in effetti facile trovare una 2-colorazione di un albero se coloriamo la radice del primo colore, poi i suoi figli del secondo, e procediamo a colorare ogni livello altermando tra i due colori.

E' anche possibile dimostrare che un grafo bipartito, se connesso, può essere partizionato in un solo modo.

Esercizio 6.4.14. Dato un grafo bipartito connesso, dimostrare che esiste una sola bipartizione V_1, V_2 .¹

¹Suggerimento: osserviamo la lunghezza dei cammini tra due nodi.

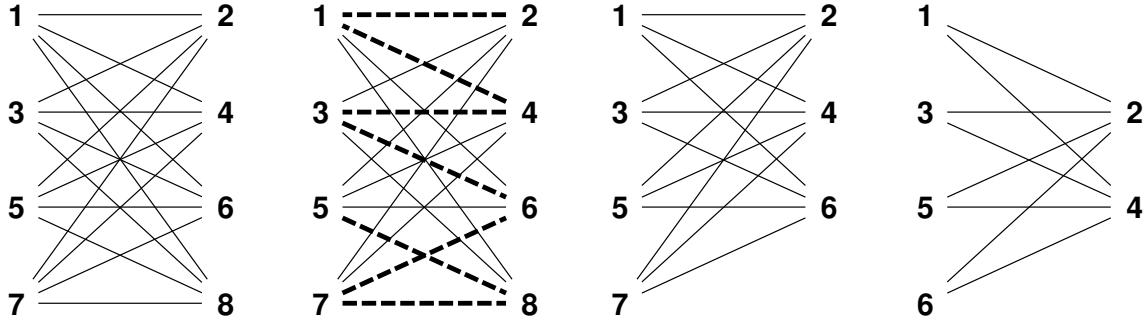


Figura 6.4: Esempi di grafi bipartiti completi. In ordine: Il grafo $K_{4,4}$, un suo path hamiltoniano, il grafo $K_{4,3}$, il grafo $K_{4,2}$.

Un grafo bipartito *completo* è un grafo dove, data la partizione V_1, V_2 , ogni nodo di V_1 è adiacente a *tutti* i nodi di V_2 (e chiaramente viceversa).

Chiediamoci anche qui, quanti shortest path esistono tra due nodi?

Esempio 6.4.15. *Dati $x, y \in K_{n,n}$, esiste un solo shortest path tra x e y se $x \in V_1$ e $y \in V_2$, dato che esiste l'arco xy . Se invece $x, y \in V_1$, allora esistono n shortest path tra x e y : per ogni $z \in V_2$, xz, zy è uno shortest path tra x e y .*

Ragioniamo ora sui path hamiltoniani:

Esempio 6.4.16. *In qualsiasi cammino su $K_{n,n}$ a ogni passo devo alternare tra V_1 e V_2 : Assumiamo di partire da V_1 ; il cammino sarà della forma $a_1, b_1, a_2, b_2, a_3, b_3, \dots, a_n, b_n$, dove tutti gli a_i appartengono a V_1 e tutti i b_i appartengono a V_2 . Tuttavia, osserviamo che per qualsiasi coppia a_i, b_j , l'arco $a_i b_j$ esiste e può essere percorso nel path.*

Possiamo quindi prendere qualsiasi permutazione di V_1 , qualsiasi permutazione di V_2 , e alternare i loro nodi uno ad uno per ottenere un valido path hamiltoniano. Visto che V_1 e V_2 hanno entrambi dimensione n , ognuno ammette $n!$ permutazioni. Inoltre, abbiamo assunto di partire da V_1 : otteniamo altrettanti path hamiltoniani se partiamo invece da V_2 , guadagnando quindi un ulteriore fattore 2.

Il grafo $K_{n,n}$ ha quindi $2(n!)^2$ path hamiltoniani.

Possiamo ragionare similmente al caso delle cricche per trovare invece i cicli hamiltoniani:

Esempio 6.4.17. *Ogni path hamiltoniano in $K_{n,n}$ che inizia da $a_1 \in V_1$ finisce in $b_n \in V_2$, o viceversa. E' quindi sempre disponibile l'arco $a_1 b_n$ (o $b_1 a_n$) che collega il primo e ultimo nodo, trasformando il path hamiltoniano in un ciclo hamiltoniano.*

Il grafo $K_{n,n}$ ha quindi $2(n!)^2$ cicli hamiltoniani.

Anche in questo caso, alcuni cicli sono equivalenti, ovvero composti dagli stessi archi: ogni ciclo ha lunghezza $2n$ e quindi lo troveremo in $2n$ rotazioni (vedi Esempio 6.4.10), e ognuna di queste avrà un suo ciclo simmetrico.

Il grafo $K_{n,n}$ ha quindi $\frac{2(n!)^2}{2 \cdot 2n} = \frac{n!(n-1)!}{2}$ cicli hamiltoniani non equivalenti.

Guardiamo ora grafi con un numero di nodi diverso tra le due partizioni.

Esempio 6.4.18. *In Figura 6.4 possiamo vedere il grafo $K_{4,3}$. E' possibile costruire un path hamiltoniano procedendo come per $K_{n,n}$. Tuttavia, notiamo che possiamo farlo solo partendo dal lato destro: possiamo intercalare 4 elementi $a_i \in V_1$ con 3 elementi $b_i \in V_2$ solo se iniziamo e terminiamo con a: $a_1, b_1, a_2, b_2, a_3, b_3, a_4$.*

Otteniamo quindi tutte le possibili permutazioni di 4, ovvero $4!$, intercalate con le permutazioni di 3, ovvero $3!$, ma perdiamo il fattore 2 rispetto a $K_{n,n}$ in quanto non abbiamo la scelta del lato iniziale.

Il grafo $K_{n,n-1}$ ha quindi $n!(n-1)!$ path hamiltoniani.

Riusciamo, utilizzando la stessa logica dell'Esempio 6.4.17, a costruire cicli hamiltoniani in $K_{4,3}$, e in generale, del grafo $K_{n,(n-1)}$?

Esercizio 6.4.19. Determinare il numero di cicli hamiltoniani nel grafo $K_{4,3}$ e $K_{n,n-1}$.

Riusciamo a costruire path hamiltoniani nel grafo $K_{4,2}$ (in Figura 6.4), e in generale, $K_{n,(n-2)}$?

Esercizio 6.4.20. Determinare il numero di path hamiltoniani nel grafo $K_{n,(n-2)}$.

CAPITOLO 7

Induzione Strutturale e Ricorsione

Nel Capitolo 3 abbiamo presentato una definizione induttiva dei numeri naturali e la abbiamo sfruttata per definire induttivamente delle funzioni su di essi, nonché per dimostrare delle proprietà in modo induttivo, sfruttando il Principio di Induzione sui naturali. Come per i numeri naturali, la maggior parte degli oggetti di studio dell'informatica sono definiti *induttivamente*, cioè sono definiti in termini di istanze più piccole di loro stessi, e questo è l'argomento principale di questo capitolo. Infatti molte strutture dati, come le liste e gli alberi binari, e perfino gli stessi programmi sono costruiti a partire da oggetti più piccoli dello stesso tipo. Per esempio, due programmi messi uno dopo l'altro (tipicamente con un punto e virgola nel mezzo) in modo che l'esecuzione del secondo inizi quando il primo è terminato, non sono altro che un programma definito in termini di due programmi più piccoli. Inoltre le funzioni definite su questi oggetti sono spesso definite anche esse in modo induttivo: il valore della funzione su un oggetto definito induttivamente è ottenuto componendo i valori della stessa funzione su oggetti più piccoli. La stessa tecnica si può usare più in generale per definire relazioni o proprietà su tali oggetti.

Le definizioni induttive, oltre a essere concise e matematicamente rigorose, garantiscono la correttezza di una tecnica di dimostrazione molto efficace, chiamata il *Principio di Induzione Strutturale*, che può essere usato appunto per dimostrare proprietà di funzioni definite su insiemi presentati induttivamente. Come vedremo, le definizioni induttive sono un caso particolare delle definizioni *ricorsive*, nelle quali in generale si permette di definire una funzione in termini del suo valore su oggetti arbitrari, non necessariamente più piccoli.

In questo capitolo illustreremo l'induzione su liste, alberi binari e alberi binari etichettati. In particolare, per ognuna di queste strutture mostremo (1) la loro definizione induttiva, (2) alcune funzioni definite induttivamente su di esse e (3) come dimostrare proprietà di tali strutture utilizzando il Principio di Induzione. Vedremo poi che questo principio può essere definito in maniera generale, cioè indipendentemente dalla struttura sotto mano, utilizzando il concetto di *termine*. Infine affronteremo la ricorsione, illustrando una tecnica sufficiente a garantire che la definizione di funzioni ricorsive sia ben data.

7.1 Liste

Le liste sono un tipo di dati di uso comune nei linguaggi di programmazione. Esse consentono la memorizzazione di sequenze di dati di lunghezza variabile (a differenza degli array), normalmente di tipo omogeneo. In molti linguaggi di programmazione, la lista contenente la sequenza $a_1, a_2 \dots, a_k$ è denotata come $[a_1, a_2, \dots, a_k]$, cioè delimitata dalle parentesi quadrate [e] e con gli elementi separati da virgolette. La *lista vuota*, denotata da [], rappresenta una sequenza senza elementi.

Il lettore avrà sicuramente notato la somiglianza con gli insiemi, dove al posto delle parentesi quadrate [e], si usano le graffe { e }. È quindi opportuno osservare che liste e insiemi sono concetti ben distinti: come abbiamo già spiegato nella Sezione 1.2, negli insiemi l'ordinamento e la molteplicità degli elementi non contano, mentre sono rilevanti nelle liste. Ad esempio $\{a, b, b, c\}$ e $\{c, a, b\}$ rappresentano lo stesso insieme, mentre $[a, b, b, c]$ e $[c, a, b]$ sono liste ben distinte.

Inoltre è importante dire che le liste sono sempre sequenze *finite* e che tutti gli elementi a_i appartengono ad uno stesso insieme A . Ad esempio, prendendo $A = \mathbb{N}$, $[1, 1, 2, 3]$ è una lista di numeri naturali, mentre l'intera sequenza di Fibonacci $[1, 1, 2, 3, 5, \dots]$ non è una lista, in quanto è infinita.

Come tutti i numeri naturali possono essere costruiti a partire da 0 con l'operazione $_ + 1$, così tutte le liste posso essere costruite a partire dalla lista vuota $[]$ utilizzando l'operazione $a : _$ che aggiunge un elemento $a \in A$ in testa ad una lista. Ad esempio, per $A = \mathbb{N}$, la lista $[1, 1, 2, 3]$ è ottenuta come $1 : (1 : (2 : (3 : [])))$. Infatti,

$$\begin{aligned} 3 : [] &= [3] \\ 2 : (3 : []) &= [2, 3] \\ 1 : (2 : (3 : [])) &= [1, 2, 3] \\ 1 : (1 : (2 : (3 : []))) &= [1, 1, 2, 3] \end{aligned}$$

In queste note, eviteremo di specificare le parentesi tonde (e) ed identificheremo sempre la lista $[a_1, a_2, \dots, a_k]$ con $a_1 : a_2 : \dots : a_k : []$.

A questo punto, possiamo dare la definizione induttiva dell'insieme delle liste di elementi appartenenti ad un insieme fissato A .

Definizione 7.1.1 (Liste). *L'insieme L_A delle liste di elementi di A è il più piccolo insieme che soddisfa:*

1. $[] \in L_A$; la lista vuota;
2. Per ogni $a \in A$, se $lst \in L_A$ allora $a : lst \in L_A$

Si noti l'analogia con la definizione induttiva dei numeri naturali (Definizione 3.1.1): la clausola base specifica che la lista vuota appartiene a L_A (nei numeri naturali $0 \in \mathbb{N}$); la clausola induttiva che $a : lst$ appartiene ad L_A per ogni elemento $a \in A$ ed ogni lista $lst \in L_A$ (nei numeri naturali $n + 1$ per ogni $n \in \mathbb{N}$). Nella Definizione 7.1.1, A è un insieme fissato ma arbitrario e, pertanto, può essere rimpiazzato con qualsiasi altro insieme. Ad esempio prendendo $A = \mathbb{N}$, si ottiene l'insieme $L_{\mathbb{N}}$ di liste di numeri naturali. Si noti infine che nella Definizione 7.1.1, non abbiamo specificato la clausola terminale: come abbiamo argomentato nel Capitolo 3, questa può essere omessa specificando “il più piccolo insieme”.

Osservazione 7.1.2. *Asserire che L_A è il più piccolo insieme che soddisfa le due proprietà della Definizione 7.1.1 significa che per ogni insieme B che soddisfa le due proprietà vale che $L_A \subseteq B$. In altre parole, se B è tale che*

1. $[] \in B$ e
 2. per ogni $a \in A$, se $lst \in B$ allora $a : lst \in B$,
- allora $L_A \subseteq B$.

Funzioni su liste

Come per i numeri naturali, anche per le liste possiamo sfruttare la loro definizione induttiva per definire induttivamente funzioni su L_A . Come per i naturali, è sufficiente specificare (1) il valore della funzione nel caso base, cioè la lista vuota $[]$ e (2) il valore della funzione nel caso induttivo, cioè $a : lst$ per ogni $a \in A$.

Vediamo adesso un po' di esempi di funzioni su liste.

Definizione 7.1.3 (Lunghezza). *La funzione $len : L_A \rightarrow \mathbb{N}$ è definita per induzione come*

1. $len([]) = 0$
2. $len(a : lst) = len(lst) + 1$, per ogni $a \in A$.

Intuitivamente la funzione len associa ad ogni lista la sua lunghezza (len sta per “length”, lunghezza in inglese). Ad esempio, $len([a, e, a]) = 3$. È utile mostrare come il numero naturale 3 sia effettivamente derivato dalla definizione appena data.

$$\begin{aligned}
len([a, e, a]) &= len(a : e : a : []) && \text{(Esplicitiamo la lista)} \\
&= len(e : a : []) + 1 && \text{(Def. di } len\text{, clausola induttiva)} \\
&= len(a : []) + 1 + 1 && \text{(Def. di } len\text{, clausola induttiva)} \\
&= len([]) + 1 + 1 + 1 && \text{(Def. di } len\text{, clausola induttiva)} \\
&= 0 + 1 + 1 + 1 && \text{(Def. di } len\text{, clausola base)} \\
&= 3
\end{aligned}$$

Osservazione 7.1.4. Si osservi che la valutazione di $len([a : e : a])$ assomiglia molto all'esecuzione di un programma e che la Definizione 7.1.3 può essere pensata come un algoritmo: rimuovi il primo elemento della lista e somma 1 fino a quando la lista non è vuota. In effetti, le definizioni di funzioni induttive e, più in generale le funzioni ricorsive che vedremo alla fine di questo capitolo, possono essere facilmente implementate nella maggior parte dei linguaggi di programmazione.

La funzione $len : L_A \rightarrow \mathbb{N}$ è definita su liste per un insieme arbitrario A . Nella seguente definizione l'insieme degli elementi A è quello dei numeri naturali.

Definizione 7.1.5 (Somma su lista). La funzione $sumList : L_{\mathbb{N}} \rightarrow \mathbb{N}$ è definita per induzione come

1. $sumList([]) = 0$
2. $sumList(n : lst) = sumList(lst) + n$, per ogni $n \in \mathbb{N}$.

La funzione $sumList$ prende in input una lista di numeri naturali e restituisce la somma di tutti i numeri della lista. Ad esempio $sumList([9, 5, 28])$ è 42:

$$\begin{aligned}
sumList([9, 5, 28]) &= sumList(9 : 5 : 28 : []) && \text{(Esplicitiamo la lista)} \\
&= sumList(5 : 28 : []) + 9 && \text{(Clausola induttiva)} \\
&= sumList(28 : []) + 5 + 9 && \text{(Clausola induttiva)} \\
&= sumList([]) + 28 + 5 + 9 && \text{(Clausola induttiva)} \\
&= 0 + 28 + 5 + 9 && \text{(Clausola base)} \\
&= 42
\end{aligned}$$

Esercizio 7.1.6. Definire per induzione la funzione $sqrList : L_{\mathbb{N}} \rightarrow L_{\mathbb{N}}$ che applicata a una lista lst restituisce la lista dei quadrati degli elementi di lst , nello stesso ordine.

Utilizzando l'induzione su L_A è possibile anche definire funzioni che hanno come insieme di partenza $L_A \times B$ per un qualche insieme B . Vediamo un esempio in cui B è esattamente l'insieme degli elementi A .

Definizione 7.1.7 (Appartenenza). La funzione $belList : L_A \times A \rightarrow \text{Bool}$ è definita per induzione come

1. $belList([], b) = \text{f}$, per ogni $b \in A$
2. $belList(a : lst, b) = \text{t}$, per ogni $a, b \in A$ tali che $a = b$.
3. $belList(a : lst, b) = belList(lst, b)$, per ogni $a, b \in A$ tali che $a \neq b$.

Intuitivamente la funzione $belList$ prende in input una lista su A ed un elemento di A e restituisce t se l'elemento appartiene alla lista e f altrimenti (bel sta per “belong”, appartiene in inglese). Ad esempio $belList([a, e, a], e)$ è t , mentre $belList([a, e, a], b)$ è f . È utile mostrare come questi valori siano effettivamente derivati dalla definizione data.

$$\begin{aligned}
belList([a, e, a], e) &= belList(a : e : a : [], e) && \text{(Esplicitiamo la lista)} \\
&= belList(e : a : [], e) && \text{(Def. di } belList\text{, clausola 3.)} \\
&= \text{t} && \text{(Def. di } belList\text{, clausola 2.)}
\end{aligned}$$

$$\begin{aligned}
 \text{belList}([a, e, a], b) &= \text{belList}(a : e : a : [], b) \quad (\text{Esplicitiamo la lista}) \\
 &= \text{belList}(e : a : [], b) \quad (\text{Def. di } \text{belList}, \text{ clausola 3.}) \\
 &= \text{belList}(a : [], b) \quad (\text{Def. di } \text{belList}, \text{ clausola 3.}) \\
 &= \text{belList}([], b) \quad (\text{Def. di } \text{belList}, \text{ clausola 3.}) \\
 &= \mathbf{f} \quad (\text{Def. di } \text{belList}, \text{ clausola 1.})
 \end{aligned}$$

Osservazione 7.1.8. La Definizione 7.1.7 può essere pensata come un algoritmo che scorre la lista da sinistra a destra cercando un elemento b . Appena lo trova restituisce \mathbf{t} ; se non lo trova, cioè arriva alla lista vuota, restituisce \mathbf{f} .

Nella seguente funzione l'insieme di partenza è ancora della forma $L_A \times B$, ma stavolta B è lo stesso L_A , cioè i dati di input sono coppie di liste (lst_1, lst_2) . L'induzione viene comunque utilizzata solo su lst_1 .

Definizione 7.1.9 (Concatenazione). La funzione $\text{app} : L_A \times L_A \rightarrow L_A$ è definita per induzione come

1. $\text{app}([], lst_2) = lst_2$
2. $\text{app}(a : lst_1, lst_2) = a : \text{app}(lst_1, lst_2)$

Intuitivamente la funzione app prende in input due liste e restituisce la loro concatenazione (app sta per “append”, concatenazione in inglese). Ad esempio $\text{app}([a, e, a], [e])$ è la lista $[a, e, a, e]$.

Esercizio 7.1.10. Valutare $\text{app}([a, e, a], [e])$ utilizzando la Definizione 7.1.9.

La seguente definizione verrà utilizzata fra poco per mostrare il Principio di Induzione su liste.

Definizione 7.1.11 (Inversione di liste). La funzione $\text{rev} : L_A \rightarrow L_A$ è definita per induzione come

1. $\text{rev}([]) = []$
2. $\text{rev}(a : lst) = \text{app}(\text{rev}(lst), a : [])$

Intuitivamente rev prende come argomento una lista lst e restituisce la lista ottenuta invertendo l'ordine degli elementi di lst (rev è abbreviazione di “reverse”). Per esempio, vediamo come si valuta $\text{rev}([b, e, a])$:¹

$$\begin{aligned}
 \text{rev}([b, e, a]) &= \text{rev}(b : e : a : []) \quad (\text{Esplicitiamo la lista}) \\
 &= \text{app}(\text{rev}(e : a : []), b : []) \quad (\text{Def. di } \text{rev}, \text{ clausola induttiva}) \\
 &= \text{app}(\text{app}(\text{rev}(a : []), e : []), b : []) \quad (\text{rev}, \text{ clausola induttiva}) \\
 &= \text{app}(\text{app}(\text{app}(\text{rev}([]), a : []), e : []), b : []) \quad (\text{rev}, \text{ clausola induttiva}) \\
 &= \text{app}(\text{app}(\text{app}([], a : []), e : []), b : []) \quad (\text{rev}, \text{ clausola base}) \\
 &= \text{app}(\text{app}(a : [], e : []), b : []) \quad (\text{app}, \text{ clausola base}) \\
 &= \text{app}(a : (\text{app}([], e : [])), b : []) \quad (\text{app}, \text{ clausola induttiva}) \\
 &= \text{app}(a : (e : []), b : []) \quad (\text{app}, \text{ clausola base}) \\
 &= a : \text{app}(e : [], b : []) \quad (\text{app}, \text{ clausola induttiva}) \\
 &= a : e : \text{app}([], b : []) \quad (\text{app}, \text{ clausola induttiva}) \\
 &= a : e : b : [] \quad (\text{app}, \text{ clausola base}) \\
 &= [a, e, b]
 \end{aligned}$$

Quindi applicando le definizioni date abbiamo avuto bisogno di circa 10 passi per calcolare l'inverso di una lista di tre elementi. Si può dimostrare che per invertire una lista di n elementi sono necessari

¹In caso di ambiguità sottolineamo dove abbiamo applicato la giustificazione.

un numero di passi dell'ordine di n^2 . Anche se la complessità computazionale della valutazione delle funzioni esula dagli argomenti di questo corso, è opportuno presentare una diversa definizione induttiva della funzione *reverse* che ha un'efficienza maggiore.

Esempio 7.1.12. La funzione $\text{rev1} : L_A \times L_A \rightarrow L_A$ è definita per induzione sul primo argomento come

1. $\text{rev1}([], lst) = lst$
2. $\text{rev1}(a : lst_1, lst_2) = \text{rev1}(lst_1, a : lst_2)$

Per invertire una lista lst basta valutare $\text{rev1}(lst, [])$. Per convincerci della correttezza della definizione, descriviamo la valutazione di $\text{rev1}([b, e, a], [])$:

$$\begin{aligned}
 \text{rev1}([b, e, a], []) &= \text{rev1}(b : a : [], []) && (\text{Esplicitiamo la lista}) \\
 &= \text{rev1}(e : a : [], b : []) && (\text{Clausola induttiva}) \\
 &= \text{rev1}(a : [], e : b : []) && (\text{Clausola induttiva}) \\
 &= \text{rev1}([], a : e : b : []) && (\text{Clausola induttiva}) \\
 &= a : e : b : [] && (\text{Clausola base}) \\
 &= [a, e, b]
 \end{aligned}$$

Quindi abbiamo ottenuto il risultato in 4 passi: per invertire una lista di n elementi con rev1 è necessario un numero di passi dell'ordine di n .

Esempio 7.1.13. L'inserimento ordinato di un numero naturale in una lista è la funzione $\text{ins} : \mathbb{N} \times L_{\mathbb{N}} \rightarrow L_{\mathbb{N}}$ definita per induzione come

1. $\text{ins}(n, []) = n : []$
2. $\text{ins}(n, m : lst) = \begin{cases} m : \text{ins}(n, lst) & \text{se } n > m \\ n : m : lst & \text{altrimenti} \end{cases}$

La funzione ins , assume che la lista di naturali passata come secondo argomento sia ordinata in modo crescente (ogni numero è minore o uguale al successivo), e inserisce nella lista il primo argomento garantendo che la lista risultante sia ancora ordinata. Vediamo un esempio di valutazione.

$$\begin{aligned}
 \text{ins}(5, [1, 3, 7]) &= \text{ins}(5, 1 : 3 : 7 : []) && (\text{Esplicitiamo la lista}) \\
 &= 1 : \text{ins}(5, 3 : 7 : []) && (\text{Clausola induttiva}, 5 > 1) \\
 &= 1 : 3 : \text{ins}(5, 7 : []) && (\text{Clausola induttiva}, 5 > 3) \\
 &= 1 : 3 : 5 : (7 : []) && (\text{Clausola induttiva}, 5 \leq 7) \\
 &= [1, 3, 5, 7]
 \end{aligned}$$

Esercizio 7.1.14.

1. Definire per induzione la funzione $\text{maxList} : L_N \rightarrow \mathbb{N} \cup \{-1\}$ che restituisce il massimo di una lista di naturali (-1 se la lista è vuota).
2. Definire per induzione la funzione $\text{preds} : \mathbb{N} \rightarrow L_{\mathbb{N}}$ che per ogni $n \in \mathbb{N}$ restituisce la lista dei suoi predecessori $[n - 1, n - 2, \dots, 0]$. Definire quindi la funzione $\text{predsList} : L_{\mathbb{N}} \rightarrow L_{L_{\mathbb{N}}}$ che applica preds a tutti gli elementi di una lista di naturali, restituendo la lista dei risultati.

Il Principio di Induzione sulle liste

Come per i naturali, anche per le liste possiamo utilizzare l'induzione per dimostrare che una generica proprietà $P(lst)$ sulle liste, cioè un'asserzione che può essere vera o falsa al variare di $lst \in L_A$, vale per tutte le liste lst .²

Il PRINCIPIO DI INDUZIONE SULLE LISTE stabilisce che:

²Si ricorda al lettore che il concetto di proprietà è introdotto formalmente nella Definizione 2.5.16.

7. Induzione Strutturale e Ricorsione

1. Se (caso base), $P([])$ è vera, e
2. se (passo induttivo), per ogni $a \in A$, per ogni lista $lst' \in L_A$, vale che se $P(lst')$ è vera allora anche $P(a : lst')$ è vera,

allora $P(lst)$ è vera per ogni $lst \in L_A$.

In modo più compatto possiamo scrivere il Principio di Induzione sulle liste come una *regola di inferenza*:

$$\frac{P([]) \quad \forall a \in A . \forall lst' \in L_A . P(lst') \Rightarrow P(a : lst')}{\forall lst \in L_A . P(lst)} \text{ P.I. su } L_A$$

Come esempi di principio di induzione sulle liste, presentiamo la dimostrazione delle prime tre proprietà della seguente proposizione, lasciando al lettore le ultime tre come esercizio.

Proposizione 7.1.15 (proprietà delle funzioni su liste). *Le seguenti uguaglianze sono vere per ogni $lst, lst_1, lst_2, lst_3 \in L_A$:*

1. $app(lst, []) = lst$
2. $app(app(lst_1, lst_2), lst_3) = app(lst_1, app(lst_2, lst_3))$
3. $rev(app(lst_1, lst_2)) = app(rev(lst_2), rev(lst_1))$
4. $len(app(lst_1, lst_2)) = len(lst_1) + len(lst_2)$
5. $len(rev(lst)) = len(lst)$
6. $belList(app(lst_1, lst_2), b) = belList(lst_1, b) \vee belList(lst_2, b)$

La prima proprietà stabilisce che la lista vuota $[]$ si comporta come unità della concatenazione, cioè concatenando una qualsiasi lista lst con la lista vuota si ottiene lst .

Dimostrazione di 7.1.15.1. Dimostriamo per induzione su lst che $app(lst, []) = lst$.

1. [CASO BASE] ($lst = []$).

$$app([], []) = [] \quad (\text{Clausola base } app)$$

2. [PASSO INDUTTIVO] ($lst = a : lst'$). Assumiamo come ipotesi induttiva che $app(lst', []) = lst'$, per dimostrare che $app(a : lst', []) = a : lst'$:

$$\begin{aligned} app(a : lst', []) &= a : app(lst', []) \quad (\text{Clausola induttiva } app) \\ &= a : lst' \quad (\text{Ipotesi induttiva}) \end{aligned}$$

■

È importante capire come, grazie al Principio di Induzione, la dimostrazione precedente permetta di concludere che la Proposizione 7.1.15.1 sia vera. La generica proprietà $P(lst)$ del Principio di Induzione, in questo caso specifico è

$$app(lst, []) = lst.$$

Il nostro obiettivo è dimostrare che $P(lst)$ è vera per ogni $lst \in L_A$. Il Principio di Induzione ci garantisce che per dimostrare questo è sufficiente dimostrare che (1) $P([])$ è vera, cioè che $app([], []) = []$ e (2) che $P(a : lst')$ è vera per ogni $a \in A$, cioè che $app(a : lst', []) = a : lst'$, assumendo che $P(lst')$ sia vera, cioè che $app(lst', []) = lst'$. Questa assunzione è l'ipotesi induttiva.

Dimostrazione di 7.1.15.2. Dimostriamo per induzione su lst_1 che $app(app(lst_1, lst_2), lst_3) = app(lst_1, app(lst_2, lst_3))$.

1. [CASO BASE] ($lst_1 = []$).

$$\begin{aligned} app(app([], lst_2), lst_3) &= app(lst_2, lst_3) && \text{(Clausola base } app\text{)} \\ &= app([], app(lst_2, lst_3)) && \text{(Clausola base } app\text{)} \end{aligned}$$

2. [PASSO INDUTTIVO] ($lst_1 = a : lst'_1$). Assumiamo come ipotesi induttiva che $app(app(lst'_1, lst_2), lst_3) = app(lst'_1, app(lst_2, lst_3))$, per dimostrare che $app(app(a : lst'_1, lst_2), lst_3) = app(a : lst'_1, app(lst_2, lst_3))$:

$$\begin{aligned} app(app(a : lst'_1, lst_2), lst_3) &= \underline{app(a : app(lst'_1, lst_2), lst_3)} && \text{(Clausola induttiva } app\text{)} \\ &= a : \underline{app(app(lst'_1, lst_2), lst_3)} && \text{(Clausola induttiva } app\text{)} \\ &= a : \underline{app(lst'_1, app(lst_2, lst_3))} && \text{(Ipotesi induttiva)} \\ &= app(a : lst'_1, app(lst_2, lst_3)) && \text{(Clausola induttiva } app\text{)} \end{aligned}$$

■

Anche in questo caso è opportuno illustrare come, grazie al Principio di Induzione, la dimostrazione permetta di concludere che la Proposizione 7.1.15.2 sia vera.

L'enunciato di tale proposizione esprime una proprietà che vale per tutte le triple di liste $lst_1, lst_2, lst_3 \in L_A \times L_A \times L_A$. Invece, il principio di induzione su liste ci permette di dimostrare che una proprietà $P(lst)$ vale per tutte le liste $lst \in L_A$. In questi casi, l'induzione può essere comunque utilizzata, ma si deve scegliere una lista sulla quale applicarla. Infatti, all'inizio della dimostrazione, si specifica che l'induzione è fatta su lst_1 . Questo significa che prendiamo come proprietà $P(lst_1)$ la seguente:

per ogni $lst_2, lst_3 \in L_A$, vale che $app(app(lst_1, lst_2), lst_3) = app(lst_1, app(lst_2, lst_3))$.

Il nostro obiettivo è dimostrare che $P(lst_1)$ è vera per ogni $lst_1 \in L_A$. Il Principio di Induzione ci garantisce che per dimostrare questo è sufficiente dimostrare che (1) $P([])$ è vera, cioè che $app(app([], lst_2), lst_3) = app([], app(lst_2, lst_3))$ e (2) che $P(a : lst')$ è vera per ogni $a \in A$, cioè che $app(app(a : lst'_1, lst_2), lst_3) = app(a : lst'_1, app(lst_2, lst_3))$, assumendo che $P(lst'_1)$ sia vera, cioè che $app(app(lst'_1, lst_2), lst_3) = app(lst'_1, app(lst_2, lst_3))$. Questa assunzione è l'ipotesi induttiva.

Dimostrazione di 7.1.15.3. Dimostriamo per induzione su lst_1 che $rev(app(lst_1, lst_2)) = app(rev(lst_2), rev(lst_1))$.

1. [CASO BASE] ($lst_1 = []$).

$$\begin{aligned} rev(app([], lst_2)) &= rev(lst_2) && \text{(Clausola base } app\text{)} \\ &= app(rev(lst_2), []) && \text{(Per la 7.1.15.1)} \\ &= app(rev(lst_2), rev([])) && \text{(Clausola base } rev\text{)} \end{aligned}$$

2. [PASSO INDUTTIVO] ($lst_1 = a : lst'_1$). Assumiamo come ipotesi induttiva che $rev(app(lst'_1, lst_2)) = app(rev(lst_2), rev(lst'_1))$, per dimostrare che $rev(app(a : lst'_1, lst_2)) = app(rev(lst_2), rev(a : lst'_1))$:

$$\begin{aligned} rev(app(a : lst'_1, lst_2)) &= rev(a : app(lst'_1, lst_2)) && \text{(Clausola induttiva } app\text{)} \\ &= app(\underline{rev(app(lst'_1, lst_2))}, a : []) && \text{(Clausola induttiva } rev\text{)} \\ &= \underline{app(app(rev(lst_2), rev(lst'_1)), a : [])} && \text{(Ipotesi induttiva)} \\ &= app(rev(lst_2), \underline{app(rev(lst'_1), a : [])}) && \text{(Per la 7.1.15.2)} \\ &= app(rev(lst_2), rev(a : lst'_1)) && \text{(Clausola induttiva } rev\text{)} \end{aligned}$$

■

Anche nella dimostrazione precedente, l'induzione è utilizzata su lst_1 . In altre parole, la proprietà $P(lst_1)$ che si dimostra valere per ogni $lst_1 \in L_A$ è la seguente:

per ogni $lst_2 \in L_A$, vale che $rev(app(lst_1, lst_2)) = app(rev(lst_2), rev(lst_1))$.

È interessante notare che in questa dimostrazione è necessario utilizzare alcune proprietà dimostrate in precedenza per app .

Esercizio 7.1.16. Dimostrare la Proposizione 7.1.15.4.

Esercizio 7.1.17. Dimostrare la Proposizione 7.1.15.5.

Esercizio 7.1.18. Dimostrare la Proposizione 7.1.15.6.

7.2 Alberi binari

Un altro tipo di dati ampiamente usato in Informatica sono gli *alberi binari*. Secondo la classificazione presentata nella Definizione 5.6.8, si tratta di *alberi radicati cardinali con k=2*, ma ne presentiamo qui una definizione induttiva che caratterizza un insieme di alberi leggermente diverso. Infatti mentre per la definizione vista nel Capitolo 5 un albero ha almeno un nodo, nella nostra definizione un albero può essere anche vuoto. In informatica entrambe queste definizioni sono ampiamente usate, anche se in contesti leggermente diversi: la prima nella Teoria dei Grafi, quella che introduciamo ora in Algoritmica.

Definizione 7.2.1 (Alberi binari). *L'insieme BT degli alberi binari è il più piccolo insieme che soddisfa:*

1. $\lambda \in BT$, l'albero vuoto;
2. se $t_1, t_2 \in BT$ allora $N(t_1, t_2) \in BT$.

Quindi un albero binario può essere l'albero vuoto, che non contiene nessun nodo, oppure un *nodo* $N(t_1, t_2)$ con due *sottoalberi* t_1 e t_2 , chiamati anche rispettivamente il *sottoalbero sinistro* e il *sottoalbero destro*. Un nodo è una *foglia* se entrambi i suoi sottoalberi sono vuoti, altrimenti è un *nodo interno*.³ Le “relazioni di parentela” tra nodi di un albero binario (figlio, genitore, antenato, discendente) sono definite esattamente come nel Capitolo 5. Nel resto di questa sezione, poiché parleremo solo di alberi *binari*, spesso ometteremo la qualificazione “binario” considerandola sottointesa.

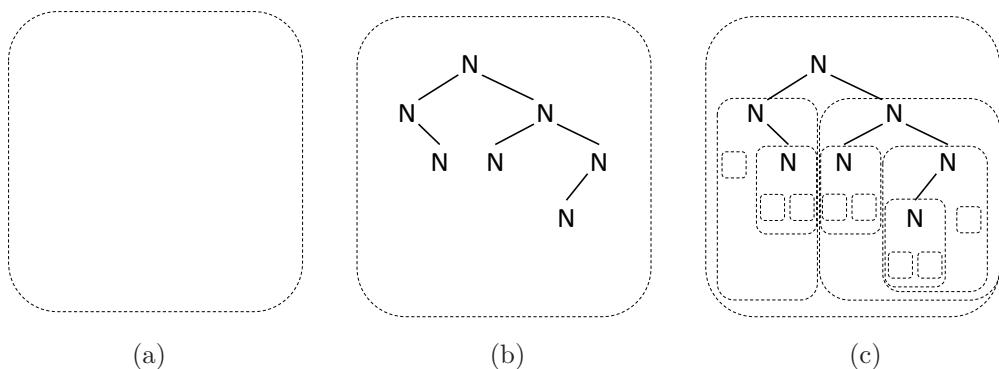


Figura 7.1: (a) L'albero vuoto; (b) Un albero binario; (c) Lo stesso albero in cui è evidenziata la struttura induttiva.

³Nella Definizione 5.6.1 le foglie sono definite come nodi con grado 1. Di fatto le due definizioni coincidono, tranne che per la radice. Infatti la radice può essere una foglia per la definizione appena data, ma non per quella del Capitolo 5.

La Figura 7.1 mostra una possibile rappresentazione grafica degli alberi appena definiti. L’albero vuoto (a) non ha una rappresentazione esplicita. La parte centrale, (b), mostra l’albero binario

$$t = N(N(\lambda, N(\lambda, \lambda)), N(N(\lambda, \lambda), N(N(\lambda, \lambda), \lambda)))$$

Il nodo più in alto è la *radice*, ci sono 3 foglie e 4 nodi interni, compresa la radice. La parte (c) della figura mostra lo stesso albero in cui abbiamo evidenziato la struttura induttiva, che richiede che ogni nodo N abbia un sottoalbero sinistro e uno destro, eventualmente vuoti.

Esercizio 7.2.2. *Usando la notazione della Figura 7.1, si disegni l’albero $N(\lambda, N(\lambda, \lambda))$.*

Esercizio 7.2.3. *Usando la notazione della Figura 7.1, si disegni l’albero $N(N(\lambda, \lambda), \lambda)$.*

Si noti l’analogia con la definizione induttiva dei numeri naturali (Definizione 3.1.1): la clausola base specifica che l’albero vuoto, denotato da λ , appartiene a BT (nei numeri naturali $0 \in \mathbb{N}$); la clausola induttiva che $N(t_1, t_2)$ appartiene ad BT per ogni albero binario $t_1, t_2 \in BT$ (nei numeri naturali $n + 1$ per ogni $n \in \mathbb{N}$). Si noti infine che nella Definizione 7.1.1, non abbiamo specificato la clausola terminale: come abbiamo argomentato nel Capitolo 3, questa può essere omessa specificando “il più piccolo insieme”.

Funzioni su Alberi Binari

Come per i numeri naturali e per le liste, anche per gli alberi binari possiamo sfruttare la loro definizione induttiva per definire induttivamente funzioni su BT . Come per naturali e liste, è sufficiente specificare (1) il valore della funzione nel caso base, cioè l’albero vuoto λ , e (2) il valore della funzione nel caso induttivo, cioè $N(t_1, t_2)$.

Vediamo adesso due esempi di funzioni su alberi binari, definite induttivamente.

Definizione 7.2.4 (Dimensione). *La funzione $size : BT \rightarrow \mathbb{N}$ è definita per induzione come*

1. $size(\lambda) = 0$.
2. $size(N(t_1, t_2)) = size(t_1) + size(t_2) + 1$

Intuitivamente la funzione *size* associa ad ogni albero la sua dimensione, cioè il suo numero di nodi. Ad esempio, la dimensione dell’albero in Figura 7.1.(b) è 7. Lasciamo la valutazione della funzione *size* su tale l’albero come esercizio e, di seguito, mostriamo la valutazione di *size* su un albero più piccolo (quello dell’Esercizio 7.2.2).

$$\begin{aligned} size(N(\lambda, N(\lambda, \lambda))) &= size(\lambda) + size(N(\lambda, \lambda)) + 1 && (\text{Def. di } size, \text{ clausola induttiva}) \\ &= 0 + size(N(\lambda, \lambda)) + 1 && (\text{Def. di } size, \text{ clausola base}) \\ &= 0 + size(\lambda) + size(\lambda) + 1 + 1 && (\text{Def. di } size, \text{ clausola induttiva}) \\ &= 0 + 0 + 0 + 1 + 1 && (\text{Def. di } size, \text{ clausola base}) \\ &= 2 \end{aligned}$$

Esercizio 7.2.5. *Valutare la funzione *size* sull’albero in Figura 7.1.(b).*

Definizione 7.2.6 (Altezza). *La funzione $height : BT \rightarrow \mathbb{N} \cup \{-1\}$ è definita per induzione come*

1. $height(\lambda) = -1$.
2. $height(N(t_1, t_2)) = \max(height(t_1), height(t_2)) + 1$

La funzione *height* associa ad ogni albero la sua altezza, cioè la lunghezza massima di un cammino dalla radice a una foglia (Definizione 5.8.7). Se consideriamo un albero del tipo $t = N(t_1, t_2)$, allora ogni cammino dalla radice di t a una foglia sarà composto da (1) un arco dalla radice di t alla radice di t_1 (o di t_2), e (2) da un cammino dalla radice di t_1 (o o di t_2) a una foglia. Questo spiega la clausola induttiva, che dice che l’altezza di t è uguale al massimo tra le altezze di t_1 e t_2 , più uno, per tener conto dell’arco aggiunto all’inizio di tutti i cammini alle foglie. Questa intuizione

ci aiuta a spiegare perché l'altezza dell'albero vuoto è -1 . Infatti consideriamo $t_0 = N(\lambda, \lambda)$, l'albero che ha un solo nodo, sia radice che foglia. Ovviamente la lunghezza del massimo cammino dalla radice a una foglia in t_0 è 0 , ma quindi usando la clausola induttiva di *height* dobbiamo avere $height(N(\lambda, \lambda)) = \max(height(\lambda), height(\lambda)) + 1 = height(\lambda) + 1 = 0$, da cui otteniamo $height(\lambda) = -1$.

Di seguito, mostriamo la valutazione di *height* sull'albero dell'Esercizio 7.2.2.

$$\begin{aligned}
 & height(N(\lambda, N(\lambda, \lambda))) \\
 &= \max(\underline{height(\lambda)}, height(N(\lambda, \lambda))) + 1 && (\text{Def. di } height, \text{ clausola induttiva}) \\
 &= \max(-1, height(N(\lambda, \lambda))) + 1 && (\text{Def. di } height, \text{ clausola base}) \\
 &= \max(-1, \max(\underline{height(\lambda)}, height(\lambda)) + 1) + 1 && (\text{Def. di } height, \text{ clausola induttiva}) \\
 &= \max(-1, \max(-1, -1) + 1) + 1 && (\text{Def. di } height, \text{ clausola base}) \\
 &= \max(-1, -1 + 1) + 1 = \max(-1, 0) + 1 = 0 + 1 = 1
 \end{aligned}$$

Esercizio 7.2.7. Valutare la funzione *height* sull'albero in Figura 7.1.(b).

Si noti che per entrambe le funzioni, nelle clausola induttiva la funzione viene invocata due volte, una volta per ogni sottoalbero. Questo fatto è del tutto naturale, vista la struttura della definizione induttiva di questi alberi.

Esercizio 7.2.8. Definire in modo induttivo la funzione $\text{numLeaves} : BT \rightarrow \mathbb{N}$ che applicata a un albero restituisce il numero di foglie.

Il Principio di Induzione sugli alberi binari

Come per le liste, anche per gli alberi possiamo utilizzare l'induzione per dimostrare che una generica proprietà $P(t)$ sugli alberi, cioè un'asserzione che può essere vera o falsa al variare di $t \in BT$, vale per tutti gli alberi t .

Il PRINCIPIO DI INDUZIONE SUGLI ALBERI BINARI stabilisce che:

1. Se (caso base), $P(\lambda)$ è vera, e
2. se (passo induttivo), per ogni $t_1, t_2 \in BT$, vale che se $P(t_1)$ è vera e $P(t_2)$ è vera allora anche $P(N(t_1, t_2))$ è vera,

allora $P(t)$ è vera per ogni $t \in BT$.

Come per le liste ed i numeri naturali possiamo utilizzare una regola di inferenza per esprimere in maniera compatta il principio di induzione:

$$\frac{P(\lambda) \quad \forall t_1, t_2 \in BT . (P(t_1) \wedge P(t_2) \Rightarrow P(N(t_1, t_2)))}{\forall t \in BT . P(t)} \text{ P.I. su } BT$$

Come esempio di principio di induzione sugli alberi, presentiamo una sola dimostrazione.

Proposizione 7.2.9. Per tutti gli alberi binari $t \in BT$ vale che

$$height(t) \leq size(t).$$

Dimostrazione. Dimostriamo per induzione su t che $height(t) \leq size(t)$.

1. [CASO BASE] ($t = \lambda$).

$$\begin{aligned}
 height(\lambda) &= -1 && (\text{Clausola base } height) \\
 &\leq 0 \\
 &= size(\lambda) && (\text{Clausola base di } size)
 \end{aligned}$$

2. [PASSO INDUTTIVO] ($t = N(t_1, t_2)$). Assumiamo come ipotesi induttiva che $height(t_1) \leq size(t_1)$ e $height(t_2) \leq size(t_2)$, per dimostrare che $height(N(t_1, t_2)) \leq size(N(t_1, t_2))$:

$$\begin{aligned} height(N(t_1, t_2)) &= \max(height(t_1), height(t_2)) + 1 && \text{(Clausola induttiva } height\text{)} \\ &\leq \max(size(t_1), size(t_2)) + 1 && \text{(Ipotesi induttiva)} \\ &\leq size(t_1) + size(t_2) + 1 \\ &= size(N(t_1, t_2)) && \text{(Clausola induttiva } size\text{)} \end{aligned}$$

■

Nella dimostrazione ci sono due passaggi senza alcuna giustificazione sulla destra: entrambi sono basati su delle ovvie leggi matematiche. Il primo, nel caso base, è $-1 \leq 0$. Il secondo, nel passo induttivo, usa il fatto ovvio che $\max(n, m) \leq n + m$ per tutti i numeri naturali $n, m \in \mathbb{N}$.

È importante capire come, grazie al Principio di Induzione, la dimostrazione precedente permetta di concludere che la Proposizione 7.2.9 sia vera. La generica proprietà $P(t)$ del Principio di Induzione, in questo caso specifico è

$$height(t) \leq size(t).$$

Il nostro obiettivo è dimostrare che $P(t)$ è vera per ogni $t \in BT$. Il Principio di Induzione ci garantisce che per dimostrare questo è sufficiente dimostrare che (1) $P(\lambda)$ è vera, cioè che $height(\lambda) \leq size(\lambda)$ e (2) che $P(N(t_1, t_2))$ è vera, cioè che $height(N(t_1, t_2)) \leq size(N(t_1, t_2))$, assumendo che $P(t_1)$ e $P(t_2)$ siano vere, cioè che $height(t_1) \leq size(t_1)$ e $height(t_2) \leq size(t_2)$. Queste due assunzioni formano l'ipotesi induttiva.

7.3 Alberi Binari Etichettati

Nella pratica siamo interessati a usare gli alberi binari come strutture dati, con la possibilità di memorizzare nelle foglie e/o nei nodi interni dei valori di un certo tipo. Introduciamo quindi una piccola variazione della definizione precedente: gli alberi binari etichettati su un insieme A .

Definizione 7.3.1 (Alberi binari etichettati). *L'insieme BT_A degli alberi binari etichettati con elementi di un insieme dato A è il più piccolo insieme che soddisfa:*

1. $\lambda \in BT_A$, l'albero vuoto;
2. se $t_1, t_2 \in BT_A$ allora $N(t_1, a, t_2) \in BT_A$ per ogni $a \in A$.

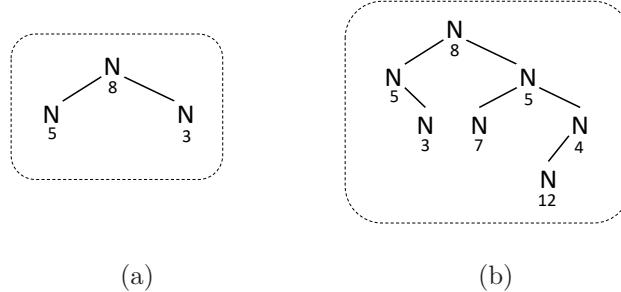


Figura 7.2: Due alberi binari etichettati sui naturali.

Quindi un albero binario etichettato può essere o l'albero vuoto, oppure un nodo $N(t_1, a, t_2)$ etichettato con a e con due sottoalberi t_1 e t_2 .

La Figura 7.2 mostra due alberi binari etichettati sui naturali, quindi elementi di $BT_{\mathbb{N}}$. Ogni etichetta è disegnata sotto il suo nodo. Quindi gli alberi in figura sono rispettivamente $N(N(\lambda, 5, \lambda), 8, N(\lambda, 3, \lambda))$ e

$$N(N(\lambda, 5, N(\lambda, 3, \lambda)), 8, N(N(\lambda, 7, \lambda), 5, N(N(\lambda, 12, \lambda), 4, \lambda)))$$

Esercizio 7.3.2. Usando la notazione della Figura 7.2, si disegni l'albero etichettato $N(\lambda, a, N(\lambda, b, \lambda))$.

Esercizio 7.3.3. Usando la notazione della Figura 7.2, si disegni l'albero etichettato $N(N(\lambda, b, \lambda), a, \lambda)$.

Funzioni su Alberi Binari Etichettati

Gli alberi binari sono spesso utilizzati per memorizzare insiemi di elementi. Dato un generico elemento a in A , uno è interessato a verificare se l'elemento appartiene o meno all'insieme rappresentato dall'albero. La seguente funzione prende in input un elemento di A ed un albero su A e restituisce come output t se l'elemento appartiene all'albero e f altrimenti.

Esempio 7.3.4. La funzione $belBT : BT_A \times A \rightarrow \text{Bool}$ è definita come

1. $belBT(\lambda, b) = f$.
2. $belBT(N(t_1, a, t_2), b) = t$, per ogni $a, b \in A$ tale che $a = b$.
3. $belBT(N(t_1, a, t_2), b) = belBT(t_1, b) \vee belBT(t_2, b)$, per ogni $a, b \in A$ tale che $a \neq b$.

Si noti che questa funzione è molto simile alla funzione su liste $belList : L_A \times A \rightarrow \text{Bool}$ della Definizione 7.1.7. La funzione che presentiamo ora, $sumBT$ è del tutto analoga a quella presentata nella Definizione 7.1.5.

Definizione 7.3.5 (Somma su albero). La funzione $sumBT : BT_{\mathbb{N}} \rightarrow \mathbb{N}$ definita per induzione come

1. $sumBT(\lambda) = 0$
2. $sumBT(N(t_1, n, t_2)) = sumBT(t_1) + sumBT(t_2) + n$

Intuitivamente, la funzione $sumBT$ prende in input un albero binario etichettato con numeri naturali (cioè l'insieme delle etichette A è esattamente \mathbb{N}) e restituisce la somma di tutte le etichette.

Esercizio 7.3.6. Valutare la funzione $belBT(T, 3)$, dove T è l'albero in Figura 7.2.(a).

Esercizio 7.3.7. Valutare la funzione $sumBT$ sull'albero in Figura 7.2.(a).

Esercizio 7.3.8. Definire in modo induttivo la funzione $maxBT : BT_{\mathbb{N}} \rightarrow \mathbb{N} \cup \{-1\}$ che applicata a un albero etichettato su \mathbb{N} restituisce il massimo tra tutte le etichette dei suoi nodi (-1 se l'albero è vuoto).

Molti algoritmi che operano su alberi binari sono basati sul concetto di *visita*, che consiste nel percorrere un cammino nell'albero, visitando sequenzialmente tutti i suoi nodi secondo una opportuna strategia. La visita di un nodo può essere associata a una opportuna azione, come stamparne l'etichetta, applicare all'etichetta una certa funzione, o comporre l'etichetta con altri valori per estrarre dall'albero un risultato. Vediamo una funzione che visita un albero binario etichettato su un insieme A e restituisce la lista delle etichette di tutti i nodi dell'albero, in un certo ordine, utilizzando la funzione *app* della Definizione 7.1.3.

Definizione 7.3.9 (Visita di albero in ordine simmetrico). La funzione $visit : BT_A \rightarrow L_A$ è definita come:

1. $visit(\lambda) = []$.
2. $visit(N(t_1, a, t_2)) = app(visit(t_1), a : visit(t_2))$

Questa funzione visita l'albero in *ordine simmetrico, da sinistra a destra*, perché per ogni nodo prima visita il sottoalbero sinistro, poi il nodo stesso e infine il sottoalbero destro: questo si vede chiaramente dagli argomenti di *app* nella clausola ricorsiva. Altri tipi di visite si possono ottenere cambiando l'ordine di queste tre azioni.

Vediamo per esempio la valutazione della funzione *visit* sull’albero di Figura 7.2 (a).

$$\begin{aligned}
 & visit(N(N(\lambda, 5, \lambda), 8, N(\lambda, 3, \lambda))) \\
 &= app(\underline{visit}(N(\lambda, 5, \lambda)), 8: \underline{visit}(N(\lambda, 3, \lambda))) && (\textit{visit}, \text{clausola induttiva}) \\
 &= app(app(\underline{visit}(\lambda), 5: \underline{visit}(\lambda)), 8: \underline{visit}(N(\lambda, 3, \lambda))) && (\textit{visit}, \text{clausola induttiva}) \\
 &= app(app([], 5: []), 8: \underline{visit}(N(\lambda, 3, \lambda))) && (\textit{visit}, \text{clausola base}) \\
 &= app(5: [], 8: \underline{visit}(N(\lambda, 3, \lambda))) && (\textit{app}, \text{clausola base}) \\
 &= app(5: [], 8: app(\underline{visit}(\lambda), 3: \underline{visit}(\lambda))) && (\textit{visit}, \text{clausola induttiva}) \\
 &= app(5: [], 8: app([], 3: [])) && (\textit{visit}, \text{clausola base}) \\
 &= app(5: [], 8: 3: []) && (\textit{app}, \text{clausola base}) \\
 &= 5: app([], 8: 3: []) && (\textit{app}, \text{clausola induttiva}) \\
 &= 5: 8: 3: [] && (\textit{app}, \text{clausola base}) \\
 &= [5, 8, 3]
 \end{aligned}$$

Il lettore è invitato a trovare l’ordine in cui vengono restituite le etichette dall’applicazione di *visit* all’albero di Figura 7.2 (b), e a confrontarlo con quello nella nota a piè di pagina. Valutare *visit* usando la definizione formale induttiva è molto laborioso, come appena visto, ma il risultato si può ottenere abbastanza rapidamente sfruttando la descrizione intuitiva della visita descritta sopra.⁴

Concludiamo introducendo un secondo tipo di visita che utilizzeremo tra poco per illustrare il principio di induzione.

Definizione 7.3.10 (Visita di albero in ordine anticipato). *La funzione $visit2 : BT_A \rightarrow L_A$ è definita come:*

1. $visit2(\lambda) = []$.
2. $visit2(N(t_1, a, t_2)) = a : app(visit2(t_1), visit2(t_2))$

Questa funzione visita l’albero in *ordine anticipato, da sinistra a destra*, perché per ogni nodo prima visita il nodo stesso, poi il sottoalbero sinistro e infine il sottoalbero destro.

Esercizio 7.3.11. *Valutare la funzione $visit2$ sull’albero in Figura 7.2.(a).*

Esercizio 7.3.12. *Quale lista è il risultato dell’applicazione della funzione $visit2$ all’albero in Figura 7.2.(b)?*

Il Principio di Induzione sugli alberi binari etichettati

Come per gli alberi binari, anche per quelli etichettati possiamo utilizzare l’induzione. L’unica differenza è che nel caso induttivo $N(t_1, a, t_2)$ dobbiamo assicurarsi che la proprietà valga per ogni possibile etichetta $a \in A$. Più precisamente, il PRINCIPIO DI INDUZIONE SUGLI ALBERI BINARI ETICHETTATI stabilisce che:

1. Se (caso base), $P(\lambda)$ è vera, e
2. se (passo induttivo), per ogni $a \in A$, per ogni $t_1, t_2 \in BT_A$, vale che se $P(t_1)$ è vera e $P(t_2)$ è vera allora anche $P(N(t_1, a, t_2))$ è vera,

allora $P(t)$ è vera per ogni $t \in BT_A$.

La corrispondente regola di inferenza è la seguente:

$$\frac{P(\lambda) \quad \forall a \in A. \forall t_1, t_2 \in BT_A. (P(t_1) \wedge P(t_2) \Rightarrow P(N(t_1, a, t_2)))}{\forall t \in BT_A. P(t)} \text{ P.I. su } BT_A$$

Come esempio di principio di induzione sugli alberi etichettati dimostriamo la seguente proposizione: trasformare prima un albero in una lista con *visit2* e poi applicare *belList* sulla lista risultante è uguale ad applicare direttamente *belBT* sull’albero. In sostanza, nessun elemento è perduto da *visit2*.

⁴Il risultato è la lista [5, 3, 8, 7, 5, 12, 4].

Proposizione 7.3.13. Per tutti gli alberi binari etichettati $t \in BT_A$ e tutte le etichette $b \in A$ vale che

$$belList(visit2(t), b) = belBT(t, b).$$

Dimostrazione. Dimostriamo per induzione su t che $belList(visit2(t), b) = belBT(t, b)$ per ogni $b \in A$.

1. [CASO BASE] ($t = \lambda$).

$$\begin{aligned} belList(visit2(\lambda), b) &= belList([\], b) && (\text{Clausola base } visit2) \\ &= \mathbf{f} && (\text{Clausola base di } belList) \\ &= belBT(\lambda, b) && (\text{Clausola base di } belBT) \end{aligned}$$

2. [PASSO INDUTTIVO] ($t = N(t_1, a, t_2)$). Assumiamo come ipotesi induttiva che $belList(visit2(t_1), b) = belBT(t_1, b)$ e $belList(visit2(t_2), b) = belBT(t_2, b)$, per dimostrare che $belList(visit2(N(t_1, a, t_2)), b) = belBT(N(t_1, a, t_2), b)$. Dobbiamo distinguere due casi

$$a = b$$

$$\begin{aligned} belList(visit2(N(t_1, a, t_2)), b) &= belList(a : app(visit2(t_1), visit2(t_2)), b) && (\text{Clausola induttiva } visit2) \\ &= \mathbf{t} && (\text{Clausola 2 di } belList) \\ &= belBT(N(t_1, a, t_2), b) && (\text{Clausola 2 di } belBT) \end{aligned}$$

$$a \neq b$$

$$\begin{aligned} belList(visit2(N(t_1, a, t_2)), b) &= belList(a : app(visit2(t_1), visit2(t_2)), b) && (\text{Clausola induttiva } visit2) \\ &= belList(app(visit2(t_1), visit2(t_2)), b) && (\text{Clausola 3 di } belList) \\ &= belList(visit2(t_1), b) \vee belList(visit2(t_2), b) && (\text{Proposizione 7.1.15}) \\ &= belBT(t_1, b) \vee belBT(t_2, b) && (\text{Ipotesi Induttiva}) \\ &= belBT(N(t_1, a, t_2), b) && (\text{Clausola 3 di } belBT) \end{aligned}$$

■

Esercizio 7.3.14. Dimostrare che per tutti gli alberi $t \in BT_{\mathbb{N}}$ vale che

$$sumList(visit2(t)) = sumBT(t).$$

Ma gli alberi sono alberi?

La definizione di albero binario della Definizione 7.2.1 sembra molto diversa dalle varie definizioni di albero viste nella Sezione 5.6. La rappresentazione grafica di Figura 7.1 (b) ci convince (abbastanza) che un albero binario come definito sopra corrisponda a un albero orientato, radicato, cardinale per $k=2$, secondo la Definizione 5.6.8. Ma come possiamo “collegare” le due definizioni?

Per rispondere a questa legittima domanda, mostriamo come si può definire usando l’induzione una funzione che associa a ogni albero binario $T \in BT$ un grafo orientato $G_T = (V_T, E_T)$ che lo rappresenta fedelmente. Consideriamo l’albero T mostrato nella Figura 7.1 (b). Il problema principale è determinare l’insieme di nodi V_T . Usiamo la seguente tecnica: etichettiamo tutti gli archi che collegano un nodo alla radice del sottoalbero sinistro con 0, e tutti gli altri con 1, come in Figura 7.3 (a). Si vede facilmente che ogni cammino dalla radice a un nodo è individuato univocamente da una stringa in $\{0, 1\}^*$, e poiché tra ogni coppia di nodi c’è un solo cammino, usiamo questa stringa come “identità” del nodo. Quindi scegliamo l’insieme V_T come un sottoinsieme di $\{0, 1\}^*$, come mostrato nella Figura 7.3 (b).

La seguente definizione per induzione definisce sia l’insieme di nodi $V_T \subseteq \{0, 1\}^*$ che l’insieme di archi $E_T \subseteq V_T \times V_T$ del grafo G_T realizzando la costruzione descritta sopra.

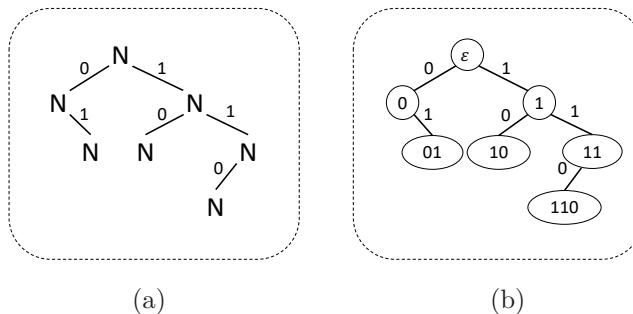


Figura 7.3: Identificazione dell'insieme di nodi di un albero $T \in BT$

- ### 1. [CLAUSOLA BASE] ($T = \lambda$)

$$grafo(\lambda) = (\emptyset, \emptyset)$$

2. [CLAUSOLA INDUTTIVA ($T = N(t_1, t_2)$)] Supponiamo per induzione che $\text{grafo}(t_1) = (V_1, E_1)$ e $\text{grafo}(t_2) = (V_2, E_2)$. Allora definiamo

$$V = \{0w \mid w \in V_1\} \cup \{1w \mid w \in V_2\} \cup \{\varepsilon\}$$

$$E = \{(0w, 0v) \mid (w, v) \in E_1\} \cup \{(1w, 1v) \mid (w, v) \in E_2\} \cup \{(\varepsilon, 0) \mid \varepsilon \in V_1\} \cup \{(\varepsilon, 1) \mid \varepsilon \in V_2\}$$

$$grafo(N(t_1, t_2)) = (V, E)$$

7.4 Induzione Strutturale

Nelle sezioni precedenti abbiamo visto varie strutture (liste, alberi binari, alberi binari etichettati) che condividono con i numeri naturali tre proprietà importanti: (1) le strutture sono definite in maniera induttiva; (2) è possibile definire induttivamente funzioni su tali strutture; (3) è possibile dimostrare proprietà su tali strutture utilizzando il Principio di Induzione.

L'attento lettore si starà domandando se, in qualche modo, è possibile definire queste tre proprietà in maniera del tutto generale, indipendentemente dalla struttura sotto mano, e in modo da poter considerare altre strutture oltre a quelle che abbiamo illustrato sino ad ora (ad esempio gli alberi ternari).

L'*induzione strutturale* fornisce una efficace risposta a tale domanda, basandosi su una struttura del tutto generale: i *termini*. I termini sono definiti in maniera parametrica rispetto ad una *segnatura*.

Definizione 7.4.1 (Segnatura). Una SEGNATURA è una famiglia di insiemi indicizzata da \mathbb{N} , cioè $\mathcal{F} = \{\mathcal{F}_n\}_{n \in \mathbb{N}}$. Gli elementi di tale famiglia sono detti SIMBOLI. Per ogni $n \in \mathbb{N}$, \mathcal{F}_n è l'insieme dei SIMBOLI DI ARIETÀ n (o con n argomenti). I simboli di arietà 0 sono detti SIMBOLI DI COSTANTE.

I simboli in \mathcal{F} sono tipicamente pensati come simboli di funzione. La loro arietà definisce il numero di argomenti che tali funzioni prendono in input. Spesso un simbolo con arietà 1, viene detto *unario*; un simbolo con arietà 2 viene detto *binario*; un simbolo con arietà n , n -ario.

Anche se la definizione è del tutto ovvia, è utile illustrare un semplice esempio.

Esempio 7.4.2. Consideriamo la segnatura \mathcal{F} definita come:

$$\mathcal{F}_0 \equiv \{a, b\} \quad \mathcal{F}_1 \equiv \{f\} \quad \mathcal{F}_2 \equiv \{g\} \quad \mathcal{F}_n \equiv \emptyset \text{ per ogni } n \geq 3.$$

In altre parole, \mathcal{F} consiste di due simboli di costante, cioè di aritetà 0, a e b , di un simbolo unario, cioè di aritità 1, f , e di un simbolo binario, cioè di aritità 2, g .

Una volta fissato il concetto di segnatura, è possibile introdurre l'insieme dei termini per una segnatura.

Definizione 7.4.3 (Termini). *Data una segnatura \mathcal{F} , l'insieme $\mathcal{F}\text{Term}$ degli \mathcal{F} -termini è il più piccolo insieme che soddisfa:*

1. per ogni simbolo $c \in \mathcal{F}_0$, $c \in \mathcal{FTerm}$;
2. per ogni $n \geq 1$ ed ogni simbolo $f \in \mathcal{F}_n$, se $t_1, \dots, t_n \in \mathcal{FTerm}$ allora $f(t_1, \dots, t_n) \in \mathcal{FTerm}$.

Per chiarire il concetto di termine è utile illustrare un esempio.

Esempio 7.4.4. Prendiamo come segnatura la famiglia \mathcal{F} dell'Esempio 7.4.2. I seguenti elementi appartengono a \mathcal{FTerm} , cioè sono \mathcal{F} -termini.

- $f(a)$
- $g(a, a)$
- $g(b, a)$
- $f(g(b, a))$
- $f(f(a))$
- $f(g(f(a), f(b)))$

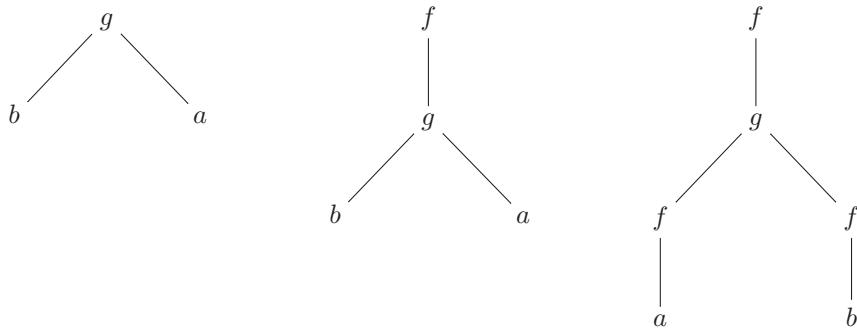
Per essere del tutto formali, mostriamo ad esempio perché $f(f(a)) \in \mathcal{FTerm}$: visto che $a \in \mathcal{F}_0$, dalla prima clausola della Definizione 7.4.3, si ha che $a \in \mathcal{FTerm}$; visto che $f \in \mathcal{F}_1$ e $a \in \mathcal{FTerm}$, dalla seconda clausola si ha che $f(a) \in \mathcal{FTerm}$; infine, visto che $f \in \mathcal{F}_1$ e $f(a) \in \mathcal{FTerm}$, dalla seconda clausola si ha che $f(f(a)) \in \mathcal{FTerm}$.

Utilizzando un ragionamento del tutto analogo, si può mostrare che tutti gli elementi elencati sopra sono \mathcal{F} -termini. Invece, i seguenti elementi non sono \mathcal{F} -termini.

- g
- $g(a)$
- $f(a, b)$
- $g(a, a, b)$

Infatti g è un simbolo di arietà 2 ed ha quindi bisogno di esattamente due termini t_1, t_2 per formare un termine $g(t_1, t_2)$. Pertanto g , $g(a)$ e $g(a, a, b)$ non sono termini. Similmente, f è un simbolo di arietà 1 ed ha quindi bisogno di un solo argomento: $f(a, a)$ non è dunque un termine.

I termini possono essere rappresentati in maniera grafica, come alberi radicati: ogni nodo è etichettato con un simbolo in \mathcal{F} ed ha tanti figli quanti l'arietà di tale simbolo. In particolare le costanti (simboli di arietà 0), sono le foglie (cioè hanno 0 figli). Ad esempio i termini $g(b, a)$, $f(g(b, a))$ e $f(g(f(a), f(b)))$ dell'Esempio 7.4.4 sono rappresentati dai seguenti tre alberi.



Esercizio 7.4.5. Sia \mathcal{F} la segnatura definita come:

$$\mathcal{F}_0 = \{a\} \quad \mathcal{F}_1 = \mathcal{F}_2 = \emptyset \quad \mathcal{F}_3 = \{g\} \quad \mathcal{F}_n = \emptyset \text{ per ogni } n \geq 4.$$

Per ognuno dei seguenti elementi dire se si tratta di un \mathcal{F} -termine e, in tal caso, disegnare l'albero corrispondente.

$$a \quad g(a) \quad g(a, a) \quad g(a, a, a) \quad g(a, a, a, a) \quad g(g(a, a, a)) \quad g(g(a, a, a), a, a)$$

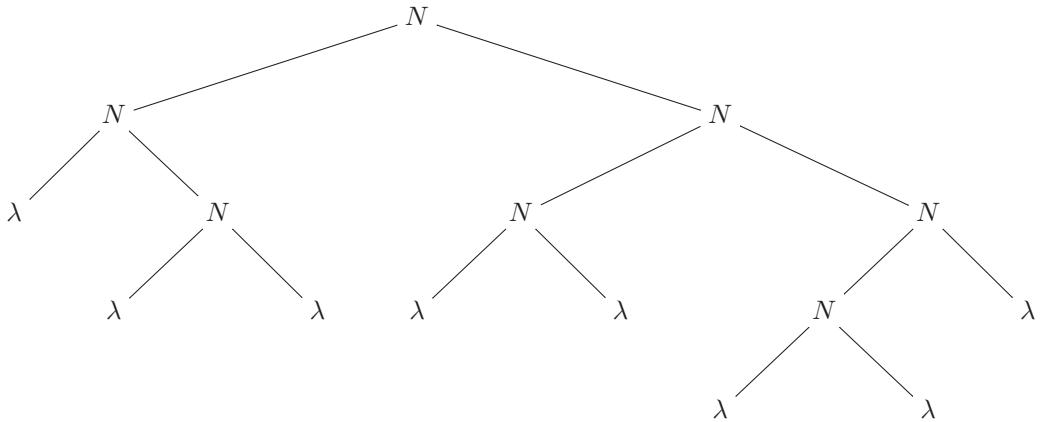


Figura 7.4: Il \mathcal{BT} -termine corrispondente all'albero binario in Figura 7.1.(b).

Esercizio 7.4.6. Sia \mathcal{F} la segnatura definita come:

$$\mathcal{F}_0 = \emptyset \quad \mathcal{F}_1 = \{g\} \quad \mathcal{F}_2 = \emptyset \quad \mathcal{F}_3 = \{f\} \quad \mathcal{F}_n = \emptyset \text{ per ogni } n \geq 4.$$

Quali sono gli \mathcal{F} -termini?

Al variare della segnatura \mathcal{F} , gli \mathcal{F} -termini possono rappresentare tante diverse strutture, molto utili in Informatica, Logica e Matematica. Di seguito mostriamo come numeri naturali, liste ed alberi binari possono essere visti come termini per una qualche segnatura, ma l'importanza dei termini va ben oltre questi casi: ad esempio, incontreremo nuovamente i termini nella Logica dei Predicati nella Sezione 9.5.

Esempio 7.4.7 (Alberi binari come termini). Consideriamo la segnatura \mathcal{BT} definita come

$$\mathcal{BT}_0 = \{\lambda\} \quad \mathcal{BT}_1 = \emptyset \quad \mathcal{BT}_2 = \{N\} \quad \mathcal{BT}_n = \emptyset \text{ per ogni } n \geq 3.$$

In altre parole, \mathcal{BT} consiste solamente di una costante, cioè simbolo di arietà 0, λ , e di un operatore binario, cioè di arietà 2, N .

I \mathcal{BT} -termini sono esattamente gli alberi binari come definiti nella Sezione 7.2. Infatti, fissando $\mathcal{F} = \mathcal{BT}$ nella Definizione 7.4.3, si ottiene esattamente la Definizione 7.2.1: dalla prima clausola della Definizione 7.4.3 si ottiene

1. $\lambda \in \mathcal{BT}\text{Term}$

visto che λ è il solo simbolo in \mathcal{BT}_0 . Dalla seconda clausola, si ottiene

2. se $t_1, t_2 \in \mathcal{BT}\text{Term}$, allora $N(t_1, t_2) \in \mathcal{BT}\text{Term}$

visto che N ha arietà 2 e, in \mathcal{BT} , N è il solo simbolo con arietà ≥ 1 .

In Figura 7.4 è rappresentato il \mathcal{BT} -termine

$$N(N(\lambda, N(\lambda, \lambda)), N(N(\lambda, \lambda), N(N(\lambda, \lambda), \lambda)))$$

che avevamo precedentemente rappresentato come albero binario nella Figura 7.1.(b). È importante notare che le due rappresentazioni sono molto simili: nella Figura 7.1, gli alberi vuoti (corrispondenti a λ) non vengono disegnati, mentre in Figura 7.4 questi vengono rappresentati da un nodo foglia etichettato con λ . Per ottenere l'albero in Figura 7.1.(b) da quello in Figura 7.4 è quindi sufficiente rimuovere tutti i nodi etichettati con λ e gli archi ad essi collegati.

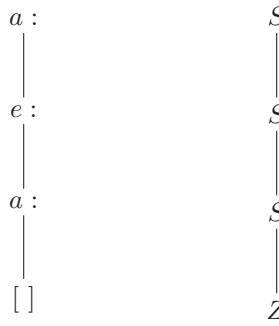


Figura 7.5: Rappresentazione grafica per il \mathcal{L}^A -termine $a : (e : (a : ([])))$, sinistra, e per il \mathcal{N} -termine $S(S(S(Z)))$, destra.

Esempio 7.4.8 (Liste come termini). *Dato un insieme qualsiasi A , definiamo la segnatura \mathcal{L}^A come*

$$\mathcal{L}_0^A = \{[\]\} \quad \mathcal{L}_1^A = \{a : \mid a \in A\} \quad \mathcal{L}_n^A = \emptyset \text{ per ogni } n \geq 2.$$

In altre parole, questa segnatura consiste solamente di una costante, $[]$, e di un operatore unario “ $a :$ ” per ogni $a \in A$.

I \mathcal{L}^A -termini sono esattamente liste di elementi nell’insieme A . Infatti, fissando $\mathcal{F} = \mathcal{L}^A$ nella Definizione 7.4.3, si ottiene esattamente la definizione induttiva di lista data nella Sezione 7.1: dalla prima clausola della Definizione 7.4.3 si ottiene

1. $[\] \in \mathcal{L}^A \text{Term}$

visto che $[\]$ è il solo simbolo in \mathcal{L}_0^A . Inoltre, visto che $\mathcal{L}_1^A = \{a : \mid a \in A\}$ e che, per ogni $n \geq 2$ $\mathcal{L}_n^A = \emptyset$, la seconda clausola della Definizione 7.4.3 si riduce a

2. *per ogni $a \in A$, se $t \in \mathcal{L}^A \text{Term}$, allora $a : (t) \in \mathcal{L}^A \text{Term}$.*

Confrontando queste due clausole con quelle dell’insieme L_A introdotto dalla Definizione 7.1.1 è facile convincersi che L_A e $\mathcal{L}^A \text{Term}$ sono essenzialmente lo stesso insieme (o, esprimendosi in maniera più formale, sono in biiezione). Infatti la sola differenza risiede nell’uso delle parentesi tonde nella seconda clausola di sopra: basta rimpiazzare “ $a : (t)$ ” con “ $a : t$ ” per ottenere esattamente la seconda clausola della Definizione 7.1.1.

Nella parte sinistra della Figura 7.5 è illustrata la rappresentazione grafica del \mathcal{L}^A -termine corrispondente alla lista $[a, e, a]$.

Esempio 7.4.9 (I naturali come termini). *La segnatura \mathcal{N} è definita come*

$$\mathcal{N}_0 = \{Z\} \quad \mathcal{N}_1 = \{S\} \quad \mathcal{N}_n = \emptyset \text{ per ogni } n \geq 2.$$

In altre parole, la segnatura \mathcal{N} consiste solamente di una costante, Z , e di un operatore unario, S .

Fissando $\mathcal{F} = \mathcal{N}$ nella Definizione 7.4.3 si ottiene

1. $Z \in \mathcal{N} \text{Term}$

2. *se $t \in \mathcal{N} \text{Term}$, allora $S(t) \in \mathcal{N} \text{Term}$*

Come mostreremo formalmente nella Proposizione 7.4.17, gli \mathcal{N} -termini sono in biiezione con i numeri naturali: intuitivamente, la costante Z rappresenta il naturale 0 ed il simbolo unario S rappresenta il successore di un numero naturale.

Dalla definizione segue che ogni elemento di $\mathcal{F} \text{Term}$ è della forma

$$\underbrace{S(S(\cdots(S(Z))\cdots))}_{n \in \mathbb{N}}$$

cioè Z preceduta da un numero n finito di S . Intuitivamente, tale termine rappresenta il numero naturale n , cioè l' n -simo successore di 0.

$$((\cdots (\underbrace{0+1}_{n \in \mathbb{N}} \cdots) + 1) + 1)$$

Nella destra della Figura 7.5 è illustrata la rappresentazione grafica per l' \mathcal{N} -termine $S(S(S(Z)))$, corrispondente al numero naturale 3.

Esercizio 7.4.10. Definire una segnatura \mathcal{F} tale che gli \mathcal{F} -termini corrispondano agli alberi binari etichettati (Definizione 7.3.1).

Esercizio 7.4.11. Definire una segnatura \mathcal{F} tale che gli \mathcal{F} -termini corrispondano agli alberi ternari.

Funzioni su Termini

Nelle sezioni precedenti abbiamo visto che è possibile definire delle funzioni su strutture definite induttivamente (come naturali, liste, alberi binari e alberi binari etichettati). Questo fatto può essere espresso in modo più generale utilizzando i termini. Infatti, è possibile definire una funzione su \mathcal{FTerm} , cioè sull'insieme dei termini per una qualche segnatura \mathcal{F} , come segue:

- (1) definire il valore della funzione per i simboli di arietà 0;
- (2) definire il valore della funzione per ogni simbolo di arietà $n \geq 1$, eventualmente utilizzando il valore della funzione calcolato su ognuno degli n -argomenti.

Illustriamo adesso due esempi di funzioni così definite, prendendo come insieme di partenza \mathcal{NTerm} .

Definizione 7.4.12 (Valutazione di \mathcal{N} -termini). *Sia $val : \mathcal{NTerm} \rightarrow \mathbb{N}$ la funzione definita induttivamente come*

1. $val(Z) = 0$
2. $val(S(x)) = val(x) + 1$

Tale funzione intuitivamente associa ad ogni \mathcal{N} -termine il numero naturale corrispondente. Ad esempio $val(S(S(S(Z)))) = 3$, come mostrato dalla seguente derivazione:

$$\begin{aligned} val(S(S(S(Z)))) &= val(S(S(Z))) + 1 && \text{(Clausola induttiva)} \\ &= val(S(Z)) + 1 + 1 && \text{(Clausola induttiva)} \\ &= val(Z) + 1 + 1 + 1 && \text{(Clausola induttiva)} \\ &= 0 + 1 + 1 + 1 && \text{(Clausola base)} \\ &= 3 \end{aligned}$$

Ricordiamo che nella Sezione 3.1 abbiamo assunto per conosciute le operazioni sui naturali. Mostriamo ora come possiamo definire sugli \mathcal{N} -termini l'operazione di addizione, in modo induttivo.

Definizione 7.4.13 (Somma di \mathcal{N} -termini). *La funzione $add : \mathcal{NTerm} \times \mathcal{NTerm} \rightarrow \mathcal{NTerm}$ è definita per induzione sul secondo argomento nel seguente modo:*

1. $add(x, Z) = x$
2. $add(x, S(y)) = S(add(x, y))$

La funzione add prende due numeri naturali in input, rappresentati come \mathcal{N} -termini, e restituisce la loro somma. Ad esempio $add(S(S(S(Z))), S(S(Z)))$ restituisce la somma di 3 e 2, cioè $S(S(S(S(S(Z)))))$ corrispondente al numero naturale 5. Infatti, abbiamo che

$$\begin{aligned} add(S(S(S(Z))), S(S(Z))) &= S(add(S(S(S(Z))), S(Z))) && \text{(Clausola induttiva)} \\ &= S(S(add(S(S(S(Z))), Z))) && \text{(Clausola induttiva)} \\ &= S(S(S(S(S(Z))))) && \text{(Clausola base)} \end{aligned}$$

Esercizio 7.4.14. Sulla falsariga della Definizione 7.4.13 e sfruttando la funzione add , definire per induzione la funzioni $\text{mul} : \mathcal{N}\text{Term} \times \mathcal{N}\text{Term} \rightarrow \mathcal{N}\text{Term}$ che calcola la moltiplicazione di due numeri naturali rappresentati come \mathcal{N} -termini.

Esercizio 7.4.15. Sulla falsariga della Definizione 7.4.13 e sfruttando la funzione mul dell'esercizio precedente, definire per induzione la funzioni $\text{exp} : \mathcal{N}\text{Term} \times \mathcal{N}\text{Term} \rightarrow \mathcal{N}\text{Term}$ che prende in input due numeri naturali rappresentati come \mathcal{N} -termini, chiamiamoli x e y , e restituisce x^y .

Il Principio di Induzione Strutturale

Abbiamo visto precedentemente che al variare della segnatura \mathcal{F} , gli \mathcal{F} -termini possono rappresentare strutture diverse, come numeri naturali, liste o alberi. Lo stesso vale per il Principio di Induzione: è possibile formulare tale principio a livello degli \mathcal{F} -termini e i principi per naturali, liste e alberi possono essere ricavati come casi speciali di questo, opportunamente fissando la segnatura \mathcal{F} . Il Principio di Induzione sui termini viene comunemente chiamato Principio di Induzione Strutturale.

Il PRINCIPIO DI INDUZIONE STRUTTURALE stabilisce che:

1. Se (caso base), per ogni simbolo $c \in \mathcal{F}_0$, $P(c)$ è vera, e
2. se (passo induttivo), per ogni $n \geq 1$, per ogni simbolo $f \in \mathcal{F}_n$, per tutti i termini $t_1, \dots, t_n \in \mathcal{F}\text{Term}$, vale che se $P(t_1), \dots, P(t_n)$ sono vere allora anche $P(f(t_1, \dots, t_n))$ è vera,

allora $P(t)$ è vera per ogni $t \in \mathcal{F}\text{Term}$.

In modo più compatto possiamo scrivere il Principio di Induzione Strutturale come una *regola di inferenza*:

$$\frac{\forall c \in \mathcal{F}_0 . P(c) \quad \forall n \geq 1 . \forall f \in \mathcal{F}_n . \forall t_1, \dots, t_n \in \mathcal{F}\text{Term} . P(t_1) \wedge \dots \wedge P(t_n) \Rightarrow P(f(t_1, \dots, t_n))}{\forall t \in \mathcal{F}\text{Term} . P(t)} \quad \text{P.I. Strutturale}$$

È interessante osservare come da questo principio si possono ricavare, come casi speciali, quello sulle liste (Sezione 7.1) e quello sugli alberi binari (Sezione 7.2)

- Fissiamo \mathcal{F} essere la segnatura \mathcal{L}^A dell'Esempio 7.4.8. In \mathcal{L}^A c'è un solo simbolo di arietà 0: la lista vuota $[]$. Pertanto il caso base del Principio di Induzione Strutturale si riduce a

$$P([]) \text{ è vera.}$$

Per quanto riguarda il passo induttivo, osserviamo che \mathcal{L}^A contiene solamente simboli di arietà 1 e che tutti sono della forma $a :$ per $a \in A$. Pertanto, il passo induttivo si riduce a

$$\begin{aligned} &\text{per ogni } a \in A, \text{ per ogni } t_1 \in \mathcal{L}^A\text{Term} \text{ vale che} \\ &\quad \text{se } P(t_1) \text{ è vera allora anche } P(a : t_1) \text{ è vera.} \end{aligned}$$

- Fissiamo \mathcal{F} essere la segnatura \mathcal{BT} dell'Esempio 7.4.7. In \mathcal{BT} c'è un solo simbolo di arietà 0: l'albero vuoto λ . Pertanto il caso base del Principio di Induzione Strutturale si riduce a

$$P(\lambda) \text{ è vera.}$$

Per quanto riguarda il passo induttivo, osserviamo che oltre a λ , \mathcal{BT} contiene solamente un simbolo di arietà 2: N . Pertanto, il passo induttivo si riduce a

per ogni $t_1, t_2 \in \mathcal{BT}$ vale che se $P(t_1)$ è vera e $P(t_2)$ è vera allora anche $P(N(t_1, t_2))$ è vera.

Mostriamo adesso due esempi di dimostrazione che utilizzano il Principio di Induzione Strutturale per $\mathcal{F} = \mathcal{N}$ (Esempio 7.4.9). Si noti che il caso base (c.b.) ed il passo induttivo (p.i.) si riducono a

(c.b.) $P(Z)$ è vera (p.i.) per ogni $t \in \mathcal{N}\text{Term}$ vale che se $P(t)$ è vera, allora anche $P(S(t))$ è vera

La prima proposizione che andiamo a dimostrare asserisce che la funzione $\text{add}: \mathcal{N}\text{Term} \times \mathcal{N}\text{Term} \rightarrow \mathcal{N}\text{Term}$ (Definizione 7.4.13) calcola effettivamente la somma di naturali, o più precisamente, di \mathcal{N} -termini che rappresentano naturali attraverso la funzione $\text{val}: \mathcal{N}\text{Term} \rightarrow \mathbb{N}$ (Definizione 7.4.12).

Proposizione 7.4.16. *Per ogni coppia di elementi $x, y \in \mathcal{N}\text{Term}$ vale che $\text{val}(\text{add}(x, y)) = \text{val}(x) + \text{val}(y)$.*

Dimostrazione. Procediamo per induzione sul secondo argomento.

1. [CASO BASE] ($y = Z$).

$$\begin{aligned} \text{val}(\text{add}(x, Z)) &= \text{val}(x) && (\text{Per } \text{add, clausola base}) \\ &= \text{val}(x) + 0 \\ &= \text{val}(x) + \text{val}(Z) && (\text{Per } \text{val, clausola base}) \end{aligned}$$

2. [PASSO INDUTTIVO] ($y = S(z)$). Assumiamo come ipotesi induttiva che $\text{val}(\text{add}(x, z)) = \text{val}(x) + \text{val}(z)$ per dimostrare che $\text{val}(\text{add}(x, S(z))) = \text{val}(x) + \text{val}(S(z))$.

$$\begin{aligned} \text{val}(\text{add}(x, S(z))) &= \text{val}(S(\text{add}(x, z))) && (\text{Per } \text{add, clausola induttiva}) \\ &= \text{val}(\text{add}(x, z)) + 1 && (\text{Per } \text{val, clausola induttiva}) \\ &= \text{val}(x) + \text{val}(z) + 1 && (\text{Per ipotesi induttiva}) \\ &= \text{val}(x) + \text{val}(S(z)) && (\text{Per } \text{val, clausola induttiva}) \end{aligned}$$

■

La seconda proposizione enuncia formalmente quanto annunciato nell’Esempio 7.4.9, cioè che l’insieme dei numeri naturali e quello degli \mathcal{N} -termini sono in biiezione.

Proposizione 7.4.17. *$\text{val}: \mathcal{N}\text{Term} \rightarrow \mathbb{N}$ è una biiezione e pertanto $\mathcal{N}\text{Term} \cong \mathbb{N}$.*

Dimostrazione. Mostriamo che $\text{val}: \mathcal{N}\text{Term} \rightarrow \mathbb{N}$ è una biiezione utilizzando la caratterizzazione attraverso funzioni invertibili (Teorema 2.6.16). In altre parole, mostriamo una funzione $j: \mathbb{N} \rightarrow \mathcal{N}\text{Term}$ tale che

$$\text{val}; j = id_{\mathcal{N}\text{Term}} \quad \text{e} \quad j; \text{val} = id_{\mathbb{N}}$$

cioè, tale che

$$(a) \quad j(\text{val}(t)) = t \quad \text{per ogni } t \in \mathcal{N}\text{Term}, \quad \text{e} \quad (b) \quad \text{val}(j(n)) = n \quad \text{per ogni } n \in \mathbb{N}.$$

Definiamo tale j induttivamente, sfruttando la definizione induttiva di \mathbb{N} (Definizione 3.1.1): la funzione $j: \mathbb{N} \rightarrow \mathcal{N}\text{Term}$ è definita induttivamente come

1. $j(0) = Z$
2. $j(n + 1) = S(j(n))$

Dimostriamo adesso che l’equazione (a) illustrata sopra vale, utilizzando l’induzione su $\mathcal{N}\text{Term}$:

1. [CASO BASE] ($t = Z$).

$$\begin{aligned} j(\text{val}(Z)) &= j(0) && (\text{Clausola base val}) \\ &= Z && (\text{Clausola base di } j) \end{aligned}$$

2. [PASSO INDUTTIVO] ($t = S(t')$). Assumiamo come ipotesi induttiva che $j(\text{val}(t')) = t'$, per dimostrare che $j(\text{val}(S(t'))) = S(t')$:

$$\begin{aligned} j(\text{val}(S(t'))) &= j(\text{val}(t') + 1) && (\text{Clausola induttiva val}) \\ &= S(j(\text{val}(t'))) && (\text{Clausola induttiva } j) \\ &= S(t') && (\text{Ipotesi Induttiva}) \end{aligned}$$

La dimostrazione dell'equazione (b) è del tutto analoga e lasciata al lettore come esercizio. ■

Esercizio 7.4.18. Dimostrare l'equazione (b) nella dimostrazione della Proposizione 7.4.17.

Esercizio 7.4.19. Si ricordi la funzione $\text{val}: \mathcal{N}\text{Term} \rightarrow \mathbb{N}$ della Definizione 7.4.12 e la funzione $\text{mul}: \mathcal{N}\text{Term} \times \mathcal{N}\text{Term} \rightarrow \mathcal{N}\text{Term}$ dell'Esercizio 7.4.14. Dimostrare che $\text{val}(\text{mul}(x, y)) = \text{val}(x) \cdot \text{val}(y)$.

Esercizio 7.4.20. Si ricordi la funzione $\text{val}: \mathcal{N}\text{Term} \rightarrow \mathbb{N}$ della Definizione 7.4.12 e la funzione $\text{exp}: \mathcal{N}\text{Term} \times \mathcal{N}\text{Term} \rightarrow \mathcal{N}\text{Term}$ dell'Esercizio 7.4.15. Dimostrare che $\text{val}(\text{exp}(x, y)) = \text{val}(x)^{\text{val}(y)}$.

Fino a questo momento abbiamo utilizzato i principi di induzione su varie strutture (naturali, liste, alberi) senza dimostrarne la correttezza. Dimostriamo adesso la correttezza del Principio di Induzione Strutturale e, da questa, seguono immediatamente i risultati di correttezza di tutti i principi visti sino ad ora.

Teorema 7.4.21. Il Principio di Induzione Strutturale è corretto.

Dimostrazione. Siano \mathcal{F} una arbitraria segnatura e P una arbitraria proprietà sugli \mathcal{F} -termini (cioè, per la Definizione 2.5.16 una funzione del tipo $P: \mathcal{F}\text{Term} \rightarrow \text{Bool}$). Assumiamo che P soddisfi le condizioni 1. e 2. del Principio di Induzione Strutturale e dimostriamo che $P(t) = \text{t}$ per ogni $t \in \mathcal{F}\text{Term}$.

Sia A l'insieme di tutti gli \mathcal{F} -termini che soddisfano P , cioè

$$A = \{t \in \mathcal{F}\text{Term} \mid P(t) = \text{t}\}.$$

Dalla definizione è immediato che $A \subseteq \mathcal{F}\text{Term}$. Per dimostrare il teorema, mostriamo che vale anche l'altra inclusione, cioè che $\mathcal{F}\text{Term} \subseteq A$. Infatti tale inclusione significa che per tutti i termini $t \in \mathcal{F}\text{Term}$ vale che $P(t) = \text{t}$.

1. Visto che abbiamo assunto che P soddisfa il caso base del principio di Induzione Strutturale, abbiamo che per ogni simbolo $c \in \mathcal{F}_0$, $P(c)$ è vera, cioè $c \in A$.
2. Visto che abbiamo assunto che P soddisfa il passo induttivo del principio di Induzione Strutturale, abbiamo che per ogni $n \geq 1$, per ogni simbolo $f \in \mathcal{F}_n$, per tutti i termini $t_1, \dots, t_n \in \mathcal{F}\text{Term}$, vale che se $P(t_1), \dots, P(t_n)$ sono vere allora anche $P(f(t_1, \dots, t_n))$ è vera. In altre parole, per ogni $n \geq 1$, per ogni simbolo $f \in \mathcal{F}_n$, se $t_1, \dots, t_n \in A$ allora anche $f(t_1, \dots, t_n) \in A$.

Pertanto A è un insieme che soddisfa le condizioni 1. e 2. della Definizione 7.4.3. Ma la Definizione 7.4.3, impone che fra tutti gli insiemi che soddisfano le due condizioni, $\mathcal{F}\text{Term}$ sia il più piccolo, cioè che per qualsiasi insieme B che soddisfa le due condizioni $\mathcal{F}\text{Term} \subseteq B$. In particolare si ha che $\mathcal{F}\text{Term} \subseteq A$. ■

Corollario 7.4.22. Il Principio di Induzione sui naturali è corretto.

Corollario 7.4.23. Il Principio di Induzione sulle liste è corretto.

Corollario 7.4.24. Il Principio di Induzione sugli alberi binari è corretto.

Esercizio 7.4.25. Si dimostri che il Principio di Induzione sugli alberi binari etichettati è corretto.

7.5 Funzioni ricorsive

In ognuna delle funzioni introdotte nelle sezioni precedenti il valore della funzione per un certo argomento era definito o direttamente (nella clausola base), oppure in termini del valore della stessa funzione su argomenti “più piccoli”. In particolare per le funzioni definite su \mathbb{N} il valore della funzione su 0 era definito direttamente perché non c’è nessun numero naturale $k < 0$.

Le funzioni definite induttivamente sono un caso particolare di FUNZIONI RICORSIVE. Una funzione è detta *ricorsiva* se il valore della funzione per un certo argomento è espresso in termini del valore della stessa funzione applicata a uno o più argomenti, non necessariamente più piccoli.

Per esempio, consideriamo la seguente funzione *due*:

$$\text{due}(n) = \begin{cases} 2 & \text{se } n > 100 \\ \text{due}(n+1) & \text{altrimenti.} \end{cases} \quad (7.1)$$

Si tratta di una definizione ricorsiva, ma non induttiva, perché nella seconda clausola il valore per n è espresso in termini del valore di *due* per un argomento maggiore, $n+1$. Per ottenere il valore della funzione per un certo argomento usiamo la solita tecnica: applichiamo la clausola che corrisponde all'argomento e otteniamo un valore finale, oppure continuiamo con la valutazione dell'espressione ottenuta. Valutiamo la funzione *due* per un paio di argomenti: 98 e 110.

$$\text{due}(110) = 2 \quad (\text{Clausola base})$$

$$\begin{aligned} \text{due}(98) &= \text{due}(99) && (\text{Clausola induttiva}) \\ &= \text{due}(100) && (\text{Clausola induttiva}) \\ &= \text{due}(101) && (\text{Clausola induttiva}) \\ &= 2 && (\text{Clausola base}) \end{aligned}$$

Si verifica facilmente che valutando $\text{due}(k)$ per un qualunque $k \in \mathbb{N}$ il risultato finale è 2. Quindi la (7.1) definisce correttamente una funzione $\text{due} : \mathbb{N} \rightarrow \mathbb{N}$.

Vediamo un altro esempio di funzione ricorsiva ma non induttiva.

$$f(n) = \begin{cases} 1 & \text{se } n \leq 1 \\ f(n/2) & \text{se } n > 1 \text{ ed è pari} \\ f(3 \cdot n + 1) & \text{se } n > 1 \text{ ed è dispari} \end{cases} \quad (7.2)$$

Valutiamo questa funzione per i primi numeri naturali, da 0 a 8:

$$\begin{aligned} f(0) &= 1 \\ f(1) &= 1 \\ f(2) &= f(1) = 1 \\ f(3) &= f(10) = f(5) = f(16) = f(8) = f(4) = f(2) = f(1) = 1 \\ f(4) &= f(2) = f(1) = 1 \\ f(5) &= f(16) = f(8) = f(4) = f(2) = f(1) = 1 \\ f(6) &= f(3) = f(10) = f(5) = f(16) = f(8) = f(4) = f(2) = f(1) = 1 \\ f(7) &= f(22) = f(11) = f(34) = f(17) = f(52) = f(26) = f(13) = f(40) = f(20) \\ &= f(10) = f(5) = f(16) = f(8) = f(4) = f(2) = f(1) = 1 \\ f(8) &= f(4) = f(2) = f(1) = 1 \end{aligned}$$

È facile convincersi che valutando la funzione per un certo $k \in \mathbb{N}$, se la valutazione termina allora il valore finale deve essere 1. Ma il numero di passi necessario per arrivare al risultato non sembra seguire una regola precisa (si veda $f(7)$). Viene naturale chiedersi: ma è sicuro che la valutazione di $f(k)$ termina per ogni k ? Torneremo a discutere questa funzione più avanti nell'Esempio 7.5.11, quando parleremo della *Congettura di Collatz*. Per il momento cerchiamo di affrontare questo problema, cioè se la valutazione di una funzione ricorsiva termina, da una prospettiva più ampia.

Nelle definizioni induttive di funzioni delle sezioni precedenti, la clausola base e la clausola induttiva corrispondevano direttamente alle clausole della definizione induttiva dell'insieme su cui la funzione era definita. Questa tecnica garantisce che la funzione sia ben definita. Anche la funzione *due* vista sopra è ben definita, ma è facile convincersi che una descrizione ricorsiva e non

induttiva potrebbe non essere accettabile, nel senso che potrebbe non definire in modo univoco una funzione, oppure potrebbe non esserci alcuna funzione che rispetti tale descrizione. Esploriamo queste problematiche con alcuni esempi di definizioni ricorsive.

$$g(n) = \begin{cases} 0 & \text{se } n = 0 \\ g(n+1) & \text{altrimenti.} \end{cases} \quad (7.3)$$

Questa definizione ha una clausola base, ma la seconda definisce g per gli altri valori in \mathbb{N} in termini del suo valore per un argomento più grande. Un possibile modo di interpretare una definizione ricorsiva come questa è quello *dichiarativo*, che consiste nel caratterizzare un insieme di funzioni, cioè tutte quelle che soddisfano entrambe le clausole. Da questo punto di vista è facile convincersi che ogni funzione che valga 0 su 0 e che sia costante per tutti gli altri naturali soddisfarebbe la (7.3).

Ma questo non è il modo in cui noi informatici vediamo le definizioni ricorsive: noi le interpretiamo in modo *operazionale* o *computazionale*, pensando a come un programma in un tipico linguaggio di programmazione valuterebbe g su un dato argomento. Quindi possiamo pensare che, fissati gli insiemi di partenza e di arrivo (supponiamo che siano entrambi \mathbb{N} , per essere concreti), una definizione ricorsiva di, per esempio, rec , caratterizzi in realtà una relazione R_{rec} su \mathbb{N} così definita:

$$R_{rec} = \left\{ (n, m) \mid \begin{array}{l} \text{partendo da } rec(n) \text{ e applicando iterativamente (e correttamente)} \\ \text{le clausole della definizione, la valutazione termina con il risultato di } m \end{array} \right\}$$

Poiché le funzioni sono relazioni totali e univalenti, diremo che una definizione ricorsiva per rec è *ben data* se R_{rec} è totale e univalente. Nel caso della (7.3), fissando l'insieme di partenza come \mathbb{N} abbiamo per esempio che per valutare $g(1)$ applicheremmo la seconda clausola, valutando poi $g(2)$, e poi $g(3)$, e così via senza mai terminare. Quindi la relazione R_g non è totale perché non contiene alcuna coppia del tipo $(1, _)$. Di conseguenza, per l'interpretazione operazionale che noi adottiamo la (7.3) non è ben data: essa non definisce correttamente una funzione, ma solo una funzione *parziale* (Definizione 2.5.25), indefinita su tutti i naturali tranne che su 0.

$$f(n) = 1 + f(n-1) \quad (7.4)$$

Possiamo considerare questa come una definizione induttiva in cui manca la clausola base. Se consideriamo \mathbb{N} come insieme di partenza di f , vediamo subito che la definizione non è ben data, perché $f(0) = f(-1)$ e $-1 \notin \mathbb{N}$. Se invece consideriamo \mathbb{Z} come insieme di partenza e proviamo a valutarla per un qualunque argomento $k \in \mathbb{Z}$ abbiamo

$$f(k) = 1 + f(k-1) = 2 + f(k-2) = 3 + f(k-3) = \dots$$

Quindi la valutazione non termina mai, mostrando che la (7.4) non definisce correttamente una funzione.

$$h(n) = \begin{cases} h(n-2) & \text{se } n \geq 2 \\ h(n+2) & \text{se } n = 1 \\ 0 & \text{se } n = 0 \end{cases} \quad (7.5)$$

Questa definizione è per certi aspetti simile alla (7.3): dichiarativamente, ogni funzione sui naturali che valga 0 sui numeri pari e una costante arbitraria k sui numeri dispari soddisfarebbe le tre clausole. Ma operazionalmente se si valuta h su un numero dispari $k \geq 3$ dopo un certo numero di passi si arriva a valutare $h(3)$ e quindi $h(1), h(3), h(1), \dots$ Quindi la (7.5) non definisce correttamente una funzione.

Viene quindi naturale chiedersi: come si può verificare se la definizione ricorsiva di una funzione è ben data? Un importante risultato della Teoria della Calcolabilità, il Teorema di Rice, ci dice che non esiste un procedimento effettivo che, data una qualunque definizione ricorsiva, ci consenta di concludere se essa caratterizza una funzione totale oppure no. Però si possono identificare delle condizioni sufficienti, nel senso che se una definizione ricorsiva le soddisfa allora è ben data.

Per esempio, tutte le definizioni induttive viste nelle sezioni precedenti sono ben date. Inoltre, per quanto riguarda l'*univalenza*, è ovvio che se per ogni valore dell'insieme di partenza solo una delle clausole della definizione è applicabile, allora la relazione indotta dalla definizione ricorsiva è univalente: questa è una condizione sufficiente facilmente verificabile con una semplice analisi della definizione.⁵

Approfondiamo invece il problema di verificare se la relazione indotta da una definizione ricorsiva è *totale*. Supponiamo di avere una definizione ricorsiva di rec su un insieme di partenza A e assumiamo che per ogni elemento $a \in A$ ci sia almeno una clausola applicabile (altrimenti possiamo già concludere che R_{rec} non è totale). In questa situazione se R_{rec} non è totale (cioè non c'è nessuna coppia del tipo $(a, _)$ in R_{rec} per un certo $a \in A$) allora vuol dire che valutando $rec(a)$ incorriamo in una computazione infinita. Vediamo come formalizzare delle condizioni che garantiscono che questo non possa mai succedere, e che quindi la definizione di rec sia ben data.

Definizione 7.5.1 (Relazione di precedenza indotta da una definizione ricorsiva). *Data una definizione ricorsiva di $rec : A \rightarrow B$, la relazione di precedenza indotta è la relazione $\prec_{rec} \in Rel(A, A)$ definita come segue:*

per ogni $x, y \in A$, $x \prec_{rec} y$ se e solo se $rec(y)$ è definita direttamente in termini di $rec(x)$

Si noti che la relazione \prec_{rec} così definita non è necessariamente aciclica, anche se il simbolo utilizzato potrebbe suggerirlo.

Esempio 7.5.2. *Per le definizioni ricorsive presentate sopra, le relazioni di precedenza da esse indotte sono le seguenti:*

- (7.1) $\prec_{due} \in Rel(\mathbb{N}, \mathbb{N})$, definita come $n + 1 \prec_{due} n$ se $n \leq 100$
- (7.3) $\prec_g \in Rel(\mathbb{N}, \mathbb{N})$, definita come $n + 1 \prec_g n$ se $n > 0$
- (7.4) $\prec_f \in Rel(\mathbb{Z}, \mathbb{Z})$, definita come $n - 1 \prec_f n$ se $n \in \mathbb{Z}$
- (7.5) $\prec_h \in Rel(\mathbb{N}, \mathbb{N})$, definita come $\prec_h = \{(n - 2, n) \mid n \geq 2\} \cup \{(3, 1)\}$

Quindi se $b \prec_{rec} a$, per valutare $rec(a)$ devo necessariamente valutare $rec(b)$. Ma allora se c'è una sequenza infinita di elementi di A tali che⁶

$$a_1 \succ_{rec} a_2 \succ_{rec} a_3 \succ_{rec} \dots$$

allora la valutazione di $rec(a_1)$ non termina mai. Questo vale per esempio per le relazioni \prec_g e \prec_f dell'Esempio 7.5.2, per le quali abbiamo, per esempio

$$1 \succ_f 0 \succ_f -1 \succ_f -2 \dots$$

$$3 \succ_g 4 \succ_g 5 \succ_g 6 \dots$$

ma anche per \prec_h , per la quale abbiamo

$$7 \succ_h 5 \succ_h 3 \succ_h 1 \succ_h 3 \succ_h 1 \dots$$

Per concludere formalmente questa discussione con una condizione che garantisce che una definizione ricorsiva sia ben data, introduciamo la seguente definizione.

Definizione 7.5.3 (Relazione ben fondata). *Una relazione \sqsubset su un insieme A , $\sqsubset \in Rel(A, A)$, è BEN FONDATA se non esiste una catena infinita decrescente*

$$a_1 \sqsubset a_2 \sqsubset a_3 \sqsubset \dots$$

di elementi $a_i \in A$.

⁵Esercizio: perché questa condizione non è necessaria, ma solo sufficiente?

⁶Abbastanza ovviamente, scriviamo \succ per \prec^{op} .

Dalle considerazioni precedenti segue il seguente risultato, che fornisce una condizione sufficiente affinché la definizione ricorsiva di una funzione sia ben data.

Proposizione 7.5.4. *Data una definizione ricorsiva di $rec : A \rightarrow B$, essa è ben data (e quindi R_{rec} è univalente e totale) se (1) per ogni elemento $a \in A$ c'è esattamente una clausola della definizione applicabile per valutare $rec(a)$ e (2) la relazione \prec_{rec} è ben fondata.*

Esempio 7.5.5. *La (7.1) induce una relazione ben fondata. Essa ha una sola catena discendente che risulta essere finita: $0 \succ_{due} 1 \succ_{due} \dots \succ_{due} 99 \succ_{due} 100 \succ_{due} 101$. Inoltre, tutti gli altri elementi elementi di \mathbb{N} non sono in relazione \prec_{due} tra loro.*

Abbiamo visto che le definizioni induttive sono un caso particolare di ricorsione, e sono sempre ben date. Vediamo perché, utilizzando il concetto di relazione ben fondata.

Definizione 7.5.6 (Relazione di precedenza indotta da definizione induttiva di un insieme). *Data una definizione induttiva di un insieme A , la relazione di precedenza indotta è la relazione $\prec_A \in Rel(A, A)$ definita come segue:*

se per la clausola induttiva l'elemento a appartiene all'insieme perché ci appartengono a_1, \dots, a_k , allora $a_1 \prec_A a, a_2 \prec_A a, \dots, a_k \prec_A a$.

Osservazione 7.5.7. *La definizione precedente può essere applicata a qualsiasi struttura definita induttivamente. Tale livello di generalità può essere mantenuto, utilizzando la nozione di \mathcal{F} -termine. La relazione $\prec_{\mathcal{FTerm}} \in Rel(\mathcal{FTerm}, \mathcal{FTerm})$ è definita come segue:*

per ogni $n \geq 1$, per ogni $f \in \mathcal{F}_n$,

$t_1 \prec_{\mathcal{FTerm}} f(t_1, \dots, t_n), \quad t_2 \prec_{\mathcal{FTerm}} f(t_1, \dots, t_n), \quad \dots \quad , t_n \prec_{\mathcal{FTerm}} f(t_1, \dots, t_n).$

Proposizione 7.5.8 (Relazione indotta da definizione induttiva è ben fondata). *La relazione $\prec_A \in Rel(A, A)$ della Definizione 7.5.6 è ben fondata.*

Infatti poiché l'insieme caratterizzato è il più piccolo che soddisfa le clausole base e induttiva, abbiamo che ogni elemento dell'insieme ci appartiene per una sequenza finita di applicazioni della clausola induttiva, e questo garantisce che la relazione \prec_A sia ben fondata.

Esempio 7.5.9 (Relazioni di precedenza). *La Definizione 3.1.1 induce sui naturali la relazione $Succ \in Rel(\mathbb{N}, \mathbb{N}) = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid y = x + 1\}$ presentata nell'Esempio 2.1.8. Si osservi che la sua chiusura transitiva (non riflessiva), $Succ^+$ coincide con la relazione $<$ (minore stretto) sui naturali.*

La Definizione 7.1.1 induce sulle liste in L_A relazione $\prec_{L_A} = \{(lst, a: lst) \mid lst \in L_A, a \in A\}$.

La Definizione 7.2.1 induce sugli alberi binari in BT la relazione $\prec_{BT} = \{(t', t) \mid t' \text{ è sottoalbero destro o sinistro di } t\}$.

Spesso la definizione induttiva di una funzione su un insieme A definito induttivamente induce la stessa relazione di precedenza, cioè \prec_A . Questo vale per esempio per la successione dei numeri triangolari (Definizione 3.1.2) e per il fattoriale (Definizione 3.1.13) definiti induttivamente sui naturali, per le operazioni strutturali su liste (Definizione 7.1.3), ed altre. Questo garantisce automaticamente che tali definizioni ricorsive siano ben date, e quindi le funzioni da esse caratterizzate sono totali.

Tuttavia non sempre è così. Per esempio la relazione di precedenza indotta dalla successione di Fibonacci (Definizione 3.3.1) è diversa da $Succ$. Infatti

$$\prec_{fib} = \{(n-1, n), (n-2, n) \mid n > 2\}$$

Non è difficile verificare direttamente che \prec_{fib} è ben fondata, ma questo segue anche immediatamente dai seguenti fatti, di cui lasciamo la facile dimostrazione al lettore.

Proposizione 7.5.10 (Proprietà di relazioni ben fondate).

1. *Se \sqsubset è una relazione ben fondata e $\sqsubset_1 \subseteq \sqsubset$, allora anche \sqsubset_1 è ben fondata.*
2. *Una relazione \sqsubset è ben fondata se e solo se lo è la sua chiusura transitiva \sqsubset^+ .*

Poichè $Succ$ è ben fondata, anche $Succ^+$ lo è per il punto 2. Nell’Esempio 7.5.9 abbiamo osservato che $Succ^+$ è la relazione $<$ su \mathbb{N} , e chiaramente $\prec_{fib} \subseteq <$, quindi \prec_{fib} è ben fondata per il punto 1. Più in generale, qualunque definizione ricorsiva sui naturali in cui il valore per un $n \in N$ è espresso in termini del valore su argomenti *strettamente* minori di n è ben data.

Per concludere questa sezione torniamo alla Definizione (7.2).

Esempio 7.5.11 (La congettura di Collatz). *Consideriamo di nuovo la funzione (7.2), che ricopiamo qui per convenienza:*

$$f(n) = \begin{cases} 1 & \text{se } n \leq 1 \\ f(n/2) & \text{se } n > 1 \text{ ed è pari} \\ f(3 \cdot n + 1) & \text{se } n > 1 \text{ ed è dispari} \end{cases} \quad (7.6)$$

È facile convincersi che come relazione è univalente, e che se $f(k)$ è definito per un $k \in \mathbb{N}$ allora deve valere $f(k) = 1$. Quindi la funzione costante $f(n) = 1$ per ogni $n \in \mathbb{N}$ soddisfa le tre clausole della definizione, guardandola “dichiarativamente”. Tuttavia è un problema aperto stabilire se la valutazione di $f(k)$ termini per qualunque $k \in \mathbb{N}$ oppure no. La Congettura di Collatz afferma che questa funzione è totale: essa rimane al momento solo una congettura, anche se si è verificato che la valutazione termina per tutti i numeri fino a $5,764 \cdot 10^{18}$. Questa congettura rimane uno dei più interessanti problemi aperti della matematica per la semplicità della sua formulazione.

7.6 Tipologie di Ricorsione

Le funzioni ricorsive (e induttive) delle sezioni precedenti presentavano tutte una forma di ricorsione *diretta*: in alcune clausole la valutazione della funzione poteva richiedere la valutazione della stessa su argomenti diversi. Esistono altre tipologie di definizioni ricorsive, di cui presentiamo di seguito una breve rassegna. Ne mostreremo qualche esempio, ma senza approfondirne la discussione: le problematiche di terminazione descritte nella sezione precedente si presentano naturalmente anche in questi casi.

Ricorsione annidata

Si ha *ricorsione annidata* quando una funzione compare come parametro in una sua chiamata ricorsiva.

Esempio 7.6.1. La funzione 91 di McCarthy è definita nel seguente modo:

$$f(n) = \begin{cases} n - 10 & \text{se } n > 100 \\ f(f(n + 11)) & \text{se } n \leq 100 \end{cases} \quad (7.7)$$

Questa funzione è definita per ricorsione annidata. Si vede subito che come relazione è univalente, ma non è immediato vedere che sia totale, perché nella clausola ricorsiva la funzione è applicata due volte. Con un po’ di sforzo si può verificare che $f(n) = 91$ per ogni $n \leq 100$ e $f(n) = n - 10$ per ogni $n > 100$.

Esempio 7.6.2. La funzione di Ackermann è definita per ricorsione annidata come segue:

$$A(m, n) = \begin{cases} n + 1 & \text{se } m = 0 \\ A(m - 1, 1) & \text{se } m > 0 \text{ e } n = 0 \\ A(m - 1, A(m, n - 1)) & \text{se } m > 0 \text{ e } n > 0 \end{cases} \quad (7.8)$$

Si può dimostare che la definizione caratterizza una funzione totale. Si tratta di una funzione che cresce molto rapidamente, anche per valori piccoli del primo argomento.

Ricorsione mutua

Quando la chiamata ricorsiva avviene indirettamente, cioè da parte di una seconda funzione chiamata a sua volta, direttamente o indirettamente, dalla prima, si parla di *ricorsione mutua* o *indiretta*.

Un esempio artificiale, dove le due funzioni sono ovviamente totali su \mathbb{N} , è il seguente:

$$ping(n) = \begin{cases} 0 & \text{se } n = 0 \\ pong(n - 1) & \text{altrimenti} \end{cases} \quad pong(n) = \begin{cases} 0 & \text{se } n = 0 \\ ping(n - 1) & \text{altrimenti} \end{cases}$$

Ricorsione procedurale

In questo capitolo abbiamo discusso della definizione di *funzioni ricorsive*, intendendo *funzioni* nel senso matematico del termine, come introdotte nella Definizione 2.5.1. Ma quando si deve risolvere un problema con un programma, nella maggior parte dei linguaggi di programmazione si possono usare anche effetti collaterali (come input e output, o modifica di variabili/array con assegnamento) che non sono rappresentabili immediatamente come funzioni matematiche. In molti linguaggi una *procedura* può essere invocata passando degli argomenti, ma non ci si aspetta che restituisca un risultato. Un tipico esempio sono le funzioni in C/C++ o i metodi in Java che sono dichiarati di tipo **void**.

Anche le procedure naturalmente possono essere ricorsive, terminando quando l'input è “abbastanza piccolo” (tipicamente senza fare niente), o altrimenti invocando se stesse su argomenti “più piccoli”. Come semplice esempio illustrativo vediamo una procedura C che stampa ricorsivamente tutti gli elementi di un array.

```
void stampa_array(int a[], int i, int n) {
    if (i < n) // clausola ricorsiva: c'e' ancora qualcosa da stampare
        {   printf("%d ", a[i]);
            stampa_array(a, i+1, n);
        }
    // clausola base: i == n; l'array a[i..n-1] e' vuoto, la procedura termina
}
```

CAPITOLO 8

Linguaggi Formali

Ogni linguaggio scritto, sia questo un linguaggio naturale come l’italiano o l’inglese, un linguaggio di programmazione o il linguaggio logico che abbiamo utilizzato talvolta nei capitoli precedenti e che vedremo più in dettaglio nel prossimo capitolo, coinvolge due aspetti che è importante distinguere: la *sintassi* e la *semantica*. La sintassi ha a che fare con la struttura (o la forma) delle frasi esprimibili. La semantica, invece, ha a che fare con il significato delle frasi esprimibili. Se consideriamo per un momento il linguaggio italiano con cui comunichiamo con i nostri simili, due frasi come “la mela mangia il bambino” e “il bambino mangia la mela” obbediscono entrambe alla sintassi dell’italiano, in quanto sono costruite secondo lo schema <soggetto> <verbo> <complemento-oggetto> a partire da stringhe costruite correttamente. D’altra parte, solo la seconda ha un significato, o semantica, ragionevole.

In questo capitolo ci concentreremo sul primo aspetto, la descrizione della sintassi dei linguaggi. Il punto di partenza del nostro studio è la concezione del termine *linguaggio* come sinonimo di insieme di frasi (sintatticamente) ammissibili. In altre parole, una volta fissato un *alfabeto* di elementi di base, detti anche *simboli*, un linguaggio non sarà altro che un sottoinsieme di tutte le stringhe (cioè sequenze di lunghezza arbitraria) di simboli.

Si pensi per esempio al linguaggio delle espressioni dell’aritmetica, ottenibili a partire dai numeri e dalle quattro operazioni $+$, $-$, \times e \div . Fra tutte le possibili sequenze (o stringhe) di numeri ed operazioni aritmetiche, ve ne sono di ammissibili, come $3 \times 4 + 2$, e di non ammissibili, come $3 + \times + 4$. Quindi, il linguaggio delle espressioni aritmetiche può essere identificato con il sottoinsieme delle stringhe ammissibili.

In modo del tutto analogo, un linguaggio di programmazione può essere identificato con l’insieme delle proprie stringhe ammissibili, che comunemente chiamiamo programmi. Secondo questo punto di vista, descrivere un linguaggio significa sostanzialmente descrivere l’insieme delle stringhe del linguaggio stesso, ovvero avere un metodo per:

1. decidere quali stringhe fanno parte di tale insieme, e quali non ne fanno parte, oppure
2. costruire tale insieme, enumerando le stringhe che lo compongono.

Il problema insomma è: *come identificare l’insieme delle stringhe ammissibili, che caratterizza un linguaggio?* Esistono due approcci principali a questo problema. Il primo si basa su uno strumento, detto *automa*, che è in grado di riconoscere (o accettare) tutte e sole le stringhe che fanno parte di un linguaggio. Il secondo si basa su uno strumento, detto *grammatica*, che è in grado di generare (o costruire) tutte e sole le stringhe che fanno parte di un linguaggio.

Molti studi sono stati dedicati alla definizione e al riconoscimento dei linguaggi. La teoria che è stata sviluppata è conosciuta con il nome di *teoria degli automi* o *teoria dei linguaggi formali* e le sue definizioni di base e tecniche fanno parte del nucleo dell’informatica.

Questo capitolo intende fornire una guida rapida alle idee e ai risultati principali riguardanti gli automi e le grammatiche, al fine di comprendere i metodi con cui si descrive la sintassi dei linguaggi formali, siano essi linguaggi di programmazione o il linguaggio logico definito nel prossimo capitolo¹.

¹Alcune parti di questo capitolo sono largamente ispirate dalle dispense “Elementi di Sintassi dei Linguaggi di Programmazione” di Barbuti, Mancarella, Pedreschi, Turini.

8.1 Alfabeti, Parole e Linguaggi

Alla luce delle considerazioni fatte, formalizziamo il concetto di linguaggio. Per cominciare abbiamo bisogno di fissare un alfabeto.

Definizione 8.1.1 (alfabeto). *Un ALFABETO è un insieme finito. I suoi elementi vengono detti SIMBOLI.*

Esempi tipici di alfabeti, con cui il lettore avrà un po' di familiarità sono l'alfabeto Latino $\{a, b, c \dots z\}$, alfabeto Latino Maiuscolo $\{A, B, C \dots Z\}$ e l'alfabeto Greco $\{\alpha, \beta, \gamma, \dots, \omega\}$. Oltre agli alfabeti delle lingue naturali, possiamo considerare anche l'alfabeto delle cifre decimali $\{0, 1, \dots, 9\}$, delle cifre esadecimali $\{0, 1, \dots, 9, A, B, \dots, F\}$, delle cifre binarie $\{0, 1\}$, o l'alfabeto per i numeri romani $\{I, V, X, L, C, D, M\}$.

Il passo successivo consiste nel definire una parola su A semplicemente come una stringa su A . Abbiamo già visto la definizione di stringa (come sequenza di lunghezza arbitraria) nella Sezione 2.7. Per comodità del lettore, la riportiamo di seguito, adeguata a questo nuovo contesto.

Definizione 8.1.2 (stringhe, parole). *Dato un alfabeto A , una STRINGA (o PAROLA) su A è una sequenza (di lunghezza arbitraria) $a_0 \dots a_n$ tale che ogni a_i è un simbolo dell'alfabeto A (cioè $a_i \in A$) ed n è un arbitrario numero naturale.² La LUNGHEZZA di una stringa $a_0 \dots a_n$ è $n + 1$. Esiste una sola stringa di lunghezza 0 che viene detta STRINGA VUOTA e denotata da ε^3 . L'insieme di tutte le stringhe su A è denotato da A^* .*

Sull'alfabeto Latino sono stringhe per esempio *pippo*, *bob*, *anna*, *hrjk*, ε , *iiiiii*. Su quello Latino Maiuscolo sono stringhe *PIPPO*, *BOB*, *ANNA*, *HRJK*, ε , *IIIIII*. Sull'alfabeto delle cifre decimali: 1981, 27, 10, 500, 0010. Sull'alfabeto delle cifre esadesimali *F2*, *23A*, *A55C*, *3A8*. Sull'alfabeto delle cifre binarie 10, 101, 1101, 111111. Sull'Alfabeto dei numeri romani *I*, *IV*, *XII*, *IIII*, *VX*.

Di seguito riportiamo una diversa (ma equivalente) definizione induttiva che ci sarà molto utile nel seguito.

Definizione 8.1.3 (stringhe, induttivamente). *Sia A un alfabeto. L'insieme A^* delle STRINGHE su A è il più piccolo insieme che soddisfa:*

1. [CLAUSOLA BASE] $\varepsilon \in A^*$, dove ε è la STRINGA VUOTA.
2. [CLAUSOLA INDUTTIVA] Se $w \in A^*$ e $a \in A$ allora $aw \in A^*$.

Esercizio 8.1.4. *Sia A un alfabeto e $\sqsubseteq_A \subseteq A \times A$ un ordinamento su A . Definire per induzione strutturale una funzione $\sqsubseteq_{A^*} : A^* \times A^* \rightarrow \text{Bool}$ tale che $\sqsubseteq_{A^*}(s, t) = \text{t}$ se e solo se s è minore uguale a t nell'ordinamento lessicografico (Definizione 4.5.12).*

Osservazione 8.1.5. *L'attento lettore avrà notato una certa somiglianza tra la definizione induttiva di A^* e quella di L_A , l'insieme delle liste su A (Definizione 7.1.1). In effetti, i due insiemi sono in biezione (la dimostrazione è lasciata come esercizio al lettore).*

Il lettore si starà dunque chiedendo: perché introdurre due diverse definizioni per lo stesso concetto? La ragione è che in molti linguaggi di programmazione per stringa si intende tipicamente una stringa in cui l'alfabeto A è fissato essere quello dei caratteri alfanumerici (torneremo a breve su questo punto) mentre, per la lista, l'insieme degli elementi A è solitamente arbitrario e può contenere elementi più strutturati come ad esempio una lista di tuple, una lista di interi, una lista di liste o una lista di programmi.

Esercizio 8.1.6. *Dimostrare per induzione che $L_A \cong A^*$.*

Nel Capitolo 7, abbiamo visto che le liste possono essere concatenate (Definizione 7.1.9). Riportiamo qua sotto una definizione non induttiva e abbastanza intuitiva di concatenazione di stringhe.

²Useremo di solito *parola* invece di *stringa* quando la sequenza di caratteri è intesa essere una parola di un linguaggio naturale.

³Assumiamo che $\varepsilon \notin A$.

Definizione 8.1.7 (concatenazione di stringhe). *La CONCATENAZIONE DI STRINGHE è una funzione $_ \cdot _ : A^* \times A^* \rightarrow A^*$ definita per tutte le stringhe $u, v \in A^*$ come*

$$u \cdot v = uv.$$

Per esempio se l’alfabeto è quello Latino si ha che

$$anna \cdot rella = annarella \quad man \cdot gio = mangio \quad hrj \cdot tyr = hrjtyr$$

Se invece consideriamo l’alfabeto Latino Maiuscolo

$$ANNA \cdot RELLA = ANNARELLA \quad ANNA \cdot \varepsilon = ANNA$$

O per l’alfabeto delle cifre decimali

$$19 \cdot 81 = 1981$$

Si noti che in alcuni linguaggi di programmazione come Java e Javascript, l’operatore \cdot è denotato da $+$.

Possiamo finalmente dare una definizione del concetto di linguaggio.

Definizione 8.1.8 (linguaggio). *Dato un alfabeto A , un LINGUAGGIO SU A è un insieme $L \subseteq A^*$. L’insieme di tutti i linguaggi è denotato con $\mathcal{P}(A^*)$.*

Per esempio, il vocabolario italiano è un linguaggio sull’alfabeto Latino $\{a, b, c \dots z\}$. L’Italiano e l’Inglese sono linguaggi sull’alfabeto che contiene i caratteri latini maiuscoli, minuscoli, spazi e punteggiatura.

L’insieme delle espressioni aritmetiche è un linguaggio su $\{0, 1, 2 \dots 9, +, -, \times, \div\}$. Si noti che non tutte le stringhe su tale alfabeto sono espressioni aritmetiche: per esempio $5 + 4$ è una espressione aritmetica, ma $5 + \times 4 \div$ non lo è.

In modo del tutto analogo l’insieme dei numeri romani è un linguaggio su $\{I, V, X, L, C, D, M\}$. Si noti che non tutte le stringhe su tale alfabeto sono numeri romani: per esempio VX non è un numero romano.

Secondo questa visione, la maggior parte dei *linguaggi di programmazione* sono essenzialmente sottoinsiemi di **ASCII**⁴, dove **ASCII** è l’alfabeto dei caratteri alfabetici, numeri e di interpunzione presenti, secondo uno standard internazionale, sulla tastiera alfanumerica di ogni computer.⁴ In altre parole, un programma non è altro che una stringa di caratteri alfanumerici **ASCII**. Ovviamente, non tutte le stringhe siffatte sono programmi accettabili, così come i diversi linguaggi di programmazione corrispondono a diversi insiemi di tali stringhe. Ogni linguaggio di programmazione ha la propria sintassi, ovvero le proprie regole che descrivono la struttura delle stringhe **ASCII** che corrispondono a programmi ammissibili, o sintatticamente ben formati.

Le Sezioni 8.2 e 8.3 sono dedicate alla presentazione di due metodi, gli automi e le grammatiche, rivolti proprio a descrivere la struttura sintattica dei linguaggi. Prima di far ciò, è interessante però studiare la struttura algebrica dell’insieme di tutti i linguaggi $\mathcal{P}(A^*)$.

Operazioni su linguaggi

I linguaggi su uno stesso alfabeto A possono essere combinati, in modi molto simili a come vengono combinate le relazioni su uno stesso insieme. Iniziamo illustrando nel seguente esempio, tre linguaggi “notevoli”.

Esempio 8.1.9. *Per un qualsiasi alfabeto A , esistono sempre i seguenti linguaggi:*

- *Il linguaggio vuoto $\emptyset = \{\}$,*
- *Il linguaggio che contiene solo la stringa vuota $\{\varepsilon\}$,*
- *Il linguaggio completo A^* .*

⁴Alcuni linguaggi usano altri standard per i caratteri, come UNICODE, ma il discorso non cambia.

8. Linguaggi Formali

È interessante notare l'analogia con le relazioni su A in cui si ha la relazione vuota ($\emptyset_{A,A}$), la relazione identità (id_A) e la relazione completa ($A \times A$).

Come le relazioni, i linguaggi possono essere combinati, utilizzando le solite operazioni insiemistiche:

- Unione: $L_1 \cup L_2 = \{w \in A^* \mid w \in L_1 \text{ oppure } w \in L_2\};$
- Intersezione: $L_1 \cap L_2 = \{w \in A^* \mid w \in L_1 \text{ e } w \in L_2\};$
- Differenza: $L_1 \setminus L_2 = \{w \in A^* \mid w \in L_1 \text{ e } w \notin L_2\};$
- Complemento: $\bar{L} = \{w \in A^* \mid w \notin L\}.$

Si noti che nel complemento l'insieme universo \mathcal{U} è inteso essere il linguaggio completo A^* .

Utilizzando la concatenazione di stringhe, si possono concatenare i linguaggi.

Definizione 8.1.10 (concatenazione di linguaggi). *La funzione $_\cdot_\colon \mathcal{P}(A^*) \times \mathcal{P}(A^*) \rightarrow \mathcal{P}(A^*)$ è definita per tutti i linguaggi $L_1, L_2 \in \mathcal{P}(A^*)$ come*

$$L_1 \cdot L_2 = \{w \in A^* \mid \text{esistono } w_1 \in L_1, w_2 \in L_2 \text{ tali che } w = w_1 \cdot w_2\}.$$

Esempio 8.1.11. Mostriamo alcuni esempi di concatenazione di linguaggi senza dichiarare esplicitamente l'alfabeto.

- Sia $L_1 = \{\text{anna, so}\}$ e $L_2 = \{\text{rella}\}$, $L_1 \cdot L_2 = \{\text{annarella, sorella}\};$
- Sia $L_1 = \{\text{anna, so}\}$ e $L_2 = \{\text{rella, le}\}$, $L_1 \cdot L_2 = \{\text{annarella, sorella, annale, sole}\};$
- Sia $L_1 = \{2, 4, 8\}$ e $L_2 = \{3, 5, 7\}$, $L_1 \cdot L_2 = \{23, 25, 27, 43, 45, 47, 83, 85, 87\};$
- Sia $L_1 = \{\varepsilon, V, L, D\}$ e $L_2 = \{I, II, III\}$,

$$L_1 \cdot L_2 = \{I, II, III, VI, VII, VIII, LI, LII, LIII, DI, DII, DIII\};$$

- Sia $L_1 = \emptyset$ e $L_2 = \{\text{anna, so}\}$, $L_1 \cdot L_2 = \emptyset.$

Teorema 8.1.12. Per tutti gli alfabeti A e per tutti i linguaggi $L, L_1, L_2, L_3 \in \mathcal{P}(A^*)$, valgono le uguaglianze nella Tabella 8.1.

associatività	$L_1 \cdot (L_2 \cdot L_3) = (L_1 \cdot L_2) \cdot L_3$
unità	$L \cdot \{\varepsilon\} = L = \{\varepsilon\} \cdot L$
assorbimento	$L \cdot \emptyset = \emptyset = \emptyset \cdot L$
distributività di \cdot su \cup (sinistra)	$L_1 \cdot (L_2 \cup L_3) = (L_1 \cdot L_2) \cup (L_1 \cdot L_3)$
distributività di \cdot su \cup (destra)	$(L_1 \cup L_2) \cdot L_3 = (L_1 \cdot L_3) \cup (L_2 \cdot L_3)$

Tabella 8.1: Leggi per la composizione di linguaggi

La dimostrazione è lasciata per esercizio.

Esercizio 8.1.13. Dimostrare il Teorema 8.1.12.

Le leggi in Tabella 8.1, assieme ad associatività, commutatività e unità di \cup (che ovviamente valgono) ci dicono che $(\mathcal{P}(A^*), \cup, \emptyset, \cdot, \{\varepsilon\})$ forma una struttura ben nota in algebra chiamata *semianello*. È interessante notare che la concatenazione \cdot di linguaggi obbedisce alle stesse leggi della composizione ; di relazioni (se pensiamo al linguaggio $\{\varepsilon\}$ come la relazione id_A).

Analogamente alle relazioni (Definizione 4.3.13) possiamo definire il linguaggio L^n per ogni numero naturale n .

Definizione 8.1.14 (concatenazione n -aria di linguaggi). *Sia A un insieme e $L \in \mathcal{P}(A^*)$ un linguaggio. Per ogni numero naturale $n \in \mathbb{N}$ definiamo L^n induttivamente:*

1. [CLAUSOLA BASE] $L^0 = \{\varepsilon\}$.
2. [CLAUSOLA INDUTTIVA] $L^{n+1} = L \cdot L^n$.

Esempio 8.1.15. Sia $A = \{a, b, c\}$ e $L = \{ab, bb\}$. Allora $L^0 = \{\varepsilon\}$.

$$\begin{aligned} L^1 &= L \cdot L^0 && (\text{CLAUSOLA INDUTTIVA}) \\ &= L \cdot \{\varepsilon\} && (\text{CLAUSOLA BASE}) \\ &= L && (\text{unità}) \\ &= \{ab, bb\} \end{aligned}$$

In modo analogo

$$\begin{aligned} L^2 &= L \cdot L^1 && (\text{CLAUSOLA INDUTTIVA}) \\ &= \{ab, bb\} \cdot \{ab, bb\} \\ &= \{abab, abbb, bbab, bbbb\} \end{aligned}$$

e

$$\begin{aligned} L^3 &= L \cdot L^2 && (\text{CLAUSOLA INDUTTIVA}) \\ &= \{ab, bb\} \cdot \{abab, abbb, bbab, bbbb\} \\ &= \{ababab, ababbb, abbbab, abbbbb, bbabab, bbabbb, bbbbbab, bbbbbbb\} \end{aligned}$$

Se $L = \{ab\}$, allora $L^0 = \{\varepsilon\}$, $L^1 = \{ab\}$, $L_2 = \{abab\}$, $L^3 = \{ababab\}$ e così via ...

Osservazione 8.1.16. Quando il linguaggio L consiste di una sola stringa w , cioè $L = \{w\}$, si ha che L^n consiste sempre di una sola stringa per tutti gli $n \in \mathbb{N}$. Questa stringa è sempre la concatenazione di w con se stessa n volte. Nel resto di queste note, ed in particolare nelle grammatiche libere da contesto, utilizzeremo spesso la notazione w^n per denotare l'unica stringa del linguaggio $\{w\}^n$, cioè la stringa che consiste della concatenazione di w con se stessa n volte. Per esempio, ab^3 è proprio $ababab$.

Definizione 8.1.17 (stella di Kleene). La stella di Kleene è una funzione $(_)^*: \mathcal{P}(A^*) \rightarrow \mathcal{P}(A^*)$ definita per tutti i linguaggi $L \in \mathcal{P}(A^*)$ come

$$L^* = \bigcup_{n \in \mathbb{N}} L^n$$

Intuitivamente la stella di Kleene di un linguaggio L è il linguaggio

$$\{\varepsilon\} \cup L \cup L \cdot L \cup L \cdot (L \cdot L) \cup L \cdot (L \cdot (L \cdot L)) \cup \dots$$

Esempio 8.1.18. Esempi sull'alfabeto $A = \{a, b, c\}$.

- Se $L = \{a\}$, allora $L^* = \{\varepsilon, a, aa, aaa, aaaa, \dots\}$
- Se $L = \{a, b\}$, allora $L^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, \dots\}$
- Se $L = \{ab\}$, allora $L^* = \{\varepsilon, ab, abab, ababab, \dots\}$
- Se $L = \{ab, c\}$, allora $L^* = \{\varepsilon, ab, c, abc, cab, ababc, abcab, cabab, \dots\}$
- Se $L = \{aa, bb\}$, allora $L^* = \{\varepsilon, aa, bb, aabb, bbaa, aaaa, bbbb, aabbaa, \dots\}$

Teorema 8.1.19. Per tutti gli alfabeti A e per tutti i linguaggi $L, L_1, L_2 \in \mathcal{P}(A^*)$, valgono le uguaglianze e le inclusioni nella Tabella 8.2.

La dimostrazione è lasciata per esercizio.

Esercizio 8.1.20. Dimostrare il Teorema 8.1.19.

riflessività	$\{\varepsilon\} \subseteq L^*$
transitività	$L^* \cdot L^* \subseteq L^*$
chiusura	$L \subseteq L^*$
idempotenza	$(L^*)^* = L^*$
\star -id	$\{\varepsilon\}^* = \{\varepsilon\}$
\star -vuoto	$\emptyset^* = \{\varepsilon\}$
distributività di \star su \cup	$L_1^* \cup L_2^* \subseteq (L_1 \cup L_2)^*$

Tabella 8.2: Leggi della stella di Kleene per linguaggi.

Osservazione 8.1.21 (Analogia con l'algebra relazionale). *Nel Capitolo 2 abbiamo introdotto operazioni su relazioni e le loro leggi algebriche. Nel Capitolo 4 abbiamo esteso queste leggi per la stella di Kleene (chiusura riflessiva e transitiva). Considerando l'insieme $\mathcal{P}(A \times A)$ di tutte le relazioni su A con le operazioni di \cup , \cdot , $\emptyset_{A,A}$ e id_A otteniamo il ben noto frammento di Kleene dell'algebra delle relazioni:*

$$(\mathcal{P}(A \times A), \cup, \emptyset_{A,A}, \cdot, id_A, (-)^*)$$

Per il linguaggi abbiamo delle operazioni analoghe:

$$(\mathcal{P}(A^*), \cup, \emptyset, \cdot, \{\varepsilon\}, (-)^*)$$

Le Tabelle 8.1 e 8.2 ci dicono che queste due strutture soddisfano le stesse proprietà algebriche.

8.2 Automi

Svariati strumenti di calcolo sono descritti attraverso *macchine a stati*, cioè vengono descritti attraverso transizioni di stato, letture di input e produzioni di output. Tipici esempi di macchine a stati che giocano un ruolo chiave in Informatica sono le macchine di Mealy e di Moore (utilizzate nella progettazione dei processori) o le ben note macchine di Turing (che possono esprimere tutti i programmi calcolabili). In questa sezione introduceremo un esempio classico di macchina a stati, gli *automi*, che permettono di riconoscere le stringhe di un linguaggio. L'idea è che queste macchine prendono come input una stringa e la leggono carattere per carattere: la lettura di un certo carattere fa innescare un passaggio di stato.

Per concretizzare questa idea, consideriamo un problema specifico di riconoscimento di stringhe: “quali parole inglesi contengono le cinque vocali in ordine alfabetico”? Per rispondere a questa domanda, possiamo utilizzare la lista di parole fornita da molti sistemi operativi. Alcune delle parole di questo file che contengono le cinque vocali in ordine, sono: *facetious, sacrilegious*. Si noti come anche la seconda parola va bene secondo la formulazione del problema: basta ignorare la prima occorrenza da sinistra della vocale *i* in *sacrilegious*. Esaminiamo come una macchina molto semplice può trovare tutte le parole che contengono le cinque vocali in ordine. La macchina può leggere ciascuna parola ed esaminarne i caratteri. Iniziando l'esame della parola da sinistra verso destra la macchina cerca una *a*. Diciamo che la macchina si trova nello “stato 0” fintanto che non trova una *a*, mentre quando la trova passa nello “stato 1”. Nello stato 1, la macchina cerca una *e* e, quando ne trova una, passa nello “stato 2”. Si procede in questo modo fino a raggiungere lo “stato 4” in cui si cerca una *u*. Se viene trovata una *u*, allora la parola contiene le vocali, ordinate alfabeticamente, e la macchina può terminare nello stato di riconoscimento, “stato 5”, in cui riconosce la parola. Esaminando il resto della parola, la macchina continua a rimanere nello stato 5 in presenza di ogni nuovo carattere letto, dato che sappiamo che la stringa in questione è riconosciuta indipendentemente da ciò che segue u. Possiamo interpretare lo stato *i* come l'indicazione che la macchina ha già trovato le prime *i* vocali, in ordine alfabetico, per $i = 0, 1, \dots, 5$. I sei stati riassumono tutto quello che il programma deve ricordare durante l'esame della parola, da sinistra a destra. Per esempio, nello stato 0, quando la macchina cerca una *a* non ha bisogno di ricordare se ha già incontrato una *e*. La ragione è che questa *e* non è preceduta da nessuna *a* e quindi non può servire come *e* nella sottosequenza *aeiou*.

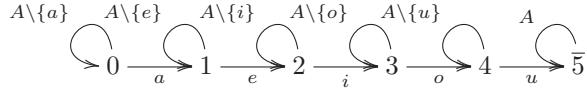


Figura 8.1: Un automa che riconosce tutte le parole che contengono le cinque vocali in ordine alfabetico. A è l’alfabeto Latino.

In Figura 8.1 è rappresentata la macchina a stati descritta nei paragrafi precedenti. Gli stati $0, \dots, 5$ sono visti come i nodi di un grafo e i passaggi di stato, in gergo tecnico *transizioni*, come degli archi. Gli archi sono etichettati da quei simboli dell’alfabeto A la cui lettura innesca la transizione. Per esempio dallo stato 0 leggendo la lettera a si passa allo stato 1. Invece leggendo un qualsiasi altro simbolo dell’alfabeto (cioè un qualsiasi elemento di $A \setminus \{a\}$) la macchina rimane nello stesso stato 0. Si noti che sopra lo stato 5 è disegnata una sbarra. Tale simbolo denota che lo stato è di accettazione: se la lettura di una stringa conduce a tale stato, allora la stringa contiene le vocali in ordine alfabetico.

Formalizziamo l’intuizione illustrata fino ad ora con la seguente definizione.

Definizione 8.2.1 (automa). *Sia A un alfabeto. Un AUTOMA sull’alfabeto A è una tripla $\mathcal{A} = (S, T, F)$ dove:*

- S è un insieme, detto INSIEME DEGLI STATI;
- $T \subseteq (A \times S) \times S$ è una relazione in $\text{Rel}(A \times S, S)$, detta RELAZIONE DI TRANSIZIONE, che associa ad ogni lettera dell’alfabeto $a \in A$ e ad ogni stato di partenza $x \in S$ zero o più stati di arrivo.
- $F \subseteq S$ è l’insieme degli STATI FINALI (anche detti stati di accettazione).

L’automa \mathcal{A} si dice A STATI FINITI se l’insieme degli stati S è finito.

Intuitivamente, gli elementi di S sono tutti gli stati, e quelli di F sono gli stati di accettazione. Un elemento $((a, x), y) \in (A \times S) \times S$ rappresenta una transizione che porta l’automa dallo stato di partenza x allo stato di arrivo y leggendo il simbolo $a \in A$.

Esempio 8.2.2. La macchina a stati che riconosce le stringhe che contengono le vocali in ordine alfabetico (rappresentata in Figura 8.1) è un automa: l’insieme degli stati è $S = \{0, \dots, 5\}$, la relazione di transizione è

$$\begin{aligned} T = & \{((a, 0), 1), ((e, 1), 2), ((i, 2), 3), ((o, 3), 4), ((u, 4), 5)\} \cup \\ & \{((i, 0), 0) \mid i \in A \setminus \{a\}\} \cup \\ & \{((i, 1), 1) \mid i \in A \setminus \{e\}\} \cup \\ & \{((i, 2), 2) \mid i \in A \setminus \{i\}\} \cup \\ & \{((i, 3), 3) \mid i \in A \setminus \{o\}\} \cup \\ & \{((i, 4), 4) \mid i \in A \setminus \{u\}\} \cup \\ & \{((i, 5), 5) \mid i \in A\} \end{aligned}$$

mentre l’insieme degli stati finali è $F = \{5\}$.

Questo esempio mostra quanto la rappresentazione di automi attraverso relazioni possa essere poco intuitiva se paragonata alla rappresentazione grafica (illustrata in Figura 8.1). Per questa ragione, è conveniente spendere qualche parola sulla notazione grafica che è proprio quella che utilizzeremo d’ora in avanti.

Un automa $\mathcal{A} = (S, T, F)$ viene rappresentato in maniera diagrammatica da un grafo orientato con etichette. Gli stati $x \in S$ sono rappresentati come i nodi del grafo. Gli stati finali $y \in F$ sono nodi con una sbarra orizzontale, tipo \bar{y} . Ogni transizione $((a, x), y) \in T$ è rappresentata da un arco orientato da x a y con etichetta a (l’etichetta può trovarsi, sopra, sotto o al centro dell’arco).

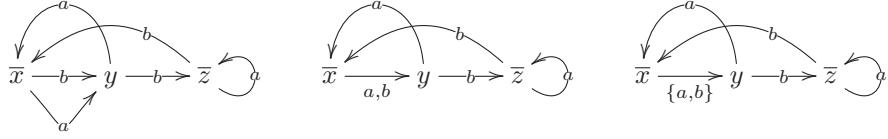


Figura 8.2: Tre diverse rappresentazioni grafiche dello stesso automa: nell'automa di sinistra tutti gli archi sono etichettati da esattamente un simbolo. Nell'automa di centro un arco è etichettato con due simboli e nell'automa di destra lo stesso arco è etichettato con un insieme di simboli.

Esempio 8.2.3. Sia A l'alfabeto $\{a, b\}$ e $\mathcal{A} = (S, T, F)$ definito come $S = \{x, y, z\}$, $F = \{x, z\}$ e

$$\begin{aligned} T = \{ & ((a, x), y), ((b, x), y), \\ & ((a, y), x), ((b, y), z), \\ & ((a, z), z), ((b, z), x) \}. \end{aligned}$$

Tale automa è rappresentato dal grafo a sinistra della Figura 8.2.

Talvolta, quando ci sono due (o più) transizioni con gli stessi nodi di partenza e di arrivo e con simboli diversi, si possono accorpate i due (o più) archi scrivendone uno solo, come illustrato per esempio dalla transizione

$$x \xrightarrow{a,b} y$$

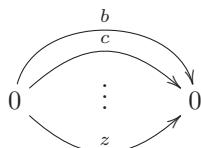
nell'automa al centro della Figura 8.2. Infine, quando i simboli sono molti, può essere conveniente etichettare l'arco con l'insieme dei simboli, come illustrato, per esempio, dalla transizione

$$x \xrightarrow{\{a,b\}} y$$

nell'automa alla destra della Figura 8.2. Si noti, come nell'automa in Figura 8.1, l'arco

$$0 \xrightarrow{A \setminus \{a\}} 0$$

potrebbe essere rappresentato come segue.



Definizione 8.2.4 (raggiungibilità in automi). Sia $\mathcal{A} = (S, T, F)$ un automa sull'alfabeto A . Per ogni $a \in A$, si definisce la relazione $T_a \in \text{Rel}(S, S)$ come

$$T_a = \{(x, y) \mid ((a, x), y) \in T\}$$

Per ogni stringa $w \in A^*$, si definisce la relazione $T_w \in \text{Rel}(S, S)$ per induzione come segue

1. [CLAUSOLA BASE] $T_\varepsilon = \text{id}_S$.
2. [CLAUSOLA INDUTTIVA] $T_{aw} = T_a; T_w$.

Se $(x, y) \in T_w$ si dice che lo stato y è RAGGIUNGIBILE da x con la stringa w o che x può raggiungere y con la stringa w .

Esempio 8.2.5. Si consideri l'automa dell'Esempio 8.2.3 rappresentato in Figura 8.2 sull'alfabeto $A = \{a, b\}$. Si ha che T_a e T_b sono le seguenti relazioni.

$$T_a = \{(x, y), (y, x), (z, z)\} \quad T_b = \{(x, y), (y, z), (z, x)\}$$

Intuitivamente T_a contiene tutte le transizioni con simbolo a e T_b contiene tutte le transizioni con simbolo b . Pensando alla rappresentazione grafica (illustrata in Figura 8.2), T_a è l'insieme di archi con etichetta a e T_b l'insieme degli archi con etichetta b . A partire da queste due relazioni possiamo costruire la relazione T_w per ogni $w \in A^*$. Per esempio per $w = abb$ si ha che

$$\begin{aligned} T_{abb} &= T_a; T_{bb} && (\text{CLAUSOLA INDUTTIVA}) \\ &= T_a; (T_b; T_b) && (\text{CLAUSOLA INDUTTIVA}) \\ &= T_a; (T_b; (T_b; T_\varepsilon)) && (\text{CLAUSOLA INDUTTIVA}) \\ &= T_a; (T_b; (T_b; id_S)) && (\text{CLAUSOLA BASE}) \\ &= T_a; T_b; T_b && (\text{associatività, unità}) \end{aligned}$$

È facile convincersi che per ogni stringa $w = a_0 \dots a_n$, la relazione T_w è esattamente

$$T_{a_0 \dots a_n} = T_{a_0}; \dots; T_{a_n}$$

Tenendo in mente l'uguaglianza riportata sopra è immediato calcolare T_w . Riportiamo di seguito tutti gli esempi con stringhe di lunghezza 2.

$$\begin{aligned} T_{aa} &= \{(x, x), (y, y), (z, z)\} \\ T_{ab} &= \{(x, z), (y, y), (z, x)\} \\ T_{ba} &= \{(x, x), (y, z), (z, y)\} \\ T_{bb} &= \{(x, z), (y, x), (z, y)\} \end{aligned}$$

Mostriamo anche un esempio con una stringa di lunghezza 3.

$$T_{aaa} = \{(x, y), (y, x), (z, z)\}$$

L'esempio appena illustrato dà un'intuizione chiara del significato di T_a e T_w . La relazione T_a contiene tutte e solo le coppie (x, y) tali che c'è una transizione da x a y con simbolo a , cioè $((a, x), y) \in T$. Nella relazione T_w ci sono tutte e sole le coppie (x, y) tali che c'è un walk (un cammino) da x a y etichettato con w . Per essere più precisi, se $w = a_0 \dots a_n$, la relazione T_w è esattamente

$$T_{a_0 \dots a_n} = \{(x, y) \mid \exists z_1, \dots, z_n \in S \text{ tali che } ((a_0, x), z_1) \in T, \dots, ((a_n, z_n), y) \in T\}.$$

Esempio 8.2.6. Consideriamo nuovamente l'automa in Figura 8.1 e concentriamoci sullo stato 0.

È facile vedere che lo stato 5 è raggiungibile dallo stato 0 con la stringa aeiou. Graficamente si può visualizzare il seguente walk:

$$0 \xrightarrow{a} 1 \xrightarrow{e} 2 \xrightarrow{i} 3 \xrightarrow{i} 4 \xrightarrow{u} 5$$

Si osservi che 5 è raggiungibile da 0 anche con la stringa baeiou, poiché $(0, 0) \in T_b$ e $(0, 5) \in T_{aeiou}$. Graficamente si può visualizzare il seguente walk:

$$0 \xrightarrow{b} 0 \xrightarrow{a} 1 \xrightarrow{e} 2 \xrightarrow{i} 3 \xrightarrow{o} 4 \xrightarrow{u} 5$$

Lo stesso vale per cbaeiou o bcbaeiou.

$$0 \xrightarrow{c} 0 \xrightarrow{b} 0 \xrightarrow{a} 1 \xrightarrow{e} 2 \xrightarrow{i} 3 \xrightarrow{o} 4 \xrightarrow{u} 5$$

$$0 \xrightarrow{b} 0 \xrightarrow{c} 0 \xrightarrow{b} 0 \xrightarrow{a} 1 \xrightarrow{e} 2 \xrightarrow{i} 3 \xrightarrow{o} 4 \xrightarrow{u} 5$$

Più in generale, per ogni stringa w_0 che non contiene a vale che 5 è raggiungibile da 0 attraverso la stringa w_0aeiou : infatti $(0, 0) \in T_{w_0}$ e $(0, 5) \in T_{aeiou}$. Con un ragionamento analogo, si può vedere che per tutte le stringhe w_1 che non contengono il simbolo e vale che 5 è raggiungibile da

0 attraverso aw_1eiou . Mostriamo di seguito la visualizzazione dei walk per $w_1 = b$, $w_1 = cb$ e $w_1 = bcb$.

$$0 \xrightarrow{a} 1 \xrightarrow{b} 1 \xrightarrow{e} 2 \xrightarrow{i} 3 \xrightarrow{o} 4 \xrightarrow{u} 5$$

$$0 \xrightarrow{a} 1 \xrightarrow{c} 1 \xrightarrow{b} 1 \xrightarrow{e} 2 \xrightarrow{i} 3 \xrightarrow{o} 4 \xrightarrow{u} 5$$

$$0 \xrightarrow{a} 1 \xrightarrow{b} 1 \xrightarrow{c} 1 \xrightarrow{b} 1 \xrightarrow{e} 2 \xrightarrow{i} 3 \xrightarrow{o} 4 \xrightarrow{u} 5$$

Ragionando in modo simile si ha che 5 è raggiungibile da 0 attraverso la stringa

$$w_0aw_1ew_2iw_3ow_4uw_5 \quad (8.1)$$

dove w_5 è una stringa arbitraria, mentre w_0, w_1, w_2, w_3, w_4 sono stringhe arbitrarie in cui non compaiono, rispettivamente, i simboli a, e, i, o, u .

La relazione T_w è fondamentale per definire il linguaggio accettato da uno stato dell'automa.

Definizione 8.2.7. Sia $\mathcal{A} = (S, T, F)$ un automa sull'alfabeto A . La funzione $\langle\langle \cdot \rangle\rangle: S \rightarrow \mathcal{P}(A^*)$ è definita per ogni stato $x \in S$ come

$$\langle\langle x \rangle\rangle = \{w \in A^* \mid \text{esiste } y \in F \text{ tale che } (x, y) \in T_w\}.$$

Il linguaggio $\langle\langle x \rangle\rangle$ è detto il LINGUAGGIO ACCETTATO da x . Se $w \in \langle\langle x \rangle\rangle$, si dice che la stringa w è ACCETTATA dallo stato x (o appartiene al linguaggio accettato da x). Se $w \notin \langle\langle x \rangle\rangle$, si dice che la stringa w NON È ACCETTATA dallo stato x (o non appartiene al linguaggio accettato da x).

Intuitivamente, per vedere se una stringa w è accettata da uno stato x si devono guardare tutti gli stati y che sono raggiungibili da x con w e, se tra questi c'è almeno uno stato finale, allora la stringa w è accettata, altrimenti non è accettata.

Esempio 8.2.8. Si consideri l'automa dell'Esempio 8.2.3 rappresentato in Figura 8.2 sull'alfabeto $A = \{a, b\}$ e consideriamo lo stato x .

Iniziamo con la stringa vuota ε . Si ha che $T_\varepsilon = id_S$ e quindi $(x, x) \in T_\varepsilon$ e, fatto ancor più interessante, x è l'unico stato raggiungibile da x con ε . Visto che $x \in F$, allora ε è accettata da x , cioè

$$\varepsilon \in \langle\langle x \rangle\rangle.$$

Consideriamo la stringa a : si ha che $(x, y) \in T_a$ e che, inoltre, y è l'unico stato raggiungibile da x attraverso la stringa a . Visto che $y \notin F$, allora la stringa a non è accettata da x , cioè

$$a \notin \langle\langle x \rangle\rangle.$$

Esattamente lo stesso argomento vale per b , possiamo quindi concludere che

$$b \notin \langle\langle x \rangle\rangle.$$

Proviamo adesso aa . L'unico stato raggiungibile da x con aa è lo stesso x che appartiene ad F . Quindi aa è accettata da x . Proviamo adesso ab . L'unico stato raggiungibile da x con ab è z che appartiene ad F . Quindi ab è accettata da x . Proviamo adesso ba . L'unico stato raggiungibile da x con ba è lo stesso x che appartiene ad F . Quindi ba è accettata da x . Proviamo adesso bb . L'unico stato raggiungibile da x con bb è z che appartiene ad F . Quindi bb è accettata da x . In sintesi:

$$aa, ab, ba, bb \in \langle\langle x \rangle\rangle.$$

Per aaa , si ha che l'unico stato raggiungibile da x con aaa è lo stato y che non appartiene ad F . Pertanto aaa non è accettata da x , cioè

$$aaa \notin \langle\langle x \rangle\rangle.$$

Esempio 8.2.9. Consideriamo nuovamente l'automa in Figura 8.1 e concentriamoci sullo stato 0. Visto che l'unico stato di accettazione è 5, una qualsiasi stringa w è accettata da 0 se e solo se 5 è raggiungibile da 0 attraverso w (cioè $(0, 5) \in T_w$). Abbiamo visto che tutte le stringhe della forma specificata da (8.1) permettono allo stato 0 di raggiungere 5. Pertanto tutte queste stringhe vengono accettate. Si noti che queste sono esattamente tutte le parole in cui c'è almeno un'occorrenza delle vocali a, e, i, o, u in ordine alfabetico, come avevamo desiderato all'inizio di questa sezione.

Esercizio 8.2.10. Sia A l'alfabeto Latino. Disegnare un automa su A in cui uno stato accetta tutte e solo le parole che iniziano con al , come per esempio algoritmo e algebra.

Esercizio 8.2.11. Si consideri l'alfabeto $A = \{a, b\}$ ed il linguaggio

$$L = \{w \in A^* \mid \text{il simbolo } a \text{ occorre un numero pari di volte in } w\}.$$

Per esempio $aba \in L$ mentre $abb \notin L$. Definire un automa in cui uno stato accetta esattamente il linguaggio L .

Automi Deterministici e Non

A questo punto, una considerazione è di fondamentale importanza. Negli esempi che abbiamo visto fino ad adesso, le relazioni di transizione sono delle funzioni: associano ad ogni coppia $(a, x) \in A \times S$ uno ed un solo stato di arrivo $y \in S$. In altre parole, da un qualsiasi stato x , leggendo un qualsiasi simbolo a , si transisce in esattamente uno stato y . Ne segue che le relazioni T_a , per ogni $a \in A$, e le relazioni T_w , per ogni $w \in A^*$, sono delle funzioni da S in S : ogni stato x raggiunge con w esattamente uno stato y . Questi automi sono chiamati *automi deterministici*.

Definizione 8.2.12 (automa deterministico). Un automa $\mathcal{A} = (S, T, F)$ si dice DETERMINISTICO se la relazione di transizione $T \in \text{Rel}(A \times S, S)$ è una funzione.

Per esempio, gli automi rappresentati nelle Figure 8.1 e 8.2 sono deterministici.

La definizione generale di automa (Definizione 8.2.1) prevede però che la relazione di transizione $T \in \text{Rel}(A \times S, S)$ sia, in tutta generalità, una relazione e non necessariamente una funzione. Ci possono quindi essere anche degli automi che non sono deterministici: questi sono noti come *automi non-deterministici*.

Esempio 8.2.13. Si consideri l'alfabeto $A = \{a, b\}$. I seguenti tre automi non sono deterministici, in quanto la relazione di transizione non è totale.

$$\begin{array}{ccc} a \text{ } \bigcirclearrowleft \text{ } \bar{x} & b \text{ } \bigcirclearrowleft \text{ } \bar{x} \xrightarrow{a} \bar{y} & x \xrightarrow{a} \bar{y} \end{array} \quad (8.2)$$

Infatti, nell'automa di sinistra e in quello di destra non esiste uno stato z tale che $((b, x), z) \in T$. Si osservi che, più in generale, x non può raggiungere alcuno stato con una qualsiasi stringa che inizia con b . Nell'automa di centro non esiste uno stato z tale che $((b, y), z) \in T$.

Il seguente automa non è deterministico in quanto la relazione di transizione non è univalente.



Infatti, vale sia che $((a, z), x) \in T$ che $((a, z), z) \in T$. Intuitivamente, leggendo il simbolo a , lo stato z può transitare sia in x che in se stesso. Si osservi che con la stringa ab lo stato z può raggiungere due stati: sia y che x . Il primo non è di accettazione, mentre il secondo lo è. Il lettore probabilmente si chiederà:

la stringa ab è accettata dallo stato z ?

La risposta è sì: per definizione di linguaggio accettato (Definizione 8.2.7) la stringa ab è accettata da z se esiste almeno uno stato di accettazione che è raggiungibile da y attraverso ab . E in questo caso, tale stato esiste: è lo stato x .

L'esempio precedente illustra un fenomeno importante che avviene negli automi che non sono deterministici: un dato stato può raggiungere con la stessa stringa diversi stati. La stringa è accettata se tra tutti questi ce n'è almeno uno di accettazione.

Questa osservazione ci permette di illustrare uno dei concetti fondamentali dell'informatica teorica: il *non determinismo*.

Osservazione 8.2.14. *Gli algoritmi che utilizziamo solitamente sono deterministici: dato uno stesso input si arriva sempre allo stesso output. Invece, negli algoritmi non deterministici uno stesso input può portare ad output diversi. Mentre gli algoritmi deterministici possono essere pensati come funzioni $\text{INPUT} \rightarrow \text{OUTPUT}$, gli algoritmi non deterministici sono delle relazioni tra INPUT e OUTPUT.*

È interessante notare che le macchine a stati con cui ci interfacciamo quotidianamente (*i computer*) sono deterministiche: procedono attraverso una serie di transizioni di stato in cui lo stato di arrivo è determinato dallo stato di partenza.

Il *non determinismo* acquista un significato ancor più specifico quando, nel contesto dell'Informatica Teorica (ad esempio nelle macchine di Turing non deterministiche), si considerano problemi con output booleano, o **t** o **f**: un algoritmo non deterministico può eseguire più computazioni che portano a risultati diversi ma, se almeno una computazione ha successo (risposta **t**), allora il problema ha risposta **t**.

Nell'esempio illustrato sopra, il problema è: $ab \in \langle\langle z \rangle\rangle$? Partendo dallo stato z e prendendo in input la stringa ab si può arrivare sia allo stato y che non è di accettazione (output **f**) che allo stato x che è di accettazione (output **t**). Il problema ha quindi risposta **t**.

La portata del *non determinismo* in informatica va ben al di là degli automi e non può essere discussa in questo corso introduttivo. Vogliamo però rammentare al lettore che il famoso problema $P = NP$ (uno dei sette problemi del millennio) è intimamente collegato a questa idea di *non determinismo*.

Nonostante che le macchine a stati con cui ci interfacciamo quotidianamente siano deterministiche, le macchine non deterministiche sono degli strumenti utilissimi per specificare problemi e modellare le loro soluzioni. Il seguente esempio illustra una comodità offerta dagli automi non deterministiche.

Esempio 8.2.15. Siamo interessati a trovare tutte le parole che finiscono con *are*, come *andare*, *mangiare*, *care* o *bare* (nella lingua italiana sono 11137). A tale scopo possiamo progettare un automa in cui uno stato accetta tutte e sole queste parole. Fissiamo A essere l'alfabeto Latino e consideriamo il seguente automa.

$$A \xrightarrow{\quad} x \xrightarrow{a} y \xrightarrow{r} z \xrightarrow{e} \bar{u} \tag{8.4}$$

Siamo interessati a $\langle\langle x \rangle\rangle$, il linguaggio accettato dallo stato x . Essendo u l'unico stato di accettazione (cioè $F = \{u\}$) una stringa w è accettata da x se e soltanto se u è raggiungibile da x attraverso w . I seguenti walk mostrano che le parole *care* e *bare* sono accettate da x .

$$x \xrightarrow{c} x \xrightarrow{a} y \xrightarrow{r} z \xrightarrow{e} u$$

$$x \xrightarrow{b} x \xrightarrow{a} y \xrightarrow{r} z \xrightarrow{e} u$$

È importante adesso notare che la relazione T_{care} non contiene solo (x, u) , ma anche la coppia (x, x) . Infatti, dopo la lettura di c (che lascia l'automa nello stato x) la lettura di a può innescare due diverse transizioni: una che finisce nello stato y , come mostrato sopra, oppure una che torna in x , come illustrato di seguito.

$$x \xrightarrow{c} x \xrightarrow{a} x \xrightarrow{r} x \xrightarrow{e} x$$

Lo stesso vale per *bare*.

$$x \xrightarrow{b} x \xrightarrow{a} x \xrightarrow{r} x \xrightarrow{e} x$$

Il lettore potrebbe adesso pensare che questo comportamento sia anomalo e che si possa eliminare considerando il seguente automa.

$$A \setminus \{a\} \xrightarrow{\quad} x' \xrightarrow{a} y' \xrightarrow{r} z' \xrightarrow{e} \bar{u}'$$

In realtà tale automa non è ciò che desideriamo in quanto lo stato x' non accetta la stringa *andare*: infatti leggendo la prima a , dallo stato x' si transisce nello stato y' e, leggendo la n da questo stato non si può andare in alcuno stato. In altre parole nessuno stato è raggiungibile da x' con la stringa *andare* e pertanto *andare* non è accettata.

La stringa *andare* invece è accettata dallo stato x dell'automa in (8.4), come mostrato dal seguente walk da x a u .

$$x \xrightarrow{a} x \xrightarrow{n} x \xrightarrow{d} x \xrightarrow{a} y \xrightarrow{r} z \xrightarrow{e} u$$

Intuivamente x può scegliere come comportarsi con a : se tornare su se stesso oppure se transire in y . Quando legge la prima a di *andare*, resta su se stesso; quando legge la seconda, transisce in y .

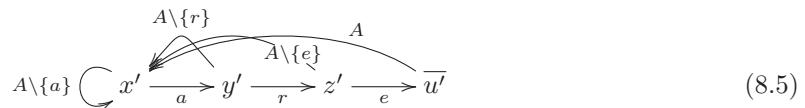
Le stesse considerazioni valgono per la stringa *mangiare* e , più in generale per qualsiasi stringa della forma $w_1\text{are}$ dove w_1 è un'arbitraria stringa su A^* . Infatti si ha che $(x, x) \in T_{w_1}$ e $(x, u) \in T_{\text{are}}$, e quindi $(x, u) \in T_{w_1\text{are}}$. Visto che $u \in F$, allora $w_1\text{are}$ è accettata dallo stato x . Questo dimostra che lo stato x accetta tutte le parole che terminano in *are*.

Esercizio 8.2.16. Dimostrare che per ogni alfabeto A , per ogni automa $\mathcal{A} = (S, T, F)$ sull'alfabeto A , per ogni stato $x \in S$, simbolo $a \in A$ e stringa $w \in A^*$ vale che:

1. $\varepsilon \in \langle\langle x \rangle\rangle$ se e solo se $x \in F$;
2. $aw \in \langle\langle x \rangle\rangle$ se e solo se esiste $y \in S$ tale che $(x, y) \in T_a$ e $w \in \langle\langle y \rangle\rangle$.

La costruzione dei sottoinsiemi

L'Esempio 8.2.15 mostra che lo stato x dell'automa non deterministico in (8.4) accetta il linguaggio di tutte le parole che finisco in *are* (cioè $\langle\langle x \rangle\rangle = \{w_1\text{are} \mid w_1 \in A^*\}$). In realtà, è possibile creare anche un automa deterministico con uno stato che accetta lo stesso linguaggio. Tale automa è illustrato di seguito.



Si noti innanzitutto che l'automa è deterministico, per ogni coppia $A \times S$ esiste esattamente uno stato successore. Inoltre, il linguaggio accettato da x' è il linguaggio di tutte le parole che terminano in *are*, cioè le parole della forma $w_1\text{are}$ per $w_1 \in A^*$. In altre parole, il linguaggio accettato da x nell'automa non deterministico in (8.4) è lo stesso del linguaggio accettato dallo stato x' nell'automa deterministico (8.5).

Osservazione 8.2.17. Visto che le macchine a stati reali sono tipicamente deterministiche l'automa in (8.5) può essere pensato come una implementazione realmente eseguibile. L'automa non deterministico in (8.4), invece non può essere realmente eseguito su una macchina a stati deterministiche, ma è comunque utile perché rappresenta una specifica, cioè una definizione formale del comportamento (nel caso degli automi, del linguaggio) desiderato.

Questo esempio suggerisce una domanda:

dato uno stato x di un automa qualsiasi (a stati finiti), è sempre possibile costruire un automa deterministico (a stati finiti) in cui uno stato accetta lo stesso linguaggio di x ?

La risposta è stata fornita da Michael Rabin e Dana Scott nel 1959 (questa scoperta ha valso loro il prestigioso premio Turing): sì, è sempre possibile trasformare un automa in uno deterministico utilizzando la *costruzione dei sottoinsiemi*. Inoltre se l'automa originale è a stati finiti, l'automa deterministico risultante da questa costruzione è anche esso a stati finiti. La costruzione è illustrata nella seguente definizione.

Definizione 8.2.18 (costruzione dei sottoinsiemi). *Dato un automa $\mathcal{A} = (S, T, F)$, l'automa deterministico*

$$\text{Det}(\mathcal{A}) = (\mathcal{P}(S), T^\sharp, F^\sharp)$$

è definito come segue:

- L'insieme degli stati è $\mathcal{P}(S)$, cioè l'insieme dei sottoinsiemi di stati dell'automa originale \mathcal{A} ;
- La funzione di transizione $T^\sharp: A \times \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ è definita per ogni $a \in A$ ed ogni sottoinsieme $X \subseteq S$ come

$$T^\sharp(a, X) = \{z \mid \text{esiste } x \in X \text{ tale che } ((a, x), z) \in T\};$$

- L'insieme degli stati finali $F^\sharp \subseteq \mathcal{P}(S)$ è definito come

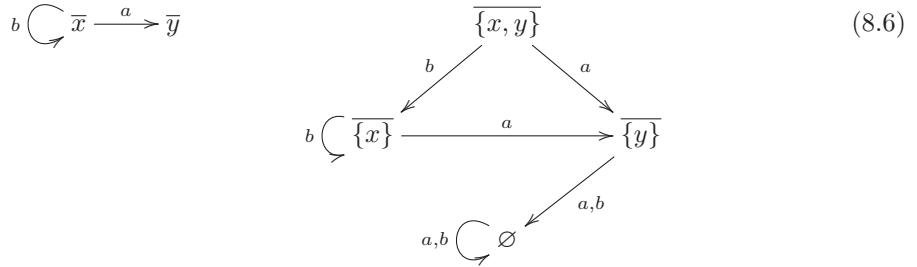
$$F^\sharp = \{X \subseteq S \mid \text{esiste } x \in X \text{ e } x \in F\}$$

Esempio 8.2.19. Prendiamo come alfabeto $A = \{a, b\}$. Consideriamo l'automa a sinistra in (8.2): $S = F = \{x\}$ e $T = \{((a, x), x)\}$. Pertanto l'insieme degli stati in $\text{Det}(\mathcal{A})$ è $\mathcal{P}(S) = \{\emptyset, \{x\}\}$, la funzione di transizione è $T^\sharp = \{((a, \{x\}), \{x\}), ((b, \{x\}), \emptyset), ((a, \emptyset), \emptyset), ((b, \emptyset), \emptyset)\}$ e $F^\sharp = \{\{x\}\}$. Di seguito illustriamo l'automa non deterministico originale (a sinistra) ed il corrispondente automa deterministico.



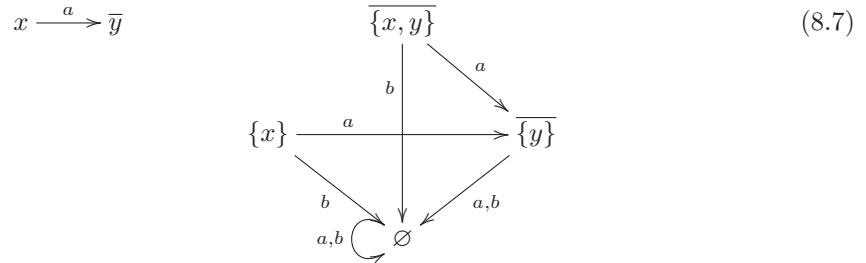
Si noti che per ogni simbolo dell'alfabeto $A = \{a, b\}$ c'è esattamente una transizione che esce da ogni stato. In particolare nell'automa determinizzato c'è una transizione da $\{x\}$ a \emptyset con il simbolo b , visto che $T^\sharp(b, \{x\})$ è, per definizione, l'insieme di tutti gli stati raggiungibili da x con b nell'automa originale, cioè nessuno. Esattamente per la stessa ragione, \emptyset ha sempre una transizione su se stesso, per ogni simbolo in A . Si noti inoltre che $\emptyset \notin F$, dal momento che non contiene alcuno stato.

Per \mathcal{A} l'automa al centro in (8.2), $\text{Det}(\mathcal{A})$ è riportato di seguito sulla destra.



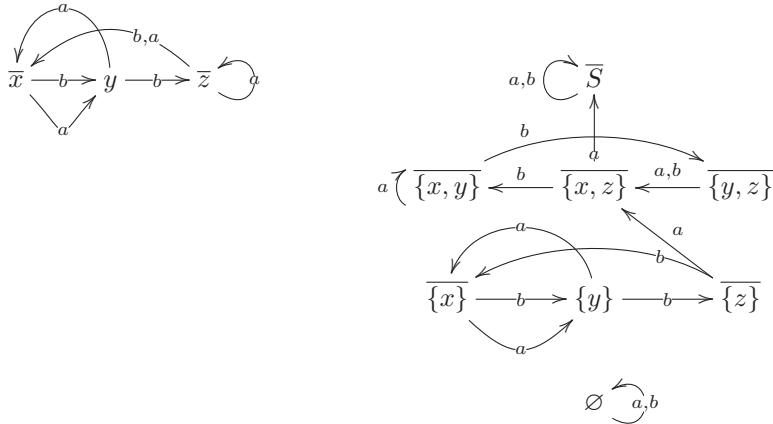
In questo automa, sono degne di nota le transizioni dello stato: $\{x, y\}$. Infatti $T^\sharp(a, \{x, y\}) = \{x\}$ poiché x è l'unico stato raggiungibile con a o da x o da y . Per la stessa ragione $T^\sharp(b, \{x, y\}) = \{y\}$ poiché y è l'unico stato raggiungibile con b o da x o da y .

Per \mathcal{A} l'automa a destra in (8.2), $\text{Det}(\mathcal{A})$ è riportato di seguito sulla destra.



Osserviamo che $\{x, y\}$ è di accettazione (cioè $\{x, y\} \in F^\sharp$) nonostante che nell'automa originale x non sia di accettazione (cioè $x \notin F$). Infatti è sufficiente che $y \in F$ per avere che $\{x, y\} \in F^\sharp$.

Per \mathcal{A} l'automa in (8.3), $\text{Det}(\mathcal{A})$ è riportato di seguito sulla destra.



Si noti che il non determinismo è stato rimosso: con il simbolo a , lo stato $\{z\}$ raggiunge esattamente uno stato $\{x, z\}$. Infatti per definizione $T^\sharp(a, \{z\}) = \{x, z\}$ è l'insieme di tutti gli stati raggiungibili da z con a . Inoltre, $T^\sharp(a, \{x, z\}) = \{x, y, z\} = S$, poiché x con a transisce in y e z con a transisce sia in x che in z .

La costruzione dei sottoinsiemi è interessante perché preserva il linguaggio accettato, come enunciato dal seguente teorema.

Teorema 8.2.20 (Teorema di corrispondenza). *Sia $\mathcal{A} = (S, T, F)$ un automa e $\text{Det}(\mathcal{A}) = (\mathcal{P}(S), T^\sharp, F^\sharp)$ il corrispondente automa deterministico. Per ogni stato $x \in S$ dell'automa \mathcal{A} vale che:*

$$\langle\langle x \rangle\rangle = \langle\langle \{x\} \rangle\rangle$$

8.3 Grammatiche Libere da Contesto

Nella sezione precedente abbiamo visto gli *automi*, certe macchine a stati che permettono di riconoscere le stringhe di un linguaggio. In questa sezione consideriamo le *grammatiche libere da contesto* che permettono di descrivere linguaggi, generando le stringhe in modo ricorsivo.

Un'importante applicazione delle grammatiche è la specifica della sintassi dei linguaggi di programmazione: le grammatiche costituiscono, infatti, una notazione comoda e concisa per descrivere la sintassi di un tipico linguaggio di programmazione e, alla fine di questa sezione, vedremo un semplice esempio a sostegno di questa affermazione. Nel prossimo capitolo vedremo come queste grammatiche permettono di specificare la sintassi del linguaggio logico.

Durante tutta questa sezione utilizzeremo come esempio il linguaggio delle *espressioni aritmetiche*. Le espressioni sono stringhe sull'alfabeto

$$A = \{+, -, \times, \div\} \cup \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}. \quad (8.8)$$

Sin dalle scuole elementari siamo abituati a manipolare queste espressioni e sappiamo bene che scrivere $5 + \times \div$ non ha alcun significato, mentre scrivere $5 \times 4 + 1$ è corretto. Dobbiamo adesso pensare che sia $5 + \times \div$ che $5 \times 4 + 1$ sono entrambe stringhe sull'alfabeto A in (8.8), ma solo la seconda appartiene al linguaggio delle espressioni aritmetiche. Per esprimere questo linguaggio in modo preciso e sistematico (in modo che un computer lo possa riconoscere) utilizziamo la seguente grammatica libera da contesto.

$$\begin{aligned} \langle\langle ExA \rangle\rangle &\rightsquigarrow \langle\langle Num \rangle\rangle \\ \langle\langle ExA \rangle\rangle &\rightsquigarrow \langle\langle ExA \rangle\rangle + \langle\langle ExA \rangle\rangle \\ \langle\langle ExA \rangle\rangle &\rightsquigarrow \langle\langle ExA \rangle\rangle - \langle\langle ExA \rangle\rangle \\ \langle\langle ExA \rangle\rangle &\rightsquigarrow \langle\langle ExA \rangle\rangle \times \langle\langle ExA \rangle\rangle \\ \langle\langle ExA \rangle\rangle &\rightsquigarrow \langle\langle ExA \rangle\rangle \div \langle\langle ExA \rangle\rangle \end{aligned} \quad (8.9)$$

8. Linguaggi Formali

Intuitivamente, la grammatica ci dice che un'espressione aritmetica ($\langle ExA \rangle$) o è un numero ($\langle Num \rangle$) oppure la somma (+), la sottrazione (-), la moltiplicazione (\times), la divisione (\div) di due espressioni aritmetiche.

Introduciamo adesso un po' di termini tecnici.

- I simboli tra parentesi angolate, come $\langle ExA \rangle$ o $\langle Num \rangle$, vengono chiamati *categorie sintattiche* o *simboli non-terminali*. Intuitivamente queste denotano linguaggi. Vedremo dopo, la definizione formale di tali linguaggi.
- Il simbolo \rightsquigarrow è un *metasimbolo*, che si legge “può essere”.
- Gli altri simboli $+, -, \times, \div$ sono i *simboli terminali*, cioè i simboli dell'alfabeto del linguaggio generato (nel nostro caso A in (8.8)).

Nella grammatica in (8.9) non è specificato cosa è un numero (rappresentato dalla categoria sintattica $\langle Num \rangle$). Per completarla, consideriamo la seguente grammatica che specifica che un numero è una sequenza di cifre (categoria sintattica $\langle Cif \rangle$), cioè una sequenza di simboli in $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

$$\begin{aligned}
 \langle Cif \rangle &\rightsquigarrow 0 \\
 \langle Cif \rangle &\rightsquigarrow 1 \\
 \langle Cif \rangle &\rightsquigarrow 2 \\
 \langle Cif \rangle &\rightsquigarrow 3 \\
 \langle Cif \rangle &\rightsquigarrow 4 \\
 \langle Cif \rangle &\rightsquigarrow 5 \\
 \langle Cif \rangle &\rightsquigarrow 6 \\
 \langle Cif \rangle &\rightsquigarrow 7 \\
 \langle Cif \rangle &\rightsquigarrow 8 \\
 \langle Cif \rangle &\rightsquigarrow 9 \\
 \langle Num \rangle &\rightsquigarrow \langle Cif \rangle \\
 \langle Num \rangle &\rightsquigarrow \langle Num \rangle \langle Cif \rangle \\
 \langle ExA \rangle &\rightsquigarrow \langle Num \rangle \\
 \langle ExA \rangle &\rightsquigarrow \langle ExA \rangle + \langle ExA \rangle \\
 \langle ExA \rangle &\rightsquigarrow \langle ExA \rangle - \langle ExA \rangle \\
 \langle ExA \rangle &\rightsquigarrow \langle ExA \rangle \times \langle ExA \rangle \\
 \langle ExA \rangle &\rightsquigarrow \langle ExA \rangle \div \langle ExA \rangle
 \end{aligned} \tag{8.10}$$

Continuiamo ad introdurre un po' di terminologia. Nelle grammatiche, ogni riga viene detta una *produzione*. Per esempio

$$\langle ExA \rangle \rightsquigarrow \langle ExA \rangle - \langle ExA \rangle$$

è una produzione. Per essere più precisi, una produzione consiste di un simbolo non terminale a sinistra di \rightsquigarrow e, alla sua destra, una sequenza di simboli terminali e non. Il simbolo non terminale a sinistra di \rightsquigarrow viene detto la *testa* della produzione, mentre la sequenza di simboli a destra di \rightsquigarrow viene detta il *corpo* della produzione.

Se S è l'insieme dei simboli non terminali ed A è l'insieme dei simboli terminali, allora a destra di \rightsquigarrow c'è un elemento dell'insieme $(S \cup A)^*$ (si assume sempre che gli insiemi A ed S siano disgiunti).

Definizione 8.3.1 (grammatica). *Una grammatica libera da contesto \mathcal{G} sull'alfabeto A consiste di una coppia (S, P) dove:*

- S è l'insieme dei simboli non terminali (S è disgiunto da A);

- P è un insieme di produzioni: ogni produzione ha la forma

$$\langle X \rangle \rightsquigarrow w$$

dove $\langle X \rangle \in S$ e $w \in (A \cup S)^*$. Si noti che w può essere anche la stringa vuota ε .

Una grammatica \mathcal{G} si dice finita se l'insieme dei simboli terminali S è finito.

Esercizio 8.3.2. Fissiamo un alfabeto A e un insieme S di simboli non terminali. Esiste una biiezione tra l'insieme di tutte le produzioni su A e S e l'insieme di tutte le relazioni tra S e $(S \cup A)^*$?

Per alleggerire la notazione, si possono rappresentare più produzioni su una sola riga: si introduce un ulteriore metasimbolo $|$ che si legge “oppure”, e le produzioni

$$\begin{aligned} \langle X \rangle &\rightsquigarrow w_1 \\ \langle X \rangle &\rightsquigarrow w_2 \\ &\dots \\ \langle X \rangle &\rightsquigarrow w_n \end{aligned}$$

si rappresentano come

$$\langle X \rangle \rightsquigarrow w_1 | w_2 | \dots | w_n.$$

Per esempio, la grammatica in (8.10) può essere espressa in modo conciso come segue.

$$\begin{aligned} \langle Cif \rangle &\rightsquigarrow 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 \\ \langle Num \rangle &\rightsquigarrow \langle Cif \rangle | \langle Num \rangle \langle Cif \rangle \\ \langle ExA \rangle &\rightsquigarrow \langle Num \rangle | \langle ExA \rangle + \langle ExA \rangle | \langle ExA \rangle - \langle ExA \rangle | \\ &\quad \langle ExA \rangle \times \langle ExA \rangle | \langle ExA \rangle \div \langle ExA \rangle \end{aligned} \tag{8.11}$$

Il significato non cambia: un'espressione aritmetica può essere un numero, oppure una somma, oppure una sottrazione oppure ...

Osservazione 8.3.3. La sintassi di molti linguaggi formali, come linguaggi di programmazione, linguaggi di specifica, protocolli di rete, linguaggi logici è definita con la notazione utilizzata sopra. Talvolta, il metasimbolo \rightsquigarrow è rimpiazzato da $::=$ e, in certi contesti, i simboli non terminali non si trovano all'interno delle parentesi angolate $\langle \cdot \rangle$.

Questo tipo di definizione è spesso chiamata Backus Normal Form o Backus-Naur Form, o con l'acronimo BNF, perché fu proposta da John Backus della definizione del linguaggio di programmazione ALGOL.

Continuiamo ad arricchire le espressioni aritmetiche ed illustriamo come rappresentare espressioni aritmetiche con variabili, come per esempio $x + 3 \times y$. Innanzitutto estendiamo l'alfabeto A in modo da poter rappresentare le variabili con degli identificatori (stringhe di caratteri $\{a, b, c, \dots, z\}$)

$$A = \{+, -, \times, \div\} \cup \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \cup \{a, b, c, \dots, z\}. \tag{8.12}$$

Poi estendiamo la precedente grammatica con la categoria sintattica degli identificatori ($\langle Ide \rangle$) che intuitivamente contiene tutte le stringhe formate dai caratteri.

$$\begin{aligned} \langle Cif \rangle &\rightsquigarrow 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 \\ \langle Num \rangle &\rightsquigarrow \langle Cif \rangle | \langle Num \rangle \langle Cif \rangle \\ \langle ExA \rangle &\rightsquigarrow \langle Num \rangle | \langle ExA \rangle + \langle ExA \rangle | \langle ExA \rangle - \langle ExA \rangle | \\ &\quad \langle ExA \rangle \times \langle ExA \rangle | \langle ExA \rangle \div \langle ExA \rangle | \\ &\quad \langle Ide \rangle \\ \langle Ide \rangle &\rightsquigarrow \langle Car \rangle | \langle Ide \rangle \langle Car \rangle \\ \langle Car \rangle &\rightsquigarrow a | b | \dots | z \end{aligned} \tag{8.13}$$

Si noti che in questa grammatica un'espressione aritmetica può essere anche un identificatore. Abbiamo quindi che un'espressione aritmetica può essere per esempio $x + 3 \times y$.

La grammatica in (8.13) può essere resa più compatta aggiungendo la categoria sintattica degli operatori aritmetici ($\langle OpA \rangle$).

$$\begin{aligned}
 \langle OpA \rangle &\rightsquigarrow + \mid - \mid \times \mid \div \\
 \langle Cif \rangle &\rightsquigarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
 \langle Num \rangle &\rightsquigarrow \langle Cif \rangle \mid \langle Num \rangle \langle Cif \rangle \\
 \langle ExA \rangle &\rightsquigarrow \langle Num \rangle \mid \langle ExA \rangle \langle OpA \rangle \langle ExA \rangle \mid \langle Ide \rangle \\
 \langle Ide \rangle &\rightsquigarrow \langle Car \rangle \mid \langle Ide \rangle \langle Car \rangle \\
 \langle Car \rangle &\rightsquigarrow a \mid b \mid \dots \mid z
 \end{aligned} \tag{8.14}$$

Come abbiamo anticipato, ogni categoria sintattica denota un linguaggio. Al fine di spiegare come tale linguaggio è generato, dobbiamo introdurre il concetto di albero di derivazione sintattica.

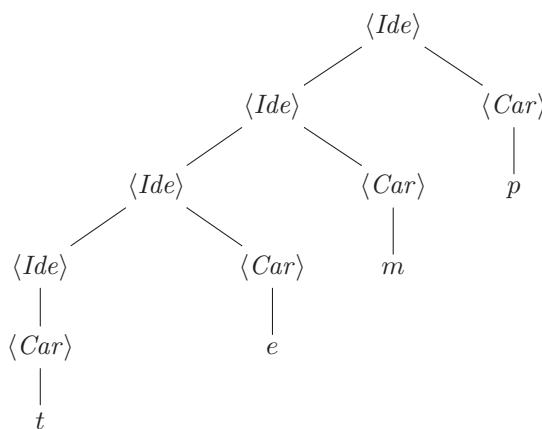
Definizione 8.3.4 (albero di derivazione sintattica, parse tree). *Sia (S, P) una grammatica libera da contesto sull'alfabeto A . Un ALBERO DI DERIVAZIONE SINTATTICA (o PARSE TREE) è un albero radicato dove:*

- *ogni nodo interno è etichettato da un simbolo non terminale (cioè un simbolo in S);*
- *ogni foglia è etichettata da un simbolo terminale (cioè un simbolo in A) o dalla stringa vuota;*
- *ogni nodo interno v rappresenta l'applicazione di una produzione, ovvero deve esistere una produzione tale che:*
 - (a) *l'etichetta del nodo v è la testa della produzione;*
 - (b) *l'etichette dei figli di v , da sinistra a destra, formano il corpo della produzione.*

Sia $w \in A^$ la stringa ottenuta leggendo da sinistra a destra i simboli terminali che etichettano le foglie dell'albero (la stringa vuota si ignora). Si dice che w è la stringa testimoniata dall'albero o che l'albero è un albero di derivazione (sintattica) di w .*

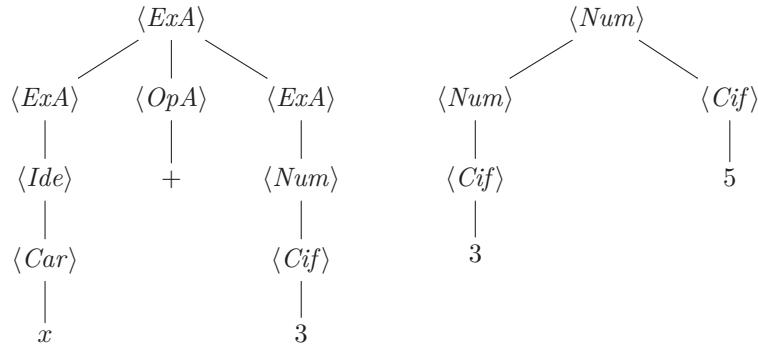
Osservazione 8.3.5. *Si noti che, utilizzando la terminologia del Capitolo 5 un parse tree è un albero ordinale: l'ordine dei figli conta.*

Per esempio, data la grammatica in (8.14), un albero di derivazione per la stringa $temp$ è rappresentato di seguito.



Si noti che ogni nodo interno è etichettato da un simbolo non terminale ($\langle Ide \rangle$ o $\langle Car \rangle$) ed ogni foglia è etichettata da un simbolo terminale. Leggendo le foglie da sinistra a destra si ha proprio la stringa $temp$ (che potrebbe essere l'identificatore di una variabile temporanea).

Gli alberi di derivazione per le stringhe $x + 3$ e 35 sono rappresentati di seguito rispettivamente sulla sinistra e sulla destra.



Esercizio 8.3.6. Data la grammatica libera da contesto in (8.14), esiste un albero di derivazione per $5 + \times \div ?$?

Osservazione 8.3.7. Un programma nella maggior parte dei linguaggi di programmazione è una stringa sull’alfabeto ASCII. L’interprete o il compilatore del linguaggio trasformano sempre questa stringa nell’albero di derivazione, per poi eseguirlo o tradurlo. Infatti la semantica dei linguaggi di programmazione è solitamente definita in modo ricorsivo, sugli alberi e non sulle stringhe.

Lo stesso avviene, in modo implicito, quando manipoliamo espressioni matematiche o logiche. Il bravo informatico, quando riflette su un programma o su una espressione, deve sempre chiedersi come prima cosa quale sia l’albero corrispondente.

A questo punto possiamo definire formalmente qual è il linguaggio generato da una categoria sintattica di una grammatica.

Definizione 8.3.8 (linguaggio generato da non terminale). Sia (S, P) una grammatica libera da contesto sull’alfabeto A . Sia $\langle X \rangle \in S$ un simbolo non terminale. Il linguaggio generato da $\langle X \rangle$ è l’insieme delle stringhe $w \in A^*$ tali che esiste un albero di derivazione per w avente come radice un nodo etichettato con $\langle X \rangle$. Denoteremo questo linguaggio con $\langle\langle X \rangle\rangle$.

Per esempio, la stringa $x + 3$ è un’espressione aritmetica, cioè appartiene al linguaggio generato da $\langle ExA \rangle$; la stringa 35 è un numero, cioè appartiene al linguaggio generato da $\langle Num \rangle$; la stringa $temp$ è un identificatore (di variabile), cioè appartiene al linguaggio generato da $\langle Ide \rangle$;

Esercizio 8.3.9. Data la grammatica libera da contesto in (8.14), la stringa 35 appartiene al linguaggio generato da $\langle ExA \rangle$? E $temp$?

Esempio 8.3.10. Ricordiamo dall’Osservazione 8.1.16, che per ogni stringa $w \in A^*$, w^n è la concatenazione della stringa w n -volte. Prendiamo adesso l’alfabeto $A = \{a, b\}$ e consideriamo il linguaggio $L \subseteq A^*$ definito come

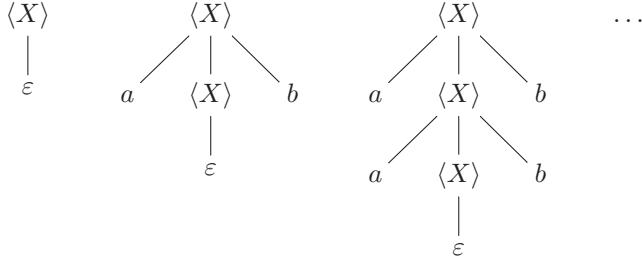
$$L = \{a^n b^n \mid n \in \mathbb{N}\}.$$

Per esempio si ha che $\varepsilon \in L$ (si prenda $n = 0$), $ab \in L$ (si prenda $n = 1$), $aabb \in L$ (si prenda $n = 2$) e $aaabbb \in L$ (si prenda $n = 3$). Si consideri adesso la seguente grammatica.

$$\langle X \rangle \rightsquigarrow \varepsilon \mid a \langle X \rangle b \tag{8.15}$$

È facile convincersi che il linguaggio generato da $\langle X \rangle$ è esattamente L : infatti per ogni $n \in \mathbb{N}$, la stringa $a^n b^n$ è testimoniata da un albero di derivazione analogo a quelli illustrati di seguito per

$n = 0, n = 1$ e $n = 2$.



Viceversa, le stringhe $a^n b^n$ sono le uniche generabili da tale grammatica: tutti gli alberi di derivazione (e quindi tutte le stringhe generate) hanno esattamente la forma degli alberi di derivazione illustrati sopra.

Osservazione 8.3.11. È importante notare che anche la definizione di una grammatica può, in qualche modo, essere vista come una definizione induttiva. Consideriamo ad esempio, la grammatica in (8.15): questa definisce $\langle\langle X \rangle\rangle \subseteq A^*$ induttivamente come:

1. [CLAUSOLA BASE] $\varepsilon \in \langle\langle X \rangle\rangle$.
2. [CLAUSOLA INDUTTIVA] Se $w \in \langle\langle X \rangle\rangle$, allora $awb \in \langle\langle X \rangle\rangle$.

Esempio 8.3.12. Data una arbitraria segnatura \mathcal{F} , è possibile definire una grammatica che genera tutti gli \mathcal{F} -termini (Definizione 7.4.3). Per ogni $n \in \mathbb{N}$, definiamo una categoria sintattica $\langle Fide_n \rangle$ tale che $\langle\langle Fide_n \rangle\rangle = \mathcal{F}_n$. Questo può essere facilmente ottenuto con una produzione

$$\langle Fide_n \rangle \rightsquigarrow f$$

per ogni $f \in \mathcal{F}_n$. La categoria sintattica degli \mathcal{F} -termini, $\langle Term \rangle$, è definita come

$$\langle Term \rangle \rightsquigarrow \langle Fide_0 \rangle \mid \langle Fide_n \rangle (\overbrace{\langle Term \rangle, \dots, \langle Term \rangle}^n), n \in \mathbb{N}^+ \quad (8.16)$$

Confrontando tale grammatica con la Definizione 7.4.3 è facile convincersi che $\langle\langle Term \rangle\rangle = \mathcal{F}Term$.

Per un esempio, si consideri la segnatura dell'Esempio 7.4.2. La grammatica corrispondente è

$$\begin{aligned} \langle Fide_0 \rangle &\rightsquigarrow a \mid b \\ \langle Fide_1 \rangle &\rightsquigarrow f \\ \langle Fide_2 \rangle &\rightsquigarrow g \\ \langle Term \rangle &\rightsquigarrow \langle Fide_0 \rangle \mid \langle Fide_1 \rangle (\langle Term \rangle) \mid \langle Fide_2 \rangle (\langle Term \rangle, \langle Term \rangle) \end{aligned}$$

Utilizzeremo le produzioni in (8.16) per definire la sintassi della Logica dei Predicati nella Sezione 9.5. In tale contesto, così come in molte altre situazioni, è conveniente partizionare l'insieme \mathcal{F}_0 dei simboli di arietà 0, in simboli di costante e simboli di variabile. Tipicamente, l'insieme dei simboli di costante è denotato da \mathcal{C} e quello delle variabili da \mathcal{V} . Al posto della categoria sintattica $\langle Fide_0 \rangle$ utilizzeremo le categorie $\langle Const \rangle$ e $\langle Var \rangle$ per, rispettivamente, \mathcal{C} e \mathcal{V} , cioè, $\langle\langle Const \rangle\rangle = \mathcal{C}$ e $\langle\langle Var \rangle\rangle = \mathcal{V}$. Questo significa avere le produzioni

$$\langle Const \rangle \rightsquigarrow c \qquad \langle Var \rangle \rightsquigarrow x$$

per ogni simbolo di costante $c \in \mathcal{C}$ ed ogni simbolo di variabile $x \in \mathcal{V}$. Di conseguenza, le produzioni in (8.16) divengono le seguenti.

$$\langle Term \rangle \rightsquigarrow \langle Var \rangle \mid \langle Const \rangle \mid \langle Fide_n \rangle (\overbrace{\langle Term \rangle, \dots, \langle Term \rangle}^n), n \in \mathbb{N}^+ \quad (8.17)$$

Esempio 8.3.13. Ci sono molte grammatiche che non fanno uso dell'induzione, ma sono comunque interessanti perché specificano formati di stringhe che possono essere utili in svariati contesti. Un esempio tipico, ben noto al lettore, è la grammatica degli indirizzi postali.

$$\begin{aligned} \langle \text{IndirizzoPostale} \rangle &\rightsquigarrow \langle \text{Destinatario} \rangle \langle \text{EOL} \rangle \langle \text{Indirizzo} \rangle \langle \text{EOL} \rangle \langle \text{Località} \rangle \\ \langle \text{Destinatario} \rangle &\rightsquigarrow \langle \text{Titolo} \rangle \langle \text{Nome} \rangle \langle \text{Cognome} \rangle \\ \langle \text{Indirizzo} \rangle &\rightsquigarrow \langle \text{Via} \rangle \langle \text{NumeroCivico} \rangle \\ \langle \text{Località} \rangle &\rightsquigarrow \langle \text{CAP} \rangle \langle \text{Comune} \rangle (\langle \text{Provincia} \rangle) \\ \langle \text{CAP} \rangle &\rightsquigarrow \langle \text{Cif} \rangle \end{aligned}$$

La categoria sintattica $\langle \text{EOL} \rangle$ stà per un qualche carattere di fine linea (dall'inglese *End Of Line*)

Esercizio 8.3.14. Dare una grammatica libera da contesto che genera il linguaggio $\{a^n b^n \mid n \in \mathbb{N}^+\}$.

Esercizio 8.3.15. Dare una grammatica libera da contesto che genera il linguaggio $\{a^n c b^n \mid n \in \mathbb{N}\}$.

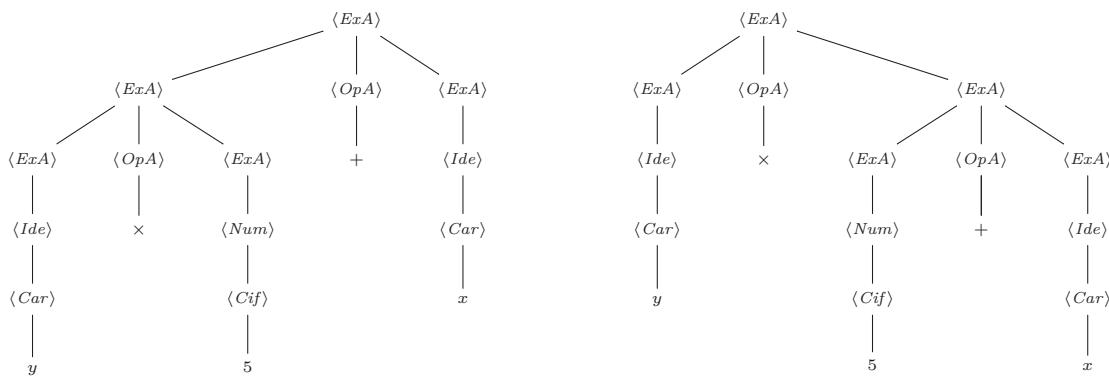
Esercizio 8.3.16. Dare una grammatica libera da contesto che genera il linguaggio $\{a^{2n} b^n \mid n \in \mathbb{N}\}$.

Esercizio 8.3.17. Data la grammatica libera da contesto in (8.9), qual'è il linguaggio generato da $\langle \text{ExA} \rangle$? E da $\langle \text{Num} \rangle$?

Esercizio 8.3.18. Data la grammatica libera da contesto in (8.14), si disegnino, se esistono, uno o più alberi di derivazione per le stringhe $y \times 5 + x$, $x + y \div 0$ e $-5 \div x$.

Ambiguità

Le grammatiche libere da contesto possono essere *ambigue* in un senso ben preciso che illustriamo di seguito. Continuiamo ad utilizzare come esempio il linguaggio delle espressioni aritmetiche e la grammatica in (8.14). Consideriamo la stringa $y \times 5 + x$. Ci sono due diversi alberi di derivazione per questa stringa che illustriamo di seguito.



Intuitivamente ai due alberi di derivazione corrispondono due diverse espressioni con parentesi: quello di sinistra corrisponde a $(y \times 5) + x$ e quello di destra a $y \times (5 + x)$. Si noti che la semantica delle due espressioni è molto diversa: per esempio per $x = 3$ e $y = 2$, la prima vale 13, mentre la seconda vale 16. Per le espressioni aritmetiche, si risolve questo problema introducendo una precedenza sugli operatori: si dice per esempio che \times lega più di $+$. Quindi l'espressione aritmetica $y \times 5 + x$ viene considerata come $(y \times 5) + x$. Per esprimere l'espressione $y \times (5 + x)$ è necessario quindi introdurre le parentesi tonde. Estendiamo prima l'alfabeto con i simboli per le parentesi

$$A = \{+, -, \times, \div\} \cup \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \cup \{a, b, c, \dots, z\} \cup \{(,)\}. \quad (8.18)$$

8. Linguaggi Formali

e poi la grammatica come segue.

$$\begin{aligned}
 \langle OpA \rangle &\rightsquigarrow + \mid - \mid \times \mid \div \\
 \langle Cif \rangle &\rightsquigarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
 \langle Num \rangle &\rightsquigarrow \langle Cif \rangle \mid \langle Num \rangle \langle Cif \rangle \\
 \langle ExA \rangle &\rightsquigarrow (\langle ExA \rangle) \mid \langle Num \rangle \mid \langle ExA \rangle \langle OpA \rangle \langle ExA \rangle \mid \langle Ide \rangle \\
 \langle Ide \rangle &\rightsquigarrow \langle Car \rangle \mid \langle Ide \rangle \langle Car \rangle \\
 \langle Car \rangle &\rightsquigarrow a \mid b \mid \dots \mid z
 \end{aligned} \tag{8.19}$$

Osservazione 8.3.19. Un problema analogo avviene anche con il linguaggio naturale. Per esempio, nella lingua italiana, la frase

Gianni vede la vecchia col binocolo

può essere interpretata in due modi: il binocolo appartiene alla vecchia oppure a Gianni.

Gianni vede (la vecchia col binocolo)

oppure

(Gianni vede la vecchia) col binocolo

Tutto ciò ci fa riflettere sull'espressività degli alberi: in un certo senso, gli alberi contengono più "informazione" delle stringhe. Ma allora perché storicamente l'umanità ha sviluppato tantissimi linguaggi scritti con stringhe e non con alberi?

A questo proposito è importante ricordare che, nel corso degli anni, gli informatici hanno sviluppato tanti linguaggi (di programmazione o di specifica) la cui sintassi sono alberi (come html o xml) o, addirittura grafi (come uml o tensor flow). L'idea dietro i linguaggi grafici è che, come i diagrammi di Eulero-Venn o i diagrammi per le relazioni, la sintassi grafica/diagrammatica è in qualche senso più intuitiva. Forse, un giorno, programmeremo solo disegnando diagrammi e non digitando caratteri su una tastiera...

Definizione 8.3.20 (stringa e grammatica ambigua). Data una grammatica libera da contesto \mathcal{G} , una stringa $w \in A^*$ è detta AMBIGUA per \mathcal{G} se esistono almeno due diversi alberi di derivazione che testimoniano w e che hanno la stessa etichetta nella radice.

Una grammatica \mathcal{G} è AMBIGUA se esiste almeno una stringa ambigua per \mathcal{G} .

Per esempio, la grammatica in (8.19) è ambigua. Esistono delle tecniche molto eleganti per disambiguare le grammatiche, ma queste vanno al di là dello scopo di questo corso.

La grammatica di un semplice linguaggio imperativo

Mostriamo adesso la grammatica libera da contesto per un semplice linguaggio di programmazione. L'idea è di riutilizzare quanto abbiamo fatto fino ad ora per le espressioni aritmetiche ed estenderlo con le categorie sintattiche per comandi ($\langle Com \rangle$) ed espressioni booleane ($\langle ExB \rangle$).

Iniziamo con le espressioni booleane. Estendiamo prima l'alfabeto

$$A = \{+, -, \times, \div\} \cup \{0, \dots, 9\} \cup \{a, b, c, \dots, z\} \cup \{(,)\} \cup \{\wedge, \vee, !, \}\cup \{>, \geq, =, <, \leq\} \tag{8.20}$$

e poi diamo la seguente grammatica.

$$\begin{aligned}
 \langle ExB \rangle &\rightsquigarrow (\langle ExB \rangle) \mid \langle CoB \rangle \mid \langle ExB \rangle \langle OpB \rangle \langle ExB \rangle \mid !\langle ExB \rangle \mid \\
 &\quad \langle Ide \rangle \mid \langle ExA \rangle \langle OpC \rangle \langle ExA \rangle \\
 \langle OpC \rangle &\rightsquigarrow > \mid \geq \mid < \mid \leq \mid = \mid ! = \\
 \langle OpB \rangle &\rightsquigarrow \wedge \mid \vee \\
 \langle CoB \rangle &\rightsquigarrow true \mid false
 \end{aligned} \tag{8.21}$$

In altre parole un'espressione booleana può essere un'espressione booleana tra parentesi tonde, oppure una costante booleana (categoria sintattica $\langle CoB \rangle$), cioè le stringhe *true* e *false*, oppure l'and (\wedge) o l'or (\vee) di due espressioni booleane, il not (!) di un'espressione booleana, una variabile, oppure il *confronto* di due espressioni aritmetiche. Tale confronto è fatto attraverso gli operatori di confronto (categoria sintattica $\langle OpC \rangle$) che sono maggiore ($>$), maggiore o uguale (\geq), minore ($<$), minore o uguale (\leq), uguale ($=$) o diverso (\neq).

Per i comandi, estendiamo ulteriormente l'alfabeto

$$A = \{+, -, \times, \div\} \cup \{0, \dots, 9\} \cup \{a, b, c, \dots, z\} \cup \{(,)\} \cup \{\wedge, \vee, !, \}\cup \{>, \geq, =, <, \leq\} \cup \{; , :, \{, \}\} \quad (8.22)$$

e consideriamo la seguente grammatica.

$$\begin{array}{lll} \langle Com \rangle & \rightsquigarrow & skip \mid \\ & & \qquad \qquad \qquad \text{(Comando vuoto)} \\ \langle Ide \rangle & := & \langle ExA \rangle \mid \\ & & \qquad \qquad \qquad \text{(Assegnamento)} \\ \langle Com \rangle ; \langle Com \rangle & | & \\ & & \qquad \qquad \qquad \text{(Composizione)} \\ if(\langle ExB \rangle) \ then \ \{ \langle Com \rangle \} \ else \ \{ \langle Com \rangle \} & | & \text{(Branching condizionale)} \\ while(\langle ExB \rangle) \ do \ \{ \langle Com \rangle \} & | & \text{(Iterazione)} \end{array} \quad (8.23)$$

Unendo la grammatica per le espressioni aritmetiche con quelle delle espressioni booleane e dei comandi, otteniamo la grammatica di un semplice linguaggio di programmazione.

$$\begin{array}{lll} \langle OpA \rangle & \rightsquigarrow & + \mid - \mid \times \mid \div \\ \langle OpB \rangle & \rightsquigarrow & \wedge \mid \vee \\ \langle OpC \rangle & \rightsquigarrow & > \mid \geq \mid < \mid \leq \mid = \mid ! = \\ \langle Cif \rangle & \rightsquigarrow & 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\ \langle CoB \rangle & \rightsquigarrow & true \mid false \\ \langle Car \rangle & \rightsquigarrow & a \mid b \mid \dots \mid z \\ \langle Ide \rangle & \rightsquigarrow & \langle Car \rangle \mid \langle Ide \rangle \langle Car \rangle \\ \langle Num \rangle & \rightsquigarrow & \langle Cif \rangle \mid \langle Num \rangle \langle Cif \rangle \\ \langle ExA \rangle & \rightsquigarrow & (\langle ExA \rangle) \mid \langle Num \rangle \mid \langle ExA \rangle \langle OpA \rangle \langle ExA \rangle \mid \langle Ide \rangle \\ \langle ExB \rangle & \rightsquigarrow & (\langle ExB \rangle) \mid \langle CoB \rangle \mid \langle ExB \rangle \langle OpB \rangle \langle ExB \rangle \mid !\langle ExB \rangle \mid \\ & & \langle Ide \rangle \mid \langle ExA \rangle \langle OpC \rangle \langle ExA \rangle \\ \langle Com \rangle & \rightsquigarrow & skip \mid \\ & & \langle Ide \rangle := \langle ExA \rangle \mid \\ & & \langle Com \rangle ; \langle Com \rangle \mid \\ if(\langle ExB \rangle) \ then \ \{ \langle Com \rangle \} \ else \ \{ \langle Com \rangle \} & | & \\ while(\langle ExB \rangle) \ do \ \{ \langle Com \rangle \} & | & \end{array} \quad (8.24)$$

Esercizio 8.3.21. Data la grammatica libera da contesto in (8.24), si disegnino, se esistono, uno o più alberi di derivazione per le stringhe

1. $while(true) do \{skip\};$
2. $if(x < 0) then \{x := 3\} else \{skip\};$
3. $x := 3; y := 5; z := x + y.$

Esercizio 8.3.22. Data la grammatica libera da contesto in (8.24), la stringa $5 + 4 > 3$ è una espressione aritmetica? Cioè, appartiene al linguaggio generato da $\langle ExA \rangle$?

8.4 Uno studio comparativo di automi e grammatiche

Nella Sezione 8.1, abbiamo introdotto i linguaggi come sottoinsiemi di A^* . Abbiamo visto nella Sezione 8.2 che alcuni linguaggi possono essere riconosciuti da automi, mentre nella Sezione 8.3 che alcuni linguaggi possono essere generati da grammatiche libere da contesto. In particolare, abbiamo mostrato che le grammatiche possono essere utilizzate per la specifica della sintassi dei linguaggi di programmazione. Vale lo stesso per gli automi? Cioè, possono gli automi a stati finiti riconoscere i linguaggi di programmazione?

Come da buona pratica scientifica, per dare risposta a questa domanda, ci poniamo in una prospettiva più ampia e ci chiediamo:

1. Tutti i linguaggi generati da grammatiche finite sono anche riconoscibili da automi a stati finiti?

Se la risposta a questa domanda fosse affermativa, allora sapremmo che, in particolare, i linguaggi di programmazione sono riconoscibili da automi (visto che sono generati da grammatiche). Prima di provare a rispondere a questa domanda, ci interroghiamo sulla domanda opposta:

2. Tutti i linguaggi riconoscibili da automi a stati finiti sono generati da grammatiche finite?

La risposta alla seconda domanda è positiva. Per dimostrarlo, illustriamo una costruzione che prende come input un automa e restituisce una grammatica che riconosce lo stesso linguaggio.

L'idea è molto semplice. Dato un automa $\mathcal{A} = (S, T, F)$ definiamo una grammatica $\mathcal{G}_{\mathcal{A}} = (S_{\mathcal{A}}, P_{\mathcal{A}})$ dove l'insieme dei simboli non terminali $S_{\mathcal{A}}$ è esattamente l'insieme contenente tutti e soli i simboli $\langle x \rangle$ per ogni stato dell'automa $x \in S$. Più precisamente

$$S_{\mathcal{A}} = \{\langle x \rangle \mid x \in S\}. \quad (8.25)$$

Ad esempio, prendendo come \mathcal{A} l'automa in (8.4), la grammatica $\mathcal{G}_{\mathcal{A}}$ corrispondente ha come insieme di simboli non terminali $S_{\mathcal{A}} = \{\langle x \rangle, \langle y \rangle, \langle z \rangle, \langle u \rangle\}$.

Le produzioni in $P_{\mathcal{A}}$ sono costruite a partire dalla relazione di transizione T e l'insieme degli stati finali F . In particolare, per ogni stato finale $x \in F$, c'è una produzione

$$\langle x \rangle \rightsquigarrow \varepsilon$$

in $P_{\mathcal{A}}$. Per ogni transizione $((a, x), y) \in T$, c'è la produzione

$$\langle x \rangle \rightsquigarrow a \langle y \rangle.$$

In sintesi, l'insieme delle produzioni è

$$P_{\mathcal{A}} = \{\langle x \rangle \rightsquigarrow \varepsilon \mid x \in F\} \cup \{\langle x \rangle \rightsquigarrow a \langle y \rangle \mid ((a, x), y) \in T\}. \quad (8.26)$$

Ad esempio, prendendo come \mathcal{A} l'automa in (8.4), la grammatica $\mathcal{G}_{\mathcal{A}}$ ha come produzioni

$$\begin{aligned} \langle u \rangle &\rightsquigarrow \varepsilon \\ \langle z \rangle &\rightsquigarrow e \langle u \rangle \\ \langle y \rangle &\rightsquigarrow r \langle z \rangle \\ \langle x \rangle &\rightsquigarrow a \langle y \rangle \\ \langle x \rangle &\rightsquigarrow a \langle x \rangle \mid b \langle x \rangle \mid \dots \mid z \langle x \rangle \end{aligned}$$

Tale costruzione è riassunta nella seguente definizione.

Definizione 8.4.1 (grammatica corrispondente a automa). *Dato un qualsiasi automa $\mathcal{A} = (S, T, F)$, la corrispondente grammatica libera da contesto è $\mathcal{G}_{\mathcal{A}} = (S_{\mathcal{A}}, P_{\mathcal{A}})$ dove, l'insieme dei simboli non terminali $S_{\mathcal{A}}$ e l'insieme delle produzioni $P_{\mathcal{A}}$ sono definiti rispettivamente come in (8.25) e (8.26).*

Per maggiore chiarezza, illustriamo un ulteriore esempio.

Esempio 8.4.2. La grammatica corrispondente all'automa in (8.3) è

$$\begin{aligned}\langle z \rangle &\rightsquigarrow \varepsilon \mid a\langle z \rangle \mid a\langle x \rangle \mid b\langle x \rangle \\ \langle y \rangle &\rightsquigarrow a\langle x \rangle \mid b\langle z \rangle \\ \langle x \rangle &\rightsquigarrow \varepsilon \mid a\langle y \rangle \mid b\langle y \rangle\end{aligned}$$

Esercizio 8.4.3. Per ognuno dei tre automi in (8.2), costruire la corrispondente grammatica libera da contesto.

La proprietà fondamentale di tale costruzione è che preserva il linguaggio accettato. Questo significa che una stringa $w \in A^*$ è accettata dallo stato $x \in S$ (in simboli $w \in \langle\langle x \rangle\rangle$) se e solo se w appartiene al linguaggio generato dalla categoria sintattica $\langle x \rangle$ della grammatica corrispondente (in simboli $w \in \langle\langle \langle x \rangle \rangle \rangle$).

Teorema 8.4.4. Per tutti gli automi $\mathcal{A} = (S, T, F)$, vale che per tutte le parole $w \in A^*$ e tutti gli stati $x \in S$

$$w \in \langle\langle x \rangle\rangle \text{ se e solo se } w \in \langle\langle \langle x \rangle \rangle \rangle.$$

Lasciamo la dimostrazione di questo teorema come esercizio per il lettore.

Esercizio 8.4.5. Dimostrare che per qualsiasi grammatica $\mathcal{G}_\mathcal{A} = (S_\mathcal{A}, P_\mathcal{A})$ corrispondente ad un automa \mathcal{A} vale che per tutti gli $\langle x \rangle \in S_\mathcal{A}$, $a \in A$ e $w \in A^*$:

1. $\varepsilon \in \langle\langle \langle x \rangle \rangle \rangle$ se e solo se $\langle x \rangle \rightsquigarrow \varepsilon \in P_\mathcal{A}$;
2. $aw \in \langle\langle \langle x \rangle \rangle \rangle$ se e solo se $\langle x \rangle \rightsquigarrow a\langle y \rangle \in P_\mathcal{A}$ e $w \in \langle\langle \langle y \rangle \rangle \rangle$ per qualche $\langle y \rangle \in S_\mathcal{A}$.

Esercizio 8.4.6. Dimostrare il Teorema 8.4.4. Si suggerisce di usare l'induzione su $w \in A^*$ e i due punti dell'Esercizio 8.2.16 e i due punti dell'Esercizio 8.4.5

Se \mathcal{A} è a stati finiti allora anche $\mathcal{G}_\mathcal{A}$ è finita e, pertanto, la seconda domanda ha risposta positiva.

La prima domanda invece ha risposta negativa. Un controsenso è fornito dal linguaggio generato dalla grammatica nell'Esempio 8.3.10. Come illustrato di seguito questo linguaggio non può essere riconosciuto da alcun automa a stati finiti.

Teorema 8.4.7. Sia $A = \{a, b\}$ e sia $L = \{a^n b^n \mid n \in \mathbb{N}\} \subseteq A^*$. Non esiste alcun automa a stati finiti $\mathcal{A} = (S, T, F)$ e alcuno stato $x \in S$ tale che $\langle\langle x \rangle\rangle = L$.

Dimostrazione. La dimostrazione procede per assurdo: assumiamo la negazione della tesi, per mostrare che si giunge ad una contraddizione.

Supponiamo per assurdo che esista un automa $\mathcal{A} = (S, T, F)$ sull'alfabeto A dove

1. esiste un $x \in S$ tale che $\langle\langle x \rangle\rangle = L$;
2. l'insieme di stati S ha una qualche cardinalità finita m .

Visto che m è un numero naturale, si ha che per definizione di L , la stringa $a^m b^m$ appartiene al linguaggio L .

Per definizione di linguaggio accettato, vale che esistono $y \in S$ e $z \in F$ tali che $(x, y) \in T_{a^m}$ e $(y, z) \in T_{b^m}$.

Per definizione di T_{a^m} si ha che esistono $x_i \in S$ per $i \in \{0, \dots, m\}$ tali che $x = x_0$, $y = x_m$ e $((a, x_i), x_{i+1})$ per $i \in \{0, \dots, m-1\}$. In altre parole deve esistere un walk

$$x = x_0 \xrightarrow{a} x_1 \xrightarrow{a} \dots \xrightarrow{a} x_{m-1} \xrightarrow{a} x_m = y .$$

Si noti che la sequenza di x_i definisce una funzione $f: \{0, \dots, m\} \rightarrow S$ che mappa ogni i in x_i . Visto che $|\{0, \dots, m\}| = m + 1$ e $|S| = m$, allora la funzione f non è iniettiva per IL PRINCIPIO DELLE BUCHE DEI PICCIONI (Corollario 6.2.4). Pertanto esistono due elementi j e k dell'insieme $\{0, \dots, m\}$ tali che $j \neq k$ e $x_j = x_k$. Visto che $j \neq k$ sappiamo che o $j < k$ o $k < j$. Assumiamo che $j < k$ (il caso $k > j$ è del tutto analogo).

8. Linguaggi Formali

Dal momento che $x_j = x_k$ possiamo costruire il walk

$$x = x_0 \xrightarrow{a} x_1 \xrightarrow{a} \dots \xrightarrow{a} x_j \xrightarrow{a} x_{k+1} \xrightarrow{a} \dots \xrightarrow{a} x_{m-1} \xrightarrow{a} x_m = y$$

e pertanto $(x, y) \in T_{a^o}$ dove $o = j + m - k$ (si noti che vale sempre $o < m$). Utilizzando adesso che $(y, z) \in T_{b^m}$, si ha che $(x, z) \in T_{a^o b^m}$ e visto che $z \in F$, la stringa $a^o b^m$ appartiene al linguaggio di x (in simboli $a^o b^m \in \langle\langle x \rangle\rangle$). Ma questo va contro l'ipotesi 1. che $\langle\langle x \rangle\rangle = \{a^n b^n \mid n \in \mathbb{N}\}$ dal momento che $o < m$.

Abbiamo raggiunto una contraddizione e quindi dimostrato il teorema. ■

Osservazione 8.4.8. Il Teorema 8.4.7 ci dice che $\{a^n b^n \mid n \in \mathbb{N}\}$ non può essere riconosciuto da un automa a stati finiti. Utilizzando un argomento simile, si può dimostrare che anche il linguaggio delle parentesi bilanciate sull'alfabeto $\{(,)\}$ non può essere riconosciuto da un automa a stati finiti. Tale linguaggio consiste di tutte le stringhe di parentesi, possibilmente annidate, che si aprono e chiudono lo stesso numero di volte. Sono ad esempio stringhe di questo linguaggio $((())$) e $((()())()$) ma non $((()$. Questo esempio suggerisce che anche la sintassi della maggior parte dei linguaggi di programmazione non può essere riconosciuta da automi a stati finiti.

Per questa ragione, la sintassi dei linguaggi di programmazione è specificata solitamente attraverso grammatiche, mentre gli automi sono utilizzati per la cosiddetta analisi lessicale. In estrema sintesi, l'analisi lessicale legge il codice sorgente un carattere alla volta e lo traduce in lessemi (token). Lo studente potrà approfondire questi argomenti in corsi più avanzati.

I Teoremi 8.4.4 e 8.4.7 ci dicono assieme una cosa molto importante: il primo stabilisce che le grammatiche sono *espressive almeno quanto* gli automi, il secondo che sono *strettamente più espressive* degli automi. Questo significa che l'insieme dei linguaggi riconosciuti dagli automi (a stati finiti) è *strettamente incluso* nell'insieme dei linguaggi generati da grammatiche (finite). I linguaggi del primo tipo sono detti *regolari* e quelli del secondo sono detti *liberi da contesto*. Queste due tipologie di linguaggi formano le prime due classi della ben nota gerarchia di Chomsky, riportata in Figura 8.3. Lo studente interessato avrà modo di conoscere ulteriori classi della gerarchia in corsi più avanzati.



Figura 8.3: La Gerarchia di Chomsky.

8.5 Espressioni Regolari

In Figura 8.3 la parola “regolare” è utilizzata per denotare i linguaggi riconoscibili da automi a stati finiti. Questi linguaggi sono chiamati così in quanto godono di una caratterizzazione algebrica in termini di *espressioni regolari*. In questa sezione introduciamo le espressioni regolari come un primo esempio di linguaggio formale: illustriamo la loro sintassi (data attraverso una grammatica

libera da contesto), la loro semantica (definita per induzione) ed enunciamo il sopramenzionato risultato di caratterizzazione.

Le espressioni regolari sono delle espressioni per denotare linguaggi su un alfabeto fissato A . Le espressioni regolari sono usate in molti contesti applicativi: nei motori di ricerca, nei comandi *search and replace* negli editor di testo, in molte utilities di text processing e nell'analisi lessicale. Inoltre molti linguaggi di programmazione provvedono librerie per le espressioni regolari.

Sintassi

Come gli automi e le grammatiche, anche le espressioni regolari sono definite per un alfabeto A fissato. Le espressioni regolari sono certe stringhe sull'alfabeto A esteso con gli operatori $\{+, \cdot, ^*\}$ e le costanti $\{0, 1\}$. Cioè, l'alfabeto delle espressioni regolari è

$$A \cup \{+, \cdot, ^*\} \cup \{0, 1\} \quad (8.27)$$

In queste note, come nella maggior parte delle presentazioni delle espressioni regolari, si assume che A sia disgiunto da $\{+, \cdot, ^*\} \cup \{0, 1\}$, ovvero che $+, \cdot, ^*, 0, 1$ siano simboli, in qualche senso speciali, che non appartengono ad A .

In modo del tutto analogo alle espressioni aritmetiche, non tutte le stringhe sull'alfabeto (8.27) sono espressioni regolari. Ad esempio per $A = \{a, b\}$, $a +$ e a^*b non sono espressioni regolari mentre $a + 1$, $a \cdot b^*$ e $a \cdot a \cdot b$ lo sono. Per definire in modo preciso cosa sia un'espressione regolare è conveniente utilizzare la seguente grammatica libera da contesto.

$$\begin{array}{ll} \langle RExp_A \rangle & \rightsquigarrow a \mid \\ & 0 \mid \\ & 1 \mid \\ & \langle RExp_A \rangle + \langle RExp_A \rangle \mid \\ & \langle RExp_A \rangle \cdot \langle RExp_A \rangle \mid \\ & \langle RExp_A \rangle^* \end{array} \quad (8.28)$$

La prima produzione $\langle RExp_A \rangle \rightsquigarrow a$ va pensata come tante (esattamente $|A|$), produzioni. Ad esempio se $A = \{a, b\}$, allora si hanno due produzioni: $\langle RExp_A \rangle \rightsquigarrow a$ e $\langle RExp_A \rangle \rightsquigarrow b$. In estrema sintesi, tale produzione ci dice che ogni simbolo in A è una espressione regolare. La seconda e la terza produzione ci dice che anche 0 e 1 sono espressioni regolari. La quarta e la quinta produzione ci dice che se e_1 ed e_2 sono espressioni regolari, allora anche $e_1 + e_2$ ed $e_1 \cdot e_2$ sono espressioni regolari, mentre la sesta ci dice che se e è un'espressione regolare allora anche e^* è un'espressione regolare.

Definizione 8.5.1 (espressione regolare). *Una espressione regolare è una stringa che appartiene a $\langle\langle RExp_A \rangle\rangle$. Utilizziamo $RExp_A$ per l'insieme di tutte le espressioni regolari. Cioè $RExp_A = \langle\langle RExp_A \rangle\rangle$.*

Esempio 8.5.2. Fissiamo $A = \{a, b, c\}$ e consideriamo le seguenti stringhe.

- $0 + 1$ è una espressione regolare
- $1 + 1$ è una espressione regolare
- $1 \cdot b$ è una espressione regolare
- $1 \cdot b^*$ è una espressione regolare
- $1 \cdot b^* + a \cdot b^*$ è una espressione regolare
- $+ \cdot^*$ non è una espressione regolare

Esercizio 8.5.3. Per le prime cinque stringhe in Esempio 8.5.2 costruire tutti gli alberi di derivazione sintattica rispetto alla grammatica in (8.28).

Prima di considerare la semantica di queste espressioni è opportuno considerare ancora la loro sintassi. Come per le espressioni aritmetiche, la grammatica per le espressioni regolari è *ambigua*. Infatti, il lettore che ha svolto correttamente l'esercizio precedente, ha potuto constatare che per la stringa $1 \cdot b^* + a \cdot b^*$ esiste più di un parse tree. Per definire delle espressioni regolari è necessario risolvere questa ambiguità. Come per le espressioni aritmetiche, si risolve questo problema introducendo una precedenza sugli operatori: si impone che $*$ lega più di \cdot e che \cdot lega più di $+$. Queste precedenze sono riassunte nella seguente tabella.

operatore	livello di precedenza
$+$	0
\cdot	1
$*$	2

Si consideri ad esempio l'espressione $a + 1 \cdot b$. Tale espressione è testimoniata da due diversi parse trees: in uno l'operatore più vicino alla radice è $+$ e nell'altro l'operatore più vicino alla radice è \cdot . Intuitivamente, il primo corrisponde a $a + (1 \cdot b)$, mentre il secondo a $(a + 1) \cdot b$. I livelli di precedenza ci dicono che quando ci troviamo di fronte all'espressione $a + 1 \cdot b$, dobbiamo considerarla come generata dal parse tree $a + (1 \cdot b)$ visto che \cdot lega di più (cioè ha precedenza più alta) di $+$. Per indicare l'espressione generata dal secondo parsee tree, utilizzeremo come sempre le *parentesi tonde*. In queste note evitiamo di redefinire la grammatica con le parentesi ed ogni volta che le incontriamo ci riferiamo al significato appena illustrato.

Esempio 8.5.4. Fissiamo $A = \{a, b, c\}$ e consideriamo le seguenti stringhe.

- $0 + 1 \cdot a$ è $0 + (1 \cdot a)$ e non $(0 + 1) \cdot a$.
- $c + a^*$ è $c + (a^*)$ e non $(c + a)^*$.
- $c \cdot b^*$ è $c \cdot (b^*)$ e non $(c \cdot b)^*$.

Dopo aver definito la semantica sarà chiaro che gli operatori $+$ e \cdot sono associativi e pertanto espressioni come $a + b + c$ o $a \cdot b \cdot c$ hanno entrambe due diversi parse trees ciascuna (corrispondenti a $a + (b + c)$ e $(a + b) + c$, rispettivamente, $a \cdot (b \cdot c)$ e $(a \cdot b) \cdot c$), ma i due hanno lo stesso significato. Pertanto, possiamo evitare di mettere parentesi in queste situazioni.

Semantica

La semantica per le espressioni regolari assegna ad ogni espressione $e \in RExp_A$ un linguaggio in $\mathcal{P}(A^*)$. Intuitivamente:

- a denota il linguaggio $\{a\}$
- 0 denota il linguaggio vuoto \emptyset
- 1 denota il linguaggio $\{\varepsilon\}$
- $+$ esegue l'unione dei linguaggi (\cup)
- \cdot esegue la concatenazione di linguaggi (\cdot)
- $*$ esegue la stella di Kleene di linguaggi ($*$)

Si ricorda al lettore che queste operazioni sui linguaggi (e le loro proprietà algebriche) sono state studiate alla fine della Sezione 8.1.

Esempio 8.5.5. Consideriamo a livello intuitivo alcuni esempi sull'alfabeto $A = \{a, b, c\}$.

- $a \cdot b$ denota il linguaggio $\{a\} \cdot \{b\} = \{ab\}$.
- $a \cdot b + c$ denota il linguaggio $(\{a\} \cdot \{b\}) \cup \{c\} = \{ab\} \cup \{c\} = \{ab, c\}$.
- $a \cdot (b + c)$ denota il linguaggio $a \cdot (\{b\} \cup \{c\}) = a \cdot \{b, c\} = \{ab, ac\}$.

- $(a \cdot b) + 1$ denota il linguaggio $(\{a\} \cdot \{b\}) \cup \{\varepsilon\} = \{ab\} \cup \{\varepsilon\}$.
- $(a \cdot b)^* + 1$ denota il linguaggio $((\{a\} \cdot \{b\}))^* \cup \{\varepsilon\} = \{ab\}^* \cup \{\varepsilon\} = \{\varepsilon, ab, abab, ababab, \dots\}$.

Una volta stabilita un'intuizione preliminare è opportuno definire la semantica in maniera *formale*, cioè come una funzione dal dominio sintattico (in questo caso le espressioni regolari) a quello semantico (in questo caso i linguaggi).

Definizione 8.5.6. [semantica di espressioni regolari] La semantica delle espressioni regolari è una funzione $\mathcal{S}: RExp_A \rightarrow \mathcal{P}(A^*)$ definita ricorsivamente per ogni $e \in RExp_A$ come

$$\begin{aligned}\mathcal{S}(e) = & \\ \{a\} & \quad \text{se } e = a \\ \emptyset & \quad \text{se } e = 0, \\ \{\varepsilon\} & \quad \text{se } e = 1 \\ \mathcal{S}(e_1) \cup \mathcal{S}(e_2) & \quad \text{se } e = e_1 + e_2 \\ \mathcal{S}(e_1) \cdot \mathcal{S}(e_2) & \quad \text{se } e = e_1 \cdot e_2 \\ \mathcal{S}(e_1)^* & \quad \text{se } e = e_1^*\end{aligned}$$

Illustriamo tale definizione con il seguente esempio.

$$\begin{aligned}\mathcal{S}((a \cdot b)^* + c) &= \mathcal{S}((a \cdot b)^*) \cup \mathcal{S}(c) \\ &= \mathcal{S}(a \cdot b)^* \cup \mathcal{S}(c) \\ &= (\mathcal{S}(a) \cdot \mathcal{S}(b))^* \cup \mathcal{S}(c) \\ &= (\{a\} \cdot \{b\})^* \cup \{c\} \\ &= \{ab\}^* \cup \{c\} \\ &= \{\varepsilon, ab, abab, ababab, \dots c\}\end{aligned}$$

Nella prima uguaglianza di sopra si prende e come $(a \cdot b)^* + c$ e si utilizza la quarta clausola della Definizione 8.5.6 dove e_1 è $(a \cdot b)^*$ ed e_2 è c . La definizione ci dice che la semantica di $(a \cdot b)^* + c$ è esattamente l'unione della semantica di $(a \cdot b)^*$ e della semantica di c .

Nella seconda uguaglianza, si prende e come $(a \cdot b)^*$ e si utilizza la sesta clausola della Definizione 8.5.6 dove e_1 è $a \cdot b$. La definizione ci dice che la semantica di $(a \cdot b)^*$ è esattamente la stella di Kleene del linguaggio ottenuto come semantica di $a \cdot b$.

Nella terza uguaglianza si procede in maniera analoga, ma utilizzando la quinta clausola. Nella quarta uguaglianza si utilizza contemporaneamente (per tre volte) la prima clausola della Definizione 8.5.6. Nella quinta e nella sesta uguaglianza si eseguono semplicemente le operazioni sui linguaggi, come definito alla fine della Sezione 8.1.

Esercizio 8.5.7. Calcolare la semantica delle espressioni $a \cdot (b^*)$, di $(a \cdot b)^*$, di $a \cdot (b + c)$, di $(a \cdot b) + (a \cdot c)$ e di $(a \cdot b) + c$.

Osservazione 8.5.8. La semantica di molti linguaggi formali viene spesso definita in maniera analoga a quella della Definizione 8.5.6: ogni operatore **op** della sintassi corrisponde ad un'operazione **op** sul dominio semantico e la semantica di una espressione del tipo $e_1 \mathbf{op} e_2$ viene definita come $\mathcal{S}(e_1 \mathbf{op} e_2) = \mathcal{S}(e_1) \overline{\mathbf{op}} \mathcal{S}(e_2)$. Questa tipologia di definizione della semantica è spesso chiamata semantica denotazionale.

Teorema di Corrispondenza

All'inizio di questa sezione abbiamo annunciato che vale una corrispondenza tra le espressioni regolari e gli automi a stati finiti. Tale corrispondenza è ben nota come teorema di Kleene. Prima introduciamo l'ultima definizione.

Definizione 8.5.9 (linguaggio regolare). Un linguaggio $L \in \mathcal{P}(A^*)$ è detto regolare se esiste una espressione regolare $e \in RExp_A$ che denota L , cioè tale che

$$\mathcal{S}(e) = L.$$

8. Linguaggi Formali

Concludiamo con l'enunciato del teorema di Kleene.

Teorema 8.5.10 (Kleene). *Sia A un alfabeto e $L \in \mathcal{P}(A^*)$ un linguaggio su A . L è regolare se e solo se esiste un automa a stati finiti $\mathcal{A} = (S, T, F)$ ed uno stato $x \in S$ tale che $\langle\langle x \rangle\rangle = L$.*

CAPITOLO 9

Logica Matematica

Nei nostri studi matematici di ogni livello, e anche nello studio di altre discipline, abbiamo incontrato innumerevoli risultati (chiamati teoremi, lemmi, proposizioni, corollari, ...), e talvolta di questi risultati ci è stata presentata una dimostrazione. Le regole della logica permettono di stabilire con precisione il significato di un enunciato matematico, rimediando a possibili ambiguità e imprecisioni del linguaggio naturale. Inoltre tali regole consentono di distinguere argomentazioni matematiche valide da quelle non valide, e quindi di capire se una dimostrazione proposta è corretta oppure no. L'obiettivo della logica non è tanto quello di stabilire la validità assoluta di un certo enunciato, ma piuttosto quello di stabilire quando, assumendo vere certe premesse, si possa affermare la validità di altri enunciati: in questo caso diremo che questi ultimi sono *conseguenze logiche* delle premesse.

Per questo motivo la logica è la base di ogni ragionamento matematico, ma essa ha anche numerose applicazioni in diversi campi dell'informatica: la progettazione di computer, l'intelligenza artificiale, l'ingegneria del software, la programmazione e tanti altri. Per esempio si diffonde sempre di più l'uso di tecniche di dimostrazione per certificare che un certo hardware o software si comporterà sempre correttamente.

Data la complessità dell'attività deduttiva e inferenziale degli esseri umani sono state sviluppate diverse logiche che si propongono di catturare aspetti sempre più complessi del ragionamento. Noi ci limitiamo a considerare le *logiche classiche*, che trattano enunciati che possono assumere uno e uno solo dei valori booleani dell'insieme $Bool = \{t, f\}$, introdotto nella Sezione 1.1. Useremo i valori di $Bool$ solo per la semantica delle formule, mentre useremo una notazione diversa per rappresentare tali valori nelle formule.

Notazione 9.0.1 (rappresentazione sintattica dei valori booleani). *Per gli elementi dell'insieme $Bool = \{t, f\}$ useremo i simboli T (per t) e F (per f) per la sintassi, cioè per la loro rappresentazione simbolica nelle formule.*

Definizione 9.0.2 (proposizioni). *Una PROPOSIZIONE è un enunciato dichiarativo (per esempio una frase in linguaggio naturale) che “afferma qualcosa” e per il quale si può dire:*

PRINCIPIO DEL TERZO ESCLUSO: che è vero oppure è falso (non ci sono altre possibilità)

PRINCIPIO DI NON CONTRADDITORIETÀ: che non è al tempo stesso sia vero che falso.

Per esempio, “*Quanti sono i numeri primi minori di 100?*” non è una proposizione, perché non “afferma qualcosa”. Al contrario, “*Sono biondo naturale*” è una proposizione, che può essere vera o falsa a seconda del soggetto che la legge.

Esempio 9.0.3 (riconoscere le proposizioni). *Vediamo per ognuna delle frasi seguenti se sono proposizioni oppure no.*

- Firenze è la capitale d'Italia.

Si tratta di una proposizione, in quanto afferma un fatto che può essere vero o falso, ma non entrambi. In particolare, la proposizione è falsa per il significato usuale che associamo alle parole Firenze, capitale e Italia.

- È possibile che Firenze sia la capitale d'Italia.

Non afferma qualcosa, ma esprime una valutazione sul fatto che l'enunciato Firenze è la capitale d'Italia sia vero, pertanto non è una proposizione.

- Firenze è la capitale d'Italia?
Si tratta di un enunciato interrogativo che non afferma qualcosa, quindi non è una proposizione.
- Firenze è stata la capitale d'Italia.
È una proposizione perché afferma un fatto che, in particolare, è vero.
- Se solo Firenze fosse la capitale d'Italia!
È un enunciato che esprime un desiderio ma non afferma qualcosa, quindi non è una proposizione.
- Probabilmente Firenze è la capitale d'Italia.
Non è una proposizione per gli stessi motivi della seconda frase vista sopra.

Le proposizioni semplici, che rappresenteremo astrattamente con lettere come A, B, C, \dots , possono essere composte per formare proposizioni più complesse usando i *connettivi logici*, alcuni dei quali li abbiamo già incontrati nella Sezione 1.3 (*and*, *or* e *not*). Nella Sezione 9.1, presentando la sintassi e la semantica del Calcolo Proposizionale, introdurremo questi connettivi come operatori algebrici, e descriveremo come ottenere il valore di verità di proposizioni complesse a partire da quello dei componenti. Nella Sezione 9.2 presenteremo le tavole di verità e il concetto di tautologia, e discuteremo della correttezza di semplici inferenze logiche. La Sezione 9.3 è dedicata al problema di dimostrare tautologie nel calcolo proposizionale. Vedremo come si possono usare dimostrazioni per sostituzione a questo scopo, e inquadreremo questo tipo di dimostrazioni nel contesto più generale dei Sistemi di Dimostrazioni (o Proof Systems), che forniscono una definizione formale di dimostrazione. Infine discuteremo della correttezza di alcune classiche tecniche di dimostrazione.

Nella Sezione 9.5 introduciamo la Logica dei Predicati, una logica più espressiva del calcolo proposizionale che grazie ai quantificatori e ai termini consente di esprimere asserti che menzionano esplicitamente elementi di un *dominio*, cioè un insieme che in casi concreti può contenere numeri, persone, o entità di altro tipo. Anche per la logica dei predicati presenteremo la sintassi e la semantica. Inoltre introduciamo delle leggi che riguardano i quantificatori, che ci permetteranno di dimostrare la validità di formule usando dimostrazioni per sostituzione.

Gli aspetti della logica matematica cui siamo interessati in questo capitolo non forniscono strumenti per dimostrare automaticamente dei teoremi o per “creare” ragionamenti o inferenze complesse, ma ci permettono di *analizzare* dimostrazioni, ragionamenti e inferenze per controllare che siano logicamente corretti. Per analizzare logicamente un ragionamento occorre *formalizzare* le proposizioni che esso contiene, cioè renderne esplicita la struttura logica: questo serve per controllare se la conclusione è una conseguenza logica delle premesse e per riconoscere ed escludere ragionamenti sbagliati che porterebbero a conclusioni erronee. Sia per il calcolo proposizionale che per la logica dei predicati discuteremo come formalizzare delle semplici proposizioni espresse in linguaggio naturale in formule logiche, e come verificare che semplici inferenze siano corrette.

9.1 Il Calcolo Proposizionale: sintassi e semantica

Il *calcolo proposizionale* (chiamato anche *logica proposizionale*) costituisce il nucleo di tutte le logiche classiche. In questa sezione mostriamo come scrivere (la sintassi) e come interpretare (la semantica) delle formule proposizionali, anche con l’ausilio delle tavole di verità introdotte nella Sezione 9.2. Successivamente, nella Sezione 9.3 presenteremo le leggi algebriche che descrivono le proprietà dei connettivi logici e discuteremo come utilizzare le dimostrazioni per sostituzione per dimostrare l’equivalenza logica di formule proposizionali. Questa tecnica può essere usata per dimostrare che certe formule sono *tautologie*, cioè sempre vere.

Ricordiamo che, nonostante la sua semplicità, il calcolo proposizionale ha svariate applicazioni in informatica, come per esempio le operazioni bit-a-bit e i circuiti binari, il flusso di controllo nella programmazione, le analisi delle specifiche, le interrogazioni su basi di dati, la ricerca booleana nei motori di ricerca, e molte altre ancora che qui non tratteremo ma che incontrerete nel corso di laurea in Informatica.

Sintassi

Le formule del calcolo proposizionale sono ottenute a partire da un insieme di *simboli proposizionali*, che rappresentano dei fatti o enunciati basilari, componendoli in modo arbitrario con le operazioni di negazione, congiunzione, disgiunzione e implicazione (semplice o doppia), chiamati anche *connettivi logici*.¹

Definizione 9.1.1 (Sintassi del calcolo proposizionale). *Fissato un insieme $X = \{A, B, C, \dots\}$ di SIMBOLI PROPOSIZIONALI, l'insieme delle FORMULE PROPOSIZIONALI \mathbf{Prop} è il linguaggio generato dalla categoria sintattica $\langle Prop \rangle$ della seguente grammatica:*²

Linguaggio proposizionale

$$\begin{aligned}\langle Prop \rangle &\rightsquigarrow \langle Atom \rangle \mid \neg \langle Atom \rangle \mid \langle Prop \rangle \langle OpB \rangle \langle Prop \rangle \\ \langle Atom \rangle &\rightsquigarrow \top \mid \bot \mid \langle X \rangle \mid (\langle Prop \rangle) \\ \langle OpB \rangle &\rightsquigarrow \wedge \mid \vee \mid \Rightarrow \mid \Leftarrow \mid \Leftrightarrow \\ \langle X \rangle &\rightsquigarrow A \mid B \mid C \mid \dots\end{aligned}$$

Dalla definizione si evince che l'alfabeto della grammatica è l'insieme

$$A_{Prop} = X \cup \{\Leftrightarrow, \Rightarrow, \Leftarrow, \wedge, \vee, \neg, \top, \bot, (,)\}$$

mentre le categorie sintattiche sono $\{\langle Prop \rangle, \langle Atom \rangle, \langle OpB \rangle, \langle X \rangle\}$. Useremo di solito le lettere P, Q, R, \dots per denotare generiche formule proposizionali, e A, B, C, \dots per i simboli proposizionali.

La categoria sintattica $\langle Atom \rangle$ genera le formule ATOMICHE, $\langle OpB \rangle$ genera i CONNETTIVI LOGICI, e $\langle X \rangle$ i simboli proposizionali.

Esempio 9.1.2 (formule proposizionali). *Dalla definizione segue che le seguenti stringhe sono formule proposizionali:*

$$\top \quad A \wedge (B \vee \neg C) \quad ((A \wedge B \vee \top) \Rightarrow \neg A \vee (C \wedge \top)) \quad (A \Rightarrow B) \wedge \neg(B \wedge \neg C) \Rightarrow (A \Rightarrow C)$$

Invece non sono formule proposizionali per esempio le stringhe $A \wedge \vee B$, $(A \neg C)$ e $A \neg$, perché non sono ottenibili con le produzioni della Definizione 9.1.1.

I connettivi logici possono essere letti in vari modi, di cui menzioniamo i più comuni:

- La NEGAZIONE $\neg P$ può essere detta “non P ”, “not P ”, “non è vero che P vale”, ...
- La CONGIUNZIONE $P \wedge Q$ può essere detta “ P e Q ”, “ P and Q ”, “ P e anche Q ”, ...
- La DISGIUNZIONE $P \vee Q$ può essere detta “ P o Q ”, “ P or Q ”, “ P oppure Q ”, ...
- L’IMPLICAZIONE $P \Rightarrow Q$ può essere detta “se P allora Q ”, “ P implica Q ”, “ P solo se Q ”, “ P è condizione sufficiente per Q ”, ... Nell’implicazione $P \Rightarrow Q$ la formula proposizionale P è chiamata la *premessa*, mentre Q è la *conseguenza* o *conclusione*.
- La CONSEGUENZA $P \Leftarrow Q$ può essere detta “ P è conseguenza di Q ”, “ P se Q ”, “ P if Q ”, “ P è condizione necessaria per Q ”, ...
- La DOPPIA IMPLICAZIONE $P \Leftrightarrow Q$ può essere detta “ P sse Q ”, “ P se e solo se Q ”, “ P iff Q ”, “ P è condizione necessaria e sufficiente per Q ”, ...

Osservazione 9.1.3. *La conseguenza $P \Leftarrow Q$ può essere considerata come un modo alternativo di scrivere l’implicazione $Q \Rightarrow P$. Quindi $Q \Rightarrow P$ può essere detta anche come “ P se Q ”, “ P if Q ”, “ P è condizione necessaria per Q ”.*

¹L’introduzione dei connettivi logici come operatori su formule (in qualche modo analoghi ai ben noti operatori algebrici su espressioni aritmetiche) risale al 1854, con la pubblicazione dell’opera “The Laws of Thought” di George Boole (1815–1864), importante logico e matematico inglese. I valori booleani prendono il nome da lui.

²Usando la terminologia della Definizione 8.3.8, $\mathbf{Prop} = \langle\langle\langle Prop \rangle\rangle\rangle$.

Formalizzare proposizioni

Per “formalizzare” intendiamo il procedimento di estrazione, da una proposizione in italiano, di una formula del calcolo proposizionale che ne ha la stessa struttura logica. Di solito questo procedimento non è molto complicato. Seguiamone i passi nel caso della proposizione *“Piove e fa freddo.”*.

- Come prima cosa occorre introdurre un simbolo proposizionale per ogni proposizione elementare, cioè ogni pezzo della frase che riconosciamo essere una proposizione (ha uno e un solo valore di verità), e che non può essere scomposta in proposizioni più piccole. Nel nostro esempio, fissiamo i simboli proposizionali P per “*piove*” e Fr per “*fa freddo*”. (Non usiamo F per “*fa freddo*” per evitare confusione con F . Osserviamo pure che i simboli proposizionali possono essere stringhe arbitrarie, non solo caratteri.)
- Successivamente si costruisce la formula collegando le occorrenze dei simboli proposizionali con connettivi logici, in modo da rispecchiare fedelmente il significato originale. Nell'esempio, la formula risultante è $P \wedge Fr$, dato che “e” rappresenta chiaramente una congiunzione.

La scelta dei connettivi non è sempre ovvia poiché in italiano possono essere resi in molti modi diversi.

Esempio 9.1.4 (da proposizioni in linguaggio naturale a formule proposizionali). *Vediamo come formalizzare altre proposizioni simili.*

1. *“Piove ma non fa freddo.”* *Usando i simboli proposizionali appena introdotti, una ragionevole formalizzazione è $P \wedge \neg Fr$. Infatti anche “ma” rappresenta una congiunzione, anche se con un significato leggermente diverso da “e”.*

2. *“Se piove o fa freddo allora ci si copre.”* *Usiamo i simboli P e Fr come sopra, e introduciamo C per “ci si copre”. Allora la formula risultante è $(P \vee Fr) \Rightarrow C$. Infatti “se … allora …” rappresenta un’implicazione, e “o” una disgiunzione.*

3. *“Se piove non si esce o si prende l’ombrelllo.”* *Usiamo E per “si esce” e O per “si prende l’ombrelllo”. Una ragionevole formalizzazione è $P \Rightarrow (\neg E \vee O)$.*

Si noti che un’altra formalizzazione possibile, data l’ambiguità della proposizione in italiano, sarebbe quella che pone in disgiunzione gli enunciati “Se piove non si esce” e “si prende l’ombrelllo”: $(P \Rightarrow \neg E) \vee O$. Il contesto e il buon senso ci portano a scartare questa possibilità.

Semantica

La semantica di una formula proposizionale, cioè il suo valore di verità, può essere calcolato per induzione strutturale in base al suo albero di derivazione. In generale tale valore non è determinato univocamente ma dipende da una *interpretazione*, cioè dal valore di verità assegnato ai simboli proposizionali che essa contiene, proprio come il valore di un’espressione algebrica dipende dal valore assegnato alle variabili che contiene.

Definizione 9.1.5 (interpretazione). *Un’INTERPRETAZIONE $\mathcal{I} : X \rightarrow \{\text{f}, \text{t}\}$ è una funzione che assegna un valore di verità a ogni simbolo proposizionale.*

Di seguito useremo la notazione $\mathcal{I} = \{\dots, A \mapsto \text{t}, \dots\}$ (oppure $\mathcal{I} = \{\dots, A \mapsto \text{f}, \dots\}$) per indicare un’interpretazione che assegna il booleano vero (oppure falso) al simbolo A .

Fissato il valore dei simboli proposizionali, la semantica di una formula proposizionale è ottenuta, per induzione strutturale, valutando i connettivi logici che compaiono in essa in base al loro significato. Abbiamo anticipato il significato dei connettivi \wedge , \vee e \neg negli Esempi 2.5.11 e 2.5.12: lo ricordiamo nella seguente definizione insieme a quello degli altri connettivi, utilizzando delle *tavole di verità*.

Definizione 9.1.6 (connettivi logici come funzioni). *Come visto nell’Esempio 2.5.11, la negazione è una funzione $\neg : \text{Bool} \rightarrow \text{Bool}$, descritta dalla tavola di verità che segue a sinistra. La prima*

colonna elenca i possibili valori assunti da $x \in \text{Bool}$, e la seconda colonna mostra i corrispondenti valori di $\neg x$.

x	y	$x \wedge y$	$x \vee y$	$x \Rightarrow y$	$x \Leftarrow y$	$x \Leftrightarrow y$
x	$\neg x$	f	f	t	t	t
f	t	f	t	t	f	f
t	f	f	t	f	t	f
		t	t	t	t	t

Analogamente, i connettivi logici binari $\wedge, \vee, \Rightarrow, \Leftarrow, \Leftrightarrow$ sono delle funzioni da $\text{Bool} \times \text{Bool}$ a Bool definite dalla seconda tavola di verità. Nella parte a sinistra della doppia linea verticale sono elencate le quattro coppie di valori booleani che possono assumere x e y . Per ognuna di queste coppie, nella parte destra della tavola sono riportati sulla stessa riga i valori delle formule $x \wedge y$, $x \vee y$, $x \Rightarrow y$, $x \Leftarrow y$ e $x \Leftrightarrow y$.

Abbiamo già commentato nella Sezione 1.3 e nell’Esempio 2.5.12 i connettivi \wedge e \vee , per cui ci limitiamo a osservare che la tavola di verità conferma il significato che gli avevamo attribuito:

- (\wedge) La congiunzione $x \wedge y$ è vera se sia x che y sono vere, altrimenti è falsa. Infatti nella colonna relativa a $x \wedge y$ c’è una sola riga con t , corrispondente ai valori t e t per x e y .
- (\vee) La disgiunzione $x \vee y$ è vera se x è vera oppure y è vera oppure entrambe sono vere, altrimenti è falsa. Infatti nella colonna relativa a $x \vee y$ c’è una sola riga con f , corrispondente ai valori f e f per x e y .

Per quanto riguarda l’implicazione $x \Rightarrow y$, analizzando la terzultima colonna della tavola di verità vediamo che essa è falsa se la premessa x è vera e la conseguenza y è falsa, mentre è vera in tutti gli altri casi. Intuitivamente, possiamo pensare a $x \Rightarrow y$ come a una regola (“se x è vera allora y deve essere vera”), e il suo valore booleano ci dice se la regola è rispettata oppure no. Allora se x è falsa la regola è automaticamente rispettata perché non si applica affatto (le prime due righe della tavola, per $x = f$). Se x è vera e anche y è vera, la regola è rispettata (l’ultima riga). L’unico caso in cui la regola non è rispettata, e quindi l’implicazione vale f , è quando x è vera ma y è falsa (la penultima riga della tavola).

Per la conseguenza $x \Leftarrow y$ si applicano considerazioni simili a quelle per l’implicazione, scambiando i ruoli di x e y per l’Osservazione 9.1.3.

Infine dall’ultima colonna della tavola si vede che la doppia implicazione $x \Leftrightarrow y$ è vera se e solo se x e y hanno lo stesso valore.

Esempio 9.1.7 (formalizzare proposizioni con implicazione). Ora che abbiamo descritto in modo preciso il significato dell’implicazione, vediamo come lo possiamo usare per formalizzare correttamente una proposizione. Consideriamo le proposizioni elementari “io vado al cinema”, rappresentata dalla variabile proposizionale V , e “tu resti a casa”, rappresentata da R . Come possiamo formalizzare la proposizione “Affinché io vada al cinema è necessario che tu resti a casa”?

Riconosciamo che la frase esprime una regola: se V è vera, necessariamente R deve essere vera, quindi la regola è violata solo se V è vera e R è falsa. Questo corrisponde precisamente alla regola associata all’implicazione $V \Rightarrow R$, come descritta sopra, che quindi è una adeguata formalizzazione della proposizione.

Consideriamo ora “Io vado al cinema se tu resti a casa”. In questo caso la regola sancita dalla proposizione è violata quando “tu resti a casa” ma “io non vado al cinema”, e quindi una adeguata formalizzazione è $R \Rightarrow V$.

Definizione 9.1.8 (Semantica del calcolo proposizionale). Data una interpretazione $\mathcal{I} : X \rightarrow \{f, t\}$, il VALORE RISPETTO AD \mathcal{I} delle formule proposizionali è dato dalla funzione

$$\llbracket _ \rrbracket_{\mathcal{I}} : \mathbf{Prop} \rightarrow \{f, t\}$$

definita sulle formule proposizionali per induzione strutturale come segue:

1. $\llbracket T \rrbracket_{\mathcal{I}} = t$ e $\llbracket F \rrbracket_{\mathcal{I}} = f$;

2. $\llbracket A \rrbracket_{\mathcal{I}} = \mathcal{I}(A)$ per ogni $A \in X$;
3. $\llbracket (P) \rrbracket_{\mathcal{I}} = (\llbracket P \rrbracket_{\mathcal{I}})$ per ogni $P \in \mathbf{Prop}$;
4. $\llbracket \neg Q \rrbracket_{\mathcal{I}} = \neg \llbracket Q \rrbracket_{\mathcal{I}}$ per ogni formula atomica Q ;
5. $\llbracket P \text{ op } Q \rrbracket_{\mathcal{I}} = \llbracket P \rrbracket_{\mathcal{I}} \text{ op } \llbracket Q \rrbracket_{\mathcal{I}}$ per ogni connettivo $\text{op} \in \{\wedge, \vee, \Rightarrow, \Leftarrow, \Leftrightarrow\}$ e per ogni $P, Q \in \mathbf{Prop}$.

Si osservi come le clausole della Definizione 9.1.8 corrispondano alle produzioni della grammatica di Definizione 9.1.1, confermando che $\llbracket _ \rrbracket_{\mathcal{I}}$ è definita per induzione strutturale: le clausole 1–3 corrispondono alla produzione $\langle \text{Prop} \rangle \rightsquigarrow \langle \text{Atom} \rangle$ (più precisamente alle produzioni di $\langle \text{Atom} \rangle$), la clausola 4 a $\langle \text{Prop} \rangle \rightsquigarrow \neg \langle \text{Atom} \rangle$ e la clausola 5 a $\langle \text{Prop} \rangle \rightsquigarrow \langle \text{Prop} \rangle \langle \text{OpB} \rangle \langle \text{Prop} \rangle$.

Esempio 9.1.9 (valutazione di una formula). *Consideriamo la formula $P = (A \wedge B) \vee \neg C$ e l'interpretazione $\mathcal{I} = \{A \mapsto t, B \mapsto f, C \mapsto f, \dots\}$,³ e calcoliamo il valore $\llbracket P \rrbracket_{\mathcal{I}}$ secondo la Definizione 9.1.8.*

$$\begin{aligned}
 \llbracket (A \wedge B) \vee \neg C \rrbracket_{\mathcal{I}} &= \llbracket (A \wedge B) \rrbracket_{\mathcal{I}} \vee \llbracket \neg C \rrbracket_{\mathcal{I}} && (\text{per la clausola 5 con op} = \vee) \\
 &= (\llbracket A \rrbracket_{\mathcal{I}} \wedge \llbracket B \rrbracket_{\mathcal{I}}) \vee \neg \llbracket C \rrbracket_{\mathcal{I}} && (\text{clausola 3 e clausola 4}) \\
 &= (\llbracket A \rrbracket_{\mathcal{I}} \wedge \llbracket B \rrbracket_{\mathcal{I}}) \vee \neg \llbracket C \rrbracket_{\mathcal{I}} && (\text{clausola 5 con op} = \wedge) \\
 &= (t \wedge f) \vee \neg f && (\text{clausola 2, tre volte}) \\
 &= f \vee t && (\text{definizione di } \wedge \text{ e } \neg \text{ da Definizione 9.1.6}) \\
 &= t && (\text{definizione di } \vee)
 \end{aligned}$$

Concludiamo questa sezione introducendo la definizione di modello di un insieme di formule proposizionali e il concetto di conseguenza logica.

Definizione 9.1.10 (modello, equivalenza, conseguenza logica). *Data una formula proposizionale P e una interpretazione \mathcal{I} , diciamo che \mathcal{I} è un MODELLO di P se P è vera in \mathcal{I} , cioè se $\llbracket P \rrbracket_{\mathcal{I}} = t$. Per questa importante nozione introduciamo una apposita notazione:*

$$\mathcal{I} \models P \quad (\mathcal{I} \text{ è modello di } P)$$

Se invece $\llbracket P \rrbracket_{\mathcal{I}} = f$ scriviamo $\mathcal{I} \not\models P$. La notazione si estende nel modo ovvio a un insieme di formule Γ (letto “Gamma”): scriviamo $\mathcal{I} \models \Gamma$ se $\mathcal{I} \models P$ per ogni $P \in \Gamma$, mentre scriviamo $\mathcal{I} \not\models \Gamma$ se c'è almeno una formula $P \in \Gamma$ tale che $\mathcal{I} \not\models P$.

Due formule proposizionali P e Q sono LOGICAMENTE EQUIVALENTI se hanno gli stessi modelli, cioè assumono lo stesso valore di verità per qualunque interpretazione. In questo caso scriviamo:

$$P \equiv Q \quad (P \text{ e } Q \text{ sono logicamente equivalenti})$$

Data una formula proposizionale P e un insieme di formule Γ , diciamo che P è una CONSEGUENZA LOGICA di Γ se P è vera in ogni interpretazione che rende vere tutte le formule di Γ , oppure, equivalentemente, se ogni modello di Γ è anche un modello di P . In questo caso scriviamo:

$$\Gamma \models P \quad (P \text{ è conseguenza logica di } \Gamma)$$

È opportuno sottolineare che dalla definizione di modello segue che per qualunque interpretazione \mathcal{I} vale $\mathcal{I} \models \emptyset$, dove \emptyset è l'insieme vuoto di formule. Infatti se Γ è un insieme di formule, intuitivamente per verificare che valga $\mathcal{I} \models \Gamma$ dobbiamo prendere una ad una le formule $P \in \Gamma$ e verificare che $\mathcal{I} \models P$. Ma se $\Gamma = \emptyset$ non c'è niente da verificare, e quindi $\mathcal{I} \models \emptyset$ vale automaticamente (o, come si dice in questo contesto, *vacuamente*).

Dalla Definizione 9.1.10 segue che possiamo definire l'equivalenza logica in termini della conseguenza logica, come enunciato nella seguente proposizione di cui lasciamo la semplice dimostrazione al lettore.

Proposizione 9.1.11 (equivalenza e conseguenza logica). *Date due formule proposizionali P e Q vale che*

$$P \equiv Q \quad \text{se e solo se} \quad \{P\} \models Q \quad \text{e} \quad \{Q\} \models P$$

³In realtà l'interpretazione è una funzione $\mathcal{I}: X \rightarrow \{f, t\}$ dove X è un insieme potenzialmente infinito di simboli, ma ci interessa il suo valore solo per i simboli proposizionali contenuti nella formula considerata.

9.2 Tavole di verità e tautologie

Nell'Esempio 9.1.9 abbiamo visto come calcolare il valore di una formula proposizionale P in un'interpretazione \mathcal{I} , usando le clausole della Definizione 9.1.8 e il significato dei connettivi logici. Le *tavole di verità* permettono di fare questa valutazione in modo più semplice ma equivalente.

Consideriamo di nuovo la formula $((A \wedge B) \vee \neg C)$ e l'interpretazione $\mathcal{I} = \{A \mapsto \text{t}, B \mapsto \text{f}, C \mapsto \text{f}, \dots\}$. Una volta assegnati i valori di verità ai simboli A , B e C , possiamo ricavare i valori di verità associati alle sottoformule $(A \wedge B)$ e $\neg C$ e quindi all'intera formula $((A \wedge B) \vee \neg C)$. Possiamo rappresentare questo ragionamento in modo compatto con la seguente tavola:

A	B	C	$((A \wedge B) \vee \neg C)$
t	f	f	t
(1)	(2)	(1)	(3)

Nella prima riga abbiamo a destra della doppia riga verticale la formula e a sinistra i simboli proposizionali che vi compaiono. Nella seconda riga, sotto i simboli proposizionali abbiamo il valore fissato dall'interpretazione, e sotto i connettivi logici il risultato determinato dal loro significato. Il valore di verità della formula è quello riportato sotto il *connettivo principale*, cioè l'unico che non compare nella formula come argomento di un altro connettivo: in questo caso è \vee . I numeri in parentesi sotto le colonne indicano l'ordine con cui viene compilata la tavola: si parte dalle colonne dei simboli proposizionali e si applicano i connettivi nell'ordine stabilito dalle parentesi (o dai livelli di precedenza dei connettivi, come vedremo). Nel nostro caso l'ordine è il seguente:

A	B	C	$((A \wedge B) \vee \neg C)$
t	f	f	t
			f

A	B	C	$((A \wedge B) \vee \neg C)$
t	f	f	t
			f

A	B	C	$((A \wedge B) \vee \neg C)$
t	f	f	t
			t
			f

Quante sono le possibili diverse interpretazioni per una data formula proposizionale P ? Considerando solo il valore delle interpretazioni per i simboli proposizionali contenuti in P , è facile convincersi che sono 2^n , dove n è il numero di simboli proposizionali distinti che compaiono in P . Per esempio, per la formula $((A \wedge B) \vee \neg C)$ abbiamo due possibili valori per A , due per B e due per C , per un totale di $2^3 = 8$ distinte interpretazioni. Una formula con 10 simboli proposizionali diversi avrebbe $2^{10} = 1024$ interpretazioni.

Data una formula proposizionale $P \in \mathbf{Prop}$, una *tavola di verità* per P elenca, una per riga, tutte le possibili interpretazioni \mathcal{I} per P e il corrispondente valore di verità $\llbracket P \rrbracket_{\mathcal{I}}$ della formula. Nella Figura 9.1 mostriamo come si può costruire la tavola di verità per la formula $((A \wedge B) \vee \neg C)$, partendo dall'elenco di tutte le possibili interpretazioni e poi annotando progressivamente i connettivi con i valori calcolati.

Osservazione 9.2.1. La grammatica della Definizione 9.1.1 è ambigua, perché ci sono delle stringhe in \mathbf{Prop} che sono testimoniate da alberi di derivazione diversi. Questo significa che valutando una formula in una data interpretazione come appena descritto, ci possiamo trovare nella situazione di dover scegliere quale connettivo valutare prima, ottenendo potenzialmente risultati diversi.

Nel seguito assumiamo che tra i connettivi logici sussistano i seguenti livelli di precedenza (in ordine crescente) e useremo le parentesi quando necessario per indicare esplicitamente l'ordine di valutazione dei connettivi (come si fa usualmente con le espressioni algebriche):

0. Possibili assegnamenti di verità ai simboli proposizionali:

A	B	C	$((A \wedge B) \vee \neg C)$
f	f	f	
f	f	t	
f	t	f	
f	t	t	
t	f	f	
t	f	t	
t	t	f	
t	t	t	

1. Valutazione dei simboli proposizionali nella formula:

A	B	C	$((A \wedge B) \vee \neg C)$
f	f	f	f
f	f	t	t
f	t	f	f
f	t	t	t
t	f	f	f
t	f	t	t
t	t	f	f
t	t	t	t

2. Valutazione delle espressioni più annidate ($(A \wedge B)$ e $\neg C$):

A	B	C	$((A \wedge B) \vee \neg C)$
f	f	f	t f
f	f	t	f t
f	t	f	t f
f	t	t	f t
t	f	f	t f
t	f	t	f t
t	t	f	t f
t	t	t	f t

3. Valutazione delle espressioni più annidate e non ancora valutate:

A	B	C	$((A \wedge B) \vee \neg C)$
f	f	f	t f
f	f	t	f f t
f	t	f	f t t f
f	t	t	f f f t
t	f	f	t f t f
t	f	t	f f f t
t	t	f	t t t f
t	t	t	t f t

connettivo	livello di precedenza
\Leftrightarrow	0
\Rightarrow, \Leftarrow	1
\wedge, \vee	2
\neg	3

Per esempio, se dobbiamo valutare la formula $A \wedge B \Rightarrow C$, poiché \wedge ha precedenza maggiore di \Rightarrow , la valuteremo come se fosse scritta $(A \wedge B) \Rightarrow C$. Nel caso di formule in cui le regole di precedenza non ci aiutano, perché abbiamo due connettivi dello stesso livello, avremo due casi:

- considereremo sintatticamente errate formule come $A \wedge B \vee C$ oppure $A \Rightarrow B \Rightarrow C$, che possono assumere valori diversi per la stessa interpretazione a seconda dell'ordine di valutazione dei connettivi;⁴
- invece accetteremo senza problemi formule come $A \wedge B \wedge C$ oppure $A \vee B \vee C$ che pur essendo sintatticamente ambigue hanno un unico possibile valore per ogni interpretazione.

In caso di dubbio, scrivendo formule complesse si consiglia di abbondare con le parentesi in modo da evitare qualunque rischio di ambiguità.

Esempio 9.2.2 (tavola di verità). Come ulteriore esempio di tavola di verità vediamo quella di una formula leggermente più complessa, dove compaiono anche le costanti \top e \perp : $((A \wedge (B \vee \perp)) \Rightarrow (\neg A \vee (C \wedge \top)))$.

A	B	C	$((A \wedge (B \vee \perp)) \Rightarrow (\neg A \vee (C \wedge \top)))$
f	f	f	f f f f f t t f t f f t
f	f	t	f f f f f t t f t t t t
f	t	f	f f t t f t t f t f f t
f	t	t	f f t t f t t f t t t t
t	f	f	t f f f f t f t f f f t
t	f	t	t f f f f t f t t t t t
t	t	f	t t t t f f f t f f f t
t	t	t	t t t t f t f t t t t t
			(1) (3) (1) (2) (1) (4) (2) (1) (3) (1) (2) (1)

Esercizio 9.2.3. Costruire le tavole di verità per $(A \wedge B) \vee C$ e $A \wedge (B \vee C)$. Per quali interpretazioni le due formule differiscono?

Esercizio 9.2.4. Costruire le tavole di verità per $(A \Rightarrow B) \Rightarrow C$ e $A \Rightarrow (B \Rightarrow C)$. Per quali interpretazioni le due formule differiscono?

Le tautologie

Abbiamo visto che il valore di verità di una formula proposizionale dipende, in generale, dall'interpretazione dei suoi simboli proposizionali. Tuttavia nel calcolo proposizionale ci interessano in particolar modo le *tautologie*, cioè le formule che risultano vere per qualunque interpretazione: esse infatti, come vedremo in seguito, ci permettono di rappresentare schemi di inferenza o di ragionamento corretti.

Definizione 9.2.5 (tautologie, contraddizioni, formule soddisfacibili). Una TAUTOLOGIA è una formula proposizionale che è sempre vera, per qualunque interpretazione. Sfruttando la notazione introdotta nella Definizione 9.1.10, se P è una tautologia scriviamo semplicemente

$$\models P \quad (P \text{ è una tautologia})$$

⁴Lasciamo come esercizio per il lettore il compito di trovare tali interpretazioni.

Una **CONTRADDIZIONE** (o formula insoddisfacibile) è una formula proposizionale che è sempre falsa, per qualunque interpretazione. Una formula proposizionale è **SODDISFACIBILE** se esiste almeno una interpretazione per la quale è vera.

Si osservi che la notazione “ $\models P$ ” introdotta per indicare che P è una tautologia non è altro che un’abbreviazione sintattica di “ $\emptyset \models P$ ”, che per la Definizione 9.1.10 significa che P è conseguenza logica dell’insieme vuoto di formule. Infatti $\emptyset \models P$ se e solo se (per definizione di conseguenza logica) P è vera in tutte le interpretazioni che rendono vere tutte le formule in \emptyset , se e solo se P è vera in tutte le interpretazioni, se e solo se P è una tautologia.

Dalla definizione segue che nella tavola di verità di una formula proposizionale la colonna sotto il connettivo principale conterrà tutti **t** se la formula è una tautologia, tutti **f** se è una contraddizione, e almeno un **t** se è soddisfacibile.

Esempio 9.2.6 (tautologie, contraddizioni e formule soddisfacibili). La seguente tavola di verità mostra che la formula $A \wedge B \Rightarrow B$ è una tautologia (e quindi anche soddisfacibile), la formula $A \wedge (B \wedge \neg A)$ è una contraddizione, mentre la formula $A \Rightarrow B$ è soddisfacibile ma non è una tautologia.

A	B	$(A \wedge B)$	\Rightarrow	B	A	\wedge	$(B \wedge \neg A)$	\Rightarrow	B
f	f	f	f	f	f	f	f	f	f
f	t	f	f	t	f	f	t	t	f
t	f	t	f	f	t	f	f	f	f
t	t	t	t	t	t	f	t	f	t
		(1)	(2)	(1)	(3)	(1)	(4)	(1)	(1)

Un problema fondamentale del calcolo proposizionale è quello di dimostrare che una data formula è una tautologia. Molti altri problemi interessanti si possono ridurre a questo. Per esempio, se dobbiamo dimostrare che la formula A è una contraddizione, ci basta dimostrare che $\neg A$ è una tautologia; se dobbiamo dimostrare che, assumendo che le formule A e B siano vere, allora anche C è vera (cioè che C è una conseguenza logica di $\{A, B\}$), ci basta dimostrare che $A \wedge B \Rightarrow C$ è una tautologia.

Per quanto visto sopra, per vedere se una formula è una tautologia sarebbe sufficiente costruire la sua tavola di verità e controllare che il valore della formula sia **t** su tutte le righe (quindi per ogni interpretazione). Questo procedimento, anche se può essere completamente automatizzato, può richiedere la costruzione di una tavola molto grande (come visto, il numero di righe è 2^n , dove n è il numero di simboli proposizionali della formula) ed è molto soggetto ad errori se fatto a mano. Pertanto nel seguito useremo solo raramente questa tecnica, e vedremo invece nella Sezione 9.3 come dimostrare che una formula è una tautologia usando le dimostrazioni per sostituzione introdotte nella Sezione 1.4.

Se invece dobbiamo mostrare che una formula è soddisfacibile, non c’è bisogno di costruire tutta la tavola di verità, ma è sufficiente trovare una singola interpretazione che renda la formula vera. Spesso questo può essere fatto in modo abbastanza efficiente ragionando sulla struttura della formula. Analogamente, per mostrare che una formula *non* è una tautologia è sufficiente trovare una interpretazione che renda la formula falsa.

Esempio 9.2.7 (formula non tautologica). Mostriamo, senza costruire la tavola di verità, che la formula $((A \Rightarrow \neg B) \wedge \neg A) \Rightarrow B$ non è una tautologia, ovvero:

$$\not\models ((A \Rightarrow \neg B) \wedge \neg A) \Rightarrow B$$

Sfruttiamo la struttura della formula per costruire un’interpretazione che renda falsa la formula. Per cominciare, osserviamo che il connettivo principale della formula è un’implicazione. L’unico caso in cui un’implicazione è falsa è quando la premessa è vera e la conseguenza è falsa (si veda la Definizione 9.1.6). Quindi l’interpretazione cercata deve associare **f** a B , la conseguenza, e **t** alla sottoformula $((A \Rightarrow \neg B) \wedge \neg A)$, la premessa. A sua volta quest’ultima formula ha come connettivo principale una congiunzione, che è vera solo se entrambi gli argomenti sono veri. Quindi in

particolare $\neg A$ deve valere t , cioè l'interpretazione deve associare f a A . Abbiamo quindi individuato l'interpretazione $\{A \mapsto f, B \mapsto f\}$, che per il ragionamento fatto è l'unica che potrebbe rendere la formula falsa. Valutando l'intera formula in questa interpretazione otteniamo effettivamente f (cioè falso), come si vede dalla seguente tavola:

A	B	$((A \Rightarrow \neg B) \wedge \neg A) \Rightarrow B$							
f	f	f	t	t	f	t	t	f	f
		(1)	(3)	(2)	(1)	(4)	(2)	(1)	(5)

Formalizzazione di inferenze e tautologie

Si può sfruttare la formalizzazione di proposizioni per mostrare la correttezza di inferenze o di semplici ragionamenti espressi in linguaggio naturale. Un'inferenza è corretta se la tesi è una conseguenza logica delle ipotesi, e questo si può accertare dimostrando che l'implicazione “*Ipotesi \Rightarrow Tesi*” è una tautologia. Vediamo due esempi: un'inferenza corretta e una errata.

Esempio 9.2.8 (dimostrazione di correttezza di un'inferenza). *Si dimostri la correttezza della seguente inferenza:*

“Studio oggi oppure domani, ma domani non studio, quindi studio oggi”

Per formalizzare la frase, introduciamo un simbolo proposizionale per ogni proposizione elementare, quindi SO per “studio oggi” e SD per “studio domani”.

Ora costruiamo una formula proposizionale che usa questi simboli e che esprime il significato inteso della frase:

$$(SO \vee SD) \wedge \neg SD \Rightarrow SO$$

Quindi abbiamo rappresentato “oppure” con la disgiunzione \vee , “non” con la negazione \neg , “quindi” con l'implicazione \Rightarrow e “ma” con la congiunzione \wedge . Si noti pure che abbiamo associato in modo abbastanza flessibile le proposizioni della frase alle sottoformule: “domani” è rappresentato da SD poiché sottintende “studio domani”, e “domani non studio” da $\neg SD$. La formula proposizionale ottenuta è una tautologia, come il lettore può facilmente verificare, confermando che l'inferenza è corretta.

Naturalmente non tutte le inferenze sono corrette: vediamo come la formalizzazione logica ci può aiutare a rivelare l'inesattezza di un ragionamento.

Esempio 9.2.9 (dimostrazione di non correttezza di un'inferenza). *È corretta la seguente inferenza?*

“Se studio oggi allora domani non studio, ma oggi non studio, quindi domani studierò”

Come nell'esempio precedente possiamo usare i simboli proposizionali SO per “studio oggi” e SD per “studio domani”. Una formula proposizionale che rappresenta fedelmente il significato della frase è:

$$(SO \Rightarrow \neg SD) \wedge \neg SO \Rightarrow SD$$

Ma come possiamo dedurre dall'Esempio 9.2.7 dove abbiamo analizzato una formula con la stessa struttura (anche se con simboli proposizionali diversi), l'interpretazione $\{SO \mapsto f, SD \mapsto f\}$ rende questa formula falsa, pertanto non è una tautologia. Quindi l'inferenza non è corretta, perché “studio domani” non è una conseguenza logica di “se studio oggi allora domani non studio” e di “oggi non studio”.

9.3 Dimostrazioni, nel calcolo proposizionale e oltre

Abbiamo visto nella Definizione 9.1.10 che una formula P è *conseguenza logica* di un insieme di formule Γ se P è vera in tutti i modelli di Γ . Anche se la abbiamo introdotta per il calcolo proposizionale, questa definizione è sostanzialmente indipendente dalla logica, e in particolare vale anche per la logica dei predicati che vedremo successivamente.

Il concetto di conseguenza logica di per sé non suggerisce alcun metodo pratico per mostrare che una formula P è o non è conseguenza logica di un insieme Γ di formule, o, detto altrimenti, per mostrare che è o meno “legittimo” concludere la conseguenza P dalle premesse Γ . Per mostrare che una formula NON È conseguenza logica di un insieme di premesse, un metodo ragionevole è quello di mostrare che esiste un modello delle premesse in cui la conclusione è falsa. Ciò è quanto abbiamo fatto nell’Esempio 9.2.7 (in quel caso l’insieme delle premesse era vuoto, quindi qualunque interpretazione ne era un modello).

Più problematico è invece stabilire che una formula P è conseguenza logica di un insieme di formule Γ date. Usare direttamente la definizione, cioè analizzare ogni possibile modello di Γ per assicurarsi che in esso P sia vera, sarebbe molto costoso per il calcolo proposizionale e sarebbe semplicemente impossibile per la logica dei predicati e per molte altre logiche, perché i modelli di una formula o di un insieme di formule sono, in generale, infiniti.

È nel concetto di *dimostrazione* che sta la risposta a questo quesito: la dimostrazione (sintattica) di un asserto P a partire da una collezione di asserti dati Γ , le premesse, mostra che P è conseguenza logica di Γ . A patto di disporre di un insieme di *regole di inferenza* sufficientemente “potenti”, è possibile ricondurre il concetto semantico di conseguenza logica al concetto puramente sintattico di dimostrazione. Infatti una dimostrazione è una sequenza di passi di pura manipolazione simbolica degli asserti in gioco, ciascuno corrispondente alla applicazione di una delle regole di inferenza. Intuitivamente, i vari passi di dimostrazione consentono di trarre una serie di conclusioni (conseguenze) intermedie fino ad arrivare alla conclusione desiderata.

Nelle prossime sezioni prima mostriamo come si possono impostare delle dimostrazioni per sostituzione nel calcolo proposizionale, dopo aver presentato un insieme opportuno di leggi. Poi introduciamo il concetto astratto di *sistema di dimostrazioni* e ne vediamo un’istanza concreta per il calcolo proposizionale, che permette di collocare le dimostrazioni per sostituzione in un contesto più formale. Infine discuteremo di come nel calcolo proposizionale è possibile ragionare sulla correttezza di semplici tecniche di dimostrazione.

Dimostrazione di tautologie nel Calcolo Proposizionale

Una *dimostrazione per sostituzione* nel calcolo proposizionale ha lo scopo di mostrare che una formula del tipo $P \Leftrightarrow Q$ è una tautologia, e si sviluppa come una sequenza di passi $P \Leftrightarrow R_1 \Leftrightarrow \dots \Leftrightarrow R_n \Leftrightarrow Q$ ognuno dei quali consiste nell’applicare una legge o una tautologia già dimostrata precedentemente.

Questo concetto di “applicare una legge per giustificare un passo” merita di essere reso più formale. Per prima cosa introduciamo una notazione.

Notazione 9.3.1 (rimpiazzamento). Siano P , Q e R formule proposizionali. Allora $P [^Q/_R]$ è definito essere la formula ottenuta da P rimpiazzando una specifica occorrenza della sottoformula R con la formula Q . Per esempio, abbiamo

$$(A \wedge B \Rightarrow (C \Rightarrow D)) [^{\neg C \vee D}/_{C \Rightarrow D}] := (A \wedge B \Rightarrow (\neg C \vee D))$$

dove $:=$ indica che il membro sinistro è definito essere il membro destro. Se la formula R occorre più volte come sottoformula in P assumeremo che sia indicata esplicitamente (per esempio sottolineandola) l’occorrenza da rimpiazzare. Inoltre se necessario aggiungeremo delle parentesi per evitare che la formula risultante sia ambigua. Per esempio,

$$\neg(A \Rightarrow B) \vee (\underline{B} \wedge C) [^{B \vee D}/_B] := \neg(A \Rightarrow B) \vee ((B \vee D) \wedge C)$$

La seguente regola di inferenza, il *Principio di sostituzione*, stabilisce che se $Q \Leftrightarrow R$ è una tautologia, allora anche $P \Leftrightarrow P [^R/_Q]$ è una tautologia.

Principio di sostituzione
$\frac{Q \Leftrightarrow R}{P \Leftrightarrow P [^R/_Q]} \text{ [PDS]}$

Ogni passo di una dimostrazione per sostituzione sarà di fatto una doppia implicazione giustificata da una legge grazie al principio di sostituzione. Per esempio, la commutatività della disunione

è stabilita dalla legge “*commutatività*: $P \vee Q \Leftrightarrow Q \vee P$ ”. Possiamo applicarla alla formula $((A \wedge B) \vee C \Rightarrow D)$ ottenendo il seguente passo di dimostrazione:

$$((A \wedge B) \vee C \Rightarrow D) \Leftrightarrow (C \vee (A \wedge B) \Rightarrow D) \quad (\text{commutatività})$$

In questo caso abbiamo sottolineato la sottoformula cui abbiamo applicato la legge, e come usuale non abbiamo menzionato esplicitamente l’uso del principio di sostituzione.

Solo per questo esempio, facciamo vedere esplicitamente che il passo di dimostrazione è *una istanza* della regola di inferenza **[PDS]**, nel senso che si ottiene da essa applicando un semplice rimpiazzamento. Per prima cosa osserviamo che $P \vee Q \Leftrightarrow Q \vee P$, come ogni legge, rappresenta *un’infinità di tautologie*, tutte quelle ottenibili sostituendo P e Q con arbitrarie formule.⁵ Quindi in particolare anche $(A \wedge B) \vee C \Leftrightarrow C \vee (A \wedge B)$ è una tautologia. A questo punto, sostituendo in **[PDS]** la P con $((A \wedge B) \vee C \Rightarrow D)$, la Q con $(A \wedge B) \vee C$ e la R con $C \vee (A \wedge B)$, otteniamo

$$\frac{(A \wedge B) \vee C \Leftrightarrow C \vee (A \wedge B)}{((A \wedge B) \vee C \Rightarrow D) \Leftrightarrow ((A \wedge B) \vee C \Rightarrow D)^{[C \vee (A \wedge B) / (A \wedge B) \vee C]}} \quad [\text{PDS}]$$

dove la doppia implicazione sotto la riga, effettuato il rimpiazzamento, è esattamente il passo di dimostrazione di sopra.

Naturalmente abbiamo bisogno di un insieme di tautologie da cui partire, verificate indipendentemente, per esempio usando le tavole di verità. Chiameremo *assiomi* queste tautologie, o anche *leggi* come avviene in altri contesti. Come il lettore noterà immediatamente, alcune di queste leggi sono analoghe a quelle per l’uguaglianza di insiemi mostrate nella Sezione 1.4, facendo corrispondere F all’insieme vuoto \emptyset , T all’universo \mathcal{U} , la negazione al complemento, la congiunzione all’intersezione, e la disgiunzione all’unione.

Proposizione 9.3.2 (alcune leggi del calcolo proposizionale). *Per tutte le formule proposizionali P , Q e R le formule delle Tabelle 9.1, 9.2 e 9.3 sono tautologie.*⁶

unità	$P \vee \mathsf{F} \Leftrightarrow P$	$P \wedge \mathsf{T} \Leftrightarrow P$
assorbimento	$P \vee \mathsf{T} \Leftrightarrow \mathsf{T}$	$P \wedge \mathsf{F} \Leftrightarrow \mathsf{F}$
idempotenza	$P \vee P \Leftrightarrow P$	$P \wedge P \Leftrightarrow P$
commutatività	$P \vee Q \Leftrightarrow Q \vee P$	$P \wedge Q \Leftrightarrow Q \wedge P$
associatività	$P \vee (Q \vee R) \Leftrightarrow (P \vee Q) \vee R$	$P \wedge (Q \wedge R) \Leftrightarrow (P \wedge Q) \wedge R$
distributività	$P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee R)$	$P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)$

Tabella 9.1: Leggi per disgiunzione e congiunzione

$\mathsf{T} : \mathsf{F}$	$\neg \mathsf{T} \Leftrightarrow \mathsf{F}$
doppia negazione	$\neg(\neg P) \Leftrightarrow P$
terzo escluso	$P \vee \neg P \Leftrightarrow \mathsf{T}$
contraddizione	$P \wedge \neg P \Leftrightarrow \mathsf{F}$
De Morgan	$\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$
	$\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$

Tabella 9.2: Leggi per negazione

Le leggi elencate descrivono alcune proprietà algebriche dei connettivi logici (vedi quelle per congiunzione, disgiunzione e negazione), oppure consentono di eliminare un connettivo sostituendolo con una opportuna combinazione di altri connettivi. Grazie alla legge di simmetria della doppia

⁵Lo stesso vale per le identità algebriche: per esempio $x \cdot (y + z) = x \cdot y + x \cdot z$ rappresenta infinite uguaglianze, ottenute sostituendo le variabili con arbitrarie espressioni.

⁶Per semplicità di esposizione, non ci preoccupiamo di introdurre un insieme minimale di leggi, quindi alcune potrebbero essere dimostrate con quelle introdotte precedentemente.

riflessività	$P \Leftrightarrow P$
simmetria	$(P \Leftrightarrow Q) \Leftrightarrow (Q \Leftrightarrow P)$
eliminazione dell'implicazione	$P \Rightarrow Q \Leftrightarrow \neg P \vee Q$
eliminazione dell'implicazione negata	$\neg(P \Rightarrow Q) \Leftrightarrow P \wedge \neg Q$
eliminazione della doppia implicazione (1)	$(P \Leftrightarrow Q) \Leftrightarrow (P \Rightarrow Q) \wedge (Q \Rightarrow P)$
eliminazione della doppia implicazione (2)	$(P \Leftrightarrow Q) \Leftrightarrow (P \wedge Q) \vee (\neg P \wedge \neg Q)$

Tabella 9.3: Leggi di altri connettivi e di eliminazione

implicazione ($(P \Leftrightarrow Q) \Leftrightarrow (Q \Leftrightarrow P)$), ogni legge può essere usata in entrambe le direzioni (sostituendo in un certo contesto il membro destro con il sinistro oppure il sinistro con il destro).

Per ognuna delle leggi elencate occorrebbe verificare che sia una tautologia: lo mostriamo solo per una legge di De Morgan, lasciando le altre come esercizio per il lettore.

Esempio 9.3.3 (correttezza delle leggi di De Morgan). *Mostriamo che la prima legge di De Morgan è una tautologia (la dimostrazione per la seconda è del tutto analoga), usando la tavola di verità della formula.*

P	Q	\neg	$(P \vee Q)$	\Leftrightarrow	\neg	P	\wedge	\neg	Q
f	f	t	f	f	t	t	f	t	t
f	t	f	f	t	t	t	f	f	f
t	f	f	t	f	t	f	t	f	t
t	t	f	t	t	t	f	t	f	t
		(3)	(1)	(2)	(1)	(4)	(2)	(1)	(3)
						(1)	(3)	(2)	(1)

Come si vede dalla colonna (4) la formula è una tautologia.

Sfruttando le leggi introdotte siamo ora in grado di dimostrare che altre formule sono tautologie. È importante sottolineare che ogni tautologia può essere usata a sua volta come giustificazione in dimostrazioni successive, con un meccanismo simile alla costruzione delle dimostrazioni di teoremi in matematica: se proviamo separatamente alcuni lemmi, essi possono essere utilizzati senza bisogno di ri-dimostrarli nella dimostrazione del teorema principale.

Per dimostrare una tautologia $P \Leftrightarrow Q$, come anticipato possiamo partire da P e cercare di trasformarla in Q con una sequenza di passi $P \Leftrightarrow R_1 \Leftrightarrow \dots \Leftrightarrow R_n \Leftrightarrow Q$ ognuno dei quali giustificato da una legge o da una tautologia già dimostrata. Il fatto che $P \Leftrightarrow Q$ sia una tautologia segue dalla transitività di \Leftrightarrow . Ovviamente dato che \Leftrightarrow è simmetrica possiamo anche partire da Q per arrivare a P , oppure possiamo ridurre sia P che Q ad una terza formula R equivalente ad entrambe.

Esempio 9.3.4 (complemento e assorbimento). *Dimostriamo che le seguenti formule sono tautologie:*

complemento	$P \vee (\neg P \wedge Q) \Leftrightarrow P \vee Q$	$P \wedge (\neg P \vee Q) \Leftrightarrow P \wedge Q$
assorbimento	$P \vee (P \wedge Q) \Leftrightarrow P$	$P \wedge (P \vee Q) \Leftrightarrow P$

Consideriamo la prima legge del complemento. Una buona strategia, che in questo caso funziona, consiste nel partire dalla formula più complessa e nell'usare le leggi per semplificarla finché non si ottiene l'altra. Partiamo quindi da $P \vee (\neg P \wedge Q)$, sottolineando la porzione di formula alla quale viene applicata la legge indicata nella giustificazione, a meno che non sia l'intera formula.

$$\begin{aligned}
 P \vee (\neg P \wedge Q) &\Leftrightarrow \underline{(P \vee \neg P) \wedge (P \vee Q)} \quad (\text{distributività}) \\
 &\Leftrightarrow \mathsf{T} \wedge (P \vee Q) \quad (\text{terzo escluso}) \\
 &\Leftrightarrow (P \vee Q) \wedge \mathsf{T} \quad (\text{commutatività}) \\
 &\Leftrightarrow (P \vee Q) \quad (\text{unità})
 \end{aligned}$$

In modo del tutto analogo si può dimostrare la seconda legge del complemento, nella quale la congiunzione e la disgiunzione giocano un ruolo simmetrico.

Vediamo invece che la stessa strategia non funziona con la prima legge dell'assorbimento, $P \vee (P \wedge Q) \Leftrightarrow P$. Infatti, partendo dal membro sinistro verrebbe naturale applicare le leggi nella sequenza che segue:

$$\begin{aligned} P \vee (P \wedge Q) &\Leftrightarrow \underline{(P \vee P)} \wedge (P \vee Q) && (\text{distributività}) \\ &\Leftrightarrow P \wedge (P \vee Q) && (\text{idempotenza}) \end{aligned}$$

Si noti che abbiamo ottenuto il membro sinistro della seconda legge dell'assorbimento, ed è facile intuire che procedendo allo stesso modo otterremmo la formula da cui eravamo partiti, chiudendo uno sterile ciclo. Quella che segue invece è una dimostrazione corretta della legge, come è facile verificare, ma essa usa in modo non ovvio le leggi dell'unità e della distributività al contrario.

$$\begin{aligned} \underline{P \vee (P \wedge Q)} &\Leftrightarrow (P \wedge \top) \vee (P \wedge Q) && (\text{unità}), \text{al contrario} \\ &\Leftrightarrow P \wedge (\top \vee Q) && (\text{distributività}), \text{al contrario} \\ &\Leftrightarrow P \wedge \top && (\text{commutatività}) \text{ e } (\text{assorbimento}) \\ &\Leftrightarrow P && (\text{unità}) \end{aligned}$$

Riassumendo, spesso una buona strategia consiste nel partire dalla formula più complessa e nell'usare le leggi per semplificarla fino a ottenere la formula equivalente cercata. Ma non sempre questa strategia funziona: in tal caso occorre un pizzico di intuizione per impostare una dimostrazione concisa e corretta.

Esempio 9.3.5 (contronominale). *Dimostriamo la seguente tautologia:*

contronominale	$P \Rightarrow Q \Leftrightarrow \neg Q \Rightarrow \neg P$
----------------	---

Partiamo dalla formula più complessa, che in questo caso è il membro destro:

$$\begin{aligned} \neg Q \Rightarrow \neg P &\Leftrightarrow \underline{\neg(\neg Q)} \vee \neg P && (\text{eliminazione dell'implicazione}) \\ &\Leftrightarrow Q \vee \neg P && (\text{doppia negazione}) \\ &\Leftrightarrow \neg P \vee Q && (\text{commutatività}) \\ &\Leftrightarrow P \Rightarrow Q && (\text{eliminazione dell'implicazione}), \text{al contrario} \end{aligned}$$

Nel seguito, come da prassi consolidata, tenderemo a semplificare le dimostrazioni non indicando esplicitamente i passaggi nei quali applichiamo le leggi della commutatività, associatività e idempotenza.

Esempio 9.3.6 (le dimostrazioni non sono uniche). *Dimostriamo ora la tautologia*

$$\neg(P \vee (\neg P \wedge Q)) \Leftrightarrow \neg P \wedge \neg Q$$

Nei riquadri che seguono mostriamo tre diverse dimostrazioni per sostituzione che partono dal membro sinistro. Come si vede, scelte diverse delle leggi da applicare nei vari passi portano a dimostrazioni di lunghezza diversa.

La dimostrazione a sinistra applica in modo sistematico le leggi per semplificare la formula a partire dall'operatore più esterno, la negazione. La dimostrazione in alto a destra invece posticipa l'uso delle leggi di De Morgan, applicandole solo quando la sottoformula più interna è stata semplificata. Infine nella dimostrazione in basso a destra abbiamo riconosciuto che si può applicare, con una opportuna sostituzione, la legge del complemento dimostrata precedentemente, il che permette di ridurre la dimostrazione a due soli passi.

$$\begin{aligned}
 & \neg(P \vee (\neg P \wedge Q)) \\
 \Leftrightarrow & (De\ Morgan) \\
 & \neg P \wedge \underline{\neg(\neg P \wedge Q)} \\
 \Leftrightarrow & (De\ Morgan) \\
 & \neg P \wedge (\underline{\neg(\neg P)} \vee \neg Q) \\
 \Leftrightarrow & (doppia\ negazione) \\
 & \neg P \wedge (P \vee \neg Q) \\
 \Leftrightarrow & (distributività) \\
 & \underline{(\neg P \wedge P) \vee (\neg P \wedge \neg Q)} \\
 \Leftrightarrow & (contraddizione) \\
 & \mathbf{F} \vee (\neg P \wedge \neg Q) \\
 \Leftrightarrow & (unità) \\
 & \neg P \wedge \neg Q
 \end{aligned}$$

$$\begin{aligned}
 & \neg(P \vee (\neg P \wedge Q)) \\
 \Leftrightarrow & (distributività) \\
 & \neg((P \vee \neg P) \wedge (P \vee Q)) \\
 \Leftrightarrow & (terzo\ escluso) \\
 & \neg(\mathbf{T} \wedge (P \vee Q)) \\
 \Leftrightarrow & (unità) \\
 & \neg(P \vee Q) \\
 \Leftrightarrow & (De\ Morgan) \\
 & \neg P \wedge \neg Q
 \end{aligned}$$

$$\begin{aligned}
 & \neg(P \vee (\neg P \wedge Q)) \\
 \Leftrightarrow & (complemento) \\
 & \neg(P \vee Q) \\
 \Leftrightarrow & (De\ Morgan) \\
 & \neg P \wedge \neg Q
 \end{aligned}$$

Come ultimo esempio di questa sezione vediamo la dimostrazione di una tautologia che non è una doppia implicazione, sfruttando il fatto che una formula P è una tautologia se e solo se lo è $P \Leftrightarrow \mathbf{T}$.

Esempio 9.3.7 (Modus Ponens). *Dimostrare che la formula $((P \Rightarrow Q) \wedge P) \Rightarrow Q$, chiamata Modus Ponens, è una tautologia. Mostriamo che è equivalente a \mathbf{T} .*

$$\begin{aligned}
 ((P \Rightarrow Q) \wedge P) \Rightarrow Q & \Leftrightarrow ((\neg P \vee Q) \wedge P) \Rightarrow Q && (\text{eliminazione dell'implicazione}) \\
 & \Leftrightarrow Q \wedge P \Rightarrow Q && (\text{complemento}) \\
 & \Leftrightarrow \neg(Q \wedge P) \vee Q && (\text{eliminazione dell'implicazione}) \\
 & \Leftrightarrow (\neg Q \vee \neg P) \vee Q && (De\ Morgan) \\
 & \Leftrightarrow \mathbf{T} \vee \neg P && (\text{commutatività}), (\text{associatività}), (\text{terzo\ escluso}) \\
 & \Leftrightarrow \mathbf{T} && (\text{assorbimento})
 \end{aligned}$$

I sistemi di dimostrazioni

In questa sezione vogliamo mostrare che le dimostrazioni per sostituzione di tautologie appena viste sono basate formalmente su di un *sistema di dimostrazioni* per il calcolo proposizionale. Ne approfittiamo per introdurre questo concetto, che si può applicare a qualunque logica, nella sua generalità.

Un sistema di dimostrazioni per una data logica stabilisce in quale modo si possono costruire delle dimostrazioni partendo da un insieme di premesse e applicando le regole di inferenza disponibili. L'obiettivo di una dimostrazione è in generale di mostrare che una data formula è conseguenza logica delle premesse.

Definizione 9.3.8 (sistema di dimostrazioni). *Dato un insieme di formule Δ (letto “Delta”), un SISTEMA DI DIMOSTRAZIONI (in inglese PROOF SYSTEM) per Δ è un insieme di REGOLE DI INFERENZA \mathcal{R} . Una regola di inferenza $r \in \mathcal{R}$ ha la struttura:*

$$\frac{P_1 \quad \dots \quad P_n}{P} [r]$$

dove P è la CONSEGUENZA e P_1, \dots, P_n sono le PREMESSE, per $n \geq 0$. Se $n = 0$ la regola è chiamata anche un ASSIOMA, altrimenti se $n > 0$ è chiamata anche una regola di inferenza PROPRIA.

Possiamo leggere una regola di inferenza come quella mostrata così: *se abbiamo una dimostrazione per le formule P_1, \dots, P_n , allora abbiamo anche una dimostrazione per P .* Equivalentemente, possiamo leggerla come *per dimostrare P è sufficiente dimostrare P_1, \dots, P_n .* Si può applicare questa lettura per esempio alla regola **[PDS]** presentata sopra per il calcolo proposizionale, che ha una sola premessa: *“se abbiamo una dimostrazione per $Q \Leftrightarrow R$, allora abbiamo una dimostrazione per $P \Leftrightarrow P [R/Q]$ ”.*

Definizione 9.3.9 (dimostrazione). Una DIMOSTRAZIONE in un proof system \mathcal{R} di una formula $Q \in \Delta$ a partire da un insieme di premesse $\Gamma \subseteq \Delta$ è una sequenza di formule Q_1, Q_2, \dots, Q_n in cui

- (1) ogni formula Q_i è un elemento di Γ oppure è ottenuta applicando una regola di inferenza di \mathcal{R} a partire dalle formule in Γ o in Q_1, \dots, Q_{i-1} ;
- (2) Q_n è proprio Q .

Se esiste una dimostrazione di Q a partire da Γ in \mathcal{R} scriveremo

$$\Gamma \vdash_{\mathcal{R}} Q \quad (Q \text{ è dimostrabile da } \Gamma \text{ (in } \mathcal{R}))$$

La (1) mette in luce il fatto che, in una dimostrazione, le premesse che si possono utilizzare in un passo sono non solo le premesse date Γ , ma anche tutte le formule derivate nei passi di dimostrazione precedenti.

Il concetto di dimostrazione è puramente sintattico. La adeguatezza di un sistema di dimostrazioni nel rappresentare sintatticamente il concetto semantico di conseguenza logica può essere espressa dalle sue caratteristiche di *correttezza* e *completezza*.

Definizione 9.3.10 (correttezza e completezza). Un sistema di dimostrazioni \mathcal{R} per Δ è detto CORRETTO se per ogni formula $P \in \Delta$ e ogni insieme di formule $\Gamma \subseteq \Delta$

$$\Gamma \vdash_{\mathcal{R}} P \quad \text{implica} \quad \Gamma \models P \quad (\text{correttezza})$$

cioè se consente di derivare solo conclusioni che sono effettivamente conseguenze logiche delle premesse date. La correttezza è un requisito che qualunque sistema di dimostrazioni deve soddisfare.

Un sistema di dimostrazioni \mathcal{R} è COMPLETO se per ogni formula $P \in \Delta$ e ogni insieme di formule $\Gamma \subseteq \Delta$

$$\Gamma \models P \quad \text{implica} \quad \Gamma \vdash_{\mathcal{R}} P \quad (\text{completezza})$$

cioè consente di dimostrare una formula a partire da un insieme di premesse se la prima è conseguenza logica del secondo.

Torniamo ora a parlare del calcolo proposizionale. Un sistema di dimostrazioni per tale logica ha lo scopo di dimostrare che due formule proposizionali P e Q sono logicamente equivalenti ($P \equiv Q$), oppure che una formula P è conseguenza logica di un insieme di formule Γ ($\Gamma \models P$). Per questo scopo giocano un ruolo fondamentale le tautologie: per dimostrare una equivalenza o una conseguenza logica è sufficiente dimosticare che una opportuna formula proposizionale è una tautologia. Infatti abbiamo il seguente risultato:

Proposizione 9.3.11 (equivalenza e conseguenza logica come tautologie). Siano P e Q formule proposizionali, e $\Gamma = \{P_1, \dots, P_n\}$ un insieme finito di formule proposizionali.⁷ Allora

1. $P \equiv Q$ se e solo se $P \Leftrightarrow Q$ è una tautologia.
2. $\Gamma \models Q$ se e solo se $P_1 \wedge \dots \wedge P_n \Rightarrow Q$ è una tautologia, dove $\Gamma = \{P_1, \dots, P_n\}$.

Dimostrazione.

1. Abbiamo che $P \Leftrightarrow Q$ è una tautologia se e solo se è vera per ogni interpretazione (Definizione 9.2.5), quindi se e solo se per ogni interpretazione P e Q assumono lo stesso valore (Definizione 9.1.8), se e solo se $P \equiv Q$ (Definizione 9.1.10).
2. $P_1 \wedge \dots \wedge P_n \Rightarrow Q$ è una tautologia se e solo se è vera per ogni interpretazione, quindi se e solo se ogni interpretazione \mathcal{I} che rende vera la formula $P_1 \wedge \dots \wedge P_n$ rende anche vera Q (Definizione 9.1.8). Ma, per la stessa definizione e per la Definizione 9.1.10, \mathcal{I} rende vera $P_1 \wedge \dots \wedge P_n$ se e solo se \mathcal{I} è un modello di $\{P_1, \dots, P_n\}$, quindi ogni modello di $\{P_1, \dots, P_n\}$ è anche un modello di Q , cioè $\{P_1, \dots, P_n\} \models Q$. ■

⁷Se Γ è un insieme infinito vale un risultato simile ma più complesso da enunciare, che tralasciamo per semplicità.

Questo fatto ci permette di rileggere le dimostrazioni di tautologie presentate nella sezione precedente come dimostrazioni di equivalenze logiche. Quelle dimostrazioni sono di fatto dimostrazioni del sistema che definiamo ora.

Definizione 9.3.12 (Proof system $\mathcal{S}_{\Leftrightarrow}$ per il Calcolo Proposizionale). *Il sistema di dimostrazioni $\mathcal{S}_{\Leftrightarrow}$ sull'insieme di formule **Prop** è costituito dalle regole di inferenza proprie Transitività [**TR**] e Principio di sostituzione [**PDS**] e, come assiomi, dall'insieme di tautologie elencate nella Proposizione 9.3.2.*

Abbiamo già presentato e discusso la regola di inferenza [**PDS**]. La regola [**TR**] è questa:

Transitività
$\frac{P \Leftrightarrow Q \quad Q \Leftrightarrow R}{P \Leftrightarrow R} \text{ [TR]}$

Essa si legge *se abbiamo una dimostrazione per $P \Leftrightarrow Q$ e $Q \Leftrightarrow R$ allora abbiamo anche una dimostrazione per $P \Leftrightarrow R$, oppure per dimostrare $P \Leftrightarrow R$ è sufficiente dimostare $P \Leftrightarrow Q$ e $Q \Leftrightarrow R$ per una qualche formula Q .*

Proposizione 9.3.13 (correttezza di $\mathcal{S}_{\Leftrightarrow}$). *Il sistema di dimostrazioni $\mathcal{S}_{\Leftrightarrow}$ è corretto, cioè per ogni insieme $\Gamma \subseteq \text{Prop}$ e per ogni formula $P \Leftrightarrow Q \in \text{Prop}$, se $\Gamma \vdash_{\mathcal{S}_{\Leftrightarrow}} P \Leftrightarrow Q$ allora $\Gamma \models P \Leftrightarrow Q$.*

Lasciamo la semplice dimostrazione per induzione di questo importante risultato al lettore interessato. Il caso base sfrutta il fatto che gli assiomi in *AX* sono tautologie, mentre il passo induttivo consiste nel dimostare per le due regole di inferenza che se un'interpretazione rende vere le premesse, allora rende vera anche la conseguenza.

Abbiamo affermato che le dimostrazioni per sostituzione della sezione precedente sono dimostrazioni del proof system $\mathcal{S}_{\Leftrightarrow}$, ma il lettore attento potrebbe giustamente obiettare che quelle dimostrazioni non hanno la struttura prescritta dalla Definizione 9.3.9. In effetti una dimostrazione per sostituzione ci permette di presentare in modo più compatto una dimostrazione secondo quella definizione, mostrando solo i passi in cui si applica il Principio di sostituzione (le applicazione della Transitività sono implicite), e evitando di ripetere più volte la stessa formula. Per esempio, riportiamo qui per convenienza del lettore la dimostrazione della prima legge del Complemento (Esempio 9.3.4):

$$\begin{aligned} P \vee (\neg P \wedge Q) &\Leftrightarrow \underline{(P \vee \neg P) \wedge (P \vee Q)} \quad (\text{distributività}) \\ &\Leftrightarrow \top \wedge (P \vee Q) \quad (\text{terzo escluso}) \\ &\Leftrightarrow (P \vee Q) \wedge \top \quad (\text{commutatività}) \\ &\Leftrightarrow (P \vee Q) \quad (\text{unità}) \end{aligned}$$

Usando il formato della Definizione 9.3.9 tale dimostrazione avrebbe il seguente formato: una sequenza di doppie implicazioni, per ognuna delle quali indichiamo la regola di inferenza e le premesse usate, e l'ultima delle quali è la formula dimostrata.

$$\begin{array}{ll} \frac{P \vee (\neg P \wedge Q) \Leftrightarrow (P \vee \neg P) \wedge (P \vee Q)}{(\text{[PDS]}, \text{distributività})} & (9.1) \\ \frac{(P \vee \neg P) \wedge (P \vee Q) \Leftrightarrow \top \wedge (P \vee Q)}{(\text{[PDS]}, \text{terzo escluso})} & (9.2) \\ \frac{\top \wedge (P \vee Q) \Leftrightarrow \top \wedge (P \vee Q)}{(\text{[TR]}, (9.1) \text{ e } (9.2))} & (9.3) \\ \frac{\top \wedge (P \vee Q) \Leftrightarrow (P \vee Q) \wedge \top}{(\text{[PDS]}, \text{commutatività})} & (9.4) \\ \frac{(P \vee Q) \wedge \top \Leftrightarrow (P \vee Q)}{(\text{[TR]}, (9.3) \text{ e } (9.4))} & (9.5) \\ \frac{(P \vee Q) \wedge \top \Leftrightarrow (P \vee Q)}{(\text{[PDS]}, \text{unità})} & (9.6) \\ \frac{P \vee (\neg P \wedge Q) \Leftrightarrow (P \vee Q)}{(\text{[TR]}, (9.5) \text{ e } (9.6))} & (9.7) \end{array}$$

Sempre per la Definizione 9.3.9, poiché nella dimostrazione abbiamo usato come premesse (come giustificazioni dei vari passi) l'insieme $\Gamma = \{\text{distributività}, \text{terzo escluso}, \text{commutatività}, \text{unità}\}$, abbiamo di fatto mostrato che

$$\Gamma \vdash_{\mathcal{S}_{\Leftrightarrow}} P \vee (\neg P \wedge Q) \Leftrightarrow P \vee Q$$

che per la correttezza di $\mathcal{S}_{\Leftrightarrow}$ ci garantisce che la legge del complemento è una conseguenza logica delle leggi in Γ , ma non immediatamente che è una tautologia: in realtà lo è perché abbiamo usato solo tautologie come premesse, come garantito dal seguente risultato.

Proposizione 9.3.14 (tautologie come premesse). *Se $\Gamma \subseteq \text{Prop}$ contiene solo tautologie e $\Gamma \vdash_{\mathcal{S}_{\Leftrightarrow}} P$, allora P è una tautologia.*

Dimostrazione. Per la correttezza del proof system, da $\Gamma \vdash_{\mathcal{S}_{\Leftrightarrow}} P$ possiamo dedurre che $\Gamma \models P$. Quindi P è vera in tutti i modelli di Γ . Ma poiché Γ contiene solo tautologie, qualunque interpretazione è modello di Γ , e quindi di P . Di conseguenza P è una tautologia. ■

Tecniche di dimostrazione e tautologie

Abbiamo visto nella Proposizione 9.3.11 che il calcolo proposizionale permette di internalizzare i concetti di equivalenza e conseguanza logica, rappresentandoli con tautologie. Questo potere espressivo consente di verificare la correttezza di semplici inferenze o tecniche di dimostrazione con il seguente procedimento: si formalizza il “ragionamento” con una formula proposizionale e si verifica se essa è una tautologia. Il ragionamento sarà corretto se e solo se la formula è una tautologia.⁸ Per esempio, consideriamo la seguente tecnica di dimostrazione:

Per dimostrare che $A \equiv B$ si può dimostrare che se vale A allora vale B , e se vale B allora vale A .

È giusto questo modo di ragionare? Sfruttando la Proposizione 9.3.11 possiamo rappresentare questa affermazione con la formula: $((A \Rightarrow B) \wedge (B \Rightarrow A)) \Rightarrow (A \Leftrightarrow B)$, che è una tautologia (lasciamo al lettore la dimostrazione). Questo dimostra che la tecnica di dimostrazione descritta è corretta, come ci aspettavamo.

In generale gli enunciati dei teoremi hanno spesso la struttura “*se valgono certe Ipotesi allora vale questa Tesi*”. Dimostrare un tale teorema significa dimostrare la conseguenza logica $\text{Ipotesi} \models \text{Tesi}$. Vediamo alcune tecniche di dimostrazione, mostrando che sono corrette perché corrispondono a delle tautologie.

Esempio 9.3.15 (dimostrazione diretta e con ipotesi non tautologiche). *Una dimostrazione diretta della conseguenza logica $\text{Ipotesi} \models \text{Tesi}$ consiste nel dimostrare direttamente che l’implicazione $(\text{Ipotesi} \Rightarrow \text{Tesi})$ è una tautologia, usando solo gli assiomi. La conseguenza logica allora vale per la Proposizione 9.3.11.*

Un’alternativa consiste nel cominciare a dimostrare la Tesi, utilizzando quando necessario una o più delle Ipotesi come premesse per proseguire nella dimostrazione. In pratica questo mostra che nel proof system c’è una dimostrazione di Tesi a partire da Ipotesi, cioè $\text{Ipotesi} \vdash \text{Tesi}$, che per la correttezza del proof system garantisce la conseguenza logica cercata.

Esempio 9.3.16 (dimostrazione per assurdo). *Dimostrare l’enunciato “se valgono certe Ipotesi allora vale questa Tesi” per assurdo significa mostrare che “se si assumono per vere le Ipotesi e si nega la Tesi, si ottiene una contraddizione”. Nel calcolo proposizionale possiamo rappresentare l’enunciato del teorema con $(\text{Ipotesi} \Rightarrow \text{Tesi})$, e l’enunciato della dimostrazione per assurdo come $(\text{Ipotesi} \wedge \neg \text{Tesi} \Leftrightarrow \mathbf{F})$, ricordando che una contraddizione è una formula sempre falsa. Quindi dimostrare la correttezza della tecnica di dimostrazione per assurdo consiste nel dimostrare che la seguente formula è una tautologia:*

$$(\text{Ipotesi} \Rightarrow \text{Tesi}) \Leftrightarrow (\text{Ipotesi} \wedge \neg \text{Tesi} \Leftrightarrow \mathbf{F})$$

Infatti, chiamando Ip le Ipotesi, mostriamo che entrambi i membri sono equivalenti alla stessa formula $\neg Ip \vee \text{Tesi}$, e quindi possiamo concludere per transitività.

$$Ip \Rightarrow \text{Tesi} \Leftrightarrow \neg Ip \vee \text{Tesi} \quad (\text{eliminazione dell’implicazione})$$

⁸Avevamo già anticipato questa tecnica negli Esempi 9.2.8 e 9.2.9: ora ne possiamo certificare la correttezza.

$$\begin{aligned}
 & (Ip \wedge \neg Tesi \Leftrightarrow F) \\
 & \Leftrightarrow (Ip \wedge \neg Tesi \wedge F) \vee (\neg(Ip \wedge \neg Tesi) \wedge \neg(F)) \quad (\text{elim. doppia implicazione}) \\
 & \Leftrightarrow F \vee (\neg(Ip \wedge \neg Tesi) \wedge T) \quad (\text{assorbimento}), (F:T) \\
 & \Leftrightarrow \neg Ip \vee \neg(\neg Tesi) \quad (\text{unità}), (\text{De Morgan}) \\
 & \Leftrightarrow \neg Ip \vee Tesi \quad (\text{doppia negazione})
 \end{aligned}$$

Esempio 9.3.17 (dimostrazione per contrapposizione). Dimostrare per contrapposizione l'enunciato “se valgono certe Ipotesi allora vale questa Tesi” consiste nel dimostrare che “se non vale la Tesi allora non valgono le Ipotesi”. Quindi consiste nel dimostrare ($Ipotesi \Rightarrow Tesi$), l'enunciato del teorema, mostrando che vale ($\neg Tesi \Rightarrow \neg Ipotesi$). La correttezza di questa tecnica è garantita dal fatto che la seguente formula contronominale è una tautologia, come mostrato nell'Esempio 9.3.5:

$$(Ipotesi \Rightarrow Tesi) \Leftrightarrow (\neg Tesi \Rightarrow \neg Ipotesi)$$

9.4 Esercizi su calcolo proposizionale

Esercizio 9.4.1. Quali delle seguenti sono proposizioni?

1. “ $2 + 2 = 4$ ”
2. “ $3 \times 5 = 10$ ”
3. “La capitale dell’Australia è Sydney?”
4. “La capitale dell’Australia è Sydney”
5. “A scuola è vietato fumare”
6. “Non fumare a scuola!”
7. “Posso fumare a scuola?”
8. “Non so se posso fumare a scuola”
9. “Che ore sono?”
10. “Esiste un valore che sommato a 1 dà 2”
11. “ $x + 1 = 2$ ”

Esercizio 9.4.2. Dire con quali connettivi possono essere resi i termini evidenziati nelle seguenti proposizioni:

1. “Mario è agile, **ma** Rosario è forte”
2. “Paolo è in campo, **anche se** ha la febbre alta”
3. “Johnny è in Italia **senza** avere il passaporto”
4. “Il fantasma appare nel castello **esattamente** a mezzanotte”

Esercizio 9.4.3 (Quanti connettivi logici esistono [Difficile]). I connettivi logici che abbiamo presentato come funzioni sui booleani nella Definizione 9.1.6 sono quelli più “intuitivi”, ma esistono molte altre possibilità di combinare valori booleani. Per esempio, per analogia con gli insiemi, si potrebbe definire il connettivo \setminus come:

$$(x \setminus y) := (x \wedge \neg y) \tag{9.8}$$

Oppure si potrebbero definire connettivi ternari come

$$(\text{if } x \text{ then } y \text{ else } z) \equiv (x \wedge y) \vee (\neg x \wedge z) \tag{9.9}$$

In generale un connettivo con n argomenti è determinato da una colonna di una tavola di verità con 2^n righe.

1. Quanti possibili connettivi unari diversi esistono?
2. Quanti possibili connettivi binari diversi esistono?
3. Quanti possibili connettivi n -ari esistono?
4. Data la colonna che identifica un qualsiasi connettivo n -ario, definire una tecnica per costruire una formula equivalente che usi solo i connettivi \neg , \wedge e \vee .

Esercizio 9.4.4. Dimostrare per induzione che ogni formula in **Prop** è equivalente a una formula scritta solo con i connettivi \wedge e \neg .

Esercizio 9.4.5. Costruire le tavole di verità degli operatori $(P \setminus Q)$ e $(\text{if } P \text{ then } Q \text{ else } R)$ definiti nei punti (9.8) e (9.9).

Esercizio 9.4.6. Costruire le tavole di verità delle seguenti formule e indicare quali di queste sono tautologie

1. $((P \wedge (Q \vee R)) \Rightarrow (P \vee (Q \wedge R)))$
2. $((P \vee (Q \wedge R)) \Rightarrow (P \wedge (Q \vee R)))$
3. $((P \wedge (Q \Rightarrow R)) \Rightarrow (Q \Rightarrow (P \wedge R)))$
4. $((Q \Rightarrow (P \wedge R)) \Rightarrow (P \wedge (Q \Rightarrow R)))$
5. $((P \Rightarrow (Q \vee R)) \Leftrightarrow (Q \Rightarrow (P \vee R)))$
6. $((\neg(P \wedge Q) \Rightarrow R) \Rightarrow (Q \vee \neg(P \wedge R)))$
7. $\neg(Q \wedge (R \Rightarrow (P \wedge R)))$
8. $((Q \wedge R) \vee ((P \wedge Q) \vee (P \wedge R)))$
9. $\neg((P \vee (Q \vee R)) \Rightarrow (P \wedge (Q \wedge R)))$

Esercizio 9.4.7. Per ognuna delle seguenti formule, determinare se è una tautologia oppure no. Se non è una tautologia fornire un'interpretazione che la rende falsa senza costruire la tavola di verità, altrimenti costruire la tavola.

1. $((P \wedge Q) \vee P) \Leftrightarrow P$
2. $(P \Rightarrow (Q \Rightarrow \neg R))$
3. $(P \Rightarrow (P \vee Q))$
4. $((P \Rightarrow Q) \Rightarrow \neg P)$

Esercizio 9.4.8. Si formalizzino le seguenti proposizioni:

1. “Aldo va al cinema ma Dario no”
2. “Luigi andrà al cinema o andrà al teatro”
3. “Se ho lezione di LMB allora è martedì o è venerdì”
4. “Non puoi montare sulle montagne russe se sei più basso di un metro e se non hai più di 16 anni”

Esercizio 9.4.9. Le seguenti proposizioni in linguaggio naturale esprimono delle implicazioni (semplici o doppie) tra i simboli proposizionali V (“io vado al cinema”) e R (“Tu resti a casa”). Indicare per ognuna di esse una formula proposizionale che la rappresenta.

1. “Io vado al cinema se tu resti a casa”

2. “Io vado al cinema solo se tu resti a casa”
3. “Io vado al cinema se e solo se tu resti a casa”
4. “Perché io vada al cinema è necessario che tu resti a casa”
5. “Perché io vada al cinema è sufficiente che tu resti a casa”
6. “Condizione necessaria e sufficiente perché io vada al cinema è che tu resti a casa”

Esercizio 9.4.10. Dire se le seguenti formule sono tautologie, contraddizioni o soddisfacibili:

1. $((P \wedge Q) \Rightarrow (P \vee Q))$
2. $((P \wedge Q) \Rightarrow P)$
3. $((P \vee Q) \Rightarrow P)$
4. $((((P \vee (Q \vee R)) \wedge (\neg P \vee \neg Q)) \wedge ((\neg P \vee \neg R) \wedge \neg(Q \wedge R)))$

Esercizio 9.4.11. Aldo, Barbara e Carlo sono tre studenti che hanno sostenuto un esame. Ponendo:

- $A = \text{“Aldo ha superato l'esame”}$
- $B = \text{“Barbara ha superato l'esame”}$
- $C = \text{“Carlo ha superato l'esame”}$

determinare le proposizioni composte che traducono le seguenti proposizioni:

1. “Solo Carlo ha superato l'esame”
2. “Solo Aldo non ha superato l'esame”
3. “Solo uno tra Aldo, Barbara e Carlo ha superato l'esame”
4. “Almeno uno tra Aldo, Barbara e Carlo ha superato l'esame”
5. “Almeno due tra Aldo, Barbara e Carlo hanno superato l'esame”
6. “Al più due tra Aldo, Barbara e Carlo hanno superato l'esame”
7. “Esattamente due tra Aldo, Barbara e Carlo hanno superato l'esame”

Esercizio 9.4.12. Aldo, Barbara e Carlo sono gli unici tre membri di una commissione che vota una proposta. Ponendo:

- $A = \text{“Aldo vota a favore”}$
- $B = \text{“Barbara vota a favore”}$
- $C = \text{“Carlo vota a favore”}$

determinare le proposizioni composte che traducono le seguenti proposizioni:

1. “La votazione è stata unanime”
2. “La proposta è passata a maggioranza”
3. “La proposta è stata respinta, ma non all'unanimità”

Esercizio 9.4.13. Sappiamo che “sono ammesse al concorso le persone che sono laureate e che hanno meno di trent'anni o hanno figli” e che:

- “Aldo non è laureato, ha ventisei anni e un figlio”;
- “Barbara è laureata, ha quarant'anni e due figli”;

- “Carlo è laureato, ha trentadue anni e non ha figli”.

Aldo può partecipare al concorso? E Barbara? E Carlo?

Esercizio 9.4.14.

1. Sapendo che “Il colpevole è il cuoco o la cameriera” e che “Il colpevole è l’autista o la cameriera”, possiamo concludere che “Il colpevole è il cuoco o l’autista”? Motivare la risposta.
2. Sapendo che “O la ventola è fuori asse o il meccanismo di calibrazione è alterato” e che “Ho controllato l’allineamento della ventola ed è ok”, possiamo concludere che “il meccanismo di calibrazione è fuori fase”? Motivare la risposta.
3. Sapendo che “Risolvete tutti gli esercizi di queste note e superate l’esame” e che “Non risolvete tutti gli esercizi di queste note” possiamo concludere che “Non superate l’esame”? Motivare la risposta.
4. Sapendo che “Risolvete tutti gli esercizi per casa e superate l’esame” e che “Non risolvete tutti gli esercizi per casa” possiamo concludere che “Superate l’esame”? Motivare la risposta.

Esercizio 9.4.15. Mostrare che le seguenti formule sono tautologie usando dimostrazioni per sostituzione:

1. $((A \wedge B) \Rightarrow (A \vee B))$
2. $(A \Rightarrow (B \Rightarrow (A \wedge B)))$
3. $(A \Rightarrow (\neg A \Rightarrow B))$
4. $((A \Rightarrow B) \Leftrightarrow \neg(A \wedge \neg B))$
5. $((A \vee B) \Leftrightarrow (\neg A \Rightarrow B))$
6. $((A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C)))$
7. $((A \Rightarrow (B \Rightarrow C)) \Leftrightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C)))$
8. $((A \Rightarrow C) \Rightarrow ((B \Rightarrow C) \Rightarrow ((A \vee B) \Rightarrow C)))$

Esercizio 9.4.16. Dimostrare le seguenti equivalenze logiche oppure fornire un controesempio:

1. $((P \vee \neg Q) \equiv (\neg P \wedge Q))$
2. $((P \vee Q) \Rightarrow (P \wedge Q)) \equiv \top$
3. $(\neg P \Rightarrow (P \Rightarrow Q)) \equiv \top$
4. $(P \Rightarrow (Q \Rightarrow R)) \equiv ((P \Rightarrow Q) \Rightarrow R)$
5. $((P \Rightarrow (Q \Rightarrow R)) \Rightarrow ((P \Rightarrow Q) \Rightarrow (P \Rightarrow R))) \equiv \top$
6. $((P \Rightarrow (Q \Rightarrow R)) \Leftrightarrow ((P \Rightarrow Q) \Rightarrow (P \Rightarrow R))) \equiv \top$

9.5 Cenni di logica dei predicati: motivazioni e sintassi

Il calcolo proposizionale introdotto nella prima parte di questo capitolo costituisce il nucleo di tutte le logiche classiche, ma ha un potere espressivo alquanto limitato. Infatti anche se consente di formalizzare in modo soddisfacente la struttura logica di proposizioni anche complesse (come inferenze e dimostrazioni) non fornisce strumenti per rappresentare gli elementi del dominio del discorso, le loro proprietà e le relazioni tra di essi. La *logica dei predicati* (o *logica del primo ordine*) arricchisce il calcolo proposizionale con costrutti sintattici che permettono appunto di esprimere predicati su proprietà e relazioni tra specifici elementi del dominio. Inoltre, con i quantificatori, permette di esprimere che una proprietà vale per tutti gli elementi o per almeno un elemento del dominio.

Nel resto di questa sezione, dopo aver motivato con alcuni esempi l'introduzione di una logica più espressiva, presentiamo la sintassi della logica dei predicati. Vedremo nella Sezione 9.6 come questa logica permette di formalizzare in modo adeguato proposizioni ben più complesse di quelle viste con il calcolo proposizionale. Quindi nella Sezione 9.7 descriveremo come deve essere arricchita la nozione di interpretazione per consentire di associare un valore di verità alle formule predicative, e ne presenteremo la semantica in modo induttivo. Infine presenteremo alcune leggi per i quantificatori nella Sezione 9.8, mostrando anche per questa logica come le dimostrazioni per sostituzione possono essere utilizzate per mostrare l'equivalenza logica di formule.

Sull'espressività della logica dei predicati

L'espressività del calcolo proposizionale è abbastanza limitata. Per esempio, pensiamo di voler formalizzare un enunciato che parla di persone, come

“Se Anna è la mamma di Bruno e Bruno ha figli, allora Anna è nonna”

Usando la tecnica discussa nella Sezione 9.1 (si veda l'Esempio 9.1.4) dobbiamo introdurre un simbolo per ogni proposizione elementare, per esempio *A* per “*Anna è la mamma di Bruno*”, *B* per “*Bruno ha figli*” e *C* per “*Anna è nonna*”. La formula proposizionale risultante sarebbe $((A \wedge B) \Rightarrow C)$: essa evidenzia correttamente la struttura logica (la congiunzione e l'implicazione) ma poiché le persone coinvolte non hanno una rappresentazione esplicita, la formula non permette di capire che Anna è coinvolta sia nella premessa *A* che nella conseguenza *C* dell'implicazione, e che entrambe le premesse *A* e *B* parlano di Bruno. Se poi consideriamo la frase “*Maria è la mamma di Giulia*”, questa è una proposizione elementare e quindi dobbiamo rappresentarla con un nuovo simbolo: non abbiamo la possibilità di mettere in evidenza che essa esprime la stessa relazione tra persone della proposizione rappresentata sopra da *A*.

Quest'analisi rende evidente che, in situazioni molto frequenti, vorremmo poter rappresentare nella sintassi delle formule: (1) le persone coinvolte, (2) le proprietà delle persone (“*aver figli*”, “*essere nonna*”) e (3) le relazioni tra persone (“*essere mamma di*”).

La logica dei predicati permette di rappresentare esplicitamente queste entità, grazie a una sintassi più ricca di quella proposizionale. Infatti, come vedremo, possiamo formalizzare le frasi viste sopra per esempio come

$$((\text{mamma}(\text{Anna}, \text{Bruno}) \wedge \text{haFigli}(\text{Bruno})) \Rightarrow \text{èNonna}(\text{Anna})) \quad \text{e} \quad \text{mamma}(\text{Maria}, \text{Giulia})$$

in modo di gran lunga più espressivo che con le formule proposizionali.

Naturalmente poter parlare di “persone” è solo un caso particolare: potremmo voler esprimere proprietà e/o relazioni su numeri (“7 è un numero primo”, “6 è il doppio di 3”), su frutti (“questa mela è rossa”), su animali (“Fido è un bassotto”), o anche su elementi di tipologie diverse (“se Fido è il cane di Giorgio allora mangia una mela rossa”).

In generale nella logica dei predicati possiamo rappresentare (con dei *simboli di costante* o dei *termini*) gli *elementi* appartenenti a un certo insieme (*dominio*). Di questi elementi possiamo rappresentare delle proprietà e delle relazioni tra essi (con dei *simboli di predicato*). Negli esempi presentati sopra, abbiamo usato i simboli di costante “*Anna*”, “*Bruno*”, “*Maria*” e “*Giulia*”, i simboli di predicato con un solo argomento (detti *unari*) “*haFigli*” ed “*èNonna*” e il simbolo di predicato a due argomenti (detto *binario*) “*mamma*”.

Quantificazione esistenziale e universale

La possibilità di riferire specifici elementi del dominio di interesse in una formula è solo una premessa necessaria per introdurre i *quantificatori*, l'ingrediente più caratterizzante della sintassi della logica dei predicati.

Nella lingua italiana usiamo comunemente pronomi o aggettivi indefiniti che indicano cose o persone senza specificarne con precisione né l'identità né la quantità, come “*tutti*”, “*qualche*”, “*alcuni*”, “*ogni*”, “*nessuno*”, ecc. Per esempio, parlando di persone, possiamo dire “*tutti gli uomini sono mortali*”, “*nessuno pesa più di 100 chili*”, “*qualcuno è più alto di suo padre*”. Esaminiamone il significato:

- *Tutti* (e sinonimi): permette di asserire che una proprietà vale per tutti gli elementi del dominio o “*universo*”, nessuno escluso. Esprime quindi una *quantificazione universale*.
- *Alcuni* (e sinonimi): permette di affermare che una proprietà vale per almeno un elemento del dominio. Non dice né quanti né quali, ma garantisce l'esistenza di almeno un elemento che la soddisfa. Esprime quindi una *quantificazione esistenziale*.
- *Nessuno* (e sinonimi): permette di asserire che una proprietà non vale per alcun elemento del dominio. Attenzione, è l'opposto di *alcuni*, non è l'opposto di *tutti*! Equivale a dire *tutti ... non ...* e quindi quantifica in maniera universale la negazione di un'enunciato.

La logica dei predicati permette di rappresentare esplicitamente le quantificazioni usando le *variabili* e i *quantificatori*.

Le variabili, di solito chiamate x, y, z, \dots , rappresentano generici elementi del dominio e quindi permettono di rappresentare enunciati generici che assumono un valore di verità che dipende da quali elementi vengono associati alle variabili. Per esempio, parlando dei numeri naturali, possiamo scrivere l'enunciato “ *x è un numero primo*”, che possiamo rappresentare con la formula della logica dei predicati $\text{primo}(x)$. Chiaramente la verità di questa formula dipende da quale elemento sostituiamo a x : $\text{primo}(5)$ è vero, mentre $\text{primo}(12)$ è falso.

I quantificatori servono per descrivere in che modo una variabile che compare in una formula deve essere sostituita da elementi del dominio, per verificare se l'intera formula è vera o falsa. Essi hanno la seguente forma, dove in generale la formula P contiene una o più occorrenze della variabile x :

QUANTIFICAZIONE ESISTENZIALE: $(\exists x . P)$, che si legge “*esiste un x tale che P vale*”

QUANTIFICAZIONE UNIVERSALE: $(\forall x . P)$, che si legge “*per ogni x vale P* ”

Per esempio, considerando il dominio dei numeri naturali, $(\exists x . \text{primo}(x))$ si legge “*esiste un x tale che $\text{primo}(x)$ vale*”, e cioè “*esiste un numero naturale che è primo*”. Si noti che mentre $\text{primo}(x)$ essendo una formula generica non ha un valore di verità, $(\exists x . \text{primo}(x))$ lo ha: esso vale “*vero*” se esiste almeno un elemento nel dominio che associato a x rende la formula $\text{primo}(x)$ vera. Quindi $(\exists x . \text{primo}(x))$ è vera.

Invece $(\forall x . \text{primo}(x))$ si legge “*per ogni x vale $\text{primo}(x)$* ”, cioè “*ogni numero naturale è primo*”. Quindi $(\forall x . \text{primo}(x))$ è vera se *ogni elemento d del dominio, se associato a x, rende la formula $\text{primo}(x)$ vera*. Poiché ci sono numeri naturali che non sono primi, la formula è falsa (basta rimpiazzare x con 4).

La sintassi delle formule predicative

La sintassi delle formule proposizionali presentata nella Definizione 9.1.1 è parametrica rispetto a un insieme di *simboli proposizionali* che, come abbiamo visto, ci servono per rappresentare le proposizioni elementari del dominio del discorso. Anche per la logica dei predicati la sintassi delle formule è parametrica: useremo simboli di costante, di predicato e di funzione presi da un alfabeto fissato a priori. Questi simboli non hanno un significato specifico: quando valuteremo le formule per vedere se sono vere o false su di un *dominio di interpretazione*, dovremo dire qual è il loro significato.

Definizione 9.5.1 (alfabeto del primo ordine). *Un ALFABETO DEL PRIMO ORDINE \mathcal{A} è una quadrupla $\mathcal{A} = (\mathcal{C}, \mathcal{F}, \mathcal{P}, \mathcal{V})$ dove:*

1. \mathcal{C} è l'insieme dei SIMBOLI DI COSTANTE;
2. $\mathcal{F} = \{\mathcal{F}_n\}_{n \in \mathbb{N}^+}$ è una famiglia di insiemi di SIMBOLI DI FUNZIONE. Per ogni $n \in \mathbb{N}^+$, \mathcal{F}_n è l'insieme dei SIMBOLI DI FUNZIONE DI ARIETÀ n (o con n argomenti).
3. $\mathcal{P} = \{\mathcal{P}_n\}_{n \in \mathbb{N}}$ è una famiglia di insiemi di SIMBOLI DI PREDICATO. Per ogni $n \in \mathbb{N}$, \mathcal{P}_n è l'insieme dei SIMBOLI DI PREDICATO DI ARIETÀ n (o con n argomenti).
4. \mathcal{V} è l'insieme delle VARIABILI.

Definizione 9.5.2 (Sintassi della logica dei predicati). *Fissato un alfabeto del primo ordine $\mathcal{A} = (\mathcal{C}, \mathcal{F}, \mathcal{P}, \mathcal{V})$, l'insieme delle FORMULE PREDICATIVE **Pred** è il linguaggio generato dalla categoria sintattica $\langle \text{Pred} \rangle$ della seguente grammatica:*

<i>Logica dei prediciati</i>	
$\langle \text{Pred} \rangle$	$\rightsquigarrow \langle \text{Atom} \rangle \mid \neg \langle \text{Atom} \rangle \mid \langle \text{Pred} \rangle \langle \text{OpB} \rangle \langle \text{Pred} \rangle \mid \underline{\langle \forall \langle \text{Var} \rangle . \langle \text{Pred} \rangle} \mid \underline{\langle \exists \langle \text{Var} \rangle . \langle \text{Pred} \rangle}$
$\langle \text{Atom} \rangle$	$\rightsquigarrow \mathbf{T} \mid \mathbf{F} \mid \langle \text{Pred} \rangle \mid \langle \text{Pide}_0 \rangle \mid \underline{\langle \text{Pide}_n \rangle} \overbrace{\langle \text{Term} \rangle, \dots, \langle \text{Term} \rangle}^n, n \in \mathbb{N}^+$
$\langle \text{Term} \rangle$	$\rightsquigarrow \underline{\langle \text{Var} \rangle} \mid \langle \text{Const} \rangle \mid \langle \text{Fide}_n \rangle \overbrace{\langle \text{Term} \rangle, \dots, \langle \text{Term} \rangle}^n, n \in \mathbb{N}^+$
$\langle \text{OpB} \rangle$	$\rightsquigarrow \wedge \mid \vee \mid \Rightarrow \mid \Leftarrow \mid \Leftrightarrow$
$(n \in \mathbb{N})$	$\langle \langle \text{Pide}_n \rangle \rangle = \mathcal{P}_n \quad \langle \langle \text{Fide}_n \rangle \rangle = \mathcal{F}_n \quad \langle \langle \text{Const} \rangle \rangle = \mathcal{C} \quad \langle \langle \text{Var} \rangle \rangle = \mathcal{V}$

Nella grammatica abbiamo sottolineato le produzioni nuove rispetto al calcolo proposizionale. L'ultima linea spiega il ruolo delle quattro componenti dell'alfabeto: per esempio, i simboli di predicato di arietà n possono essere usati dove compare la categoria sintattica $\langle \text{Pide}_n \rangle$, mentre i simboli di costante possono comparire al posto di $\langle \text{Const} \rangle$.

Un TERMINE, cioè un elemento del linguaggio $\langle \langle \langle \text{Term} \rangle \rangle \rangle$, può essere una variabile, un simbolo di costante, oppure un simbolo di funzione di arietà n (per $n \in \mathbb{N}^+$) applicato a esattamente n termini, come descritto dall'ultima produzione. Nel seguito chiameremo **Term** l'insieme $\langle \langle \langle \text{Term} \rangle \rangle \rangle$. Si noti che nei termini non possono comparire simboli di predicato, che invece servono a costruire le *formule*.⁹

Una FORMULA ATOMICA, cioè un elemento del linguaggio $\langle \langle \langle \text{Atom} \rangle \rangle \rangle$, oltre che \mathbf{T} , \mathbf{F} , o una qualunque formula racchiusa tra parentesi, può essere un simbolo di predicato di arità zero (equivalente a un simbolo proposizionale), oppure un simbolo di predicato di arietà n applicato ad esattamente n termini, con $n \in \mathbb{N}^+$.

Una FORMULA PREDICATIVA (o semplicemente FORMULA), cioè una stringa generata da $\langle \text{Pred} \rangle$, può essere costruita in tutti i modi visti con il calcolo proposizionale e in più può essere una FORMULA QUANTIFICATA della forma $(\forall x . P)$ oppure $(\exists x . P)$, dove x è una variabile e P una formula.

Osservazione 9.5.3. Si noti che tutte le formule del calcolo proposizionale sono anche formule predicative, considerando i simboli proposizionali come simboli di predicato di arietà zero, cioè elementi di \mathcal{P}_0 .

Naturalmente anche la grammatica appena presentata è ambigua, e quindi adotteremo le stesse convenzioni del calcolo proposizionale riassunte nell'Osservazione 9.2.1. Queste convenzioni sono sufficienti perché le nuove produzioni di questa logica (quelle che nella Definizione 9.5.2 sono sottolineate) non introducano ulteriori ambiguità:

⁹Come spiegato dall'Esempio 8.3.12, i termini definiti attraverso questa grammatica corrispondono agli \mathcal{F} -termini introdotti nella Definizione 7.4.3.

- Le formule quantificate sono sempre obbligatoriamente racchiuse tra parentesi,¹⁰ quindi anche se sono annidate è sempre chiaro dove finisce il campo di azione di ogni quantificatore (Definizione 9.7.1).
- La categoria sintattica di una stringa della forma $p(t_1, \dots, t_n)$, che potrebbe essere sia $\langle \text{Atom} \rangle$ (per la produzione $\langle \text{Atom} \rangle \rightsquigarrow \langle Pide_n \rangle(\langle \text{Term} \rangle, \dots, \langle \text{Term} \rangle)$) che $\langle \text{Term} \rangle$ (per la produzione $\langle \text{Term} \rangle \rightsquigarrow \langle Fide_n \rangle(\langle \text{Term} \rangle, \dots, \langle \text{Term} \rangle)$), è determinata dal simbolo p : sarà una formula atomica se $p \in \mathcal{P}_n$, e un termine se $p \in \mathcal{F}_n$.

In realtà, nell'ultimo caso anche non sapendo se p è un simbolo di predicato o di funzione, il contesto in cui $p(t_1, \dots, t_n)$ compare è sempre sufficiente a disambiguare, perché determina se la stringa deve essere una formula o un termine. Per esempio, se in una formula compare “ $42 + p(x, y)$ ” inferiamo che p è un simbolo di funzione perché $p(x, y)$ è argomento del simbolo di funzione $+$, e quindi deve essere un termine; invece se compare in $Q \Rightarrow p(x, y)$ inferiamo che p è un simbolo di predicato perché $p(x, y)$ deve essere una formula.

Vediamo alcuni esempi di alfabeti del primo ordine.

Esempio 9.5.4 (alcuni alfabeti del primo ordine).

1. L'alfabeto $\mathcal{A}_{\mathcal{T}} = (\mathcal{C}_{\mathcal{T}}, \mathcal{F}_{\mathcal{T}}, \mathcal{P}_{\mathcal{T}}, \mathcal{V}_{\mathcal{T}})$ (dove \mathcal{T} sta per toy, perché lo useremo per esempi giocattolo), è costituito da:

- i simboli di costante $\mathcal{C}_{\mathcal{T}} = \{k\}$;
- i simboli di funzione $\mathcal{F}_{\mathcal{T}} = \{\mathcal{F}_{\mathcal{T},n}\}_{n \in \mathbb{N}^+}$, con $\mathcal{F}_{\mathcal{T},1} = \{f\}$ e $\mathcal{F}_{\mathcal{T},i} = \emptyset$ per tutti gli $i \in \mathbb{N}^+ \setminus \{1\}$;
- i simboli di predicato $\mathcal{P}_{\mathcal{T}} = \{\mathcal{P}_{\mathcal{T},n}\}_{n \in \mathbb{N}}$, con $\mathcal{P}_{\mathcal{T},1} = \{A\}$, $\mathcal{P}_{\mathcal{T},2} = \{B\}$ e $\mathcal{P}_{\mathcal{T},i} = \emptyset$ per tutti gli $i \in \mathbb{N} \setminus \{1, 2\}$;
- le variabili $\mathcal{V}_{\mathcal{T}} = \{x, y, \dots\}$.

Attenzione: Per alleggerire la notazione, nel seguito adottiamo la convenzione di considerare \mathcal{F} e \mathcal{P} come insiemi invece che come famiglie di insiemi, indicando per ogni simbolo l'arietà nel seguente modo: se g ha arietà n , allora lo scriviamo $g(\underbrace{_, \dots, _}_n)$. Per esempio, usando questa convenzione i simboli di funzione e di predicato dell'alfabeto $\mathcal{A}_{\mathcal{T}}$ appena visto possono essere definiti in modo equivalente ma più semplice come:

- $\mathcal{F}_{\mathcal{T}} = \{f(_)\}$
- $\mathcal{P}_{\mathcal{T}} = \{A(_), B(_, _)\}$.

2. L'ALFABETO DEI NATURALI $\mathcal{A}_{\mathbb{N}} = (\mathcal{C}_{\mathbb{N}}, \mathcal{F}_{\mathbb{N}}, \mathcal{P}_{\mathbb{N}}, \mathcal{V}_{\mathbb{N}})$ è costituito dai seguenti insiemi (i puntini sospensivi ci permetteranno eventualmente di aggiungere altri simboli nel seguito):

- i simboli di costante $\mathcal{C}_{\mathbb{N}} = \mathbb{N}$;
- i simboli di funzione $\mathcal{F}_{\mathbb{N}} = \{\text{succ}(_), _, + _, _, \times _, _, / _, _, \% _, \dots\}$;
- i simboli di predicato $\mathcal{P}_{\mathbb{N}} = \{_, \leq _, _, \geq _, _, < _, _, > _, _, = _, \text{primo}(_), \text{pari}(_), \dots\}$;
- le variabili $\mathcal{V}_{\mathbb{N}} = \{x, y, \dots, n, m, \dots\}$.

Si noti che abbiamo adottato la convenzione di scrivere $_ op _$ invece di $op(_, _)$ per gli operatori binari per i quali normalmente si adotta la notazione infissa.

3. L'ALFABETO DELLE PERSONE $\mathcal{A}_{\mathcal{P}} = (\mathcal{C}_{\mathcal{P}}, \mathcal{F}_{\mathcal{P}}, \mathcal{P}_{\mathcal{P}}, \mathcal{V}_{\mathcal{P}})$ è costituito dai seguenti insiemi:

- i simboli di costante $\mathcal{C}_{\mathcal{P}} = \{\text{Aldo}, \text{Bruna}, \text{io}, \text{Joe Biden}, \dots\}$;
- i simboli di funzione $\mathcal{F}_{\mathcal{P}} = \{\text{padre}(_), \text{nonnoMaterno}(_), \dots\}$;
- i simboli di predicato $\mathcal{P}_{\mathcal{P}} = \{\text{padre}(_, _), \text{nonno}(_, _), \text{figlio}(_, _), \text{fratelli}(_, _), \dots\}$;

¹⁰Questa è una convenzione adottata in queste dispense per semplificare la presentazione: nella notazione matematica usuale le parentesi non sono necessarie.

- le variabili $\mathcal{V}_P = \{x, y, \dots, p, q, \dots\}$.

4 L'ALFABETO DEGLI INSIEMI $\mathcal{A}_S = (\mathcal{C}_S, \mathcal{F}_S, \mathcal{P}_S, \mathcal{V}_S)$ è costituito dai seguenti insiemi:

- i simboli di costante $\mathcal{C}_S = \mathbb{N} = \{\emptyset, \mathbb{N}, \mathbb{Z}, \mathbb{N}^+, \dots, 0, 1, 2, \dots\}$;
- i simboli di funzione $\mathcal{F}_S = \{_, \cup, _, \cap, _, \backslash, _, \neg, \dots\}$;
- i simboli di predicato $\mathcal{P}_S = \{\text{isSet}(_), \text{vuoto}(_), _, \in, _, \subset, _, \subseteq, _, _, \supseteq, _, _, =, _, \dots\}$;
- le variabili $\mathcal{V}_S = \{x, y, \dots, X, Y, \dots\}$: useremo le variabili maiuscole per gli insiemi, le minuscole per gli elementi.

Si noti che per gli alfabeti dei naturali, delle persone e degli insiemi abbiamo introdotto dei simboli che, avendo nomi che suggeriscono un preciso significato, ci renderanno le formule più facili da leggere. Tuttavia da un punto di vista formale è importante ricordare che tali simboli non hanno automaticamente un significato: questo gli viene attribuito successivamente con un'interpretazione (Definizione 9.7.3). Per esempio, del simbolo di funzione $_ + _$ possiamo dire che è binario e infisso, ma solo quando ne forniremo l'interpretazione potremo dire che esso rappresenta l'addizione tra naturali, come atteso, e non un'altra operazione.

Esempio 9.5.5 (sintassi di termini e formule). *Si consideri l'alfabeto dei naturali $\mathcal{A}_{\mathbb{N}}$. Le seguenti stringhe sono sintatticamente corrette, ovvero possono essere generate dalla grammatica della Definizione 9.5.2:*

- 0, 5, 21934, x, n : sono simboli di costante o variabili, quindi sono termini, cioè elementi di $\langle\langle \text{Term} \rangle\rangle$;
- $3 + 5, \text{succ}(x + 3), (3 + 5) \times 0$: sono simboli di funzione applicati a un numero di termini pari alla propria arietà, quindi sono anche essi termini in $\langle\langle \text{Term} \rangle\rangle$;
- $x \leq 6, \text{pari}(42), \text{primo}(3 \times (5 + x)), 4 + x = 4 - y$: sono simboli di predicati applicati a un numero di termini pari alla propria arietà, quindi sono formule atomiche, elementi di $\langle\langle \text{Atom} \rangle\rangle$;
- $(x \leq 6 \wedge \text{pari}(x)), (\text{primo}(x) \Rightarrow x > 0), (\forall x . x \geq 0), (x > 0 \Rightarrow (\exists y . y < x))$: sono connettivi logici oppure quantificatori applicati a formule, quindi sono formule, elementi di $\langle\langle \text{Pred} \rangle\rangle$.

Invece le seguenti espressioni non sono sintatticamente corrette, quindi non hanno alcun significato:

- $12+$: “+” è un simbolo di funzione binario, non può essere applicato a un solo argomento;
- $(x \leq y) \leq z$: il secondo simbolo di predicato “ \leq ” è applicato a una formula atomica e a un termine invece che a due termini; questa espressione, sintatticamente non corretta, viene spesso usata come abbreviazione della formula $(x \leq y) \wedge (y \leq z)$;
- $(\forall x . x + y)$: l'espressione quantificata non è una formula ma un termine.

9.6 Formalizzazione di frasi

Con la logica dei predicati siamo in grado di formalizzare molte frasi in maniera più espressiva di quanto sia possibile con il calcolo proposizionale. Vediamo alcuni esempi, basati sugli alfabeti degli interi e delle persone. In questi esempi ci basiamo su una comprensione intuitiva del significato dei quantificatori presentati alla fine della Sezione 9.5, perché presenteremo la loro semantica formale solo nella prossima sezione. Comunque assumiamo che il lettore conosca bene i connettivi logici.

1. “Esiste un numero che è sia pari che primo”

Ovviamente rappresentiamo “Esiste” con una quantificazione esistenziale, e sfruttiamo i simboli di predicato $\text{primo}(_)$ e $\text{pari}(_)$. La formula risultante è

$$(\exists x . (\text{pari}(x) \wedge \text{primo}(x)))$$

2. “*Tutti gli uomini sono mortali*”

Estendiamo l’alfabeto \mathcal{A}_P con il simbolo di predicato $mortale(_)$ e il significato intuitivo che $mortale(p)$ è vero se e solo se p è mortale. Inoltre rappresentiamo “*Tutti*” con una quantificazione universale. La formula risultante è:

$$(\forall x . mortale(x))$$

 3. “*Tutti i numeri naturali dispari sono maggiori di zero*”

In questo caso la quantificazione universale non è su tutti i naturali, ma è ristretta a un sottoinsieme: i naturali dispari. Un modo naturale di formalizzarla è di riconoscere che c’è un’implicazione隐含的: “*Per ogni naturale vale che se è dispari allora è maggiore di zero*”. La formula risultante è

$$(\forall x . (dispari(x) \Rightarrow (x > 0)))$$

 4. “*Esiste un numero naturale dispari che è minore o uguale a zero*”

Anche qui la quantificazione è ristretta ai numeri dispari, ma è una quantificazione esistenziale e possiamo leggerla come “*Esiste un numero che è dispari ed è minore o uguale a zero*”. Quindi può essere formalizzata con

$$(\exists x . (dispari(x) \wedge (x \leq 0))) \quad (9.10)$$

Si noti che l’uso di un’implicazione (come nel caso precedente) sarebbe errata: la formula

$$(\exists x . (dispari(x) \Rightarrow (x \leq 0))) \quad (9.11)$$

ha un significato diverso. Infatti la formula (9.10) è falsa nell’interpretazione standard sui naturali (così come è falso l’enunciato originale), mentre la formula (9.11) è vera, poiché il numero 42 (e qualunque altro numero pari) soddisfa $(dispari(42) \Rightarrow (42 \leq 0))$.

 5. “*Ogni persona in questa classe ha un amico che parla inglese o francese*”

Estendiamo l’alfabeto delle persone con i simboli di predicato unari *questaClasse*, *parlaENG* e *parlaFR* e con il simbolo di predicato binario *amico*, con il seguente significato intuitivo:

- $questaClasse(p) = \texttt{t}$ se e solo se la persona p appartiene a questa classe
- $parlaENG(p) = \texttt{t}$ se e solo se la persona p parla inglese
- $parlaFR(p) = \texttt{t}$ se e solo se la persona p parla francese
- $amico(p, q) = \texttt{t}$ se e solo se le persone p e q sono amiche.

Quella che segue è una possibile formalizzazione della frase:

$$(\forall x . (questaClasse(x) \Rightarrow (\exists y . (amico(x, y) \wedge (parlaENG(y) \vee parlaFR(y))))))$$

Abbiamo usato di nuovo un’implicazione per restringere la quantificazione universale a un sottoinsieme delle persone. Che cosa succede se spostiamo la quantificazione esistenziale fuori dall’implicazione?

$$(\forall x . (\exists y . (questaClasse(x) \Rightarrow (amico(x, y) \wedge (parlaENG(y) \vee parlaFR(y))))))$$

Rileggendo la formula con attenzione si vede che il significato è rimasto invariato. Il prossimo esempio mostra che questo non è vero se si scambiano due quantificatori diversi.

 6. “*Ogni persona ha un padre*”

Usando il simbolo di predicato binario *padre*, formalizziamo questa semplice frase nel modo seguente:

$$(\forall x . (\exists y . padre(y, x)))$$

Se rileggiamo la formula abbiamo “*Per ogni persona ne esiste un’altra che è suo padre*”. Cosa succede se scambiamo i due quantificatori? La formula risultante è:

$$(\exists y . (\forall x . padre(y, x)))$$

che, riletta in italiano, dice “*Esiste una persona che è il padre di ogni altra persona*”. È evidente che il significato delle due formule è completamente diverso, e in particolare la prima è vera nell’interpretazione standard delle persone mentre la seconda è falsa.

7. “Ogni persona felice ha un solo amico del cuore”

Estendiamo l’alfabeto delle persone con i simboli di predicato $felice(_)$ e $amiciDelCuore(_, _)$ con l’ovvia interpretazione. Come primo tentativo, proviamo a formalizzare la frase così:

$$(\forall x . (\exists y . (felice(x) \Rightarrow amiciDelCuore(x, y))))$$

Tuttavia questa formula non coglie il concetto di *unicità*! La seguente formula invece lo cattura in modo corretto:

$$(\forall x . (\exists y . (felice(x) \Rightarrow (amiciDelCuore(x, y) \wedge (\forall z . (amiciDelCuore(x, z) \Rightarrow y = z))))))$$

9.7 Interpretazioni e semantica delle formule predicative

Vediamo ora come determinare la semantica, cioè il valore di verità, di formule predicative su un certo alfabeto del primo ordine. La prima osservazione è che non tutte tali formule possono avere una semantica ben definita. Per esempio, interpretando in modo ovvio il simbolo di predicato \leq come la relazione “minore o uguale” sui naturali, la formula $x \leq 0$ non ha un valore di verità ben determinato: essa vale **t** se sostituiamo x con 0, e **f** se sostituiamo x con un qualunque altro naturale. Invece la semantica delle formule $(\exists x . x \leq 0)$ e $(\forall x . x \leq 0)$ è ben definita, perché le quantificazioni $\forall x$ e $\exists x$ indicano come deve essere sostituita x per determinare il valore di verità: $(\exists x . x \leq 0)$ vale **t** perché c’è almeno un naturale minore o uguale a 0, mentre $(\forall x . x \leq 0)$ vale **f** perché non tutti i naturali sono minori o uguali a 0.

Definizione 9.7.1 (campo di azione, variabili legate e libere, formule aperte e chiuse). *Il CAMPO D’AZIONE (o PORTATA o SCOPE) della quantificazione $\forall x$ nella formula $(\forall x . P)$ è l’intera formula P , e analogamente per la quantificazione esistenziale. Una specifica occorrenza di una variabile x in una formula P è detta LEGATA se compare nella portata di una quantificazione $\forall x$ o $\exists x$, altrimenti è detta LIBERA. Una formula P è CHIUSA se ogni occorrenza di variabile in essa è legata, altrimenti è aperta. Invece una formula come $x \leq 0$ viene detta APERTA perché contiene una variabile, la x , che compare nella formula senza essere legata a un corrispondente quantificatore. Una variabile che non è legata a un quantificatore viene detta LIBERA.*

Esempio 9.7.2 (campo d’azione, formule aperte e chiuse). *Nella formula seguente le parentesi graffe mostrano il campo d’azione dei due quantificatori. La formula è chiusa perché ogni occorrenza di variabile (le due x e la y) compare nel campo d’azione di un corrispondente quantificatore.*

$$(\forall x . \overbrace{x \leq 0 \Rightarrow (\forall y . \underbrace{y \geq x})}^{\forall x})$$

Invece la formula che segue è aperta, perché la seconda x non è nel campo di azione di un $\forall x$ o $\exists x$.

$$(\forall x . \underbrace{x > 0}_{\forall x}) \Rightarrow (\exists y . \underbrace{y < \overbrace{x}^{\text{libera}}}_{\exists y})$$

Data una formula su un certo alfabeto \mathcal{A} , per associarle un valore di verità dobbiamo fissare il significato dei simboli di costante, di funzione, di predicato e delle variabili che compaiono in essa, esattamente come per una formula proposizionale dovevamo fissare un valore di verità per ogni simbolo proposizionale. Per far questo fisseremo un *dominio di interpretazione*, cioè un insieme, e un’interpretazione dei simboli di costante come elementi del dominio, e dei simboli di funzione e di predicato come opportune funzioni o relazioni sul dominio. Le variabili verranno trattate diversamente, perché il loro assegnamento a elementi del dominio può cambiare dinamicamente durante la valutazione della semantica di una formula, a differenza dell’interpretazione degli altri simboli che rimana fissa.

Quindi il concetto di interpretazione, già visto nel caso più semplice del calcolo proposizionale, viene arricchito nel modo seguente.

Definizione 9.7.3 (interpretazione per alfabeto del primo ordine). *Un'INTERPRETAZIONE \mathcal{I} per un alfabeto del primo ordine $\mathcal{A} = (\mathcal{C}, \mathcal{F}, \mathcal{P}, \mathcal{V})$ è una coppia $\mathcal{I} = (\mathcal{D}, \alpha)$ dove \mathcal{D} è il DOMINIO (DI INTERPRETAZIONE) (un insieme di valori), mentre α è un'ASSOCIAZIONE costituita da tre componenti $\alpha = \langle \alpha_{\mathcal{C}}, \alpha_{\mathcal{F}}, \alpha_{\mathcal{P}} \rangle$, dove:*

- $\alpha_{\mathcal{C}} : \mathcal{C} \rightarrow \mathcal{D}$ associa a ogni simbolo di costante in \mathcal{C} un elemento del dominio;
- $\alpha_{\mathcal{F}} = \{\alpha_{\mathcal{F}_n}\}_{n \in \mathbb{N}^+}$ è una famiglia di funzioni, in cui $\alpha_{\mathcal{F}_n}$ associa a ogni simbolo di funzione $f \in \mathcal{F}_n$ una funzione $\alpha_{\mathcal{F}_n}(f) : \mathcal{D}^n \rightarrow \mathcal{D}$, cioè una funzione n -aria su \mathcal{D} .
- $\alpha_{\mathcal{P}} = \{\alpha_{\mathcal{P}_n}\}_{n \in \mathbb{N}}$ è una famiglia di funzioni, dove
 - $\alpha_{\mathcal{P}_0}$ associa a ogni simbolo di predicato di arietà zero un valore booleano, cioè $\alpha_{\mathcal{P}_0} : \mathcal{P}_0 \rightarrow \text{Bool}$.
 - per ogni $n \in \mathbb{N}^+$, $\alpha_{\mathcal{P}_n}$ associa a ogni simbolo di predicato $A \in \mathcal{P}_n$ un sottoinsieme $\alpha_{\mathcal{P}_n}(A) \subseteq \mathcal{D}^n$.

Una volta fissata un'interpretazione per un alfabeto \mathcal{A} possiamo utilizzarla per associare un valore di verità (la semantica) a ogni formula chiusa scritta con simboli in \mathcal{A} .

Ma prima di vedere come si fa, introduciamo le interpretazioni per gli alfabeti dell'Esempio 9.5.4, in modo da poter poi presentare degli esempi. Nel caso degli alfabeti dei naturali, delle persone e degli insiemi queste interpretazioni associano ai vari simboli il significato standard che ci aspettiamo, ma sono comunque necessarie per collegare le entità sintattiche (i simboli degli alfabeti) con la corrispondente semantica definita sul dominio di interpretazione.

Definizione 9.7.4 (esempi di interpretazioni).

1. Per l'alfabeto $\mathcal{A}_{\mathcal{T}} = (\mathcal{C}_{\mathcal{T}}, \mathcal{F}_{\mathcal{T}}, \mathcal{P}_{\mathcal{T}}, \mathcal{V}_{\mathcal{T}})$ dell'Esempio 9.5.4 introduciamo l'interpretazione $\mathcal{I}_{\mathcal{T}} = (\mathcal{D}_{\mathcal{T}}, \alpha^{\mathcal{T}})$ dove

- $\mathcal{D}_{\mathcal{T}} = \{a, b, c\}$, quindi il dominio è finito;
- $\alpha^{\mathcal{T}} = \langle \alpha_{\mathcal{C}}^{\mathcal{T}}, \alpha_{\mathcal{F}}^{\mathcal{T}}, \alpha_{\mathcal{P}}^{\mathcal{T}} \rangle$;
- $\alpha_{\mathcal{C}}^{\mathcal{T}}(k) = a$;
- $\alpha_{\mathcal{F}}^{\mathcal{T}}(f) = \{a \mapsto b, b \mapsto c, c \mapsto a\}$;
- $\alpha_{\mathcal{P}}^{\mathcal{T}}(A) = \{a, b\}$;
- $\alpha_{\mathcal{P}}^{\mathcal{T}}(B) = \{(a, a), (c, a), (c, b), (b, c)\}$.

Si noti che coerentemente con la convenzione adottata nell'Esempio 9.5.4, consideriamo $\alpha_{\mathcal{F}}^{\mathcal{T}}$ e $\alpha_{\mathcal{P}}^{\mathcal{T}}$ come delle funzioni e non come famiglie di funzioni. La loro definizione per ogni simbolo è consistente con la corrispondente arietà, che non è scritta esplicitamente ma si può ricavare dalla definizione dell'alfabeto. Infatti $\alpha_{\mathcal{F}}^{\mathcal{T}}(f) : \mathcal{D}_{\mathcal{T}} \rightarrow \mathcal{D}_{\mathcal{T}}$ è una funzione unaria, $\alpha_{\mathcal{P}}^{\mathcal{T}}(A)$ è un sottoinsieme di $\mathcal{D}_{\mathcal{T}}$ e $\alpha_{\mathcal{P}}^{\mathcal{T}}(B)$ è un sottoinsieme di $\mathcal{D}_{\mathcal{T}}^2$.

2. L'interpretazione (standard) per l'alfabeto dei naturali $\mathcal{A}_{\mathbb{N}} = (\mathcal{C}_{\mathbb{N}}, \mathcal{F}_{\mathbb{N}}, \mathcal{P}_{\mathbb{N}}, \mathcal{V}_{\mathbb{N}})$ è la coppia $\mathcal{I}_{\mathbb{N}} = (\mathcal{D}_{\mathbb{N}}, \alpha^{\mathbb{N}})$ dove il dominio $\mathcal{D}_{\mathbb{N}}$ è costituito dall'insieme dei naturali \mathbb{N} , mentre $\alpha^{\mathbb{N}} = \langle \alpha_{\mathcal{C}}^{\mathbb{N}}, \alpha_{\mathcal{F}}^{\mathbb{N}}, \alpha_{\mathcal{P}}^{\mathbb{N}} \rangle$ dove

- $\alpha_{\mathcal{C}}^{\mathbb{N}} : \mathcal{C}_{\mathbb{N}} \rightarrow \mathbb{N}$ associa a ogni simbolo di costante $c \in \mathcal{C}^{\mathbb{N}} (= \mathbb{N})$ se stessa;
- $\alpha_{\mathcal{F}}^{\mathbb{N}}$ associa a ogni simbolo di funzione in $\mathcal{F}_{\mathbb{N}}$ la corrispondente funzione sui naturali. Per esempio, $\alpha_{\mathcal{F}}^{\mathbb{N}}(\text{succ})$ è la funzione successore che mappa ogni numero x su $x + 1$, quindi per esempio $\alpha_{\mathcal{F}}^{\mathbb{N}}(\text{succ})(5) = 6$; invece $\alpha_{\mathcal{F}}^{\mathbb{N}}(+)$ è l'addizione su naturali, quindi per esempio $\alpha_{\mathcal{F}}^{\mathbb{N}}(+)(3, 5) = 8$, e così via.
- $\alpha_{\mathcal{P}}^{\mathbb{N}}$ associa a ogni simbolo di predicato in $\mathcal{P}_{\mathbb{N}}$ di arietà n la corrispondente proprietà sui naturali, cioè l'insieme di n -uple di naturali che la soddisfano. Per esempio, per tutti i naturali $n, m \in \mathbb{N}$, $(n, m) \in \alpha_{\mathcal{P}}^{\mathbb{N}}(\leq)$ se e solo se n è minore o uguale a m ; mentre $n \in \alpha_{\mathcal{P}}^{\mathbb{N}}(\text{primo})$ se e solo se n è un numero primo.

3. L'interpretazione (standard) $\mathcal{I}_{\mathcal{P}}$ dell'alfabeto delle persone $\mathcal{A}_{\mathcal{P}} = (\mathcal{C}_{\mathcal{P}}, \mathcal{F}_{\mathcal{P}}, \mathcal{P}_{\mathcal{P}}, \mathcal{V}_{\mathcal{P}})$ ha come dominio appunto l'insieme di tutte le persone. I simboli di costante, di funzione e di predicato sono associati nel modo ovvio a persone, a funzioni o a relazioni definite su persone. Per esempio, $\alpha_{\mathcal{C}}^{\mathcal{P}}(\text{Aldo})$ è una determinata persona di nome Aldo; $\alpha_{\mathcal{C}}^{\mathcal{P}}(\text{Joe Biden})$ è un politico americano, attualmente Presidente Eletto degli USA; $\alpha_{\mathcal{F}}^{\mathcal{P}}(\text{padre})$ è la funzione che associa ogni persona al suo padre naturale; e $(p, q) \in \alpha_{\mathcal{P}}^{\mathcal{P}}(\text{fratelli})$ se e solo se le persone p e q sono fratelli.

4. Infine l'interpretazione (standard) dell'alfabeto degli insiemi ha come dominio un insieme che ha come elementi sia gli insiemi di nostro interesse (come \mathbb{N}, \mathbb{Z} , ecc.) che i loro elementi. I simboli di costante, di funzione e di predicato elencati nell'Esempio 9.5.4 sono associati in modo ovvio al corrispondente significato rispettando la usuale notazione matematica.

Semantica di termini e di formule predicative

Come nel caso del calcolo proposizionale, la semantica di una formula predicativa per una data interpretazione può essere definita in modo induttivo. Dobbiamo però considerare separatamente la categoria sintattica dei termini, visto che la loro semantica non è un booleano ma un elemento del dominio. Inoltre sia formule che termini possono in generale contenere variabili libere, per cui la semantica è parametrica rispetto a un *assegnamento* che associa a ogni variabile un elemento del dominio.

Definizione 9.7.5 (assegnamento). *Dato un alfabeto \mathcal{A} con insieme di variabili \mathcal{V} e un'interpretazione $\mathcal{I} = \langle \mathcal{D}, \alpha \rangle$, un ASSEGNAIMENTO è una funzione parziale $r : \mathcal{V} \rightarrow \mathcal{D}$.*¹¹

Dati un assegnamento $r : \mathcal{V} \rightarrow \mathcal{D}$, una variabile $x \in \mathcal{V}$ e un elemento $d \in \mathcal{D}$, con $r[x \mapsto d]$ denotiamo l'assegnamento così definito:

$$r[x \mapsto d](y) = \begin{cases} d & \text{se } y = x \\ r(y) & \text{altrimenti.} \end{cases}$$

Se $\emptyset : \mathcal{V} \rightarrow \mathcal{D}$ è la relazione vuota, scriveremo semplicemente $[x \mapsto d]$ per $\emptyset[x \mapsto d]$.

Definizione 9.7.6 (semantica dei termini). *Data una interpretazione $\mathcal{I} = \langle \mathcal{D}, \alpha \rangle$ e un assegnamento $r : \mathcal{V} \rightarrow \mathcal{D}$, il VALORE RISPETTO AD \mathcal{I} ED r di un termine è dato dalla funzione parziale*¹²

$$\llbracket _ \rrbracket_{\mathcal{I}}^r : \mathbf{Term} \rightarrow \mathcal{D}$$

definita per induzione strutturale come segue:

1. $\llbracket x \rrbracket_{\mathcal{I}}^r = r(x)$ per ogni $x \in \mathcal{V}$ (il valore di una variabile è determinato dall'assegnamento);
2. $\llbracket c \rrbracket_{\mathcal{I}}^r = \alpha_{\mathcal{C}}(c)$ per ogni $c \in \mathcal{C}$ (il valore di un simbolo di costante è determinato da $\alpha_{\mathcal{C}}$);
3. Se f è un simbolo di funzione di arietà n e $\llbracket t_1 \rrbracket_{\mathcal{I}}^r = d_1, \dots, \llbracket t_n \rrbracket_{\mathcal{I}}^r = d_n$, allora

$$\llbracket f(t_1, \dots, t_n) \rrbracket_{\mathcal{I}}^r = \alpha_{\mathcal{F}}(f)(d_1, \dots, d_n)$$

(si valutano ricorsivamente i sottotermini e si applica la funzione $\alpha_{\mathcal{F}}(f)$).

Esempio 9.7.7 (da termini a elementi del dominio).

1. Consideriamo l'alfabeto $\mathcal{A}_{\mathcal{T}}$ dell'Esempio 9.5.4 e l'interpretazione $\mathcal{I}_{\mathcal{T}} = (\mathcal{D}_{\mathcal{T}}, \alpha^{\mathcal{T}})$ della Definizione 9.7.4. Esempi di termini sono la variabile x , la costante k , $f(k)$, e $f(f(x))$. Dato una assegnamento $r : \mathcal{V}_{\mathcal{T}} \rightarrow \mathcal{D}_{\mathcal{T}} = \{a, b, c\}$ tale che $r(x) = b$, i corrispondenti valori sono:

- $\llbracket x \rrbracket_{\mathcal{I}_{\mathcal{T}}}^r = r(x) = b$ (Definizione 9.7.6, clausola 1);

¹¹Ricordiamo che diversamente da una funzione, una *funzione parziale* è una relazione univalema ma non necessariamente totale (Definizione 2.5.25).

¹²Poiché r è una funzione parziale, $\llbracket t \rrbracket_{\mathcal{I}}^r$ potrebbe non essere definito per un termine t . Per come useremo gli assegnamenti questo non può mai succedere, quindi per semplicità ignoriamo questa possibilità e consideriamo la funzione totale.

- $\llbracket k \rrbracket_{\mathcal{I}_T}^r = \alpha_{\mathcal{C}}^T(k) = a \quad (\text{Definizione 9.7.6, clausola 2});$
- $$\begin{aligned} \llbracket f(k) \rrbracket_{\mathcal{I}_T}^r &= \alpha_{\mathcal{F}}^T(f)(\llbracket k \rrbracket_{\mathcal{I}_T}^r) \quad (\text{Def. 9.7.6, cl. 3}) \\ &= \alpha_{\mathcal{F}}^T(f)(a) \quad (\text{Def. 9.7.6, cl. 2}) \\ &= b \quad (\text{Def. 9.7.4, punto 1}) \end{aligned}$$
- $$\begin{aligned} \llbracket f(f(x)) \rrbracket_{\mathcal{I}_T}^r &= \alpha_{\mathcal{F}}^T(f)(\llbracket f(x) \rrbracket_{\mathcal{I}_T}^r) \quad (\text{Def. 9.7.6, cl. 3}) \\ &= \alpha_{\mathcal{F}}^T(f)(\alpha_{\mathcal{F}}^T(f)(\llbracket x \rrbracket_{\mathcal{I}_T}^r)) \quad (\text{Def. 9.7.6, cl. 3}) \\ &= \alpha_{\mathcal{F}}^T(f)(\alpha_{\mathcal{F}}^T(f)(b)) \quad (\text{Def. 9.7.6, cl. 2}) \\ &= \alpha_{\mathcal{F}}^T(f)(c) \quad (\text{Def. 9.7.4, punto 1}) \\ &= a \quad (\text{Def. 9.7.4, punto 1}) \end{aligned}$$

2. Consideriamo ora l'alfabeto dei naturali $\mathcal{A}_{\mathbb{N}}$. Esempi di termini sono 5 , $7+x$, e $\text{succ}((y \times 4)/5)$. Dato un assegnamento $r : \mathcal{V}_{\mathbb{N}} \rightarrow \mathbb{N}$ tale che $r(x) = 8$ e $r(y) = 3$, i corrispondenti valori del dominio rispetto all'interpretazione standard $\mathcal{I}_{\mathbb{N}}$ e a r sono:

- $\llbracket 5 \rrbracket_{\mathcal{I}_{\mathbb{N}}}^r = \alpha_{\mathcal{C}}^{\mathbb{N}}(5) = 5;$
- $\llbracket 7 + x \rrbracket_{\mathcal{I}_{\mathbb{N}}}^r = \alpha_{\mathcal{F}}^{\mathbb{N}}(+)(\llbracket 7 \rrbracket_{\mathcal{I}_{\mathbb{N}}}^r, \llbracket x \rrbracket_{\mathcal{I}_{\mathbb{N}}}^r) = \alpha_{\mathcal{F}}^{\mathbb{N}}(+)(7, r(x)) = 7 + 8 = 15;$
- $$\begin{aligned} \llbracket \text{succ}((y \times 4)/5) \rrbracket_{\mathcal{I}_{\mathbb{N}}}^r &= \alpha_{\mathcal{F}}^{\mathbb{N}}(\text{succ})(\llbracket (y \times 4)/5 \rrbracket_{\mathcal{I}_{\mathbb{N}}}^r) \quad (\text{Def. 9.7.6, clausola 3}) \\ &= \alpha_{\mathcal{F}}^{\mathbb{N}}(\text{succ})(\llbracket y \rrbracket_{\mathcal{I}_{\mathbb{N}}}^r \times \llbracket 4 \rrbracket_{\mathcal{I}_{\mathbb{N}}}^r / \llbracket 5 \rrbracket_{\mathcal{I}_{\mathbb{N}}}^r) \quad (\text{Def. 9.7.6, cl. 3 e Def. 9.7.4, cl. 1}) \\ &= \alpha_{\mathcal{F}}^{\mathbb{N}}(\text{succ})(\llbracket r(y) \times 4 \rrbracket_{\mathcal{I}_{\mathbb{N}}}^r / 5) \quad (\text{Def. 9.7.6, clausole 1 e 2}) \\ &= \alpha_{\mathcal{F}}^{\mathbb{N}}(\text{succ})(\llbracket 3 \times 4 \rrbracket_{\mathcal{I}_{\mathbb{N}}}^r / 5) \quad (r(y) = 3 \text{ per ipotesi}) \\ &= \alpha_{\mathcal{F}}^{\mathbb{N}}(\text{succ})(2) \quad (\text{Calcolo } (/ \text{ è divisione intera})) \\ &= 2 + 1 = 3. \quad (\text{Def. 9.7.4, punto 2}) \end{aligned}$$

La semantica delle formule è definita anche essa per induzione strutturale in modo parametrico rispetto a una interpretazione e a un assegnamento. Se una formula è aperta, essa può assumere valori di verità diversi nella stessa interpretazione a seconda del valore che l'assegnamento associa alle variabili libere (si pensi semplicemente a $\text{primo}(x)$ valutato nell'interpretazione $\mathcal{I}_{\mathbb{N}}$ con $r(x) = 3$ o $r(x) = 6$). Invece se una formula è chiusa, il suo valore in una interpretazione è univocamente determinato. Presentiamo prima la semantica delle formule generali e poi quella delle formule chiuse.

Definizione 9.7.8 (Semantica della logica dei predicati (anche formule aperte)). *Data una interpretazione $\mathcal{I} = (\mathcal{D}, \alpha)$ per un alfabeto \mathcal{A} e un assegnamento $r : \mathcal{V} \rightarrow \mathcal{D}$, il VALORE RISPETTO AD \mathcal{I} ED r delle formule predicative è dato dalla funzione*

$$\llbracket _ \rrbracket_{\mathcal{I}}^r : \mathbf{Pred} \rightarrow \mathbf{Bool}$$

definita sulle formule per induzione strutturale come segue:

1. $\llbracket T \rrbracket_{\mathcal{I}}^r = t$ e $\llbracket F \rrbracket_{\mathcal{I}}^r = f$;
2. Per ogni $A \in \mathcal{P}_0$, $\llbracket A \rrbracket_{\mathcal{I}}^r = \alpha_{\mathcal{P}_0}(A)$.
3. Per ogni $A \in \mathcal{P}_n$ di arietà $n > 0$,

$$\llbracket A(t_1, \dots, t_n) \rrbracket_{\mathcal{I}}^r = \begin{cases} t & \text{se } (\llbracket t_1 \rrbracket_{\mathcal{I}}^r, \dots, \llbracket t_n \rrbracket_{\mathcal{I}}^r) \in \alpha_{\mathcal{P}_n}(A) \\ f & \text{altrimenti.} \end{cases}$$

4. $\llbracket (P) \rrbracket_{\mathcal{I}}^r = (\llbracket P \rrbracket_{\mathcal{I}}^r)$ per ogni $P \in \mathbf{Pred}$;
5. $\llbracket \neg P \rrbracket_{\mathcal{I}}^r = \neg \llbracket P \rrbracket_{\mathcal{I}}^r$ per ogni formula atomica P ;

6. $\llbracket P \text{ op } Q \rrbracket_{\mathcal{I}}^r = \llbracket P \rrbracket_{\mathcal{I}}^r \text{ op } \llbracket Q \rrbracket_{\mathcal{I}}^r$ per ogni connettivo $\text{op} \in \{\wedge, \vee, \Rightarrow, \Leftarrow, \Leftrightarrow\}$ e per ogni $P, Q \in \mathbf{Pred}$.
7. $\llbracket (\forall x . P) \rrbracket_{\mathcal{I}}^r = t$ se e solo se $\llbracket P \rrbracket_{\mathcal{I}}^{r[x \mapsto d]} = t$ per ogni $d \in \mathcal{D}$.
8. $\llbracket (\exists x . P) \rrbracket_{\mathcal{I}}^r = t$ se e solo se $\llbracket P \rrbracket_{\mathcal{I}}^{r[x \mapsto d]} = t$ per almeno un $d \in \mathcal{D}$.

Definizione 9.7.9 (Semantica della logica dei predicati (formule chiuse)). *Data una interpretazione $\mathcal{I} = (\mathcal{D}, \alpha)$ per un alfabeto $\mathcal{A} = (\mathcal{C}, \mathcal{F}, \mathcal{P}, \mathcal{V})$ e una formula chiusa P , usando la Definizione 9.7.8 il valore di P rispetto ad \mathcal{I} è definito come*

$$\llbracket P \rrbracket_{\mathcal{I}} = \llbracket P \rrbracket_{\mathcal{I}}^{\emptyset}$$

dove $\emptyset : \mathcal{V} \rightarrow \mathcal{D}$ è la relazione vuota.

Vediamo qualche esempio di valutazione della semantica di formule chiuse senza e con quantificatori.

Esempio 9.7.10 (semantica di formule chiuse senza quantificatori). *Consideriamo l'interpretazione standard dei naturali $\mathcal{I}_{\mathbb{N}}$, e vediamo la semantica di alcune formule chiuse senza quantificatori (e quindi senza variabili).*

- Mostriamo che $\llbracket 0 \leq 1 \rrbracket_{\mathcal{I}_{\mathbb{N}}} = t$:

$$\begin{aligned} \llbracket 0 \leq 1 \rrbracket_{\mathcal{I}_{\mathbb{N}}} &= \llbracket 0 \leq 1 \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{\emptyset} && (\text{Def. 9.7.9}) \\ &= (\llbracket 0 \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{\emptyset}, \llbracket 1 \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{\emptyset}) \in \alpha_{\mathcal{P}}^{\mathcal{I}_{\mathbb{N}}}(\leq) && (\text{Def. 9.7.8, clausola 3}) \\ &= (0, 1) \in \alpha_{\mathcal{P}}^{\mathcal{I}_{\mathbb{N}}}(\leq) && (\text{Def. 9.7.6, clausola 2}) \\ &= 0 \leq 1 && (\text{Def. 9.7.4, punto 2}) \\ &= t \end{aligned}$$

- Mostriamo che $\llbracket \text{pari}(2) \Rightarrow \text{primo}(2) \rrbracket_{\mathcal{I}_{\mathbb{N}}} = t$:

$$\begin{aligned} \llbracket \text{pari}(2) \Rightarrow \text{primo}(2) \rrbracket_{\mathcal{I}_{\mathbb{N}}} &= \llbracket \text{pari}(2) \Rightarrow \text{primo}(2) \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{\emptyset} && (\text{Def. 9.7.9}) \\ &= \llbracket \text{pari}(2) \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{\emptyset} \Rightarrow \llbracket \text{primo}(2) \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{\emptyset} && (\text{Def. 9.7.8, clausola 6}) \\ &= (\llbracket 2 \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{\emptyset} \in \alpha_{\mathcal{P}}^{\mathcal{I}_{\mathbb{N}}}(\text{pari})) \Rightarrow (\llbracket 2 \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{\emptyset} \in \alpha_{\mathcal{P}}^{\mathcal{I}_{\mathbb{N}}}(\text{primo})) && (\text{Def. 9.7.8, clausola 3}) \\ &= (2 \in \alpha_{\mathcal{P}}^{\mathcal{I}_{\mathbb{N}}}(\text{pari})) \Rightarrow (2 \in \alpha_{\mathcal{P}}^{\mathcal{I}_{\mathbb{N}}}(\text{primo})) && (\text{Def. 9.7.6, clausola 2}) \\ &= t \Rightarrow t && (\text{Def. 9.1.6}) \\ &= t \end{aligned}$$

Esempio 9.7.11 (semantica di formule chiuse con quantificatori). *Consideriamo l'alfabeto $\mathcal{A}_{\mathcal{T}}$ dell'Esempio 9.5.4 e l'interpretazione $\mathcal{I}_{\mathcal{T}} = (\mathcal{D}_{\mathcal{T}}, \alpha^{\mathcal{T}})$ della Definizione 9.7.4. Valutiamo la semantica della formula*

$$\Phi = (\forall z . A(f(z))) \vee (\forall y . (\exists x . B(x, y) \wedge A(x)))$$

In questo caso particolare, poiché il dominio $\mathcal{D}_{\mathcal{T}} = \{a, b, c\}$ è finito, possiamo gestire i quantificatori molto facilmente, valutando il quantificatore universale come una congiunzione e quello esistenziale come una disgiunzione. Mostriamo che la formula è falsa nell'interpretazione data.

Per prima cosa valutiamo $\llbracket A(f(z)) \rrbracket_{\mathcal{I}_T}^{[z \mapsto b]}$. Abbiamo:

$$\begin{aligned}
 & \llbracket A(f(z)) \rrbracket_{\mathcal{I}_T}^{[z \mapsto b]} \\
 &= \llbracket f(z) \rrbracket_{\mathcal{I}_T}^{[z \mapsto b]} \in \alpha_P^T(A) \quad (\text{Def. 9.7.8, clausola 3}) \\
 &= \alpha_F^T(f)(\llbracket z \rrbracket_{\mathcal{I}_T}^{[z \mapsto b]}) \in \alpha_P^T(A) \quad (\text{Def. 9.7.6, clausola 3}) \\
 &= \alpha_F^T(f)(b) \in \alpha_P^T(A) \quad (\text{Def. 9.7.6, clausola 1}) \\
 &= c \in \{a, b\} = f \quad (\text{Def. 9.7.4, punto 1})
 \end{aligned} \tag{9.12}$$

Valutiamo ora la formula Φ :

$$\begin{aligned}
 & \llbracket (\forall z . A(f(z))) \vee (\forall y . (\exists x . B(x, y) \wedge A(x))) \rrbracket_{\mathcal{I}_T} \\
 &= \llbracket (\forall z . A(f(z))) \vee (\forall y . (\exists x . B(x, y) \wedge A(x))) \rrbracket_{\mathcal{I}_T}^\emptyset \quad (\text{Def. 9.7.9}) \\
 &= \llbracket (\forall z . A(f(z))) \rrbracket_{\mathcal{I}_T}^\emptyset \vee \llbracket (\forall y . (\exists x . B(x, y) \wedge A(x))) \rrbracket_{\mathcal{I}_T}^\emptyset \quad (\text{Def. 9.7.8, clausola 6}) \\
 &= \left(\llbracket A(f(z)) \rrbracket_{\mathcal{I}_T}^{[z \mapsto a]} \wedge \underline{\llbracket A(f(z)) \rrbracket_{\mathcal{I}_T}^{[z \mapsto b]}} \wedge \llbracket A(f(z)) \rrbracket_{\mathcal{I}_T}^{[z \mapsto c]} \right) \quad (\text{Def. 9.7.8, cl. 7, dom. finito}) \\
 &\quad \vee \llbracket (\forall y . (\exists x . B(x, y) \wedge A(x))) \rrbracket_{\mathcal{I}_T}^\emptyset \quad (\text{Formula sottolineata } f \text{ per (9.12)}) \\
 &= \llbracket (\forall y . (\exists x . B(x, y) \wedge A(x))) \rrbracket_{\mathcal{I}_T}^\emptyset \quad ((\text{assorbimento}), (\text{unità})) \\
 &= \llbracket (\exists x . B(x, y) \wedge A(x)) \rrbracket_{\mathcal{I}_T}^{[y \mapsto a]} \wedge \underline{\llbracket (\exists x . B(x, y) \wedge A(x)) \rrbracket_{\mathcal{I}_T}^{[y \mapsto b]}} \quad (\text{Def. 9.7.8, cl. 7, dom. finito}) \\
 &\quad \wedge \llbracket (\exists x . B(x, y) \wedge A(x)) \rrbracket_{\mathcal{I}_T}^{[y \mapsto c]}
 \end{aligned}$$

Concludiamo mostrando che l'ultima formula sottolineata è anche essa f nell'interpretazione data, e quindi lo è anche la formula iniziale Φ .

$$\begin{aligned}
 & \llbracket (\exists x . B(x, y) \wedge A(x)) \rrbracket_{\mathcal{I}_T}^{[y \mapsto b]} = \quad (\text{Def. 9.7.8, cl. 8, dominio finito}) \\
 &= \llbracket B(x, y) \wedge A(x) \rrbracket_{\mathcal{I}_T}^{[y \mapsto b][x \mapsto a]} \vee \llbracket B(x, y) \wedge A(x) \rrbracket_{\mathcal{I}_T}^{[y \mapsto b][x \mapsto b]} \vee \llbracket B(x, y) \wedge A(x) \rrbracket_{\mathcal{I}_T}^{[y \mapsto b][x \mapsto c]} \\
 &= ((a, b) \in \alpha_P^T(B) \wedge a \in \alpha_P^T(A)) \vee ((b, b) \in \alpha_P^T(B) \wedge b \in \alpha_P^T(A)) \vee ((c, b) \in \alpha_P^T(B) \wedge \\
 &\quad \wedge c \in \alpha_P^T(A)) \quad (\text{Def. 9.7.4: } \alpha_P^T(A) = \{a, b\}, \alpha_P^T(B) = \{(a, a), (c, a), (c, b), (b, c)\}) \\
 &= (f \wedge t) \vee (f \wedge t) \vee (t \wedge f) = f
 \end{aligned}$$

Consideriamo ora l'interpretazione dei naturali, e valutiamo la semantica di alcune formule chiuse con quantificatori.

$$\begin{aligned}
 1. \quad & \llbracket (\forall x . (0 \leq 1)) \rrbracket_{\mathbb{N}} \\
 &= \llbracket (\forall x . (0 \leq 1)) \rrbracket_{\mathbb{N}}^\emptyset \quad (\text{Def. 9.7.9}) \\
 &= \llbracket 0 \leq 1 \rrbracket_{\mathbb{N}}^{[x \mapsto d]}, \text{ per ogni } d \in \mathbb{N} \quad (\text{Def. 9.7.8, cl. 7}) \\
 &= (\llbracket 0 \rrbracket_{\mathbb{N}}^{[x \mapsto d]}, \llbracket 1 \rrbracket_{\mathbb{N}}^{[x \mapsto d]}) \in \alpha_{\mathbb{P}}^{\mathbb{N}}(\leq), \text{ per ogni } d \in \mathbb{N} \quad (\text{Def. 9.7.8, cl. 3}) \\
 &= (0, 1) \in \alpha_{\mathbb{P}}^{\mathbb{N}}(\leq), \text{ per ogni } d \in \mathbb{N} \quad (\text{Def. 9.7.6, cl. 2}) \\
 &= t \quad \text{Non dipende da } d \text{ perché } x \text{ non occorre nella formula}
 \end{aligned}$$

$$\begin{aligned}
 2. \quad & \llbracket (\forall x . (x \leq 1)) \rrbracket_{\mathbb{N}} \\
 &= \llbracket (\forall x . (x \leq 1)) \rrbracket_{\mathbb{N}}^\emptyset \quad (\text{Def. 9.7.9}) \\
 &= \llbracket x \leq 1 \rrbracket_{\mathbb{N}}^{[x \mapsto d]}, \text{ per ogni } d \in \mathbb{N} \quad (\text{Def. 9.7.8, cl. 7}) \\
 &= (\llbracket x \rrbracket_{\mathbb{N}}^{[x \mapsto d]}, \llbracket 1 \rrbracket_{\mathbb{N}}^{[x \mapsto d]}) \in \alpha_{\mathbb{P}}^{\mathbb{N}}(\leq), \text{ per ogni } d \in \mathbb{N} \quad (\text{Def. 9.7.8, cl. 3}) \\
 &= (d, 1) \in \alpha_{\mathbb{P}}^{\mathbb{N}}(\leq), \text{ per ogni } d \in \mathbb{N} \quad (\text{Def. 9.7.6, cl. 1 e 2}) \\
 &= d \leq 1 \text{ per ogni } d \in \mathbb{N} \quad (\text{Def. 9.7.4, punto 2})
 \end{aligned}$$

Quindi la formula è falsa, scegliendo per esempio $d = 2$.

I prossimi esempi li presentiamo in modo più informale, ma comunque rigoroso.

3. $\llbracket (\exists x . (x \leq 1)) \rrbracket_{\mathcal{I}_{\mathbb{N}}} :$ per la clausola 8 della Def. 9.7.8, è vera se e solo se $\llbracket (x \leq 1) \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{[x \mapsto d]} = t$ per almeno un $d \in \mathbb{N}$. Quindi è vera, perché scegliendo per esempio $d = 0$ abbiamo $\llbracket (x \leq 1) \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{[x \mapsto 0]} = (\llbracket x \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{[x \mapsto 0]} \leq 1) = (0 \leq 1) = t$.

4. $\llbracket (\forall y . (\exists y . x \leq y)) \rrbracket_{\mathcal{I}_{\mathbb{N}}} :$ per la clausola 7 della Def. 9.7.8, è vera se e solo se $\llbracket (\exists y . x \leq y) \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{[x \mapsto d]} = t$ per ogni $d \in \mathbb{N}$. A sua volta, $\llbracket (\exists y . x \leq y) \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{[x \mapsto d]}$ è vera se e solo se (clausola 8) $\llbracket (x \leq y) \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{[x \mapsto d][y \mapsto d']} = t$ per almeno un $d' \in \mathbb{N}$. Qualunque sia il valore d , possiamo sicuramente trovare un d' tale che $d \leq d'$, per esempio $d' := d + 1$. Quindi la formula è vera.

5. $\llbracket (\exists y . (\forall x . x \leq y)) \rrbracket_{\mathcal{I}_{\mathbb{N}}} :$ per la clausola 8 della Def. 9.7.8, è vera se e solo se $\llbracket (\forall x . x \leq y) \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{[y \mapsto d]} = t$ per almeno un elemento $d \in \mathbb{N}$. A sua volta, $\llbracket (\forall x . x \leq y) \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{[y \mapsto d]}$ è vera se e solo se (clausola 7) $\llbracket (x \leq y) \rrbracket_{\mathcal{I}_{\mathbb{N}}}^{[y \mapsto d][x \mapsto d']} = t$ per ogni $d' \in \mathbb{N}$. Ora, supponendo di aver fissato un valore per d , non è possibile che per ogni $d' \in \mathbb{N}$ valga $d' \leq d$ (basta prendere $d' := d + 1$). Quindi la formula è falsa. Gli esempi 4 e 5 mostrano che in generale scambiando le posizioni di un quantificatore esistenziale e di uno universale il significato di una formula può cambiare, come avevamo già visto nel sesto esempio della Sezione 9.6.

Per concludere questa sezione, ricordiamo che tutti i concetti introdotti per il calcolo proposizionale nella Definizione 9.1.10 si applicano anche alla logica dei predicati, considerando come interpretazioni quelle introdotte nella Definizione 9.7.3, e le formule predicative chiuse invece di quelle proposizionali. Riportiamo qui le definizioni con gli aggiustamenti necessari.

Definizione 9.7.12 (modello, equivalenza, conseguenza logica). *Data una formula predicativa chiusa P e una interpretazione \mathcal{I} , diciamo che \mathcal{I} è un MODELLO di P se P è vera in \mathcal{I} , cioè se $\llbracket P \rrbracket_{\mathcal{I}} = t$, e in questo caso scriviamo:*

$$\mathcal{I} \models P \quad (\mathcal{I} \text{ è modello di } P)$$

Se invece $\llbracket P \rrbracket_{\mathcal{I}} = f$ scriviamo $\mathcal{I} \not\models P$. La notazione si estende nel modo ovvio a un insieme di formule Γ (letto “Gamma”): scriviamo $\mathcal{I} \models \Gamma$ se $\mathcal{I} \models P$ per ogni $P \in \Gamma$, mentre scriviamo $\mathcal{I} \not\models \Gamma$ se c’è almeno una formula $P \in \Gamma$ tale che $\mathcal{I} \not\models P$.

Due formule chiuse P e Q sono LOGICAMENTE EQUIVALENTI se hanno gli stessi modelli, cioè assumono lo stesso valore di verità per qualunque interpretazione. In questo caso scriviamo:

$$P \equiv Q \quad (P \text{ e } Q \text{ sono logicamente equivalenti})$$

Data una formula chiusa P e un insieme di formule Γ , diciamo che P è una CONSEGUENZA LOGICA di Γ se P è vera in ogni interpretazione che rende vere tutte le formule di Γ , oppure, equivalentemente, se ogni modello di Γ è anche un modello di P . In questo caso scriviamo:

$$\Gamma \models P \quad (P \text{ è conseguenza logica di } \Gamma)$$

9.8 Dimostrazioni per sostituzione di formule valide

La Definizione 9.7.9 stabilisce come, fissata un’interpretazione \mathcal{I} per un alfabeto del primo ordine \mathcal{A} , si può assegnare un valore di verità a ogni formula predicativa chiusa. Questo ci permette di introdurre per la logica dei predicati una classificazione delle formule simile a quanto visto con la Definizione 9.2.5 per il calcolo proposizionale.

Definizione 9.8.1 (formule valide, soddisfacibili e insoddisfacibili). *Una FORMULA VALIDA è una formula predicativa chiusa che è sempre vera, per qualunque interpretazione. Se P è valida, scriveremo anche $\models P$. Una FORMULA INSODDISFACIBILE è una formula predicativa chiusa che è sempre falsa, per qualunque interpretazione. Una formula predicativa chiusa è SODDISFACIBILE se esiste almeno una interpretazione per la quale è vera.*

Esercizio 9.8.2. *Si dimostri che ogni tautologia P , vista come formula della logica dei predicati, è valida. Si dimostri che, analogamente, ogni contraddizione P è insoddisfacibile.*

Rispetto al calcolo proposizionale il problema principale è che le interpretazioni possibili da considerare generalmente sono infinite. Inoltre, anche se fissiamo un'interpretazione, il dominio dei valori da considerare potrebbe essere infinito. Questo rende il problema di dimostrare l'equivalenza o la conseguenza logica estremamente più complicati. Infatti non esiste per la logica dei predicati una tecnica analoga a quella delle tavole di verità.

Fortunatamente, il principio di sostituzione e tutte le leggi relative ai connettivi logici che abbiamo visto per il calcolo proposizionale continuano a valere. Più precisamente, il sistema di dimostrazioni introdotto nella Definizione 9.3.12 è corretto anche per la logica dei predicati. Quindi possiamo dimostrare la validità di certe formule con delle dimostrazioni per sostituzione, sfruttando le leggi o altre formule valide già note. Alle leggi già presentate per il calcolo proposizionale aggiungiamone alcune che riguardano le formule quantificate.

Teorema 9.8.3 (alcune leggi sui quantificatori). *Le seguenti doppie implicazioni su formule quantificate chiuse sono valide:*

commutatività	$(\exists x . (\exists y . P)) \Leftrightarrow (\exists y . (\exists x . P))$	$(\forall x . (\forall y . P)) \Leftrightarrow (\forall y . (\forall x . P))$
distributività	$(\exists x . (P \vee Q)) \Leftrightarrow ((\exists x . P) \vee (\exists x . Q))$	$(\forall x . (P \wedge Q)) \Leftrightarrow ((\forall x . P) \wedge (\forall x . Q))$
De Morgan	$\neg(\exists x . P) \Leftrightarrow (\forall x . \neg P)$	$\neg(\forall x . P) \Leftrightarrow (\exists x . \neg P)$

Tabella 9.4: Leggi per quantificatori

La dimostrazione di queste leggi va al di là degli obiettivi di questa dispensa, ma usando le dimostrazioni per sostituzione possiamo mostrare che in ogni coppia di leggi sulla stessa riga della tavola l'una è conseguenza logica dall'altra.

Esempio 9.8.4 (dimostrazioni per sostituzione: $\neg(\forall x . P) \Leftrightarrow (\exists x . \neg P)$). Vediamo per esempio come la seconda legge di De Morgan può essere dimostrata usando la prima:

$$\begin{aligned}
 & \neg(\forall x . \underline{P}) \\
 \Leftrightarrow & (doppia \ negazione) \\
 & \neg(\forall x . \neg(\neg P)) \\
 \Leftrightarrow & (prima \ legge \ (De \ Morgan), \ applicata \ da \ destra \ verso \ sinistra \ sostituendo \ P \ con \ \neg P) \\
 & \neg(\neg(\exists x . \neg P)) \\
 \Leftrightarrow & (doppia \ negazione) \\
 & (\exists x . \neg P)
 \end{aligned}$$

Esempio 9.8.5 (dimostrazioni per sostituzione: $(\forall x . (P \wedge Q)) \Leftrightarrow ((\forall x . P) \wedge (\forall x . Q))$). In modo analogo al caso precedente, possiamo dimostrare la seconda legge di distributività usando la prima:

$$\begin{aligned}
 & (\forall x . (P \wedge Q)) \\
 \Leftrightarrow & (doppia \ negazione) \\
 & \neg(\neg(\forall x . (P \wedge Q))) \\
 \Leftrightarrow & (De \ Morgan) \ per \ \forall \\
 & \neg(\exists x . \underline{\neg(P \wedge Q)}) \\
 \Leftrightarrow & (De \ Morgan) \ per \ \wedge \\
 & \neg(\exists x . (\neg P \vee \neg Q)) \\
 \Leftrightarrow & (distributività) \ per \ \exists \\
 & \neg(\underline{(\exists x . \neg P)} \vee \underline{(\exists x . \neg Q)}) \\
 \Leftrightarrow & (De \ Morgan) \ per \ \forall, \ due \ volte \\
 & \neg(\neg(\forall x . P) \vee \neg(\forall x . Q)) \\
 \Leftrightarrow & (De \ Morgan) \\
 & \neg(\neg((\forall x . P) \wedge (\forall x . Q))) \\
 \Leftrightarrow & (doppia \ negazione) \\
 & ((\forall x . P) \wedge (\forall x . Q))
 \end{aligned}$$

Relativamente alle leggi di distributività per i quantificatori, è bene riflettere sul fatto che il quantificatore esistenziale distribuisce *solo* rispetto alla disgiunzione, mentre quello universale *solo* rispetto alla congiunzione. Infatti, per esempio, la formula $(\forall x . (P \vee Q)) \Leftrightarrow ((\forall x . P) \vee (\forall x . Q))$ non è valida, anche se è soddisfacibile.

Esempio 9.8.6 (controesempio per $(\forall x . (P \vee Q)) \Leftrightarrow ((\forall x . P) \vee (\forall x . Q))$). Come controesempio consideriamo l'alfabeto dei naturali con la corrispondente interpretazione $\mathcal{I}_{\mathbb{N}}$, e le due formule

$$(\forall x . (pari(x) \vee dispari(x))) \quad ((\forall x . pari(x)) \vee (\forall x . dispari(x)))$$

Chiaramente $\mathcal{I}_{\mathbb{N}} \models (\forall x . (pari(x) \vee dispari(x)))$, perché è vero che ogni numero naturale o è pari o è dispari. Però abbiamo $\mathcal{I}_{\mathbb{N}} \not\models (\forall x . pari(x))$, perché non è vero che tutti i naturali sono pari, e analogamente $\mathcal{I}_{\mathbb{N}} \not\models (\forall x . dispari(x))$. Quindi abbiamo $\mathcal{I}_{\mathbb{N}} \not\models ((\forall x . pari(x)) \vee (\forall x . dispari(x)))$.

Analogamente, la formula $(\exists x . (P \wedge Q)) \Leftrightarrow ((\exists x . P) \wedge (\exists x . Q))$ non è valida, cioè la quantificazione esistenziale non distribuisce in generale rispetto alla congiunzione. Il lettore è invitato a trovare un controesempio a questa equivalenza, magari sfruttando quello visto precedentemente.

Concludiamo quest'analisi delle leggi per i quantificatori osservando che la legge di commutatività vale solo per coppie di quantificatori dello stesso tipo. Infatti la formula $(\exists y . (\forall x . P)) \Leftrightarrow (\forall x . (\exists y . P))$ non è valida, anche se è soddisfacibile.

Intuitivamente, la formula $(\exists y . (\forall x . P))$ asserisce che esiste (almeno) un particolare valore di y che rende valida P indipendentemente dall' x considerato. La formula $(\forall x . (\exists y . P))$ invece asserisce che per ciascun valore x possiamo trovare un valore y che rende vera P , ma per valori di x diversi potrebbero servire valori di y diversi.

Esempio 9.8.7 (controesempio per $(\exists y . (\forall x . P)) \Leftrightarrow (\forall x . (\exists y . P))$). Un esempio che rende falsa questa formula si ottiene facilmente dalle formule dei punti 4 e 5 dell'Esempio 9.7.11. Infatti abbiamo visto che $\mathcal{I}_{\mathbb{N}} \models (\forall x . (\exists y . x \leq y))$ perché la formula asserisce che per ogni naturale possiamo trovarne uno maggiore, cosa che è ovviamente vera. Ma $\mathcal{I}_{\mathbb{N}} \not\models (\exists y . (\forall x . x \leq y))$: infatti questa formula asserisce che esiste un naturale più grande di tutti gli altri, cosa che è chiaramente falsa. Quindi possiamo concludere che

$$\mathcal{I}_{\mathbb{N}} \not\models (\exists y . (\forall x . x \leq y)) \Leftrightarrow (\forall x . (\exists y . x \leq y))$$

e quindi la formula non è valida.

Formalizzazione di enunciati sulla teoria degli insiemi

Nel Capitolo 1 abbiamo introdotto le principali relazioni e operazioni tra insiemi in modo preciso ma discorsivo, usando il linguaggio naturale. Un utile esercizio di formalizzazione consiste nel riformulare quelle definizioni usando formule predicative. La formalizzazione risulta particolarmente utile quando tali definizioni devono essere manipolate in qualche modo, come vedremo nell'Esempio 9.8.10. Nel resto di questa sezione useremo l'alfabeto per gli insiemi \mathcal{A}_S introdotto nell'Esempio 9.5.4.

Esempio 9.8.8 (formalizzazione di relazioni tra insiemi). Con formule predicative basate sull'alfabeto degli insiemi \mathcal{A}_S possiamo definire in modo rigoroso alcune delle relazioni tra insiemi introdotte nelle Definizioni 1.2.1 e 1.2.5. Siano A e B due generici insiemi:

Inclusione: “ A è un **sottoinsieme** di B , scritto $A \subseteq B$, se e solo se ogni elemento di A è anche elemento di B ”

$$(A \subseteq B \Leftrightarrow (\forall x . (x \in A \Rightarrow x \in B))) \tag{9.13}$$

Disuguaglianza: “ A e B sono **diversi**, scritto $A \neq B$, se e solo se esiste almeno un elemento che è contenuto in uno dei due insiemi ma non nell'altro”

$$(A \neq B \Leftrightarrow (\exists x . ((x \in A \wedge x \notin B) \vee (x \notin A \wedge x \in B)))) \tag{9.14}$$

Inclusione stretta: “ A è un **sottoinsieme stretto** di B , scritto $A \subset B$, se e solo se $A \subseteq B$ e $A \neq B$ ”

$$(A \subset B \Leftrightarrow (A \subseteq B \wedge A \neq B)) \tag{9.15}$$

Naturalmente possiamo anche formalizzare le classiche operazioni su insiemi.

Esempio 9.8.9 (formalizzazione di operazioni su insiemi). *Definiamo con formule predicative basate sull'alfabeto degli insiemi le operazioni su insiemi, introdotte nella Sezione 1.3, e in particolare la relazione di appartenenza (\in) per insiemi ottenuti come risultato di queste operazioni. Siano A e B due generici insiemi:*

<i>Intersezione:</i>	$(x \in A \cap B \Leftrightarrow (x \in A \wedge x \in B))$
<i>Unione:</i>	$(x \in A \cup B \Leftrightarrow (x \in A \vee x \in B))$
<i>Complemento (rispetto a \mathcal{U})</i>	$(x \in \bar{A} \Leftrightarrow (x \in \mathcal{U} \wedge x \notin A))$
<i>Differenza:</i>	$(x \in A \setminus B \Leftrightarrow (x \in A \wedge x \notin B))$

Esempio 9.8.10 (dimostrazioni su insiemi). *Nell'Esempio 9.8.8 abbiamo formalizzato con la seguente formula la relazione di inclusione stretta tra due insiemi:*

$$(A \subset B \Leftrightarrow (A \subseteq B \wedge A \neq B))$$

D'altra parte dopo la Definizione 1.2.5 avevamo affermato che “per dimostrare che $A \subset B$ si può mostrare che ogni elemento di A appartiene a B , ma che esiste un elemento di B che non appartiene ad A ”. Formalizzando questa frase in logica dei predicati otteniamo la seguente formula:

$$(((\forall x . (x \in A \Rightarrow x \in B) \wedge (\exists x . (x \notin A \wedge x \in B))) \Rightarrow A \subset B)$$

Vediamo che questo è vero, mostrando che premessa e conseguenza sono anzi logicamente equivalenti:

$$\begin{aligned} & A \subset B \\ \Leftrightarrow & \text{(formula (9.15) dell'Esempio 9.8.8)} \\ & (A \subseteq B \wedge A \neq B) \\ \Leftrightarrow & \text{(formula (9.13) dell'Esempio 9.8.8)} \\ & ((\forall x . (x \in A \Rightarrow x \in B)) \wedge A \neq B) \\ \Leftrightarrow & \text{(formula (9.14) dell'Esempio 9.8.8)} \\ & ((\forall x . (x \in A \Rightarrow x \in B)) \wedge (\exists x . ((x \in A \wedge x \notin B) \vee (x \notin A \wedge x \in B)))) \\ \Leftrightarrow & \text{(distributività di } \exists \text{ su } \vee) \\ & ((\forall x . (x \in A \Rightarrow x \in B)) \wedge ((\exists x . (x \in A \wedge x \notin B)) \vee (\exists x . (x \notin A \wedge x \in B)))) \\ \Leftrightarrow & \text{(distributività di } \wedge \text{ su } \vee) \\ & (((\forall x . (x \in A \Rightarrow x \in B)) \wedge (\exists x . (x \in A \wedge x \notin B))) \vee \\ & \quad ((\forall x . (x \in A \Rightarrow x \in B)) \wedge (\exists x . (x \notin A \wedge x \in B)))) \\ \Leftrightarrow & \text{(doppia negazione) e (De Morgan)} \\ & (((\forall x . (x \in A \Rightarrow x \in B)) \wedge \neg(\exists x . (x \in A \wedge x \notin B))) \vee \\ & \quad ((\forall x . (x \in A \Rightarrow x \in B)) \wedge \neg(\exists x . (x \notin A \wedge x \in B)))) \\ \Leftrightarrow & \text{(De Morgan) e (eliminazione implicazione) al contrario} \\ & (((\forall x . (x \in A \Rightarrow x \in B)) \wedge \neg(\forall x . (x \in A \Rightarrow x \in B))) \vee \\ & \quad ((\forall x . (x \in A \Rightarrow x \in B)) \wedge \neg(\exists x . (x \notin A \wedge x \in B)))) \\ \Leftrightarrow & \text{(contraddizione)} \\ & (\mathbf{F} \vee ((\forall x . (x \in A \Rightarrow x \in B)) \wedge (\exists x . (x \notin A \wedge x \in B)))) \\ \Leftrightarrow & \text{(unità)} \\ & ((\forall x . (x \in A \Rightarrow x \in B)) \wedge (\exists x . (x \notin A \wedge x \in B))) \end{aligned}$$

Cerchiamo ora di esplicitare come si può dimostrare che $A \not\subset B$. Abbiamo:

$$\begin{aligned} & A \not\subset B \\ \Leftrightarrow & \text{(per la dimostrazione appena fatta)} \\ & \neg((\forall x . (x \in A \Rightarrow x \in B)) \wedge (\exists x . (x \notin A \wedge x \in B))) \\ \Leftrightarrow & \text{(De Morgan) per } \wedge \\ & (\neg(\forall x . (x \in A \Rightarrow x \in B)) \vee \neg(\exists x . (x \notin A \wedge x \in B))) \\ \Leftrightarrow & \text{(De Morgan), due volte} \end{aligned}$$

$$\begin{aligned}
 & ((\exists x . \underline{\neg(x \in A \Rightarrow x \in B)}) \vee (\forall x . \underline{\neg(x \notin A \wedge x \in B)})) \\
 \Leftrightarrow & ((\text{eliminazione implicazione}) \text{ e } (\text{De Morgan})) \\
 & ((\exists x . \underline{\neg(\neg(x \in A) \vee x \in B)}) \vee (\forall x . \underline{(x \in A \vee \neg(x \in B))})) \\
 \Leftrightarrow & ((\text{De Morgan}), (\text{doppia negazione}) \text{ e } (\text{eliminazione implicazione}) \text{ al contrario}) \\
 & ((\exists x . (x \in A \wedge x \notin B)) \vee (\forall x . (x \in B \Rightarrow x \in A)))
 \end{aligned}$$

Rileggendo questa formula in italiano possiamo concludere che “per dimostrare che $A \not\subset B$ si può mostrare che esiste un elemento di A che non appartiene a B , oppure che ogni elemento di B appartiene ad A ”.

Indice analitico

- alberi binari
 - definizione induittiva di, 148
 - etichettati, 151
- albero, 107
 - completo, 114
 - di derivazione sintattica, 186
 - pieno, 114
 - radicato, 109
- alfabeto, 170
- altezza
 - di un albero, 114
 - di un nodo, 114
- automa, 175
 - deterministico, 179
- biiezione, 48
- Binet
 - formula di, 69
- calcolo proposizionale
 - semantica, 203
 - sintassi, 201
- cardinalità, 21
- chiusura
 - riflessiva, 75
 - riflessiva e transitiva, 80
 - simmetrica, 76
 - transitiva, 78
- ciclo, 98
 - hamiltoniano, 112
- circuito, 98
 - euleriano, 110
- classe di equivalenza, 84
- combinazione, 129
- componente
 - fortemente connessa, 99
- componente connessa, 107
- concatenazione
 - di linguaggi, 172
 - di stringhe, 171
- connettivi logici, 202
- conseguenza logica, 204
- contraddizione, 208
- costruzione dei sottoinsiemi, 181
- cricca, 136
- DAG, 100
- definizione induittiva
 - dei numeri naturali, 58
 - di alberi binari, 148
 - di alberi binari etichettati, 151
 - di altezza di albero, 149
 - di appartenenza a liste, 143
 - di concatenazione di liste, 144
 - di dimensione di albero, 149
 - di fattoriale, 62
 - di inversione di liste, 144
 - di liste, 142
 - di lunghezza di liste, 142
 - di numeri triangolari, 59
 - di somma di \mathcal{N} -termini, 159
 - di somma su albero, 152
 - di somma su lista, 143
 - di stringhe, 170
 - di valuazione di \mathcal{N} -termini, 159
 - di visita di albero, 152
- definizione ricorsiva
 - di distanza su grafo, 114
- diagrammi di Eulero-Venn, 5
- dimostrazione
 - in proof system, 215
- dimostrazioni
 - discursive, 14
 - per sostituzione, 12
- disposizione, 128
- distanza, 113
 - quasi-metrica, 113
 - su grafo, 113
- distanza su grafo
 - definizione ricorsiva di, 114
- equivalenza
 - classe di, 84
 - logica, 204
- espressione regolare, 195
- Eulero-Venn

- diagrammi di, 5
- fattoriale
 - definizione induttiva di, 62
- Fibonacci
 - numeri di, 67
- formula
 - di Binet, 69
 - di Gauss, 59
 - di Nicomaco, 65
 - soddisfacibile, 208
- formule
 - proposizionali, 201
- funzione, 41
 - biettiva, 48
 - iniettiva, 47
 - kernel di, 84
 - parziale, 47
 - surgettiva, 47
- Gauss
 - formula di, 59
- grafi
 - isomorfismo di, 115
- grafo
 - aciclico, 98
 - bipartito, 137
 - cardinale, 109
 - ciclico, 98
 - connesso, 106
 - diametro di, 114
 - etichettato, 95
 - fortemente connesso, 99
 - non orientato, 102
 - ordinale, 109
 - orientato, 91
 - orientato aciclico, 100
 - pesato, 95
- grammatica, 184
 - ambigua, 190
 - corrispondente a automa, 192
- insieme
 - delle parti, 17
 - di arrivo, 25
 - di partenza, 25
- insiemi, 1
 - k -insieme, 21
 - cardinalità di, 21
 - complemento, 7
 - confrontare, 4
 - differenza di, 7
 - disgiunti, 5
 - famiglia di, 17
 - in biezione, 50
 - inclusione di, 5
 - intersezione di, 7
- naturali come, 19
 - per enumerazione, 2
 - per proprietà, 2
 - prodotto cartesiano di, 19
 - uguaglianza di, 4
 - unione di, 6
- interpretazione, 202
- kernel, 84
- Kleene
 - stella di, 80, 173
- leggi
 - su insiemi, 9
 - su relazioni, 33
- linguaggio, 171
 - accettato da automa, 178
 - generato da non terminale, 187
 - regolare, 197
- lista di adiacenza, 94
- liste
 - definizione induttiva di, 142, 152
- matrice di adiacenza, 93
- modello, 204
- naturali
 - come insiemi, 19
 - definizione induttiva di, 58
- Nicomaco
 - formula di, 65
- nodo
 - altezza di, 114
 - grado di ingresso di, 92
 - grado di uscita di, 92
 - profondità di, 114
 - vicinato di, 92
- numeri
 - di Fibonacci, 67
 - triangolari, 59
- operatori
 - booleani, 8
 - ordinamento, 88
 - lessicografico, 88
 - parziale, 87
 - topologico, 101
- parola, 170
- parse tree, 186
- partizione, 18
- path, 96
 - hamiltoniano, 112
- permutazione, 126
- principio di induzione
 - forte, 68
 - sui naturali, 59

- prodotto cartesiano, 19
- produttoria, 64
- profondità di un nodo, 114
- proof system
 - completo, 215
 - corretto, 215
 - per calcolo proposizionale, 216
- proposizione, 199
- proprietà, 44
- quantificatore
 - esistenziale, 29
 - universale, 30
- quantificatori, 29
- relazione, 25
 - anti-simmetrica, 74
 - ben fondata, 165
 - complemento di, 28
 - completa, 25
 - composizione n -aria di, 77
 - di equivalenza, 83
 - di precedenza indotta, 165, 166
 - identità, 26
 - iniettiva, 37
 - insieme di arrivo di, 25
 - insieme di partenza di, 25
 - inversa, 49
 - invertibile, 49
 - opposta, 30
 - ordinamento, 88
 - ordinamento parziale, 87
 - riflessiva, 72
 - simmetrica, 73
- su un insieme, 71
- successore, 27
- surgettiva, 36
- totale, 35
- transitiva, 73
- univalente, 36
- vuota, 25
- relazioni
 - composizione di, 28
 - differenza di, 28
 - intersezione di, 28
 - operazioni insiemistiche su, 28
 - unione di, 28
- segnatura, 155
- sequenza
 - di Fibonacci, 67
- sequenze, 53
 - di lunghezza arbitraria, 54
 - di lunghezza fissa, 53
- sistema di dimostrazioni, 214
- sommatoria, 64
- sottoinsiemi, 5
- stringa, 170
- tautologia, 207
- termini, 155
- trail, 96
 - euleriano, 110
- tringa
 - ambigua, 190
- walk, 96